RAIRO-Oper. Res. 60 (2026) 1907–1930
https://doi.org/10.1051/ro/2026059

RAIRO Operations Research
www.rairo-ro.org

OPTIMIZING LOAD BALANCED 3D-BIN PACKING WITH PRODUCT FAMILY: A MULTI-OBJECTIVE GENETIC ALGORITHM

Vildan Özkir[1,*], Seda Erbayrak[2] and Umman Mahir Yildirim[3]

Abstract. Logistics companies continuously look for advanced methods to improve operational efficiency, and the problem of three-dimensional bin packing poses a significant challenge due to its complexity and practical relevance. This study proposes a novel hybrid genetic algorithm designed to address three key objectives simultaneously: (i) minimizing the number of containers used, (ii) achieving load balance by reducing deviations from the ideal barycenter, and (iii) preserving family unity within containers. The proposed algorithm presents a comprehensive approach to three-dimensional bin packing problems, addressing not only the three-dimensional placement and non-overlapping constraints emphasized by conventional approaches but also integrating real-world considerations such as orientation, load balancing, stacking, and product family constraints. The efficiency and robustness of the proposed algorithm have been demonstrated by extensive computational experiments and comparative analyses. The results highlight its potential to support more efficient and constraint-aware container loading strategies in real-world logistics operations.

Mathematics Subject Classification. 90B06, 97D50.

Received June 4, 2025. Accepted May 12, 2026.

1. Introduction

With technological advancements intensifying market competition, companies are developing sophisticated cost optimization strategies to reduce variable expenses and maintain their competitive edge. In the logistics industry, better decisions for well-planned transportation operations are critical. Due to the limited loading time, loading resources, and energy used, container loading operations are under stress [8]. Poor loading plans cause ineffective transportation operations and delays, resulting in additional costs. Companies should develop better methodologies to ensure optimal loading plans and incorporate these methodologies into their processes in the long run.

Container loading problems aim to improve the efficiency of the logistics system by maximizing container capacity utilization [8]. Container loading problems are also known as Three-Dimensional Bin Packing Problems (3D-BPP), and they are computationally NP-hard. The purpose of the 3D-BPPs is to place a finite set of rectangular items of varying sizes in the fewest number of bins possible. However, in real practice, focusing solely on

Keywords. Load balance, family unity, hybrid genetic algorithm.

[1] Yıldız Technical University, Istanbul, Turkey.
[2] Istanbul Gelisim University, Istanbul, Turkey.
[3] Istanbul Bilgi University, Istanbul, Turkey.
*Corresponding author: vildan@yildiz.edu.tr

this objective is insufficient. To meet real-world requirements, effective three-dimensional bin packing strategies must integrate multiple critical considerations, including load balance, stacking constraints, and family-based item groupings. The balance of loads has recently been studied in the literature, despite being a crucial factor for all modes of transport. If the characteristics of the loads or items are not properly considered in the packing plan, there is a greater risk of problems arising during transportation and can also affect routing decisions. In addition, load stability is another important factor in reducing potential risks during transportation. The container loading literature also tends to focus less on item- and order-related considerations, such as stacking constraints and the need for family unity of packed items. For optimal stability, the planned center of gravity should align closely with the actual center of gravity of the packed load. Family unity is a recent concept that addresses items that must be packed together in the same bin for practical reasons, such as having the same destination or handling requirements or storage conditions, which are attached to each item individually. For further details on the concept of family unity, we refer the reader to [18]. This paper proposes a new heuristic for solving large-scale 3D-BPPs with item-related and load-related factors. To the best of our knowledge, previous studies proposing heuristics for 3D-BPPs are limited to a set of packing-related constraints and do not consider the impact of item-related constraints. This study contributes to the literature by:

- Introducing an improved genetic algorithm-based heuristic approach specifically designed for complex 3D bin packing, featuring a specialized chromosome representation, a crossover, and a selection mechanism to enhance search efficiency.
- Developing a comprehensive mathematical formulation that simultaneously optimizes bin usage, load balance, and family unity objectives, providing trade-offs compared to traditional single-objective models.
- Incorporating a wide range of real-world constraints – including item orientation, stacking requirements, and grouping requirements – which allows for direct application in logistics to improve operational efficiency.

By addressing practical limitations in current methodologies, the research provides a flexible computational solution for addressing complex three-dimensional packing challenges. To tackle the difficulties in highly constrained bin packing problems, we explore innovative solution methodologies and assess their efficacy on benchmark instances. The subsequent sections of this paper are structured as follows. We provide a comprehensive literature review on highly constrained bin packing problems in Section 2. In Section 3, we detail the genetic algorithm-based heuristics developed in this study. We present a comparative analysis based on benchmark instances and the real case application in Section 4. Finally, we conclude with key findings and propose directions for future research in Section 5.

2. Review of 3D-bin packing literature

The 3D-BPPs are NP-hard problems [39]. Although there are studies that propose a mathematical model for BPPs, their solution is quite limited [1,9,38,48]. In order to solve large-scale 3D-BPPs where mathematical models are insufficient, a number of heuristics, metaheuristics, and hybrid approaches have been proposed in the literature. The purpose of this section is to provide a thorough overview of the research on heuristic algorithms for 3D-BPP.

Gilmore and Gomory [27] introduced the idea of optimizing the packing of objects into containers with multiple dimensions, which is the first study to address multi-dimensional packing problems. A heuristic, the wall-building algorithm, was proposed by George and Robinson [26] to solve 3D-BPPs where item orientation is not allowed. Several studies in the literature that stack items in layers parallel to bin walls were pioneered by their study [3, 4, 15, 44, 46]. The rotation of items was incorporated into the wall-building algorithm by Moura and Oliveira [42]. Their algorithm allows rotation when needed and modifies the placement boundaries according to the dimensions of the element beneath. This change guarantees complete support for the packed items, which improves stability. Pisinger [44] proposed a novel heuristic based on the wall-building approach, using a tree search algorithm to determine the placement order of the items, replacing the traditional ranking criteria. Dereli and Daş [15] also proposed an extension to the wall-building algorithm and utilized Ant Colony

optimization approach for container loading problems. Sheng *et al.* [46] presented an algorithm that incorporates stability and orientation constraints and combines a wall-building strategy with a tree search method. In order to optimize space utilization, the algorithm assesses various layer orientations. In order to optimize space utilization, the algorithm assesses various layer orientations. In the literature, genetic algorithms have been used to solve packing problems; when paired with placement algorithms, their performance increases significantly, generating effective solutions [4, 32, 45, 49]. Feng *et al.* [22] developed a genetic algorithm hybridized with the strategy of the largest left space first item assignment to solve the bin-packing problem with orientation constraint. Kang *et al.* [34] developed a greedy heuristic algorithm, namely deepest bottom-left-fill heuristic, and hybridized it with the genetic algorithm by considering item orientations and load stability. Moon and Nguyen [41] propose a hybrid algorithm that combines the deepest bottom-left fill heuristic with genetic algorithms, effectively addressing orientation and balance constraints. Gonzalez *et al.* [29] employed genetic algorithms to maximize bin volume and weight utilization, incorporating item orientation constraints. The method, tested on randomly generated bins and items, showed promising results, particularly for instances with limited item variety. Gonçalves and Resende [28] propose a random key-based genetic algorithm for two-dimensional BPP and 3D-BPPs, allowing item orientations. The proposed algorithm introduces a novel fitness function – adjusted number of bins – that assesses the potential for improvement of bin packing solutions. Yeung and Tang [50] combine the genetic algorithm with lower-left corner heuristic, which is a placement algorithm for load balanced bin packing problems. The literature on heuristic methods for solving bin packing problems is extensive. Table 1 provides a summary of studies addressing balance (B), stability (S), and orientation (O) constraints, along with the proposed methods for solving these problems.

The problem of simultaneously satisfying all constraints remains pending despite a significant amount of literature on heuristic bin packing algorithms. Even though soft constraints are used in the literature and are important in practice, there is still a lack of literature that adequately addresses challenges that arise in real-world situations [5].

3. Proposed method

This section proposes hybrid genetic algorithm-based heuristics for solving highly constrained 3D-BPPs, as illustrated in Figure 1. The algorithm aims to determine the optimal packing plan by focusing on three key objectives: (1) minimizing the number of bins used, (2) minimizing the total deviation of balance from the ideal barycenter for each bin, and (3) ensuring family unity within the bins.

The methodology follows a three-phase sequence: Phase 1 allocates items to bins; Phase 2 determines their exact coordinates; and Phase 3 improves the load distribution for stability. This approach decomposes the complex packing problem into manageable steps while satisfying all operational constraints. The algorithm consists of three main phases:

Phase 1: The problem is initially reduced to a one-dimensional BPP by considering only the volume of the items, ignoring their dimensions. This phase employs a genetic algorithm to determine item-bin allocations, ensuring that families stay together.

Once the volume-based allocation is fixed, the solution moves to a more realistic 3D packing scenario. The family groupings established in Phase I are preserved as constraints, so the problem now focuses on arranging these grouped items' physical placement within each bin.

Phase 2: The problem is then treated as a 3D-BPP, where the physical dimensions, orientations of items, and stacking constraints are considered. The algorithm packs the solution established in Phase 1 by respecting these geometrical arrangement constraints while preserving the family unity.

After the items are physically packed into bins, the solution proceeds to the final phase for load balance evaluation. Any weight distribution imbalances within the bins are addressed to improve stability and handling.

Phase 3: The algorithm focuses on improving the balance of each bin by making the appropriate adjustments, ultimately producing the final loading plan. This may involve repositioning or slight reallocations to ensure that the load is evenly distributed, resulting in a practical and safe loading plan.

TABLE 1. 3D bin packing studies in the literature.

