Designing a Flexible Evaluation of Container Loading Using Physics Simulation

Shuhei Nishiyama[1], Chonho Lee[2], and Tomohiro Mashita[2]

[1] Graduate School of Information Science and Technology, Osaka University, Japan
nisiyama.syuhei@lab.ime.cmc.osaka-u.ac.jp
[2] Cybermedia Center, Osaka University, Japan
leech@cmc.osaka-u.ac.jp,
mashita@ime.cmc.osaka-u.ac.jp

Abstract. In this work, an optimization method for 3D container loading problem with multiple constraints is proposed. The method consists of a genetic algorithm to generate an arrangement of cargoes and a fitness evaluation using physics simulation. The fitness function considers not only the maximization of container density or value but also a few different constraints such as stability and fragility of the cargoes during transportation. We employed a container shaking simulation to include the effect of the constraints to the fitness evaluation. We verified that the proposed method successfully provides the optimal cargo arrangement to the small-scale problem with 10 cargoes.

Keywords: Container loading · Genetic algorithm · Physics simulation.

1 Introduction

Container loading problem (CLP) is to find an arrangement of cargoes in a container, which comes up in various scene in daily life and business including a suitcase packing for traveling, bagging purchased items in a supermarket, container packing for logistics, and so on. Those packing problems are practically solved by the sense of a person working on the packing. However, to obtain an acceptable solutions quickly, the person working on the packing need some level of experiences or training.

To obtain a reasonable loading pattern in a practical packing scenario by a computation has significant advantages, which includes training and evaluation of a person working on a packing task, optimizing a robot packing in a automated logistic system, and so on. In case of a solver for a practical problem setting, basically it should be a custom one for that practical problem because a packing problem in a practical scenario has some constraints of weight, fragility, orientation, stability of loading pattern etc. Moreover, a particular problem has its own importance of the constraints. Bortfeldt and Wäscher [2] reviewed many works of CLP with various type of constraints. Basically, existing works have focused on each specific application scenario. Nevertheless these works meet the demands of practical situations, it makes those solutions only available for the specific part

of the problem. If anyone wish to adapt them to another type of problem, the whole algorithm must be renewed. On the other hand, meta-heuristics methods are generally more flexible about an adaptation of a method to different problem settings.

In this paper we propose a method to obtain a reasonable loading pattern which can consider various constraints and its importance. Our method consists of a physics simulation of loading pattern and meta heuristic optimization. In the physics simulation, we make a loading pattern in the simulator and shake the container to simulate a motion caused by transport. Then our method evaluates the damage of each container. The optimization algorithm considers not only the static evaluation including density, weight but also the constraints of damage during transportation.

Contribution

This paper proposes a flexible method for CLP combining GA and physics simulation. We design the method to separate design of GA, arrangement of cargoes and fitness evaluation. The method will be adapted various situations and constraints modifying only the fitness function. The function can be designed with much information from physics simulation. Physics simulation takes an acceleration scenario, adding velocity to the container and cargoes. The cargoes result its trail, contacts etc. and calculate its value of fitness w.r.t. container loading.

2 Related works

2.1 Genetic Algorithm

Container loading problem (CLP) is known as a NP-hard optimization problem, which have been approached with meta-heuristics algorithms [3, 4]. This paper focuses on genetic algorithm (GA) [5], which is inspired by a process of natural selection to solve optimization problems. GA repeatedly modifies a population of candidate solutions called individuals to get better solutions. Each individual has genes representing a solution, which are encoded in many ways such as bit-string [6], real value [7] and permutation [8]. At each step, GA iteratively applies genetic operations (e.g., crossover, mutation and selection) to one or some individuals (called parents) and produces new individuals (called children). Children inherits some part of 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 CLP context, genes represent how the cargoes are loaded in a specific manner. For example, real-polarized genetic algorithm [9] encodes the loading order of the cargoes as its genes. Wu [10] used two segments of encoding in GA, including the number and the rotation of the cargoes. In sequence-triple [11], genes represent the cargoes positions with three arrays of cargoes order. Relative positions of each gene represent relative spatial position. Similar to [8, 9], we

encodes the loading order into genes. The loading location is straightforward in a bottom-left-back manner. Different from the other work, we run GA with physics simulation to compute fitness under several realistic constrains (e.g., rotation, stability, fragility) in a practical scenario.

