applied sciences

MDPI

Article

Correlation-Aware Stowage and Storage Assignment in Robotic Mobile Fulfillment Systems: Optimizing Efficiency and Picker Workload Balance

Tianrong Ding [1], Teng Huang [2] and Yuankai Zhang [3,*]

[1] School of Economics and Management, North China Electric Power University, Beijing 102206, China; dingtianrong@ncepu.edu.cn
[2] School of Business, University of Auckland, Auckland 1142, New Zealand; teng.huang@auckland.ac.nz
[3] School of Economics and Management, Beihang University, Beijing 100191, China
* Correspondence: ykzhang6635@126.com

Featured Application

Warehouse operators increasingly deploy robotic mobile fulfillment systems (RMFSs) to improve order fulfillment efficiency. Operational data from an e-commerce retailer show workload differences across picking stations, motivating the evaluation of storage planning alongside operational control. This study examines how managers can use SKU demand and co-purchase information to design SKU-to-rack and rack-to-zone assignments. Using large-scale data from an industry partner, our results show that correlation-aware storage strategies can reduce rack travel distance by up to 23%, increase throughput by 27%, and improve workload balance by over 60% compared with existing benchmarks. The comparisons help managers assess trade-offs between rack travel distance and station workload balance.

Abstract

The rapid adoption of Robotic Mobile Fulfillment Systems (RMFSs) is transforming e-commerce logistics by combining robotic efficiency with human flexibility. However, substantial workload differences between picking stations present an operational challenge in these human–robot collaborative environments. Such imbalance may cause congestion, worker fatigue, and reduced system performance. This paper addresses this efficiency–equity trade-off by evaluating proactive storage planning alongside real-time operational control. We study a joint stowage and storage assignment problem that determines how SKUs should be allocated to racks and where racks should be positioned in the warehouse in order to simultaneously reduce robot travel distances and balance workloads across picking stations. To solve this problem, we develop a data-driven framework that exploits SKU correlation information to cluster frequently co-purchased items and strategically position high-demand racks within the warehouse. Using large-scale operational data from a major e-commerce retailer in China, numerical experiments demonstrate that the combined storage-and-control framework significantly improves system performance. Compared with existing benchmark approaches, the proposed method reduces robot travel distances by 23%, increases system throughput by 27%, and improves workload balance by 62.16%. These results highlight the importance of incorporating workload considerations into storage planning alongside operational control and provide practical guidance for managers seeking to design more efficient and human-centered RMFS operations under the tested operating conditions.

Check for updates

Academic Editor: Yutaka Ishibashi

Received: 17 September 2026
Revised: 14 October 2026
Accepted: 16 October 2026
Published: 20 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: robotic mobile fulfillment system; stowage and storage assignment; SKU co-purchase correlation; scattered storage; workload balance; human–robot collaboration

1. Introduction

The exponential growth of e-commerce has fundamentally transformed warehouse logistics, compelling leading retailers such as Amazon and JD.com to adopt Robotic Mobile Fulfillment Systems (RMFS) at scale [1–4]. As a canonical class of cyber-physical systems, RMFS seamlessly integrates the speed and precision of Autonomous Guided Vehicles (AGVs) with the cognitive flexibility of human pickers. Rather than dispatching workers to retrieve items, these systems deliver mobile storage racks directly to stationary workstations, fundamentally replacing the traditional picker-to-parts paradigm with a parts-to-picker configuration [5–7]. While this architectural shift substantially curtails human travel time [8,9], it simultaneously introduces a more subtle and consequential challenge: effectively coordinating heterogeneous resources in which robotic efficiency imperatives and human ergonomic constraints are inherently in tension.

This tension manifests most acutely in the trade-off between system throughput and workload equity. From a purely robotic standpoint, efficiency is maximized by minimizing rack travel distances. In practice, however, optimizing solely for travel distance tends to concentrate high-velocity racks at specific workstations, creating pronounced workload imbalances among human pickers. Critically, such disparities are not merely a matter of fairness—they directly impair overall system performance by accelerating picker fatigue and inducing persistent bottlenecks at overloaded stations [8,10,11]. Empirical evidence from our industrial partner reveals statistically significant workload disparities between picking stations.

Despite growing awareness of these challenges, existing research has predominantly addressed RMFS performance through operational-level scheduling mechanisms, including order batching [12], robot path planning [13], and conflict-free scheduling [14]. While a growing body of work integrates human factors—such as picker fatigue and physical state variation—into real-time scheduling frameworks [8,15,16], these approaches invariably treat the warehouse storage configuration as a fixed, exogenous constraint. This operational focus suggests a complementary opportunity for system-wide improvement: the tactical design of storage. We examine two interdependent storage decisions—stowage (the allocation of SKUs to racks) and storage assignment (the positioning of racks within storage zones)—as potential contributors to workload distribution. By jointly optimizing these decisions, managers can proactively shape the storage configuration before orders are released and complement subsequent scheduling adjustments.

This study bridges the gap between tactical storage management and human-centric warehouse operations by proposing a data-driven framework for the joint stowage and storage assignment problem. Our approach uses SKU co-purchase correlations to link tactical stowage and rack-to-zone planning with picking station selection and rack returns. Specifically, we introduce a three-stage algorithmic framework: (1) density-based SKU clustering to determine scattering across racks, (2) capacity-constrained associative stowage based on co-purchase correlations, and (3) correlation-adjusted velocity-based zone assignment with workload-aware station selection and rack repositioning.

Using large-scale real-world operational data from a leading Chinese e-commerce retailer, we demonstrate that efficiency and workload equity can be simultaneously improved by the integrated framework in the evaluated cases. The principal contributions of this paper are as follows:

- **Empirical motivation for workload-aware storage planning.** We document statistically significant workload disparities between picking stations and two illustrative within-picker comparisons. These observations motivate evaluating storage planning alongside operational scheduling, without identifying storage configuration as the primary source of workload imbalance.
- **A novel three-stage data-driven optimization framework.** We develop a unified algorithmic framework that jointly addresses the stowage and storage assignment problem through three tightly integrated stages: SKU co-purchase correlation mining, capacity-constrained rack clustering for associative and scattered stowage, and a correlation-adjusted velocity-based zone assignment with flexible zone-sharing repositioning. Crucially, workload equity is not treated as a post hoc adjustment but is explicitly embedded as a hard constraint within the optimization model, enabling simultaneous pursuit of efficiency and fairness objectives.
- **Empirical validation on large-scale real-world data.** Through extensive computational experiments using operational data from a leading e-commerce retailer, we demonstrate that the proposed methodology substantially outperforms state-of-the-art benchmarks [17,18]. Specifically, our approach achieves a 23% reduction in rack travel distance and a 27% increase in system throughput while simultaneously improving picker workload balance by 62.16%. These results provide empirical evidence that the integrated framework can improve operational efficiency and picker workload balance together under the tested conditions.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 presents empirical evidence of workload imbalance to motivate the study. The joint stowage and storage assignment problem is formulated in Section 4, with a focus on modeling the efficiency–equity trade-off. In Section 5, we develop a data-driven three-stage algorithmic framework to solve this problem. Section 6 conducts extensive numerical experiments and sensitivity analyses using real-world industrial data. Section 7 discusses the findings from a management perspective and offers actionable implications for practitioners. Finally, Section 8 summarizes the study and suggests directions for future research.

2. Literature Review

This study optimizes the stowage and storage decisions in RMFS while guaranteeing the workload balance. Thus, the literature review is bifurcated into three domains: rack storage location assignments, scattered stowage, and workload balance in human–robot coordinated systems.

2.1. The Storage Problem

The storage problem decides which location a rack should return to after an order is picked. Figure 1 illustrates a commonly adopted layout in RMFS warehouses [19,20]. The storage area is divided into blocks composed of storage locations, with picking stations evenly spaced along one side of the warehouse. In Figure 1, color coding indicates the distance to the nearest picking station, where redder locations are closer and thus more preferable.

Merschformann et al. [21] summarize five prevalent rack storage location assignment rules: random storage, fixed storage, shortest path storage, station-based storage, and class-based storage. Among these, shortest path and class-based storage strategies demonstrate particularly strong performance in the literature. Shortest path storage focuses on minimizing travel distance by assigning a picked rack to the nearest available storage location. The literature studies this approach extensively because it directly affects operational

performance. Weidinger et al. [22] introduce an adaptive programming approach for rack assignment and compare it against three heuristic policies. Their results indicate that a simple shortest path strategy yields remarkably competitive performance, often rivaling more sophisticated optimization methods. Similarly, Roodbergen et al. [23] establish that shortest path strategies significantly reduce travel time, thereby enhancing order fulfillment speed. However, while effective at reducing travel distance, this strategy does not inherently address workload balancing constraints.

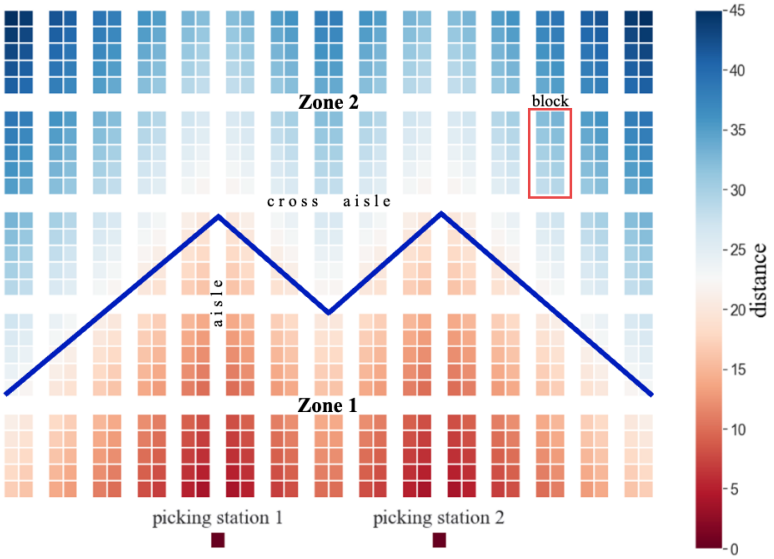


Figure 1. Distances from rack storage locations to the nearest station.

Class-based storage groups racks according to demand frequency, dividing storage locations into zones based on their proximity to picking stations. High-demand racks are prioritized for zones nearest to the stations. This method shows theoretical and empirical superiority in various warehouse contexts, particularly when using velocity-based policies derived from sales data. Yuan et al. [17] propose two velocity-based policies that account for rack movement frequency. Their simulations demonstrate that a system with only two or three classes captures the majority of potential efficiency gains, especially when rack velocity variance is high. This is corroborated by Cezik et al. [24], who find that a two-class policy achieves 75% of the maximum possible travel-time reduction, while a three-class policy increases this to 90%. Furthermore, Lamballais et al. [19] show that maximum throughput in an RMFS can increase by nearly 50% when utilizing turnover-based storage zones. Given the proven computational efficiency and scalability of this approach for large-scale industrial instances, we adopt a class-based policy in this work. We aim to further enhance this method by refining how locations and racks are grouped using problem-specific data correlations.

Despite these advancements, the literature exhibits specific limitations. Existing studies rarely address the decision-making process regarding the optimal number of racks across which a specific SKU should be distributed based on historical data. Furthermore, the simultaneous optimization of stowage decisions and rack storage assignment remains largely unexplored. This paper addresses these gaps by integrating these decision layers while explicitly optimizing for workload balance at picking stations.

2.2. The Stowage Problem

The stowage problem addresses the allocation of SKUs to racks. In the context of orders containing multiple SKUs, literature suggests that both correlated and scattered storage strategies offer distinct advantages [9,25].

A significant body of research emphasizes correlated storage, which leverages product affinity to group frequently ordered items. Kim et al. [26] address the SKU assignment problem in RMFS by maximizing the similarity value of SKUs within each rack. They develop a constructive heuristic to consolidate frequently co-ordered SKUs, validating the approach on instances ranging from 10 to 500 items. Similarly, Mirzaei et al. [27] use historical order data to determine product affinity. They propose an integrated cluster allocation policy that groups correlated products to minimize retrieval time. Their results demonstrate that this affinity-based policy can reduce total retrieval time by up to 40% compared to traditional turnover-based or class-based policies.