Publication	Single bin	Multiple bins		Constraints			Method
		Identical	Variable	(B)	(S)	(O)	
Martello *et al.* [39]		+					Branch & Bound
Eley [17]		+		+	+	+	Greedy Heuristic
Pisinger [44]	+				+	+	Tree Search
Bortfeldt *et al.* [6]	+				+	+	Parallel Tabu Search
de Castro Silva *et al.* [13]		+		+		+	Greedy Heuristic
Lodi *et al.* [37]		+				+	Tabu Search
Moura & Oliveira [42]	+				+	+	GRASP
Parreño *et al.* [43]	+						GRASP
Tarantilis *et al.* [47]		+			+		Tabu search, GLS
Fuellerer *et al.* [23]		+			+	+	Ant Colony
Iori & Martello [31]		+			+	+	Column generation
He & Huang [30]	+					+	Fit Degree Algorithm
Liu *et al.* [36]	+			+	+	+	Tabu Search
de Almeida & Figueiredo [12]		+				+	Lower Bound Algorithm
Fanslau & Bortfeldt [21]	+				+	+	Tree Search
de Queiroz *et al.* [14]			+			+	Dynamic programming, Column Generation
Lim *et al.* [35]		+				+	Greedy heuristic
Araya & Riff [2]	+				+	+	Beam Search
Crainic *et al.* [11]			+				Progressive Hedging
Trivella & Pisinger [48]		+		+			Graph representation
Gajda *et al.* [24]	+			+	+	+	Randomized Constructive Heuristic
This study			+	+	+	+	Genetic Algorithm

The following subsections describe each phase of the proposed algorithm in detail. The pseudocode of the algorithm is given in Table 2.

3.1. Phase I: genetic algorithm

Genetic algorithms are population-based evolutionary algorithms that improve efficiency in a wide range of optimization applications. Genetic algorithms employ an iterative evolutionary framework that begins with the creation of a population of candidate solutions encoded as chromosomes. Each chromosome is evaluated for fitness using an objective function that quantifies solution quality. The algorithm then uses selection mechanisms to identify high-performing individuals as potential parents for reproduction. Crossover operators combine parental genetic material to produce offspring, whereas mutation introduces stochastic variations to maintain population diversity and prevent premature convergence. This reproductive cycle produces successive generations until the predetermined termination criteria are met.

The first phase of the proposed algorithm is based on the work of Falkenauer [19]. In this context, items and bins are identified solely by their volume, and the problem is solved using a one-dimensional BPP. This phase aims to assign items to bins in order to maintain family unity while meeting volume restrictions. This

TABLE 2. Pseudocode of the proposed algorithm.

Algorithm: Hybrid genetic algorithm.

```
Input: ItemList, BinSize, Parameters
Output: Best 3D packing solution
    Initialize population:
      Shuffle Item List and create chromosomes by 1D bin packing
    while termination condition not met do
      Evaluate fitness value (Eq. 1) of each chromosome
      Select parents using roulette wheel selection method
     Apply crossover:
          Calculate the selection value for each gene using equation (6)
           Select the lowest selection value as identify crossover point and apply crossover
           Exchange these genes between Parent 1 and Parent 2
           Remove duplicate items from both chromosomes
           Reallocate remaining items to new bins by prioritizing product family
           Form new offspring (Child 1 and Child 2) from the resulting chromosomes.
      Apply mutation:
           Form new generation with elites and offspring.
    end while
Select best chromosomes as 1D packing solutions
    for each 1D packing solution do
      Create empty 3D bins corresponding to each 1D bin
       while items remain in the 1D bin do
          Select item according to priority rule
          if current layer has space then
            Place item in current layer
            Add items of the same product family to priority set
          else
          Select item according to priority rule
          if current layer has space then
            Place item in current layer
            Add items of the same product family to priority set
          else
           Open new layer and place item
          end if
            if item does not fit in bin then
               Add item to remaining items pool
            end if
       end while
    end for
    for each item in remaining items pool do
        Try to place in existing bins with space and same-family preference
        if item does not fit in any bin then
           Create a new bin and place item
        end if
   end for
    Apply balancing
           Iterate over permutations of layer order and orientations
           Calculate center of gravity for each permutation
           Retain configuration with best balance
    Return best balanced 3D packing solution
```

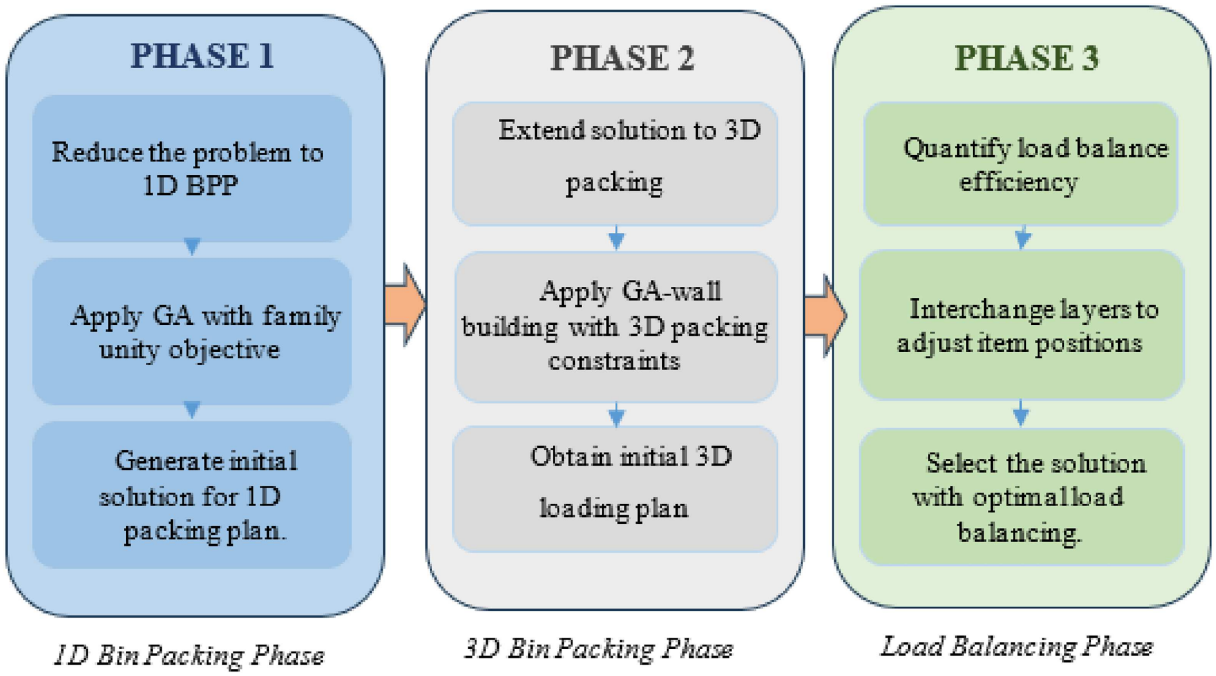


FIGURE 1. Hybrid genetic algorithm for solving 3D-BPP with load balance and family unity.

phase of the proposed algorithm differs from Falkenauer [19] in the following respects: (i) the fitness function includes multiple objectives, the fill rate, and the family unity, and (ii) a novel procedure is introduced to identify crossover points and reposition items.

3.1.1. Representation and initialization

Chromosome encoding is fundamental to genetic algorithm performance. The chromosome fundamentally represents a solution encoding that specifies item allocation among bins. For bin packing optimization, three primary encoding schemes exist in the literature. In *the item-based chromosome structure*, the items are represented as genes on the chromosome, and the chromosome is separated into bins based on how items are grouped. Each bin on the chromosome holds a subset of items. In *the bin-based chromosome structure*, the representation differs from the item-based approach. Each gene corresponds to a bin. The chromosome itself has a fixed length that corresponds to the number of items, and the value of each gene indicates which bin the respective item belongs to. *The group-based chromosome structure*, proposed by Falkenauer [19], is a hybrid representation that combines aspects of both bin-based and item-based structures. The structure is divided into two parts: (i) the first part encodes the assignment of items to bins similarly in bin-based structure and the second part addresses groups of bins. The grouping structure is used during genetic operations (such as crossover and mutation) to control how bins are reorganized. The second part is used in genetic operations, but the first part is only needed to identify group members [40]. To illustrate the chromosome structures for a problem with 5 items and 2 bins, Figure 2 presents sample chromosomes for three different chromosome structures, demonstrating how items are grouped and assigned to bins.

We have reviewed various chromosome structures in the literature, particularly those applied to bin packing and partitioning problems. These structures, including item-based, bin-based, and group-based formats, were carefully examined to evaluate their effectiveness in solving different extensions of the BPPs. The insights gained from this review informed the design of the chromosome representations used in this study. Table 3 illustrates

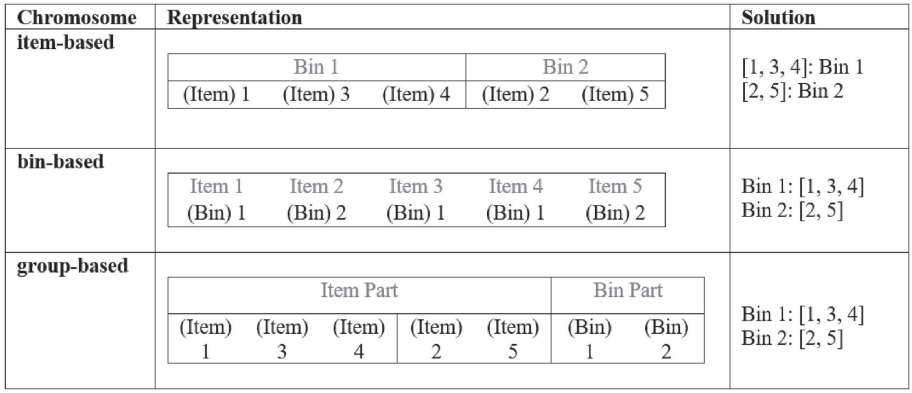


FIGURE 2. Chromosome representations for BPPs.

