Solving 3D Container Loading Problems Using Physics Simulation for Genetic Algorithm Evaluation

Shuhei NISHIYAMA \(^{1}\), Chonho LEE\(^{1,3}\), Nonmembers, and Tomohiro MASHITA\(^{1}\), Member

SUMMARY In this work, an optimization method for the 3D container loading problem with multiple constraints is proposed. The method consists of a genetic algorithm to generate an arrangement of cargo and a fitness evaluation using a physics simulation. The fitness function considers not only the maximization of the container density and fitness value but also several different constraints such as weight, stack-ability, fragility, and orientation of cargo pieces. We employed a container shaking simulation for the fitness evaluation to include constraint effects during loading and transportation. We verified that the proposed method successfully provides the optimal cargo arrangement for small-scale problems with about 10 pieces of cargo.

**key words:** container loading, genetic algorithm, physics simulation

1. Introduction

The container loading problem (CLP) involves finding the optimal arrangement of cargo pieces in a container, and it arises in various scenarios in daily life and business including packing a suitcase for traveling, bagging purchased items in a supermarket, and packing a container in logistics. Such packing problems are practically solved by the common sense of the person doing the packing. However, to obtain an acceptable solution quickly, the person doing the packing needs some level of experience or training.

To obtain a reasonable loading pattern in a practical packing scenario by computation has significant advantages, including training and evaluation of a person working on a packing task, and optimizing packing by robots in an automated logistics system. A solver for a practical problem setting basically should be customized for that particular problem because of constraints of weight, fragility, orientation, and stability of the loading pattern. Moreover, for each problem, the importance of the constraints are different. Bortfeldt and Wäschler [3] reviewed many CLPs with different types of constraints. To date, studies have focused on specific application scenario, and although they meet the demands of practical situations, solutions are available only for specific parts of the problem. To adapt them to another type of problem, the whole algorithm must be revised. Conversely, meta-heuristic methods are generally more flexible and adaptable for different problem settings.

In this paper, we propose a method to obtain a reasonable loading pattern in a practical scenario with various constraints and consider the importance of these constraints. Our method consists of a meta-heuristic optimization and a physics simulation (PS) of the loading pattern. In the PS, we shake the container to simulate motion during transportation and evaluate the possible damage to individual cargo pieces. Thus, the proposed method involves not only a static evaluation, including container density, but also an evaluation considering damage during transportation, which is caused by the weight, stack-ability, fragility, and orientation of cargo pieces.

Contribution

We focus on a practical container loading scenario with constraints such as weight, stack-ability, fragility, and orientation of cargo pieces. To solve this problem, this paper proposes a flexible method combining a genetic algorithm (GA) and a PS. This method separates the design of GA, arrangement of cargo, and fitness evaluation. The method can be adapted to various situations and constraints by modifying only the fitness function. The function can be designed with the ample information obtained from the PS. The method simulates an acceleration scenario by adding velocity to the container and cargo. We obtain information such as trails of individual cargo pieces and contact between cargo pieces, from the simulation and use it for calculating the fitness value with respect to container loading.

2. Related Work

2.1 Genetic Algorithm

The CLP is known as an NP-hard optimization problem, and it has been approached using meta-heuristic algorithms [4], [5]. This paper focuses on GAs [6], which are inspired by the process of natural selection to solve optimization problems. A GA repeatedly modifies a population of candidate solutions called individuals to obtain better solutions. Each individual has genes representing a solution, which are encoded in many ways such as bit-strings [7], real values [8] and permutations [9]. At each step, the GA iteratively applies genetic operations (GOs) (e.g., crossover, mutation and selection) to one or several individuals (called parents) and
produces new individuals (called children). Children inherit some of their parents’ genes, which are variables of the solution. In the selection process, individuals are evaluated by a fitness function, and those with higher fitness will survive to the next generation.

