Jurnal Teknik Industri, Vol. 20, No. 2, December 2018,
ISSN 1411-2485 print / ISSN 2087-7439 online

DOI: 10.9744/jti.20.2.89-94

A Hybrid Greedy Algorithm and Simulated Annealing for Single Container Loading Problem: A Case Study

I Gede Agus Widyadana [1]*, Audrey Tedja Widjaja[2], Kun Jen Wang [2]

Abstract: A single container loading problem is a problem to effectively load boxes in a three-dimensional container. There are many researchers in this problem try to find the best solution to solve the problem with feasible computation time and to develop some models to solve real case problem. Heuristics are the most method used to solve this problem since the problem is an NP-hard. In this paper, we introduce a hybrid greedy algorithm and simulate annealing algorithm to solve a real container loading problem in one flexible packaging company in Indonesia. Validation is used to show that the method can be applied practically. We use seven real cases to check the validity and performance of the model. The proposed method outperformed the solution developed by the company in all seven cases with feasible computational time.

Keywords: Single container loading problem, greedy algorithm, simulated annealing.

Introduction

Container loading problems sometimes are called the packing problem, have been explored by many researchers since it plays important roles in logistics. There are some container loading problems have been explored to deal with problems in practice since there are many constraints should be considered such as constraint related with a container, item, cargo, positioning, and load (Bortfeldt and Wascher [1]). One type of container loading problem is a single container loading problem. The single container routing problem is a packing problem where a set of boxes are arranged to be put in a three-dimensional container with objectives to maximize space utilization. Araya and Riff [2] used beam search strategy to solve a single container outing problem and claimed their method outperform some preceding methods such as Zhu and Lim [3], Zhu *et al.* [4], Goncalves and Resende [6] and Fanslau and Bortfeldt [6].

There are some variations of a single container loading problem. Lim *et al.* [7] developed a heuristic model to solve a single container loading problem with axle weight constraints that are applied in the California Vehicle Code (CVC). Wang *et al.* [8] developed a single container loading problem by considering shipment priority that is common in a real situation. The research focus on a single container loading problem is not only about problem variations but also methods to solve the problem. There are some researchers try to find the best solution with feasible computation time. Huang and He [9] used a heuristic caving degree approach to solve a single container loading problem. Zhu and Lim [3] solved a single container loading problem by modifying a greedy algorithm. Huang *et al.* [9] proposed an effective heuristic to solve a single container loading problem. A heuristic method to solve a single container loading problem was developed by Araya *et al.* [2] and they called the method as VCS. Most research used a heuristic approach to solve a single container loading problem and no one used a metaheuristic method. However, some metaheuristic methods are used to solve container loading problem such as Tabu Search (Liu *et al.* [10]). In this paper, we try to develop a hybrid heuristic and metaheuristic method to solve a single container loading problem and apply the method to one flexible packaging company in Indonesia. The hybrid method is applied to get efficient computation time and effective result. This paper is presented in four sections. The first section present background of the paper, the second section show model and solution development, section 3 shows the application of the solution to a real case on a company and the last section give the conclusion of this research.

[1]Faculty of Industrial Technology, Department of Industrial Engineering, Petra Christian University, Jl. Siwalankerto 121-131 Surabaya 60238, Indonesia.
[2] School of Management, Industrial Management Department, National Taiwan University of Science and Technology, 43, Sec.4, Keelung Road, Taipei 106, Taiwan, ROC
Email: gede@petra.ac.id, kjwang@mail.ntust.edu.tw

