Hindawi
Complexity
Volume 2018, Article ID 2024184, 12 pages
https://doi.org/10.1155/2018/2024184

Research Article

Optimization of Heterogeneous Container Loading Problem with Adaptive Genetic Algorithm

Xianbo Xiang[1,2] **Caoyang Yu**[1] **He Xu**[3] **and Stuart X. Zhu**[4]

[1]*School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China*
[3]*School of Management, Huazhong University of Science and Technology, Wuhan 430074, China*
[4]*Department of Operations, University of Groningen, P.O. Box 800, 9700 AV Groningen, Netherlands*

Correspondence should be addressed to Xianbo Xiang; xbxiang@mail.hust.edu.cn and Stuart X. Zhu; x.zhu@rug.nl

Received 4 August 2018; Accepted 18 September 2018; Published 1 November 2018

Guest Editor: Andy Annamalai

This paper studies an optimized container loading problem with the goal of maximizing the 3D space utilization. Based on the characteristics of the mathematical loading model, we develop a dedicated placement heuristic integrated with a novel dynamic space division method, which enables the design of the adaptive genetic algorithm in order to maximize the loading space utilization. We use both weakly and strongly heterogeneous loading data to test the proposed algorithm. By choosing 15 classic sets of test data given by Loh and Nee as weakly heterogeneous data, the average space utilization of our algorithm reaching 70.62% outperforms those of 13 algorithms from the related literature. Taking a set of test data given by George and Robinson as strongly heterogeneous data, the space utilization in this paper can be improved by 4.42% in comparison with their heuristic algorithm.

1. Introduction

Container loading problems mainly address the issues of planning the loading order and loading position on the basis of ensuring certain constraints [1–3]. Study on optimization of container loading problems has an extensive engineering background and various applications on the modern management of container terminals and container shipping [4–6]. The container loading problem is a Nondeterministic Polynomial- (NP-) Hard problem that focuses on establishing mathematical models and seeking efficient algorithms depending on the specific environments [7, 8]. Depending on the types of containers and cargoes, the container loading problem can be divided into 14 categories [9]. Our paper studies two categories of loading a single container with selections from either weakly or strongly heterogeneous set of cargoes such that the value of the loaded items is maximized.

Intelligent optimized algorithms, such as simulated annealing, tabu search algorithm, and genetic algorithm, have been proposed in literature, in order to compute the optimal solution of the complex NP-hard problem including the practical case of container loading [10, 11]. The existing algorithms can be divided into two categories: placement heuristic and improved heuristic. The placement heuristic is also known as the basic heuristic, which establishes direct search strategies and search rules with practical experience and search solutions based on these rules. The basic heuristic has a high practical value for the optimization of the packing problem. Although it cannot guarantee the optimal solution, it can usually obtain a satisfactory feasible solution. The improved heuristic is a hybrid algorithm which combines basic heuristic with neighborhood search algorithm, such as genetic algorithm, tabu search algorithm, and greedy algorithm [12–14].

The representative placement heuristic was proposed by George and Robinson who firstly introduced the concept of layers [15]. Based on the algorithm of George and Robinson, Bischoff and Marriott constructed 14 heuristics by mixing 6 sequencing rules and 3 fill methods that already existed [16]. Loh and Nee studied a heuristic for problem of weakly heterogeneous, by taking the charge density as the objective function, building horizontal layers and loading from bottom

to top [17]. In addition, they designed 15 sets of test data which was often used as a classic test data for later algorithms. Ngoi et al. proposed a heuristic algorithm that the cargoes can only be loaded in the horizontal direction of rotation [18]. They abandoned the concept of layers and created a special method of matrix representation of space which simplified the loading steps.

Bischoff and Ratcliff proposed a heuristic with multiple destination constraints where the cargoes were ordered in accordance with the arrival order [19]. This algorithm did not build layers but columns in order to load similar cargoes in the same column space. They also used the matrix space representation of Nogi et al. [18]. Gehring and Bortfeldt proposed a typical hybrid genetic algorithm and introduced two-dimensional loading problem into three-dimensional loading problem [20]. Bortfeldt and Gehring proposed a tabu search algorithm for solving container loading problem considering stability constraints, rotation constraints, stack constraints, and weight constraints [10]. They have created two composite block loading methods; one block contained only the same cargoes, and the other block contained two types of cargoes. Similar to George and Robinson's idea of building walls, Chien and Wu used a tree search strategy to find the best loading plan [21]. Each node of the tree is a set of walls depths; its child nodes are corresponding to the strips' widths. Bortfeldt and Gehring's hybrid genetic algorithm combined greedy algorithm with vertical layers [13]. In addition to stability constraints, rotation constraints, stack constraints, and weight constraints, they also considered balance constraints.

Chien and Deng [22] proposed a heuristic similar to the matrix space representation of Nogi and coworkers [18]. Their heuristic divides a loading space into two subspaces and then searches the suitable subspace for the current cargo to load. Moura and Oliveira [23] carried out two kinds of deformation based on George and Robinson's method. The first deformation introduced the stability constraint. In the second deformation, the width of the new layers must be less than the width of the old one, which was convenient to merge the loading space and improve the loading stability. Lim et al. proposed two kinds of heuristics for homogeneous packing and heterogeneous packing [24]. The heuristic of Wang et al. was designed for a special kind of dynamic space decomposition method based on the trigeminal tree [25]. Wu et al. used two segments of encoding in genetic algorithm, including the number and the rotation of the cargoes [26]. Their algorithm applied simultaneous genetic operation to both segments of the encoding.

He and Huang designed a caving degree based flake arrangement to solve the packing problem [27], which constructed the flake cargoes block consisting of the same cargoes. In [28], Dereli et al.'s hybrid bee(s) algorithm inspired by the bees foraging behavior was similar to genetic algorithm where the optimal parents and some offspring were reserved to the next generation. The algorithm considered the rotation constraints of the cargoes and the LDB concept. Tian et al. considered the cargo transport priority and structured tree search algorithm based on the greedy heuristic so that blocks of similar cargoes were loaded into a container [29]. There were five evaluation functions to select the blocks which constructed the search tree branches. Lim et al. designed single container and multicontainer loading algorithms for a practical storage management system, using the dynamic sequencing method to determine the priority of cargoes [30]. In the single container loading problem, the cargoes with a high priority are placed near the bottom of the container location. In the multicontainer loading problem, the cargoes with a high priority are placed in the higher priority container. Goncalves and Resende's genetic algorithm adopted two-segment encoding of cargo loading sequence and position [31]. The cargo loading sequence encoding used a biased random key strategy in order to avoid the correction of the sequence encoding.