TABLE 3. Chromosome structures for BPPs in the literature.

Study	Bin		Chromosome structures		Gene structure
	Single	Multi	Item based	Group based	
Gehring and Bortfeldt [25]	+		+		PO, RI
Wu *et al.* [49]	+		+		PO, RI
Kang *et al.* [34]	+		+		PO
Gonçalves and Resende [28]	+		+		PO, RI
Moon and Nguyen [41]	+		+		PO, RI
Dokeroglu and Cosar [16]		+		+	Bins
Gonzalez *et al.* [29]	+		+		TI, RI, No. of items
Jamrus and Chien [32]	+		+		PO
Kabasakal and Keskin [33]		+		+	Bins
Ramos *et al.* [45]	+		+		PO, RI
Fan *et al.* [20]		+		+	Bins

Notes: PO: Packing order, RI: Rotation of items, TI: Types of items.

TABLE 4. Chromosome structure used in the study.

Item no	Volume	Family	Item no	Volume	Family	...	Bin	Item no	Volume	Family	Item no	Volume	Family	...	Bin	...	...	Bin
i	v_i	f_i	j	v_j	f_j	$\cdots$	1	k	v_k	f_k	l	v_l	f_l	$\cdots$	2	$\cdots$	$\cdots$	m

that the item-based chromosome structure is commonly used in single-objective BPPs with a single bin, while the group-based chromosome structure is more frequently applied in BPPs involving multiple bins.

Falkenauer [19] demonstrates that for BPPs with multiple bins, the group-based chromosome structure performs better than both the bin-based and item-based structures. Based on this, we have developed an algorithm that uses the group-based chromosome structure, where each gene corresponds to a single bin containing items. As presented in Table 4, we employ a standardized chromosome representation in which each gene consists of three encoded parameters: item identifier (bold), volume (italics), and family index (normal text).

TABLE 5. Example 1D packing instance showing item volumes and corresponding product families.

Items	1	2	3	4	5	6	7	8	9	10
Volume	10	20	20	25	10	30	25	10	20	40
Family	1	1	2	1	2	2	3	4	4	3
Bin	1	1	3	1	3	3	1	2	2	2

We employed a random population generation process to initialize the genetic algorithm. The random initialization step is critical to ensure a diverse initial population and contributes to the optimization process without requiring additional computational effort.

3.1.2. Fitness function

The fitness function, generally the objective function of the problem, determines the probability of the chromosomes being passed on to the next generation. The objective of the first phase is to maximize the fill rate of each bin and the family unity ratio in each bin, which yields a multiple objective optimization problem. In the proposed algorithm, the fitness function is formulated as a weighted function of these two components, the average fill rate ($\mathcal{FR}$) and the ratio of family unity ($\mathcal{PR}$) as represented in equation (1),

$$F = w_{\text{FR}}\mathcal{FR} + w_{\text{PR}}\mathcal{PR} \tag{1}$$

where $w_{\text{FR}} \in [0,1]$ and $w_{\text{PR}} \in [0,1]$ represent the weights of the average fill rate and the ratio of family unity, respectively.

The average fill rate is a commonly used measure in the BPP literature [19]. It is defined as the mean fill rate across all bins, ie. ($\mathcal{FR} = \frac{1}{m}\sum_{j=1}^{m} \text{FR}_j$) where FR_j is the fill rate of bin j as given in equation (2),

$$\text{FR}_j = \sum_{i \in I_j} \sum_{f \in F} \frac{v_{if}}{V_j}. \tag{2}$$

Here, v_{if} represents the volume of item i belonging to product family f, V_j denotes volume of bin j, and I_j represents the set of items packed in bin j, and m denotes the total number of bins.

The ratio of family unity is a metric used to assess how efficiently the items of a family are packed into bins, taking into account both the minimum number of bins required and the actual number of bins used in the packing plan. It is calculated by dividing the minimum number of bins required to pack the family's items N'_f by the total number of bins actually used for the family in the packing plan N''_f . The calculation of N'_f, N''_f and $\mathcal{PR}$ are provided in equations (3)–(5), where I_f be the set of items belonging to family f.

$$N'_f = \left\lceil \frac{\sum_{i \in I} v_{if}}{V_j} \right\rceil \tag{3}$$

$$N''_f = |\{j \mid \exists i \in I_j,\ i \in I_f\}| \tag{4}$$

$$\mathcal{PR} = \frac{\sum_f N'_f}{\sum_f N''_f}. \tag{5}$$

A higher ratio (closer to 1) of $\mathcal{PR}$ indicates efficient packing, where the number of bins used is close to the theoretical minimum in terms of family unity. To illustrate the calculation of the Family Unity Ratio, Table 5 presents an example of packing data with 10 items packed into 3 bins. The volume of each bin is assumed to be identical, with a capacity of 80 units.

The minimum number of bins required to pack family 1 can be computed as $(10+20+25)/80 = 0.6875$. The minimum number of bins required is calculated as 0.7500, 0.8125 and 0.3750 for families 2, 3, and 4, respectively. The total number of required bins is calculated by summing up the smallest integer greater than the minimum number of bins required for each family, which is 4 for this example. The total number of bins used to pack family f in the current solution is calculated as follows. The items from the first family are packed in one single bin, the items from the second family are packed in one single bin, the items from the third family are packed in two different bins and finally the items from the fourth family are packed in one bin. Thus, the total number of bins used to pack these items is 5. Therefore, the Family Unity Ratio ($\mathcal{PR}$) is calculated as $4/5 = 0.80$.

3.1.3. Selection

Parents should be selected from existing solutions to create a new generation after calculating the fitness value of each chromosome. The algorithm employs the roulette wheel method to select parent chromosomes. The roulette wheel selection method is a probability-based technique commonly used in genetic algorithms to choose chromosomes for reproduction. In this method, the probability of selecting a chromosome is proportional to its fitness value in comparison to the population's overall fitness. This ensures that chromosomes with higher fitness scores have a better chance of being chosen while also giving less fit chromosomes a chance, thereby maintaining population diversity. Although the method increases the likelihood of passing on chromosomes with higher fitness to the next generation, it does not guarantee that the best-performing chromosomes are always selected. As a result, it gives a probabilistic preference to chromosomes with higher fitness, establishing a balance between preserving diversity (exploration) and selecting the best solutions (exploitation).

3.1.4. Crossover and mutation

The crossover operation genetic algorithms combine the genetic information of two parents to generate new offsprings, which are expected to have higher fitness scores than their parents. To maintain family unity, we propose a new crossover operator for individuals with high fitness scores. To promote the transmission of solutions with homogeneous product families to the next generation, the proposed method calculates the selection value δ for each gene as shown in equation (6),

$$\delta = (1 - \mathrm{FR}_j) + [(f_j + 1)(n + 1)] + [(f_j + 1)(n - m_j)] + F_j^{HI} \tag{6}$$

where f_j,m_j, F_j^{HI} and n are defined as the number of different families in bin j, the total number of items packed in bin j, family heterogeneity index, and the number of items, respectively. The selection value (δ) consists of four terms. The first term, $(1 - \mathrm{FR}_j)$, represents the total free space in bin j to encourage higher fill rates. The second term, $(f_j + 1)(n + 1)$, is incorporated to promote genes with low family diversity. This encourages solutions where each bin contains fewer families and minimizes empty space within the bins. The third term, $(f_j+1)(n-m_j)$, is included to favor genes with low family diversity and high number of items. The last term, F_j^{HI}, is incorporated to account for the quantity of items from the same family and the number of families packed in a bin simultaneously. It is calculated by multiplying the total number of items for each family packed in the bin. A lower selection value indicates that the gene is more homogeneous in terms of product family composition.

Table 6 shows 11 examples of calculating δ in different packing scenarios with a single bin. The Gene column lists the items within each bin along with their respective family associations. The remaining columns, which also highlight different assessment perspectives, contain the information needed to calculate the selection value.

The selection procedure evaluates multiple scenarios through a comprehensive formula that considers family diversity, empty space utilization, and item density. When scenarios exhibit similar characteristics, four tie-breaking mechanisms are incorporated within the delta parameter δ to provide systematic ranking. Scenario 1 demonstrates a four-item bin configuration with items distributed between families 1 and 3. Scenarios 2 and 4 exhibit identical characteristics: both contain two distinct families, maintain 30% unused capacity, and achieve equivalent family unity scores alongside identical empty space ratios. This similarity creates selection ambiguity when multiple scenarios yield comparable fitness values. The delta parameter (δ) addresses this challenge by

TABLE 6. Sample scenarios for selection operation.

Scenario	Gene	$1-\mathrm{FR}_j$	$(f_j+1)(n+1)$	$(f_j+1)(n-m_j)$	F_j^{HI}	δ
1	1133	0.5	30	15	4	49.5
2	1133	0.3	30	15	4	49.3
3	113	0.5	30	18	2	50.5
4	113	0.3	30	18	2	50.3
5	2	0.4	20	18	1	39.4
6	222	0.4	20	12	3	35.4
7	222	0.6	20	12	3	35.6
8	1234	0.1	50	25	1	76.1
9	12344	0.6	50	20	2	72.6
10	111113	0.1	30	9	5	44.1
11	111333	0.1	30	9	9	48.1

introducing a tie-breaking mechanism that prioritizes solutions with reduced family diversity and higher item density, thereby enhancing the algorithm's ability to distinguish between seemingly equivalent alternatives. To resolve tie-breaking situations, Scenarios 10 and 11 were constructed with identical values across the first three deltaδ terms. To establish a systematic preference ranking between tied scenarios, we introduced the family heterogeneity index (FHI) that quantifies the degree of family unity within each gene set. For Scenario 10, the gene composition includes five items from family 1 and one item from family 3, yielding $F^{HI} = 5$. Scenario 11 produces $F^{HI} = 9$. Based on this quantitative assessment, Scenario 10 is superior to Scenario 11, as it demonstrates stronger family-based clustering. Scenarios 5, 6, and 7 have items from a single family. Since the empty space in Scenario 7 is greater than Scenario 6, the total selection value is better for Scenario 6. The number of product families and the empty space are the same in Scenario 5 and 6; however, the number of items packed in these bins are different. So, the solutions are distinguished in the third and fourth elements of the formula (6), which relates the connection between the item and product variety, and it is seen that scenario 6 is better. Finally, by evaluating each gene through various aspects using the selection value formula, the gene with the lowest selection value – specifically the bin in Scenario 6 – is selected.