* Corresponding author

Methods

Mathematical Model

In this model, we use similar notation as Chen *et al.* [11] and Huang *et al.* [9] as below:

n	:	Total boxes to be loaded
N	:	A set, $N = \{1,2,\ldots,n\}$
M	:	max $\{\bar{x}, \bar{y}, \bar{z}\}$
(p_i, q_i, r_i)	:	Length, width and height

		of box i, respectively
(x_i, y_i, z_i)	:	coordinates of the left-front-bottom corner of box i
(l_{xi}, l_{yi}, l_{zi})	:	binary variables showing whether the length of box i is parallel to the x-axis, y-axis or z-axis.
(w_{xi}, w_{yi}, w_{zi})	:	binary variables showing whether the width of box i is parallel to the x-axis, y-axis or z-axis.
(h_{xi}, h_{yi}, h_{zi})	:	binary variables showing whether the height of box i is parallel to the x-axis, y-axis or z-axis.
$(\alpha_{ij}, \beta_{ij}, \delta_{ij})$	:	binary variables showing the relative positions of box i and box j, such as: $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,0,1)$ if box i *is* on the left-hand side of box j; $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,1,0)$ or $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,0,0)$ if box i is behind box j; $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (0,1,1)$ if box i *is* in front of box j; $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,0,1)$ if box i *is* below of box j; $(\alpha_{ij}, \beta_{ij}, \delta_{ij}) = (1,1,0)$ if box i *is* above of box j;

The Mixed Integer Linear Programming (MILP) from Huang *et al.* [9] is:

The fitness function is minimizing container length to pack all the boxes.

$$Min\ x \tag{1}$$

In the first constraints, all boxes can't overlap.

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \le x_j + M(1 + \alpha_{ij} + \beta_{ij} - \delta_{ij}), \forall i,j \in N, i < j \tag{2}$$

$$x_j + p_j l_{xj} + q_j w_{xj} + r_j h_{xj} \le x_i + M(1 + \alpha_{ij} - \beta_{ij} + \delta_{ij}), \forall i,j \in N, i < j \tag{3}$$

$$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \le y_j + M(1 - \alpha_{ij} + \beta_{ij} + \delta_{ij}), \forall i,j \in N, i < j \tag{4}$$

$$y_j + p_j l_{yj} + q_j w_{yj} + r_j h_{yj} \le y_i + M(2 + \alpha_{ij} - \beta_{ij} - \delta_{ij}), \forall i,j \in N, i < j \tag{5}$$

$$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \le z_j + M(2 - \alpha_{ij} + \beta_{ij} - \delta_{ij}), \forall i,j \in N, i < j \tag{6}$$

$$z_j + p_j l_{zj} + q_j w_{zj} + r_j h_{zj} \le z_i + M(2 - \alpha_{ij} - \beta_{ij} + \delta_{ij}), \forall i,j \in N, i < j \tag{7}$$

$$1 \le \alpha_{ij} + \beta_{ij} + \delta_{ij} \le 2, \ \forall i,j \in N, i < j \tag{8}$$

For the second constraint, all boxes can be put in a container.

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \le \bar{x}, \quad \forall i \in N \tag{9}$$

$$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \le \bar{y}, \quad \forall i \in N \tag{10}$$

$$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \le \bar{z}, \quad \forall i \in N \tag{11}$$

In the third constraint, the length, wide and high of box i only parallel with one axis x, y, and z.

$$l_{xi} + l_{yi} + l_{zi} = 1, \quad \forall i \in N \tag{12}$$

$$w_{xi} + w_{yi} + w_{zi} = 1, \quad \forall i \in N \tag{13}$$

$$h_{xi} + h_{yi} + h_{zi} = 1, \quad \forall i \in N \tag{14}$$

$$l_{xi} + w_{xi} + h_{xi} = 1, \quad \forall i \in N \tag{15}$$

$$l_{yi} + w_{yi} + h_{yi} = 1, \quad \forall i \in N \tag{16}$$

$$l_{zi} + w_{zi} + h_{zi} = 1, \quad \forall i \in N \tag{17}$$

where:

$$x_i, y_i, z_i \ge 0 \tag{18}$$

$$0 < x \le \bar{x} \tag{19}$$

$$0 < y \le \bar{y} \tag{20}$$

$$0 < z \le \bar{z} \tag{21}$$

Greedy Algorithm

In the first step, we group pallets with the same size to set the height of stacks are not more than the height of a container. We use a greedy algorithm to solve the first step as follows:

1. Sort boxes from the largest size
2. Choose a box with the largest size, put it in first level, and add one box with the same size and put it above the first box.
3. Check the total height, if the total height is less than the container height than choose one box with the same height and put it on the next level.
4. Continue step 3 until no boxes can be put above other boxes.
5. Find a new box with largest size and continue with step 1.
6. Choose one box with the same size and less height and goes to step two.
7. Continue steps one to six until all boxes have been stacked.