In this paper, we propose a novel adaptive genetic algorithm integrating two-stage real-number encoding method and dynamic space division method to improve the existing placement heuristic, which effectively avoids a certain amount of space loss. The novelty of the proposed method is described as follows. (1) According to the characteristics of the loading problem, a two-stage real-number encoding method is developed, where the priority of the cargoes is designed as the first half of encoding and then the placement state is designed as the second half so that the algorithm can search for the optimal solution in large search sets. (2) Dynamic adaptive crossover and mutation operators are designed in the genetic algorithm in order to avoid the bad convergence caused by coding changes. More importantly, we use both weakly and strongly heterogeneous loading data to test the proposed algorithm. By choosing 15 classic sets of test data in [17] as weakly heterogeneous data, the average space utilization reaching 70.62% outperforms those of 13 algorithms from the related literature. Taking a set of test data in [15] as strongly heterogeneous data, the space utilization can be improved by 4.42% in comparison with [15].

The rest of the paper is organized as follows. The next section provides a description ofthe general container problem model, constraints and objective functions. Sections 3 and 4 discuss the proposed algorithm, including the placement heuristic (how a layout is constructed) and the improved heuristic (how to search for better solutions). Experimental results, along with benchmark data sets, are analyzed in Section 5. Conclusions are given in Section 6.

2. Problem Statement

Given a container and a set of boxes, the container loading problem aims to determine the load position and place rotation in the container in order to maximize the space utilization with basic geometric constraints [32]. This paper addresses the problem of loading a single container with the following assumptions [28, 33, 34]:

(i) Each cargo can be loaded into a container

(ii) The cargoes do not have a priority during the loading procedure

(iii) The shape of the cargoes is regular rectangle

(iv) Fillers are used to fill the gap between the cargoes to ensure the loading stability

(v) The placement of the cargoes must be parallel or orthogonal to the container walls

(vi) The placement of cargoes can be arbitrarily rotated

Our model uses a three-dimensional Cartesian coordinate system, as shown in Figure 1. In the coordinate system, the x-axis represents the length of the container, the y-axis represents the width, and the z-axis represents the height. The origin point represents the left corner of the container.

The relevant parameters of the model are as follows:

(i) L = the length of the container

(ii) W = the width of the container

(iii) H = the height of the container

(iv) i = the ID number of the cargo type

(v) n_i = the quantity of the type-i cargo

(vi) l_i = the length of the type-i cargo

(vii) w_i = the width of the type-i cargo

(viii) h_i = the height of the type-i cargo

The decision variables of the model are the loaded number of the type i cargo denoted by m_i. Our paper focuses on maximizing the space utilization of the container. The objective function of the model is given by

$$Maximize \quad F = \frac{\sum_i m_i l_i w_i h_i}{LWH} \times 100\% \tag{1}$$

To appropriately model the process of loading container, we require that the feasible solution satisfies the basic geometric constraints; i.e., all the cargoes must be fully loaded in the container and the overlap is not allowed. The basic geometric constraints can be expressed as follows:

$$s.t. \quad \sum_i m_i l_i w_i h_i \le LWH \tag{2}$$

$$0 \le m_i \le n_i \tag{3}$$

In the following sections, we first propose a placement heuristic integrated with a novel dynamic space division method by examining the characteristics of the mathematical model. Then, a dedicated adaptive genetic algorithm is designed in order to optimize the loading space utilization based on the characteristics of the heuristic.

3. The Placement Heuristic

In this paper, our proposed placement heuristic improves George and Robinson's algorithm [15]. In our heuristic, the same type of cargoes with the same rotation is put together in order to avoid some gaps and improve the loading efficiency. We also use a variety space distribution method based on different loading location, which efficiently avoids some loss of loading space. Remaining spaces are merged with abandoned spaces so that the abandoned spaces are reused and the space utilization is improved. There are three rules to realize the improved placement heuristic. In the following parts, these three rules are adopted to build the placement heuristic.

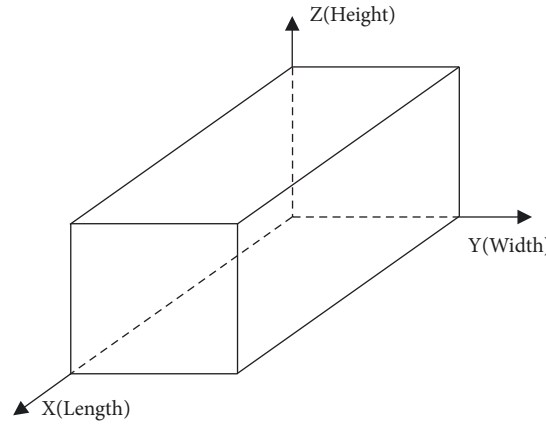

Figure 1: Coordinate system.

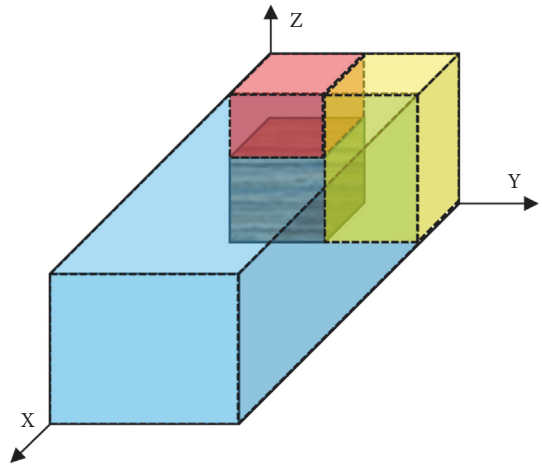

Figure 2: Z-Y-X division mode.

3.1. Placement Rule. As shown in Figure 1, this coordinate system has been built. This placement rule takes the corner strategy; i.e., the cargoes are firstly placed near the origin point. Without constraints of loading precedence, the loading order is flexible.

3.2. Space Division Rule. The optimization of container loading is mainly reflected by the space division rule. After the cargo is loaded into the left corner of the current space, three spaces are formed as shown in Figure 2, including the up space, the right space, and the front space. The following cargoes are firstly loaded in the up space, then the right space, and last the front space. Robinson and George algorithm introduces the concept of "layer". They use YZ layers. The YZ layers are layers parallel to the YZ plane. Each loading forms three spaces and cargoes are loaded according to the Z-Y-X traversing order until all the remaining cargoes cannot

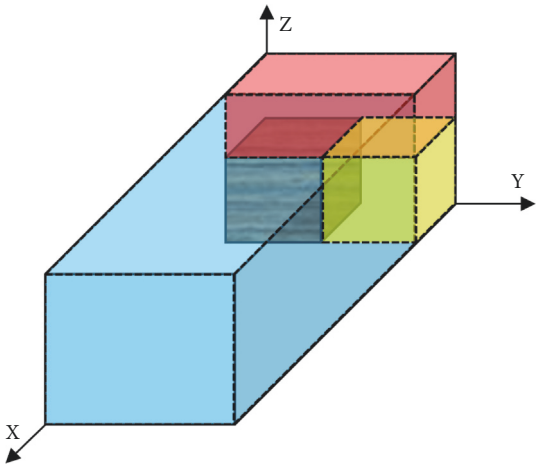

Figure 3: Y-Z-X division mode.

