A new framework for Solving the 3D Bin Packing problem using PSOGA

Group 16

Shokrzad, Reza s1056369
Koprcina, Josip s1062758
Kulkarni, Pratik s1051211

June 2021

1 Introduction

A common occurrence in real-world industries is item storage. Items need to be stored for a wide range of purposes such as in a warehouses for easier keeping, or in trucks for easier transportation. This, seemingly easy problem is called the 3D bin packing problem and is considered an NP-Hard problem that hasn't yet been solved. To be more precise, the problem is packing a set amount of 3D items into as little space as possible. This way, in real-world situations, the cost would be reduced as less space would be required to store or transport the items. There are many extra constrains that arise when implementing this, such as the weight of the items (for example a truck can only carry a certain amount of weight), whether or not different items can be kept in the same space or if the space is split into smaller sub-spaces for easier organization or division into categories.

We have chosen to tackle this problem with a mixture of a Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Our goal is not only to make a model that solves the problem properly but is also simple to understand and use in practice. The dataset has been provided as excel files in which the amount and measurements of the items and the trucks in which they will be placed are given. We then present these as tuples and try to order them so that the least amount of space is taken. GA is the main algorithm that will be solving this. This has already been done in literature and has proven solid results. We removed one of it's steps, mutation, to minimize the random effect of our model. Instead, we will add PSO as one of the steps in the loop of GA to increase the accuracy and speed. Therefore our research question will be weather or not GA together with PSO can solve the 3D bin packing problem well enough to be used in real-world scenarios. The approach has been tested in python and the code can be found on github.

2 Related Work

So far this problem has been studied extensively., but being an NP-Hard problem, no optimal solution has been found yet. The usual approach is to try and solve the problem in simplified versions first, not taking into account certain restrictions. Researchers have started by solving

the 2D bin packing problem, which is the same, but with one less dimension. [1] used a heuristic approach to write a lower bounds and a dominance criterion to derive a reduction algorithm. They solve the problem with a very good score, but with a fairly complicated solution. The paper who's psychology of simplicity we'll follow is [2]. They aim to solve the 2D problem with a good score but also a focus on algorithm simplicity. It uses an evolutionary based heuristic approach and reached slightly worse results than the state of the art while being a lot easier to code and understand. They use tricks such as smart mutation, non-hybridized approach and effective size reduction. [3] gives a wide view of a few methods used to solve the 2D problem. It explains some common "tricks" that people came up with during the years such as the "Bottom-Left Rule" (BL), where all the items are stacked towards the bottom left corner. We'll be using the same approach but a 3D version, where it's the bottom-lower-left corner.

An example of a solution to the 3D bin packing problem is seen in [4], where a modi cited genetic algorithm (MGA) was used. They reached an average packing density of 94.56%. Their paper showed how the method works better than the basic methods such as GA and PSO that reach a density of around 90%. An approach that looks at the problem with more constraints such as item conflict and fragmentation is given in [5]. They don't use a heuristic approach but rather try to mathematically make a method that always converges to the global optimum. Methods we read about commonly are either heuristic, that use an approach that can't guarantee the best possible solution, but the computation is much quicker, and the non-heuristic ones that use complex math to always reach the optimum at the cost of algorithm speed and increase in complexity.

3 Data

As previously stated, the data has been given to us in multiple excel files. Each file contains one example we will run. We extracted the information we needed from them. We took the *products* which we used as our items and the *trucks* which we used as our space to fill the items with. For those we took the measurements, width, height, length and their id. The original measures for width, height and length were given in *mm*, but that proved too much for the resources we had access to (we kept getting memory errors because the dimensions multiplying would lead to numbers too large for our RAM). Because of that we changed the measure to *cm* which made the model run smoothly. These values will be used in the first basic testing of the method with minimum constraints and will be a POC of sorts. This is then used to generate the starting population of the algorithm.