Simulated Annealing

Simulated Annealing (SA) algorithm used in this problem is a simple SA algorithm as shown in Figure 1.

Encoding Method

Encoding method represents the solution to the problem. The solutions are coded into row strings where the first row represents the loading sequence into a container and the second row represents the position of the boxes and represented by a binary number. An example of the strings is shown in Figure 2. Figure 2 shows the first box loaded is box number 2, the second box is box 4 and so on. In the second row shows the rotation of the box where 1 represent the

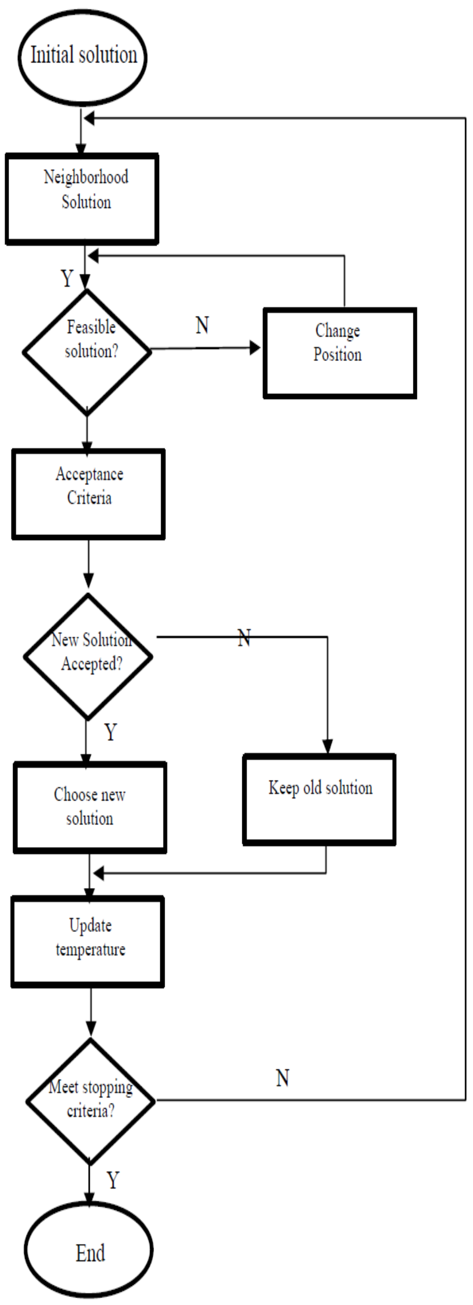


Figure 1 The simulated annealing algorithm

2	4	10	7	1	3	9	6	8	5
0	1	1	1	1	0	0	1	1	0

Figure 2. A sample of solution

length of a box follows x-axis and 1 if the length of the box follows y-axis.

Fitness Function

A fitness function is a criterion that has to be optimized. In this paper, we try to minimize the area of the unoccupied container. The fitness function is equal to the total wide of a container minus the total area of the boxes loaded in the container. There is a possibility that the solution is not feasible since the total width or the total length of loaded boxes are bigger than the container's width and length. When a feasible solution can not be found then we set the fitness function as a big number.

Generation Mechanism of Neighbourhood Solution

Generation mchanism of neighbourhood solution is a mechanism to generate a new solution in each iteration. The generation mechanism in this paper is as follows:

1. Choose one position randomly and insert to any new position randomly
2. Check feasibility of the solution, when the solution is not feasible to change the position from 0 to 1 or from 1 to 0.
3. Try step 2 until a feasible solution is found. When a feasible solution still can not be found, set fitness function with big value.

