Hindawi
International Journal of Intelligent Systems
Volume 2024, Article ID 4456261, 18 pages
https://doi.org/10.1155/2024/4456261

Research Article

A Genetic Algorithm with Lower Neighborhood Search for the Three-Dimensional Multiorder Open-Size Rectangular Packing Problem

Jianglong Yang,[1,2] **Huwei Liu**,[3] **Kaibo Liang**,[4] **Man Shan**,[1] **Lingjie Kong**,[1] **and Li Zhou**[1,2]

[1]*Beijing Wuzi University, School of Information, Beijing 101149, China*
[2]*Beijing Intelligent Logistics System Collaborative Innovation Center, Beijing 101149, China*
[3]*Beijing Institute of Economics and Management, School of Airport Economics and Management, Beijing 100102, China*
[4]*Capital University of Economics and Business, School of Management and Engineering, Beijing 100070, China*

Correspondence should be addressed to Kaibo Liang; liangkaibo2955@gmail.com

Received 1 August 2023; Revised 6 April 2024; Accepted 24 April 2024; Published 15 May 2024

Academic Editor: Vasudevan Rajamohan

This paper addresses the multiorder open-dimension three-dimensional rectangular packing problem (3D-MOSB-ODRPP), which involves packing rectangular items from multiple orders into a single, size-adjustable container. We propose a novel metaheuristic approach combining a genetic algorithm with the Gurobi solver. The algorithm incorporates a lower neighborhood search strategy and is underpinned by a mathematical model representing the multiorder open-dimension packing scenario. Extensive experiments validate the effectiveness of the proposed approach. The LNSGA algorithm outperforms Gurobi and the traditional genetic algorithm in solution quality and computational efficiency. For small-scale instances, LNSGA achieves optimal values in most cases. LNSGA demonstrates significant optimization improvements over Gurobi and the genetic algorithm for large-scale instances. The superior performance is attributed to the effective integration of the lower neighborhood search mechanism and the Gurobi solver. This study offers valuable insights for optimizing the packing process in e-commerce warehousing and logistics operations.

1. Introduction

Amidst the rapid growth of e-commerce, the global express business surpassed 170 billion pieces in 2021, with a year-on-year surge of over 25% [1]. In China, the postal industry achieved 139.1 billion dispatched parcels in 2022, with express business reaching 110.58 billion pieces, showcasing a year-on-year growth of 2.1% [2]. Efficient parcel packaging profoundly impacts user experience and can lead to savings of up to 30% in logistics expenses [3], making it strategically significant for e-commerce enterprises.

The three-dimensional bin packing problem originated in the 1960s, focusing on single-container packing [4]. The problem was proven to be NP-hard and exact algorithms were time-consuming [5, 6]. Research transitioned to metaheuristics, and as the 21st century progressed, the focus shifted to multicontainer [7], dynamic packing [8], and machine learning methodologies [9]. Recent advancements have led to specialized cases such as open-dimension packing [10] and multiorder mixed packing problems [11].

Open-dimension bin packing relaxes container size constraints to better reflect real-world scenarios. Existing literature has employed algorithms such as simulated annealing [12] and particle swarm optimization [13], which rely on random search and can get trapped in local optima. The multiorder mixed packing problem involves factors such as order priority and time limits, increasing complexity. The current research focuses on single objectives, with limited attention to multiobjective decision-making [11, 14].

As packing tasks become more intricate, traditional exact algorithms need help with multiple constraints. Metaheuristics such as genetic algorithms [15], simulated annealing, and

particle swarm algorithms [16, 17] have shown promise but face challenges in obtaining satisfactory solutions for complex problems. Researchers have explored integrating metaheuristics with exact solvers, giving rise to metaheuristics [18, 19]. Metaheuristics combines the global exploration capabilities of metaheuristics with the local optimization power of exact solvers, demonstrating superior performance in routing [20, 21], scheduling [22], and packing problems [23, 24].

This paper introduces a novel metaheuristic approach to address the multiorder open-dimension 3D rectangular packing problem. The proposed hybrid algorithm combines a genetic algorithm with the Gurobi solver to manage problem complexity and improve upon traditional metaheuristics. The algorithm is underpinned by a mathematical model representing the packing scenario and incorporates a lower neighborhood search strategy. Comparative experiments validate the approach's effectiveness in tackling modern packing problems and set a foundation for future research.

The paper is structured as follows. Section 2 reviews the literature on packing problems and solution algorithms. Section 3 presents the problem model and algorithm design. Section 4 focuses on experimental results and analysis. Section 5 concludes the paper and discusses future directions.

2. Literature Review

In this section, we review related research on the packing problem from three perspectives: application scenarios, dimensionality, and solution algorithms, respectively.

2.1. Packing Problems in Different Scenarios. The research on packing problems spans various practical domains, such as pallet loading, nesting problems, warehouse cargo packing, container loading, and cutting stock filling problems, among several others.

(1) Pallet loading problem: the pallet loading problem (PLP) involves placing multiple items on a pallet of a certain size to find the loading solution with the lowest stacking height [25–27].

(2) Nesting problem: nesting problems focus on efficiently cutting materials to obtain final items while adhering to shape-based rules [28, 29]. These problems are relevant across diverse industries, such as textile, clothing, sheet metal cutting, furniture manufacturing, and automobile industry [30, 31].

(3) Warehouse cargo packing problem: this problem involves efficiently packing goods based on order information from a warehouse. Scholars have explored irregular parts packing [32], flexible packing with nonfixed box sizes in e-commerce, and loading goods onto vehicles during the outbound warehouse process [33, 34].

(4) Container loading problem: the container loading problem (CLP) aims to load items into a container efficiently, maximizing the loading rate or minimizing the number of containers required, while considering support and stability constraints [35]. It encompasses three-dimensional container loading for transportation via ships, trucks, or railroad cars and air cargo packing. Researchers have also focused on multicontainer loading problems [36, 37].

(5) Cutting stock filling problem: the cutting stock filling problem involves efficiently arranging multiple items within a designated space in industrial contexts to maximize space utilization or minimize the total volume occupied [38, 39].

This paper focuses on the e-commerce warehousing state's packing problem, specifically packing warehouse goods. The subsequent section reviews studies on various dimensions of the packing problem.

2.2. Packing Problems in Different Dimensions. Packing problems can be classified based on dimensions, mainly one-dimensional, two-dimensional, and three-dimensional problems.

2.2.1. One-Dimensional and Two-Dimensional Packing Problems. This study focuses on packing problems in the context of e-commerce warehousing. One-dimensional packing problems have been studied but must be more relevant here [40]. Two-dimensional packing problems have received significant attention, including the guillotine knapsack packing problem, strip-packing problem with knifing constraints [41], and case packing for regular and irregular items [42]. The rectangular packing problem (RPP) describes the rectangular strip-packing problem (RSPP) [43]. Researchers have also investigated the undercutting, strip, and open-dimension 2D packing problems for various irregular objects [44].

2.2.2. Three-Dimensional Packing Problem. The three-dimensional packing problem extends classical one- and two-dimensional problems and has gained substantial attention. Researchers often convert 3D problems into 2D variants for easier solution procedures [45]. The three-dimensional packing problems include the following:

(1) 3D bin packing problems (3D-BPP): packing items of different sizes into the minimum number of boxes [46, 47].

(2) 3D knapsack loading problems (3D-KLP): maximizing the total value of items loaded into a knapsack [48, 49].

(3) 3D container packing problems (3D-CPP): loading items into a container to minimize volume or maximize loaded items [48, 50].

Scholars have studied 3D cutting and packing problems with nonoverlapping constraints and container loading problems within logistics platforms [51]. Container loading problems include multicontainer, LTL, and single-container

loading with transportation priority [52, 53]. Real-world problems also involve multipoint constraints requiring specific box accessibility at each delivery point.

This paper focuses on the 3D bin packing problem in e-commerce settings. However, the complexity of 3D packing problems challenges traditional exact algorithms. The next section reviews solution algorithms for complex 3D packing problems.

2.3. Algorithmic Research Based on the Three-Dimensional Packing Problem. The 3D bin packing problem is an NP-hard combinatorial optimization problem. Early studies used exact algorithms, but as problem scales grew, computation times became lengthy, and satisfactory solutions took time to obtain. From the 1970s, researchers turned to heuristic algorithms, with genetic algorithms performing prominently. In the 21st century, metaheuristic algorithms were widely adopted. Gezici and Livatyali improved the Harris Hawks algorithm's random parameter generation strategy to enhance the global search ability [54]. Xiong's team proposed a deep reinforcement learning-based method for online packing problems [55].

Unlike single-objective studies, El Yaagoubi et al. used NSGA-II with heuristic rules for fast solving at different scales [56]. Liu et al. targeted multibin packing problems with irregular items using 3D point cloud techniques and deep Q-networks [57]. Wang et al. abstracted resource allocation in open RAN networks into a 2D bin packing model and used a self-play reinforcement learning algorithm [58]. Gzara et al. designed an efficient algorithm for pallet loading problems with vertical support and load constraints [27]. Erbayrak et al. incorporated packing stability and same-type product grouping constraints into a multiobjective model [59]. Chen et al. used a biogeography-based optimization and differential evolution hybrid for the bin design problem [15].

As packing problems continue to grow in complexity, researchers are constantly seeking to enhance solution algorithms. To address the three-dimensional multiple bin size bin packing problem with open dimension and reserve parameter (3D-MOSB-ODRPP), this paper proposes an improved genetic algorithm that incorporates a lower neighborhood searchapproach.

2.4. Research Gap. The comprehensive review of packing problems and solution algorithms reveals several research gaps this study aims to address. First, existing research on packing problems has primarily concentrated on fixed-dimension containers, with limited attention given to open-dimension scenarios. However, in e-commerce warehousing, the flexibility to adjust container sizes based on order requirements is crucial for optimizing resource utilization and reducing costs. This study incorporates open-dimension packing into the problem formulation, allowing for the determination of optimal container dimensions.

Second, although metaheuristic algorithms have been widely adopted for solving packing problems, traditional approaches often need help to balance global exploration and local exploitation effectively. This study introduces a novel metaheuristic approach that combines the strengths of a genetic algorithm for global search with the local optimization capabilities of the Gurobi solver. By integrating these two components, the proposed algorithm aims to overcome the limitations of traditional metaheuristics and enhance solution quality.

Lastly, while previous studies have considered various constraints and practical factors, integrating multiple orders and open dimensions in a single packing problem has received limited attention. This study addresses this research gap by formulating the 3D-MOSB-ODRPP, which optimizes the packing of multiple orders with varying item sizes and quantities into a single, size-adjustable container.