For crossover operation, two chromosomes are selected as parents. Let Parent 1 and Parent 2 represent the solutions for 5 bins-10 items instance, where the volumes and families of items are identical to the instance in Table 5 and the volume of bins is assumed to be 60.

Parent 1: [**3** *20* 2 **1** *10* 1 : B1] [**10** *40* 3 : B2] [**6** *30* 2 **4** *25* 1 : B3]
[**5** *10* 2 **8** *10* 4 **2** *20* 1 : B4] [**7** *25* 3 **9** *20* 4 : B5]
Parent 2: [**3** *20* 2 **1** *10* 1 **2** *20* 1 : B1] [**4** *25* 1 **5** *10* 2 : B2] [**6** *30* 2 **7** *25* 3 : B3]
[**8** *10* 4 **9** *20* 4 : B4] [**10** *40* 3 : B5].

Using equation (6), the selection value of each gene is calculated as 58.63, 41.50, 58.31, 73.50, 58.44 for Parent 1 and 56.17, 58.56, 58.31, 40.63, and 41.50 for Parent 2, respectively. As crossover points, genes with a low selection value are chosen. The crossover operation begins with the exchange of genes between parent solutions.

Parent 1: [**3** *20* 2 **1** *10* 1 : B1] [**10** *40* 3 : B2] [**6** *30* 2 **4** *25* 1 : B3]
[**5** *10* 2 **8** *10* 4 **2** *20* 1 : B4] [**7** *25* 3 **9** *20* 4 : B5] [**8** *10* 4 **9** *20* 4 : B4]
Parent 2: [**3** *20* 2 **1** *10* 1 **2** *20* 1 : B1] [**4** *25* 1 **5** *10* 2 : B2] [**6** *30* 2 **7** *25* 3 : B3]
[**8** *10* 4 **9** *20* 4 : B4] [**10** *40* 3 : B5] [**10** *40* 3 : B2].

Gene 2 in Parent 1 and Gene 4 in Parent 2 are removed regarding the selection value, and the items in these genes are deleted from the chromosomes. For example, since the gene transferred to Parent 1 includes items 8

and 9, we remove them from Parent 2, and vice versa. Subsequently, the remaining items in the genes (in blue) are assigned to a new bin. When reallocating items to new bins, the families of items are prioritized, ensuring that the bin's capacity is not exceeded. If the bin cannot accommodate the item, the bin with the next-highest number of items from the same family is selected. If such a bin is unavailable, items are allocated to the first bin with sufficient capacity. If no bin meets the requirement, a new bin is created, and the items are packed into it. The new generation, consisting of Offspring 1 and Offspring 2, is then formed after the crossover operation, as follows.

Offspring 1: [**3** *20* 2 **1** *10* 1 **5** *10* 2 **2** *20* 1 : B1] [**10** *40* 3 **7** *25* 3 : B2] [**6** *30* 2 **4** *25* 1 : B3] [**8** *10* 4 **9** *20* 4 : B4]

Offspring 2: [**3** *20* 2 **1** *10* 1 **2** *20* 1 : B1] [**4** *25* 1 **5** *10* 2 : B2] [**6** *30* 2 **7** *25* 3 : B3] [**8** *10* 4 **9** *20* 4 : B4] [**10** *40* 3 : B5].

Following the crossover operation, the mutation operation is used to diversify the new population. The genes of the new population are mutated at a certain rate and are selected randomly. The selected gene is removed from the chromosome, and the items in this gene are reintroduced using the method available in the crossover, preserving the product family unity.

3.2. Phase II: wall building

In the first phase, the problem is solved as one-dimensional BPP, and the items are assigned to the bins considering the product family unity. The second phase addresses 3D-BPP, and the items are repositioned using the wall-building algorithm. The solutions from the first stage are accepted as the initial solution, and the product family unity is preserved as much as possible. The pioneering study of George and Robinson [26], Wall Building Algorithm, aims to solve 3D-BPP by generating layers parallel to the bin walls for efficient loading plans. This study suggests some modifications to the wall building algorithm. Firstly, in determining the layer size, the width and length of all the items are compared, and the largest edge of each item is determined. Then, the items are sorted according to their largest edge in descending order, and the loading sequence is created. The first item in the sequence is packed at the starting position parallel to the long side of the bin, forming the first layer. While the layers are being filled, the items from the family of the first item have higher priority. This process is repeated until there is no more space left in the layer, and a new layer is created. However, since the initial solution is obtained by solving one-dimensional BPP, it may yield an infeasible solution for 3D-BPP. In this situation, items that cannot be accommodated in any bin are listed in a pool. Additionally, items that decrease the product family ratio in the bin are also packed in the pool to be loaded again. While the items in the pool are being packed again, the families of items are considered first. If there is enough space, they are packed in the existing bins; otherwise, a new bin is introduced, and the remaining items are packed. An illustrative example for the wall building algorithm in this study is presented in Table 7, where the dimensions and families of items are summarized. The bins are assumed to be identical with dimensions of $100\times100\times100$ units for this example.

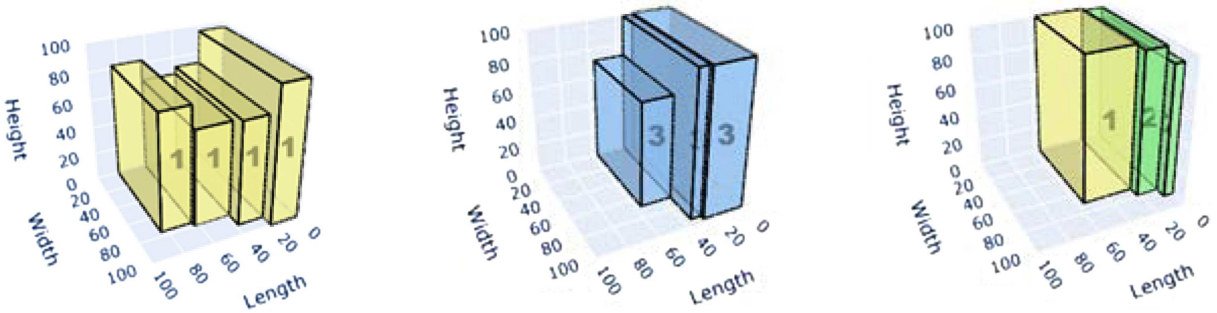
The initial solution obtained from Phase I is the items 4, 7, 10, 8, 2, 1, and 5 are assigned to Bin 1 and the items 9, 6, and 3 are assigned to Bin 2. The wall building algorithm establishes item packing sequences for each bin based on family prioritization. In Bin 1, item 1 (family 1) initiates the sequence at coordinates (0, 0, 0), followed by same-family items 10, 4, 7, and 2. Geometric constraints permit accommodation of items 1, 10, 4, and 7, while item 2 requires relocation. To maintain family unity, items 5 and 8 (family 2) are also designated for relocation. Bin 2 successfully accommodates its assigned sequence – items 9, 3, and 6 – as all belong to the same family and satisfy geometric feasibility requirements. The relocation pool contains items 2, 5, 8, necessitating a third bin and increasing the total bin count to three while preserving family grouping objectives. The item coordinates within each bin are provided in Table 8, and Figure 3 illustrates 3D packing configuration incorporating family constraints.

TABLE 7. Example 3D packing instance showing item dimensions and corresponding product families.

Item	1	2	3	4	5	6	7	8	9	10
Length	20	34	7	25	7	21	21	15	28	18
Width	100	71	94	80	87	69	76	79	97	91
Height	96	100	99	68	91	72	82	99	100	73
Family	1	1	3	1	2	3	1	2	3	1

TABLE 8. Solution for the example 3D-BPP instance with product family.

No	Bin 1 Coordinate			Family	No	Bin 2 Coordinate			Family	No	Bin 3 Coordinate			Family
	x	y	z			x	y	z			x	y	z	
1	0	0	0	1	9	0	0	0	3	5	0	0	0	2
10	0	20	0	1	3	0	28	0	3	8	0	7	0	2
4	0	38	0	1	6	0	35	0	3	2	0	22	0	1



(A) Bin 1 (B) Bin 2 (C) Bin 3

FIGURE 3. Packing plan for the example 3D-BPP instance with product family.

In order to assess the influence of product family constraints on packing configurations, the initial solution was generated by omitting the family unity ratio from the optimization process. In Phase 1, the initial solution allocates items 1, 2, 4, 6, and 9 to Bin 1 and items 1, 10, 7, 8, and 6 to Bin 2. This packing configuration achieves the expected improvement in bin utilization, requiring only 2 bins compared to the family-constrained solution. The corresponding packing plan is represented in Table 9 and illustrated in Figure 4.