A primary limitation of strict clustering algorithms, such as the one proposed by Mirzaei et al. [27], is that they typically assign an SKU to a single cluster and subsequently to a single rack. This restricts each SKU to a unique location, failing to exploit the benefits of scattered storage. Scattered storage, where a single SKU is distributed across multiple racks, further enhances performance. Lamballais et al. [9] demonstrate that system throughput improves substantially when inventory is spread across multiple locations, as it increases the probability of a relevant rack being close to a picking station. Building on this, Ma et al. [18] jointly consider SKU classification and correlation to propose a novel scattered storage policy. They formulate a mixed-integer programming model and solve it using a variable neighborhood search algorithm equipped with self-adaptation mechanisms. Their comparative analysis indicates that this scattered approach outperforms the clustering method of Mirzaei et al. [27], yielding performance improvements in the range of 6.31% to 19.44%. While the above studies primarily focus on stowage policies, little attention is devoted to the role of human workers in increasingly automated warehouse systems.

2.3. Workload Balancing and Human Factors in Collaborative Warehouses

In human–robot collaborative environments, robots assume the burden of high-intensity travel and heavy lifting, theoretically offering pickers a more ergonomic experience. In such systems, picker workloads depend on robot dispatch, task allocation and service conditions. Traditionally, these algorithms prioritize the minimization of robot travel distance. This singular focus can lead to imbalanced workloads among pickers, raising the critical challenge of how to maintain operational efficiency while ensuring an equitable and sustainable human–robot work environment.

Recent literature emphasizes the necessity of integrating human factors into operational planning. Kumar et al. [28] identify collaborative picking that accounts for human fatigue as a burgeoning area of interest. As noted by Grosse et al. [29], effective workload balancing is essential for maintaining pickers in an optimal state, thereby preventing fatigue-induced errors or slowdowns. Wang et al. [8] investigate the impact of picker fatigue, observing that pickers may require varying amounts of time to process identical items on different days due to fluctuating physical states. To address this, they formulate the problem as a stochastic dynamic program that generates robot schedules responsive to picker state fluctuations. Solving this via an approximate dynamic programming based branch-and-price approach, they demonstrate that accounting for human variation can reduce picking times by 10% compared to static models.

Zhuang et al. [16] address workload balance and rack conflicts through joint order and rack sequencing. The present framework links tactical SKU-to-rack stowage and rack-to-zone design with operational station selection and rack-return decisions. Its distinguishing focus is correlation-informed stowage and zone assignment rather than joint optimization or workload balancing alone. Operational scheduling and storage planning are complementary. Stage 3 still includes operational balancing interventions.

3. Empirical Evidence of Workload Imbalance

In this section, we present empirical evidence illustrating the prevalence of workload imbalance, derived from the operational data of our industrial partner. We quantify workload at picking stations by measuring the throughput rate, defined as the number of items picked per hour. This hourly item count describes completed work at a station and is the primary basis for salary calculations at the data-providing company. It does not distinguish picker ability from task supply or other operating conditions.

The dataset originates from two RMFS warehouses operated by a prominent e-commerce retailer in China. The data includes comprehensive transactional details, such as timestamps, order IDs, SKU identifiers, and quantities. Furthermore, the picking logs capture precise operational events, including rack arrival and departure times at workstations, specific items picked, and the rack layers accessed during selection. Table 1 summarizes the data for these comparisons in three cases. The Case Size column counts unique SKU–order combinations for each case. The Sample Size column counts hourly observations per station for the comparisons, with each observation representing the number of items picked per worker at a station during one sampled hour.

Table 1. Results of Shapiro–Wilk normality test.

Case ID	Warehouse ID	Time Span	Case Size *	Sample Size ‡	$p_1^\dagger$	$p_2^\dagger$
1	1	1 October–10 October	52,565	158	0.0559	0.1959
2	2	1 May–10 May	38,038	125	0.3887	0.1520
3	2	1 June–10 June	34,431	132	0.0937	0.7019

†: The p-values of picking stations 1 and 2 are abbreviated as p_1 and p_2, respectively; *: Case size is the number of unique SKU-order combinations; ‡: Sample size counts hourly observations per station and is the same for both stations.

We first conducted the Shapiro–Wilk normality test on the number of hourly picked items per worker for each station. As indicated in the last two columns of Table 1, the resulting p-values exceeded 5%, leading us to retain the null hypothesis that the datasets are normally distributed. This suggests that the number of items picked per hour per worker at different picking stations at all examined time slots conforms to normal distributions.

The two-sample t-tests in Table 2 compare the mean number of items picked per hour per worker between picking stations within the same warehouse and over the same time window. With the prerequisite of a normal distribution met, a two-sample t-test was implemented to evaluate the equality of workloads between two picking stations. The detailed results of this test are tabulated in Table 2. Notably, all p-values obtained are below the 5% threshold. This indicates a substantial disparity in the hourly workload between the two picking stations. These test outcomes describe workload differences across stations in the observed periods. They do not identify storage configuration as the primary cause of those differences.

Table 2. Results of two-sample t-tests.

	Case 1	Case 2	Case 3
p-value	0.0001	0.0006	0.0400
95% CI †	[20.02, 56.84]	[10.21, 36.90]	[−31.07, −0.90]
mean of station 1	242.63	176.08	130.06
mean of station 2	204.20	152.52	146.05

† CI is short for confidence interval.

To describe within-picker workload variation across stations, we present two exploratory comparisons of the same picker at different stations. This involved measuring a

picker's workload over the same day period on two different days. Each day, the picker works at a distinct picking station. We identify workers who work at two different stations in the same warehouse during the same time period of a day but on different days. We managed to identify two such pickers. They are labeled as Picker 1 and Picker 2 in Table 3. On average, Picker 1 processed 243 items per hour from 13:00 to 24:00 on 2 October and 187 items on 3 October. Picker 2 processed 264 items per hour from 7:00 to 18:00 on 3 October and 201 items on 5 October.

For Table 3, an observation is a within-picker pair of hourly workloads at matching clock-hour positions on the two listed days. Both paired comparisons contain 11 complete hour pairs, with no missing or excluded hours: Picker 1 covers 13:00–24:00 on 2 and 3 October, and Picker 2 covers 07:00–18:00 on 3 and 5 October. Each comparison includes 22 single-day hourly observations, which form 11 matched pairs. The paired samples t-test describes workload differences for the two individual pickers working across both stations. As shown in Table 3, the results yield p-values below 0.05, indicating a statistically significant difference in the number of items picked by the same individual depending on the station. Holding picker identity fixed does not hold fatigue, order composition, task allocation or other conditions constant across different days. These within-picker comparisons illustrate hourly workload differences across picking stations and motivate examining workload balance alongside storage planning and operational control. Their observational design does not isolate picker-specific and station-level effects or establish storage configuration as the sole or primary source of workload imbalance.

Table 3. Results of paired samples t-test.

	Day	Time	Station	Picked Items *	p-Value	Diff. in Means	95% CI †
Picker 1	2 October 3 October	13:00–24:00 13:00–24:00	1 2	243 187	0.0225	56.0	[9.73, 102.63]
Picker 2	3 October 5 October	7:00–18:00 7:00–18:00	1 2	264 201	0.0489	75.8	[0.45, 151.15]

† CI is short for confidence interval; *: The number of items picked per hour.

4. Problem Description and Formulation

In this section, we present the mathematical formulation for the stowage and storage assignment problem, explicitly incorporating constraints to ensure workload balance within RMFS.

4.1. Problem Description

We investigate the stowage and storage assignment problems to enhance operational efficiency in RMFS warehouses. The stowage problem addresses the tactical allocation of SKUs to mobile racks, whereas the storage problem determines the locations for returning racks after item retrieval.

First, we formulate the stowage problem. Consider an RMFS warehouse comprising a set of products $\mathcal{I}$, a set of racks $\mathcal{R}$, and a set of rack storage zones $\mathcal{Z}$. The stowage problem addresses the assignment of products $i \in \mathcal{I}$ to racks $r \in \mathcal{R}$. The primary objective is to maximize the co-purchase correlation of SKUs stored on the same rack. The stowage problem encompasses two key decisions: determining the number of racks across which a specific SKU should be distributed (scattered storage) and selecting which SKUs should be co-located on a single rack. The storage capacity of each rack is limited to an upper bound of N SKUs. Conversely, while scattering SKUs across multiple racks enhances picking efficiency, it simultaneously increases replenishment costs. To balance these trade-offs, an individual SKU may be scattered across a maximum of M racks. Because co-purchase correlations are estimated from order data over a relatively long horizon (e.g., a

quarter), stowage decisions operate at the tactical level. In contrast, the subsequent storage assignment problem represents an operational decision that fluctuates with every order-picking cycle. Consequently, we assume that stowage decisions remain static during the short-term order-picking process. This assumption aligns with standard industrial practice, where stowage strategies are optimized based on long-term historical data—with occasional seasonal adjustments—and remain unchanged during daily operations.

Next, we formulate the storage problem. It decides which location the rack will be returned to after picking up the order. There is a strong connection between rack storage assignment and the order-picking process. The order-picking process is complex. It includes order batching, robot scheduling, and picker scheduling. One special feature of the storage decision in RMFS is that the rack can return to a different location after picking [30]. As picking proceeds, the locations of the racks change. When rack A is retrieved for picking SKUs, another rack C might occupy a storage location previously occupied by rack A. In other words, the storage location's availability status changes in real-time. In this study, we follow class-based storage and relocate the rack to a possibly different location to minimize the travel distance of all racks to fulfill a set of online orders. The RMFS is dynamic because orders with random items arrive, and rack locations change ad hoc. This poses a challenge when calculating traveling distances. We treat it as a multi-stage decision problem: cyclic rolling to assign picked racks to storage locations for order batches, and to evaluate the performance of stowage and rack storage assignment in RMFS. At each stage, we minimize the travel distance of visited racks. The total travel distance can be obtained by summing over all stages. Meanwhile, the workload must be balanced between stations. The gap in workload between any two workstations cannot exceed the threshold G.

Based on observations of several RMFS warehouses operated by a large online retailer, the model adopts the following assumptions to focus on stowage and rack storage assignment.

- Product inventory is sufficient to fulfill all incoming orders, so stockouts do not occur in the model [31]. The partner company maintains safety stock to reduce the occurrence of stockouts. This assumption limits the analysis to conditions in which all incoming orders can be fulfilled without stock shortages.
- The order pool is dynamic, and orders follow a First-Come-First-Served (FCFS) policy [32]. FCFS simplifies order processing because order batching and sequencing are outside the primary focus of this study. This assumption fixes the order-processing rule and excludes order-batching and priority decisions, but it may change the absolute travel and throughput values in facilities that batch orders or prioritize urgent orders.

We propose the mathematical model in the following subsection. Table 4 provides a summary of all notations.

Table 4. Notations of the model.

Sets	Definitions
$\mathcal{I}$	The set of SKUs in the assortment
$\mathcal{L}$	The set of storage locations in the warehouse
O^s	The set of customer orders for stowage decisions
O^r	The set of customer orders for rack storage assignment
$\mathcal{P}$	The set of picking stations in the warehouse
$\mathcal{R}$	The set of racks in the system
$\mathcal{Z}$	The set of location zones in the warehouse
Parameters	**Definitions**
$c_{i,o}$	Binary: 1, if the SKU $i \in \mathcal{I}$ is requested in order $o \in \mathcal{O}$, 0 otherwise
$d_{l,p}$	The travel distance of location $l \in \mathcal{L}$ to the picking station $p \in \mathcal{P}$

Table 4. *Cont.*

G	Threshold to control workload balance of workstations
h_r	The velocity of rack r in historical orders
$J_{r,l}$	Binary: 1, if rack $r \in \mathcal{R}$ is parked at location $l \in \mathcal{L}$, 0 otherwise; $\sum_{l \in \mathcal{L}} J_{r,l} \leq 1, \forall r \in \mathcal{R}$
$m_{l,z}$	Binary: 1, if location $l \in \mathcal{L}$ is assigned to zone $z \in \mathcal{Z}$, 0 otherwise; $\sum_{z \in \mathcal{Z}} m_{l,z} = 1, \forall l \in \mathcal{L}$
M	The maximum number of racks for scattered storage of SKUs
N	The maximum number of SKUs that can be stored on a rack
$q_{i,o}$	Quantity of requested SKU $i \in \mathcal{I}$ in order $o \in \mathcal{O}$
$s_{i,j}$	The correlation between SKUs i and j in the training data set
Variables	**Definitions**
$a_{i,r}$	Binary: 1, if SKU $i \in \mathcal{I}$ is assigned to rack $r \in \mathcal{R}$, 0 otherwise
$b_{r,l}$	Binary: 1, if requested rack $r \in \mathcal{R}$ backs to location $l \in \mathcal{L}$ after picking, 0 otherwise
$k_{r,z}$	Binary: 1, if rack $r \in \mathcal{R}$ is assigned to zone $z \in \mathcal{Z}$, 0 otherwise
v_i	The quantity of racks for scattered storage of SKU $i \in \mathcal{I}$
$x_{i,o,r}$	Binary: 1, if rack $r \in \mathcal{R}$ is requested to pick SKU $i \in \mathcal{I}$ in order $o \in \mathcal{O}$, 0 otherwise
$y_{o,r,p}$	Binary: 1, if rack $r \in \mathcal{R}$ travels to the picking station $p \in \mathcal{P}$ to pick order $o \in \mathcal{O}$, 0 otherwise

4.2. Math Models

The stowage problem is a tactical decision based on order data over a relatively long period. Therefore, the dynamic character of the racks will not affect stowage decisions. So, we first model the stowage problem. Decision variables of the stowage model are used as parameters to link the rack storage assignment model.

