

Joint storage assignment and picker routing optimization in warehouses

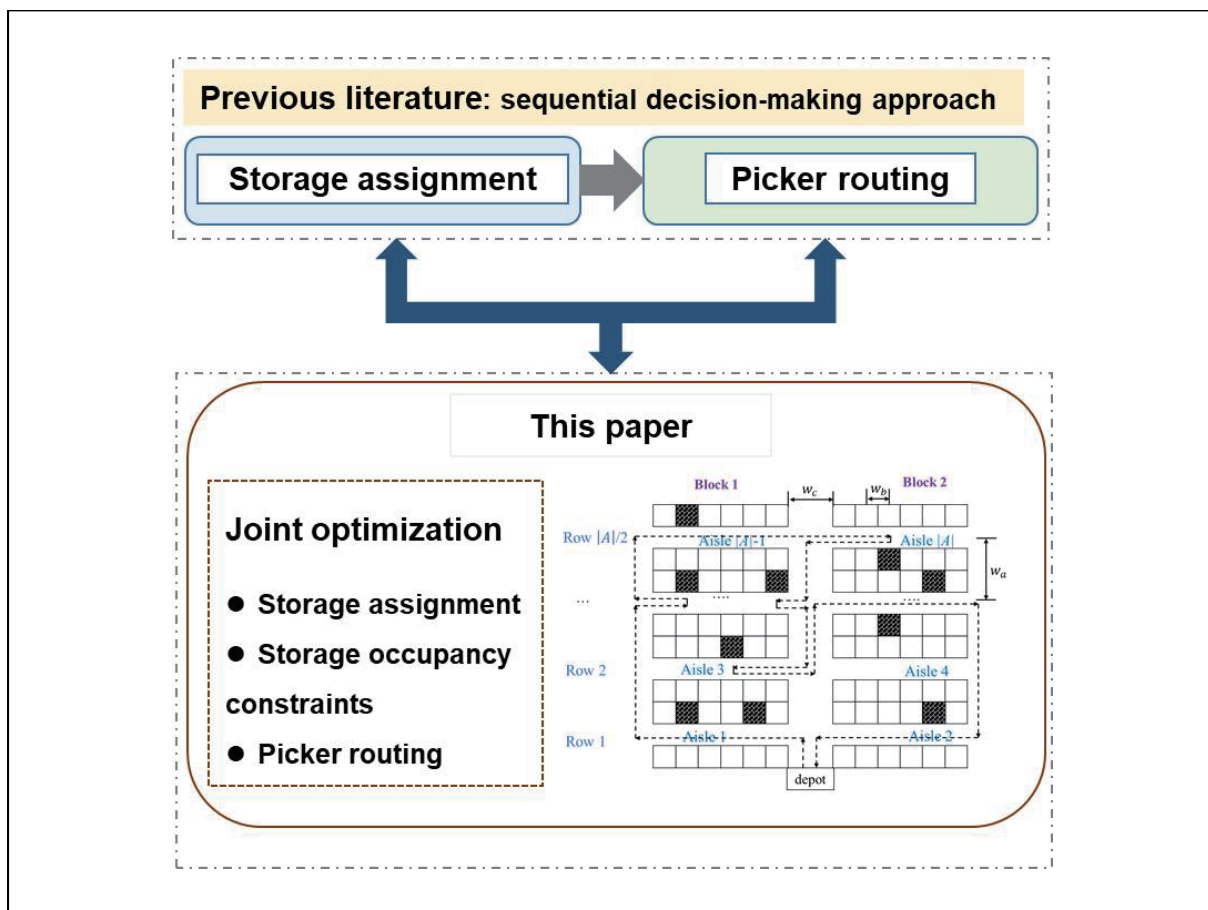
Ran Chen, Jingjing Yang , and Zhongqiang Ma

International Institute of Finance, School of Management, University of Science and Technology of China, Hefei 230026, China

 Correspondence: Jingjing Yang, E-mail: yjj2015@mail.ustc.edu.cn

© 2026 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Graphical abstract



Transition from the sequential approach to the joint optimization framework.

Public summary

- An integrated optimization model for storage assignment and picker routing with storage occupancy constraints is proposed.
- The numerical results show that the proposed algorithm quickly attains optimal or near-optimal solutions.
- Compared with the sequential approach, the integrated approach reduces the total processing time by approximately 12.6%.

Joint storage assignment and picker routing optimization in warehouses

Ran Chen, Jingjing Yang , and Zhongqiang Ma

International Institute of Finance, School of Management, University of Science and Technology of China, Hefei 230026, China

 Correspondence: Jingjing Yang, E-mail: yjj2015@mail.ustc.edu.cn

© 2026 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).



Cite This: *JUSTC*, 2026, 56(X): (10pp)



Read Online



Supporting Information

Abstract: Storage assignment and picker routing are critical factors influencing warehouse operational efficiency, with inherent interdependence between them. However, existing studies rarely investigate their coupling relationships, instead adopting a sequential decision-making approach, optimizing one while keeping the other fixed. Moreover, most prior studies assume an initially empty warehouse, neglecting the practical scenario where certain storage locations are occupied during previous picking operations. This paper establishes an integrated optimization model for storage assignment and picker routing that incorporates storage occupancy constraints, aiming to minimize the total processing time required to complete given orders. The problem is proven to be NP-hard, and a variable neighborhood search (VNS) algorithm is proposed. Numerical experiments demonstrate that the algorithm quickly obtains optimal solutions for small-scale instances and near-optimal solutions for large-scale instances within a short computation time. Compared with the sequential approach, the integrated approach reduces the total processing time by approximately 12.6%. Furthermore, in contrast to methods in the literature that ignore storage occupancy constraints, the proposed algorithm can reduce the total processing time by up to 7.8%. Finally, a real-world case study is used to validate the superiority of the model and algorithm.

Keywords: warehousing; storage assignment; variable neighborhood search algorithm; routing

CLC number: C934

Document code: A

1 Introduction

Order picking constitutes over 55% of the total operating costs in DCs^[1], necessitating optimization to improve operational efficiency. Many distribution centers partition their space into a forward picking area (FPA) and a reserve storage area (RSA). The FPA is primarily used for rapid order picking, whereas the RSA is used for bulk storage and replenishment, as shown in Fig. 1. Designing the FPA involves critical decisions, such as determining its size, selecting

stored SKUs, and their respective quantities. These issues have prompted in-depth research, with one key assumption being that the FPA is sufficiently small, rendering the travel distance of pickers negligible^[2-5].