To further highlight the unique contributions of this study, Table 1 compares the present research with the existing studies across several key dimensions.

As evident from Table 1, the present study simultaneously addresses multiple key aspects of the 3D packing problem. Hile's existing studies have individually considered some of these dimensions but still need to integrate them into a comprehensive problem formulation and solution approach. By bridging these research gaps, this study aims to provide a more realistic and effective solution to the complex 3D-MOSB-ODRPP encountered in e-commerce warehousing.

3. Methodology

This section introduces the 3D-MOSB-ODRPP model proposed in this paper and the improved genetic algorithm for solving this model. First, we will elaborate on the mathematical model for the multiorder open-dimension 3D packing problem, considering practical characteristics such as mixed orders and adjustable bin sizes. Then, we will elucidate the solution approach of the lower neighborhood search-based enhanced genetic algorithm, laying the groundwork for the computational experiments in later sections.

3.1. Problem Description. This paper primarily studies the 3D-MOSB-ODRPP. This problem expands on the 3D-ODRPP proposed by Tsai by comprehensively considering the optimization of adjustable container length, width, and height under multiple orders with a single box type. Specifically, as shown in Figure 1 given multiple packing orders where each order contains rectangular items with known length, width, and height, the goal is to determine a unified box size so that all items across orders can be packed into a container of that size, thus maximizing the container space utilization. This problem is more practical than fixed-size packing by tuning the box dimensions to balance order demands, but the complex constraints also increase the difficulty of solving it. This paper formulates a mixed-integer programming model to obtain accurate solutions and designs an improved genetic algorithm for effective solving, obtaining feasible schemes for the 3D-MOSB-ODRPP.

Table 1: Comparison of the present research with existing studies.

Literatures	Open-dimension packing	Precision algorithm + metaheuristic algorithm	Multiple orders	Practical constraints
[57]			✓	✓
[60]			✓	✓
[61]				✓
[62]				✓
[59]			✓	✓
[63]	✓		✓	✓
[64]	✓		✓	✓
[65]	✓	✓		✓
[66]	✓			✓
[67]	✓		✓	✓
This study	✓	✓	✓	✓

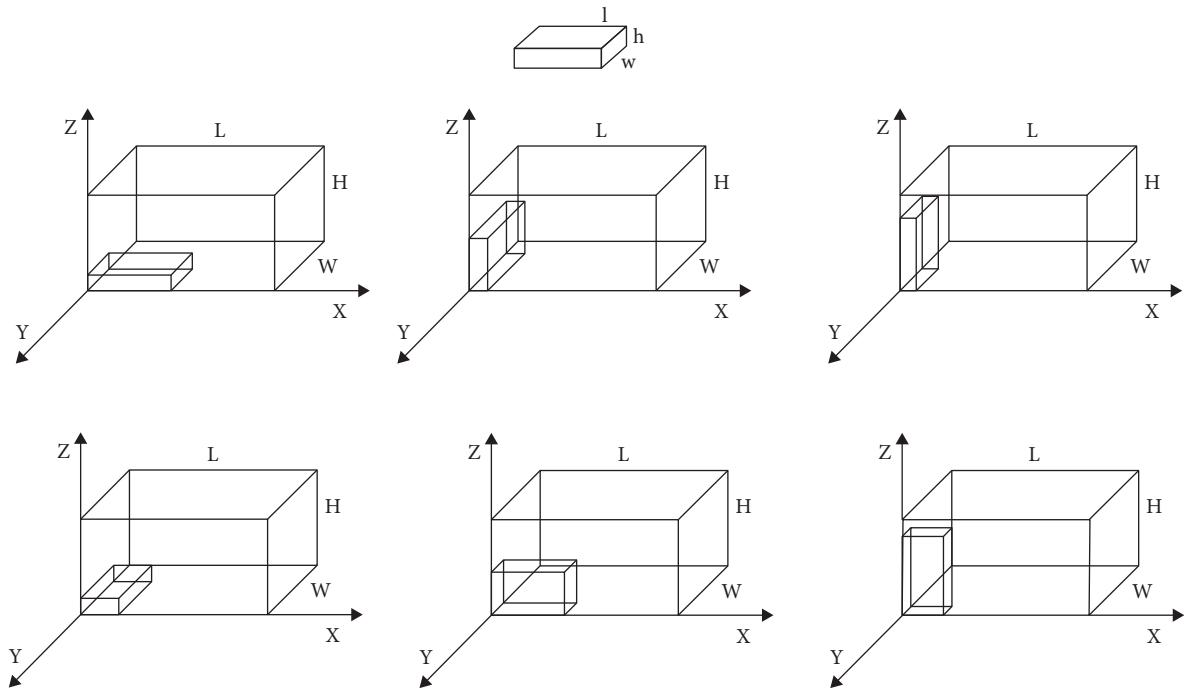

Figure 1: Six kinds of rectangular items rotation direction.

3.2. Model Construction

3.2.1. Variable Assumptions. This paper primarily involves two types of variables: conditional variables and decision variables. Specifically, in the multiorder open-dimension 3D packing problem studied, the length, width, height of items, and number of orders are conditional variables representing predetermined inputs. The container length, width, height, and item positioning are decision variables that need to be determined. In subsequent sections, we will define the conditional and decision variables involved in the studied problem and present their exact representations in the mathematical model.

(1) Conditional variables:

A detailed description of the condition variables is shown in Table 2.

(2) Decision variables:

Table 2: Description of condition variables.

Variables	Description
$l^I_{i_r}$	The length of the i_r-th item in the r-th order
$w^I_{i_r}$	The width of the i_r-th item in the r-th order
$h^I_{i_r}$	The height of the i_r-th item in the r-th order
q	Number of orders
M	A sufficiently large positive number

A detailed description of the decision variables is shown in Table 3.

3.2.2. Mixed-Integer Programming Model. We construct a mathematical model in this section to further solve the 3D-MOSB-ODRPP proposed in this paper. First, in order to maximize the space utilization efficiency while determining a unified container size, we define the

Table 3: Description of the decision variables.

Variables	Description
l^B	The length of the j-th packing case
w^B	The width of the j-th packing case
h^B	The height of the j-th packing case
x_{i_r}	Coordinates of the x-axis position of the i_r-th item in the r-th order loaded into the packing box
y_{i_r}	Coordinates of the y-axis position of the i_r-th item in the r-th order loaded into the packing box
z_{i_r}	Coordinates of the z-axis position of the i_r-th item in the r-th order loaded into the packing box
$T_{11}^{i_r}$, $T_{12}^{i_r}$, $T_{13}^{i_r}$, $T_{21}^{i_r}$, $T_{22}^{i_r}$, $T_{23}^{i_r}$, $T_{31}^{i_r}$, $T_{32}^{i_r}$, $T_{33}^{i_r}$	0-1 variables that determine the direction of rotation when the i_r-th item in the $r-$th order is boxed
$D_{i_r,i_r'}^k$	0-1 variable for designing nonoverlapping constraints between item i_r and item i_r' in the r-th order

objective function of the mixed-integer programming model as

$$\max z = l^B \cdot w^B \cdot h^B, \tag{1}$$

where z represents the box's volume, i.e., the product of three dimensions: length, width, and height.

Then, we introduce the constraint conditions in the mathematical model. First, to ensure that items can be smoothly packed into the container during the packing operation, we need to limit the range of each item inside the container to three dimensions-length, width, and height. The equations are as follows:

$$x_{i_r} + T_{11}^{i_r} \cdot l_{i_r}^I + T_{12}^{i_r} \cdot w_{i_r}^I + T_{13}^{i_r} \cdot h_{i_r}^I \le l^B, \tag{2}$$

$$y_{i_r} + T_{21}^{i_r} \cdot l_{i_r}^I + T_{22}^{i_r} \cdot w_{i_r}^I + T_{23}^{i_r} \cdot h_{i_r}^I \le w^B, \tag{3}$$

$$z_{i_r} + T_{31}^{i_r} \cdot l_{i_r}^I + T_{32}^{i_r} \cdot w_{i_r}^I + T_{33}^{i_r} \cdot h_{i_r}^I \le h^B, \tag{4}$$

where equation (2) calculates the range occupied by the item in the x-axis direction inside the container, considering the projection on the x-axis after rotation, which should be less than or equal to the container length l^B to ensure feasibility; equation (3) calculates the range occupied in the y-axis direction inside the container, which should be less than or equal to the container width w^B to ensure fit along the y-axis; and equation (4) calculates the range occupied in the z-axis direction inside the container, which should be less than or equal to the container height h^B to ensure fit along the z-axis.

After constraining items from exceeding container boundaries, we need to constrain that items do not overlap when placed inside the container, as shown in equation (5). Since after limiting items to be within the container as in equations (2)–(4), solely satisfying this condition cannot guarantee that the final packing solution is feasible. Packaging multiple items into one container will likely overlap in placement, leading to infeasibility. Thus, nonoverlap constraints must be added to clearly define the relative placement of items so that they do not overlap inside the container, thereby ensuring solution feasibility. Together, equations (2)–(5) ensure that the packing solution meets two key requirements, filling the container and no overlaps.

$$D_{i_r,i_r'}^k = 0 \text{ or } 1, \tag{5}$$

where $D_{i_r,i_{r'}}^k$ indicates whether items i_r and $i_{r'}$ overlap in the k-th direction (k ranges from 1 to 6, representing the positive and negative directions of x, y, z axes). When $D_{i_r,i_{r'}}^k$ takes the value 0, it means that the two items do not overlap in this dimension; when it takes the value 1, it means overlap is allowed between the two items. In other words, 0 indicates no overlap between the two items, and 1 indicates potential overlap. By defining the 0-1 variable D, we can explicitly formulate the relative placement relationship between items in each axis, thus laying the foundation for subsequent nonoverlap constraint calculations. Note that, the D variable is unrelated to the specific overlap situation; it merely indicates whether overlap is permitted between two items on a given axis.

In addition, to comprehensively and accurately ensure no overlap between any two items inside the container, we further need to add nonoverlap constraints between items with different IDs, as shown in equation (6). Although the previously defined 0-1 variable D clarified whether overlap is allowed between two items in each axis, it did not specifically define the absolute spatial relationship between two items. To achieve absolute nonoverlap inside the container, we need to calculate the specific coordinate ranges of different items in each axis and restrict these ranges from intersecting. This ensures that the two items do not overlap in a single-dimensional axis and avoids overlap when integrated in a three-dimensional space.

$$i_r, i_r' = 1, 2, \cdots m; \quad i_r \ne i_r', \tag{6}$$

where i_r and $i_{r'}$ represent the IDs of two different items in the same order, ranging from 1 to the total number of items m in the order. The item IDs i_r and $i_{r'}$ must differ and cannot take the same value. This is because, for the same item, its spatial projection region will not overlap with itself. If the IDs i_r and $i_{r'}$ take the same value, this nonoverlap constraint will degrade to a constraint between an item and itself, failing to achieve nonoverlap between different items. Therefore, the nonoverlap constraint must be expressed between two items with different IDs; only then can it truly

define the relative placement relationship between different items inside the container, avoiding overlap.