4 Genetic Algorithm

Genetic algorithms are search based algorithms inspired by genetics. Just as our genes can be represented by a list of ordered characters, so are solutions we seek when running the method. In our case, that would mean representing our solution with a list of ordered numbers, always in groups of three. The first number in each group would be the "Id" which we use to identify which item is the one in question, the second number would show the location of the left most corner of the item on the y axis, and the third would show the same mentioned corner on the x axis. This way all the items will be placed in the mentioned space. The z axis will be

ignored and the items will simply be placed on each other. Genetic algorithms all have some same basic principles. They start by initializing a "population". This would be a number of possible solutions randomly initialized on certain constraints we set up before-head. This population then goes through certain steps called "generations". Each consecutive generation should be closer to our optimal goal than the last one was. This change that happens is a result of "crossover" and "mutation". The change from one generation to the next starts with calculating the "fitness function". This is a function that will calculate how close to the solution all members of the population are. The crossover step will take the items in the population and mix them to create new ones. Those that had a smaller loss will be more likely to be selected. After the new generation is created from mixing the old one, we apply mutation. This is a random change in parts of an item of the population. It's used to keep some diversity and randomness and usually has a low probability of happening. We then remove unnecessary items in the population by only keeping the best n items (n being a variable we choose). These best n items will be our new generation. We measure the fitness function to see did we reach an optimum that we are pleased with. If we are, the algorithm stops and shows us the result we want, if not, the loop starts over.

5 Particle Swarm Optimization

PSO on the other hand, is inspired by natural behaviour of certain animals, such as birds or fish. When in large numbers, the individual decisions are influenced by what they themselves feel and by the movement of the whole group. We say the movement of the individual is a type of search for the "local optimum" as it looks for the best possible solution in it's immediate range, while following the group would be looking for the "global optimum" because the group covers a large area. The algorithm is great when searching for approximate solutions. We initialize a population of a certain size and place them randomly on the space. We then calculate the fitness function of each member and the population as a whole. We decide on the global optimum or the, so-far, best found solution, and the local optimum for each member, or in what direction is the best solution for each member. After the calculation we iterate through each member of the population and make him move a certain amount in a certain direction. Both of these are dependant on the loss value of the individual, the loss of the group, the velocity (speed at which we wish to converge, a variable we set ahead of time for each member of the population). We then calculate the loses again. If we're satisfied with the values we end the cycles, otherwise we loop over the population again. This method has the bad side of the possibility of converging to a local optimum instead of a global optimum, but makes up for it with it's simplicity to implement and overall performance.

6 Approach and Method

To address the 3D BPP, we introduce a novel heuristic approach that is a hybrid method of PSO and GA. We focused on the advantages of these two evolutionary algorithms to find a better solution in optimizing fitness. In terms of making a model for this problem, we needed a specific representation for each solution. Before introducing our modeling approach, we need to indicate that in the literature of evolutionary methods, the term *solution* is also called

individual, genome, or chromosome in GA and particle in PSO. As the PSO part of this work is more important than the other parts in improving the solutions, we call them as particles in the rest of the report.

We inspired our representation from Ntanjana thesis [6] in which she tried to simplify the solutions by considering the order of entering, the unique number of each product, and one of the product's dimensions in modeling a 2D problem. Each particle has four properties. Three of them are presented in a 3-tuple like (i, y, x) and the last one is the order in which it appears. As it is also illustrated in figure 1 , i stands for the entering order of product, y and x refer to the width and length of each product, and the order of tuples shows the number of each product.

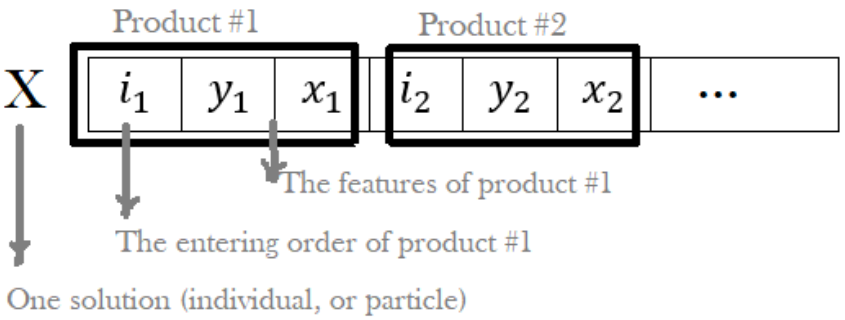

Figure 1: The representation of the particles

To clarify the representation, we would consider two first entered products as (20, 230, 112), (5, 90, 170). In the first tuple, 20 means item number one (#1) is entering as the twentieth particle. This box has y=230 and x=112. Followingly, item number two (#2) will be entering as the fifth item. It is necessary to mention, each particle is an array with a length that equals the number of items multiplied by three (len(particle) = 3 $\times$ the number of items). Thus, for each item, there will be one tuple.