Acceptance Criteria for the Neighbourhood Solution

The following criterion is used to evaluate whether a neighbourhood solution is accepted as a new solution or not.

$$\Delta = S - S^* \qquad (22)$$

Where S is a new solution generated by neighborhood scheme and S is the old solution before neighborhood scheme is employed. When Δ is negative then the new solution is better than the old one, but when Δ is positive, there is a possibility for the new solution to be accepted with certain probability. The acceptance probability can be represented as Eq. 23.

$$p = e^{(\frac{\Delta}{T_i})} \qquad (23)$$

where:

T_i = temperature at iteration-i

The next step is generating a random number pm where $0 < pm < 1$. When pm is less than p then a new solution is accepted, otherwise the new solution is rejected.

Temperature updating scheme

The temperature updating scheme used in this paper is the commonly geometric updating scheme as shown in Eq. 24.

$$T_{i+1} = k \times T_i \qquad (24)$$

where k is the rate parameter in terms of initial temperature.

Stopping Criteria

The simulated annealing algorithm is stopped when the temperature (T) is less than a specific temperatu-re defines in advance.

Result and Discussion

The model is used to solve a problem at one flexible packaging company in Indonesia. There are seven cases are used to verify and validate the model. The simplest case is case 1 and the most complicated case is case 7, as shown in Table 1 and 2. Table 1 and are the result of the Greedy Algorithm. For example in Table 2 pallet number 28 is stacked above pallet number 27 and both pallet become one group pallet. Pallet number 15 cannot be stacked above pallet 14, since the total height is more than the container height. One pallet can be stacked with other pallet and become one group pallet if they have the same length and width.

The good solution of simulated annealing is determined by right parameters setting which is consist of initial temperature, stopping temperature, *k*, and the number of replication. The program is run under Macro software in OS Windows 8.1 64-bit with *processor* Intel(R)Core(TM) i5-4590 CPU @3.30GHz and RAM 4,00 GB. We use four parameters set as shown in Table 3. The result of four parameters set is shown in Table 4.

Table 4 shows the best solution in case 1 to 6 for the four parameters set are the same and parameters set 4 giving the best solution for case 7. Running time for parameters set 1 and 3 significantly faster than parameters set 2 and 4. Parameters set 1 result in convergence solutions in 6 cases, parameters set 2 in 2 cases and parameters set 3 and 4 in 7 cases. Therefore, we choose parameters set 3 since the parameters set is the best for running time and convergence. Even though the solution quality is less than solution quality or parameters set 4, the difference is not significant. The solution for seven cases for every temperature is shown in Figure 3.

We validate the model result with company's loading method and the unused area is shown in Table 5.

Table 1. Data of case 1

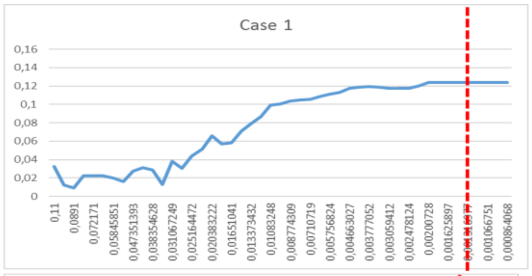
Pallet group	Pallet number	Pallet size (mm)			Box height (mm)
1	9	880	1150	160	1030
1	15	880	1150	160	1070
2	10	880	1150	160	1030
2	16	880	1150	160	1070
3	11	880	1150	160	1030
3	17	880	1150	160	1070
4	12	880	1150	160	1030
4	18	880	1150	160	1070
5	13	880	1150	160	1030
5	19	880	1150	160	1070
6	14	880	1150	160	1070
6	20	880	1150	160	1070
7	21	880	1150	160	1070
7	22	880	1150	160	1070
8	1	780	1150	160	1030
8	3	780	1150	160	1070
9	2	780	1150	160	1030
9	4	780	1150	160	1070
10	5	780	1150	160	1030
10	7	780	1150	160	1070
11	6	780	1150	160	1030
11	8	780	1150	160	1070
12	23	680	1150	160	1070
12	24	680	1150	160	1070