After ensuring no conflicts between the items to be packed and the container, we need to add constraints between different items in the same container to avoid overlaps and other conflicts between items, as shown in the following equations:

$$x_{i_{r'}} + T_{11}^{i_{r'}} \cdot l_{i_{r'}}^I + T_{12}^{i_{r'}} \cdot w_{i_{r'}}^I + T_{13}^{i_{r'}} \cdot h_{i_{r'}}^I \le x_{i_r} + \left(1 - D_{i_r,i_{r'}}^1\right) \cdot M, \tag{7}$$

$$x_{i_r} + T_{11}^{i_r} \cdot l_{i_r}^I + T_{12}^{i_r} \cdot w_{i_r}^I + T_{13}^{i_r} \cdot h_{i_r}^I \le x_{i_{r'}} + \left(1 - D_{i_r,i_{r'}}^2\right) \cdot M, \tag{8}$$

$$y_{i_{r'}} + T_{21}^{i_{r'}} \cdot l_{i_r}^I + T_{22}^{i_{r'}} \cdot w_{i_r}^I + T_{23}^{i_{r'}} \cdot h_{i_{r'}}^I \le y_{i_r} + \left(1 - D_{i_r,i_{r'}}^3\right) \cdot M, \tag{9}$$

$$y_{i_r} + T_{21}^{i_r} \cdot l_{i_r}^I + T_{22}^{i_r} \cdot w_{i_r}^I + T_{23}^{i_3} \cdot h_{i_r}^I \le y_{i_{r'}} + \left(1 - D_{i_r,i_{r'}}^4\right) \cdot M, \tag{10}$$

$$z_{i_{r'}} + T_{11}^{i_{r'}} \cdot l_{i_r}^I + T_{12}^{i_{r'}} \cdot w_{i_r}^I + T_{13}^{i_{r'}} \cdot h_{i_{r'}}^I \le z_{i_r} + \left(1 - D_{i_r,i_{r'}}^5\right) \cdot M, \tag{11}$$

$$z_{i_r} + T_{11}^{i_r} \cdot l_{i_r}^I + T_{12}^{i_r} \cdot w_{i_r}^I + T_{13}^{i} \cdot h_{i_r}^I \le z_{i_{r'}} + \left(1 - D_{i_r,i_{r'}}^6\right) \cdot M. \tag{12}$$

In equation (7), i_r and $i_{r'}$ represent two different items, $x_{i_{r'}} + T_{11}^{i_{r'}} \cdot l_{i_{r'}}^I + T_{12}^{i_{r'}} \cdot w_{i_{r'}}^I + T_{13}^{i_{r'}} \cdot h_{i_{r'}}^I$ calculates the range occupied by item $i_{r'}$ in the x-axis direction. $x_{i_r} + (1 - D_{i_r,i_{r'}}^1)$, and M calculates the x-axis coordinate of item i_r multiplied by a large non-negative number M. $D_{i_r,i_{r'}}^1$ is a 0-1 variable, taking the value 1 when the two items do not overlap, making M equal to 0 and satisfying the constraint. When the two items overlap, $D_{i_r,i_{r'}}^1$ is 0, making M a very large value, the right side greater than the left, and the constraint unsatisfied. Similarly, equation (8) ensures that items i_r and $i_{r'}$ do not overlap in the x-axis direction.

Equation (9), similar to equation (7), mainly calculates the range occupied by item $i_{r'}$ in the y-axis direction, comparing it with the y-coordinate of item i_r and a large number M. When the two items do not overlap on the y-axis, the 0-1 variable $D_{i_r,i_{r'}}^3$ is 1, making M equal to 0 and satisfying the constraint. If the two items overlap on the y-axis, $D_{i_r,i_{r'}}^3$ is 0, making M a large value and violating the constraint. Equation (10) is similar to equation (8), utilizing the 0-1 variable and the large number M to ensure that items i_r and $i_{r'}$ do not overlap in the y-axis direction.

Equation (11), similar to equations (7) and (9), calculates the range occupied by item $i_{r'}$ in the z-axis direction and compares it with the z-coordinate of item i_r and a large number M. When the two items do not overlap on the z-axis, the 0-1 variable $D_{i_r,i_{r'}}^5$ is 1, making M equal to 0 and satisfying the constraint. If overlap exists, $D_{i_r,i_{r'}}^5$ is 0, making M a large value and violating the constraint. Finally, equation (12) utilizes the 0-1 variable and large number M to ensure i_r and $i_{r'}$ do not overlap on the z-axis.

In order to further conflict between the items in the box, the following constraints are therefore set:

$$D_{i_r,i_{r'}}^1 + D_{i_r,i_{r'}}^2 + D_{i_r,i_{r'}}^3 + D_{i_r,i_{r'}}^4 + D_{i_r,i_{r'}}^5 + D_{i_r,i_{r'}}^6 \ge 1, \tag{13}$$

where $D_{i_r,i_{r'}}^1$ to $D_{i_r,i_{r'}}^6$ are 0-1 variables indicating whether items i_r and $i_{r'}$ overlap in the three dimensions. Since each D variable can only take 0 or 1, at least one of these variables must take the value 1, meaning the two items do not overlap in that dimension. Thus, this constraint ensures that any two items i_r and $i_{r'}$ do not overlap in at least one dimension, avoiding three-dimensional conflicts between them.

The abovementioned constraints are mainly used to avoid overlap conflicts between items and between items and container boundaries. To accurately calculate the space occupied by items in the container's length, width, and height dimensions, all possible placement directions of items need to be considered. For example, a rectangular item can be placed horizontally, vertically, or at various tilts. However, computing and storing all direction combinations for each item will greatly increase model complexity and solving difficulty. Therefore, this model uniquely determines the placement direction of each item. Specifically, we introduce 0-1 variables T to indicate the rotation decision of each item in three dimensions and construct rotation matrices to represent the projected length in each direction. As shown in equations (14)–(19), each item only chooses one placement direction, significantly reducing the model difficulty.

$$T_{11}^{i_r} + T_{12}^{i_r} + T_{13}^{i_r} = 1, \tag{14}$$

$$T_{21}^{i_r} + T_{22}^{i_r} + T_{23}^{i_r} = 1, \tag{15}$$

$$T_{31}^{i_r} + T_{32}^{i_r} + T_{33}^{i_r} = 1, \tag{16}$$

$$T_{11}^{i_r} + T_{21}^{i_r} + T_{31}^{i_r} = 1, \tag{17}$$

$$T_{12}^{i_r} + T_{22}^{i_r} + T_{32}^{i_r} = 1, \tag{18}$$

$$T_{13}^{i_r} + T_{23}^{i_r} + T_{33}^{i_r} = 1, \tag{19}$$

where equation (14) indicates that item i_r has three potential rotation states in the positive x-axis direction of the container, requiring that only one of these three states can take the value one and the rest 0. This ensures a unique rotation decision for the item in the x-axis direction. Equations (15) and (16) are similar, limiting item i_r to only one rotation state in the positive y-axis and z-axis of the container. Then, equation (17) further requires that for the rotation states of each axis, only one can be uniquely determined from the three options. That is, the rotation combination of an item must take one state per axis and cannot repeat selecting two or more states in one axis. Finally, equations (18) and (19) ensure that the length, width, and height of an item can only have unique effects in the three axes.

The following constraint conditions are set to ensure the uniqueness of item i_r's rotation state:

$$T_{11}^{i_r}, T_{12}^{i_r}, T_{13}^{i_r}, T_{21}^{i_r}, T_{22}^{i_r}, T_{23}^{i_r}, T_{31}^{i_r}, T_{32}^{i_r}, T_{33}^{i_r} = 0 \text{ or } 1, \tag{20}$$

where $T_{jk}^{i_r}(j = 1, 2, 3; k = 1, 2, 3)$ represents the 0-1 variable for the k-th rotation state of item i_r in the j-th axis direction. Each $T_{jk}^{i_r}$ variable can only take 0 or 1, 0 meaning item i_r does not select the corresponding rotation state in that axis direction, and 1 meaning it selects that rotation state in that axis. These 0-1 variables are introduced to explicitly represent the rotation decision of items, providing a quantitative way to model the rotation states. Since each item can only select one rotation state, by requiring the T variables in each axis to sum to 1, the uniqueness of the rotation scheme can be ensured.

In addition, to define the value range of the rotation decision k and to ensure accurate computing of the coordinate's positional relationship between items and the container, the following equation is set as:

$$k = 1, 2, 3, 4, 5, 6, \tag{21}$$

where k takes different integer values to represent the correspondence between item length and width directions and the three coordinate axes: $k = 1$ indicates that the item length direction is consistent with the x-axis, $k = 2$ indicates that the item length direction is consistent with the y-axis, $k = 3$ indicates that the item length direction is consistent with the z-axis, $k = 4$ indicates that the item width direction is consistent with the x-axis, $k = 5$ indicates that the item width is consistent with the y-axis, and $k = 6$ indicates that the item width is consistent with the z-axis. This variable is introduced to establish the connection between item length and width directions and the spatial coordinate axes directions, with k indicating possible item placement directions. After determining the value of k, items' spatial occupancy and coordinate positions can be calculated based on their projected dimensions on the corresponding axes.

Finally, to facilitate subsequent calculations of the model, a relaxation constraint is set here as

$$M \gg 0. \tag{22}$$

3.2.3. Algorithm Design. In order to address the challenges posed by the 3D-MOSB-ODRPP, this section proposes a hybrid approach termed the Gurobi-enhanced local neighborhood search genetic algorithm (LNSGA). Traditional genetic algorithms focus on global exploration, whereas the local neighborhood search mechanism targets local exploration. Consequently, the LNSGA algorithm integrates both the global and local exploration organically. Moreover, following the local neighborhood search, invoking the Gurobi solver ensures that the returned solution represents the globally optimal solution for the three-dimensional item packing.

A plain genetic algorithm (GA) is also implemented for the 3D-MOSB-ODRPP problem to compare and analyze the performance of the LNSGA algorithm. The GA follows the basic structure of genetic algorithms, including population initialization, fitness evaluation, selection, crossover, and mutation. However, unlike the LNSGA algorithm, the GA does not incorporate the Gurobi solver or the lower neighborhood search mechanism.