In the CLP context, genes represent how cargo is loaded in a specific manner. For example, a real-polarized GA [10] encodes the loading order of the cargo pieces as its genes. Wu [11] used two segments of encoding in a GA, including the number and the rotation of the cargo pieces. In the sequence-triple method [12], genes represent the cargo positions with three arrays of cargo order. Relative positions of each gene represent relative spatial positions. Similar to [9], [10], we encoded the loading order into the genes. The loading location is straightforward in a bottom-left-back manner. In contrast with other work, we ran the GA with a PS to compute the fitness under several realistic constraints (e.g., rotation, stability, and fragility) in a practical scenario.

2.2 Container Loading with Soft Constraints

In the review by Bortfeldt and Wäscher [3], they state:

Constraints in container loading are usually introduced as hard constraints. This may be due to the fact that in the design of algorithms such constraints can be handled in a more straightforward way than soft constraints. Correspondingly, only very few publications consider soft constraints.

Many studies have handled constraints as hard constraints, and only a few types of constraints such as weight constraints, allocation constraints and positioning constraints were addressed as soft constraints. Our work tries to represent more types of constraints as soft constraint and handle them simultaneously.

2.3 Physics Simulation

PS solve problems with calculations involving the laws of physics. By calculating the motions of multiple objects (multibody dynamics), useful information can be obtained for packing evaluation such as the stability of cargo arrangements. Evaluation of stability can be classified into static stability and dynamic stability. Truong et al. [1] used the idea of base support to evaluate static stability and evaluated whether the bottom of one piece of cargo is supported by other pieces. However, in the evaluation of static stability, it is difficult to evaluate whether the cargo shifts due to vibration during transportation. StableCargo [13] is a tool to simulate the transportation of a container, and it focuses on simulating how cargo moves in the container during transport. StableCargo uses metrics to evaluate the dynamic stability of the container. The interpretation of actual transportation was not included in this study.

3. Method

We propose a flexible method for CLPs that combines a GA and PS. Given specific cargo compositions and their constraints, the GA iteratively finds loading patterns for a container. Then, the simulates transportation by shaking the container and its cargo to evaluate the stability of the loading pattern, which becomes the fitness value of the GA. During the shaking process, forces are applied to the cargo, which are affected by vehicle acceleration and suspension and road condition.

Figure 1 shows the basic process flow for the proposed method. First, a population including $N$ individuals with random genes is generated. Then, from $N$ individuals, the GA produces $N$ children by performing crossover and mutation, as described in Sect. 3.1. For all $2N$ individuals, the PS simulates the loading of cargo in a container with the loading pattern specified by their genes, and it shakes the container. Individuals are sorted in descending order of fitness values, and the top $N$ individuals are selected as the population for the next generation.

3.1 Design of Genetic Algorithm

In this paper, genes $g_1 \ldots g_c$ $(c=1\ldots M)$ represent the loading order of $M$ cargo pieces where each gene specifies one of the pieces. For example, given five cargoes (i.e., $M=5$), $g_1(2,4,5,1,3)$ indicates the loading order from cargo piece 2 to piece 3. As shown in Fig. 2, crossover randomly takes two parents $g_i$ and $g_j$ to produce two children $g_{N+i}$ and $g_{N+j}$. Each parent randomly selects a continuous part of the gene, and this sequence is passed down to its child, as illustrated by large open arrowheads. A child receives the remaining genes from the other parent. The gene order is retained, as illustrated by black thin arrows. After crossover, mutation is performing by swapping two genes in each child. In the selection operation, all individuals containing parents and children in a population (i.e., $P = \{g_1,\ldots,g_N, g_{N+1},\ldots,g_{2N}\}$) are sorted in descending order of their fitness values, and the top $N$ individuals are selected for the next generation.

3.2 Physics Simulation

The container shaking simulation is implemented as a multibody dynamics simulation. All cargo pieces and the container are represented as rigidbodies (which never bend). Each piece of cargo is one rectangular rigidbody, and the container is constructed with five rectangular rigidbodies, a floor and four walls. The acceleration scenario is implemented by changing the velocity of the container, which is placed on a vast plane with no friction.