However, diverse warehouse configurations often necessitate spatially extensive FPAs. Industry practices reveal two prevalent spatial designs: (i) Multi-level layout: This design uses vertical space, with the lower level serving as the picking zone and the upper levels reserved for storage. In this configuration, the FPA footprint spans the entire warehouse^[6,7]. (ii) Separated planar layout: Here, the functional areas are distributed across independent racking systems, requiring the FPA to occupy a large operational surface. Empirical examples include a 60-meter-deep cosmetics logistics picking zone^[8] and a 300-meter linear footwear picking channel^[9]. In both designs, the picker travel distance becomes non-negligible and is jointly influenced by storage assignment and picker routing decisions. Studies have demonstrated that optimized storage assignment enhances rack space utilization and reduces inventory management costs by dynamically aligning with stock keeping unit (SKU) turnover rates and access frequencies^[10]. Moreover, picker routing optimization can directly decrease the order processing time^[11].

In warehouse optimization, storage assignment and picker routing exhibit significant interactions. The storage location distribution determines the spatial topology for routing, while routing policies, in turn, shape the evaluation criteria for stor-

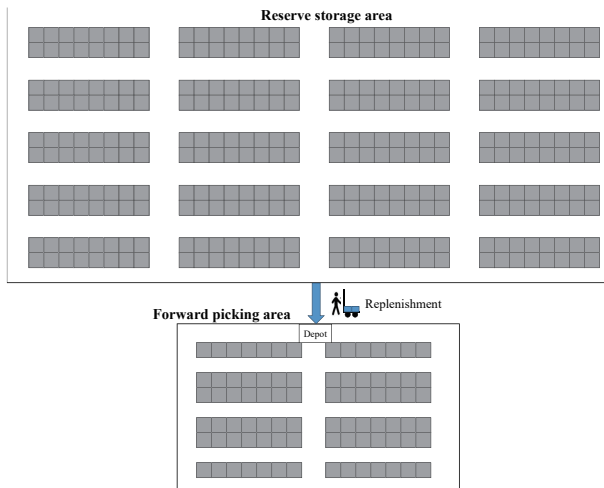


Fig. 1. Schematic layout of a typical RSA and FPA.

age assignment. This bidirectional coupling creates inherent limitations for sequential decision-making approaches. Optimizing one decision while keeping the other fixed typically yields suboptimal solutions, leading to inefficient picker productivity and challenges in meeting delivery time requirements.

This study is motivated by FPA operations in a Chinese distribution center. The FPA operates in waves, with each wave consisting of consecutive replenishment and order picking operations. At the start of a wave, picking orders are confirmed (i.e., the demand for various SKUs is known). Before order picking begins, pickers must replenish the FPA to ensure that all the required SKUs are available. The SKUs are classified into two categories: (i) Already-stored SKUs: They remain in the FPA from previous waves, with their storage locations fixed until the current wave ends; (ii) To-be-stored SKUs: They are not present in the FPA before the wave starts and must be transferred from the RSA to the FPA. Similarly, storage locations are divided into (i) occupied locations that hold already stored SKUs and (ii) open locations that are available for storing to be stored in SKUs. At present, the distribution center adopts a sequential decision-making approach, employing a random storage policy for storing SKUs followed by a return routing policy (detailed in Section 5). This method restricts further improvements in picker efficiency.

Furthermore, most previous studies on storage assignment assume an initially empty warehouse, where all locations are freely available. However, in wave-based FPAs, the assignment of locations to SKUs to be stored represents a dynamic operational-level problem that evolves with each wave. This dynamic nature stems from two primary factors: order variability across waves, where each wave typically involves different SKU combinations, and fluctuating inventory states, where the quantity and composition of already stored SKUs vary at each wave's initiation, consequently changing the set of required SKUs to be stored. A critical operational constraint arises from the fact that already stored SKUs remain fixed in their locations throughout the wave, meaning that new SKUs can only be placed in the limited available open locations, whose quantity and spatial distribution vary across waves. The occupancy state of locations directly influences routing efficiency. High occupancy conditions with few open locations significantly constrain route optimization, particularly for orders requiring minimal SKU storage, whereas greater availability of open locations provides more opportunities for strategic placement to optimize routing paths. Recent research has begun to address warehouse operations subject to storage occupancy constraints. Weidinger and Boysen^[12] investigated multi-location placement of single items in e-commerce fulfillment, developing a bisection-based algorithm to minimize the maximum travel distance. Similarly, Guo et al.^[13] examined storage assignment subject to storage occupancy constraints for predetermined routing strategies and proposed a dynamic programming solution. Li et al.^[14] and Li et al.^[15] addressed task assignment and multi-AGV scheduling in an automated container terminal and an intelligent warehouse, respectively. The former proposed a hierarchical framework that combines reinforcement learning

with path planning, whereas the latter developed a hyper-heuristic based on a multi-armed bandit algorithm. Santos et al.^[16] employed an A* search-based method for path generation in automated warehouses. This study advances the literature in two significant aspects. First, it incorporates real-time order information available at wave start, reflecting actual operational conditions in practice. Second, more importantly, it recognizes and addresses the inherent interdependence between storage assignment and picking routing through an integrated optimization approach, moving beyond the limitations of sequential decision-making prevalent in current implementations.

This paper investigates the joint optimization of storage assignment and picking routing in FPAs while accounting for storage occupancy constraints. Specifically, three research questions are considered: First, how should to-be-stored SKUs be allocated to available open locations, and what picking route should be followed to minimize the total processing time for the given orders? Second, what performance improvements can the proposed algorithm achieve? Third, how do storage occupancy constraints affect system performance in this joint optimization framework?

In response to these three research questions, we make three contributions to the literature. First, we develop a novel integer programming model for the joint optimization problem subject to storage occupancy constraints and design an efficient variable neighborhood search (VNS) algorithm. The proposed solution method demonstrates strong computational performance. Second, numerical experiments reveal that, compared with conventional sequential decision-making methods, our joint optimization approach reduces the total processing time by up to 17.5%. Third, the results show that explicitly considering storage occupancy constraints yields significant benefits, with our algorithm achieving up to a 7.8% greater total processing time reduction than approaches that neglect these constraints.

The remainder of this paper is organized as follows. Section 2 describes the problem setting and presents the mathematical model. The proposed VNS algorithm is developed in Section 3. Section 4 reports the numerical experiments, while a real-world case study is presented in Section 5. Several extensions are discussed in Section 6, and concluding remarks are provided in Section 7.