In the GA implementation, the three-space (TS) heuristic algorithm is employed to calculate the packing configuration of items. The TS algorithm is a fundamental heuristic approach in three-dimensional packing problems. It operates by recursively dividing the remaining space into three subspaces and selecting the most suitable subspace for placing the next item. The process continues until all items are packed or no feasible subspace is available.

The pseudocode of the GA algorithm is shown in Algorithm 1, and its key steps are as follows:

(1) Initialization: generating an initial population of individuals, each representing a potential packing sequence of items.

(2) Fitness evaluation: evaluating each individual's fitness in the population using the TS algorithm. The volume utilization ratio of the packing configuration determines the fitness value.

(3) Selection: applying a selection operator, such as tournament selection or roulette wheel selection, to choose individuals with higher fitness values for reproduction.

(4) Crossover: performing a crossover operation, such as one-point crossover or two-point crossover, to create offspring individuals by exchanging genetic information between selected parent individuals.

(5) Mutation: applying a mutation operator, such as swap mutation or insertion mutation, to introduce random variations in the offspring individuals.

(6) Replacement: replacing a portion of the population with the newly generated offspring individuals based on their fitness values.

(7) Termination: repeating steps 2–6 until a predefined termination criterion is met, such as reaching a maximum number of generations or achieving a satisfactory solution quality.

Compared to the LNSGA algorithm, the GA relies solely on the global exploration capabilities of genetic algorithms and the basic TS heuristic for packing calculations. It does not benefit from the targeted lower neighborhood search

Require: population size N, crossover probability p_c, mutation probability p_m, and maximum number of generations $G_{\max}$
Ensure: best solution s^*
(1) Initialize population P with N randomly generated individuals
(2) Evaluate the fitness of each individual in P using the TS heuristic
(3) $g \leftarrow 0$
(4) **while** $g < G_{\max}$ **do**
(5) $P' \leftarrow \varnothing$
(6) **while** $|P'| < N$ **do**
(7) Select parents p_1 and p_2 from P using the selection operator
(8) $o_1, o_2 \leftarrow$ Crossover (p_1, p_2) with probability p_c
(9) $o_1 \leftarrow$ Mutate (o_1) with probability p_m
(10) $o_2 \leftarrow$ Mutate (o_2) with probability p_m
(11) Evaluate the fitness of o_1 and o_2 using the TS heuristic
(12) $P' \leftarrow P' \cup \{o_1, o_2\}$
(13) **end while**
(14) $P \leftarrow P'$
(15) $g \leftarrow g + 1$
(16) **end while**
(17) **return** Best solution s^* from P

Algorithm 1: Genetic algorithm (GA) for 3D-MOSB-ODRPP.

mechanism or the global optimization capabilities of the Gurobi solver. As a result, the GA may need more performance in terms of solution quality and computational efficiency, especially for complex 3D-MOSB-ODRPP instances.

The pseudocode of the LNSGA algorithm is shown in Algorithm 2, and its key steps are as follows:

(1) Let $\mathcal{O} = O_1, O_2, \ldots, O_q$ denote the set of q orders, where each order O_r $(r \in 1, 2, \ldots, q)$ is considered as a distinct three-dimensional open-dimension rectangular packing problem (3D-ODRPP). The Gurobi solver is utilized to obtain the optimal packing dimensions $(L^r, W^r, and\ H^r)$ for each order O_r. Traditional genetic algorithms focus on global exploration by searching the solution space $\mathcal{S}$ through population evolution and information exchange. However, for complex 3D-MOSB-ODRPP instances, global exploration alone may lead to local optima. A local neighborhood search mechanism is introduced to perform local optimization on the current solution $s \in \mathcal{S}$ to enhance local search capability.

(2) We sort the optimal packing dimensions of each order $(L^r, W^r, and\ H^r)$ in a descending order. Then, we determine the maximum length, width, and height values across all orders, denoted as $(L^{\mathrm{up}}, W^{\mathrm{up}}, and\ H^{\mathrm{up}})$, which serve as the upper bounds for the 3D-MOSB-ODRPP solution. Mathematically, it is represented as

$$\begin{aligned} L^{\mathrm{up}} &= \max L^1, L^2, \cdots, L^q, \\ W^{\mathrm{up}} &= \max W^1, W^2, \cdots, W^q, \\ H^{\mathrm{up}} &= \max H^1, H^2, \cdots, H^q. \end{aligned} \tag{23}$$

(3) The lower neighborhood search scope for open dimensions is defined as follows: given the upper bounds $(L^{\mathrm{up}}, W^{\mathrm{up}}, and\ H^{\mathrm{up}})$, the 1-step lower neighborhood $\mathcal{N}_1(L^{\mathrm{up}}, W^{\mathrm{up}}, and\ H^{\mathrm{up}})$ consists of 7 value options, obtained by adding 0 or −1 to each dimension, excluding the case where all increments are 0. The lower neighborhood search aims to find better solutions $s' \in \mathcal{N}_1(s)$ within the neighborhood of the current solution s. By designing appropriate neighborhood structures and search strategies, the algorithm can escape from local optima and explore the solution space more deeply. The lower neighborhood search adjusts the length, width, and height dimensions to generate neighboring solutions, thereby selecting the best one as the new current solution (numbered 1–7 as shown in Table 4).

(4) Based on the open dimensions' 1-step lower neighborhood search scope values, a step search path is designed as shown in Figure 2. The seven nodes in each search step correspond to the seven cases in the 1-step lower neighborhood search, and the multi-step search accumulates value increments based on the 1-step search. For example, starting from $(L^{\mathrm{up}}, W^{\mathrm{up}}, and\ H^{\mathrm{up}})$, when the search path is $5 \longrightarrow 2 \longrightarrow 3 \longrightarrow 3$, the lower neighborhood search values experienced sequentially by the open dimension values are shown in Table 5.

(5) The sequence of g node numbers in a lower neighborhood search path is used as the chromosome encoding of an individual in the genetic algorithm. Let $C_i = (n_1, n_2, \ldots, n_g)$ denote the chromosome of the i-th individual, where $n_j \in 1, 2, \ldots, 7$ represents the node number at the j-th step of the search path.

(6) We randomly generated an initial population $P = C_1, C_2, \ldots, C_{M_{\mathrm{group}}}$ consisting of M_{group} chromosomes, each corresponding to a lower neighborhood search path.

Require: population size N, crossover probability p_c, mutation probability p_m, maximum number of generations $G_{\max}$, and neighborhood size k
Ensure: best solution s^*
(1) Initialize population P with N randomly generated individuals
(2) **for** each individual s in P **do**
(3) Evaluate the fitness of s using the Gurobi solver
(4) $s' \leftarrow$ LocalNeighborhoodSearch(s, k)
(5) $s^* \leftarrow$ Gurobi (s')
(6) Update s with s^* in P
(7) **end for**
(8) $g \leftarrow 0$
(9) **while** $g < G_{\max}$ **do**
(10) $P' \leftarrow \varnothing$
(11) **while** $|P'| < N$ **do**
(12) Select parents p_1 and p_2 from P using the selection operator
(13) $o_1, o_2 \leftarrow$ Crossover (p_1, p_2) with probability p_c
(14) $o_1 \leftarrow$ Mutate (o_1) with probability p_m
(15) $o_2 \leftarrow$ Mutate (o_2) with probability p_m
(16) **for** each offspring o in o_1, o_2 **do**
(17) Evaluate the fitness of o using the Gurobi solver
(18) $o' \leftarrow$ LocalNeighborhoodSearch(o, k)
(19) $o^* \leftarrow$ Gurobi (o')
(20) Update o with o^* in P'
(21) **end for**
(22) $P' \leftarrow P' \cup \{o_1, o_2\}$
(23) **end while**
(24) $P \leftarrow P'$
(25) $g \leftarrow g + 1$
(26) **end while**
(27) **return** Best solution s^* from P

Algorithm 2: Local neighborhood search genetic algorithm (LNSGA) for 3D-MOSB-ODRPP.

Table 4: Lower neighborhood search for open dimensions 1-step range taking the case.

Number	Increments in length, width, and height	Variable neighborhood values
1	(−1, 0, 0)	$(L^{\text{up}} - 1, W^{\text{up}}, H^{\text{up}})$
2	(0, −1, 0)	$(L^{\text{up}}, W^{\text{up}} - 1, H^{\text{up}})$
3	(0, 0, −1)	$(L^{\text{up}}, W^{\text{up}}, H^{\text{up}} - 1)$
4	(−1, −1, 0)	$(L^{\text{up}} - 1, W^{\text{up}} - 1, H^{\text{up}})$
5	(0, −1, −1)	$(L^{\text{up}}, W^{\text{up}} - 1, H^{\text{up}} - 1)$
6	(−1, 0, −1)	$(L^{\text{up}} - 1, W^{\text{up}}, H^{\text{up}} - 1)$
7	(−1, −1, −1)	$(L^{\text{up}} - 1, W^{\text{up}} - 1, H^{\text{up}} - 1)$

(7) The fitness function is then designed to evaluate the quality of each individual C_i in the population as

$$\text{fitness}(C_i) = L^{\text{up}}(C_i) \cdot W^{\text{up}}(C_i) \cdot H^{\text{up}}(C_i), \quad (24)$$

where $L^{\text{up}}(C_i)$, $W^{\text{up}}(C_i)$, and $H^{\text{up}}(C_i)$ represent the maximum length, width, and height of the packing solution obtained by following the search path encoded in chromosome C_i, and the objective is to minimize the fitness value.

(8) The roulette wheel selection is applied to select individuals with high fitness values to form a new population P'. In each iteration, the individuals in the current population P are sorted by their fitness values. The individual with the minimum fitness value is selected with a probability of $(0.5)^1$, the second minimum with a probability of $(0.5)^2$, and so on, until a new population P' with the same size as P is formed.

(9) Based on the crossover probability p_c, we randomly selected two chromosomes C_i and C_j from the population P'. A crossover point $k \in 1, 2, \ldots, g-1$ is randomly chosen and the gene segments are exchanged after the crossover point between C_i and C_j to generate two offspring chromosomes C_i' and C_j'. C_i and C_j in the population are then replaced with C_i' and C_j'. The crossover process is shown in Figure 3.