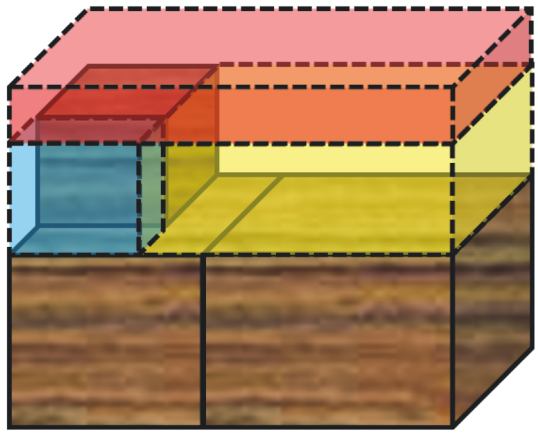

Figure 4: X-Y-Z division mode.

be placed in any of the remaining spaces, where X means the front space, Y means the right space, and Z means the up space.

Similar to George and Robinson's algorithm, we also build the similar YZ layers and build Y ties in every layer in which the Y ties are parallel to Y-axis. The space division rule is applied based on the following three cases.

Case 1. The loading cargo is the first cargo both in a layer and in a tie.

Case 2. The loading cargo is not the first cargo in a layer but the first cargo in a tie.

Case 3. The loading cargo is neither the first cargo in a layer nor the first cargo in a tie.

In the first case, we divide the current space into the right space, the up space, and the front space, as shown in Figure 3. In the second case, we divide the current space into the front space, the right space, and the up space, as shown in Figure 4. In the third case, we divide the current space into the front space, the up space, and the right space, as shown in Figure 5.

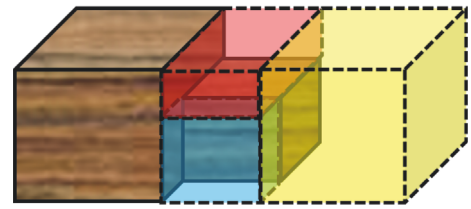

Figure 5: X-Z-Y division mode.

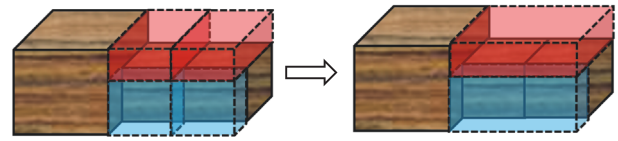

Figure 6: Right-and-left space consolidation.

In fact, the space is often divided by the Y-Z-X division mode shown in Figure 3. However, in the second case, the length of the current space is determined by the first loaded cargo in the layer. As the space length is small, the divided front space may be narrow. As a consequence, such a narrow front space could be treated as an abandoned space that cannot load any cargo. In the third case, the length of the current space is determined by the first loaded cargo in the layer, and the width is determined by the first loaded cargo in the tie. As both the space length and the space width are small, the divided front space and up space could be narrow. The front space and up space could easily be abandoned spaces. In this paper, the narrow space is regarded as an abandoned space in advance. The volume of available space can be increased by adopting the above space division rule in order to improve the overall space utilization.

3.3. Space Consolidation Rule. If there are abandoned spaces before each loading, we combine the current space with the abandoned spaces so that the abandoned spaces can be used again. There are three directions of space consolidation, including right-and-left space consolidation in Figure 6, up-and-down space consolidation in Figure 7, and front-and-back space consolidation Figure 8. The consolidation steps are given as follows.

Step 1. Before loading, search the abandoned spaces.

Step 2. Determine whether the abandoned space and the current space are in the same XZ plane, in the XY plane, or in the YZ plane in turn.

Step 3. If they are in the same plane, combine them as shown in Figures 6–8.

Step 4. Use the combined space as the new current loading space.

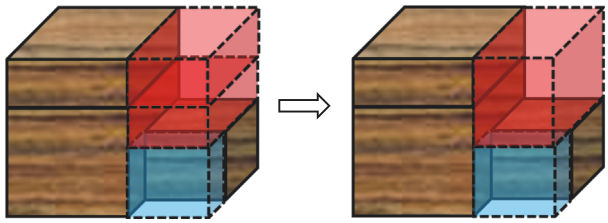

Figure 7: Up-and-down space consolidation.

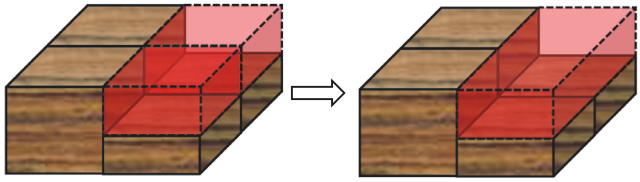

Figure 8: Front-and-back space consolidation.

4. Adaptive Genetic Algorithm

This section gives an improved genetic algorithm, which does not define the cargoes' priority in the placement heuristic but chooses the priority as the first half of encoding and then introduces the placement state as the second half so that it is able to search the optimal solution in a larger search space. In addition, dynamic adaptive crossover and mutation operators are designed in the genetic algorithm to avoid the bad convergence caused by coding changes. In this sense, the proposed adaptive genetic algorithm is applicable for both weakly heterogeneous problem and strongly heterogeneous problem. The adaptive genetic algorithm is constructed by four steps, including the encoding and decoding, fitness function and selection, adaptive crossover operator and mutation operator, and the optimal preservation strategy.

4.1. Encoding and Decoding. Suppose that there are k types of cargoes to be loaded, and the quantity of each cargo is n_i, for $i \in \{1, 2, \ldots, k\}$. A two-stage coding method composed of the sequence and placement state of the cargoes is adopted in this encoding and the length of the encoding is $2n$. The previous n bits show the sequence of the cargoes, and the latter n bits indicate the placement state of the cargoes. A typical encoding is expressed as follows:

$$P = \{s_1, s_2, \ldots, s_k, s_{k+1}, s_{k+2}, \ldots, s_{2k}\} \tag{4}$$

where $s_1, s_2, \ldots, s_k$ represent the loading sequence of the cargoes and $s_{k+1}, s_{k+2}, \ldots, s_{2k} \in \{1, 2, 3, 4, 5, 6\}$ indicates the corresponding types of rotation of $s_1, s_2, \ldots, s_k$. In order to reduce the amount of the wasted space, the same cargoes are placed in the same placement state defined in Table 1. For example, the number of the rotations of cargo i is 1, which indicates the length, the width, and the height of the cargo are placed in parallel to that of the container; i.e., $x_i \| X, y_i \| Y, z_i \| Z$. Since the placement of the cargo can be arbitrarily rotated as assumed in Section 2, 6 kinds of the placement state of the overall space are formed and shown in Table 1.