2.2 Container Loading with Soft Constraints

In the review by Bortfeldt and Wäscher [2], they mentioned as follows.

> 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 works handle constraints as hard constraints, and only a few types of constraints such as weight constraints, allocation constraints and positioning constraints are addressed as soft constraints. Our work tries to represent more types of constraints as soft constraint and handle them simultaneously.

2.3 Physics Simulation

Physics simulation (PS) calculates the laws of physics. Calculating motions of multiple objects (multibody dynamics) is used for CLP. StableCargo [12] is a tool to simulate the transportation of a container, focusing on to simulate how the cargoes move in the container while transportation. It proposed a metrics to evaluate the dynamic stability of the container with the simulation. The interpretation of real transportation is not included in this work.

3 Method

We propose a flexible method for CLP combining genetic algorithm (GA) and physics simulation (PS). Given particular cargoes and their constraints, GA iteratively finds loading pattern in a container. Simulating transportation, PS shakes the container and cargoes to evaluate the stability of loading pattern, which becomes the fitness value of GA. During the shake, forces are applied to the cargoes, which are affected by vehicle acceleration, suspension, road condition, etc.

Fig.1 shows the basic process flow of the proposed method. A population including N individuals with random genes is generated at first. Then, from N individuals, GA produces N children by performing crossover and mutation, as described in Subsection 3.1. For all $2N$ individuals, PS simulates to load cargoes in a container along 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 a population for the next generation.

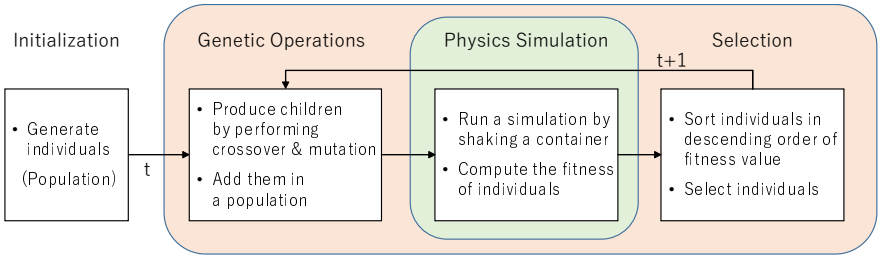

Fig. 1. Flow diagram of the proposed method

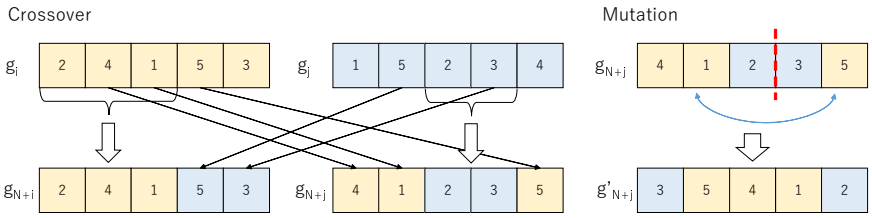

Fig. 2. An example of crossover and mutation operations.

3.1 Design of Genetic Algorithm

In this paper, genes g=g_c (c=1...M) represent the loading order of M cargoes where each gene specifies one of the cargoes. For example, given five cargoes (i.e., M=5), g=(2, 4, 1, 5, 3) indicates the loading order from cargo2 to cargo3. 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 genes, and the part is kept to its child, illustrated by a tick arrow, respectively. Remaining genes are given by another parent whose order is kept, illustrated by thin arrows. After crossover, mutation operation is operated by swapping two parts of the genes in each child. By 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 cargoes and the container is represented as rigidbodies (which never bend). Each cargo is one rectangular rigidbody, and the container is constructed with five rectangular rigidbodies, one bottom and four walls. The accelaration scenario is implemented by changing velocity of the container. The container is put on a vast plane with no friction.