For a set of orders denoted as $\mathcal{O}^s$, we first calculate the correlation between SKUs pairs i and j, $s_{i,j} = \sum_{o \in \mathcal{O}^s} c_{i,o} \times c_{j,o}, \forall i, j \in \mathcal{I}$. $c_{i,o}$ equals to 1 if $i \in \mathcal{I}$ is requested in order $o \in \mathcal{O}$, and 0 otherwise.

Then, we formulate the stowage problem as $\mathrm{P_S}$. Its object is to maximize SKU association on the same rack, as indicated in (1). Constraints (2) mean that SKU i is scattered on v_i racks. Constraints (3) ensure that the number of SKUs stored on the rack r is at most N. Constraints (4) define the upper bound on the number of racks for an SKU scattered storage. It cannot be more than M racks. The binary decision variable $a_{i,r} \in \{0, 1\}$ indicates whether to put SKU i on rack r. Constraints (5) define the domains of the decision variables.

$$\mathrm{P_S}: f_1 = \max \sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{I}} \sum_{j > i} s_{i,j} a_{i,r} a_{j,r} \tag{1}$$

$$\text{s.t.,} \sum_{r \in \mathcal{R}} a_{i,r} = v_i, \forall i \in \mathcal{I} \tag{2}$$

$$\sum_{i \in \mathcal{I}} a_{i,r} \leq N, \forall r \in \mathcal{R} \tag{3}$$

$$v_i \in [0, M] \cup \mathbb{Z}_+, \forall i \in \mathcal{I} \tag{4}$$

$$a_{i,r} \in \{0, 1\}, \forall i \in \mathcal{I}, r \in \mathcal{R} \tag{5}$$

Next, we formulate the rack storage problem. While fulfilling orders in O^r, available locations to return to after the worker picking off the items change ad hoc. We handle the dynamics of racks' locations by rolling the time horizon. We slice time into slots and assume racks' locations and storage locations' availability are unchanged within each time slot. Denote the length of a single time slot as Δt. The value of Δt is set artificially. We build a model $\mathrm{P_R}$ of the storage assignment in each slot.

$$\mathrm{P_R}: \ f_2 = \min \sum_{o \in \mathcal{O}^r} \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}} y_{o,r,p} (J_{r,l} d_{l,p} + b_{r,l} d_{l,p}) \tag{6}$$

$$\text{s.t.}, c_{i,o} \leq \sum_{r \in \mathcal{R}} x_{i,o,r}, \forall i \in \mathcal{I}, o \in \mathcal{O}^r \tag{7}$$

$$x_{i,o,r} \leq a_{i,r}, \forall i \in \mathcal{I}, o \in \mathcal{O}^r, r \in \mathcal{R} \tag{8}$$

$$y_{o,r,p} \leq \sum_{l \in \mathcal{L}} J_{r,l}, \forall o \in \mathcal{O}^r, r \in \mathcal{R} \tag{9}$$

$$x_{i,o,r} \leq \sum_{p \in \mathcal{P}} \sum_{i' \in \mathcal{I}} y_{o,r,p} c_{i',o}, \forall i \in \mathcal{I}, o \in \mathcal{O}^r, r \in \mathcal{R} \tag{10}$$

$$\sum_{p \in \mathcal{P}} y_{o,r,p} \leq 1, \forall o \in \mathcal{O}^r, r \in \mathcal{R} \tag{11}$$

$$\sum_{l \in \mathcal{L}} b_{r,l} \leq 1, \forall r \in \mathcal{R} \tag{12}$$

$$\sum_{z \in \mathcal{Z}} k_{r,z} = 1, \forall r \in \mathcal{R} \tag{13}$$

$$\sum_{p \in \mathcal{P}} y_{o,r,p} = \sum_{l \in \mathcal{L}} b_{r,l}, \forall o \in \mathcal{O}^r, r \in \mathcal{R} \tag{14}$$

$$b_{r,l}(k_{r,z} - m_{l,z}) = 0, \forall r \in \mathcal{R}, l \in \mathcal{L}, z \in \mathcal{Z} \tag{15}$$

$$\sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}^r} \sum_{r \in \mathcal{R}} x_{i,o,r} y_{o,r,p} \mathfrak{q}_{i,o} \leq \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{I}} \sum_{o \in \mathcal{O}^r} \sum_{r \in \mathcal{R}} x_{i,o,r} \mathfrak{q}_{i,o} \times (1 + G), \quad \forall p \in \mathcal{P} \tag{16}$$

$$b_{r,l}, k_{r,z} \in \{0, 1\}, \forall r \in \mathcal{R}, l \in \mathcal{L}, z \in \mathcal{Z} \tag{17}$$

$$x_{i,o,r}, y_{o,r,p} \in \{0, 1\}, \forall i \in \mathcal{I}, o \in \mathcal{O}^r, r \in \mathcal{R}, p \in \mathcal{P} \tag{18}$$

The objective of determining the returning location of each rack is to minimize the total travel distance, denoted by (6). It consists of two parts: the distance of racks traveling from the current location to a picking station $\sum_{o \in \mathcal{O}^r} \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}} y_{o,r,p} J_{r,l} d_{l,p}$ and the distance that the racks return to a storage location $\sum_{o \in \mathcal{O}^r} \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}} y_{o,r,p} b_{r,l} d_{l,p}$. Constraints (7) make sure that at least one requested rack covers the items needed for an order o during order picking. Constraints (8) ensure that an SKU may be picked from a rack only if it has been assigned to that rack. $a_{i,r}$ is determined in the stowage problem. So, in this formulation, it is known. Constraints (9) represent that only racks parked at a location can be selected for order picking. Constraints (10) guarantee that a rack will be retrieved for an order if at least one product from that order is requested from the rack. Constraints (11) mean that if rack r is requested, it travels to at most one picking workstation. Constraints (12) denote that, at most, one storage location will be assigned to the picked rack r to return to after fulfilling the order. Constraints (13) require each rack to be assigned one zone. Constraints (14) mean a storage location must be assigned to each requested rack r. Constraints (15) ensure that the racks in zone z must return to a location in zone z. Constraints (16) ensure that the difference in the number of picked items by each picking station cannot exceed a certain threshold moderated by parameter G. These constraints ensure that the workload between stations is balanced. Constraints (17) and (18) define the domains of the decision variables.

The values of $J_{r,l}$, $y_{o,r,p}$, and $b_{r,l}$ are linked across consecutive time slots. At slot t, if rack r is requested to pick one or more SKUs in order o, i.e., if $\sum_{p \in \mathcal{P}} y_{o,r,p} = 1$, the location l occupied by rack r is released, i.e., $\sum_{l \in \mathcal{L}} J_{r,l} = 0$ at slot t. After picking, the rack r will return to a free location l', i.e., $b_{r,l'} = 1$. Rack r takes $\frac{d_{l',p}}{e}$ time to arrive at location l', where e is the speed of the AGVs. Then, from time slot t to $t + \left\lceil \frac{d_{l',p}}{e\Delta t} \right\rceil$, the status of location l' is updated to occupied, i.e., $J_{r,l'} = 1$. Ensuring $\sum_{r \in \mathcal{R}} J_{r,l'} \leq 1$ prevents multiple racks from parking in the same location.

The stowage and storage problems are two NP-hard problems. The formulation is nonlinear because it contains products of binary decision variables. For their model, Mirzaei et al. [31] report that Gurobi 9.0 solves instances of up to 100 order lines from 50 products within one day. A medium-sized RMFS can involve approximately 1000 racks, more than 10,000 SKUs, and up to 4000 orders per day. We do not convert the formulation into a mixed-integer linear program (MILP) or solve it with an exact optimization solver. This focus on practical operating scales motivates the use of the three-stage heuristic in Section 5 for the computational evaluation.

5. The Solution Method

We propose a three-stage approach to solve the proposed model in Section 4. In Section 5.1, we analyze the SKU features. Then, we design a three-stage algorithm in Section 5.2 to determine stowage and storage assignment decisions by incorporating the insights obtained from the SKU feature analysis. Figure 2 shows an overview of the proposed solution scheme.

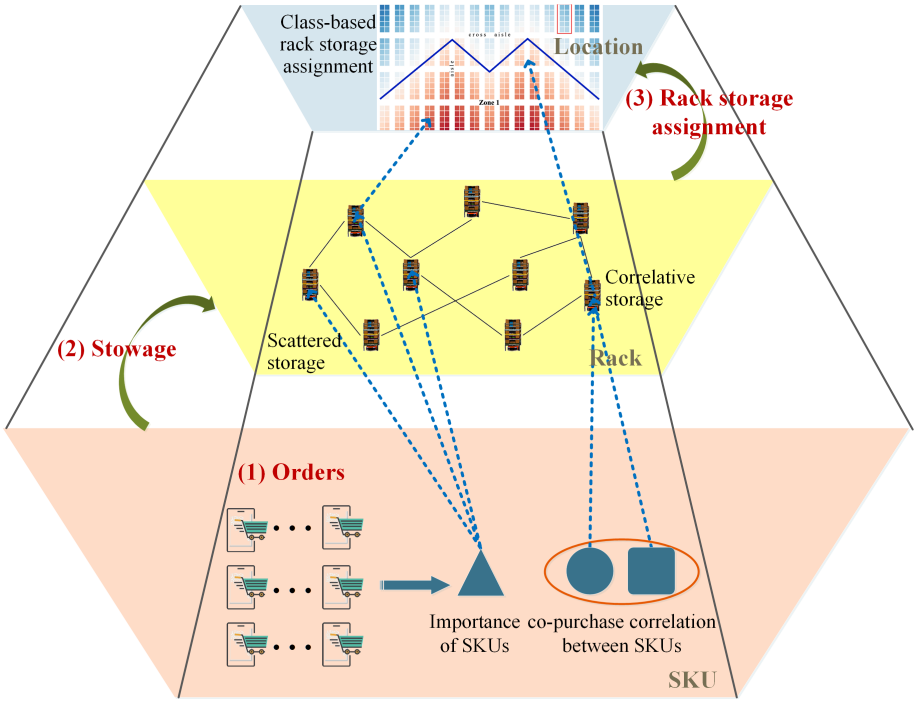


Figure 2. An overview of the proposed solution scheme to the stowage and storage assignment problems with workload balance constraints. (1) Starting from the bottom of this figure, we analyze co-purchase correlations of SKUs embedded in the orders. (2) At the middle rack layer, these co-purchase correlations can assist stowage decisions. It consists of two decisions: how many racks the SKU should be scattered (scattered storage) and which SKUs should be stored on one rack (correlative storage). (3) After that, rack storage assignment is performed to achieve workload balance.

5.1. From Order Data to SKU Features

We construct five features to characterize SKU popularity, temporal stability, and correlation. These features are detailed below. We leverage historical order data to derive SKU features, aiming to cluster products for optimized rack allocation. While existing literature typically groups SKUs based solely on turnover or sales [24,33], our approach incorporates multidimensional demand characteristics.