2 Problem description

This section describes the manual warehouse configuration considered in this study. Fig. 2 represents the layout of the FPA. The FPA has $|A|$ picking aisles and three cross aisles, with the middle cross aisle dividing racks into two identical blocks. $A = \{1, 2, \dots, |A|\}$ is the set of picking aisles. Pickers start and end at the depot. The distances between adjacent aisles and adjacent storage locations are w_a and w_b , respectively. The middle cross-aisle has a width of w_c , and each side cross-aisle is $w_c/2$ in width. For clarity, racks adjacent to aisles l and $l+1$ (where l is odd) belong to row $(l+1)/2$.

Define S_N as the set of to-be-stored SKUs replenished from the RSA, S_A as the set of already stored SKUs in the FPA, and $S = S_N \cup S_A$ as the total SKU set. The set of stor-

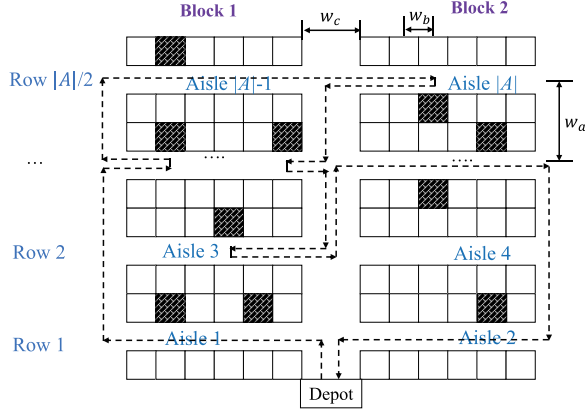


Fig. 2. Layout of the FPA. The black square represents the storage locations to be visited, and the dotted line represents the picker route.

age locations L comprises occupied locations L_A and open locations L_E . Let r_k denote the location of the already stored SKU k . Given a set of orders O , let γ_o represent the number of SKUs in the order $o \in O$. The number of orders is n . This paper addresses the joint optimization of storage assignment and picker routing subject to storage occupancy constraints. The model simultaneously determines the assignment of to-be-stored SKUs to open locations and the sequence of visiting storage locations to fulfill the demand of each order. The objective is to minimize the total processing time for all orders. The processing time for an order comprises a fixed setup time for order preparation, a picking time for SKU retrieval at each location, and a travel time for movement between locations (including travel to and from the depot). We define the setup time per order as t^s , the picking time per SKU as t^p , and the travel time per length unit as t' .

The following assumptions are made:

(i) Our model focuses solely on minimizing the order-picking processing time, excluding replenishment costs. This focus is justified by the fact that order picking operates under tighter time windows and demands higher labor intensity than replenishment activities do.

(ii) We assume that pickers can reach all items regardless of the rack's height, so the vertical movement up or down the shelves to different levels for picking is ignored. This assumption is relaxed in Section 6.1, where height variations are incorporated.

(iii) It is assumed that replenishment and order picking operations are scheduled in separate, non-overlapping time windows rather than being performed concurrently. This sequential mode prevents stockouts during picking by ensuring that inventory is replenished beforehand; it also avoids congestion and resource conflicts between equipment and personnel.

(iv) Order information is known before the wave starts under wave picking.

(v) Pickers walk along the centerline of aisles; turning distance errors are negligible.

The travel distance d_{ij} between locations i (block b_i , row l_i) and j (block b_j , row l_j) depends on their relative positions. Let p be the length of a picking aisle. The formulas of d_{ij} are given in Table 1.

Decision variables: $x_{oij} = 1$ if the order o is completed by

Table 1. Distance calculation formulas for different cases.

Cases	d_{ij}
$b_i = b_j, l_i = l_j$	$w_b i - j $
$b_i \neq b_j, l_i = l_j$	$w_b i + j + w_c$
$b_i = b_j, l_i \neq l_j$	$w_a l_i - l_j + \min(w_b(i + j) + w_c, -w_b(i + j) + w_c + 2p)$
$b_i \neq b_j, l_i \neq l_j$	$w_a l_i - l_j + \min(w_b(i + j) + w_c, w_b i - j + 2w_c + p)$

traveling from location i to j ; otherwise, $x_{oij} = 0$. $y_{ki} = 1$ if to-be-stored SKU k is assigned to open location i ; otherwise, $y_{ki} = 0$.

The joint optimization model for storage assignment and picker routing, considering the storage occupancy constraints, is as follows:

$$\min \sum_{o \in O} \sum_{i \in L} \sum_{j \in L \setminus \{i\}} t' d_{ij} x_{oij} + n t^s + \sum_{o \in O} t^p \gamma_o \quad (1)$$

$$\text{s.t.} \quad \sum_{i \in L} y_{ki} = 1 \quad \forall k \in S; \quad (2)$$

$$\sum_{k \in S} y_{ki} \leq 1 \quad \forall i \in L; \quad (3)$$

$$\sum_{k \in \Omega_o} y_{ki} = \sum_{j \in L \cup \{0\}} x_{oij} \quad \forall i \in L \cup \{0\}, o \in O; \quad (4)$$

$$\sum_{k \in \Omega_o} y_{kj} = \sum_{i \in L \cup \{0\}} x_{oij} \quad \forall j \in L \cup \{0\}, o \in O; \quad (5)$$

$$\sum_{i, j \in P} x_{oij} \leq |P| - 1 \quad \forall P \subseteq L, 1 < P < |L|, o \in O; \quad (6)$$

$$y_{k, r_k} = 1 \quad \forall k \in S_A; \quad (7)$$

$$x_{oij} \in \{0, 1\} \quad \forall i, j \in L \cup \{0\}, o \in O; \quad (8)$$

$$y_{ki} \in \{0, 1\} \quad \forall k \in S, i \in L. \quad (9)$$

Objective (1) minimizes the total processing time for all orders. Constraints (2) and (3) ensure that each to-be-stored SKU is assigned to one open location and that each open location holds at most one SKU, respectively. Constraints (4) and (5) link the assignment and routing variables. Constraint (6) eliminates subtours. Constraint (7) fixes already stored SKUs to their occupied locations. Constraints (8) and (9) define the variable domains.

Silva et al.^[17] proved that the joint optimization problem without storage occupancy constraints (P1) in a single-block warehouse is NP-hard. Consider an instance where Block 2 contains no pickable SKUs and no orders contain SKUs from S_A . P1 reduces to our problem, establishing its NP-hardness. Thus, finding optimal solutions for large-scale instances in reasonable time is intractable. We design a variable neighborhood search (VNS) algorithm to efficiently obtain near-optimal solutions.