The cargoes are put in the container in the order of genes represent. The first cargo is put in left-front-bottom corner of the container, then following cargoes

are put into the next space in top-right-back order. At first, cargoes are stacked up while the following cargo have smaller face than the top face of previous cargo and the cargo do not beyond the top of the container. When the following cargo cannot be stacked, it is put on the right space while not to stick out to front-axis. When the container is filled and some cargoes are remained to not packed yet, these unloaded cargoes are not included in the simulation.

Unity [13] is used to implement the physics simulation. It is a widely used IDE for video game, AR/VR application and so on. It has a physics engine which is based on PhysX [14].

3.3 Fitness Evaluation

Physics simulation generates much valuable information to evaluate the packing, transform, rotation, contact and velocity, etc. Various fitness functions can be designed such information without modifying algorithm flow. Branches or conditions also can be introduced.

E is the fitness function (Eq. 1) used in selection of GA phase.

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

$\#C$ is the number of cargoes in the container. $\#S$ is the number of cargoes with stacking constraints, and $\#F$ is the one with fragility constraints. f_1 indicates the density of the container (Eq. 2). f_2 and f_3 are the translation (Eq. 3) and rotation (Eq. 4) of each cargo, respectively. f_4 and f_5 are binary values indicating that the constraint is met or not. Each cargo has the value 1 if the constraint is not satisfied, otherwise 0. f_4 is for the stacking constraint, and f_5 is for the fragility constraint.

$$f_1(\text{Density}) = 1 - \frac{\sum cargoes.V}{Container.V} \quad (2)$$

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

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

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

Overlap (Eq. 5) indices the translation of a cargo by how much it remains in the initial position. As Fig.3 shows a 2d example, a cargo remains only some volume (area) in the space that it initially there, after it moved. The value of translation is the ratio of this remaining volume (area) par whole volume of the cargo. This normalizes the value with cargoes sizes. Note that the translation calculation (Eq. 3) ignores any rotational move of cargo, which is evaluated in f_3.

"Density" is the ratio of container space usage. Higher ratio is considered to be better. "Move" averages the cargoes move, excluding rotational move. Each

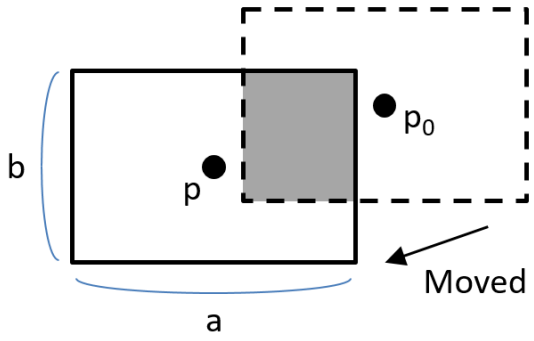

Fig. 3. Overlap shown in 2D. The rectangles represent a cargo before/after it moves, the gray area is the overlap.

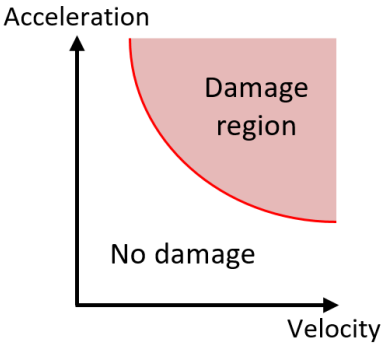

Fig. 4. Typical Damage Boundary Curve.

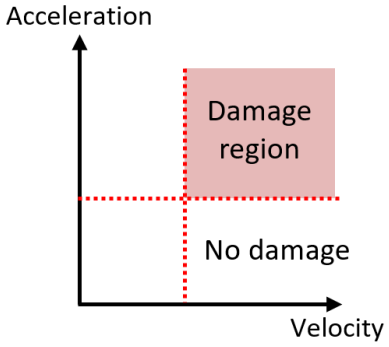

Fig. 5. Implemented damage boundaries. Two thresholds on velocity and acceleration mark the damage region.

cargoes' position is represented as a point of center of mass, "Move" evaluates the transform of the point. The implemented equation evaluates how much volume remains in initial space.