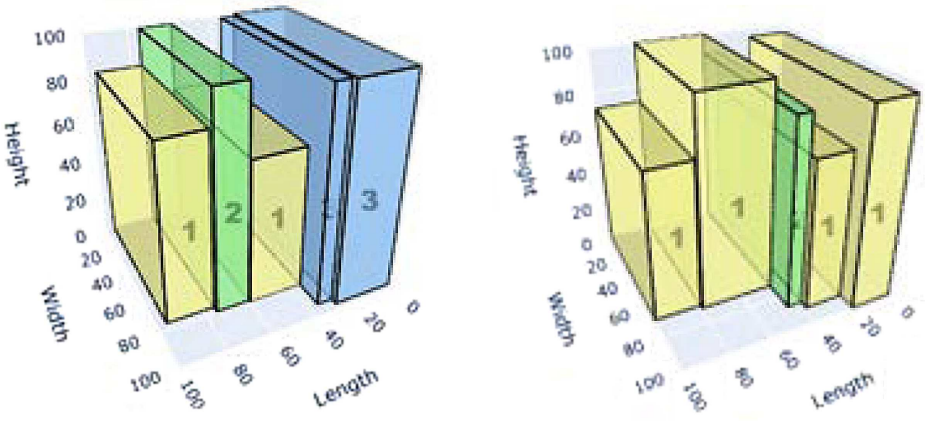
Excluding product family considerations reduces bin count, but bin quantity alone inadequately measures logistics performance. As established in the introduction, product family unity provides substantial operational advantages beyond spatial efficiency, including enhanced loading and unloading productivity, streamlined routing and material handling processes, and improved overall logistics coordination.

3.3. Phase III: improving load balance

The final phase aims to improve the load balance of the 3D loading plan that is created in Phase II. The objective of the final phase is to minimize the deviation from the ideal center of gravity by switching the positions of layers in each bin. Balanced load distribution in a bin eliminates several risks during transportation

TABLE 9. Solution for the example 3D-BPP instance without product family consideration.

Bin 1					Bin 2				
No	Coordinate			Product family	No	Coordinate			Product family
	x	y	z			x	y	z	
9	0	0	0	3	1	0	0	0	1
3	0	28	0	3	10	0	20	0	1
4	0	35	0	1	5	0	38	0	2
8	0	60	0	2	2	0	45	0	1
7	0	75	0	1	6	0	79	0	1



(A) Bin 1 (B) Bin 2

FIGURE 4. Packing plan for the example 3D-BPP instance without product family consideration.

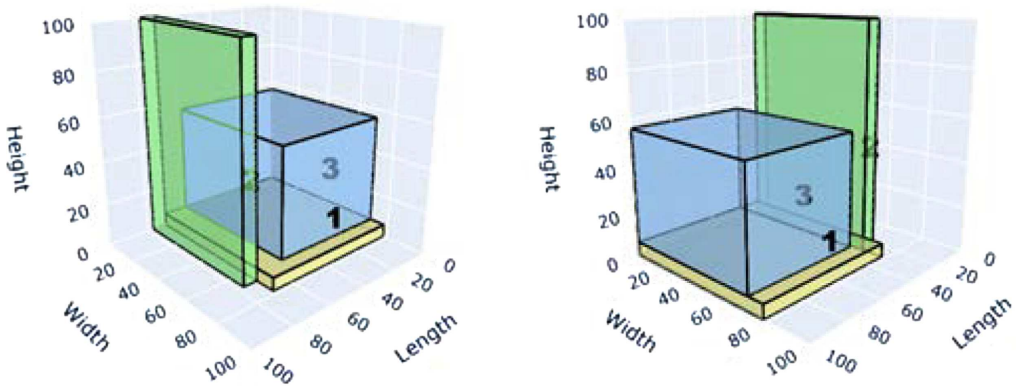
TABLE 10. Layer mirroring process (only the y-axis).

Items				Current coordinate			Coordinates after mirroring			Rotated	Layer
No	W	L	H	x	y	z	x	y	z		
1	78	70	6	0	0	0	0	30	0	xyz	
2	68	7	10	0	70	0	0	23	0	xyz	Layer1
3	58	70	50	0	0	6	0	42	6	yxz	

and increases load safety. With balanced loading plans, additional costs and potential risks associated with handling and transportation operations can be reduced [3]. Therefore, the center of gravity of the load in a bin is assumed to be close to the midpoint at the bottom of the bin, as proposed in the work of Eley [17].

In this study, we define a layer mirroring process in which the layers can be rotated in the x and/or y axis. An explanatory instance for rotation in y axis is presented in Table 10 and depicted in Figure 5.

In order to improve the load balance in the bin, we also investigate potential layer interchanges in addition to the layer mirroring procedure. Table 11 and Figure 6 provide an illustrative example of interchanging layers to improve the load balance in the bin.



(A) (B)

FIGURE 5. Illustration of mirrored layer on only the y-axis.

TABLE 11. Example of layers being rearranged within themselves.

Items				Current coordinate			Coordinates after interchanging			Rotated	Layer
No	W	L	H	x	y	z	x	y	z		
1	80	22	67	0	0	0	14	0	0	Xyz	
2	76	29	67	0	22	0	18	22	0	Xyz	Layer 1
3	74	48	94	0	51	0	20	51	0	Xyz	
4	68	14	79	80	0	0	0	0	0	Yxz	Layer 2

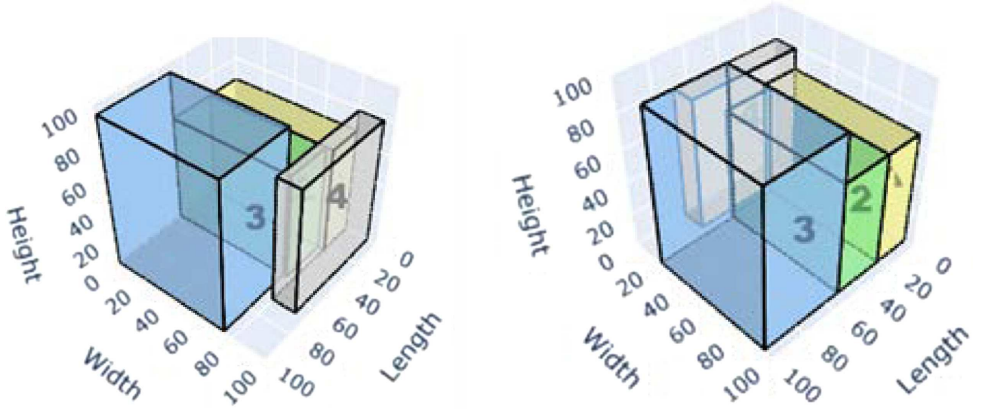
Finally, we examine the effect of shifting layers in the bin. Each layer is shifted in the x and/or y-axis in the bin, it is packed close to the ideal barycenter, and packing plans have been created. We illustrate the effect of layer shifting in Figure 6 and Table 11, the layer has been shifted to the right in the x-axis to improve load balance. Phase III focuses on evaluating all possible modifications across the layers within a bin to achieve balance. Initially, the impact of changes within individual layers is assessed through the mirroring process. Next, interactions between layers are analyzed using the layer interchange operation. Finally, the effects of shifting the entire load within the bin are explored through the layer shifting operation, all with the goal of improving load balance.

4. COMPUTATIONAL EXPERIMENTS

In this section, a numerical study with instances of literature is carried out to test for the algorithm's accuracy. Furthermore, the effect of the parameters used in the algorithm on the solution is examined. Eventually, a real-case study that stimulates this work to include item-related concerns in solving 3D-BPPs.

4.1. Numerical study

To verify and validate the proposed algorithm, a comparative analysis is performed with the heuristic algorithm for load-balanced 3D-BPPs introduced by Trivella and Pisinger [48]. The data for the 3D-BPP is sourced from Martello *et al.* [39] and further expanded by Trivella and Pisinger [48] through the inclusion of item weights. Erbayrak *et al.* [18] incorporate product family information into the existing dataset, as no dataset in the literature includes product families. The product family data is randomly generated and analyzed within



(A) (B)

FIGURE 6. Illustration of layers displaced among themselves.

the range of [1,3]. Ten instances are created for each data class, with the ranges specified. All bins are identical, with each bin having the same width, length, and height. In order to evaluate the performance of the proposed method for the 3D-BPP, we introduce Case I, which is consistent with Trivella and Pisinger [48], where all items belong to the same product family. This case serves as a baseline for comparing the proposed algorithm's performance in a simplified scenario where product family considerations are absent. To incorporate the product family unity concept into the 3D-BPP, we introduce Case II, where the items are divided into different product families. This case evaluates how the proposed algorithm handles the complexity introduced by family-based constraints and assesses its effectiveness in achieving load balance while respecting family unity. The algorithm is coded in Java on an Intel i5 2.50 GHz computer with 8.0 GB RAM and 64-bit operating system. For each class, we generate 10 instances and present the results by calculating the average of these 10 instances for each case. The results are provided in Tables 12 and 13, respectively, for Case I and Case II.

When the results are compared, they align with those of Trivella and Pisinger [48] for Classes 1, 2, 3, 4, and 5, as shown in Table 12. Notably, the results for Class 2 demonstrate an improved solution since the proposed algorithm, which is based on the wall building method, prioritizes items with larger length/width. The wall building algorithm performs better for the data with low variety in item dimensions. Recent studies in packing problems, especially those involving two and 3D-BPP, highlight the impact of item dimension variability on packing efficiency [7]. In particular, algorithms designed to handle problems with low variation in item dimensions tend to perform more efficiently, as the consistency in item size allows for better and predictable packing strategies. This aligns with our findings, where the proposed algorithm demonstrates improved performance in scenarios where there is low variability in item dimensions.

In contrast, for Classes 6, 7, and 8, where there is higher diversity in item dimensions, the number of bins used increases. These results also show that as the number of items increases, so does the number of bins, and the total deviation from the ideal barycenter decreases. Furthermore, the deviation from the ideal center of gravity remains consistent across all classes. The proposed heuristic algorithm has been solved by ignoring the product family and its reliability is tested by comparing the results with the literature, as represented in Case I. Then, the data for Case II are created to reveal the impact of the family unity in respect to two objectives: minimizing the number of bins used and maximizing family unity in bins. Table 13 summarizes the results, where in Case II(a) we prioritize the objective of minimizing the number of bins used, and in Case II(b) we prioritize the objective of maximizing family unity in bins. The columns below Case II(a) present the number of bins that are filled homogeneously in terms of product family (NBFB), when the objective of

TABLE 12. Comparative Results for the problem without product family.