3 VNS algorithm

The VNS algorithm was first introduced by Ref. [18] and has

been widely used to solve various combinatorial optimization problems, such as the vehicle routing problem^[19], the flexible job-shop scheduling problem^[20], and the personnel scheduling problem^[21]. The core idea of VNS is to expand the search scope, after constructing an initial solution, by using multiple different neighborhoods to find a local optimum within the current search space. If local search fails to find a better solution, a perturbation operator is used to transition from one neighborhood structure to a new one, thereby broadening the search range, followed by another local search. The details of the main features are described below.

Initial solution: To generate an initial feasible solution, we decompose the problem into two sequential subproblems: (i) storage assignment and (ii) picker routing. For the storage assignment subproblem, the open storage locations nearest to the depot are first selected. We consider two storage assignment rules. The first prioritizes SKUs with a high turnover frequency, assigning them to storage locations that minimize the travel distance to the depot. The second rule implements a correlated storage policy, where the compatibility between SKUs and storage locations is evaluated via the method established by Pang and Chan^[22]. This assignment problem is formulated to maximize the total compatibility value and solved to optimality via the Hungarian algorithm.

Neighborhood structures: Once a storage assignment decision is given, the algorithm proposed by Roodbergen and De Koster^[23] can be used to derive the optimal picker route. Therefore, this paper designs three neighborhood structures for storage assignment as follows:

Neighborhood Structure 1: replacing one open storage location with another.

Given the set S_N^* of chosen open storage locations in the current solution and that the to-be-stored SKU k is stored at open storage location i ($y_{ki} = 1$), Neighborhood Structure 1 randomly selects an open storage location j from $S_N \setminus S_N^*$ and sets $y_{kj} = 1$. The worst-case size of this neighborhood structure is $\frac{|S_N|^2}{4}$.

Neighborhood Structure 2: swapping SKUs between two open storage locations.

Given an open storage location i containing to-be-stored SKU k ($y_{ki} = 1$) and an open storage location j containing to-be-stored SKU h ($y_{hj} = 1$), Neighborhood Structure 2 assigns k to location j and h to location i , i.e., sets $y_{kj} = 1$ and $y_{hi} = 1$. The worst-case size of this neighborhood structure is $|S_N|^2$.

Neighborhood Structure 3: swapping SKUs between two picking aisles.

Find two picking aisles, then swap all to-be-stored SKUs between the two aisles. Neighborhood Structure 3 requires that the sets of chosen open storage locations within the two swapped aisles be identical. In other words, let the binary variable $e_{ak} = 1$ indicate that SKU k is stored in picking aisle a . Given that SKU k is stored in picking aisle a ($e_{ak} = 1$) and SKU h is stored in picking aisle b ($e_{bh} = 1$) in the current solution, Neighborhood Structure 3 swaps the SKUs k and h such that $e_{ah} = e_{bk} = 1$. The worst-case size of this neighborhood structure is $|A|^2$.

Perturbation: The purpose of perturbation is to prevent the algorithm from becoming trapped in local optima. In this

paper, perturbation is applied to the current best solution by randomly swapping SKUs between two open selected storage locations.

Solution acceptance and termination criterion: This paper adopts an acceptance criterion analogous to that of the simulated annealing algorithm. Let z denote the current solution and z^* denote a new solution. The objective function value of a feasible solution z is $f(z)$. When $f(z^*) < f(z)$, the current solution z is replaced by the new solution z^* . Otherwise, z^* replaces z with a probability of $\exp(-(f(z^*) - f(z))/T)$, where T represents the temperature. The temperature decreases linearly at a cooling rate of 0.95, with the initial temperature set to 100. The proposed VNS is designed to terminate once a predetermined number of iterations, numIterMax, has been reached.

Algorithm procedure: The procedure of the VNS is as follows: First, an initial solution is obtained and designated as the current solution z . Subsequently, Neighborhood Structures 1, 2, and 3 are employed sequentially for searching until no improvement occurs. During each computation step, the current best solution is perturbed to generate a new solution. If the new solution is superior to the current solution, the current solution is replaced, and the search continues within the current neighborhood. Otherwise, the acceptance criterion of the simulated annealing is used to determine whether to accept the new solution. If accepted, the search continues within the same neighborhood; if rejected, the algorithm switches to the next neighborhood structure and continues computation. This process repeats until the stopping criterion is satisfied, at which point the algorithm stops.

4 Numerical analysis

This section validates the performance of the VNS algorithm through numerical experiments. The overall code for the VNS algorithm is written in MATLAB R2021a. The experimental hardware environment consists of a PC with an Intel Core i5-8250U CPU at 1.6 GHz and 8 GB of RAM running 64-bit Windows 10. The JSAR model is solved via CPLEX 13, with a runtime limit of 3600 s for CPLEX.

4.1 Instance generation

This section uses two types of test instances: One type consists of small-scale instances where CPLEX can find the optimal solution (4 picking aisles, each with 10 storage locations); the other type consists of large-scale instances (20 picking aisles, each with 40 storage locations). Furthermore, the warehouse shape parameters originate from a case company: $w_a = 4$, $w_b = 1$, $w_c = 5$, $t^s = 10$, $t^t = 1$, and $t^p = 5$. If CPLEX exceeds the 3600-second runtime limit, the solution obtained by CPLEX at the end of its execution is output. Considering the cross-combinations of $n = \{50, 100, 200\}$ (number of orders), $\alpha = \{0.2, 0.5, 0.8\}$ (the ratio of empty locations to the total number of locations), and $\beta = \{10, 15, 20\}$ (the maximum number of SKUs per order), a total of 27 groups of instances at different scales are generated. For each group, 10 random instances are generated, and the average travel distance is taken. Let $f(z_{\text{CPLEX}})$ denote the solution obtained by CPLEX, $f(z_{\text{ini}})$ the initial solution for the VNS, and $f(z_{\text{VNS}})$

the final solution obtained by the VNS.