"Rotation" evaluates the rotational move of a cargo. In Eq. 4, the angles are in degree. The rotation around x- and z-axis means tipping of the cargo, thus it has higher value than y-axis rotation. "Move" and "rotation" value take an average of all cargoes in the container. Cargoes that not in the container, NOT included in the simulation, are also NOT included the evaluation.

"Do not stack" is one of popular constraints in CLP research. A cargo with this property mustn't put under any other cargoes. In PS, all contacts between cargoes, and between cargo and container are calculated. Each cargo which has a property of "Do not stack" watches its all contacts during the simulation. If the contact point is on the top surface, the cargo returns 1. This value also take

an average of cargoes, note that the number of cargoes which have the "Do not stack" property and in the container.

"Fragility" is another constraint to prevent a delicate cargo from any rough handling during transportation. To evaluate the damage of a cargo during transportation, Damage Boundary Curve (DBC) was introduced [12]. The curve divides damage region considering that the damage occurs on a combination of acceleration and velocity as Fig. 4. Damage Boundary Curve (DBC) is determined with tests on the real object. In this work we simplified the curve to two thresholds on velocity and acceleration as 5. The cargo with this "Fragile" property watches its velocity and acceleration are under the thresholds. If velocity and acceleration are both over the threshold simultaneously, the cargo return 1. Then take an average of cargoes which have the property and in the container.

4 Experiments

In this section, we perform five experiments to verify that fitness equations shown in the previous section properly reflect constraints such as density, move, rotation, stack and fragility of cargoes in a container. We first consider small-scale problems with less than 10 cargoes, which have particular solutions under one of the constraints. we verify that the proposed method successfully finds the proper solutions to the problems. Then, we also consider more complex problem under all constrains and apply the proposed method to it.

4.1 Experiment on Density

To evaluate "Density" constraint, we simulate eight cargoes, four are tall and the others are short. As shown in Fig. 6, tall cargoes height are the same of the container's height, and all cargoes width and depth are 1/2 of container. Short cargoes cannot be stacked in the container because the height are larger than 1/2 of container. Fig. 7 shows an example of possible arrangement whose "Density" fitness is the worse (left) and the best (right).

Therefore, the expected answer is to load four Tall cargoes to maximize the container space usage. The experiment are run with 30 individuals for 30 generations. Fig. 8 and 9 show the trace of fitness evaluation (Eq. 1) and "Density" term (Eq. 2), respectively. The graphs show the change of values while 30 generations (29 iterations). At the last generation, we obtained the expected answer in all 30 individuals. This confirms that the "Density" fitness affect the solution.

Note that the solution is not yet converged. Swapping cargoes with the same size do not affect the evaluation even though their genes are different. This implies that there are $4! \times 4!$=576 variations of genes. The optimal solution is to load four Tall cargoes first and four Short cargoes next although some cargoes cannot be loaded due to the limited space.

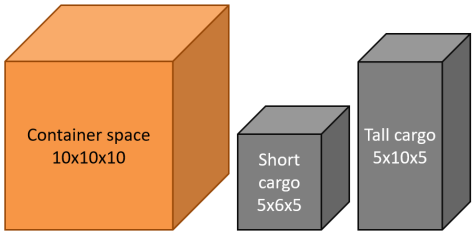

Fig. 6. Container space and cargoes sizes which are used to experiment on "Density".

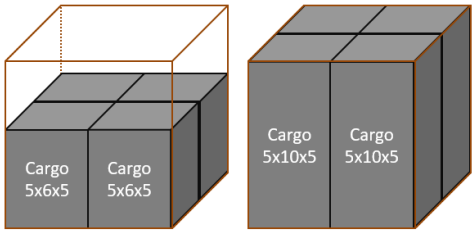

Fig. 7. The "Density" term has better value with the right arrangement than the left arrangement.