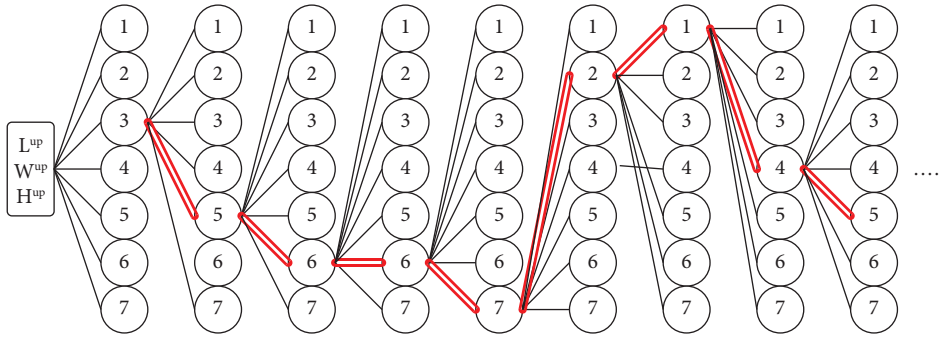

Figure 2: Schematic of a g-step search line in the lower neighborhood.

Table 5: Neighborhood search fetches experienced sequentially by the open dimension.

Search nodes	Variable neighborhood increment	Open-size node cumulative fetch
5	(0, −1, −1)	$(L^{up}, W^{up} - 1, H^{up} - 1)$
2	(0, −1, 0)	$(L^{up}, W^{up} - 2, H^{up} - 1)$
3	(0, 0, −1)	$(L^{up}, W^{up} - 2, H^{up} - 2)$
5	(0, −1, −1)	$(L^{up}, W^{up} - 3, H^{up} - 3)$
3	(0, 0, −1)	$(L^{up}, W^{up} - 3, H^{up} - 4)$

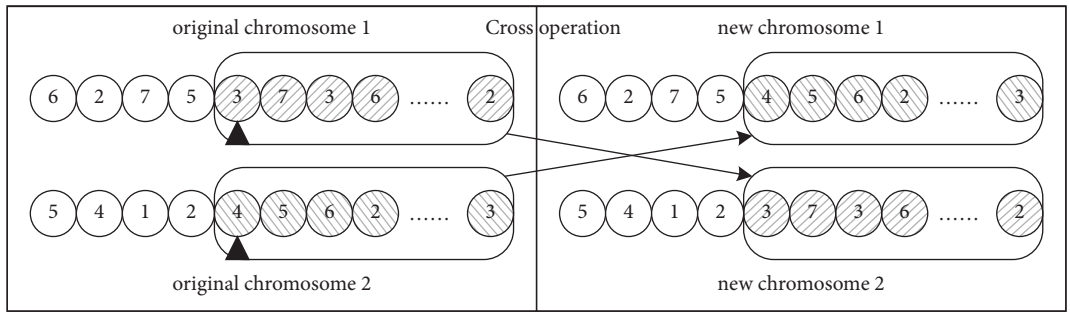

Figure 3: Chromosome crossover operations.

(10) Based on the mutation probability p_m, we randomly selected a chromosome C_i from the population P'. A mutation point $k \in 1, 2, \ldots, g$ is randomly chosen and the gene at position k is replaced with a randomly selected value from $1, 2, \ldots, 7$ to generate a new chromosome C_i'. C_i is then replaced with C_i' in the population. The mutation process is shown in Figure 4.

(11) Steps 7 through 10 are repeated until the maximum number of iterations EPOCH is reached. To further enhance performance, the Gurobi solver is invoked after each local neighborhood search to perform global optimization on the current solution s. The Gurobi solver is an exact method that solves the 3D packing problem to obtain the global optimal solution s^* based on s. By combining local neighborhood search with the Gurobi solver, the LNSGA algorithm achieves an organic integration of global exploration and local search, enabling it to escape from the local optima and obtain high-quality global optimal solutions.

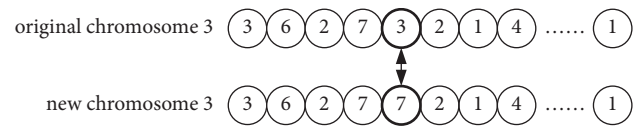

Figure 4: Chromosome mutation operations.

3.3. Computational Experiment

3.3.1. Dataset Design. The data required for the current experiment includes item size information and order data information.

(1) Item Information.

The SKU of an item represents the smallest packaging unit of storage items and serves as an essential information for order items. Different items have varying sizes and SKU dimensions. This study sets the upper and lower limits, along with the change step, for each item' SKU's length, width, and height, as shown in Table 6. Accordingly, there are ten possible values for each dimension of the item SKU,

Table 6: Order item SKU size generation information.

	Lower limit	Upper limit	Variation of the step size
Length	21	30	1
Width	11	20	1
Height	1	10	1
Total number of SKUs		1000	

resulting in 1000 different SKU combinations when considering the variations across all three dimensions. Thus, 1000 unique item sizes form the basis for generating item order information.

(2) Multiscale order dataset.

From the generated 1000 types of items, 2 to 6 items are randomly selected as the item information for an order. For example, when dealing with an order consisting of 4 items, its schematic diagram is shown in Figure 5. In order to conduct computational experiments with different order quantities, two sets of cases with different scales of order quantities are randomly selected, totaling 24 scenarios for the experiments. Each scenario includes orders of varying quantities, and all orders in each scenario are required to be packed into packaging boxes of the same size and model.

3.3.2. Algorithm's Hyperparameter Settings. This study conducted computational experiments on six sets of small-scale test cases to determine the final hyperparameter values for the LNSGA algorithm. Two of these test cases exhibited relatively unique order item quantities and sizes, resulting in identical optimal fitness values under different parameter settings. Although such instances were relatively infrequent, the careful analysis revealed that when data scales were small or the problems were relatively simple, any combination of hyperparameters could easily attain the optimal solution. Alternatively, in specific scenarios, all combinations of hyperparameters could readily converge to the same local optimum, resulting in identical optimal solution values across all hyperparameter combinations.

It is worth noting that among the experimental results of the six test cases, the final values of the hyperparameterspopulation size (GROUP) and the number of algorithm iterations (EPOCH) were consistent, 35 and 10, respectively. However, there were multiple possible values for the hyperparameters pc (crossover probability) and pm (mutation probability). The pc values were 0.4, 0.65, and 0.9, while the pm values were 0.1, 0.2, and 0.3. Considering that pc and pm represent the population's crossover probability and mutation probability, respectively, larger values of pc and pm within the allowed range indicate more active crossover and mutation in the population, thus increasing the likelihood of finding solutions with higher optimization. Therefore, the final values of pc and pm were determined to be 0.9 and 0.3, respectively.

The hyperparameters to be determined include the GROUP, the EPOCH, the pc, and the pm. The upper and lower limits for each hyperparameter search, along with the number of variation steps, are shown in Table 7. The computational process of the three test cases with two orders each in the small-scale example serves as the reference basis for determining the hyperparameters. As shown in Table 7, the optimal parameter combination for the four hyperparameter test cases is GROUP = 35, EPOCH = 10, pc = 0.4, 0.65, or 0.9, and pm = 0.1, 0.2, or 0.3. The optimal results for the three test cases under different parameter combinations and computation times are illustrated in Figure 6. The final chosen values for the hyperparameter combination are GROUP = 35, EPOCH = 10, pc = 0.9, and pm = 0.3 to ensure that the fitness function value converges stably to the optimal state.

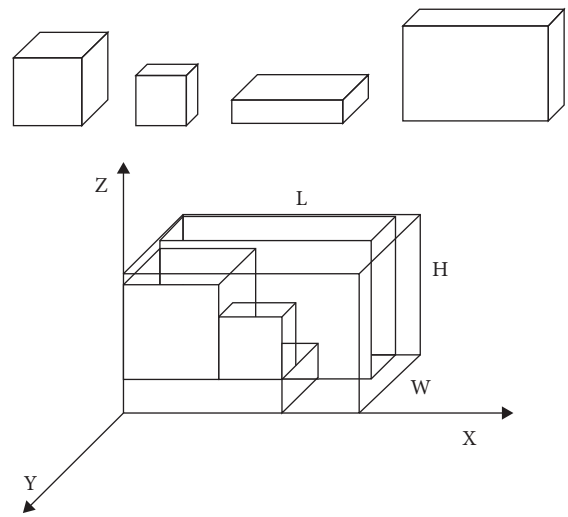

Figure 5: Boxing schematic for orders.

4. Calculation Results and Analysis

4.1. Comparative Experiments with Gurobi

(1) Computational experiments for small-scale orders.

To substantiate the efficacy of the lower neighborhood search genetic algorithm (LNSGA) for the three-dimensional multiorder single-box open-dimension rectangular packing problem (3D-MOSB-ODRPP), we conducted a series of experiments focusing on small-scale order cases. These cases, comprising 2–8 orders, were selected to provide a controlled environment for assessing the algorithm's performance. The determined hyperparameters for the LNSGA, as outlined in Figure 6, were utilized to compute these test cases within the generated dataset.

Table 8 provides a comprehensive overview of the solutions obtained from both LNSGA and Gurobi, including the container dimensions (L, W, and H), the objective values, and the computational time required to reach these solutions. To evaluate the performance of LNSGA, we calculated the percentage gap (GAP) between the objective values achieved by LNSGA and the optimal values determined by Gurobi.

Table 7: Parameter setting of the genetic algorithm.

	Lower limit of change	Upper limit of change	Variation of the step size	Optimal solution parameter set	Final parameter selection
Group	20	50	3	35	35
EPOCH	10	100	3	10	10
pc	0.4	0.9	3	0.4, 0.65, 0.9	0.9
pm	0.1	0.3	3	0.1, 0.2, 0.3	0.3

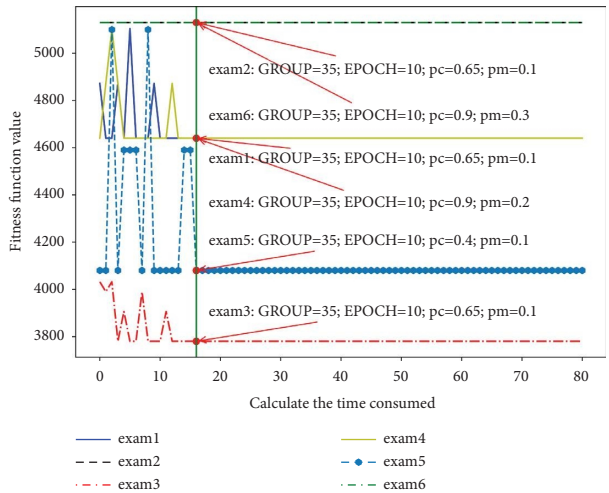

Figure 6: Hyperparameter settings to calculate experimental results.

Experimental results show that in small-sample instances requiring rapid decision-making, LNSGA demonstrates a good balance between solution quality and computational resources. Specifically, without imposing a time limit, the LNSGA algorithm performs significantly better than the Gurobi solver, achieving optimal values in the first three test cases. Despite some deviation from the optimal values as the number of orders increases (with a maximum GAP value of 17.5% for LNSGA), it still exhibits considerable advantages over Gurobi (with a maximum GAP value of 72.81%).