Table 1: Encoding values of placement state.

Encoding value	Edge Parallel to X	Edge Parallel to Y	Edge Parallel to Z
1	x_i	y_i	z_i
2	y_i	x_i	z_i
3	y_i	z_i	x_i
4	z_i	y_i	x_i
5	z_i	x_i	y_i
6	x_i	z_i	y_i

Note that the encoding values determine the sequence and rotation. The placement heuristic rule and space division rule determine the position of the cargo placement. Therefore, we can obtain the solution of loading operations by decoding, which is the process of converting encoding into loading results. In decoding, we set the number of loaded cargoes s_i as m_i, which is expressed as follows:

$$Q = \left\{ \underbrace{s_1, s_1, \ldots, s_1}_{m_1}, \underbrace{s_2, s_2, \ldots, s_2}_{m_2}, \ldots, \underbrace{s_k, s_k, \ldots, s_k}_{m_k} \right\} \tag{5}$$

4.2. Fitness Function and Selection. Generally, the fitness function of genetic algorithm is determined by the objective function. In this paper, the fitness function refers to the space utilization of the objective function, which is defined as follows:

$$F = \frac{\sum_i m_i x_i y_i z_i}{XYZ} \times 100\% \tag{6}$$

Roulette-wheel-selection in [35] is chosen as the selection method of this paper. The popular size is set as M, and the parental generation is $\{P_1, P_2, P_3, \ldots, P_M\}$. The steps of Roulette algorithm are described as follows.

Step 1. Calculate the fitness of each individual in the group.

Step 2. Calculate the relative fitness of each individual, which is the probability to be selected as parental generation. The probability of individual j is defined as

$$p_{js} = \frac{F_j}{\sum_j F_j} \tag{7}$$

Step 3. Calculate the cumulative probability of each individual; i.e.,

$$p_j = \begin{cases} p_{js}, & j = 1 \\ p_{j-1} + p_{js}, & j = 2, 3, \ldots \end{cases} \tag{8}$$

Step 4. Generate a random number between 0 and 1 and then determine the range of it. If the number is in $[p_{j-1}, p_j]$, then P_j is selected, which is shown in Figure 9.

Step 5. Repeat Step 4 until M individuals are generated, which makes up the new group $\{P'_1, P'_2, P'_3, \ldots, P'_M\}$.

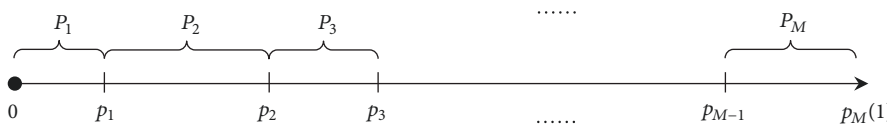

Figure 9: Roulette algorithm.

4.3. Adaptive Crossover Operator and Mutation Operator. Next, we propose an adaptive genetic algorithm so that the crossover probability and mutation probability can be adjusted dynamically. When the fitness value of the population tends to be consistent or local optimum, the crossover probability and mutation probability increase, and then the population diversity increases. However, when the fitness value of the population is relatively dispersed, the crossover probability and mutation probability decrease in order to prevent the diversity of the population from being destroyed. Furthermore, for the individuals with higher fitness values than the average values of the population, the crossover probability and mutation probability should be set smaller in order to protect the solution and go into the next generation. For the individuals with lower fitness values, the crossover probability and mutation probability should be set larger so that the probability of being knocked out of the solution increases.

The crossover operator is carried out on the encoding of adaptive genetic algorithm. Since we use a two-stage encoding and the characteristic of two-stage encoding is different from each other, their crossover operators are different. Partial mapped crossover operator method is used in the first half of this encoding, and two-point crossover operator is chosen for the second half. In general, the recommended probability of crossover is in the interval $[0.4, 0.99]$ [36]. The crossover probability used in this paper is described as follows:

$$P_c = \begin{cases} P_{c1}, & f' < f_{avg} \\ P_{c1} - \dfrac{(P_{c1} - P_{c2}) \times (f' - f_{avg})}{(f_{max} - f_{avg})}, & f' \geq f_{avg} \end{cases} \quad (9)$$

where P_{c1} and P_{c2} are the upper bound and lower bound of P_c; i.e., $P_{c1} = 0.99$ and $P_{c2} = 0.40$. f' is the fitness value of the larger individuals which are ready to cross. f_{avg} is the average fitness value of the population. f_{max} is the maximum fitness value of the population.

$\{P_1', P_2', P_3', \ldots, P_M'\}$ is the population made up of M individuals ready to cross. The encoding crossover steps are described as follows.

Step 1. Select two adjacent parent individuals P_1' and P_2' to cross and calculate the crossover probability P_c.

Step 2. Generate a random number between $[0, 1]$ with rand() function. If the number is larger than P_c, then do not cross the former part of the code; if not, then carry out the partially mapped crossover operator to the former part of P_1' and P_2';

Step 3. Generate two random numbers a and b $(a < b)$ between $[1, k]$. Firstly, exchange the codes between bit a and b of P_1' and P_2' and then modify the code values out of the cross region according to the mapping relationship of the value of the cross region.

Step 4. Generate a random number within $[0, 1]$ with rand() function. If the number is larger than P_c, then do not cross the latter part of the code; if not, carry out the two-point crossover operator to the latter part of P_1' and P_2'.

Step 5. Generate two random numbers c and d $(c < d)$ within $[k + 1, 2k]$ as the crossover point and exchange the codes between bits c and d of P_1' and P_2'. There is an example of the crossover process described in Figure 10.

Step 6. Repeat Steps 1–5 to the individuals remained and then a new population made up of M individuals comes out as $\{P_1'', P_2'', P_3'', \ldots, P_M''\}$.

According to the encoding, the sequence reversed mutation operator is used in the former part of this code, and the basic bit mutation operator is used in the latter part of the code. The probability of mutation is supposed to be in $[0.0001, 0.1]$ [36]. The mutation probability used in this paper is described as follows:

$$P_m = \begin{cases} P_{m1}, & f < f_{avg} \\ P_{m1} - \dfrac{(P_{m1} - P_{m2}) \times (f_{max} - f)}{(f_{max} - f_{avg})}, & f \geq f_{avg} \end{cases} \quad (10)$$

where P_{m1} and P_{m2} are the upper bound and lower bound of P_m; i.e., $P_{m1} = 0.1$ and $P_{m2} = 0.0001$. f is the fitness value of the individual which is ready to cross.

$\{P_1'', P_2'', P_3'', \ldots, P_M''\}$ is the population ready to mutate. The encoding mutation steps are described as follows.

Step 1. Select an adjacent parent P_1'' individual to mutate, and calculate the crossover probability P_m.