5.1.1. SKU Sales

Sales volume is a primary indicator of product popularity, distinguishing high-demand from low-demand SKUs. Let q_i denote the total sales of SKU i over a specific period (e.g., the training horizon). It is calculated using Equation (19), where $\mathrm{q}_{i,o}$ represents the quantity of SKU i in order o.

$$\mathrm{q}_i = \sum_{o \in \mathcal{O}} \mathrm{q}_{i,o}, \forall i \in \mathcal{I} \tag{19}$$

5.1.2. SKU Frequency

SKU frequency f_i represents the number of distinct orders containing SKU i. This metric complements sales volume q_i by differentiating between bulk purchases and frequent purchases. For instance, an SKU with 100 units in a single order has $\mathrm{q}_i = 100$ and $f_i = 1$, whereas an SKU appearing in 100 separate orders has $f_i = 100$. f_i is defined in Equation (20), where $c_{i,o}$ is a binary indicator for the presence of SKU i in order o.

$$f_i = \sum_{o \in \mathcal{O}} c_{i,o}, \forall i \in \mathcal{I} \tag{20}$$

5.1.3. Short/Long-Term Bestseller

Aggregate sales metrics (q_i and f_i) may fail to capture temporal distribution, such as sudden sales spikes driven by promotional events. To distinguish between consistent long-term bestsellers and short-term trends, we introduce the concept of non-zero purchase days. Let O_t denote the set of orders on day $t \in T$. The binary variable $g_{i,t}$ indicates whether SKU i was purchased on day t, as shown in Equation (21). T is a relatively long period of time. In our experiments, T corresponds to one quarter.

$$g_{i,t} = \begin{cases} 1 & \text{if } \exists o \in O_t \text{ s.t. } c_{i,o} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Subsequently, n_i aggregates the total number of days SKU i was purchased, calculated via Equation (22).

$$n_i = \sum_{t \in T} g_{i,t}, \forall i \in \mathcal{I} \tag{22}$$

5.1.4. Co-Purchase Degree

We analyze product relationships by constructing a co-purchase graph where vertices represent SKUs. The degree of a vertex, d_i, represents the number of unique SKUs that have been co-purchased with SKU i at least once. This is defined in Equation (23).

$$\mathrm{d}_i = \sum_{j \in \mathcal{I}} \min\left(\sum_{o \in \mathcal{O}} c_{i,o} \times c_{j,o}, 1 \right), \forall i \in \mathcal{I} \tag{23}$$

5.1.5. Weighted Co-Purchase

To capture the strength of these relationships, we incorporate edge weights based on co-purchase frequency. Let $s_{i,j}$ denote the total number of times SKUs i and j appear in the same order, calculated as Equation (24).

$$s_{i,j} = \sum_{o \in \mathcal{O}} c_{i,o} \times c_{j,o}, \forall i, j \in \mathcal{I} \tag{24}$$

Finally, we define the weighted co-purchase feature w_i in Equation (25), which integrates both the node degree $\mathtt{d}_i$ and the co-purchase frequency $s_{i,j}$.

$$w_i = \sum_{j \in \mathcal{I}} \sqrt{\mathtt{d}_i \times s_{i,j}}, \forall i \in \mathcal{I} \tag{25}$$

5.2. The Three-Stage Storage Assignment Algorithm

This section describes a three-stage storage assignment algorithm for solving the stowage and storage problem. We first explain the rationale for the three stages.

Practical experience and prior research indicate that using SKU correlations in stowage can improve picking efficiency [18,31]. There are two strategies to achieve this improvement: placing SKUs on multiple racks [9] and placing frequently co-purchased SKUs on the same rack [27]. These two strategies can reduce the number of racks needed to retrieve the SKUs in an order. The framework combines these strategies and evaluates their effects on rack demand and travel distance. The first stage (Section 5.2.1) clusters SKUs using the features in Section 5.1. It determines the number of racks required for scattering SKUs. The second stage (Section 5.2.2) groups frequently co-purchased SKUs on the same rack subject to rack capacity. In the third stage (Section 5.2.3), we address workload balance through rack storage assignment using a class-based policy.

To address the stowage problem, we develop a clustering algorithm that incorporates capacity constraints. The algorithm enhances the probability of retrieving multiple SKUs from a single rack, helping reduce the number of requested racks and improving operational efficiency. We then implement a rack storage assignment policy with workload-aware station selection. In the policy, we introduce a novel method called SKU-Correlation-Based Rack Velocity Calculation to evaluate the frequency at which racks need to be repositioned, in which storage locations are classified into zones based on their proximity to the nearest picking station. In contrast, the racks are classified according to their velocity. Finally, requested racks are randomly repositioned to their pre-assigned storage zones after picking. The class-based policy permits shared return locations. In addition, we use cumulative picked-item counts to redirect requested racks to the less-loaded station when the relative gap exceeds the trigger.

5.2.1. Scattered SKU Stowage

Stage 1 uses the five demand and network-summary features in Section 5.1 to determine SKU replication counts. Before DBSCAN, each feature is standardized over the training SKU set by subtracting its mean and dividing by its population standard deviation, as indicated by StandardScaler in Algorithm 1. DBSCAN uses Euclidean distance in this standardized feature space. The Calinski–Harabasz (CH) index compares between-group and within-group dispersion. It provides a feature-space criterion rather than a direct measure of travel distance or workload balance. We evaluate the two key DBSCAN parameters, EPS and MinPts, through a grid search using the CH index to compare clustering quality.

We set the maximum number of racks in which an SKU i can be stored as M. We then describe how to get the value of v_i for each SKU cluster $c \in \{1, \ldots, K\}$. For each cluster c, we first obtain its rank x as described in line 24 of Algorithm 1. The v_i of the SKU i in cluster c equals $max(1, round(\frac{x-1}{K-1} \times M)$. This ensures that each SKU will be assigned one rack at least. The higher the rank, the closer the v_i is to M. The pseudo-code for the first stage is shown in Algorithm 1.

Algorithm 1: Scattered stowage based on DBSCAN clustering (First stage)

Input: q_i, f_i, n_i, d_i, w_i of SKU i; upper bound for scattered stowage of SKUs M; parameter of DBSCAN: ε (neighborhood radius), $MinPts$ (threshold for core points)

Output: scattered stowage solution: v_i of SKU $i \in \mathcal{I}$

$q_i \leftarrow StandardScaler(q_i)$
$f_i \leftarrow StandardScaler(f_i)$
$n_i \leftarrow StandardScaler(n_i)$
$d_i \leftarrow StandardScaler(d_i)$
$w_i \leftarrow StandardScaler(w_i)$
initialize all nodes as unvisited;
for *each node i in* $\mathcal{I}$ **do**
    if *node i is visited* **then**
        continue
    mark node i as visited
    get the neighbors B_i of i
    if $|B_i| < MinPts$ **then**
        mark i as noise
    else
        create a cluster g, and add i to g
        for *each neighbor node j in* B_i **do**
            if *j is not visited* **then**
                mark j as visited
                get the neighbors B_j of j
                if $|B_j| \geq MinPts$ **then**
                    $B_i \leftarrow B_i \cup \{j' \in B_j : j' \text{ is unvisited}\}$
            if *j does not belong to any cluster* **then**
                add j to cluster g;
for $c = 1$ *to* K **do**
    obtain the rank value x of cluster $c \leftarrow$ ascending ranking cluster c according to $avg(c) = \frac{\sum_{i \in c} w_i}{|c|}$
    while $i \leq |\mathcal{I}|$ **do**
        if *i in cluster c* **then**
            get the v_i for SKU $i \leftarrow max(1, round(\frac{x-1}{K-1} \times M)$

5.2.2. Associative Stowage Based on SKU Correlations

The second stage involves determining the assignment of SKUs to each rack for scattered storage. To solve this assignment problem, a basic idea is to cluster the SKUs according to the SKU correlation network, where each cluster corresponds to a rack. However, there are two challenges in designing the clustering algorithm: a. The rack capacity constraints. Most clustering methods in machine learning often result in clusters with varying numbers of nodes, which may not be feasible for practical applications due to rack capacity constraints. b. Scattered storage. Existing clustering methods for the RMFS can only cluster each node into a single rack, whereas scattered storage requires that each SKU be assigned to multiple racks [26,27].

To overcome the two challenges, we introduce a novel clustering algorithm based on the SKU correlation network. In contrast to clustering all SKUs at once, this algorithm iteratively forms groups by updating the remaining SKU correlation network. The algorithm first selects initial cluster centers by introducing a weighted degree. Then, following a greedy rule, it groups nodes with the strongest correlations into the same cluster, thereby maximizing the correlation among SKUs within the same cluster. By doing so, it can reduce the search space, ensure that rack capacity constraints are satisfied and allow for the storage of a single SKU in multiple racks. The pseudo-code of the proposed algorithm is shown in Algorithm 2.

Algorithm 2: Associative stowage based on co-purchase correlations (Second stage)

Input: weighted co-purchase $s_{i,j}$; scattered stowage solution v_i; the maximum number of SKUs that can be stored on a rack N; the set of SKUs $\mathcal{I}$

Output: associative stowage solution $a_{i,r}$

initial $|\mathcal{R}| \leftarrow \lceil \frac{\sum_{i \in \mathcal{I}} v_i}{N} \rceil$, $a_{i,r} \leftarrow 0$
for $r = 1$ *to* $|\mathcal{R}|$ **do**
  initial clustering center $i \leftarrow max(w_i)$
  $a_{i,r} \leftarrow a_{i,r} + 1, v_i \leftarrow v_i - 1$
  if $v_i == 0$ **then**
    update $\mathcal{I} \leftarrow$ delete i from $\mathcal{I}$
    $s_{i,j} \leftarrow 0, \forall j \in \mathcal{I}$
  while $|r| \leq N$ **do**
    select SKU $j \leftarrow max(\sum_{i \in r} s_{i,j})$
    add j to r
    $a_{j,r} \leftarrow a_{j,r} + 1, v_j \leftarrow v_j - 1$
    if $v_j == 0$ **then**
      update $\mathcal{I} \leftarrow$ delete j from $\mathcal{I}$
      $s_{i,j} \leftarrow 0, \forall j \in \mathcal{I}$
    if $|\mathcal{I}| == 0$ **then**
      break

5.2.3. The Rack Storage Assignment with Workload Balance

Stage 3 combines rack-to-zone assignment with operational station selection and rack-return decisions. Its contribution therefore includes operational workload control. The framework does not identify storage design as the primary source of imbalance. A relative workload trigger guides reassignment, but indivisible rack visits need not yield an exact final bound for every demand sequence.

The third stage monitors station workload and redirects requested racks to the less-loaded station when reassignment is needed. Shared class-based storage can affect workload distribution together with the dispatch policy. Following the zone-based method in [17], we divide locations into zones according to their distance from the nearest picking station. It aims to optimize the storage assignment by placing frequently accessed racks (high-velocity racks) closer to the picking stations. There are two challenges to applying the policy: calculating the velocity of racks and assigning the racks to the zones.

For the first challenge, the traditional method of calculating rack velocity is based solely on the turnovers of individual SKUs on each rack, which cannot involve the important factor: SKU correlations. For a given rack r, the velocity in the traditional method is determined by summing q_i of all SKUs present on rack r [17,24]. If we directly extend

the traditional method to consider SKU correlations, it suffers from the issue of double counting turnovers for co-purchased SKUs. This issue may lead to inaccuracy in identifying the high-velocity racks and, thus, lower efficiency of assigning racks to storage zones. To address the limitation of the traditional method, we propose a new approach to calculate rack velocity h_r.

$$h_r = \sum_{i \in \mathcal{I}_r} (\lambda_{\{i\}} - \sum_{\substack{j \in \mathcal{I}_r \\ j<i}} \lambda_{\{i,j\}} + \sum_{\substack{j \in \mathcal{I}_r \\ j<i}} \sum_{\substack{k \in \mathcal{I}_r \\ k<j<i}} \lambda_{\{i,j,k\}} - \cdots + (-1)^{|\mathcal{I}_r|-1} \sum_{\substack{j \in \mathcal{I}_r \\ j<i}} \cdots \sum_{\substack{s \in \mathcal{I}_r \\ s<\cdots<j<i}} \lambda_{\{i,j,\cdots,s\}}), \forall r \in \mathcal{R}. \quad (26)$$

$\mathcal{I}_r$ denotes the set of SKUs stored in rack r, and λ_w denotes the number of customer orders that contain all SKUs in the set w. Here, w is a subset of $\mathcal{I}_r$ and may represent any combination of SKUs, such as pairs, triplets, quadruplets, or quintuplets. For example, $\lambda_{\{i\}}$ represents the number of customer orders that contain SKU i in the training order set. Similarly, $\lambda_{\{i,j\}}$ denotes the number of orders that include the combination of SKUs $\{i, j\}$, while $\lambda_{\{i,j,k\}}$ denotes the number of orders containing the combination of SKUs $\{i, j, k\}$, and so forth. The computation of h_r follows a mechanism similar to the inclusion–exclusion principle, which is commonly used to determine the number of elements in the union of multiple finite sets [34]. In implementation, rack velocity counts distinct training orders containing at least one SKU on the rack. This computation evaluates the inclusion–exclusion expression without enumerating SKU subsets.