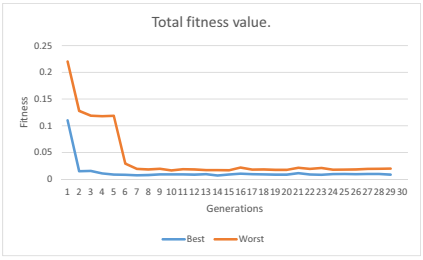

Fig. 8. Fitness value of the best and worst individual in each generation.

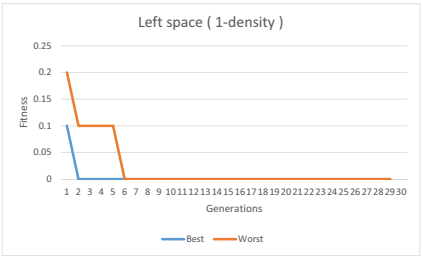

Fig. 9. "Density" value of the best individual in each generation.

4.2 Experiment on Move

To evaluate "Move" constraint, we simulate four $5 \times 6 \times 5$ cargoes and one thin $5 \times 4 \times 5$ cargo. Fig. 10 shows the possible arrangement to minimize the move of cargoes. In this case, putting one thin cargo under another cargo results in better fitness (right) rather than putting the thin cargo on the another cargoes (left). Experiment ran with 30 individuals for 10 generations. Fig. 11 and 12 show the trace of fitness evaluation (Eq. 1) and "Move" term (Eq. 3), respectively. At the last generation, all individuals have the expected solution. This shows that "Move" term (Eq. 3) affects the solution.

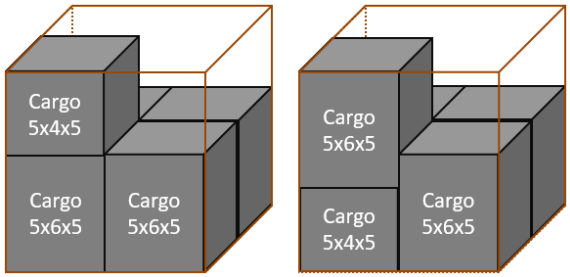

Fig. 10. The "Move" term has better value with the right arrangement than the left one. In the left arrangement, the short cargo on the top can move around and makes the value worse.

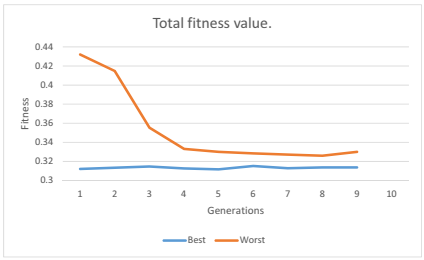

Fig. 11. Fitness value of the best and worst individual in each generation.

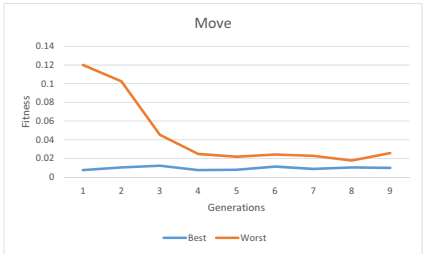

Fig. 12. "Move" term of the best and worst individual in each generation.

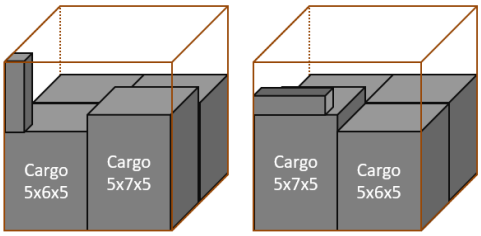

Fig. 13. The right arrangement has better value of "Rotation" term.