Step 2. Generate a random number within $[0, 1]$ with rand() function. If the number is larger than P_m, then do not mutate the former part of the code; if not, then carry out the sequence reversed mutation operator to the former part of P_1''.

Step 3. Generate two random numbers a and b $(a < b)$ within $[1, k]$ and reverse the codes between bits a and b of P_1''.

Step 4. Generate a random number within $[0, 1]$ with rand() function. If the number is larger than P_m, then do not cross the latter part of the code; if not, then carry out the basic bit mutation operator to the latter part of P_1''.

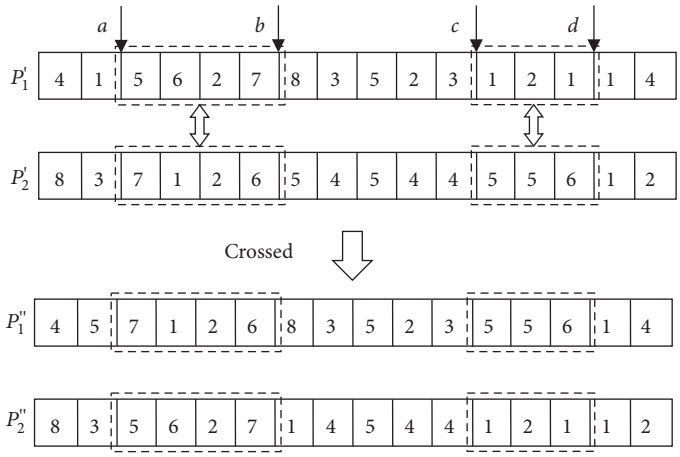

Figure 10: Partially mapped crossover and two-point crossover ($a = 3$, $b = 6$, $c = 12$, and $d = 14$).

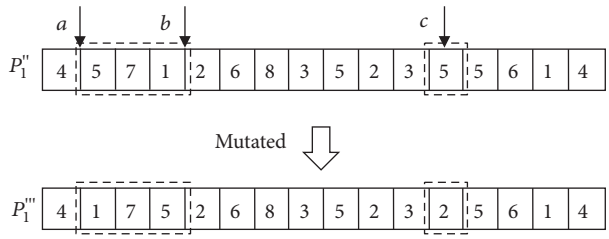

Figure 11: Sequence reversed mutation operator and basic bit mutation operator ($a = 2$, $b = 4$, and $c = 12$).

Step 5. Generate a random number c within $[k+1, 2k]$ as the mutation point, and exchange the code in bit c of P_1'' with a random number in $\{1, 2, 3, 4, 5, 6\}$. There is an example of the crossover process described in Figure 11.

Step 6. Repeat Steps 1–5 to the individuals remained, and then a new population made up of M individuals comes out as $\{P_1''', P_2''', P_3''', \ldots, P_M'''\}$.

4.4. Optimal Preservation Strategy. Good individuals of the parental population can be retained by adopting optimal preservation strategy, in order to avoid disappearing in genetic iteration. The steps of optimal preservation strategy are described as follows.

Step 1. Calculate the fitness value of all the individuals of the parental $\{P_1''', P_2''', P_3''', \ldots, P_M'''\}$ and get the individual with the largest fitness value as the optimal parental generation, whose fitness value is F_{max}.

Step 2. Calculate the fitness value of all the individuals of the offspring $\{P_1''', P_2''', P_3''', \ldots, P_M'''\}$ and get the individuals with the minimum fitness as the worst ones, the fitness of which is F_{min}''.

Step 3. Compare the optimal parental generation with the worst offspring. If $F_{min}'' < F_{max}$, then replace the worst offspring by the optimal parental generation.

In summary, the proposed adaptive genetic algorithm can be described in Table 2 where *GEN* denotes the maximum generation.

5. Numerical Results

In this section, we examine the performance of our algorithm by executing two tests: the weakly heterogeneous data test and the strongly heterogeneous data test. Usually the problem can be seen as strongly heterogeneous if many different types of cargoes need to be loaded. In the case of a small set of cargo types in the container, the problem is denoted weakly heterogeneous. For each test, we first state the test data, then present the test results, and finally perform the corresponding result analysis.

5.1. Weakly Heterogeneous Data Test

5.1.1. Test Data. We use 15 groups of classic weakly heterogeneous data (called LN01–LN15 in our paper) given in Loh and Nee's study [17] to test the performance of our algorithm and in the test $M = 30$ and $GEN = 600$. Further, we compare the performance of our algorithm with the performance of 13 algorithms presented in the related literature.

5.1.2. Test Results. The experiments are carried out on 15 sets of test data provided by Loh and Nee [17]. All the cargoes in the other 13 sets of test data expect LN02 and LN06 can be loaded into the container. There are 200 cargoes in LN02, and 162 of them can be loaded into the container after loading optimization. The space utilization rate reached 95.41%. 200 cargoes are provided in LN06, and 181 of them can be loaded into the container after loading optimization. In this case, the space utilization reaches 93.98%. The detail of the solution optimization by the algorithm is shown in Tables 3 and 4. "Loading sequence" denotes the priority for loading; "Serial number" is the ID number of the cargo type; "Quantity installed" is the quantity that can be loaded into the container; "Quantity not installed" is the quantity of the cargoes that

Table 2: Adaptive genetic algorithm.

Step	AGA (adaptive genetic algorithm)
(1)	**Input**: *Box*, *X*, *Y*, *Z*, *M*, *Pc*, *Pm*, *GEN*
(2)	Generate a group of initial population encoding *P*
(3)	**for** *s* = 1 to *GEN* do
(4)	decode *P* with heuristic rules
(5)	calculate the fitness value *Fi* of *P*
(6)	search the individual max *p* with the highest fitness value, whose fitness is max *fi*
(7)	do the selection operator to *P*, the result is *SelectP*
(8)	do partial mapped crossover operator to the former part coding of *SelectP*
(9)	do two point crossover operator to the latter part coding of *SelectP*
(10)	store the result as *CrossP*
(11)	do sequence reversed mutation operator to the former part of *CrossP*
(12)	do basic bit mutation operator to the latter part of *CrossP*
(13)	store the mutation result as *MutateP*
(14)	decode *MutateP* with heuristic rules
(15)	calculate the fitness value *Fi* of *MutateP*
(16)	search the individual max *Mutatep* with the highest fitness value in *MutateP*, whose fitness is max *Mutatefi*
(17)	search the individual min *Mutatep* with the lowest fitness value in *MutateP*, whose fitness value is min *Mutatefi*
(18)	**if** max *Mutatefi* < max *fi* **then**
(19)	replace min *Mutatep* in *MutateP* with max *p* in *P*
(20)	**end if**
(21)	*P* = *MutateP*
(22)	**end for**
(23)	search the individual max *p* with highest fitness value in the *GEN* generation, whose fitness is max *fi*
(24)	**Output**: max *p*, max *fi*

Table 3: Test result in LN02.