(2) Computational experiments for large-scale orders.

Based on the same hyperparameter settings, the LNSGA algorithm is used to compute test cases with large-scale order quantities in the generated dataset (for large-scale order cases, refer to cases with 12 to 18 orders). The results are compared with the cases solved directly using the Gurobi solver, as shown in Table 9.

Under the same computation time for each test case, the proposed LNSGA algorithm in this paper achieves significant optimization improvements over the Gurobi solver, with the highest optimization level being 71.59% and the average being 50.16%. In addition, it can be seen from Table 8 that as the order quantity scale increases, the CPU computation time also increases accordingly.

4.2. Comparative Experiments with GA. In the previous section, we compared the performance gap between the commercial solver Gurobi and the proposed LNSGA algorithm. However, despite the significant advantages of the LNSGA algorithm in terms of both runtime and optimization accuracy over Gurobi, more is needed to demonstrate that the LNSGA algorithm is optimal for solving 3D-MOSB-ODRPP. Therefore, to validate the necessity of our modifications to the GA, this section will introduce comparative experiments between GA and LNSGA.

Although LNSGA and GA have the same iteration limit (EPOCH), LNSGA demonstrates faster computational speed than GA. This can be attributed to several factors as follows:

(1) The GA algorithm does not invoke the Gurobi solver for obtaining packing solutions. Instead, it employs a heuristic algorithm based on the three-space strategy. The three-space strategy requires maintaining a list of spaces and calculating the splitting and merging of spaces, which is relatively time-consuming.

(2) The basic neighborhood search mechanism in GA can lead to the inefficient search of the packing box dimensions. The packing method needs to be recalculated when larger or smaller packing box dimensions are explored. Excessively large or small packing box dimensions require more time to calculate the splitting and merging of spaces, resulting in increased time for updating the space list.

Table 8: Case results for small-scale order sizes (comparison with Gurobi).

Number	Number of orders	Optimal solution	Optimal value	Gurobi			LNSGA			Time limit (s)
				Solution (L,W,H)	Objective value: $L \times W \times H$	GAP (%)	Solution (L,W,H)	Objective value: $L \times W \times H$	GAP (%)	
1	2	(29, 20, 8)	4640	(29, 26, 10)	7540	38.46	(29, 20, 8)	4640	0.00	25.53
2	2	(30, 19, 9)	5130	(46, 28, 9)	11592	55.75	(30, 19, 9)	5130	0.00	36.72
3	2	(30, 18, 7)	3780	(30, 21, 14)	8820	57.14	(30, 18, 7)	3780	0.00	21.7
4	4	(30, 19, 10)	5800	(33, 17, 30)	20097	71.14	(30, 20, 10)	6000	3.33	68.1
5	4	(29, 20, 9)	5130	(33, 17, 30)	16830	69.52	(30, 20, 9)	5400	5.00	81.93
6	4	(28, 20, 10)	5600	(35, 30, 16)	16800	66.67	(30, 20, 10)	6000	6.67	72.96
7	6	(29, 18, 10)	5880	(30, 24, 21)	15120	61.11	(30, 20, 10)	6000	2.00	89.98
8	6	(32, 19, 10)	7000	(34, 30, 11)	11220	37.61	(37, 20, 10)	7400	5.41	107.78
9	6	(28, 18, 10)	5510	(30, 24, 20)	14400	61.74	(30, 20, 10)	6000	8.17	107.88
10	8	(28, 19, 10)	6200	(38, 30, 20)	22800	72.81	(32, 20, 10)	6400	3.13	168.41
11	8	(52, 17, 9)	9405	(47, 24, 20)	22560	58.31	(57, 20, 10)	11400	17.50	120.61
12	8	(27, 18, 10)	5040	(30, 25, 19)	14250	64.63	(30, 20, 10)	6000	16.00	127.33
Average		—	5760	—	15169	59.57	—	6179	5.60	85.74

Table 9: Case results for large-scale order sizes (comparison with Gurobi).

Number	Number of orders	Gurobi		LNSGA		CPU time (s)	Optimization of LNSGA over Gurobi (%)
		Solution (L, W, H)	Objective value: (L, W, H)	Solution (L, W, H)	Objective value: (L, W, H)		
1	12	(57, 20, 20)	22800	(89, 20, 10)	17800	228.92	21.93
2	12	(32, 30, 22)	21120	(30, 20, 10)	6000	202.35	71.59
3	12	(30, 28, 24)	20160	(30, 20, 10)	6000	244.26	70.24
4	14	(32, 28, 25)	22400	(56, 20, 10)	11200	293.26	50.00
5	14	(30, 28, 24)	20160	(30, 20, 10)	6000	293.41	70.24
6	14	(30, 30, 21)	18900	(30, 20, 10)	6000	254.57	68.25
7	16	(31, 27, 23)	19251	(30, 20, 10)	6000	257.23	68.83
8	16	(44, 30, 18)	23760	(55, 20, 10)	11000	320.07	53.70
9	16	(36, 30, 20)	21600	(37, 20, 10)	7400	376.72	65.74
10	18	(30, 30, 20)	18000	(88, 20, 10)	17600	386.37	2.22
11	18	(30, 27, 23)	18630	(78, 20, 10)	15600	316.59	16.26
12	18	(32, 29, 20)	18560	(53, 20, 10)	10600	356.51	42.89
Average		—	20445	—	10100	294.19	50.16

(3) In complex scenarios involving space splitting and merging, the computation time for calculating the item packing solution also increases to a certain extent.

On the other hand, LNSGA incorporates a targeted lower neighborhood search mechanism, which efficiently explores the packing box dimensions by adjusting the length, width, and height within a specific range. This targeted search reduces the occurrence of excessively large or small packing box dimensions, thereby avoiding unnecessary calculations of space splitting and merging. Moreover, LNSGA utilizes the Gurobi solver to obtain globally optimal packing solutions for each lower neighborhood search, further enhancing its computational efficiency. Therefore, the GA's reliance on the three-space strategy and its basic neighborhood search mechanism leads to slower computational speed than LNSGA. The targeted lower neighborhood search and integration of the Gurobi solver in LNSGA contribute to its faster performance.

(1) Computational experiments for small-scale orders.

In this section, we conducted experiments with small-scale order data to compare the performance of the LNSGA algorithm with the genetic algorithm (GA) and the variable neighborhood search (VNS) algorithm. The experimental results, as shown in Table 10, demonstrate the LNSGA algorithm's superiority in terms of solution quality and computational efficiency.

From the perspective of the objective value, which represents the volume of the packing box $(L \times W \times H)$, the LNSGA algorithm consistently outperformed the GA and VNS algorithms in all 12 test cases. Compared to the GA, the LNSGA algorithm achieved a significant improvement in the target function, ranging from 34.05% to 71.93%. On average, the LNSGA algorithm obtained an objective value of 6179, which is 7.38% lower than the GA's average of 6671 and 9.69% lower than the VNS's average of 6842. This indicates that the LNSGA algorithm can find more compact packing solutions, thereby minimizing the required box volume.

Regarding computation time, the LNSGA algorithm exhibited remarkable efficiency compared to the GA and VNS algorithms. Across all test cases, the LNSGA algorithm reduced the runtime by more than 50% compared to the GA. As the number of orders increased from 2 to 8, the computation time of the GA algorithm showed a significant upward trend, ranging from 60 seconds to approximately 450 seconds. In contrast, the computation time of the LNSGA algorithm increased more gradually, reaching a maximum of around 168 seconds for the test case with eight orders. On average, the LNSGA algorithm had a computation time of 85.74 seconds, which is 58.38% faster than the GA's average of 206.08 seconds.

Compared to the VNS algorithm, the LNSGA algorithm demonstrated comparable computation times for smaller test cases with 2–4 orders. However, as the number of orders increased to 6 and 8, the LNSGA algorithm maintained its computational efficiency, while the VNS algorithm's computation time increased more rapidly. Despite the slightly higher average computation time of the LNSGA algorithm (85.74 seconds) compared to the VNS algorithm (50.72 seconds), the LNSGA algorithm achieved significantly better objective values, justifying the marginal increase in computation time.

(2) Computational experiments for large-scale orders.

In this section, we mainly compared the performance of GA and LNSGA with more orders. The comparison results are shown in Table 11. From the perspective of the objective value, the LNSGA algorithm achieved better packing solutions than the GA algorithm in all cases. Although the average improvement in the target function value was not as significant as in the small-scale cases, the quality of solutions obtained by LNSGA still exhibited a significant advantage. In terms of computation time, the LNSGA algorithm also outperformed the GA algorithm, reducing the runtime by approximately 50%. As the number of

Table 10: Case results for small-scale order sizes (comparison with VNS and GA).

Number	Number of orders	GA			VNS			LNSGA		
		Solution (L, W, H)	Objective value: LWH	CPU time (s)	Solution (L, W, H)	Objective value: $L \times W \times H$	CPU time (s)	Solution (L, W, H)	Objective value: $L \times W \times H$	CPU time (s)
1	2	(8, 26, 29)	6032	60.76	(9, 25, 29)	6525	24.61	(8, 20, 29)	4640	25.53
2	2	(9, 22, 30)	5940	92.58	(10, 21, 31)	6510	29.92	(9, 19, 30)	5130	36.72
3	2	(7, 28, 30)	5880	43.83	(8, 27, 30)	6480	28.25	(7, 18, 30)	3780	21.7
4	4	(10, 21, 30)	6300	157.01	(10, 20, 29)	5800	49.96	(10, 20, 30)	6000	68.1
5	4	(9, 20, 30)	5400	190.5	(10, 21, 29)	6090	25.43	(9, 20, 30)	5400	81.93
6	4	(10, 20, 30)	6000	167.44	(10, 20, 30)	6000	52.53	(10, 20, 30)	6000	72.96
7	6	(10, 20, 30)	6000	198.16	(10, 20, 31)	6200	64.82	(10, 20, 30)	6000	89.98
8	6	(10, 28, 30)	8400	283.27	(10, 22, 32)	7040	58.74	(10, 20, 37)	7400	107.78
9	6	(10, 20, 30)	6000	296.28	(11, 20, 29)	6380	78.4	(10, 20, 30)	6000	107.88
10	8	(10, 20, 31)	6200	444.52	(12, 20, 30)	7200	66.49	(10, 20, 32)	6400	168.41
11	8	(10, 20, 52)	10400	264.08	(13, 19, 55)	11495	61.54	(10, 20, 57)	11400	120.61
12	8	(10, 25, 30)	7500	274.55	(11, 20, 29)	6380	67.91	(10, 20, 30)	6000	127.33
Average		—	6671	206.08	—	6842	50.72	—	6179	85.74

Table 11: Case results for larger-scale order sizes (comparison with GA).