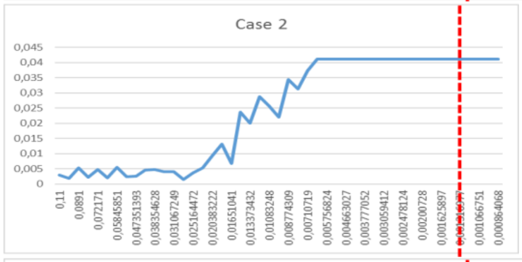
Table 2. Data of case 7

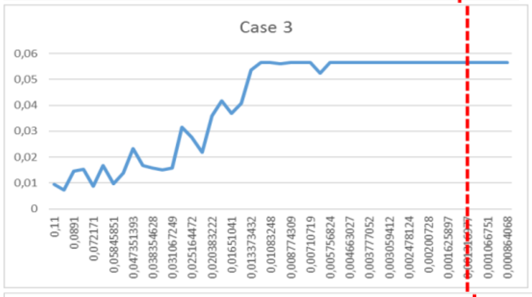
Pallet group	Pallet number	Pallet size (mm)			Box height (mm)
1	14	1620	1120	240	1250
2	15	1620	1120	240	1250
3	27	1260	1320	170	800
3	28	1260	1320	170	800
4	31	1240	1320	170	800
4	32	1240	1320	170	800
5	33	990	1320	170	800
5	34	990	1320	170	800
6	16	1090	1120	170	1180
6	19	1090	1120	170	675
7	17	1090	1120	170	1180
8	18	1090	1020	170	1180
9	1	1020	1020	140	1080
9	2	1020	1020	140	1080
10	3	1020	1020	140	1080
10	4	1020	1020	140	1080
11	5	1020	1020	140	1080
11	6	1020	1020	140	1080
12	7	1020	1020	140	1080
12	10	1020	1020	140	1036
13	8	1020	1020	140	821
13	11	1020	1020	140	1036
14	9	1020	1020	140	821
14	12	1020	1020	140	1036
15	13	1020	1020	140	1036
16	20	1390	690	140	770
16	21	1390	690	140	770
17	22	1390	690	140	770
17	23	1390	690	140	770
18	24	1370	690	140	770
18	25	1370	690	140	770
19	26	1370	690	140	770
20	29	1260	690	170	800
20	30	1260	690	170	800

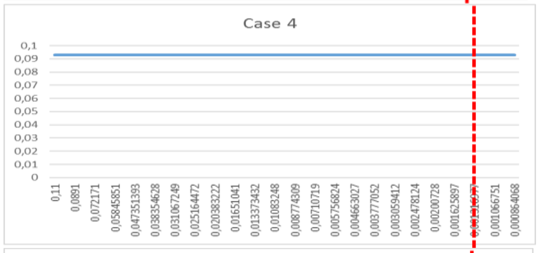
Table 3. Parameters of simulated annealing

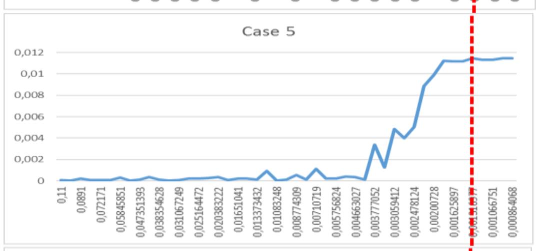
	1	2	3	4
Initial temperature	0.11	0.11	0.11	0.11
Stopping	0.001	0.001	0.0008	0.0008
temperature	3	3	5	5
k	0.9	0.95	0.9	0.95
Number of replication	30	30	30	30