Regarding the height feature of each product, Ntanjana in her thesis assumed the products in falling from a random point (x, y of the point is important in this assumption) and landing to a stable point that cannot fall anymore. This level is the floor of the truck or the top of another product. We also consider z as an updated point through the entering process. In this respect, we have had an innovation that helps the speed of updating process. We considered the boxes enter from the bottom since counting down from the top of the truck would be more RAM-consuming than counting from the bottom. In this approach, the model is obliged to count much less than Ntanjana's proposed direction.

Additionally, according to our heuristic model, four components are involved in solving the 3D BPP. These parts are listed below:

- The initialization part.
- The fitness function.
- The Genetic algorithm operations (reproduction and cross-over).
- Particle Swarm Intelligence (PSO) algorithm.

6.1 The initialization part

One of our most significant innovations is considering the truck as a three-dimensional matrix (figure 2-a). To depict this assumption, we presumed the truck is a sum of voxels like a grid. These voxels are $1 \times 1 \times 1 cm^3$ and zero-initialized as an empty place. Once a voxel gets occupied by an item, it's value changes to one. Figure 2-b shows the illustration of grid representation in 2D form to understand the changing of voxels value better. While a product is moving on the truck (to go back and left or right), the value of busy voxels would mask to 1, and the other values remain 0 (figure 2-c).

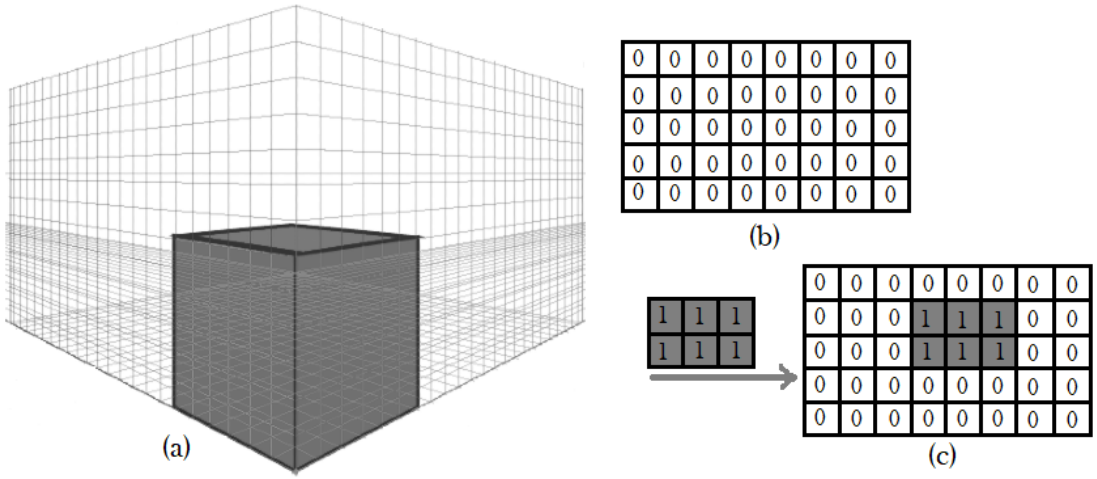

Figure 2: (a) The representation of truck as a grid (b) the empty truck is represented by zero values in 2d form (c) the way of changing value of each voxel including an external box in 2d view

The strategy of boxes arrangement is pushing them to the truck's corners of the back and bottom. In this regard, Ntanjana's thesis proposed a well-known approach in the BPP domain called back-left-bottom. Interestingly, our other novelty is pushing boxes to the back and bottom but both left and right, every other box (figure 3-(a)). In this way, we received a better fitness from 111 in the just left strategy to 97. in our new attitude. In our perception, when we are going to accumulate the boxes in the same direction, the probability of having an empty area in the layout would increase. In this part, we tried to have a better solution in y direction.