The cargo pieces are placed in the container in the order dictated by their genes. The first piece is placed in the left-front-bottom corner of the container, and then the subsequent pieces are placed in the next space in a top-right-back
order. At first, cargo is stacked up when subsequent pieces have a smaller face than the top face of the previous piece, and cargo is not stacked beyond the top of the container. A piece of cargo is rotated as needed to fit it into the remaining space. When placing a piece of cargo in a space, at first the algorithm tries to give the piece its default orientation. If the piece of cargo cannot be placed in the default orientation, the algorithm rotates it and tries to fit it in with up to five other orientations. When a piece of cargo does not fit into the space with any orientation, the algorithm tries to put it in the next space. When a subsequent cargo piece cannot be stacked, it is placed in the right space so as not to stick out beyond the front axis. When the container is filled and some cargo pieces remain unpacked, these unloaded pieces are omitted from the simulation.

Unity [14], a widely used IDE for video games and AR/VR applications is used to implement the PS. It has a physics engine that is based on PhysX [15].

3.3 Fitness Evaluation

Physics simulation generates much valuable information to evaluate the packing, transformation, rotation, contact and velocity of individual cargo pieces. Various fitness functions can be designed to create and exploit this information without modifying the algorithm flow.

The fitness function $E$ (Eq. (1)) to be minimized is computed in the PS phase of our method.

$$
\text{minimize } E = f_1 + \frac{1}{\#C} \left( \sum_C (f_2 + f_3) \right) + \frac{1}{\#S} \left( \sum_S f_4 \right) + \frac{1}{\#F} \left( \sum_F f_5 \right) \quad (1)
$$

The first term $f_1$ indicates the density of the container (Eq. (2)); the second term, $\#C$ is the number of cargo pieces in the container; $f_2$ and $f_3$ are the translation (Eq. (3)) and rotation (Eq. (4)) of each pieces, respectively; $\#S$ in the third term is the number of pieces with a stack-ability constraint (i.e., “do not stack” cargo), $\#F$ in the fourth term is the number of pieces with a fragility constraint (i.e., “fragile” cargo), and $f_4$ and $f_5$ are binary values indicating whether a piece of cargo’s stack-ability and fragility constraints are satisfied, respectively.

Density : $f_1 = 1 - \frac{\sum_{\text{cargo}} V}{\text{Container} . V}$, \quad (2)

Translation : $f_2 = 1 - \frac{\text{Overlap}}{\text{cargo} . V}$, \quad (3)

Rotation : $f_3 = \frac{\text{rot.y}}{360} + \frac{\text{rot.x}}{90} + \frac{\text{rot.z}}{90}$, \quad (4)

$$
\text{Overlap} = (a-|p_0 . x-p.x|)(b-|p_0 . y-p.y|)(c-|p_0 . z-p.z|), \quad (5)
$$

In Eq. (5), Overlap indicates the translation of a cargo with respect to how much it remains in the initial position, where $a$, $b$, and $c$ are the length, width, and height of a cargo, respectively, and $p_0$ and $p$ are the initial and present position, respectively. As Fig. 3 shows a 2D example, a piece of cargo retains only some of the area in the space that it initially occupied after it moves. The value of the translation is the ratio of this remaining volume (area) to the whole volume of the cargo. This normalizes the value for different sizes of cargo pieces. Note that the translation calculation (Eq. (3)) ignores any rotational cargo movement, which is
independently evaluated in \( f_3 \) (Eq. (4)).

“Density” is the ratio of container space usage. A higher ratio is considered to be better.

“Translation” averages the cargo movement, excluding rotational movement. Each piece’s position is represented as a point corresponding to the center of mass, and “translation” evaluates the transformation of the point. The implemented equation evaluates how much volume remains in the initial space.

“Rotation” evaluates the rotational movement of cargo. In Eq. (4), the angles are in degrees. Rotation around the x- and z-axes means tipping of the cargo, and thus it has a higher value than y-axis rotation. Movement and rotation values are averaged for all pieces of cargo in the container. Cargo pieces outside the container are included neither in the simulation nor the evaluation.

