Communications of the IIMA

Volume 7 | Issue 4 Article 9

2007

A Hybrid Meta-heuristic for the Container Loading Problem

Shyi-Ching Liang
Department of Information Management Chaoyang University of Technology

Chi-Yu Lee
Department of Information Management Chaoyang University of Technology

Shih-Wei Huang
Department of Information Management Chaoyang University of Technology

Recommended Citation

Liang, Shyi-Ching; Lee, Chi-Yu; and Huang, Shih-Wei (2007) "A Hybrid Meta-heuristic for the Container Loading Problem," *Communications of the IIMA*: Vol. 7 : Iss. 4 , Article 9.
Available at: https://scholarworks.lib.csusb.edu/ciima/vol7/iss4/9

A Hybrid Meta-heuristic for the Container Loading Problem

Shyi-Ching Liang
Chi-Yu Lee
Shih-Wei Huang
Department of Information Management
Chaoyang University of Technology, Taiwan

ABSTRACT

It is very common in an enterprise daily operation to solve Container Loading Problem (CLP). Especially, it is an important issue in the logistic management. The problem aims to determine the arrangement of objects with the best utilization ratio in a container. It belongs to the combinatorial optimization problem. In this paper, a two-phased method focusing on the improvement of the efficiency and on the reducing of the problem size is proposed. In the first phase, a constructive method incorporated with a decision rule borrowing from ant colony optimization is used to construct tower set. The pheromone updating mechanism is useful in choosing proper object while constructing tower using decision rule. In the second phase, an improvement method based on genetic algorithm is used. First, the method sorts the towers by the utilization ratio and then assigns a number to each tower accordingly. The chromosome is a sequence of tower numbers which represents the arrangement of towers in the container's bottom plane. The fitness function is defined as the utilization ratio. A new structure to store the pheromone is proposed which can help the ant in choosing the appropriate object while constructing tower. In this way, the efficiency of the method and the utilization of the container are improved.

INTRODUCTION

Logistics and storage management have already become key factors for modern enterprises to succeed in the market. For enterprises, common problems include transportation routing, human appointment, resource distribution, and etc. Container Loading Problem is one of the problems too. Its application is more extensive, and it can solve the container utilizing of logistics industry or warehousing problem. The main purpose of the problems of this kind is to increase space utilization ratio and reduce the cost.

So, the effective space utilization becomes the main core of the problem. The kind of problem belongs to the combinatorial optimization problem. The problem focuses on how to achieve optimal space utilization in a limited space. The given objects are arranged, according to respective constraints and characteristics to achieve effective space utilization and maximize space utilization ratio.

Container Loading Problem is a kind of scholarly research topic. Its application is quite widespread and it has practical values. It belongs to the combinatorial optimization problem. In operation management, it is an important topic. According to the existing literatures, if considered from the loading space and the object, there are mainly four kinds of fitting situations, namely the single and the multi container fittings, the same and the different volume object fittings. And the research methods include Linear Programming, Dynamic Programming, Heuristics, and Meta-heuristics. Meta-heuristics is the most common method.

RELATED WORKS

Container Loading Problem has already been solved by many scholars by different solutions. Bortfeldt and Gehring (2001) first solved the problem by a two-phased method. In the first phase, they used the greedy method to choose the objects whose volume is comparatively bigger to build the tower *T* gradually. In the second phase, they could only consider the arrangement of the tower *T* on the container's bottom plane directly and used the GA (Genetic Algorithm) to solve. Bortfeldt and Gehring also proposed the new method based on the phased concept. He built the objects into several layers then the GA is used. Moura and Oliveira (2005) also used the phased method by constructing wall-build.

It is known from the literature, when the most of scholars are solving CLP optimization problem, in order to avoid involving too many considerations to complicate problems, they all simplify the problem by the gradual method.

In this paper, we propose a two-phased method focusing on the improvement of the efficiency and the reduction of the problem size. We first use ACO (Ant Colony Optimization) pheromone feedback technique to construct a serial of *TSet* (Tower Set), and the *TSet* in second phase is solved again by GA.