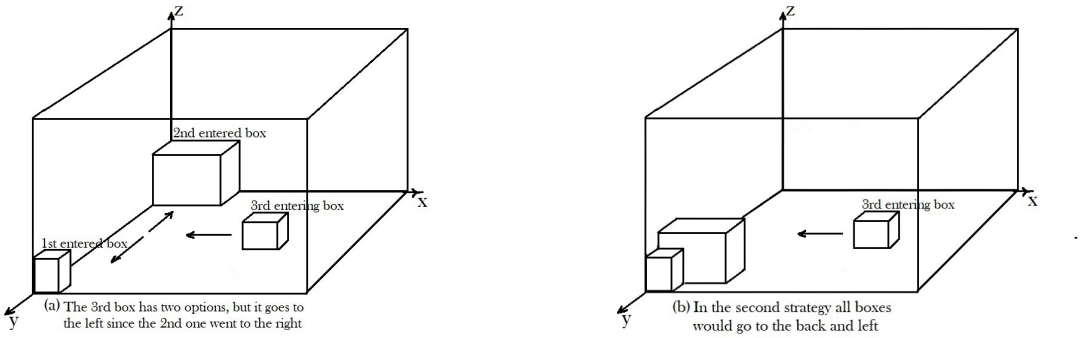

Figure 3: The strategies of boxes arrangement (a) left and right back-bottom every other product strategy (b) left-back-bottom strategy

To initialize the model, a feasible particle is required as the input. To make a feasible particle, it is necessary to contemplate probable constraints. In this matter, One of the most significant restriction was the accumulated boxes' dimensions should be less than the dimensions of the truck. So, the entering product's features are margins from truck's boundaries that has been defined to fulfill this restriction.

Furthermore, since we prefer to set the products from the back of the truck in real-world situations, we generated random solutions based on a Gaussian distribution with a higher chance in the back and lower odds in front of the truck. This technique facilitated the calculation reduction (consequently, the speed increasing) and having a better layout because it places a high importance on placing products on top of each other to make sure there's empty spaces in the back of the truck. Figure 4 displays the idea of the way boxes are distributed. In this part, we tried to have a better solution in z direction.

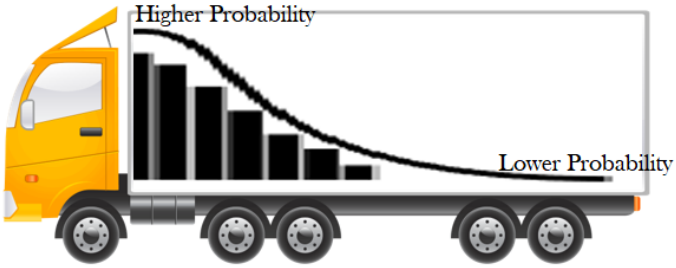

Figure 4: Probability distribution of the particle entering point

6.2 The fitness function

To define a simple effective fitness function, we assumed that if we can organize all products in the back of the truck as much as possible, we would have more free space in the truck to add other products. To respond to this objective, we simulated a model in which the length of occupied space should be minimum. This means the fitness function would minimize the length (x) of the truck, at the point where the last product is set. To have a better understanding, imagine a big thin plate is coming from the back of the truck towards the end of it. (Figure 5).

The denser the layout, the more empty space there is. Therefore, the fitness needs to reduce the length of the layout in each step. In this part, we tried to have a better solution in x direction.

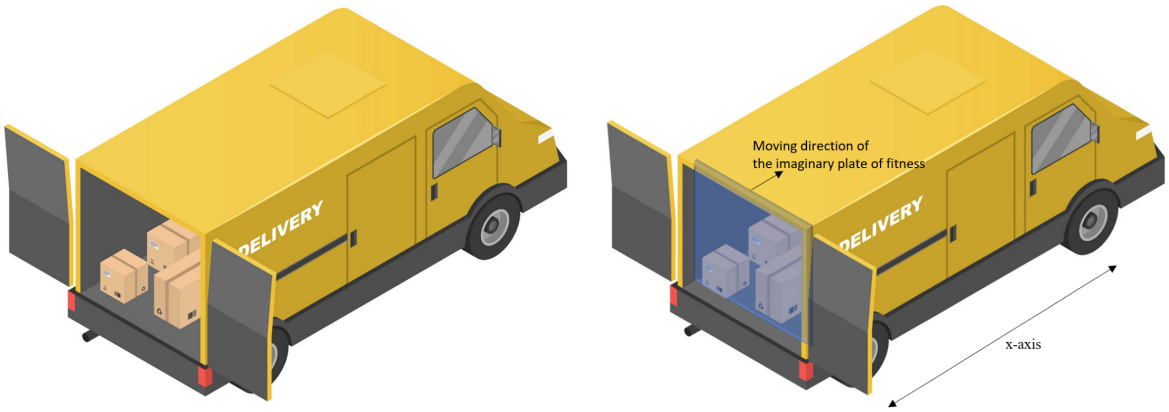

Figure 5: An imagination of how minimizing fitness function (in direction of x-axis) is trying to maximize free space in a truck. The less value of x, the better fitness.