For the second challenge, we develop a rack storage assignment method based on the proposed h_r. Racks with higher turnover rates are assigned to zones closer to the picking stations. Let $\mathcal{Z}$ denote the set of zones and $\mathcal{L}$ the set of storage locations. Each zone $z \in \mathcal{Z}$ corresponds to a subset of storage locations $\mathcal{L}_z \subseteq \mathcal{L}$, and the subsets $\{\mathcal{L}_z : z \in \mathcal{Z}\}$ form a partition of $\mathcal{L}$. Locations are grouped into zones according to the distance $d_{l,p}$ from location l to its nearest picking station p. The racks are sorted in descending order of h_r, while the zones are ordered by their proximity to the picking stations from closest to farthest. Let $|\mathcal{L}_z|$ denote the number of storage locations in zone z. Racks are then assigned sequentially according to this order: the first $|\mathcal{L}_{z_1}|$ racks are assigned to zone z_1, the next $|\mathcal{L}_{z_2}|$ racks are assigned to zone z_2, and the process continues until all racks have been assigned.

Given rack and location zones, orders are picked on the 'First-Come-First-Served' rule. When fulfilling orders, priority is given to the rack with the most SKUs available for picking. In general, the racks requested are transported to the closest picking station. If the workload is not balanced, reassign the racks requested to the lowest workload station. After order picking, the selected racks are randomly allocated to an unoccupied storage location following the class-based storage zone policy. The process of rack storage assignment is detailed in Algorithm 3.

Algorithm 3: Rack storage assignment based on novel rack velocity (Third stage)

Input: orders to be picked $\mathcal{O}^r$; historical orders $\mathcal{O}^s$; stowage assignment $a_{i,r}$; racks $\mathcal{R}$; zones $\mathcal{Z}$, $m_{l,z}$; status of locations $J_{r,l}$; threshold of workload balance G
Output: rack-to-zone assignment $k_{r,z}$; SKUs picked on racks $x_{i,o,r}$; requested racks $y_{o,r,p}$; rack repositioning $b_{r,l}$

calculate h_r using order data $\mathcal{O}^s$ according to Formula (26)
initial $k_{r,z} \leftarrow 0$
for $z = 1$ *to* $|\mathcal{Z}|$ **do**
  if $\sum_{r\in\mathcal{R}}\sum_{z\in\mathcal{Z}} k_{r,z} < |\mathcal{R}|, \sum_{r\in\mathcal{R}} k_{r,z} < 1$ **then**
    select the unassigned rack r with the maximum h_r
    $k_{r,z} \leftarrow 1$
initial traveling distance $\delta \leftarrow 0$, $t \leftarrow$ earliest arrival time in O^r
while $|O^r| > 0$ **do**
  unpicked orders $O_a \leftarrow$ arrival time of $O^r < t$ (in ascending order of arrival time)
  update the status of locations $J_{r,l}$
  for $o = 1$ *to* $|\mathcal{O}_a|$ **do**
    filter subset $\mathcal{R}_o$ from $\mathcal{R} \leftarrow$ select the racks that satisfy $\sum_{i\in\mathcal{I}} a_{i,r} \times c_{i,o} > 0$
    select rack r with the most SKUs picked from $\mathcal{R}_o \leftarrow$ rack r of max $(\sum_{i\in\mathcal{I}} a_{i,r} \times c_{i,o})$
    if $\sum_{l\in\mathcal{L}} J_{r,l} = 1$ **then**
      if *workload is balanced* **then**
        select picking station $p \leftarrow$ station p satisfying $min(d_{l,p}|J_{r,l} = 1)$
      else
        select picking station $p \leftarrow$ station p with the least workload
      $y_{o,r,p} \leftarrow 1, \delta \leftarrow \delta + \sum_{l\in\mathcal{L}} J_{r,l} \times d_{l,p}$
      for $i = 1$ *to* $|\mathcal{I}|$ **do**
        if $a_{i,r} \times c_{i,o} > 0$ **then**
          $x_{i,o,r} \leftarrow 1, c_{i,o} \leftarrow 0$
      select a location $l \leftarrow random(l|J_{r,l} = 0, m_{l,z} \times z = k_{r,z} \times z)$
      $b_{r,l} \leftarrow 1, \delta \leftarrow \delta + d_{l,p}$
    delete rack r from $\mathcal{R}_o$
    if $|\mathcal{R}_o| > 0$ *and* $\sum_{r\in\mathcal{R}_o}\sum_{i\in\mathcal{I}} a_{i,r} \times c_{i,o} > 0$ **then**
      go to step 13
  delete picked order o from O^r
  $t \leftarrow t + \Delta t$

6. Numerical Experiments

In this section, we evaluate the proposed method through numerical experiments using real-life order data from a large Chinese e-commerce retailer. The discrete-time simulator uses Python 3.9.12 and runs on a MacBook Pro with 16 GB of memory and macOS 12.3. The experiments compare the three-stage algorithm with a benchmark from the literature and examine DBSCAN parameters, workload thresholds, shared/dedicated zoning, rack repositioning and SKU scattering. We use the simulator to compare the two algorithms in terms of rack travel distance, throughput and workload balance (Sections 6.3.1 and 6.3.2). In Section 6.3.3, we compare five combinations of SKU scattering, stowage and rack storage rules by processing the same test orders and recording rack requests and total rack travel distance. This comparison omits station queues, robot contention and picker service times. In Section 6.4.5, we vary EPS and MinPts, recluster the SKUs and rebuild the SKU-to-rack assignments using Stage 2. For each assignment, we select racks to cover the requested SKUs in each evaluation order and count each selection as one rack request.

6.1. Benchmark Algorithm

The benchmark combines two methods from the literature to specify stowage and rack storage assignment. For stowage assignment, Ma et al. [18] propose a scattered-correlation storage policy based on SKU classification and correlation in RMFS. The algorithm of Ma et al. [18] considers both multi-rack scattered storage of SKUs and correlation-based assignment of SKUs to racks. We select this method for the benchmark because it combines multi-rack SKU scattering with correlation-based SKU-to-rack assignment, addressing both stowage decisions considered in this study. For the rack storage assignment, Yuan et al. [17] focus on the storage decision, which is to decide to which storage location within the storage zone to return a picked rack. Their class-based storage policy uses rack velocities. We use this policy for the benchmark's rack storage assignment.

6.2. The Data and the Experimental Design

This study utilizes data from a prominent Chinese e-commerce retailer. The retailer's RMFS warehouse, encompassing 3800 square meters, houses over 10,000 SKUs. We divide the order data into a training set and a test set. Specifically, the training subset comprises 405,389 orders from July to September 2019. The test set contains 86,673 orders from 1 to 14 October 2019. Notably, 43% of these orders contain multiple SKUs, with an average of 1.8 items per order. The full training subset informs stowage and rack storage assignment. We evaluate the benchmark and the three-stage algorithm on the full test subset and on the 48,665 orders in the final week of the training subset. Each day's orders form one experimental case, and training-set operational results refer to this final-week subset.

In the actual RMFS warehouse, each rack stores an average of 29 SKUs, ranging from 5 to 77. The RMFS warehouse has 10007 SKUs and is equipped with two picking workstations. Referring to the practical scenario, we set $|\mathcal{P}|$ to 2, $|\mathcal{I}|$ to 10,007, N to 29, M to 2, $|\mathcal{R}|$ to 366, $|\mathcal{L}|$ to 390. The threshold G to control workload balance is 5%. The Δt of each stage is 30 s. The storage area is divided into two zones. Each zone occupies 50% of the locations. For parameter selection, we vary EPS from 0.1 to 10 in increments of 0.1 and MinPts from 5 to 200 in increments of 5, and we calculate the CH index for each combination in the standardized feature space described in Section 5.2.1. The grid contains 4000 parameter combinations, and selection uses only the training features. The experiments use an EPS value of 0.8 and a MinPts value of 100, which give the highest CH index of 7857.41 in the reported search. The CH criterion selects the feature-space partition and does not directly optimize the downstream operational outcomes.

Cluster 1 contains 9281 SKUs, and Cluster 2 contains 726 SKUs. We label the group of SKUs with higher average q_i, f_i, n_i, d_i, and w_i as Cluster 2. In Table 5, we report the summary statistics of the five training features in the two resulting clusters. The SKUs in Cluster 2 exhibit higher demand in terms of SKU turnover $\mathtt{q}_i$ and SKU frequency f_i, as well as stronger correlations reflected by $\mathtt{d}_i$ and w_i. We also perform t-tests to compare their means. The results indicate that the means of the five features differ statistically significantly in the two resulting clusters. We believe the clustering analysis has separated our sampled SKUs effectively. Supported by extant literature, no more than one rack is assigned for each SKU in Cluster 1, as they are less frequently purchased with other SKUs. In contrast, SKUs in Cluster 2 could be stored on multiple racks if doing so could increase the probability of picking multiple SKUs on a requested rack. We set v_i to be identical for all SKUs from the same cluster.

Table 5. SKU clustering results.

Cluster	SKUs	Statistic	q_i	f_i	n_i	d_i	w_i
1	9281	max	625	400	83	203	317
		min	1	1	1	1	1
		mean	44	36	18	26	41
		std	67	51	17	31	51
2	726	max	10,109	4824	92	1092	3048
		min	65	60	5	8	46
		mean	868	561	68	221	481
		std	1077	506	19	155	348
t-test		*p*-value	2.2×10^{-16}	2.2×10^{-16}	2.2×10^{-16}	2.2×10^{-16}	2.2×10^{-16}
		95% CI of the diff. in means	[850, 1033]	[547, 632]	[48, 51]	[193, 220]	[452, 510]
		mean of cluster 1	46.74	38.79	18.85	28.05	44.14
		mean of cluster 2	987.99	628.14	68.40	234.35	525.16

Feature calculation, the DBSCAN search, SKU-to-rack assignment and rack-velocity ranking form an offline planning step. A dense co-purchase matrix requires storage that grows quadratically with the number of SKUs, while the grid search repeats clustering for each parameter combination. During order fulfillment, the algorithm reuses the resulting assignments and updates rack availability, station workloads and return locations. Practical deployment requires historical order–SKU records, rack capacities, location-to-station distances and current warehouse states. Assignment updates can take place between fulfillment waves, subject to rack capacity constraints.

6.3. Performance of the Three-Stage Algorithm

6.3.1. Picking Efficiency

We first compare the performance of the benchmark algorithm and the three-stage algorithm for stowage assignment. In Table 6, correlation is the sum of the number of co-purchases of any two SKUs on the rack. It is calculated as $\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}} s_{i,j} \times a_{i,r} \times a_{j,r}$. Rack requests count the racks selected to fulfill the given orders. The main comparison uses identical daily orders and warehouse dimensions, with 30 simulations per case. Storage assignments and workload-control policies differ, so the improvements reflect their combined effects.

Table 6 shows that the proposed algorithm increases SKU correlation and reduces rack requests relative to the benchmark. The total correlation among SKUs stored on the same rack increases by 82%, while rack requests decrease by 23% on the test set and 25% on the training set. The reduction in rack requests is slightly larger on the training set. The test results also show fewer rack requests for orders not used to develop the storage assignments.

Table 6. Results of the benchmark and three-stage algorithm.

	Benchmark Algorithm	Three-Stage Algorithm
Correlation (f_1)	28,404	51,827 (+82%)
Requested Racks (test set)	145,006	111,650 (−23%)
Requested Racks (training set)	82,105	61,244 (−25%)

Figures 3–5 present the simulation results for the training cases (24–30 September) and test cases (1–14 October). For each daily case, we conduct 30 simulation runs using the settings described in Section 6.2. Figure 3 shows lower rack travel distances for the three-stage algorithm on both evaluation sets. Relative to the benchmark, the three-stage algorithm reduces mean rack travel distance by 21% on the test set and 23% on the training

set. We calculate the percentage reduction in rack travel distance as the benchmark mean minus the proposed mean, divided by the benchmark mean and multiplied by 100. For each instance, travel-distance variance across 30 simulations is lower for the three-stage algorithm than for the benchmark. This indicates more consistent performance across runs under the tested settings.