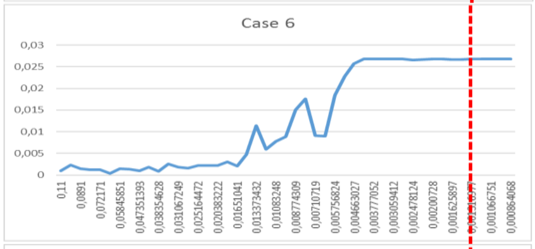












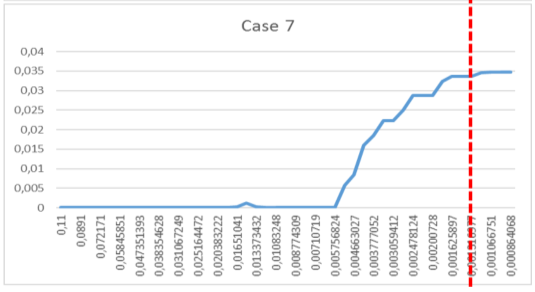


Table 4. Parameters set solution

Parameters set	Quality of solution	Running time	Convergence	Total
1	6	7	6	19
2	6	0	5	11
3	6	7	7	20
4	7	0	7	14

Table 5. Comparison of company's calculation and the research method

Case	Unused area (m^2)	
	Company's method	Our method
1	0,184	1,978
2	0,322	0,552
3	0,3795	0,759
4	1,242	2,553
5	0,253	0,322
6	0,736	0,736
7	0,989	1,196

Table 5 shows that our method has bigger unused area compare with the company's method for all cases. Since in average palette size is 1 m^2, then we can not add more pallet for cases 2, 3, 5, and 6. Using our method, we can add more pallet for cases 1, 4, and 7.

Conclusion

In this research, a hybrid greedy algorithm and a simulated annealing is developed to solve a single container loading problem in one company. Since some parameters are crucial to get efficient and effective solutions, we try four parameter sets and find the best parameters set. The method is validated using seven different cases form the company and the result is compared with the company's solution. The proposed methods outperform company's solution in seven cases with feasible computation time.

The paper can be extended by considering some real constraints that have not been considered in this paper. For example, some buyer asks the weight of the container should be not too much difference between the front, middle and back area.

References

1. Bortfeldt A., and Wascher G. Constraints in Container :oading – A State-of-the-art Review, *European Journal of Operational Research*, 229, 2013, pp. 1-20.
2. Araya I., and Riff M.C., A Beam Search Approach to the Container Loading Problem, *Computers & Operations Research*, 43, 2014, pp. 100-107.
3. Zhu W, Lim A, A New Iterative-doubling Greedy-look Ahead Algorithm for the Single Container

Loading Problem, *European Journal of Operational Research*, 222 (3), 2012, pp.408–417.

4. Zhu W, Hon W, Lim A, and Weng Y. The Six Elements to Block-building Approaches for the Single Container Loading Problem, *Applied Intelligence*, 37(3), 2012, pp. 1–15.
5. Gonçalves J, and Resende M, A Parallel Multi-population Genetic Algorithm for a Constrained Two-dimensional Orthogonal Packing Problem, *Journal of Combinatorial Optimization*, 22(2), 2011, pp. 180–201.
6. Fanslau T, and Bortfeldt A, A Tree Search Algorithm for Solving the Container Loading Problem, *INFORMS Journal on Computing*, 22(2), 2010, pp.22–35.
7. Lim A., Ma H., Qiu C., and Zhu W., The Single Container Loading Problem with Axle Weight Constraints, *International Journal of Production Economics*, 144, 2013, pp. 358-369.
8. Wang N., Lim A., and Zhu W., A Multi-round Partial Beam Search Approach for the Single Container Loading Problem with Shipment Priority, *International Journal of Production Economics*, 145, 2013, pp. 531-540.
9. Huang W., and He K., A Caving Degree Approach for the Single Container Loading, *European Journal of Operational Research*, 196, 2009, pp. 93-101.
10. Liu J., Yue Y., Donng Z., Maple C., and dam Keech M., A Novel Hybrid Tabu Search to Container Loading, 38, 2011, pp. 797-807.
11. Chen, C., Lee, S., and Shen, Q. An Analytical Model for the Container Loading Problem, *European Journal of Operational Research*, 80(1), 1995, pp. 68-76.