The performance of the VNS is indeed significantly influenced by the termination condition (i.e., the maximum number of iterations, numIterMax). To determine an appropriate value for numIterMax, we conduct a parameter tuning analysis on newly generated small-scale instances with $n = \{50, 100\}$, $\alpha = \{0.2, 0.5\}$, and $\beta = \{10, 15\}$. For each parameter combination, five instances are generated. We evaluate the performance of VNS under different numIterMax values by comparing the average percentage deviation from the best-known solutions obtained by CPLEX (denoted as “gap”) and

the average computational time (denoted as “sec”). The results are summarized in Table 2. Our analysis shows that increasing numIterMax beyond a moderate value leads to only negligible improvements in the solution quality while increasing the computational time. Therefore, to balance solution accuracy and efficiency, we select numIterMax = 100 as a robust setting for the VNS termination criterion.

4.2 Experimental results

The results for the small-scale instances are presented in Table 3. Here, g_1 represents the percentage gap between the

Table 2. Tuning results.

	numIterMax						
	40	60	80	100	120	140	160
Gap (%)	1.4	1.0	0.8	0.5	0.5	0.5	0.5
Sec (s)	1.7	2.5	3.6	3.9	4.7	5.9	7.3

Table 3. Comparison of CPLEX and VNS on small-scale instances.

$n/\alpha/\beta$	CPLEX		VNS			g_1	g_2
	$f(z_{\text{CPLEX}})$	CPU time (s)	$f(z_{\text{ini}})$	$f(z_{\text{VNS}})$	CPU time (s)		
50/0.2/10	1469.4	3.6	1515.1	1469.4	4.3	0%	3.1%
50/0.2/15	1523.5	6.7	1584.5	1523.5	3.7	0%	4.0%
50/0.2/20	1839.4	9.5	1889.1	1839.4	3.4	0%	2.7%
50/0.5/10	1126.5	10.6	1161.7	1126.5	3.4	0%	3.1%
50/0.5/15	1382.6	15.5	1423.9	1382.6	5.4	0%	3.0%
50/0.5/20	1554.2	17.4	1616.4	1554.2	5.3	0%	4.0%
50/0.8/10	946.7	78.2	946.7	946.7	3.4	0%	0%
50/0.8/15	1026.8	85.6	1066.0	1026.8	4.8	0%	3.8%
50/0.8/20	1472.3	55.3	1511.7	1472.3	4.2	0%	2.7%
100/0.2/10	2304.1	16.2	2367.6	2304.1	4.3	0%	2.8%
100/0.2/15	2663.5	16.8	2783.7	2663.5	5.1	0%	4.5%
100/0.2/20	3350.2	17.0	3438.6	3350.2	4.6	0%	2.6%
100/0.5/10	1837.1	64.6	1912.4	1837.1	3.7	0%	4.1%
100/0.5/15	2484.3	84.5	2553.8	2484.3	4.8	0%	2.8%
100/0.5/20	2937.9	131.6	2937.9	2937.9	5.5	0%	0%
100/0.8/10	1801.0	281.1	1883.7	1801.0	5.5	0%	4.6%
100/0.8/15	1925.6	297.0	1925.6	1925.6	3.7	0%	0%
100/0.8/20	2215.7	304.7	2290.6	2215.7	3.4	0%	3.4%
200/0.2/10	5787.5	38.6	5950.1	5787.5	5.3	0%	2.8%
200/0.2/15	5937.6	40.9	6189.7	5937.6	4.8	0%	4.2%
200/0.2/20	6752.2	43.6	7004.3	6752.2	4.0	0%	3.7%
200/0.5/10	4080.1	90.7	4244.8	4080.1	3.8	0%	4.0%
200/0.5/15	4254.7	93.0	4367.9	4254.7	4.1	0%	2.7%
200/0.5/20	4886.7	92.7	5087.9	4886.7	4.6	0%	4.1%
200/0.8/10	3608.9	419.6	3720.3	3608.9	5.2	0%	3.1%
200/0.8/15	3780.9	396.2	3860.3	3780.9	4.0	0%	2.1%
200/0.8/20	4664.8	411.7	4833.8	4664.8	6.5	0%	3.6%
Avg	2874.6	115.7	2965.5	2847.6	4.5	0%	3.0%

solution found by the VNS and the optimal solution found by CPLEX, calculated via the following formula:

$$g_1 = \frac{f(z_{\text{VNS}}) - f(z_{\text{CPLEX}})}{f(z_{\text{VNS}})} \times 100\%.$$

g_2 represents the percentage gap between the initial solution and the final solution obtained by the VNS, calculated via the following formula:

$$g_2 = \frac{f(z_{\text{ini}}) - f(z_{\text{VNS}})}{f(z_{\text{ini}})} \times 100\%.$$

The numerical results indicate that both CPLEX and the VNS can find optimal solutions, and the computation time of the VNS is consistently less than that of CPLEX across all instances.

Table 4 presents the results of large-scale instances. CPLEX found the optimal solution in 44% of the instances. In

these cases, the gap between the solutions obtained by the VNS and the optimal solutions is less than 1.7%. For the remaining instances, CPLEX can find only a feasible solution within the limited time frame. In contrast, as shown in g_2 column, the VNS improves the solution quality by an average of 7.5%. Considering all instances, the average travel distance obtained by the VNS is 3.0% lower than that of CPLEX. In terms of computational efficiency, the average computation time of the VNS is 24.7 s, which is significantly less than 2727.3 s consumed by CPLEX. Additionally, as the number of empty storage locations increases, the travel distance of the pickers for completing orders gradually decreases, which is attributed to the increased flexibility in choices offered by the greater number of empty storage positions. When the number of picking aisles and the number of SKUs in the order increase, the travel distance obtained by all algorithms also increases. This conclusion is relatively intuit-

Table 4. Comparison of CPLEX and VNS on large-scale instances.