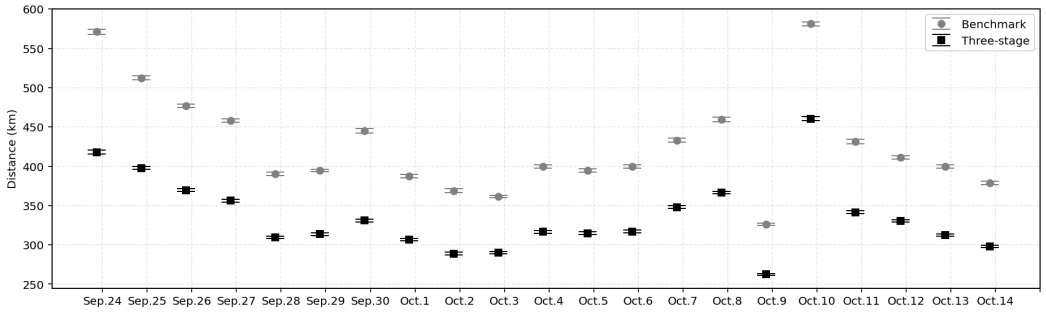


Figure 3. Rack travel distance comparison.

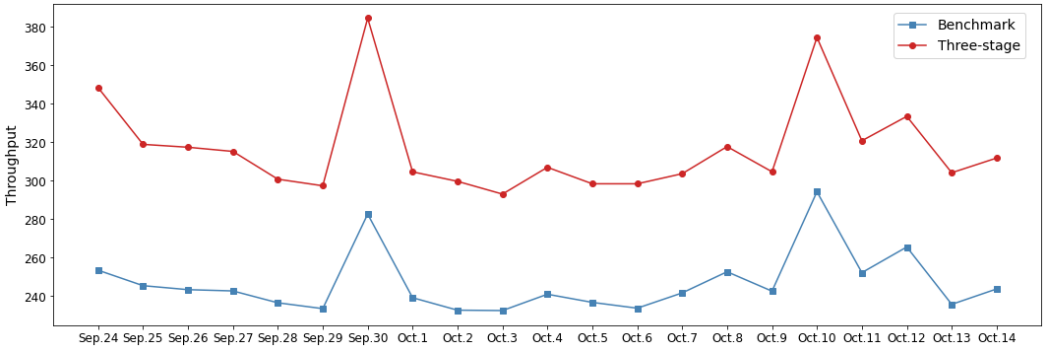


Figure 4. Throughput comparison.

We compare throughput, defined as the average number of items picked per hour including rack transportation time, between the benchmark and the three-stage algorithm. As shown in Figure 4, the three-stage algorithm outperforms the benchmark significantly. On the training set, it picks 91 more items per hour on average, and on the test set, 66 more items per hour, representing a 27% increase in throughput. We express the throughput gain as a percentage of the benchmark mean. These results highlight the efficiency of the three-stage algorithm in reducing picking distances and boosting throughput. Overall, the three-stage algorithm improves correlation, rack requests, rack travel distance and throughput on the evaluated training and test cases.

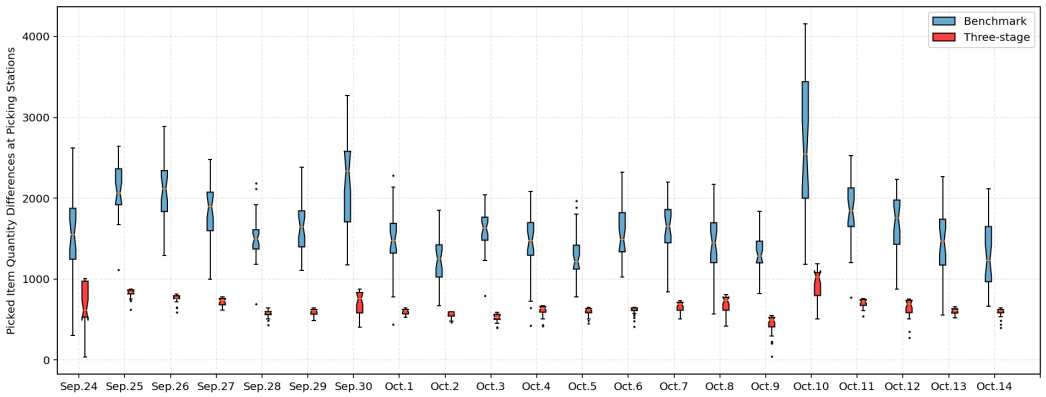


Figure 5. Workload gap comparison.

6.3.2. Workload Balance

In this section, we analyze the difference in workload at the picking stations. The distribution of the gap in the number of picked items by the two stations is shown in Figure 5. The gap is the absolute difference between the picked items of the two picking stations. In the test set, the three-stage algorithm reaches an average gap of 630 picked items per day, while the benchmark algorithm is 1564 on average. The three-stage algorithm improves workload balance by 59.72% compared to the benchmark algorithm ($\frac{1564-630}{1564} = 59.72\%$). In the training set, the gap in the picked items of the three-stage algorithm is 700 on average, while the benchmark algorithm is 1850 on average. The three-stage algorithm improves workload balance by 62.16% compared to the benchmark algorithm ($\frac{1850-700}{1850} = 62.16\%$). The three-stage algorithm performs better on the training set. From the box plot in Figure 5, the workload gap distribution of the three-stage algorithm is more concentrated, with lower variance and mean. These distributions indicate less variation in workload differences between picking stations across 30 simulations per daily case in the training and test subsets.

6.3.3. Stage Component Contributions

We compare the incremental effects of Stage 1 SKU scattering, Stage 2 associative stowage and Stage 3 rack velocity storage. The basic policy randomly selects SKUs to scatter across two racks, uses random stowage subject to rack capacity and assigns racks to random storage locations. We denote this policy by B0. We replace random SKU scattering in B0 with Stage 1 DBSCAN-based scattering to obtain B1. We replace random stowage in B0 with Stage 2 associative stowage to obtain B2. We combine Stages 1 and 2 to obtain B3. Finally, we replace random rack storage in B3 with Stage 3 rack velocity storage to obtain B4. Comparing B3 with B2 assesses the contribution of DBSCAN-based SKU scattering under associative stowage. The five policies use the same order sequence, storage capacity and 5% workload threshold. We evaluate the five policies using the training and test datasets described in Section 6.2.

Table 7 reports rack requests and total rack travel on the full test set over 30 runs of the simplified component replay. This replay omits queues, concurrent robot contention, picker service times and inventory depletion or replenishment. The five policies process orders in FCFS order and repeatedly select the rack covering the largest number of remaining distinct SKUs requested by the current order. Each selected rack goes to the station with fewer cumulative picked items when the absolute difference between the two station totals exceeds 5% of their sum. Otherwise, it goes to the nearest station. This comparison does not estimate throughput or establish how the storage components affect workload balance in the complete simulator. In the rack request and total travel columns, the first value is the mean and the value after $\pm$ is the half-width of the 95% Monte Carlo confidence interval for that mean. The Reduction (%) columns report the decrease in each policy's mean relative to B0, expressed as a percentage of the B0 mean. Racks return to random free locations in the storage area under B0–B3. B4 uses zone-based rack storage, assigning racks to zones according to rack velocity and returning them to random free locations within their assigned zones.

Table 7 shows that associative stowage contributes most of the reduction in rack requests and also substantially reduces rack travel. DBSCAN-based SKU scattering provides a modest improvement with random stowage and an additional benefit when combined with associative stowage. Rack velocity storage further reduces travel distance without changing rack requests. The full three-stage policy combines fewer rack requests with the shortest rack travel among the five policies.

Table 7. Controlled component comparison.

Policy	Rack Requests	Reduction (%)	Total Travel (km)	Reduction (%)
B0: basic policy	$148,728 \pm 37$	–	5658.03 ± 3.40	–
B1: add Stage 1	$148,334 \pm 65$	0.26	5639.83 ± 3.87	0.32
B2: add Stage 2	$113,291 \pm 61$	23.83	4309.60 ± 8.22	23.83
B3: add Stage 1, 2	$111,650 \pm 0$	24.93	4248.41 ± 7.76	24.91
B4: add Stage 1, 2, 3	$111,650 \pm 0$	24.93	3411.87 ± 2.32	39.70

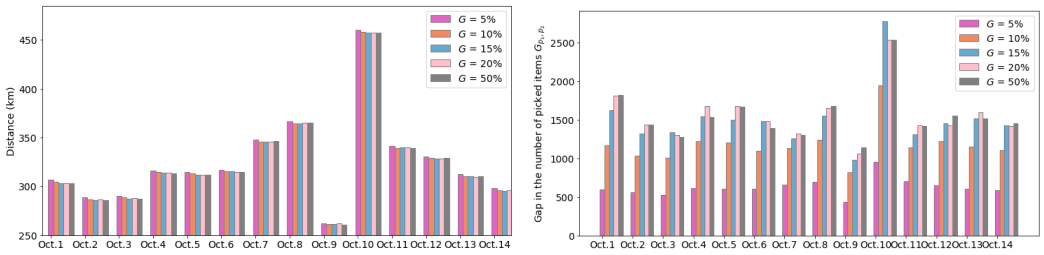
6.4. Sensitivity Analysis

6.4.1. Threshold for Workload Balance

The relative workload threshold G and the absolute station item gap G_{p_1,p_2} describe different aspects of workload balance. Equation (27) defines the absolute gap, with smaller values indicating better balance.

$$G_{p_1,p_2} = \left| \sum_{o \in \mathcal{O}} \sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{I}} x_{i,o,r} \times y_{o,r,p_1} \times \mathrm{q}_{i,o} - \sum_{o \in \mathcal{O}} \sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{I}} x_{i,o,r} \times y_{o,r,p_2} \times \mathrm{q}_{i,o} \right| \tag{27}$$

We compare workload thresholds of 5%, 10%, 15%, 20% and 50% on the test set, with the other parameters fixed at the settings in Section 6.3. This comparison examines how the threshold affects travel distance and workload balance. The average results under 30 runs are shown in Figure 6. The distance represents the average traveling distance for 30 simulations. Workload is reported as the mean absolute station item gap G_{p_1,p_2} across 30 simulations. A larger workload threshold reduces travel distance while increasing workload imbalance. When G exceeds 20%, its impact becomes limited. The experimental results are not significantly different from those at G = 50%.



(**a**) Average rack travel distances (**b**) Average absolute item gap

Figure 6. Workload threshold sensitivity.

The main experiments use a workload threshold of 5% for G to prioritize near-equal station workloads. This is the strictest tested setting and represents a conservative balance target rather than a travel-distance optimum. The existing sensitivity results include a 20% threshold for G, which offers an alternative when travel efficiency receives greater weight.

6.4.2. Dedicated or Shared Zones

Shared zones mean that all picking stations share the locations of the zone. When rack r at picking station p returns to zone z, it can be assigned any free location in zone z. Since racks are randomly assigned locations within zone z, it is possible to assign a far location to rack r instead of a location closer to picking station p. Therefore, we use another zoning called the dedicated zone. The zone z is further divided into $|\mathcal{P}|$ sub-zones based on the distance between the location and the station. The locations in shared zone z are assigned to the nearest picking station p and are dedicated to picking station p. Figure 7 shows an example of shared and dedicated zones.