CASE I Objective function: minimum bin used						
Results of Trivella and Pisinger (2016)				Phase I	Phase II	Phase III
Class	Size	Bins	Deviation	Bins	Bins	Deviation
1	100	25.64	41.98	17.1	26.4	47.25
2	100	26.12	44.49	16	25.50	45.02
3	100	26.08	41.92	17,4	26.9	47.48
4	100	60.60	38.42	31,7	61	41.65
5	100	14.60	42.35	9,7	15.2	47.13
6	100	20.08	44.91	13.5	22.50	46.40
7	100	12.36	42.36	8.5	14.40	45.04
8	100	17.08	42.08	12.37	20.75	45.46
1	200	51.16	42.56	33.9	51.40	47.20
2	200	50.80	45.20	31.3	49.10	45.84
3	200	51.24	42.67	34.4	52.00	47.15
4	200	122.2	38.19	61.80	122.80	41.17
5	200	27.36	44.14	18.70	28.90	46.62
6	200	38.24	45.77	27.00	42.00	46.14
7	200	22.40	44.51	16.6	28.3	45.26
8	200	32.04	44.37	27.0	43.2	46.37

minimizing the total number of bins used (NB) is employed. The columns below Case II(b) present the number of bins used (NB), when the objective function aims to maximize family unity in used bins. When the results of Case II(a) and Case II(b) are compared with respect to the objective of minimizing the number of bins used, the results are consistent. Moreover, the number of homogeneously filled bins increases considerably in Case II(b). The objectives of minimizing the number of bins used, maximizing family unity, and achieving the best load balance in bins are conflicting objectives as stated in Erbayrak *et al.* [18]. Therefore, as the number of homogeneously filled bins increases, the total deviation from the ideal barycenter is expected to rise. However, this comparative analysis indicates that the total deviation remains unaffected by this change. The primary reason for this outcome is the increase in the number of bins used. As more bins are employed, additional free space is available within the bins. This extra space provides the last phase of the proposed algorithm with more opportunities to evaluate replacement options between layers, ultimately improving the load balance within the bins. Consequently, it is seen that there is a trade-off between the objectives of minimizing the number of bins used and maximizing family unity. However, the maximizing family unity and load balancing objectives can be improved simultaneously due to the structure of the proposed algorithm. This reflects the nature of multi-objective optimization problems and reveals that different solution alternatives can be obtained depending on the decision-maker's priorities.

Furthermore, we investigate the solutions by varying the importance weights assigned to these objectives to assess their impact on the results. This analysis helps to understand how different weightings influence the overall performance and balance between the competing objectives. We generate a dataset from Class 6 and investigate 7 different scenarios by varying the weights assigned to the objectives. This allows us to analyze how different weightings affect the solution and the overall performance of the algorithm under various conditions. Table 14 presents the solutions obtained under different weight settings, where w_1 denotes the importance of minimizing the total number of bins objective, and w_2 denotes the importance of maximizing family unity objective. As observed, increasing w_1 leads to solutions with higher space utilization, whereas increasing w_2 promotes the number of bins with items from the same product family, thereby highlighting the trade-off between packing efficiency and family unity.Scenario 1 and Scenario 7 represent the solutions for single-objective cases, while

TABLE 13. Comparative results for Case II (a) and Case II (b).

Class	Size	Case II (a) Objective function: minimize number of bins used				Case II (b) Objective function: maximize family unity			
		Phase I	Phase II		Phase III	Phase I	Phase II		Phase III
		NB (1D)	NB (3D)	NHFB (3D)	Deviation	NB (1D)	NB (3D)	NHFB (3D)	Deviation
1	100	17.10	26.90	12.30	47.82	17.20	27.90	21.10	47.90
2	100	16.00	26.70	13.90	44.90	16.00	27.80	18.90	44.89
3	100	17.40	27.50	11.60	47.84	17.40	28.60	21.40	47.52
4	100	31.90	61.10	49.60	41.61	31.90	63.30	61.80	41.32
5	100	9.70	15.80	5.30	47.83	9.70	16.10	11.50	47.20
6	100	13.70	22.70	7.11	46.35	13.70	22.50	16.80	45.80
7	100	8.50	15.00	2.60	45.77	8.50	15.70	8.50	44.04
8	100	11.5	20.7	9.1	45.60	11.5	21.8	16.1	44.90
1	200	34.10	52.40	23.96	47.13	34.10	54.10	42.10	47.22
2	200	31.40	49.80	25.93	46.43	31.40	51.00	35.60	45.81
3	200	34.20	53.20	28.4	47.84	34.20	54.40	39.80	47.62
4	200	64.50	121.90	98.96	41.60	64.50	125.00	122.00	41.32
5	200	18.70	29.60	9.93	47.10	18.70	30.30	22.80	47.20
6	200	27.40	42.50	17.7	46.00	27.40	43.40	32.10	45.80
7	200	16.60	28.60	4.96	45.04	16.60	28.70	16.60	44.04
8	200	27.9	46.9	26.3	45.80	27.9	48.9	37	44.90

TABLE 14. Comparative results for different weight combinations.

Scenario	w_1	w_2	Homogeneous bins	Used bins	Total deviations
1	0	1	31	41	45.5
2	0.1	0.9	30	40	48.1
3	0.3	0.7	26	39	46.6
4	0.5	0.5	25	37	48.6
5	0.7	0.3	24	37	49.1
6	0.9	0.1	23	37	48.9
7	1	0	13	36	48.8

Scenarios 2 to 6 demonstrate the results of weighted aggregation of multiple objectives. These scenarios enable a comparative analysis of the performance under single-objective optimization versus the aggregation of multiple objectives with different weightings.

The analysis results are illustrated in Figure 7, which shows that Scenario-1 ($w_1 = 0, w_2 = 1$) exhibits the highest family unity ratio and Scenario-7 ($w_1 = 1, w_2 = 0$) exhibits the lowest. The number of bins used in Scenarios 4-5-6 has not changed. As the weight given to the objective of maximizing family unity increased, the number of homogeneously filled bins in terms of product family also increased. Figure 7 shows the minimum number of used bins and the number of homogeneously filled bins for 7 scenarios. In the proposed heuristic algorithm, the decision-maker will be able to create loading plans in line with the weights to be determined by considering the loading strategies.

The performance of the proposed algorithm for the 3D Bin Packing Problem is assessed using three key metrics: the number of bins used, the number of homogeneous bins, and the deviation of balance from the

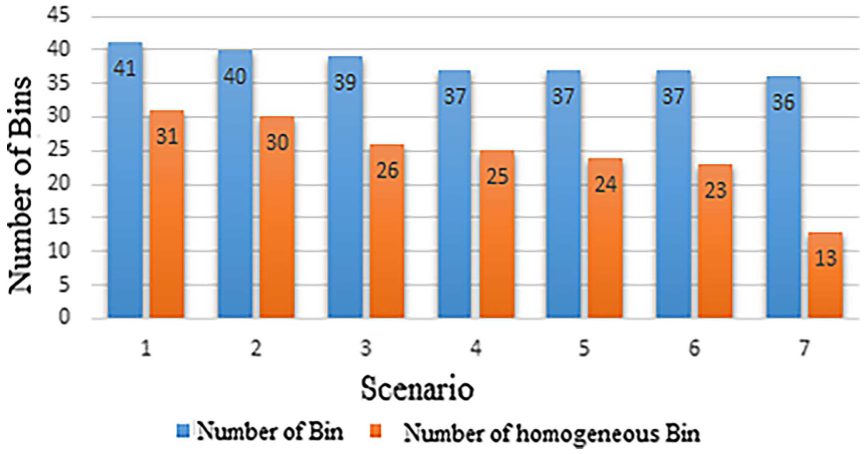


FIGURE 7. Number of bins for each scenario.

TABLE 15. The computational performance of the proposed algorithm.

Ins.	Case I – without product family				Case II – with product family				
	Used bins	Deviations	Best iteration	Run time	Used bins	Max. homogeneous bins	Deviations	Best iteration	Run time
1	28	45.20	4	357	32	21	45.87	1	125
2	22	43.50	1	167	29	21	44.50	1	174
3	24	43.82	1	179	28	18	42.93	3	157
4	22	41.80	1	208	29	22	41.41	3	99
5	24	45.12	1	292	28	18	44.10	1	101
6	25	46.65	3	235	26	18	46.25	4	106
7	28	46.81	4	321	23	16	47.41	2	86
8	27	45.21	1	216	27	17	46.21	1	85
9	26	46.25	2	366	24	15	46.14	1	92
10	29	44.66	1	345	32	23	44.16	2	164

ideal center of gravity. To analyze the algorithm's computational performance, we examine run time efficiency, convergence speed, and robustness. For this purpose, 10 datasets are generated, and each dataset is subjected to five evaluation iterations. The analyses are conducted under two scenarios: in the first scenario, the product family is excluded, while in the second scenario, it is incorporated into the algorithm. Results excluding the product family are reported below. The results are provided in Table 15.

In the analysis of Case I, the average run time is computed to be 269 s. Regarding the convergence speed, the algorithm achieved its best results mainly within the first 2 iterations, highlighting its rapid convergence capability. The analysis revealed an average usage of 25.5 bins, with a low standard deviation of approximately ± 2.3 bins, indicating consistent performance of the algorithm. For Case II, when considering the product family, the average run time is calculated to be 119 s. Similar to the previous case, the best results were predominantly achieved within the first 2 iterations, demonstrating that the algorithm maintains its high level of convergence performance. In the analysis for this case, the average number of homogeneous bins used is 18.9, with a standard deviation of 2.6. The average number of bins utilized is 27.8, with a standard deviation of 2.9. These relatively low variations further illustrate the algorithm's consistent performance across different problem instances. When

TABLE 16. Comparison of mutation rates.