Number	Number of orders	GA			LNSGA		
		Solution (L, W, H)	Objective value: (L, W, H)	CPU time (s)	Solution (L, W, H)	Objective value: (L, W, H)	CPU time (s)
1	2	(84, 22, 10)	18480	566.97	(89, 20, 10)	17800	228.92
2	2	(31, 21, 10)	6510	479.96	(30, 20, 10)	6000	202.35
3	2	(30, 20, 10)	6000	577.57	(30, 20, 10)	6000	244.26
4	4	(50, 20, 10)	10000	684.31	(56, 20, 10)	11200	293.26
5	4	(31, 20, 10)	6200	679.49	(30, 20, 10)	6000	293.41
6	4	(30, 20, 10)	6000	571.58	(30, 20, 10)	6000	254.57
7	6	(25, 30, 10)	7500	657.76	(30, 20, 10)	6000	257.23
8	6	(49, 20, 10)	9800	834.24	(55, 20, 10)	11000	320.07
9	6	(34, 23, 10)	7820	1012.27	(37, 20, 10)	7400	376.72
10	8	(88, 20, 10)	17600	937.61	(88, 20, 10)	17600	386.37
11	8	(78, 20, 10)	15600	703.22	(78, 20, 10)	15600	316.59
12	8	(45, 23, 10)	10350	903.98	(53, 20, 10)	10600	356.51
Average		—	10155	717.41	—	10100	294.19

orders in the test cases increased, the computation time of the GA algorithm showed an upward trend, while the growth of computation time for the LNSGA algorithm was more gradual. This once again validated the computational efficiency advantage of the LNSGA algorithm when dealing with large-scale complex orders.

Thus, the LNSGA algorithm obtained superior packing solutions by introducing lower neighborhood search while ensuring computational efficiency. Even in complex environments with many orders, integrating genetic algorithm and local search in LNSGA still demonstrated significant effectiveness.

5. Conclusions and Prospects

5.1. Conclusion. In this paper, we introduced a novel metaheuristic approach to address the multiorder open-dimension 3D rectangular packing problem (3D-MOSB-ODRPP), a complex challenge exacerbated by the rapid growth of e-commerce. The proposed algorithm, a hybrid of a genetic algorithm and the Gurobi solver, effectively manages the complexity of the problem by combining global exploration and local optimization. The algorithm is underpinned by a mathematical model that accurately represents the multiorder open-dimension packing scenario and an enhanced genetic algorithm incorporating a lower neighborhood search strategy.

Extensive comparative experiments validate the effectiveness of the proposed approach. The LNSGA algorithm consistently outperforms the commercial solver Gurobi and the traditional genetic algorithm (GA) regarding both solution quality and computational efficiency. For small-scale instances, the LNSGA algorithm achieves optimal values in most test cases, with minimal deviation from the optimal values as the number of orders increases. For large-scale instances, the LNSGA algorithm demonstrates significant optimization improvements over Gurobi, with an average improvement of 50.16%. Compared to the GA, the LNSGA

algorithm achieves a significant improvement in the objective function, ranging from 34.05% to 71.93%, while reducing the runtime by more than 50%.

The superior performance of the LNSGA algorithm can be attributed to its effective integration of the lower neighborhood search mechanism and the Gurobi solver. The targeted lower neighborhood search efficiently explores the packing box dimensions, avoiding unnecessary calculations and enhancing computational efficiency. The Gurobi solver, invoked after each local neighborhood search, ensures globally optimal packing solutions. This hybrid approach achieves an organic integration of global exploration and local search, enabling the algorithm to escape from local optima and obtain high-quality global optimal solutions.

Thus, combining a genetic algorithm with the Gurobi solver and incorporating a lower neighborhood search strategy, the proposed metaheuristic approach demonstrates significant effectiveness in tackling the complex 3D-MOSB-ODRPP. The approach addresses the current complex scenarios in e-commerce warehousing and sets a foundation for future research in the field of 3D packing optimization.

5.2. Management Insights. The findings of this study offer valuable insights for managers in e-commerce warehousing and logistics operations. The proposed metaheuristic approach provides an effective tool for optimizing the packing process, enabling managers to make informed decisions that maximize space utilization and minimize logistics costs.

First, the multiorder open-dimension packing model allows managers to simultaneously consider the varying requirements of different orders. By optimizing the packing of multiple orders into a single, size-adjustable container, managers can improve order consolidation and reduce the number of containers required, leading to significant cost savings in transportation and storage.

Second, the lower neighborhood search strategy incorporated in the enhanced genetic algorithm offers a practical approach for managers to explore optimal packing configurations. By adjusting the container's length, width, and height within a specific range, managers can identify the most suitable container dimensions that balance the demands of different orders while maximizing space utilization.

Third, integrating the Gurobi solver with the mathematical approach ensures that the packing solutions obtained are globally optimal. This gives managers confidence that the recommended packing configurations are the best possible solutions, considering all relevant constraints and objectives.

Furthermore, the comparative experiments demonstrate the superior performance of the LNSGA algorithm over traditional methods, such as the commercial solver Gurobi and the pure genetic algorithm. Managers can leverage these findings to justify adopting the proposed approach in their operations, as it significantly improves both solution quality and computational efficiency.

5.3. Limitations and Future Work. While the proposed metaheuristic approach effectively addresses the 3D-MOSB-ODRPP, several limitations warrant attention in future research.

Primarily, the study focuses on rectangular items and containers, simplifying real-world packing scenarios. Future research could enhance the approach to handle irregular shapes and additional constraints.

In addition, computational experiments use randomly generated datasets, limiting real-world applicability. Future work should validate performance using real-world e-commerce data to assess scalability and robustness.

Furthermore, extending the approach to dynamic and online packing scenarios is essential for real-time operations, necessitating adaptive algorithms.

Integrating sustainability considerations, such as minimizing environmental impact, is a promising avenue for future research, enhancing the social responsibility of packing optimization.

Lastly, exploring applications beyond e-commerce, such as manufacturing and logistics, could leverage the approach's versatility to solve optimization challenges in diverse industries.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Authors' Contributions

J.Y. acquired the funding and wrote the original draft. H.L. and L.Z. supervised the study. K.L. validated the study. M.S. and L.K wrote, reviewed, and edited the study. All authors have read and agreed to the published version of the manuscript. Jianglong Yang and Huwei Liu contributed equally to this work and are co-first authors.

Acknowledgments

This study was supported by the Beijing Wuzi University Youth Research Fund (Project name: Research on Intelligent e-commerce Unmanned Warehouse Packing and Loading Optimization Strategy and Green Recycling Mode; Project no. 2023XJQN14), in part by the Beijing Municipal Education Commission Social Science General Project (Project name: Research on the Recycling and Sharing Mode of e-commerce Express Packaging Boxes in the Context of Digitalization in China; Project no. SM202410037007).

References

[1] D. W. Alexander and R. Merkert, "Applications of gravity models to evaluate and forecast us international air freight markets post-gfc," *Transport Policy*, vol. 104, pp. 52–62, 2021.

[2] Xinhua, "China's postal industry to see steady growth in 2022," 2023, https://english.www.gov.cn/statecouncil/ministries/202201/09/content_WS61dadf7cc6d09c94e48a35d5.html.

[3] K. Verghese and H. Lewis, "Environmental innovation in industrial packaging: a supply chain approach," *International Journal of Production Research*, vol. 45, no. 18-19, pp. 4381–4401, 2007.

[4] L. Few, "A mixed packing problem," *Mathematika*, vol. 7, no. 1, pp. 56–63, 1960.

[5] P. C. Gilmore and R. E. Gomory, "Multistage cutting stock problems of two and more dimensions," *Operations Research*, vol. 13, no. 1, pp. 94–120, 1965.

[6] J. E. Beasley, "Algorithms for unconstrained two-dimensional guillotine cutting," *Journal of the Operational Research Society*, vol. 36, no. 4, pp. 297–306, 1985.

[7] J. Y. Shiau and M. C. Lee, "A warehouse management system with sequential picking for multi-container deliveries," *Computers and Industrial Engineering*, vol. 58, no. 3, pp. 382–392, 2010.

[8] G. Cintra, F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier, "Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation," *European Journal of Operational Research*, vol. 191, no. 1, pp. 61–85, 2008.

[9] S. S. Yeo, K. Chen, and H. Liu, "Pattern recognition technologies for multimedia information processing," *Multimedia Tools and Applications*, vol. 74, no. 1, pp. 179–183, 2015.

[10] L. Junqueira and R. Morabito, "On solving three-dimensional open-dimension rectangular packing problems," *Engineering Optimization*, vol. 49, no. 5, pp. 733–745, 2017.

[11] J. Verstichel, P. De Causmaecker, F. C. R. Spieksma, and G. Vanden Berghe, "Exact and heuristic methods for placing ships in locks," *European Journal of Operational Research*, vol. 235, no. 2, pp. 387–398, 2014.

[12] S. C. H. Leung, D. Zhang, C. Zhou, and T. Wu, "A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem," *Computers and Operations Research*, vol. 39, no. 1, pp. 64–73, 2012.

[13] M. K. Omar and K. Ramakrishnan, "Solving non-oriented two dimensional bin packing problem using evolutionary particle swarm optimisation," *International Journal of Production Research*, vol. 51, no. 20, pp. 6002–6016, 2013.

[14] S. H. Song, "An integrated formulation for hierarchical cast design problems in the steel making industry," *International Journal of Production Research*, vol. 52, no. 5, pp. 1443–1454, 2014.

[15] M. Chen, J. Huo, and Y. Duan, "A hybrid biogeography-based optimization algorithm for three-dimensional bin size designing and packing problem," *Computers and Industrial Engineering*, vol. 180, Article ID 109239, 2023.

[16] W. Zhu, Z. Zhang, W.-C. Oon, and A. Lim, "Space defragmentation for packing problems," *European Journal of Operational Research*, vol. 222, no. 3, pp. 452–463, 2012.

[17] X. Li and K. Zhang, "A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers," *Computers and Industrial Engineering*, vol. 90, pp. 305–313, 2015.

[18] C. Li, G. Chen, G. Liang, F. Luo, J. Zhao, and Z. Y. Dong, "Integrated optimization algorithm: a metaheuristic approach for complicated optimization," *Information Sciences*, vol. 586, pp. 424–449, 2022.