Loading sequence	Serial number	Quantity installed	Quantity not installed	Rotation
1	7	25	0	6
2	4	19	0	3
3	2	37	0	5
4	5	16	0	1
5	6	10	7	6
6	1	29	0	1
7	3	3	31	1
8	8	23	0	3

cannot be loaded into the container; "Rotation" indicates the direction of the corresponding cargoes placed and the value of them refers to the definition of Table 1.

Table 4: Test result in LN06.

Loading sequence	Serial number	Quantity installed	Quantity not installed	Rotation
1	5	25	0	2
2	6	23	0	5
3	1	34	0	2
4	8	0	17	6
5	2	37	0	3
6	4	25	2	4
7	7	14	0	6
8	3	23	0	5

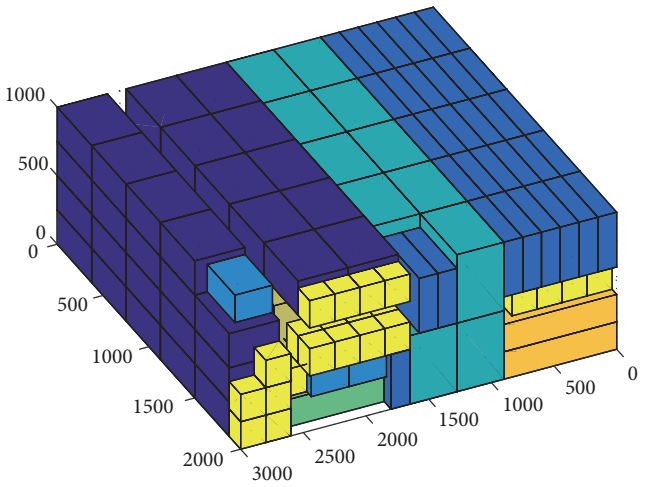

Figure 12: Packing result in LN02.

The packing simulation results of LN02 and LN06 are shown in Figures 12 and 13. There are 8 kinds of cargoes in LN02, and each color represents a kind of cargoes in LN02 in Figure 12. There are also 8 kinds of cargoes in LN06, and each color represents a kind of cargoes in LN06 in Figure 13.

5.1.3. Result Analysis. The results of performance comparison are listed in Table 5 and Figure 14. It can be seen that the average space utilization of our algorithm reaches 70.62%, which is the best among all the algorithms.

5.2. Strongly Heterogeneous Data Test

5.2.1. Test Data. The example in Robinson and George's paper [15] is selected as the test data for the strongly heterogeneous problem. The container is a 20 foot international standard container and the specification is 2352mm×2388mm×5899mm. Table 6 gives the details of cargoes to be loaded. This group of data contains 30 cargoes. The sizes of cargoes are different except Cargoes 4 and 5. The difference in the size of cargoes is very big so that the test is consistent with the strongly heterogeneous problem.

Table 5: Table of space utilization.

(a)

Data	BM[16]	N[18]	BR[19]	GB[20]	BG[10]	C[21]	B[13]
LN01	62.5%	62.5%	62.5%	62.5%	62.5%	62.5%	62.5%
LN02	90%	80.7%	90%	89.5%	96.6%	76.02%	89.8%
LN03	53.4%	53.4%	53.4%	53.4%	53.4%	53.4%	53.4%
LN04	55%	55%	55%	55%	55%	55%	55%
LN05	77.2%	77.2%	77.2%	77.2%	77.2%	77.19%	77.2%
LN06	83.1%	88.7%	83.1%	91.1%	91.2%	79.51%	92.4%
LN07	78.7%	81.8%	78.7%	83.3%	84.7%	72.14%	84.7%
LN08	59.4%	59.4%	59.4%	59.4%	59.4%	59.42%	59.4%
LN09	61.9%	61.9%	61.9%	61.9%	61.9%	61.89%	61.9%
LN10	67.3%	67.3%	67.3%	67.3%	67.3%	67.29%	67.3%
LN11	62.2%	62.2%	62.2%	62.2%	62.2%	62.16%	62.2%
LN12	78.5%	78.5%	78.5%	78.5%	78.5%	71%	78.5%
LN13	78.1%	84.1%	78.1%	85.6%	84.3%	84.14%	85.6%
LN14	62.8%	62.8%	62.8%	62.8%	62.8%	62.81%	62.8%
LN15	59.5%	59.5%	59.5%	59.5%	59.5%	59.46%	59.5%
Average	68.64%	69.00%	68.64%	69.95%	70.43%	66.93%	70.15%

(b)

Data	E[22]	M[24]	L[25]	W[26]	D[29]	LM[31]	This paper
LN01	62.5%	62.5%	62.5%	62.5%	62.5%	62.5%	62.5%
LN02	90.8%	92.6%	80.4%	90.7%	86.3%	96.4%	95.41%
LN03	53.4%	53.4%	53.4%	53.4%	53.4%	53.4%	53.43%
LN04	54.96%	55%	55%	55%	55%	55%	54.96%
LN05	77.2%	77.2%	77.2%	77.2%	77.2%	77.2%	77.19%
LN06	87.9%	91.7%	84.8%	92.9%	89.2%	93.5%	93.98%
LN07	84.7%	84.7%	77%	84.7%	83.2%	84.7%	84.66%
LN08	59.4%	59.4%	59.4%	59.4%	59.4%	59.4%	59.42%
LN09	61.9%	61.9%	61.9%	61.9%	61.9%	61.9%	61.89%
LN10	67.3%	67.3%	67.3%	67.3%	67.3%	67.3%	67.29%
LN11	62.2%	62.2%	62.2%	62.2%	62.2%	62.2%	62.16%
LN12	78.5%	78.5%	69.5%	78.5%	78.5%	78.5%	78.52%
LN13	85.6%	68%	73.3%	85.6%	85.6%	84.9%	85.61%
LN14	62.8%	62.8%	62.8%	62.8%	62.8%	62.8%	62.81%
LN15	59.5%	59.5%	59.5%	59.5%	59.5%	59.5%	59.46%
Average	69.91%	69.11%	67.08%	70.24%	69.60%	70.61%	70.62%

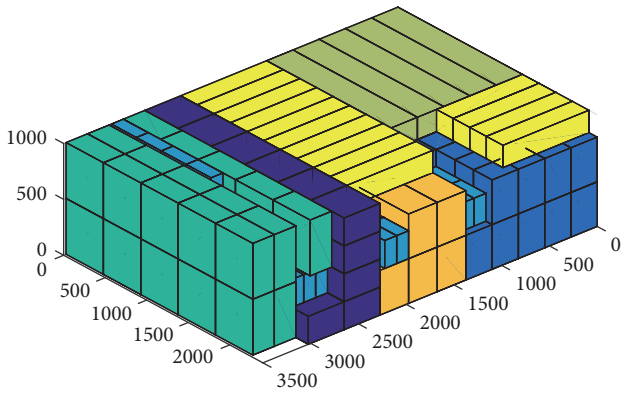

Figure 13: Packing result in LN06.