Instance	Mutation rate: 0.01				Mutation rate: 0.05			
	Min bin used		Max family unity		Min bin used		Max family unity	
	NB (3D)	NHFB (3D)	NHFB (3D)	NB (3D)	NB (3D)	NHFB (3D)	NHFB (3D)	NB (3D)
1	30	20	21	32	31	23	24	32
2	27	18	21	29	27	19	20	28
3	28	18	18	28	27	14	17	28
4	29	22	22	29	28	16	20	29
5	25	16	18	28	27	14	18	27
6	25	17	18	26	25	14	19	26
7	23	16	16	23	24	13	14	24
8	25	12	17	27	26	13	18	26
9	24	15	15	24	24	13	16	25
10	29	21	23	32	28	17	21	30
Average	26.5	17.5	18.9	27.8	26.7	15.6	18.7	27.5

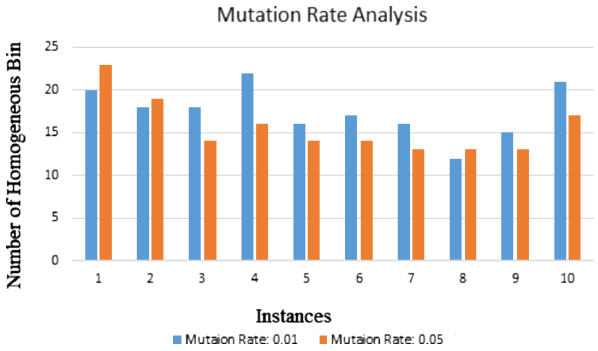


FIGURE 8. Number of bins for each scenario.

product families are taken into account, the reduction observed in run time can be attributed to a smaller, more structured search space resulting from this approach. When product families are not considered, the algorithm aims to minimize the number of bins by comprehensively evaluating all possible combinations, which necessitates a higher level of computational effort. In contrast, by grouping similar products, the consideration of product families significantly decreases the complexity of the problem and accelerates the run time. This implies that the proposed approach not only improves run time but also maintains a rapid convergence speed when employing family-based data structures. Overall, the experimental results indicate that the proposed algorithm exhibits high robustness, delivering consistent performance across diverse datasets with minimal variation in solution quality between different runs.

Furthermore, this study provides an analysis for different mutation rates. For each mutation rate, the performance of the genetic algorithm is tested for the same dataset. There are slight variations in the outcomes when we changed the mutation rates between 0.01 and 0.39. Notably, when the mutation rates are set at 0.01 and 0.05, the number of homogeneous bins varied relatively. The results for these two mutation rates are presented in Table 16.

Both mutation rates displayed similar performance; however, the average homogeneous bin count is greater at a mutation rate of 0.01 compared to 0.05. Figure 8 shows the number of homogeneously filled bins for 10 instances. Therefore, a mutation rate of 0.01 is chosen for the remainder of the study.

TABLE 17. Effect of crossover rate on solution quality.

Instance	Crossover rate	Best solution (All Runs)		Average of all runs		Run time (s)
		Min bin used	Max homog. bins	Avg. min bin used	Avg. max homog.	
1	0.95	33	23	32.2	19.4	29
	0.90	32	22	31.4	18.8	40
	0.85	32	24	31.8	21.0	37
	0.80	33	22	32.0	19.8	30
	0.75	33	24	32.4	20.6	32
	0.70	31	21	31.0	19.6	52
	0.65	33	23	31.6	19.4	50
2	0.95	30	22	28	18	36
	0.90	29	20	27.0	16.8	35
	0.85	28	21	27.8	19.8	31
	0.80	28	19	28.2	17.6	42
	0.75	29	20	28.2	18.6	33
	0.70	29	20	28.4	17.4	42
	0.65	27	20	27.2	18.0	40

Lastly, we examined the impact of the crossover rate on the performance of the algorithm. Increasing the crossover probability generally accelerated convergence by promoting greater genetic diversity early in the search. To evaluate the effects of this rate, we conducted analysis on 2 different data sets. For each data set, the crossover rate is varied from 0.65 to 0.95, and the algorithm is executed 5 times for each rate. The results are shown in Table 17.

According to the results, the crossover rate has a decisive effect on solution quality, homogeneity, and run time. Crossover rate (0.75–0.85) provides sufficient diversity to ensure a balanced number of homogeneous bins while also yielding satisfactory results in terms of minimum bin usage. Lower crossover rates (<0.70) may improve minimum bin usage to some extent, but they increase the run time and can have negative effects on homogeneity. On the other hand, while high crossover rates (>0.90) accelerate the solution, they cause early convergence, reducing diversity and showing more limited performance in terms of minimum bin usage and homogeneity. Based on these findings the optimal crossover rate for this problem type has been determined to be 0.85.

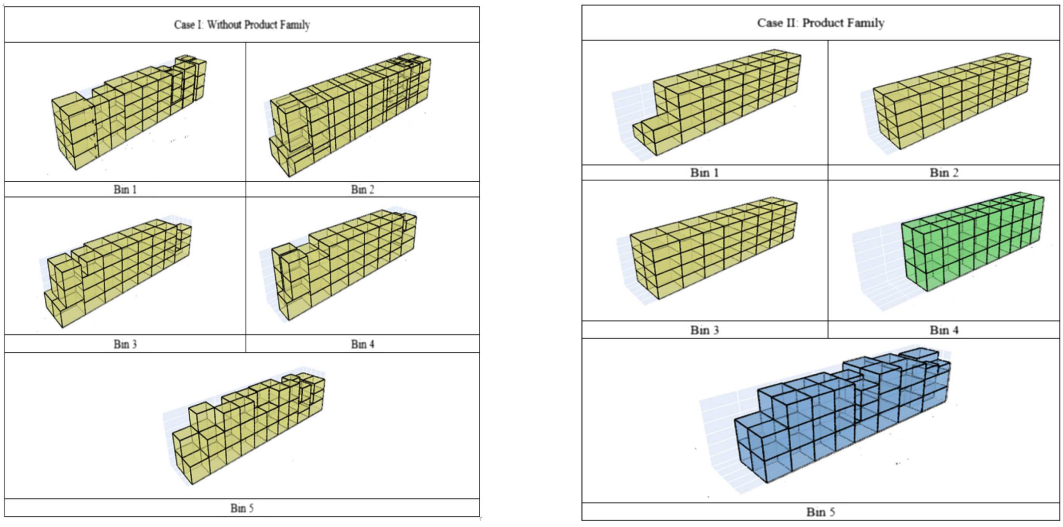
4.2. Real case study

This section provides an application of the proposed algorithm for a real case study in Portugal, as referenced by Costa and Captive [10]. The dataset encompasses the loading plan for a specific day along with the characteristics of the products to be transported. This dataset categorizes items into 10 types based on size, rather than product families. Therefore, we identify three categories of product families based on their sizes and overall quantity. The extended data presented in Table 18 is analyzed in two distinct scenarios: Case I, which disregards product family characteristics, and Case II, which takes product families into account. This analysis aims to evaluate the impact of product families on the outcomes.

The layouts of the items in each bin are illustrated in Figures 9a and 9b. The total number of bins used is determined as five for both cases, Case I and Case II, thereby aligning with the study conducted by Costa and Captive [10]. In Case II, the products belonging to the same product family are packed together, as shown in Figure 9b. Consequently, the proposed algorithm demonstrated its effectiveness when applied to real-world problems, yielding outcomes that are consistent with existing literature. Grouping similar products significantly improved the efficiency of loading plans in terms of time and cost, while also enhancing handling

TABLE 18. Item dimensions and weights of Instance [10].

Item	Length (mm)	Width (mm)	Height (mm)	No. of items	Weight (kg)	Product family
1	1200	800	460	1	29.98	3
2	1200	800	592	1	13	3
3	1200	1000	750	3	44.1	3
4	1600	1200	750	187	66.26–95	1
5	2080	1200	930	8	191.4	3
6	2400	800	950	4	240.9	3
7	1200	1000	975	16	66.36–70.43	3
8	1200	1000	990	7	23.85–42.46	3
9	1210	1010	990	54	68.60–100.92	2
10	1600	1200	1000	8	82.99–152.60	3



(A) Case I (B) Case II

FIGURE 9. Packing plan for Cases I–II.

processes. Additionally, the method's capability to maintain bin homogeneity reduces the need for various handling equipment and, simplifies loading and delivery planning.

5. CONCLUSIONS AND FUTURE RESEARCH

Efficient loading strategies are critical in logistics operations, where poor planning leads to increased transportation costs and operational delays. A key real-world constraint often overlooked in academic literature is product families – items sharing similar characteristics that benefit from being packed together. Family-based packing improves handling efficiency, optimizes storage space utilization, and enhances inventory management, leading to reduced operational costs and improved service levels. Therefore, integrating product families into bin packing algorithms not only reflects real-world logistics constraints but also provides practical benefits that enhance overall system efficiency. This study introduces a novel genetic algorithm-based approach for the load-balanced 3D bin packing problem (3D-BPP) that maximizes the number of homogeneously filled bins while considering item orientations and product family unity. The algorithm was validated using extended instances

from Trivella and Pisinger [48], with performance evaluation across multiple item classes. The empirical results demonstrate that the algorithm performs optimally for Class 2 data, where item dimensions exhibit low diversity. Performance decreases as item dimensional diversity increases in Classes 6, 7, and 8, resulting in higher bin utilization. As expected, increasing the number of items leads to more bins being used, while deviations from the ideal barycenter decrease. Notably, center-of-gravity deviations remain consistent across all classes. When product families are introduced, two distinct scenarios emerge: Case II(a) minimizes total bins while optimizing homogeneous bins, achieving results comparable to existing literature; Case II(b) maximizes homogeneous bins at the cost of increased total bin usage. The contributions of this study to the literature are as follows:

- We introduce hybrid genetic algorithm specifically designed for load-balanced 3D-BPP with integrated product family considerations, addressing a significant gap in the literature.
- This study simultaneously addresses multiple practical considerations typically treated separately, including stacking rules, bi-directional item rotation, and family unity constraints.
- We integrate load balancing directly into the optimization process, improving loading stability and transportation efficiency for real-world logistics applications.
- This study demonstrates strong potential for real-world implementation through empirical testing, providing solutions that effectively balance load requirements while maintaining product family cohesion.