“Do not stack” is a popular constraint in CLP research. A piece of cargo with this property must not be placed under any other cargo. In the PS, all contact between cargo pieces, and between the cargo pieces and container are calculated. All physical contacts for each cargo piece that has a property of “do not stack” are monitored during the simulation. If the contact point is on the top surface, the cargo returns 1. This value is also averaged for all pieces in the container, or the number of pieces having the “do not stack” property.

Fragility is another constraint to protect a delicate cargo from any rough handling during transportation. To evaluate the damage to a cargo during transportation, the damage boundary curve was introduced [13]. The curve divides the damage region by considering that damage occurs due to a combination of acceleration and velocity as shown in Fig. 4. The damage boundary curve is determined with tests on an actual object. In this work we simplified the curve to two thresholds for velocity and acceleration, as shown in Fig. 5. The velocity and acceleration of cargo with a fragile property is monitored to determine whether these values are under their respective thresholds. If velocity and acceleration are both over the threshold simultaneously, the cargo returns 1. Then, the scores for cargo pieces with the fragility property in the container are averaged.

4. Experiments

We performed five experiments to verify that the fitness equations described in the previous section properly reflect constraints such as density, movement, rotation, stackability and fragility of cargo in a container. We first considered small-scale problems with less than 10 pieces, which have specific solutions under one of the constraints. We verified that the proposed method successfully finds the proper solutions to the problems. Then, we also considered more complex problems under all constraints and applied the proposed method to them. In the following experiments, we confirmed that there was no unnatural behavior of the cargo due to numerical errors. Thus, we believe that numerical errors did not affect the results in our experiments.

4.1 Experiments on Acceleration Scenario

In the following experiments, we produced shaking by changing the velocity of the container. Equations 6 and 7 show the definitions of velocity. The formulation is inspired by a Lissajous curve (or Bowditch curve) to obtain various shaking directions. Equation 6 defines the velocity of the container along the x-axis, and Eq. (7) defines it along the z-axis.

\[
V_x = A_x \sin(F_x \Delta t),
\]

\[
V_z = A_z \sin(F_z \Delta t),
\]

where \( V \) is the velocity of the container in each time step; \( A \) amplifies the velocity of the container, leading to larger vibration; and \( F \) defines the shaking pattern – we set \( F_x = 2 \) and \( F_z = 3 \). Velocity was not applied along the height (y) direction.

4.2 Experiment on Density

To evaluate how the density constraint affects the loading arrangement, we simulated eight cargo pieces, four tall and four short. As shown in Fig. 6, the height of tall pieces is the same as the container’s height, and the width and depth of each pieces are half of those of the container. Short pieces cannot be stacked in the container because their height is larger than half of the container height. Figure 7 shows an example of possible arrangements whose fitness values against the density constraint are the worst (left) and best (right).

Therefore, the expected solution is to load four tall
pieces to maximize the container space usage. The experiment was run with 30 individuals for 30 generations. Figures 8 and 9 show the traces of the fitness evaluation (Eq. (1)) and density term (Eq. (2)), respectively. The graphs show the change in values for 30 generations (29 iterations). In the last generation, we obtained the expected answer for all 30 individuals. This confirms that the density fitness affects the solution.

Note that the solution did not converge. Swapping pieces of the same size did not affect the evaluation even though their genes were different. This implies that there are $4! \times 4! = 576$ genes variations. The optimal solution is to load the four tall pieces first and the four short pieces next, although some pieces do not fit (cannot be loaded) due to the limited space.