Since the fitness is a minimization function, we also considered two substantial penalties for some cases through the search process. Apart from the considered penalties for initial particles, we need to control the updating solutions after each evolutionary adjustment. Exceeding from truck size is one case of the mentioned difficulties. One of the most restrictive occurrences is generating negative or zero dimensions during the process. As we were updating the solutions based on random coefficient in the PSO process, their value could be negative.

6.3 The Genetic algorithm operations

To enrich our optimal solution, we applied the advantages of both GA and PSO. Similarly, stated in section 4, GA has three phases, reproduction, cross-over, and mutation, for chromosome improvements. Since the mutation process is random and we were not going to change our solutions without any purpose, we ignored this part of GA. Interestingly, we applied cross-over to update the solutions. We implemented a one-fold cross-over for this problem. In this regard, we set a margin from both sides of the arrays to make the operation meaningful. Due to the nature of our representation, the cross-over performers without any obstacles, as is shown in figure 6.

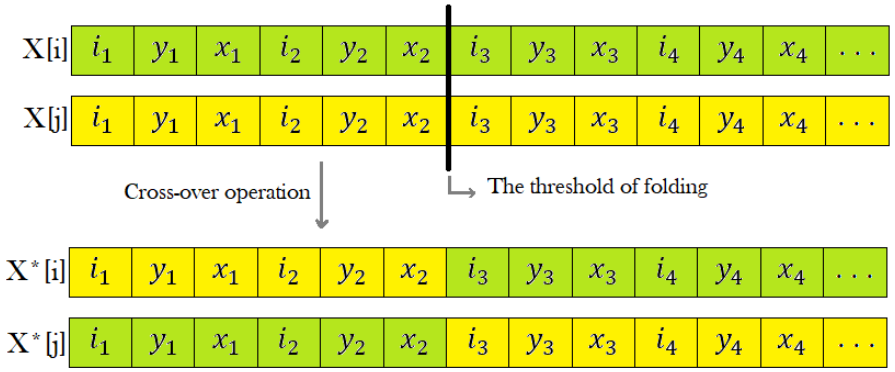

Figure 6: One-fold cross-over strategy

Moreover, regarding the reproduction operation, we provided a 5-time chance for individuals to improve themselves. If it cannot improve itself, we would reproduce it by calling the initial generation function.

6.4 PSO operations

In this project, we also applied the PSO algorithm to revise the state of the particles in the task of finding better particles. In this regard, we specified three vectors to renew the situation of a particle. These mentioned parts are the velocity, the personal best, and the global best. Besides, it is necessary to state that GA, in this work, influences the order of product entering, and PSO affects the entering position (x, y). Figure 7 illustrates the effect of operations.

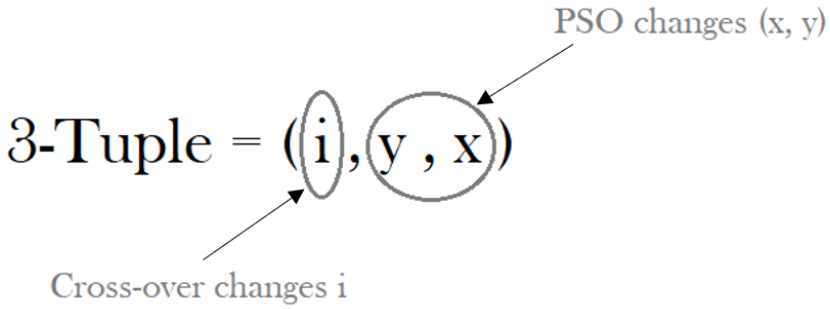

Figure 7: the way GA and PSO influence on particles

7 Results and Analysis

As the task is searching among several feasible layouts, using heuristic algorithms would be time-consuming. We ran the code in many different cases, such as with and without all PSO, cross-over, reproduction, and two organizing strategies. Every run took almost 7 hours. Besides, Table 1 displays a list of parameters and their descriptions followed by the final value after tuning.