$n/\alpha/\beta$	CPLEX		VNS		g_1	g_2
	$f(z_{\text{CPLEX}})$	CPU time (s)	$f(z_{\text{ini}})$	$f(z_{\text{VNS}})$		
50/0.2/10	21864.1	591.1	23149.0	22107.3	1.1%	4.5%
50/0.2/15	23643.1	624.7	24950.2	24052.0	1.7%	3.6%
50/0.2/20	24457.7	652.1	26805.1	24982.3	2.1%	6.8%
50/0.5/10	17232.5	1018.6	18204.9	17494.9	1.5%	3.9%
50/0.5/15	32848.9	1089.7	36269.3	33113.8	0.8%	8.7%
50/0.5/20	22214.6	934.6	25028.6	22575.8	1.6%	9.8%
50/0.8/10	17585.6	2473.2	20960.1	17962.8	2.1%	14.3%
50/0.8/15	28556.2	2618.7	31025.9	29288.4	2.5%	5.6%
50/0.8/20	22185.4	2738.6	24045.9	22242.5	0.3%	7.5%
100/0.2/10	50132.7	1767.3	53516.7	50947.9	1.6%	4.8%
100/0.2/15	71095.0	2282.7	79589.8	73143.0	2.8%	8.1%
100/0.2/20	81379.0	2851.3	89272.1	83380.2	2.4%	6.6%
100/0.5/10	52264.9	3600	56769.8	52114.7	-0.3%	8.2%
100/0.5/15	64229.1	3600	68374.0	64339.9	0.2%	5.9%
100/0.5/20	52651.0	3600	55707.8	51975.3	-1.3%	6.7%
100/0.8/10	30495.9	3600	33131.6	29752.1	-2.5%	10.2%
100/0.8/15	47748.4	3600	49720.5	45345.1	-5.3%	8.8%
100/0.8/20	54058.1	3600	55938.7	51631.4	-4.7%	7.7%
200/0.2/10	83549.6	3600	85805.6	78597.9	-6.3%	8.4%
200/0.2/15	111138.3	3600	117482.2	106556.4	-4.3%	9.3%
200/0.2/20	112832.1	3600	116514.2	103231.6	-9.3%	11.4%
200/0.5/10	61585.1	3600	62652.1	57076.1	-7.9%	8.9%
200/0.5/15	94549.4	3600	91891.2	85642.6	-10.4%	6.8%
200/0.5/20	133522.6	3600	129891.3	119110.3	-12.1%	8.3%
200/0.8/10	69568.5	3600	69946.5	63301.6	-9.9%	9.5%
200/0.8/15	96122.5	3600	88020.9	84764.1	-13.4%	3.7%
200/0.8/20	90421.7	3600	83260.6	78764.5	-14.8%	5.4%
Avg	58071.6	2727.3	59923.1	55314.6	-3%	7.5%

ive, as in larger warehouses, the travel distance of pickers naturally increases.

4.3 Comparative analysis with the literature on sequential decision-making

This section presents a comparative experimental analysis between sequential approaches and the integrated approach (e.g., the proposed VNS). Two sequential approaches are described as follows:

(I) Given a route, optimized storage (GROS): Here, the picker route is predetermined before the storage assignment decision is optimized. Three commonly used routing policies are evaluated: S-shaped routing, return routing, and midpoint routing. For each policy, the algorithm proposed by Guo et al.^[13] is used to determine the optimal storage assignment for to-be-stored SKUs. The most effective solution across all three routing policies is selected.

(II) Given storage, the optimized routing approach (GSOR): In this approach, the storage assignment is fixed before routing optimization. Four storage assignment policies are assessed, including random storage (RND), full-turnover storage (FT), and the minimum delay algorithm (MDA)^[24]. RND randomly assigns a to-be-stored SKU to an open location. The FT places SKUs in the closest open location on the basis of the demand frequency. Following Ref. [24], the MDA first generates SKU sequences to minimize the linear travel distance, then allocates SKUs to locations accordingly. The algorithm proposed by Roodbergen and De Koster^[23] is subsequently used to solve the picker routing problem. The best solution among the four storage assignment policies is reported.

Specifically, we consider large-scale instances with 100 orders. The default parameters are set as $n = 50$ and $\beta = 10$. To comprehensively evaluate parameter impacts on warehouse performance, the following value sets are tested: $n = \{20, 50, 100, 150\}$ and $\beta = \{5, 10, 20, 30\}$. Each experimental group consists of 100 instances, and the average travel distance is calculated. The percentage gap (g_{GROS}) represents the solution difference between GROS and VNS, whereas g_{GSOR} represents the difference between GSOR and VNS. The results are shown in Figs. 3 and 4.

The following observations are made: (i) As the number of orders increases, the advantage of the integrated approach becomes more pronounced. Compared with GROS, the total processing time reduction ranges from 4.5% to 17.5%. Relative to GSOR, improvement spans 2.6% to 12.6%. Optimizing picker routes yields better results than optimizing storage assignment. This is because optimizing storage assignment decisions under a predetermined routing policy merely achieves the optimal travel distance for that specific routing strategy at best. Compared with the overall optimal strategy, significant room for improvement remains. (ii) When the number of SKUs per order is large ($\beta = 30$), the sequential approach maintains competitive performance. This occurs because SKUs are distributed throughout the entire FPA, rendering optimal routes nearly identical to S-shaped routing. As a result, the travel distances under different policies are almost identical. Additionally, the probability of encountering stored SKUs in the last picking aisle is high, meaning that the

last aisle must be visited regardless of storage assignment optimization. This leads to similar travel distances across the aisles. Consequently, the total processing times obtained by all three routing policies are nearly identical, with negligible gaps.

4.4 Impact of storage occupancy constraints on warehouse performance

We investigate the value of incorporating the storage occupancy constraints, under which certain locations are unavailable due to pre-stored items. We consider the following benchmark algorithm. Storage occupancy constraints are ignored, and all storage locations are assumed to be available. Let rr_k be the storage location assigned to SKU k . This solution may be infeasible if rr_k is occupied. Infeasibly assigned SKUs need to be reassigned to minimize the total relocation distance from their assigned locations to all feasible open locations. This problem can be solved optimally in polynomial time as a linear assignment problem via the Hungarian method. After assignment, the optimal picker routes are determined via the algorithm proposed by Roodbergen and De Koster^[23].

Fig. 5 compares the results from the benchmark algorithm and our VNS. Compared with the benchmark, the VNS reduces the total processing time by 1.8%–7.8%. This reduction increases with the proportion of available storage locations, as greater availability enhances assignment flexibility and routing optimization. These findings underscore the importance of incorporating storage occupancy constraints in decision-making. Therefore, warehouse managers should base decisions on actual open storage location positions and quantities.



Fig. 3. Impact of the order quantity on the improvement of joint optimization.

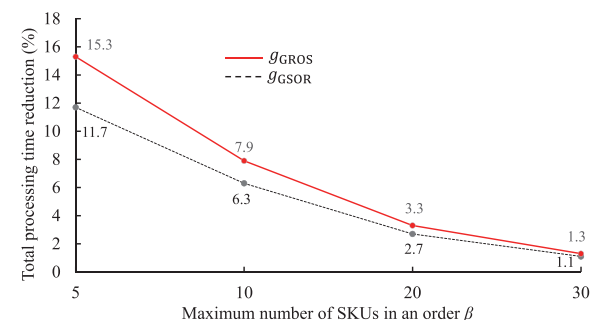


Fig. 4. Impact of the maximum number of SKUs per order on the improvement of joint optimization.