PROBLEM DESCRIPTION

In the aspect of loading space characteristic, a single loading space is considered; but in the object characteristic aspect, various sizes and the fixed stacking directions are considered to develop the algorithm according to the object and the container attribute.

Therefore, according to the definition of problem set, the container's attributes include length *cx*, width *cy* and high *cz*. In addition, object *b* has the following attribute information:

Object *b* totally has the *nb* types (*b*=1,2,3,…*nb*).
Object *b* with the length *bdim1*[*b*], the width *bdim2*[*b*] and the high *bdim3*[*b*].
Object *b* with the weight *bweight*[*b*] and volume *bvolume*[*b*].
The indicators of object *b* are the length *bind1*[*b*], the width *bind2*[*b*] and the high *bind3*[*b*]. Each dimension indicator constrains the object rotation. It indicates whether placement in the vertical orientation is permissible (=1) or not (=0).

Object stacking in container may calculate total placement volume and container utilization ratio. In this problem, the final goal is to derive the maximized spatial use ratio *R*.

$$R = \frac{\sum_{r=1}^{v} V_r}{\sum_{s=1}^{u} C_s} \qquad (1)$$

v：Already used object quantity.
Vr：The r_{th} has used object volume (r =1…v).
u：The number of containers.
Cs：The s_{th} container volume (s=1…u).

RESEARCH METHOD

Ant Colony Optimization

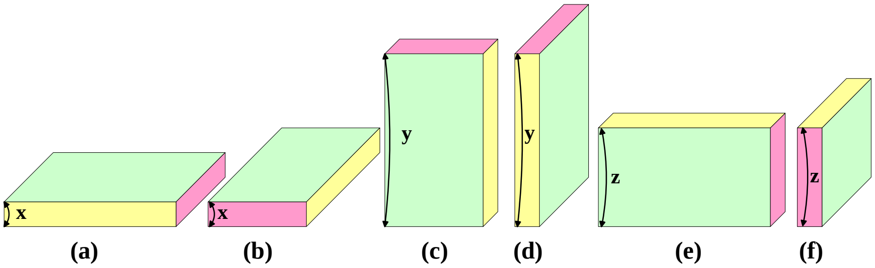
ACO is the method which imitates the real world ant's behaviors. The basic idea is to imitate the cooperative behavior of ant colonies in order to solve combinatorial optimization problems within a reasonable amount of time. ACO has good problem compatibility. In this paper, we proposed the ACO pheromone feedback technique to stack the object in the tower set *TSet* for the process of the next step.

Pheromone trial and initialization

Pheromone is an important medium for ant groups to search for communication information. Therefore, the design of the pheromone will directly affect the result of computation. Hence, according to the operation of ACO, we need to set up one two-dimensional pheromone structure first.

In order to make the settlement of the problem compatible with the operation of ACO, when the objects carry on stack, we make the basis object *g*, g_b(b=1,2,…,nb) is regarded as the type of the object *b*. The object put on *g* is *h*, and *hb* represents the type of object *h*. The objects of every classification all have six kinds of different rotations (fig. 1). Based on the stacking relations, the object may have (nb×6)×(nb×6) kinds of different combinations. We utilize this relation to get stacking situations with different type objects, and then construct the pheromone structure.

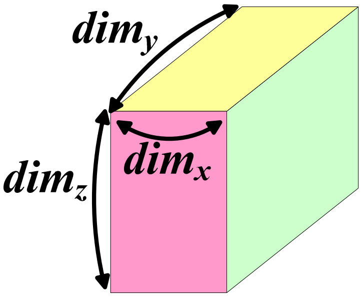
Figure 1: Six different rotations of an object.