Parameter name	Description	Tuned value
Maximum Iteration	#Search iteration	1000
Particle Numbers	#Individuals that are comparing in each iteration	50
c_1	Coefficient of personal best in PSO	0.1
c_2	Coefficient of global best in PSO	0.2
w	The inertia weight of velocity in PSO	0.05
Penalty	The fitness function returns it in infeasible cases	$100 \times \#products$

Table 1: Description and final values of parameters

Based on our experimental setup, the penalty is $100 \times \#products$ because by increasing the number of products, the fitness value would enhance. So, if we estimate a random static figure for the punishment, it is likely that the fitness value for more items exceeds the penalty. After tuning parameters of PSO, we realized the importance of global-best is more than personal-best. In the domain of this problem, inertia weight is less important than two others coefficients.

Figure 8-(a) explains the best-fitness changes during the search process in 1000 iteration. As we can see, the diagram falls in the initial search steps that show fitness makes a considerable improvement at the beginning. Just before iteration 400, there is a plateau with a value near the best fitness. The tiny fluctuation in mean-fitness (see Figure 8-(b)), is related to the *reproduction* process. Because generating particles is random in the process of reproduction, some solutions would be worse than the previous one. Hence, there would be little peaks in the specific iteration with more number of worse solutions.

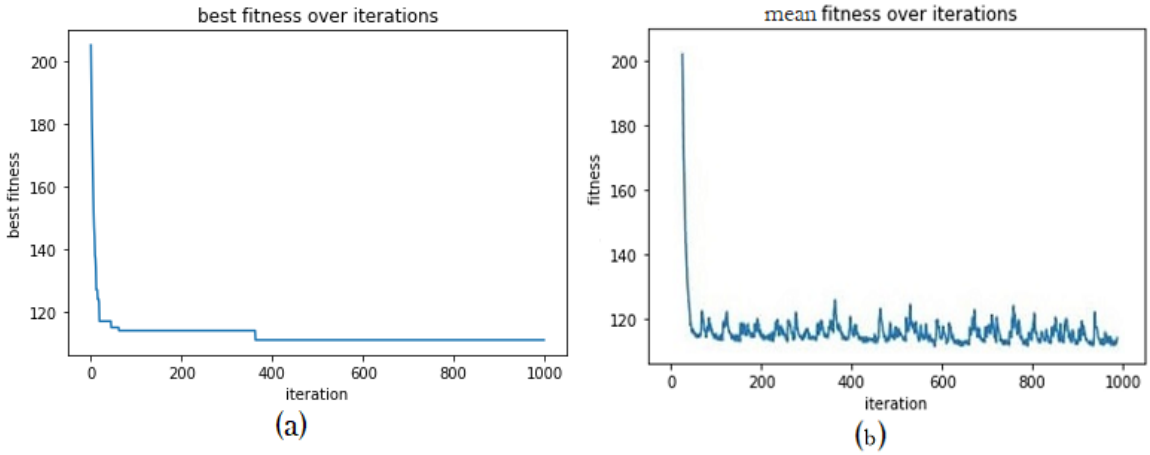

Figure 8: (a) Best and (b) average fitness alters through 1000 iteration

We tested the model in 4 distinctive forms. First, with layout strategy 1 (every other product would go to the left and right), second, with layout strategy 2 (all products would go to the left), third, without PSO, finally, without cross-over in GA. the results of the best and average fitness listed in figure 9, respectively.