Fig. 5. The impact of storage location occupancy constraints on warehouse efficiency.

4.5 Robustness of the model against travel time deviations

In the problem under our study, the warehouse layout features ample aisle width, allowing at least two pickers to pass each other or operate simultaneously with minimal interference. For this reason, route conflicts are not considered in our model. However, in warehouses with narrower aisles, route conflicts and congestion can indeed become critical issues that affect picking efficiency. To address concerns regarding potential conflicts or congestion when multiple pickers operate simultaneously, we conducted additional experiments under a multi-picker setting. In these tests, we assume that there are 40 orders and five pickers and that the orders are assigned to pickers via the seed algorithm^[25]. We use our VNS to determine the picker route for each picker, and the objective is to minimize the makespan, which is defined as the maximum processing time among all pickers.

Since congestion essentially increases the expected travel time between locations, we perform robustness tests to evaluate whether our algorithm remains effective when travel times are perturbed. Specifically, we identify shared route segments across pickers and applied a 5%–50% increase to the travel times of those segments to simulate congestion. The results presented in Fig. 6 show that the makespan based on the planned routes deviates by less than 1.6%, even when actual travel times exceed estimates by up to 50%. This demonstrates the robustness of our algorithm against unanticipated delays, including those caused by route conflicts or congestion.

5 Case study

This section tests the proposed algorithm via real-world data

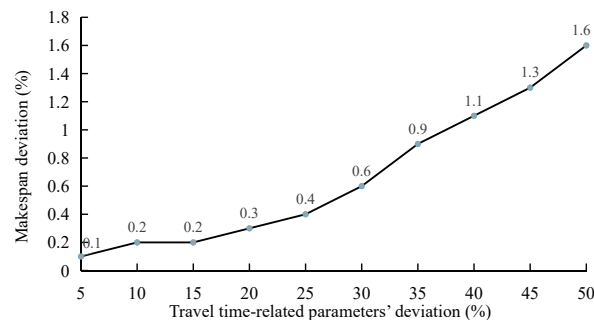


Fig. 6. Robustness test of travel time-related parameters for the proposed algorithm.

from a third-party logistics (3PL) provider (Company A), which was established in 2015 and specializes in warehousing and distribution. The company operates a distribution center that processes thousands of orders daily. Its FPA contains four picking aisles, each with 40 storage locations. Adjacent aisles are spaced 4 m apart, whereas adjacent storage locations are 1 m apart.

The current operational procedure of Company A involves cyclic replenishment and picking within waves. After a wave begins, pickers replenish the FPA from storage (taking approximately one hour) to ensure item availability. Order picking must then be completed within a short timeframe. The company handles multiple waves daily. Currently, the FPA uses a random storage policy for storing SKUs and a return routing policy (entering and exiting aisles from the same end), often resulting in unnecessarily long travel distances and shipment delays. Company A's dataset includes 2456 orders spanning eight waves during May 17–18, 2022. These orders involve 52 SKU types, averaging 10.7 items per order. The proposed VNS was embedded into the warehouse management system. Fig. 7 presents the computational results for the total processing time, showing that VNS significantly outperforms the company's current method across all waves, achieving an average objective reduction of approximately 9.6%.

6 Extensions

In this section, we investigate three extensions for various complex scenarios, including multifloor layouts, mixed storage systems, and dynamic obstacles. These extensions can all be integrated into our proposed algorithm.

6.1 Multifloor layouts

In our basic model, we do not consider vertical movement up or down the shelves. However, to extend our algorithm to multi-floor settings involving elevators or forklifts for picking SKUs at various heights, we can redefine the travel distance between any two locations by incorporating vertical movement. For locations i (block b_i , row l_i , height h_i) and j (block b_j , row l_j , height h_j), the horizontal distance d_{ij} is computed as in Table 1. The vertical distance q_{ij} is defined as follows: If locations i and j are on the same rack, $q_{ij} = |h_j - h_i|$; otherwise, $q_{ij} = h_j + h_i$ (e.g., accounting for vertical travel to and from the bottom). The total travel time between locations i and j becomes $\hat{t}_{ij} = d_{ij}t^h + q_{ij}t^v$, where t^h is

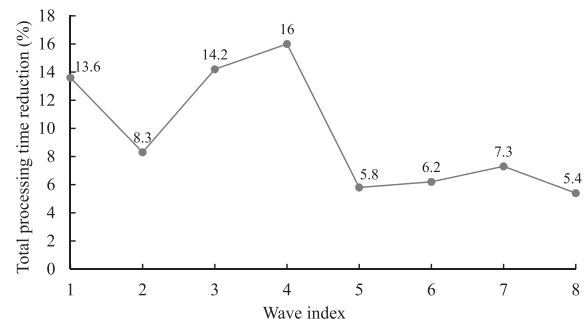


Fig. 7. Performance comparison between the proposed algorithm and the case company's method on real data.

the horizontal travel time per length unit of pickers and t^v is the vertical travel time per length unit of elevators or forklifts that are used for picking SKUs. Since this modification affects only the input distance matrix, our algorithm remains unchanged and directly applicable.

6.2 Mixed storage systems

In mixed storage systems, as implemented by companies such as Amazon, SKUs are scattered across multiple racks. A key challenge in this environment is that each SKU may have multiple alternative storage locations, and the selection of which specific location to use for picking must be determined. To address this problem, we modify our VNS algorithm as follows: Before determining the optimal picking route via the algorithm in Ref. [23], we first preselect the nearest available storage location for each SKU that is required by an order until its demand is fulfilled.

6.3 Dynamic obstacles

In cases where certain nodes become dynamically inaccessible (e.g., owing to temporarily placed obstacles, ongoing replenishment, or safety constraints that change over time), we employ a rolling horizon framework, which divides the planning period into multiple time segments. At the start of each segment, the travel distance matrix is updated in real time to reflect the current accessibility conditions. If a node becomes blocked, all travel distances involving that node are set to infinity for the duration of that segment, effectively removing it from the feasible routing network during that interval. Since this adjustment only modifies the input distance matrix, our algorithm remains applicable and responsive to dynamic environmental changes.