The algorithm's performance decreases with higher dimensional diversity, suggesting opportunities for adaptive mechanisms. Future research could explore dynamic family grouping strategies and integration with vehicle routing problems for comprehensive logistics optimization. Also, different pareto-based approaches could be tested as they have the potential to further enrich the analysis and offer broader set of compromise solutions. Along with the objectives of this study, objectives such as reducing operational costs and packing time could be integrated into the algorithm. Additionally, incorporating stochastic elements, like uncertain demands could enhance the algorithm's adaptability for real-world logistics applications.

This research bridges the gap between theoretical bin packing algorithms and practical logistics requirements, offering a robust solution that addresses real-world constraints while maintaining computational efficiency. The proposed approach provides logistics companies with a practical tool for optimizing loading operations while respecting product family requirements, ultimately contributing to more efficient and cost-effective transportation operations.

Data availability statement

The data used in this paper is available online in a Github repository: https://github.com/sedaerbayrak/3D-Binpacking-with-product-family.

References

[1] M.T. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño and J.M. Tamarit, Mathematical models for multicontainer loading problems. *Omega* **66** (2017) 106–117.

[2] I. Araya and M.C. Riff, A beam search approach to the container loading problem. *Comput. Oper. Res.* **43** (2014) 100–107.

[3] E.E. Bischoff and M.S.W. Ratcliff, Issues in the development of approaches to container loading. *Omega* **23** (1995) 377–390.

[4] A. Bortfeldt and H. Gehring, A hybrid genetic algorithm for the container loading problem. *Eur. J. Oper. Res.* **131** (2001) 143–161.

[5] A. Bortfeldt and G. Wäscher, Constraints in container loading – a state-of-the-art review. *Eur. J. Oper. Res.* **229** (2013) 1–20.

[6] A. Bortfeldt, H. Gehring and D. Mack, A parallel tabu search algorithm for solving the container loading problem. *Parallel Comput.* **29** (2003) 641–662.

[7] F. Braam and D. van den Berg, Which rectangle sets have perfect packings? *Oper. Res. Perspect.* **9** (2022) 100211.

[8] P.B. Castellucci, F.M. Toledo and A.M. Costa, Output maximization container loading problem with time availability constraints. *Oper. Res. Perspect.* **6** (2019) 100126.
[9] C.S. Chen, S.M. Lee and Q.S. Shen, An analytical model for the container loading problem. *Eur. J. Oper. Res.* **80** (1995) 68–76.
[10] M.G. Costa and M.E. Captivo, Weight distribution in container loading: a case study. *Int. Trans. Oper. Res.* **23** (2016) 239–263.
[11] T.G. Crainic, L. Gobbato, G. Perboli and W. Rei, Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. *Eur. J. Oper. Res.* **253** (2016) 404–417.
[12] A. de Almeida and M.B. Figueiredo, A particular approach for the three-dimensional packing problem with additional constraints. *Comput. Oper. Res.* **37** (2010) 1968–1976.
[13] J.L. de Castro Silva, N.Y. Soma and N. Maculan, A greedy search for the three-dimensional bin packing problem: the packing static stability case. *Int. Trans. Oper. Res.* **10** (2003) 141–153.
[14] T.A. de Queiroz, F.K. Miyazawa, Y. Wakabayashi and E.C. Xavier, Algorithms for 3D guillotine cutting problems: unbounded knapsack, cutting stock and strip packing. *Comput. Oper. Res.* **39** (2012) 200–212.
[15] T. Dereli and G.S. Das, A hybrid 'bee(s) algorithm' for solving container loading problems. *Appl. Soft Comput.* **11** (2011) 2854–2862.
[16] T. Dokeroglu and A. Cosar, Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms. *Comput. Ind. Eng.* **75** (2014) 176–186.
[17] M. Eley, A bottleneck assignment approach to the multiple container loading problem. *OR Spectr.* **25** (2003) 45–60.
[18] S. Erbayrak, V. Özkır and U.M. Yıldırım, Multi-objective 3D bin packing problem with load balance and product family concerns. *Comput. Ind. Eng.* **159** (2021) 107518.
[19] E. Falkenauer, A hybrid grouping genetic algorithm for bin packing. *J. Heuristics* **2** (1996) 5–30.
[20] Y. Fan, J. Chu and H. Xu, Improvement grouping genetic algorithm for solving the bin packing problem. *J. Phys. Conf. Ser.* **1550** (2020) 032168.
[21] T. Fanslau and A. Bortfeldt, A tree search algorithm for solving the container loading problem. *INFORMS J. Comput.* **22** (2010) 222–235.
[22] X. Feng, I. Moon and J. Shin, Hybrid genetic algorithms for the three-dimensional multiple container packing problem. *Flex. Serv. Manuf. J.* **27** (2015) 451–477.
[23] G. Fuellerer, K.F. Doerner, R.F. Hartl and M. Iori, Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *Eur. J. Oper. Res.* **201** (2010) 751–759.
[24] M. Gajda, A. Trivella, R. Mansini and D. Pisinger, An optimization approach for a complex real-life container loading problem. *Omega* **107** (2022) 102559.
[25] H. Gehring and A. Bortfeldt, A genetic algorithm for solving the container loading problem. *Int. Trans. Oper. Res.* **4** (1997) 401–418.
[26] J.A. George and D.F. Robinson, A heuristic for packing boxes into a container. *Comput. Oper. Res.* **7** (1980) 147–156.
[27] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem – part II. *Oper. Res.* **11** (1963) 863–888.
[28] J.F. Gonçalves and M.G. Resende, A biased random key genetic algorithm for 2D and 3D bin packing problems. *Int. J. Prod. Econ.* **145** (2013) 500–510.
[29] Y. Gonzalez, G. Miranda and C. Leon, Multi-objective multi-level filling evolutionary algorithm for the 3D cutting stock problem. *Procedia Comput. Sci.* **96** (2016) 355–364.
[30] K. He and W. Huang, An efficient placement heuristic for three-dimensional rectangular packing. *Comput. Oper. Res.* **38** (2011) 227–233.
[31] M. Iori and S. Martello, Routing problems with loading constraints. *Top* **18** (2010) 4–27.
[32] T. Jamrus and C.F. Chien, Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. *Comput. Ind. Eng.* **96** (2016) 227–236.
[33] İ. Kabasakal and F.D. Keskin, Decision support system based on genetic algorithm for variable sized bin packing problem with item conflicts. *Int. Rev. Econ. Manag.* **5** (2017) 1–17.
[34] K. Kang, I. Moon and H. Wang, A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Appl. Math. Comput.* **219** (2012) 1287–1299.
[35] A. Lim, H. Ma, J. Xu and X. Zhang, An iterated construction approach with dynamic prioritization for solving the container loading problems. *Expert Syst. Appl.* **39** (2012) 4292–4305.
[36] J. Liu, Y. Yue, Z. Dong, C. Maple and M. Keech, A novel hybrid tabu search approach to container loading. *Comput. Oper. Res.* **38** (2011) 797–807.

[37] A. Lodi, S. Martello and D. Vigo, Tspack: a unified tabu search code for multi-dimensional bin packing problems. *Ann. Oper. Res.* **131** (2004) 203–213.
[38] S. Martello and D. Vigo, Exact solution of the two-dimensional finite bin packing problem. *Manag. Sci.* **44** (1998) 388–399.
[39] S. Martello, D. Pisinger and D. Vigo, The three-dimensional bin packing problem. *Oper. Res.* **48** (2000) 256–267.
[40] N. Mohamadi, Application of genetic algorithm for the bin packing problem with a new representation scheme. *Math. Sci.* **4** (2010) 253–266.
[41] I. Moon and T.V.L. Nguyen, Container packing problem with balance constraints. *OR Spectr.* **36** (2014) 837–878.
[42] A. Moura and J.F. Oliveira, A grasp approach to the container-loading problem. *IEEE Intell. Syst.* **20** (2005) 50–57.
[43] F. Parreño, R. Alvarez-Valdés, J.M. Tamarit and J.F. Oliveira, A maximal-space algorithm for the container loading problem. *INFORMS J. Comput.* **20** (2008) 412–422.
[44] D. Pisinger, Heuristics for the container loading problem. *Eur. J. Oper. Res.* **141** (2002) 382–392.
[45] A.G. Ramos, E. Silva and J.F. Oliveira, A new load balance methodology for container loading problem in road transportation. *Eur. J. Oper. Res.* **266** (2018) 1140–1152.
[46] L. Sheng, S. Xiuqin, C. Changjian, Z. Hongxia, S. Dayong and W. Feiyue, Heuristic algorithm for the container loading problem with multiple constraints. *Comput. Ind. Eng.* **108** (2017) 149–164.
[47] C.D. Tarantilis, E.E. Zachariadis and C.T. Kiranoudis, A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Trans. Intell. Transp. Syst.* **10** (2009) 255–271.
[48] A. Trivella and D. Pisinger, The load-balanced multi-dimensional bin-packing problem. *Comput. Oper. Res.* **74** (2016) 152–164.
[49] Y. Wu, W. Li, M. Goh and R. de Souza, Three-dimensional bin packing problem with variable bin height. *Eur. J. Oper. Res.* **202** (2010) 347–355.
[50] L.H.W. Yeung and W.K.S. Tang, A hybrid genetic approach for container loading in logistics industry. *IEEE Trans. Ind. Electron.* **52** (2005) 617–627.