In order to define the stacking relation between the objects in the pheromone structure, we must define a rule of the rotation for all objects. Provided that object o_b, as shown in fig. 2 has the rotation methods defined as follows:

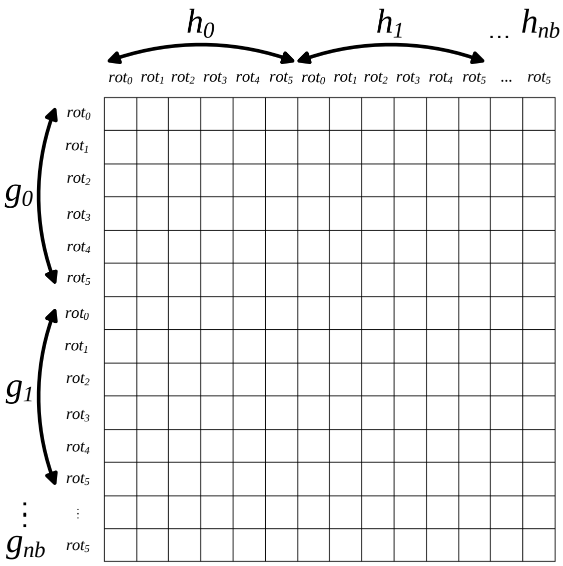
rot_0: The *x* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x>dim_y, as shown in figure 1(a).
rot_1: The *x* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x<dim_y, as shown in figure 1(b).
rot_2: The *y* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x>dim_y, as shown in figure 1(c).
rot_3: The *y* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x<dim_y, as shown in figure 1(d).
rot_4: The *z* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x>dim_y, as shown in figure 1(e).
rot_5: The *z* dimension of the object *ob* is perpendicular to the horizontal plane, and dim_x<dim_y, as shown in figure 1(f).

Figure 2: The dimensional definition of object o_b.



According to above-mentioned definitions, *g* and *h* have *nb* types respectively, and the objects of each type have six kinds of different rotations, so the stacking relation of the object *g* and *h* can be described by the following pheromone structure (fig. 3). Its initial value can be calculated by the following formula (1). If the rotation of objects *g* or *h* is restricted (its indicator is 0), then its initial value is 0.

Figure 3: The pheromone structure of the objects *x* and *y*.


h_0
h_1
$\dots h_{nb}$
rot_0 rot_1 rot_2 rot_3 rot_4 rot_5 rot_0 rot_1 rot_2 rot_3 rot_4 rot_5 ... rot_5
g_0
rot_0 rot_1 rot_2 rot_3 rot_4 rot_5
g_1
rot_0 rot_1 rot_2 rot_3 rot_4 rot_5
g_{nb}
rot_5

$$\left(rot_p, rot_q\right) = \begin{cases} 1 & if\left(rot_p, rot_q\right) \in Feasible \\ 0 & else \end{cases}$$
$$where \quad p = 0,1,\dots,5 \qquad q = 0,1,\dots,5 \qquad b = 1,2,\dots,nb$$
$$rot_p \in g_b \qquad rot_q \in h_b \tag{2}$$

Pheromone decision

During the stacking process, the decision of the ant is under the circumstances that any object g is a basis, then it chooses object *h* that is relatively suitable to be put on *g*. But the decision of the ant is a choice based on probability. So how to choose the suitable object to put on the basis in numerous objects depends on the transmission of the pheromone between the ants. We can calculate a probability P_{ij} of rotation of all possible objects *h* which can stack on *g* use the following formula (2).

$$P_{ij} = \frac{\tau_{ij} + \eta_{ij}}{\sum_{k \in D} (\tau_{ik} + \eta_{ik})} \tag{3}$$

Among them *i* is the index of rot_p□g_b in the pheromone structure (*i*=0,1,2, … , *nb*×6-1); j is the index of rot_q□h_b in the pheromone structure (*j*=0,1,2, … , *nb*×6-1). *D* is the rotations of all objects that may be chosen, and τ_{ij} is the pheromone value.

In addition to the influence of pheromone, the decision of probability is also affected by a heuristic value η_{ij}, which can let the ant move to the direction which may produce the direction of better solution and increase search efficiency. In order to make the algorithm more compliant to the problem, when object g and h are stacked on each other, η_{ij} is the contact area.