4.3 Experiment on Movement

To evaluate how the movement constraint affects the loading arrangement, we simulated four $5 \times 6 \times 5$ pieces of cargo and one thin $5 \times 4 \times 5$ piece. Figure 10 shows the possible arrangement to minimize the movement of cargo. In this case, putting one thin piece under another piece results in better fitness (right) rather than putting the thin piece on another piece (left). The experiment ran with 30 individuals for 10 generations. Figures 11 and 12 show the traces of fitness evaluation (Eq. (1)) and the movement term (Eq. (3)), respectively. In the last generation, all individuals have the expected solution. This shows that the movement term
4.4 Experiment on Rotation

To evaluate how the rotation constraint affects the loading arrangement, we used one slender cargo piece, three thin \(5 \times 6 \times 5\) cargo pieces, and one tall \(5 \times 7 \times 5\) cargo piece. The slender piece can be stacked on other pieces while standing upright, but vibrations can cause the cargo to fall. Thus, in this problem, it is assumed that the slender piece is placed sideways on tall pieces (Fig. 13). The width and depth of the tall and thin cargo pieces are half of the corresponding container dimensions, so four are arranged without gaps. An experiment with 30 individuals ran for 30 generations. Figures 14 and 15 show the traces of the fitness evaluation (Eq. (1)) and rotation term (Eq. (4)), respectively. In the last generation, all individuals show the arrangement that places the slender piece sideways on tall pieces.

(Eq. (3)) affects the solution.

4.5 Experiment on Stackability

To confirm the stacking term, we considered a problem for the same \(5 \times 5 \times 5\) size of eight cargo pieces and a \(10 \times 10 \times 10\) container. Four pieces have a stacking constraint, and thus should be placed in the upper layer. As shown in Fig. 16, the expected solution fills the container with all the cargo. An experiment with 30 individuals ran for 100 generations. Figures 17 and 18 show the traces of fitness evaluations (Eq. (1)) and the average stacking values \(f_4\) in Eq. (1)) for individuals, respectively. In the last generation, all individuals places the four cargo pieces with the stacking property on the other four pieces. Thus, it is confirmed that the stacking term affects the result.

As seen in the figures, the solution converges. However, this is accidental because some pieces are physically equivalent, and thus swapping them did not affect the fitness evaluation. Many apparent different chromosomes were equivalent from the point of view of physics, and thus
4.6 Experiment on Fragility Constraint

To confirm the effect of the fragility constraint, we considered a problem for the same $5 \times 5 \times 5$ size of eight cargo pieces and a $14 \times 10 \times 14$ container. The container size is large enough to place the entire cargo in a $2 \times 2 \times 2$ arrangement, with some space remaining. Four pieces had the fragility property. In this case, the fragile pieces should be placed on the bottom instead of the other cargo (Fig. 19). Figures 20 and 21 show the traces of fitness evaluations (Eq. (1)) and the average fragility values ($f_5$ in Eq. (1)) for individuals, respectively. An experiment with 30 individuals ran for 100 generations. In the last generation, 27 individuals placed the all four fragile pieces in the bottom layer, and 18 are converged to one answer. The fragility constraint forces fragile cargo to be placed in the bottom layer to avoid falling from a height. The fragility term was confirmed to affect the results.

4.7 Experiment for Multiple Objectives (Complexity)

We considered a more complex problem with various sizes of cargo pieces as shown in Fig. 22. The container size was $10 \times 10 \times 20$, and 10 cargo pieces had various sizes. One piece had the “do not stack” property (green), and another had the “fragile” property (red). We performed three trials of this experiment with 30 individuals for 30 generations. Figure 23 shows that the fitness value gradually converges to around 1; the proposed method successfully seeks an optimal arrangement in complex problems as well.

4.8 Processing Time

The processing time was almost proportional to the numbers of cargo pieces, generations, and populations. Table 1 shows the processing time for the problem instance with ten pieces of cargo. We ran 9 experiments with different numbers of generations and populations. Each experiment was run with 10, 30 and 100 individuals and 10, 30, and 100 generations. A population of 10 individuals over 10 generations needed 13 sec to obtain the result. Thirty individuals over 10 generation needed 40 sec, almost threefold the time needed for 10 individuals and 10 generations.