4.3 Experiment on Rotation

To evaluate "Rotation" constraint, one slender cargo, three thin $5 \times 6 \times 5$ cargoes and one tall $5 \times 7 \times 5$ cargo. Slender cargo can be stacked on other cargoes with standing upright, but a vibration by shake will cause the cargo to fall. Thus, in this problem, it is assumed that a slender cargo is placed on tall cargo sideways (Fig. 13). The width and depth of the tall and thin cargoes are half of each container, so four are arranged without gaps. An experiment with 30 individuals

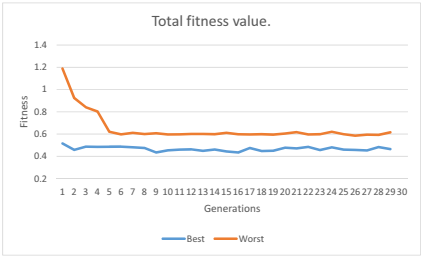

Fig. 14. Fitness value of the best and worst individual in each generation.

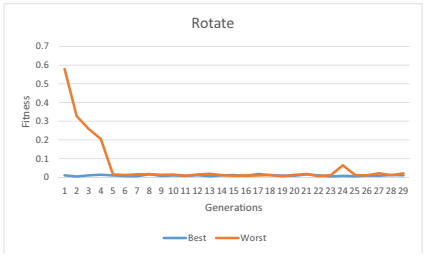

Fig. 15. "Rotation" term of the best and worst individual in each generation.

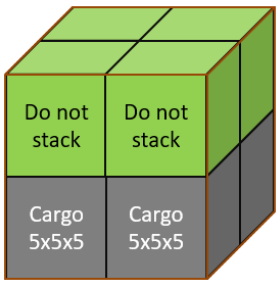

Fig. 16. An expected answer for stacking problem.

runs for 30 generations. Fig. 14 and 15 show the trace of fitness evaluation (Eq. 1) and "Rotation" term (Eq. 4), respectively. At the last generation, all individuals show the arrangement with putting slender cargo on the tall cargo sideways.

4.4 Experiment on Stacking Constraint

To confirm "Stacking" term, we consider a problem for the same $5 \times 5 \times 5$ size of 8 cargoes and a 10x10x10 container. 4 cargoes have the stacking property, thus should be put on the upper layer. As shown in Fig. 16, the expected solution fills the container by all cargoes. An experiment with 30 individuals runs for 100 generations. Fig. 17 and 18 show the trace of fitness evaluation (Eq. 1) and the average "Stacking" value (f_4 in Eq. 1) of individuals, respectively. At the last generation, all individuals put the 4 cargoes with the property on the other 4 cargoes. It is confirmed that the "Stacking" term affect the result.

In this experiment the solution is converged. It seems to be accidentally happen because some cargoes are physically equivalent, thus swapping them never affect the fitness evaluation. Many apparent different chromosomes are equivalent in point of view of physics, thus fitness value won't differ. (In fact

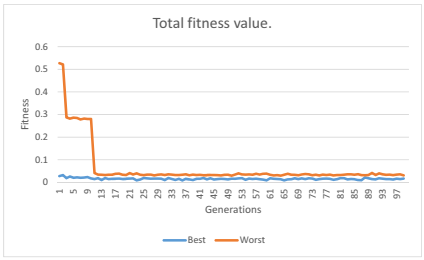

Fig. 17. Fitness value of the best and worst individual in each generation.

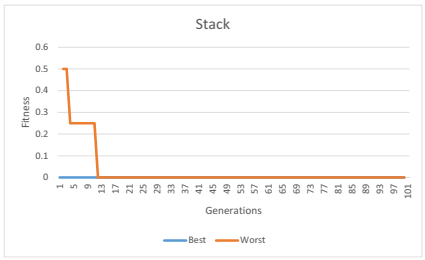

Fig. 18. "Stacking" property evaluations of the best and the worst individual in each generation.

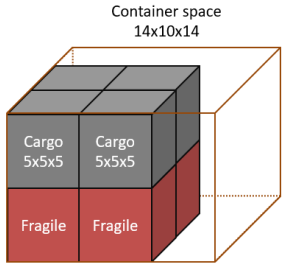

Fig. 19. An expected solution that fragile cargoes were put at bottom layer.

other experiences don't converge to one answer, we discuss the result adding up the equivalent individuals.)