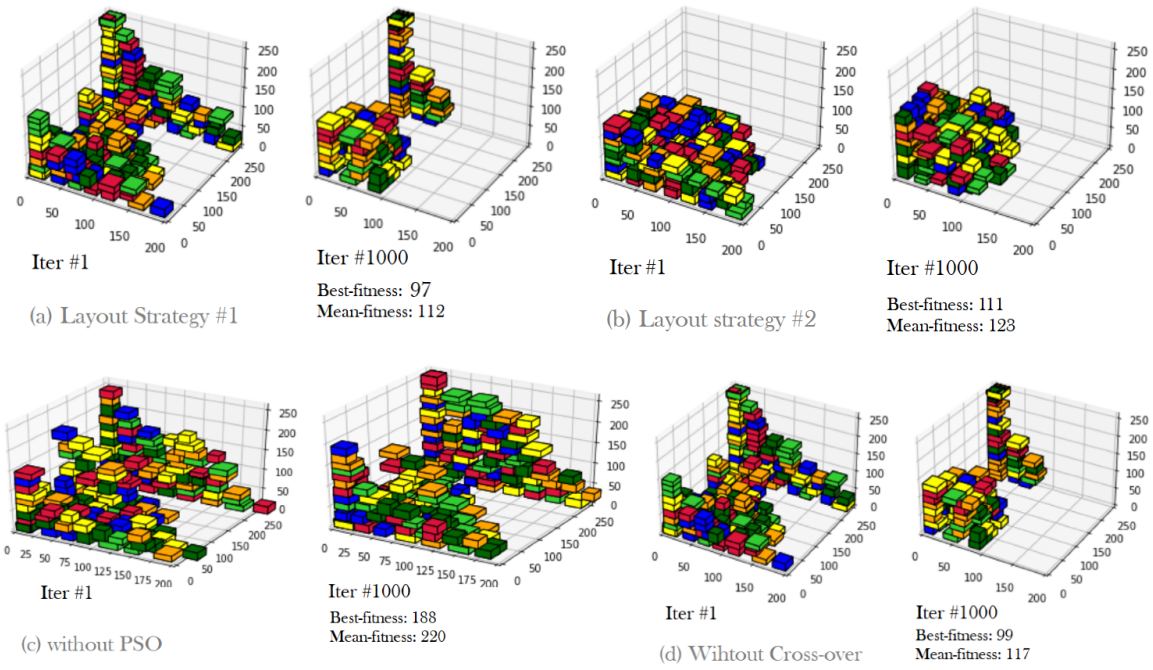

Figure 9: Different scenarios to understand the importance of each item.

According to the results, both diagrams and the fitness values, it is provable strategy one is better than two in the layout of the truck. That was one of our innovations in this work. In the case without PSO, it is obvious even from the layout that the model was not successful in improving the solution. That means PSO has had a crucial role in this model. On the contrary, the negligible change in fitness without cross-over shows the weak role of cross-over in comparison with PSO. However, in presence of cross-over the fitness got better. Also, the reproduction part of GA is another reason of its importance.

8 Discussion and Outlook

There is a long list of constraints that can be a part of this problem. 12 of them are indicated in Ntanjana's work [6], section 4.1.1. We considered some of them in this work. Among these restrictions, one of them is called *load stability*. We attempted to extend that in this way that if a box wants to stand on top of another one, the common area should be 70 percent. Because of time restrictions, we could not address this constraint in the end. Adding this constraint would make the work closer to real-world situations. There is a lot of room for improvement to make the model more accurate. Another bright idea for future works would be designing a model to manage non-rectangular-shaped products such as circular ones.Furthermore, the fitness function could get improved by considering some other criteria as user specification features. In some cases, the order of products in the layout is critical due to their fragility or bulkiness. Moreover, for the products which can be rotated, changing the landing face can provide more free space. So, if we add an item in our tuple representation for the degree of spin, this constraint could have been satisfied. Finally, applying other complementary heuristic methods to make our model more sophisticated would be beneficial to find better optimal solutions.

9 Conclusion

3D Bin Packing Problem (BPP) is a real-world optimizing problem. Due to the nature of the problem, heuristic algorithms would be beneficial to find an optimal solution. We applied a hybrid method of PSO and GA to address this problem. Results revealed this hybrid method works much better than applying PSO or GA alone. This work also reveals two strategies of truck layout and how we can represent products to feed the fitness function.

10 Author Contributions

- Reza Shokrzad: paper reading and topic research (came up with the idea in the project), programming (the fitness and main loop), report writing (sections 6 to 9), literature review (the whole of thesis, and some papers), slide (creating the presentation slide).
- Josip Koprčina: paper reading and topic research, very little programming (data preprocessing), report writing (section 1 to 5), presentation.
- Pratik Kulkarni: paper reading and helped in programming and visualisation of the data.

References

[1] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete Applied Mathematics*, vol. 28, no. 1, pp. 59–70, 1990.

[2] A. Stawowy, "Evolutionary based heuristic for bin packing problem," *Science Direct*, vol. 55, January 2008.

[3] E. Hopper, "Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods," May 2000.

[4] V. Mokshin, D. Maryashina, N. Stadnik, A. Zolotukhin, and L. Sharin, "Modified genetic algorithm as a new approach for solving the problem of 3d packaging," *CEUR-WS*, vol. 2667, pp. 91–97, May 2020.

[5] A. Ekici, "Bin packing problem with conflicts and item fragmentation," *Computers Operations Research*, vol. 126, p. 105113, 2021.

[6] A. Ntanjana, "Two and three – dimensional bin packing problems : An efficient implementation of evolutionary algorithms," 2018.