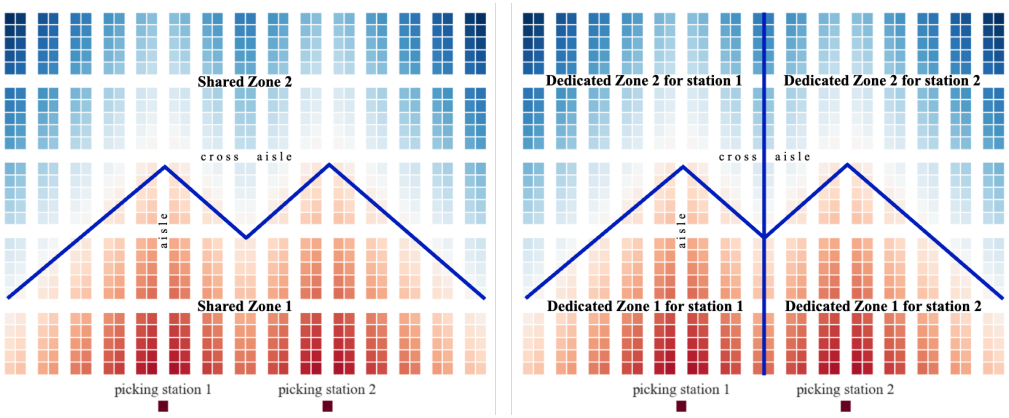
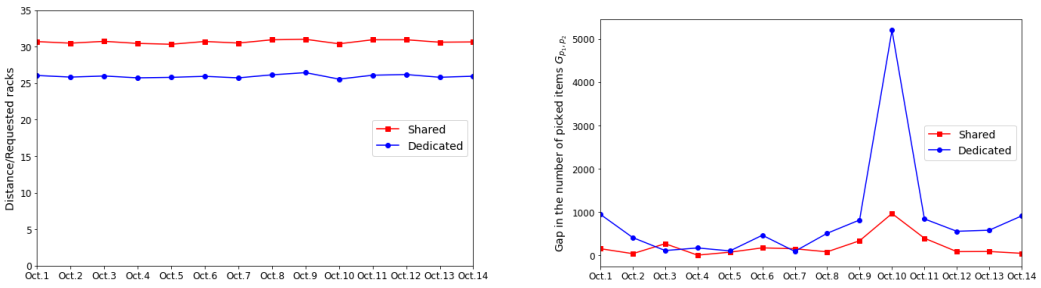


Figure 7. Layout of dedicated and shared zones. Blue lines mark zone boundaries. Colors indicate the distance from each rack storage location to the nearest picking station, from red (shorter) to blue (longer).

We compare shared and dedicated zones without imposing the workload balance constraint. The other parameters match those in Section 6.3, where the workload threshold is 5%. Figure 8 reports the means over 30 runs. Shared zoning increases mean travel per rack request by 4.72 m and reduces the station item gap by 49.77% in this comparison. The choice therefore depends on the relative priorities of travel efficiency and workload balance.

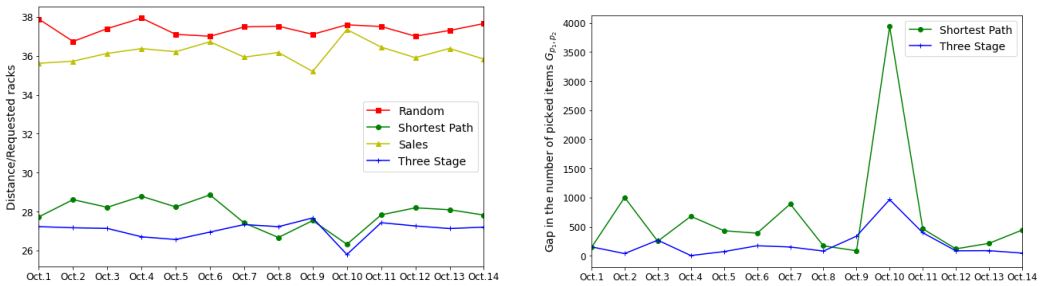


(**a**) Average travel distance per request (**b**) Average absolute item gap

Figure 8. Shared and dedicated zones.

6.4.3. Rack Repositioning Strategy

Besides class-based assignment rules, there are many other rack-repositioning rules. We compare the proposed return policy with three established rack-repositioning rules while keeping stowage assignments fixed. Figure 9 reports average travel distance per rack request in meters and the average absolute difference in picked items between stations for each rack repositioning strategy. These methods include random storage [21], shortest path storage [22], and sales-based storage [32]. The random storage assignment method acts as a benchmark by randomly parking racks in unoccupied rack storage locations. The shortest path storage method matches a picked rack to the closest unoccupied storage location. Sales-based storage prioritizes nearby unoccupied locations for racks with higher sales volumes. Both the rack travel and workload comparisons in Figure 9 use five storage zones and the test cases. They compare return rules within this configuration and do not isolate the effect of zone count. The proposed return policy achieves the lowest mean travel distance among the tested rules and a smaller mean item gap than the closest-location policy under this five-zone configuration.



(**a**) Average travel distance per request (**b**) Average absolute item gap

Figure 9. Rack repositioning comparison.

6.4.4. Number of Racks for Scattered Storage of SKUs

Multi-rack storage of SKUs improves the probability of picking all the SKUs of an order o from a single rack. In the three-stage algorithm, the number of racks for SKU scattered storage is decided by clusters. SKUs in the same cluster have identical replication counts. From the clustering results in Section 5.2.1, the DBSCAN algorithm divides the SKUs into two clusters. Cluster 2 contains 726 SKUs, and Cluster 1 contains 9281 SKUs. The warehouse has a length of 82 m, a width of 81 m, and a capacity of up to 3510 racks. In this sensitivity analysis, the replication count for the selected cluster varies from 1 to 10, while the other cluster retains one rack per SKU. This range replaces the two-rack replication limit in the main experiments. This analysis uses two storage zones and retains the number of stations, SKU assortment and per-rack SKU capacity in Section 6.2. Rack travel distance and throughput are evaluated on the full 1–14 October test set described in that section. Figure 10 shows the results of the sensitivity analysis.

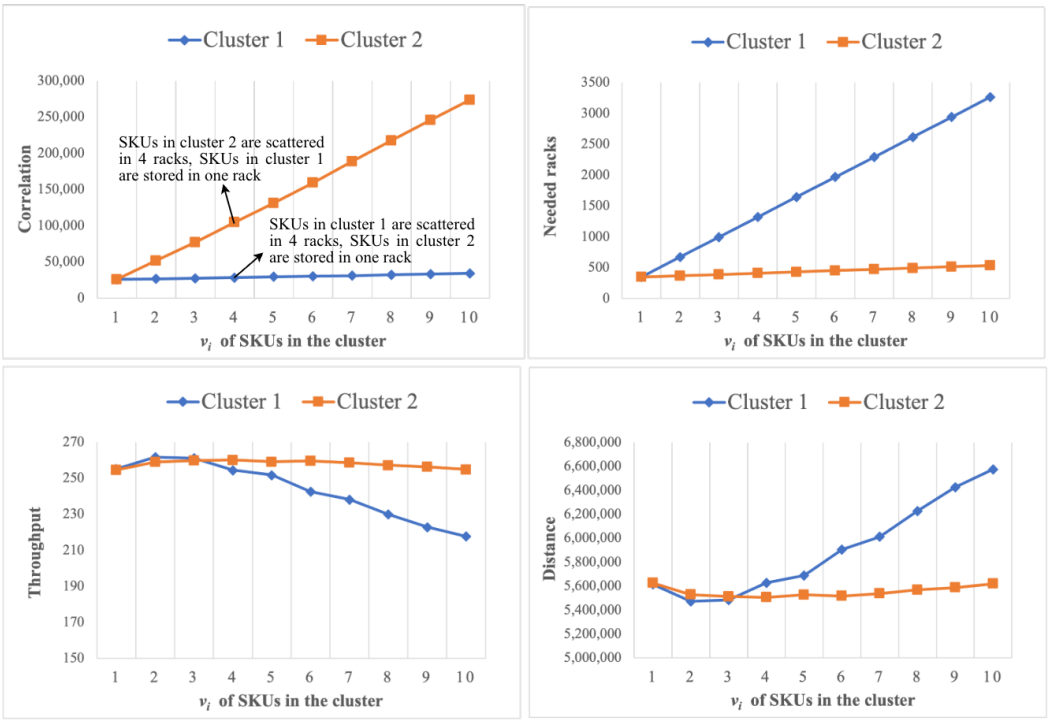


Figure 10. SKU scattering sensitivity.

Scattering SKUs in Cluster 2 across multiple racks rapidly increases total co-purchase correlation while requiring relatively few additional racks. Scattering SKUs in Cluster 1 requires substantially more racks without a comparable reduction in travel distance. For SKUs in Cluster 1, replication across more than three racks increases travel distance and re-

duces throughput. For SKUs in Cluster 2, increasing the replication count from one to three rapidly reduces travel distance, after which the distance stabilizes. When rack resources are limited, replication should prioritize SKUs in Cluster 2. The analysis recommends replication across fewer than three racks under this configuration.

The analysis uses a warehouse with a length of 82 m, a width of 81 m and up to 3510 rack positions. The replication recommendations apply to this configuration, with the number of picking stations and per-rack SKU capacity fixed. The preferred replication count may change with SKU popularity, co-purchase patterns or warehouse layout. The training orders and test orders cover consecutive periods from the same retailer, while performance under seasonal or promotional demand shifts and in other warehouses requires further validation.

6.4.5. DBSCAN Parameters

We vary EPS and MinPts around the main setting and regenerate the SKU-to-rack assignment for each reported combination using the Stage 2 procedure. SKUs in the group with the higher mean co-purchase weighted degree occupy two distinct racks, while SKUs in the other group occupy one rack. Each rack holds at most 29 distinct SKUs, and the available rack budget remains 366. Table 8 reports the CH index and rack requests for the final week of the training set and the full test set described in Section 6.2. This comparison counts rack requests for each order using fixed SKU-to-rack assignments. It does not simulate rack movements, station service or replenishment. For each order, we repeatedly select the rack containing the most remaining distinct requested SKUs until all requested SKUs are covered. Rack requests count these selections, including repeated requests for the same rack across orders.

For the two alternatives reported in Table 8, rack requests differ from the main setting by at most 0.27% on the training set and 0.35% on the test subset. These comparisons indicate stable rack demand for the tested feasible settings. They do not establish parameter robustness in travel distance, throughput or workload balance.

Table 8. DBSCAN parameter sensitivity.

EPS	MinPts	CH	Requested Racks (Training Set)	Requested Racks (Test Set)
0.8	100	7857.41	61,244	111,650
0.8	50	7478.38	61,391	111,993
0.9	100	7682.01	61,408	112,039

7. Discussion and Managerial Implications

7.1. Discussion

In the evaluated two-station warehouse, the complete policy improves picking efficiency and station workload balance by combining proactive stowage and rack-to-zone assignment with picking station selection and rack returns. The empirical observations do not establish storage configuration as the primary source of workload imbalance.

For the orders in the training set, the complete policy reduces rack travel distance and the absolute station workload gap, with the latter decreasing by over 60%. The simplified component comparison supports conclusions about rack requests and travel distance under its execution rules, without attributing the main throughput or workload gains to individual stages. The main simulation results suggest that co-purchase information can support both human-centric and efficiency-oriented design objectives when storage planning is combined with operational balancing. The performance improvements reflect the combined policy and do not isolate the effects of storage design from station selection and rack returns. The sensitivity results support choosing workload thresholds, zoning and

SKU scattering according to the warehouse configuration and the relative priorities of travel efficiency and workload balance. The DBSCAN comparison shows similar rack requests for the reported nearby parameter settings without establishing stability in throughput, travel distance or workload balance.

The three-stage algorithm is a general algorithm that can be extended to scenarios with an arbitrary number of picking stations. The two-station configuration in the experiments reflects the industrial partner's warehouse layout. To extend the algorithm to three or more stations, we retain the stowage algorithm and update only the zoning of storage locations for the new station layout. We calculate the distance from each storage location to each picking station and redefine the zones according to each location's distance to its nearest station. The existing rack storage assignment procedure then uses the updated zones. If order demand also changes, the training features and rack velocities require updating. Performance in these configurations requires further validation.

7.2. Managerial Implications

Drawing on the numerical results and sensitivity analyses presented in Section 6, we offer the following actionable recommendations for engineering managers and logistics practitioners responsible for RMFS design and operations.

Combine storage planning with operational workload control. Workload imbalance among human pickers can reflect both inventory placement and operational scheduling decisions. Managers should coordinate SKU-to-rack stowage and rack-to-zone assignment with real-time workload balancing. Firms can use off-peak periods, such as intervals between fulfillment waves or overnight windows, to update these storage assignments.

Monitor station workload alongside picking efficiency. Managers can monitor differences in station picked-item counts alongside throughput and rack travel distance. These counts indicate workload distribution but do not directly measure picker fatigue, picking errors or worker well-being. The workload threshold of 5% for G represents the strictest tested setting and prioritizes balance, while larger values can reduce rack travel distance when efficiency receives greater weight.