4.5 Experiment on Fragility Constraint

To confirm the effect of "Fragility" constraint, we consider a problem for the same $5 \times 5 \times 5$ size of 8 cargoes and a $14 \times 10 \times 14$ container. The container size is enough to put all cargoes by $2 \times 2 \times 2$ arrangement, remaining some spaces. 4 cargoes have the "Fragility" property. In this case, those fragile cargoes should be put on the bottom rather than other cargoes (Fig. 19). Fig. 20 and 21 show the trace of fitness evaluation (Eq. 1) and the average "Fragility" value (f_5 in Eq. 1) of individuals, respectively. An experiment with 30 individuals runs for 100 generations. At the last generation, 27 individuals put the all 4 fragile cargoes on the bottom layer. 18 are converged to one answer. "Fragility" constraint makes cargoes to be put bottom layer to avoid falling down those cargoes. It is confirmed that "Fragility" term affects the result.

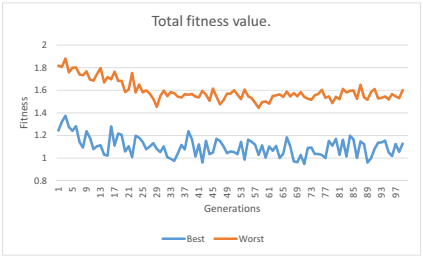

Fig. 20. Fitness value of the best and worst individual in each generation.

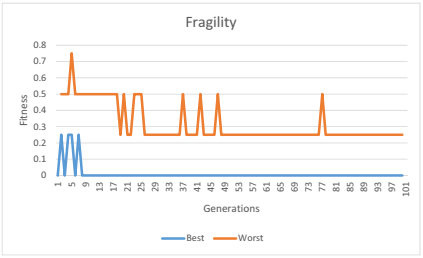

Fig. 21. "Fragility" evaluations of the best and the worst individual in each generation.

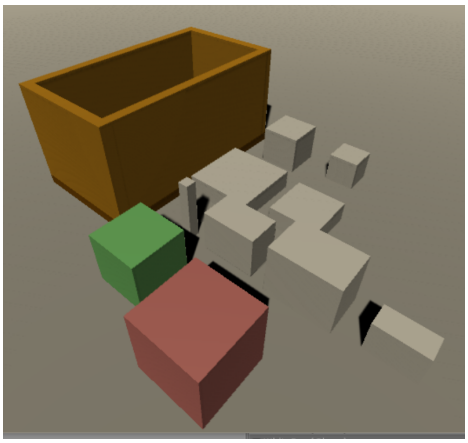

Fig. 22. An example cargoes and container space for a complex experiment.

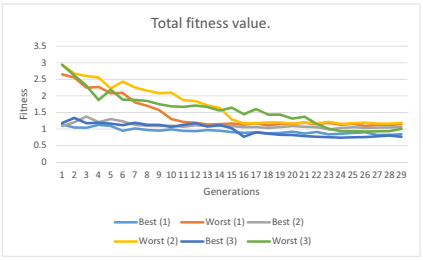

Fig. 23. Fitness value of the best and worst individuals in each generation over three trials.

4.6 Experiment for Multiple Objectives

We consider more complex problem with various cargoes as shown in Fig. 22. The container size is $10 \times 10 \times 20$, and 10 cargoes with various size. One cargo has "Do not stack" property (green), and other one has "Fragile" property (red). We perform this experiment with 30 individuals for 30 generations three times. As shown in Fig. 23, fitness value converges to around 1.

4.7 Processing time

The processing time is almost proportional to the number of cargoes, generations and populations. Table 1 shows the processing time of 10 cargoes problem instance. We ran 9 experiments with each different number of generations and populations. Each experiment is run with 10, 30 and 100 individuals and 10,

30, 100 generations. Population 10 (one generation has 10 individuals) with 10 generation needs 21 seconds to obtain the result. 30 individuals with 10 generation needs 57 seconds, almost three times of the time of 10 individuals and 10 generations.