We also measured the processing time for each part of
Table 1  Relationships between processing time (sec.) and number of generations and populations.

| Generations | Population |
|-------------|------------|
| 10          | 13         | 40         | 126        |
| 30          | 36         | 111        | 395        |
| 100         | 115        | 358        | 1288       |

Fig. 24  Relationship between the number of cargo pieces and the processing time for three processes: GO, PS, and selection. Of the total operation time shown, GO took less than 10 sec and selection took less than 1 sec.

The process. As shown in Fig. 1 the process comprises three parts: GO, PS, and selection. We ran four experiments with different cargo sizes (21, 31, 41 and 68 pieces) and measured the processing time. The results are shown in Fig. 24. The processing time for the PS part was 99.65% of the overall processing time for 21 pieces, 99.35% for 31 pieces, 99.49% for 41 pieces, 99.25% for 68 pieces. The average time for each generation of the PS part was 8 sec for 21 pieces, 10 sec for 31 and 41 pieces, and 13 sec for 68 pieces.

4.9 Limitation of the Cargo Size

The limitation of the proposed method is that it cannot cope with large problem sizes, namely, a large number of cargo pieces. When the number of pieces is large, the PS processing time increases, and the time required to obtain a plausible solution becomes long. In addition, due to the disadvantage of the decoding algorithm, it is difficult to pack the cargo tightly when the container is larger than the volume of the cargo. The current decoding algorithm is based on bottom-left-back method, in which a cargo piece can only be placed next to a piece already in the container, that has a top or side surface larger than that of the new piece. For this reason, it is difficult to arrange many packages side by side in the bottom (top) direction and left (right) direction. Therefore, it is difficult to obtain a plausible solution in a container that is large with respect to the cargo. Thus, we conducted additional experiments to investigate the limitation of the problem size. We verified the number of cargo pieces placed in a container against the different problem sizes listed below and evaluated each 10 times.

We prepared four problems. The size of each cargo piece was 4, 5, or 6 for height, width, and depth. The container was sized to allow several pieces to be lined up, and the dimensions were given in multiples of 5 per side. The number of cargo pieces was set to more than 85% of the maximum number of pieces that could fit in the container (21 for 2×3×4). The four container sizes were 10×15×20, 15×15×20, 15×20×20, and 20×25×20 and the numbers of cargo pieces were 21, 31, 41, and 68, respectively, for these sizes.

First, the number of GA generations was determined by preliminary experiments. The graphs in Figs. 25 and 26 show the transition of the fitness value over 5 trials for 2 problem sizes with 41 and 68 cargo pieces, respectively. The horizontal axis represents the generations from 1 to 100, and the vertical axis is the fitness value for the individual with the highest fitness at each generation. We found that the fitness value reached around 0.5 at about generation 30. Thus, we terminated processing at generation 50.

The fitness value was verified for different problem sizes. Ten trials were conducted for each problem size, and the transition of the highest evaluation value in each generation was obtained. The graph in Fig. 27 shows the fitness values at the 2nd, 10th, 20th, 30th, 40th, and 50th (final) generation from the left. Each box plot represents 10 trials. The graph shows that the tendencies between the problem sizes of 31 (3×3×4) and 41 (3×4×4). In contrast, between 41 (3×4×4) and 68 (4×5×4), there is less difference than between 31 and 41. From this, it is estimated that if the number of cargo pieces arranged in the left (right) direction exceeds four, the cargo cannot be arranged well with the current algorithm.

Figure 28 shows examples of the solution for each problem size. Each image shows the best solution for each
Fig. 27   Traces of fitness values over 10 trials with different problem sizes. Box plots show the quartiles at generations 2, 10, 20, 30, 40, and 50 (final). Different colors indicate different problem (pieces of cargo) sizes: 21 (red), 31 (yellow), 41 (green), and 68 (blue).

Fig. 28   Example solutions for four different cargo sizes, at the 5th, 25th, and 50th generations.

generation and each problem size. It can be seen that as the number of pieces increases, the container becomes more empty. This is considered to be a problem with the current algorithm, especially the decoding algorithm from gene to cargo arrangement.