Evaluate shared zoning against travel and workload priorities. While dedicated zone configurations offer marginally shorter travel paths, our sensitivity analysis reports 49.77% better workload balance with shared zones at an additional 4.72 m per rack movement without imposing the workload balance constraint. These results support selecting the zoning policy according to travel and workload priorities, while performance under flash sales, promotional events or seasonal peaks requires further validation. A hybrid policy that shares zones for high-velocity racks while reserving dedicated zones for other racks is a possible extension, but the current comparisons do not evaluate it.

Use co-purchase data to inform stowage and rack storage assignment. SKU co-purchase information is routinely harvested for front-end applications such as product recommendation and promotional bundling. Our results demonstrate that this same data asset carries substantial untapped value for back-end logistics. The three-stage framework combines targeted scattering of the top 7% of SKUs in the high-demand, high-correlation cluster with associative stowage and rack velocity storage. Managers can use historical order data to update SKU-to-rack and rack-to-zone assignments, subject to rack capacity and operational constraints.

8. Conclusions

This paper investigates the joint stowage and rack storage assignment problem in Robotic Mobile Fulfillment Systems, with explicit attention to the trade-off between operational efficiency and picker workload balance. Motivated by station-level workload

differences observed in operational data, we develop a data-driven three-stage algorithmic framework that integrates SKU co-purchase correlation analysis, capacity-constrained rack clustering, and a correlation-adjusted velocity-based zone assignment with flexible zone-sharing repositioning. The simulation results show that the proposed three-stage algorithm achieves a 23% reduction in rack travel distance, a 27% increase in throughput and a 62.16% improvement in workload balance relative to the benchmark from the literature on the training set. These gains reflect storage assignments together with station selection and rack returns under the modeled conditions. The simplified component comparison quantifies changes in rack requests and travel distance from adding SKU scattering, associative stowage and rack velocity storage to the random basic policy. These results do not identify storage configuration as the sole or primary cause of imbalance. The findings support coordinating stowage and rack storage assignment with operational workload control. The component results do not isolate the sources of the throughput or workload improvements reported for the proposed algorithm, and transfer to other facilities requires further validation.

The main simulator represents order arrivals, rack and location availability, workload-based dispatch and rack transportation over time. It does not explicitly model traffic conflicts between robots, variable picker service times or replenishment. Robot congestion can add waiting time along shared routes, so shorter travel distance need not produce a proportional throughput gain. Variable service times can create station queues even when item counts are similar. Replenishment can occupy racks and routes and change inventory availability, affecting rack requests and station workloads. The main results remain conditional on these modeling assumptions, and the simplified comparisons do not quantify these operational effects.

This study has several limitations that point toward productive avenues for future inquiry. First, the two illustrative within-picker comparisons remain limited and do not separate station, day and worker-condition effects. This study uses operational data from one Chinese e-commerce retailer during specific periods in 2019, with workload evidence from two RMFS warehouses. The training–test split provides a temporal holdout but does not establish generalizability across retailers, seasons, promotions, or warehouse geometries. Second, the current framework assumes that all test-set SKUs also appear in the training data. Newly emerging SKUs cannot be handled directly by the trained stowage policy. Future work should investigate dynamic allocation strategies for cold-start SKUs, potentially drawing on transfer learning or contextual bandit approaches. Third, the model adopts FCFS order processing and assumes sufficient inventory for tractability. Integrating order batching, priority scheduling, stockout states, and replenishment decisions could change the relative value of storage policies. Fourth, the main comparisons use a two-station warehouse, while the scattering sensitivity analysis uses a larger configuration. Extending the approach to facilities with more stations, different zone topologies, or different rack inventories requires additional calibration and validation. Finally, future research can investigate how changes in order demand and patterns of robot congestion inform periodic adjustments to SKU-to-rack and rack-to-zone assignments.

Author Contributions: Conceptualization, T.D., T.H. and Y.Z.; methodology, T.D. and T.H.; software, T.D.; validation, T.H. and Y.Z.; formal analysis, T.D., T.H. and Y.Z.; investigation, T.D., T.H. and Y.Z.; data curation, T.D., T.H. and Y.Z.; writing—original draft preparation, T.D., T.H. and Y.Z.; writing—review and editing, T.D., T.H. and Y.Z.; visualization, T.D.; supervision, Y.Z.; project administration, T.D., T.H. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant number 72501157). Teng Huang's work was partially supported by the National Natural Science Foundation of China (grant numbers 72571292 and 72101277).

Data Availability Statement: The order data provided by the company cannot be made publicly available due to confidentiality restrictions. Details of the experimental settings and the source code used in this study are available from the authors upon request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Zou, B.; Gong, Y.; Xu, X.; Yuan, Z. Assignment rules in robotic mobile fulfillment systems for online retailers. *Int. J. Prod. Res.* **2017**, *55*, 6175–6192. [CrossRef]
2. Li, X.; Hua, G.; Huang, A.; Sheu, J.B.; Cheng, T.; Huang, F. Storage assignment policy with awareness of energy consumption in the Kiva mobile fulfillment system. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *144*, 102158. [CrossRef]
3. Allgor, R.; Cezik, T.; Chen, D. Algorithm for robotic picking in amazon fulfillment centers enables humans and robots to work together effectively. *INFORMS J. Appl. Anal.* **2023**, *53*, 266–282. [CrossRef]
4. Sheu, J.B.; Kumar, S.; Kundu, T. Will too many cooks (robots) spoil the broth (order fulfillment)? Navigating priority orders in warehousing operations of e-commerce supply chains. *IEEE Trans. Eng. Manag.* **2024**, *71*, 6063–6077. [CrossRef]
5. Löffler, M.; Boysen, N.; Schneider, M. Picker routing in AGV-assisted order picking systems. *INFORMS J. Comput.* **2022**, *34*, 440–462. [CrossRef]
6. Boysen, N.; Schwerdfeger, S.; Stephan, K. A review of synchronization problems in parts-to-picker warehouses. *Eur. J. Oper. Res.* **2023**, *307*, 1374–1390. [CrossRef]
7. Zhai, M.; Wang, Z.; Sheu, J.B.; Hu, B. Item storage assignment for mobile-rack warehouses in the industry 4.0 Era. *IEEE Trans. Eng. Manag.* **2025**, *72*, 1877–1891. [CrossRef]
8. Wang, Z.; Sheu, J.B.; Teo, C.P.; Xue, G. Robot scheduling for mobile-rack warehouses: Human-robot coordinated order picking systems. *Prod. Oper. Manag.* **2022**, *31*, 98–116. [CrossRef]
9. Lamballais, T.; Roy, D.; De Koster, R. Inventory allocation in robotic mobile fulfillment systems. *IISE Trans.* **2020**, *52*, 1–17. [CrossRef]
10. De Koster, R.; Le-Duc, T.; Roodbergen, K.J. Design and control of warehouse order picking: A literature review. *Eur. J. Oper. Res.* **2007**, *182*, 481–501. [CrossRef]
11. Vanheusden, S.; Van Gils, T.; Braekers, K.; Ramaekers, K.; Caris, A. Analysing the effectiveness of workload balancing measures in order picking operations. *Int. J. Prod. Res.* **2022**, *60*, 2126–2150. [CrossRef]
12. Cheng, B.; Xie, T.; Wang, L.; Tan, Q.; Cao, X. Deep reinforcement learning driven cost minimization for batch order scheduling in robotic mobile fulfillment systems. *Expert Syst. Appl.* **2024**, *255*, 124589. [CrossRef]
13. Sun, Y.; Zhao, N.; Tang, L.; Luo, L. Breaking the Limit on the Number of Robots Through the Conflict-Free Scheduling in Robotic Mobile Fulfillment Systems. *IEEE Trans. Autom. Sci. Eng.* **2024**, *22*, 7324–7334. [CrossRef]
14. Justkowiak, J.E.; Pesch, E. A column generation driven heuristic for order-scheduling and rack-sequencing in robotic mobile fulfillment systems. *Omega* **2023**, *120*, 102897. [CrossRef]
15. Sheu, J.B.; Choi, T.M. Can we work more safely and healthily with robot partners? a human-friendly robot–human-coordinated order fulfillment scheme. *Prod. Oper. Manag.* **2023**, *32*, 794–812. [CrossRef]
16. Zhuang, Y.; Zhou, Y.; Yuan, Y.; Hu, X.; Hassini, E. Order picking optimization with rack-moving mobile robots and multiple workstations. *Eur. J. Oper. Res.* **2022**, *300*, 527–544. [CrossRef]
17. Yuan, R.; Graves, S.C.; Cezik, T. Velocity-based storage assignment in semi-automated storage systems. *Prod. Oper. Manag.* **2019**, *28*, 354–373. [CrossRef]
18. Ma, Z.; Wu, G.; Ji, B.; Wang, L.; Luo, Q.; Chen, X. A novel scattered storage policy considering commodity classification and correlation in robotic mobile fulfillment systems. *IEEE Trans. Autom. Sci. Eng.* **2022**, *20*, 1020–1033. [CrossRef]
19. Lamballais, T.; Roy, D.; De Koster, R. Estimating performance in a robotic mobile fulfillment system. *Eur. J. Oper. Res.* **2017**, *256*, 976–990. [CrossRef]
20. Ding, T.; Zhang, Y.; Wang, Z.; Hu, X. Velocity-based rack storage location assignment for the unidirectional robotic mobile fulfillment system. *Transp. Res. Part E Logist. Transp. Rev.* **2024**, *186*, 103533. [CrossRef]
21. Merschformann, M.; Lamballais, T.; De Koster, M.; Suhl, L. Decision rules for robotic mobile fulfillment systems. *Oper. Res. Perspect.* **2019**, *6*, 100128. [CrossRef]

22. Weidinger, F.; Boysen, N.; Briskorn, D. Storage assignment with rack-moving mobile robots in KIVA warehouses. *Transp. Sci.* **2018**, *52*, 1479–1495. [CrossRef]
23. Roodbergen, K.J.; Vis, I.F. A survey of literature on automated storage and retrieval systems. *Eur. J. Oper. Res.* **2009**, *194*, 343–362. [CrossRef]
24. Cezik, T.; Graves, S.C.; Liu, A.C. Velocity-based stowage policy for a semiautomated fulfillment system. *Prod. Oper. Manag.* **2025**, *34*, 3400–3415. [CrossRef]
25. Weidinger, F.; Boysen, N. Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transp. Sci.* **2018**, *52*, 1412–1427. [CrossRef]
26. Kim, H.J.; Pais, C.; Shen, Z.J.M. Item assignment problem in a robotic mobile fulfillment system. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 1854–1867. [CrossRef]
27. Mirzaei, M.; Zaerpour, N.; De Koster, R. The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance. *Transp. Res. Part E Logist. Transp. Rev.* **2021**, *146*, 102207. [CrossRef]
28. Kumar, S.; Sheu, J.B.; Kundu, T. Planning a parts-to-picker order picking system with consideration of the impact of perceived workload. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *173*, 103088. [CrossRef]
29. Grosse, E.H.; Glock, C.H.; Jaber, M.Y.; Neumann, W.P. Incorporating human factors in order picking planning models: framework and research opportunities. *Int. J. Prod. Res.* **2015**, *53*, 695–717. [CrossRef]
30. Aldarondo, F.J.; Bozer, Y.A. Expected distances and alternative design configurations for automated guided vehicle-based order picking systems. *Int. J. Prod. Res.* **2022**, *60*, 1298–1315. [CrossRef]
31. Mirzaei, M.; Zaerpour, N.; de Koster, R.B. How to benefit from order data: Correlated dispersed storage assignment in robotic warehouses. *Int. J. Prod. Res.* **2022**, *60*, 549–568. [CrossRef]
32. Boysen, N.; Briskorn, D.; Emde, S. Parts-to-picker based order processing in a rack-moving mobile robots environment. *Eur. J. Oper. Res.* **2017**, *262*, 550–562. [CrossRef]
33. Hausman, W.H.; Schwarz, L.B.; Graves, S.C. Optimal storage assignment in automatic warehousing systems. *Manag. Sci.* **1976**, *22*, 629–638. [CrossRef]
34. Flamand, T.; Ghoniem, A.; Maddah, B. Store-wide shelf-space allocation with ripple effects driving traffic. *Oper. Res.* **2023**, *71*, 1073–1092. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.