[19] N. Cheimanoff, P. Féniès, M. N. Kitri, and N. Tchernev, "Exact and metaheuristic approaches to solve the integrated production scheduling, berth allocation and storage yard allocation problem," *Computers and Operations Research*, vol. 153, Article ID 106174, 2023.

[20] X. Chen, R. Bai, R. Qu, J. Dong, and Y. Jin, "Deep reinforcement learning assisted genetic programming ensemble hyper-heuristics for dynamic scheduling of container port trucks," *IEEE Transactions on Evolutionary Computation*, vol. 1, 2024.

[21] J. Lehmann and M. Winkenbach, "A matheuristic for the two-echelon multi-trip vehicle routing problem with mixed pickup and delivery demand and time windows," *Transportation Research Part C: Emerging Technologies*, vol. 160, Article ID 104522, 2024.

[22] A. Maroof, B. Ayvaz, and K. Naeem, "Logistics optimization using hybrid genetic algorithm (hga): a solution to the vehicle routing problem with time windows (vrptw)," *IEEE Access*, vol. 12, pp. 36974–36989, 2024.

[23] E. da Silva, A. A. S. Leão, F. Toledo, and T. Wauters, "A matheuristic framework for the three-dimensional single large object placement problem with practical constraints," *Computers and Operations Research*, vol. 124, Article ID 105058, 2020.

[24] A. A. Ananno and L. Ribeiro, "A multi-heuristic algorithm for multi-container 3-d bin packing problem optimization using real world constraints," *IEEE Access*, vol. 12, pp. 42105–42130, 2024.

[25] D. Pisinger, "Heuristics for the container loading problem," *European Journal of Operational Research*, vol. 141, no. 2, pp. 382–392, 2002.

[26] S. Elhedhli, F. Gzara, and B. Yildiz, "Three-dimensional bin packing and mixed-case palletization," *INFORMS Journal on Optimization*, vol. 1, no. 4, pp. 323–352, 2019.

[27] F. Gzara, S. Elhedhli, and B. C. Yildiz, "The pallet loading problem: three-dimensional bin packing with practical constraints," *European Journal of Operational Research*, vol. 287, no. 3, pp. 1062–1074, 2020.

[28] L. Wei, Q. Hu, A. Lim, and Q. Liu, "A best-fit branch-and-bound heuristic for the unconstrained two-dimensional non-guillotine cutting problem," *European Journal of Operational Research*, vol. 270, no. 2, pp. 448–474, 2018.

[29] J. A. Bennell and J. F. Oliveira, "The geometry of nesting problems: a tutorial," *European Journal of Operational Research*, vol. 184, no. 2, pp. 397–415, 2008.

[30] A. K. Sato, T. C. Martins, A. M. Gomes, and M. S. G. Tsuzuki, "Raster penetration map applied to the irregular packing problem," *European Journal of Operational Research*, vol. 279, no. 2, pp. 657–671, 2019.

[31] L. R. Mundim, M. Andretta, and T. A. de Queiroz, "A biased random key genetic algorithm for open dimension nesting problems using no-fit raster," *Expert Systems with Applications*, vol. 81, pp. 358–371, 2017.

[32] H. Wu, S. C. H. Leung, Y. w Si, D. Zhang, and A. Lin, "Three-stage heuristic algorithm for three-dimensional irregular packing problem," *Applied Mathematical Modelling*, vol. 41, pp. 431–444, 2017.

[33] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3d bin packing problem with deep reinforcement learning method," *arXiv preprint arXiv:1708.05930*, 2017.

[34] J. F. Gonçalves and M. G. C. Resende, "A biased random key genetic algorithm for 2d and 3d bin packing problems," *International Journal of Production Economics*, vol. 145, no. 2, pp. 500–510, 2013.

[35] Y. H. Huang, F. J. Hwang, and H. C. Lu, "An effective placement method for the single container loading problem," *Computers and Industrial Engineering*, vol. 97, pp. 212–221, 2016.

[36] J. Correcher, M. T. Alonso, F. Parreño, and R. Alvarez-Valdés, "Solving a large multicontainer loading problem in the car manufacturing industry," *Computers and Operations Research*, vol. 82, pp. 139–152, 2017.

[37] M. T. Alonso, R. Alvarez-Valdés, M. Iori, and F. Parreño, "Mathematical models for multi container loading problems with practical constraints," *Computers and Industrial Engineering*, vol. 127, pp. 722–733, 2019.

[38] T. Romanova, J. Bennell, Y. Stoyan, and A. Pankratov, "Packing of concave polyhedra with continuous rotations using nonlinear optimisation," *European Journal of Operational Research*, vol. 268, no. 1, pp. 37–53, 2018.

[39] M. Verkhoturov, A. Petunin, G. Verkhoturova, K. Danilov, and D. Kurennov, "The 3d object packing problem into a parallelepiped container based on discrete-logical representation," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 1–5, 2016.

[40] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961.

[41] L. Wei and A. Lim, "A bidirectional building approach for the 2d constrained guillotine knapsack packing problem," *European Journal of Operational Research*, vol. 242, no. 1, pp. 63–71, 2015.

[42] L. Wei, T. Tian, W. Zhu, and A. Lim, "A block-based layer building approach for the 2d guillotine strip packing problem," *European Journal of Operational Research*, vol. 239, no. 1, pp. 58–69, 2014.

[43] F. M. B. Toledo, M. A. Carravilla, C. Ribeiro, J. F. Oliveira, and A. M. Gomes, "The dotted-board model: a new mip model for nesting irregular shapes," *International Journal of Production Economics*, vol. 145, no. 2, pp. 478–487, 2013.

[44] D. Zhang, L. Shi, S. C. H. Leung, and T. Wu, "A priority heuristic for the guillotine rectangular packing problem," *Information Processing Letters*, vol. 116, no. 1, pp. 15–21, 2016.

[45] S. Ali, A. G. Ramos, M. A. Carravilla, and J. F. Oliveira, "Online three-dimensional packing problems: a review of off-line and on-line solution approaches," *Computers and Industrial Engineering*, vol. 168, Article ID 108122, 2022.

[46] Q. Que, F. Yang, and D. Zhang, "Solving 3d packing problem using transformer network and reinforcement learning," *Expert Systems with Applications*, vol. 214, Article ID 119153, 2023.

[47] A. Moura, T. Pinto, C. Alves, and J. Valério de Carvalho, "A matheuristic approach to the integration of three-dimensional bin packing problem and vehicle routing problem with simultaneous delivery and pickup," *Mathematics*, vol. 11, no. 3, p. 713, 2023.

[48] A. Lim, B. Rodrigues, and Y. Wang, "A multi-faced buildup algorithm for three-dimensional packing problems," *Omega*, vol. 31, no. 6, pp. 471–481, 2003.

[49] A. Lim, B. Rodrigues, and Y. Yang, "3-d container packing heuristics," *Applied Intelligence*, vol. 22, no. 2, pp. 125–134, 2005.

[50] Z. Shen, J. Song, K. Mittal, and S. Gupta, "Ct-cpp: coverage path planning for 3d terrain reconstruction using dynamic coverage trees," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 135–142, 2022.

[51] E. F. Silva, T. A. M. Toffolo, and T. Wauters, "Exact methods for three-dimensional cutting and packing: a comparative study concerning single container problems," *Computers and Operations Research*, vol. 109, pp. 12–27, 2019.

[52] H. Iwasawa, Y. Hu, H. Hashimoto, S. Imahori, and M. Yagiura, "A heuristic algorithm for the container loading problem with complex loading constraints," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 10, no. 3, 2016.

[53] A. G. Ramos, E. Silva, and J. F. Oliveira, "A new load balance methodology for container loading problem in road transportation," *European Journal of Operational Research*, vol. 266, no. 3, pp. 1140–1152, 2018.

[54] H. Gezici and H. Livatyali, "An improved Harris hawks optimization algorithm for continuous and discrete optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 113, Article ID 104952, 2022.

[55] H. Xiong, K. Ding, W. Ding, J. Peng, and J. Xu, "Towards reliable robot packing system based on deep reinforcement learning," *Advanced Engineering Informatics*, vol. 57, Article ID 102028, 2023.

[56] A. El Yaagoubi, M. Charhbili, J. Boukachour, and A. El Hilali Alaoui, "Multi-objective optimization of the 3d container stowage planning problem in a barge convoy system," *Computers and Operations Research*, vol. 144, Article ID 105796, 2022.

[57] H. Liu, L. Zhou, J. Yang, and J. Zhao, "The 3d bin packing problem for multiple boxes and irregular items based on deep q-network," *Applied Intelligence*, vol. 25, 2023.

[58] X. Wang, J. D. Thomas, R. J. Piechocki, S. Kapoor, R. Santos-Rodríguez, and A. Parekh, "Self-play learning strategies for resource assignment in open-ran networks," *Computer Networks*, vol. 206, Article ID 108682, 2022.

[59] S. Erbayrak, V. Özkır, and U. Mahir Yıldırım, "Multi-objective 3d bin packing problem with load balance and product family concerns," *Computers and Industrial Engineering*, vol. 159, Article ID 107518, 2021.

[60] S. Polyakovskiy and R. M'Hallah, "Just-in-time two-dimensional bin packing," *Omega*, vol. 102, Article ID 102311, 2021.

[61] H. Zhao, C. Zhu, X. Xu, H. Huang, and K. Xu, "Learning practically feasible policies for online 3d bin packing," *Science China Information Sciences*, vol. 65, no. 1, Article ID 112105, 2022.

[62] S. Berndt, K. Jansen, and K. M. Klein, "Fully dynamic bin packing revisited," *Mathematical Programming*, vol. 179, no. 1-2, pp. 109–155, 2020.

[63] M. Witteman, Q. Deng, and B. F. Santos, "A bin packing approach to solve the aircraft maintenance task allocation problem," *European Journal of Operational Research*, vol. 294, no. 1, pp. 365–376, 2021.

[64] A. Ekici, "Variable-sized bin packing problem with conflicts and item fragmentation," *Computers and Industrial Engineering*, vol. 163, Article ID 107844, 2022.

[65] R. Ding, B. Deng, and W. Li, "Meta-heuristic algorithms for the generalized extensible bin packing problem with overload cost," *IEEE Access*, vol. 10, pp. 124858–124873, 2022.

[66] J. Yang, K. Liang, H. Liu et al., "Optimizing e-commerce warehousing through open dimension management in a three-dimensional bin packing system," *PeerJ Computer Science*, vol. 9, Article ID e1613, 2023.

[67] K. Liang, J. Yang, M. Shan, L. Kong, and H. Liu, "A customizable optimization model for green e-commerce packing considering multiple orders and diverse box types," *Journal of Cleaner Production*, vol. 444, Article ID 141249, 2024.