5.2.2. Test Results. The results are shown in Table 7. The cargoes that are not loaded into containers are not listed in the table. "The length of edges parallel to X", "The length of edges parallel to Y", and "The length of edges parallel to Z" indicate the parallel relationship with X, Y, and Z axis when they are placed. There are 30 kinds of cargoes in the test. 18 of them can be loaded into the container after optimization by our algorithm, and 12 cannot be loaded into the container. The space utilization reaches 84.42%.

The loading simulation result is shown in Figure 15. Since 18 cargoes are loaded into the container. Hence, 18 different colors are used to represent them.

5.2.3. Result Analysis. The space utilization reached 80% in George and Robinson's paper [15], but in this paper it goes

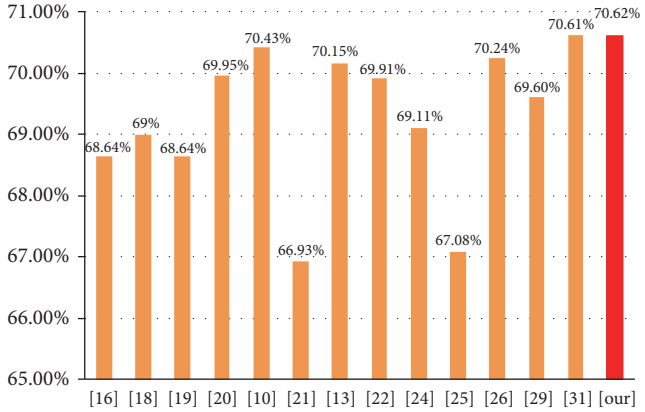

Figure 14: Bar chart of space utilization in weakly heterogeneous data test.

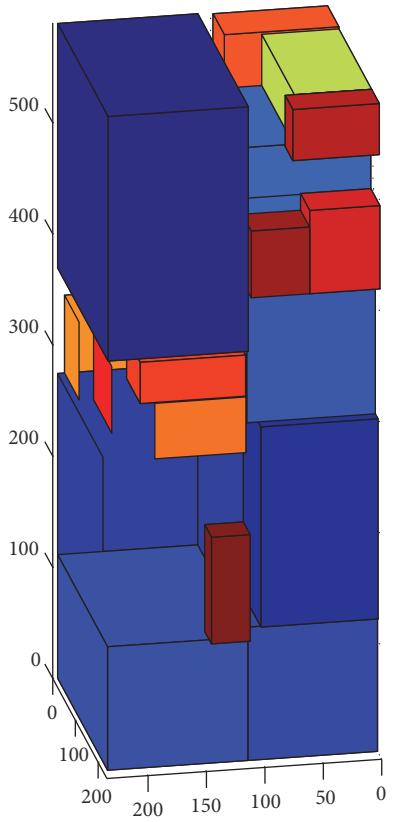

Figure 15: Strongly heterogeneous packing result.

up to 84.42%, which has improved 4.42% space utilization as shown in Table 8. The space division rule in this paper is modified based on the method proposed by George and Robinson, which is a dynamic space partition method. The narrow space generated in the loading process is considered to be an abandoned space in advance, and the corresponding space partition rule is adopted. Hence, our method increases the volume of available space, optimizes the utilization of waste space, and improves the overall space utilization. The

Table 6: The test data of strongly heterogeneous problem.

Serial number	The length of the longest edge (cm)	The length of the middle edge(cm)	The length of the shortest edge (cm)
1	222	220	120
2	222	180	100
3	200	163	120
4	220	120	111
5	220	120	111
6	210	120	110
7	190	120	110
8	150	140	100
9	122	103	97
10	168	98	68
11	132	96	66
12	130	96	50
13	120	80	60
14	110	72	68
15	120	80	60
16	130	70	54
17	144	66	46
18	173	69	36
19	84	78	62
20	95	66	60
21	68	68	68
22	70	64	54
23	78	62	50
24	98	50	48
25	90	60	37
26	80	60	40
27	75	60	40
28	74	46	34
29	60	50	40
30	96	33	30

results of this experiment show that the algorithm is significantly higher than George and Robinson's algorithm, which indicates that the improved method of space partitioning is effective for the optimization of container loading problem.

6. Conclusions

In this paper, we construct the general loading mathematical model based on the goal of maximizing space utilization. We propose a dynamic space division method to improve the placement heuristic which designs different space division methods for different cargoes and effectively avoids a certain amount of space loss. Based on the dynamic space division method, we develop a dedicated genetic algorithm that uses a two-stage real-number encoding method. The encoding method consisting of the sequence of cargoes and rotation lifts a single rule for cargoes loading order restriction.

Table 7: Test result of strongly heterogeneous problem.

Load sequence	Serial number	The length of edge parallel to X	The length of edge parallel to Y	The length of edge parallel to Z
1	5	111	220	120
2	2	100	222	180
3	6	110	210	120
4	4	110	220	120
5	17	60	144	46
6	30	96	33	30
7	26	80	60	40
8	7	120	190	110
9	24	50	48	98
10	29	60	40	50
11	3	120	200	163
12	25	90	37	60
13	28	74	34	46
14	15	120	60	70
15	13	120	80	60
16	19	78	84	62
17	27	40	75	60
18	1	120	220	222

Table 8: Table of space utilization.

Method	Unloaded cargoes	space utilization
George and Robinson [15]	Unavailable	80%
Jiang et al. [37]	13	82.8%
This paper	12	84.42%

This algorithm searches for the best combination of cargo's sequence and rotation through genetic algorithms, which provides a larger search space for the algorithm to find a better optimal solution. Based on this genetic code, this paper chooses the suitable genetic operators that contribute to more diversity of genetic search space. As the improvement of encoding greatly increases the size of the search space, it becomes easy to converge to a premature local optimization which renders some difficulties for the search. We apply the dynamic adaptive technology to the genetic algorithm, which dynamically adjusts the crossover and mutation probability based on the individual fitness. This can improve the search efficiency and avoid premature convergence.

To examine the performance of our algorithm, we use both weakly and strongly heterogeneous loading data to test the proposed algorithm. By choosing 15 classic sets of test data in [17] as weakly heterogeneous data, the average space utilization of our algorithm reaching 70.62% outperforms those of 13 algorithms from the related literature. Taking a set of test data in [15] as strongly heterogeneous data, the space utilization can be improved by 4.42% in comparison with [15]. Thus, the numerical experiments confirm that our algorithm can achieve a better performance than that of other algorithms.

Data Availability

The data used to support the findings of this study are included within the article, by referring to some classic literatures.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 51209100, in part by the Shenzhen Science and Technology Plan Project under Grant JCYJ20170413115305468, and in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFYXJJ005.