5 Discussion and Limitations

This work handles some constraints of transportation as soft constraints, which includes stacking constraints or fragility constraints. Although it makes easy to implement flexible evaluation, there are some disadvantages which should be discussed. Our method currently ignores remaining cargoes when the capacity of a container were not enough to contain all cargoes. This makes latter part of genes meaningless because it cannot affect the fitness evaluation. However, this affect the GA performance both the speed of convergence and extensive search.

In the problem instances for experiments, there are a possibility that all cargoes do not fit in the container and some cargoes are left. In that case our system exclude the remained cargoes from the fitness evaluation. This causes a convergence that prefers to exclude propertied cargoes. As explained Eq. 1, propertied cargoes have additional terms in its fitness evaluation and mostly be worse than other non-propertied cargoes in the same condition.

6 Conclusion

In this work, an optimization method for 3D container loading problem with multiple constraints is proposed. The method consists of a genetic algorithm to generate an arrangement of cargoes and a fitness evaluation using physics simulation. The fitness function considers not only the maximization of container density or value but also a few different constraints such as stability and fragility of the cargoes during transportation. We employed a container shaking simulation to include the effect of the constrains to the fitness evaluation. We verified that the proposed method successfully provides the optimal cargo arrangement to the small-scale problem with 10 cargoes. In future, we will investigate the large-size problem and tackle a case when a container cannot contain all cargoes.

Table 1. Relationships between processing time (sec.) and number of generations and populations.

Generations	Populations		
	10	30	100
10	21	57	201
30	76	212	807
100	229	799	2859

References

1. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
2. Bortfeldt, A., Wäscher, G., "Constraints in container loading - a state-of-the-art review", *European Journal of Operational Research,* vol. 229, 2013.
3. Bortfeldt, A., Wäscher, G., "Applying tabu search to container loading problems," *in Operations Research Proceedings*, pp.553-558, 1994.
4. G. Cabrera-Guerrero, C. Lagos, C. Casta˜neda, F.Johnson, F.Paredes, and E. Cabrera, "Parameter tuning for local-search-based matheuristic methods," *Complexity*, vol.2017, ArticleID1702506, 2017.
5. J. H. Holland, "Genetic Algorithms," *Scientific American*, vol. 267, no. 1, 1992.
6. Olsen A. L. "Penalty Functions and the Knapsack Problem," In *Proc. of the 1st International Conference on Evolutionary Computation*, 1994.
7. Deb, K. "Multi-objective optimization using evolutionary algorithms", Chichester, UK, Wiley, 2001.
8. G.R. Raidl, G. Kodydek, "Genetic algorithms for the multiple container packing problem," *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, vol. 1498, Springer, pp. 875-884, 1998
9. Dornas, A. H., Martins, F. V. C., João, F. M. S. and Wanner, E. F., "Real - Polarized genetic algorithm for the three - dimensional bin packing problem", In *Proc. of GECCO '17, Berlin, Germany*, 2017.
10. Y. Wu, W. Li, M. Goh, and R. de Souza, "Three-dimensional binpacking problem with variable bin height," *European Journal of Operational Research*, vol.202, no.2, 2010.
11. Yamazaki, H., Sakanushi, K., Nakatake, S. and Kajitani, Y., "The 3d-packing by meta data structure and packing heuristics", *trans. fundamentals,* vol. E83-A, no. 4, April, 2000.
12. Ramos, A. G., Jacob, J., Justo, J. F., Oliveira, J. F., Rodrigues, R. and Gomes, A. M., "Cargo dynamic stability in the container loading problem - a physics simulation tool approach", *Int. J. Simulation and Process Modelling,* vol. 12, no. 1, pp. 29-41. 2017.
13. Unity Real-Time Development Platform, https://unity.com/
14. GameWorks PhysX Overview, https://developer.nvidia.com/gameworks-physx-overview