7 Conclusions

This paper addresses the joint optimization of storage assignment and picker routing in the forward area of manual warehouse systems while considering storage occupancy constraints. We first develop a mathematical model and design a VNS algorithm. Numerical experiments demonstrate that the VNS can obtain optimal solutions for small-scale instances in a short time and yield high-quality solutions for large-scale instances within a reasonable timeframe. Compared with CPLEX, the proposed algorithm has significant advantages in both solution quality and computation time. Finally, using actual order data from a third-party logistics provider, we verify that the algorithm significantly improves enterprise order-picking processes.

The current study assumes a single depot, whereas warehouses often feature multiple depots to accommodate simultaneous truck loading/unloading and avoid single-depot bottlenecks. Future research could extend joint optimization to multi-depot scenarios. Additionally, this study focuses on warehouses with parallel picking aisles. Warehouse layouts vary (e.g., fishbone, U-shape, butterfly), necessitating adjusted distance formulas between locations and optimized shelf placement angles. The proposed methodology can be extended to these configurations.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (72401281, 72401271, and 72471222), the National Social Science Fund of China (24CGL045), the Anhui Provincial Natural Science Foundation Youth Project (2508085QG263), the Fundamental Research Funds for the Central Universities (WK2040000086, WK2040000080, and WK2040000099), and the China Postdoctoral Science Foundation-funded Project (2024M763166).

Conflict of interest

The authors declare that they have no conflict of interest.

Biographies

Ran Chen is currently an Associate Professor at the School of Management, University of Science and Technology of China (USTC). He received his Ph.D. degree in Management Science and Engineering from USTC in 2021. His research mainly focuses on intelligent logistics and warehouse operations management.

Jingjing Yang is a postdoctoral fellow at the University of Science and Technology of China (USTC). She received her Ph.D. degree in Management Science and Engineering from USTC in 2023. Her research mainly focuses on intelligent optimization and supply chain management.

References

- [1] Boysen N, De Koster R. 50 years of warehousing research—An operations research perspective. *European Journal of Operational Research*, **2025**, 320 (3): 449–464.
- [2] Walter R, Boysen N, Scholl A. The discrete forward-reserve problem—Allocating space, selecting products, and area sizing in forward order picking. *European Journal of Operational Research*, **2013**, 229 (3): 585–594.
- [3] Jiang M, Huang G Q. Intralogistics synchronization in robotic forward-reserve warehouses for e-commerce last-mile delivery. *Transportation Research Part E: Logistics and Transportation Review*, **2022**, 158: 102619.
- [4] Leung E K H. Total fulfillment management: principles, practices and use cases. *Transportation Research Part E: Logistics and Transportation Review*, **2025**, 194: 103888.
- [5] Cordeau J F, Legato P, Mazza R M. Integrating storage allocation with manual order picking and replenishment operations in a distribution centre. *International Journal of Production Research*, **2025**, 63 (7): 2694–2710.
- [6] Thomas L M, Meller R D. Developing design guidelines for a case-picking warehouse. *International Journal of Production Economics*, **2015**, 170: 741–762.
- [7] Liu J, Liao H, White J. Stochastic analysis of an automated storage and retrieval system with multiple in-the-aisle pick positions. *Naval Research Logistics*, **2021**, 68 (4): 454–470.
- [8] Velez-Gallego M C, Smith A E. Optimization of a fast-pick area in a cosmetics distribution center. In: 15th IMHRC Proceedings. Savannah, GA, USA: Georgia Southern University, **2018**: 28.
- [9] Bahrami B, Aghezzaf E H, Limère V. Enhancing the order picking process through a new storage assignment strategy in forward-reserve area. *International Journal of Production Research*, **2019**, 57 (21): 6593–6614.
- [10] Zhang G, Shang X, Alawneh F, et al. Integrated production planning and warehouse storage assignment problem: An IoT assisted case. *International Journal of Production Economics*, **2021**, 234: 108058.
- [11] Haouassi M, Kergosien Y, Mendoza J E, et al. The picker routing

- problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, **2025**, 63 (4): 1304–1325.
- [12] Weidinger F, Boysen N. Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, **2018**, 52 (6): 1412–1427.
- [13] Guo X, Chen R, Du S, Yu Y. Storage assignment for newly arrived items in forward picking areas with limited open locations. *Transportation Research Part E: Logistics and Transportation Review*, **2021**, 151: 102359.
- [14] Li K, Liu T, Kumar P, et al. A reinforcement learning-based hyper-heuristic for AGV task assignment and route planning in parts-to-picker warehouses. *Transportation Research Part E: Logistics and Transportation Review*, **2024**, 185: 103518.
- [15] Li S, Fan L, Jia S. A hierarchical solution framework for dynamic and conflict-free AGV scheduling in an automated container terminal. *Transportation Research Part C: Emerging Technologies*, **2024**, 165: 104724.
- [16] Santos J, Rebelo P, Rocha L, et al. A* based routing and scheduling modules for multiple AGVs in an industrial scenario. *Robotics*, **2021**, 10 (2): 72.
- [17] Silva A, Coelho L C, Darvish M, et al. Integrating storage location and order picking problems in warehouse planning. *Transportation Research Part E: Logistics and Transportation Review*, **2020**, 140: 102003.
- [18] Mladenović N, Hansen P. Variable neighborhood search. *Computers & Operations Research*, **1997**, 24 (11): 1097–1100.
- [19] Su Y, Zhang S, Zhang C. A lightweight genetic algorithm with variable neighborhood search for multi-depot vehicle routing problem with time windows. *Applied Soft Computing*, **2024**, 161: 111789.
- [20] Xu Y, Zhang M, Yang M, et al. Hybrid quantum particle swarm optimization and variable neighborhood search for flexible job-shop scheduling problem. *Journal of Manufacturing Systems*, **2024**, 73: 334–348.
- [21] Chen P S, Huang W T, Chen G Y H, et al. Constructing modified variable neighborhood search approaches to solve a nurse scheduling problem. *International Journal of Production Research*, **2024**, 62 (19): 7186–7204.
- [22] Pang K W, Chan H L. Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, **2017**, 55 (14): 4035–4052.
- [23] Roodbergen K J, De Koster R. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, **2001**, 133 (1): 32–43.
- [24] Wutthisirisart P, Noble J S, Chang C A. A two-phased heuristic for relation-based item location. *Computers & Industrial Engineering*, **2015**, 82: 94–102.
- [25] Pardo E G, Gil-Borrás S, Alonso-Ayuso A, et al. Order batching problems: Taxonomy and literature review. *European Journal of Operational Research*, **2024**, 313 (1): 1–24.