4.10 Crossover Variation

Two types of crossover methods, named $X_1$ and $X_2$, were compared. As described in Sect. 3, $X_1$ preserves a random portion of one parent and sorts the rest in the order in which they appear in the genes of the other parent. $X_2$ preserves gene segments of random length from the beginning of the chromosome.

Figures 26 and 29 show the traces of fitness values for five trials over generations using crossovers $X_1$ and $X_2$, respectively. The problem size is 68 cargo pieces. We verified that both $X_1$ and $X_2$ improve the fitness values. In addition, Fig. 30 shows the quartiles at specific generations. Red plots indicate the results for $X_1$, and green plots indicate $X_2$. While both the fitness values for $X_1$ and $X_2$ converge to about 0.5, it seems that $X_2$’s fitness value improves more quickly and has a smaller variance.

5. Discussion and Limitations

This work deals with some constraints of transportation as soft constraints that consider the stackability and fragility of cargo. Although our method makes it easy to implement a flexible evaluation, it has some disadvantages that should be discussed. In some cases, not all of the cargo fits into a container and some pieces are left out. When the container capacity is not enough to hold all cargo, the method currently ignores remaining pieces and excludes them from the fitness evaluation. In that case, the latter part of the gene sequence becomes meaningless because it does not affect the fitness value during the GA process. This may mean that the GA preferentially excludes cargo with special handling properties. As explained in Eq. (1), cargo with special handling properties has additional terms in its fitness evaluation and generally ranks lower than other cargo with no special handling properties under the same condition.

As Eq. (1) shows, all terms of the fitness evaluation have the same weight. In this paper, we focused on vali-
dation of the PS. The weight parameters, which have a significant effect in multi-objective optimization, should be examined further.

In this work, we set a fixed loading location policy, such as bottom-left-back. Thus, in some cases, the proposed method is not able to fill spaces in the container with the remaining cargo. We will try to solve this issue by designing other loading location policies.

6. Conclusion

We proposed an optimization method for 3D CLPs with multiple constraints. The method consists of a GA to generate a cargo arrangement and a fitness evaluation using PS. The fitness function considers not only the maximization of the container density and fitness value but also different constraints, such as weight, stability, fragility, and orientation of individual cargo pieces. We employed a container shaking simulation for the fitness evaluation in the PS to include the effect of the constraints during transportation. Through several experiments, we verified that the proposed method successfully provides the optimal cargo arrangement for a small-scale problem with around 10 pieces. As discussed, the proposed method has a limitation when solving larger problems. Investigating various loading location policies remains as future work.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number JP21K12030.

References

[1] C. Truong, L. Amodeo, and F. Yalaoui, “A mathematical model for three-dimensional open dimension packing problem with product stability constraints,” Proc. Int. Conf. Optimization and Learning, vol.1173, pp.241–254, Springer, Cham, 2020.
[2] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn., Springer-Verlag, Berlin Heidelberg New York, 1996.
[3] A. Bortfeldt and G. Wäscher, “Constraints in container loading - a state-of-the-art review,” European J. Operational Research, vol.229, pp.1–20, Aug. 2013.
[4] A. Bortfeldt and G. Wäscher, “Applying tabu search to container loading problems,” Operations Research Proceedings, pp.553–558, 1994.
[5] 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, ArticleID1705290, 2017.
[6] J.H. Holland, “Genetic algorithms,” Scientific American, vol.267, no.1, 1992.
[7] A.L. Olsen, “Penalty functions and the knapsack problem,” Proc. the 1st Int. Conf. Evol. Comput., 1994.
[8] K. Deb, Multi-objective optimization using evolutionary algorithms, Chichester, UK, Wiley, 2001.
[9] G.R. Raidl and G. Kodydek, “Genetic algorithms for the multiple container packing problem,” Parallel Problem Solving from Nature, Lect. Notes Comput. Sci., vol.1498, Springer, pp.875–884, 1998.
[10] A.H. Dornas, F.V.C. Martins, F.M.S. João, and E.F. Wanner, “Real -