References

[1] C.-H. Tang, "A scenario decomposition-genetic algorithm method for solving stochastic air cargo container loading problems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 47, no. 4, pp. 520–531, 2011.

[2] A. Lim, H. Ma, C. Qiu, and W. Zhu, "The single container loading problem with axle weight constraints," *International Journal of Production Economics*, vol. 144, no. 1, pp. 358–369, 2013.

[3] Z. Xue, C. Zhang, W.-H. Lin, L. Miao, and P. Yang, "A tabu search heuristic for the local container drayage problem under a new operation mode," *Transportation Research Part E: Logistics and Transportation Review*, vol. 62, pp. 136–150, 2014.

[4] J.-N. Zheng, C.-F. Chien, and M. Gen, "Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem," *Computers & Industrial Engineering*, vol. 89, pp. 80–87, 2015.

[5] A. Ramos, J. F. Oliveira, and M. P. Lopes, "A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions," *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 215–238, 2016.

[6] J. Christensen and D. Pacino, "A matheuristic for the Cargo Mix Problem with Block Stowage," *Transportation Research Part E: Logistics and Transportation Review*, vol. 97, pp. 151–171, 2017.

[7] S. Yan, Y.-L. Shih, and F.-Y. Shiao, "Optimal cargo container loading plans under stochastic demands for air express carriers," *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 3, pp. 555–575, 2008.

[8] S. Liu, X. Shang, C. Cheng, H. Zhao, D. Shen, and F. Wang, "Heuristic algorithm for the container loading problem with multiple constraints," *Computers & Industrial Engineering*, vol. 108, pp. 149–164, 2017.

[9] X. Zhao, J. A. Bennell, T. Bektas, and K. Dowsland, "A comparative review of 3D container loading algorithms," *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 287–320, 2016.

[10] A. Bortfeldt and H. Gehring, "Applying tabu search to container loading problems," in *Operations Research Proceedings*, pp. 533–538, 1997.

[11] G. Cabrera-Guerrero, C. Lagos, C. Castañeda, F. Johnson, F. Paredes, and E. Cabrera, "Parameter tuning for local-search-based matheuristic methods," *Complexity*, vol. 2017, Article ID 1702506, 2017.

[12] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," *International Journal of Production Economics*, vol. 174, pp. 93–110, 2016.

[13] A. Bortfeldt and H. Gehring, "A hybrid genetic algorithm for the container loading problem," *European Journal of Operational Research*, vol. 131, no. 1, pp. 143–161, 2001.

[14] Y. Zhou, L. Zhou, Y. Wang, Z. Yang, and J. Wu, "Application of Multiple-Population Genetic Algorithm in Optimizing the Train-Set Circulation Plan Problem," *Complexity*, vol. 2017, Article ID 3717654, 14 pages, 2017.

[15] J. A. George and D. F. Robinson, "A heuristic for packing boxes into a container," *Computers & Operations Research*, vol. 7, no. 3, pp. 147–156, 1980.

[16] E. E. Bischoff and M. D. Marriott, "A comparative evaluation of heuristics for container loading," *European Journal of Operational Research*, vol. 44, no. 2, pp. 267–276, 1990.

[17] T. Loh and A. Nee, "A packing algorithm for hexahedral boxes in," in *Proceedings of the Conference of Industrial Automation*, pp. 115–126, Singapore, 1992.

[18] B. K. A. Ngoi, M. L. Tay, and E. S. Chua, "Applying spatial representation techniques to the container packing problem," *International Journal of Production Research*, vol. 32, no. 1, pp. 111–123, 1994.

[19] E. E. Bischoff and M. S. W. Ratcliff, "Issues in the development of approaches to container loading," *Omega* , vol. 23, no. 4, pp. 377–390, 1995.

[20] H. Gehring and A. Bortfeldt, "A genetic algorithm for solving the container loading problem," *International Transactions in Operational Research*, vol. 4, no. 5-6, pp. 401–418, 1997.

[21] C.-F. Chien and W.-T. Wu, "A recursive computational procedure for container loading," *Computers & Industrial Engineering*, vol. 35, pp. 319–322, 1998.

[22] C.-F. Chien and J.-F. Deng, "A container packing support system for determining and visualizing container packing patterns," *Decision Support Systems*, vol. 37, no. 1, pp. 23–34, 2004.

[23] A. Moura and J. F. Oliveira, "A GRASP approach to the container-loading problem," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 50–57, 2005.

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

[25] Z. Wang, K. W. Li, and J. K. Levy, "A heuristic for the container loading problem: a tertiary-tree-based dynamic space decomposition approach," *European Journal of Operational Research*, vol. 191, no. 1, pp. 84–97, 2008.

[26] Y. Wu, W. Li, M. Goh, and R. de Souza, "Three-dimensional bin packing problem with variable bin height," *European Journal of Operational Research*, vol. 202, no. 2, pp. 347–355, 2010.

[27] K. He and W. Huang, "A caving degree based flake arrangement approach for the container loading problem," *Computers & Industrial Engineering*, vol. 59, no. 2, pp. 344–351, 2010.

[28] T. Dereli and G. S. Das, "A hybrid 'bee(s) algorithm' for solving container loading problems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2854–2862, 2011.

[29] J. Ren, Y. Tian, and T. Sawaragi, "A tree search method for the container loading problem with shipment priority," *European Journal of Operational Research*, vol. 214, no. 3, pp. 526–535, 2011.

[30] A. Lim, H. Ma, J. Xu, and X. Zhang, "An iterated construction approach with dynamic prioritization for solving the container loading problems," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4292–4305, 2012.

[31] J. F. Gonçalves and M. G. C. Resende, "A parallel multi-population biased random-key genetic algorithm for a container loading problem," *Computers & Operations Research*, vol. 39, no. 2, pp. 179–190, 2012.

[32] I. Araya and M.-C. Riff, "A beam search approach to the container loading problem," *Computers & Operations Research*, vol. 43, no. 1, pp. 100–107, 2014.

[33] L. Junqueira, R. Morabito, and D. Sato Yamashita, "Three-dimensional container loading models with cargo stability and load bearing constraints," *Computers & Operations Research*, vol. 39, no. 1, pp. 74–85, 2012.

[34] Y. Zhu, X. Xiang, and Y. Yang, "General cargo ship loading problems based on the hybrid genetic algorithm," *Chinese Journal of Ship Research*, vol. 10, no. 6, pp. 126–132, 2015.

[35] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.

[36] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *The Computer Journal*, vol. 27, no. 6, pp. 17–26, 1994.

[37] Y. Jiang, J. Zha, and D. He, "Research on the packing of loading rectangular freight into a container," *Journal of The China Railway Society*, vol. 22, no. 6, pp. 13–18, 2000 (Chinese).