4.13 Local and Global Updating

The pheromone trial will be increased or decreased depending on the route that the ant passes by. A better route will usually increase more pheromone trail. During the process of building the tower, utilize pheromone feedback like this to help lead to a better combination gradually. So during the process, it determines that a new object is disposed, and then makes local updating to the pheromone structure immediately. The formula is shown as in (4) whereτ_0 is the average of all values when the pheromone trail is initialized. After the *m* ants all finish their own work; we will get the *m* tower sets. We sort the tower set by the tower utilization ratio separately in *m* tower sets, then calculate the average utilization ratio of the first 60% towers, define fitness value *F*. Finally we choose a tower set *TSet* with the best utilization ratio, and then make global updating to the pheromone structure.

$$\tau_{ij} = (1-\sigma)\cdot\tau_{ij} + \tau_0 \tag{4}$$

σ:Pheromone evaporated coefficient in local updating.

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + F\times 0.1 \tag{5}$$

ρ: Pheromone evaporated coefficient in global updating.

The method for construction towers

We have already introduced the role that ants play in the algorithm. Now we will describe how to use the ACO to construct the tower (as shown in figure. 4).

Figure 4: The ant colony constructed procedure.

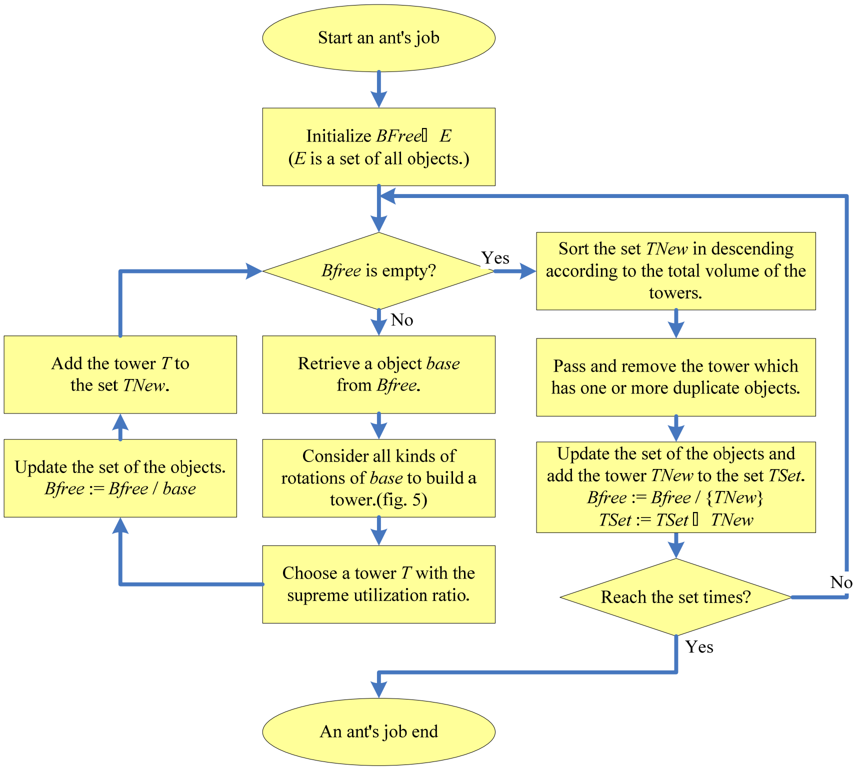
```
=========Phase I. ACO=========
Initialize Pheromone Trail

for  (nAntAge)
{
        for eachAnt
        {
                gen_TowerSet();
                LocalUpdating();
        }
        TSet = IterationBestTowerSet;

        GlobalUpdating(TSet);
}
TSet=GlobalBestTowerSet;

==========Phase II. GA==========
GA Solution(TSet);
```

Figure 5: The procedure of the constructing tower set.


Start an ant's job
Initialize *BFree* *E*
(*E* is a set of all objects.)
Bfree is empty?
Yes
No
Sort the set *TNew* in descending according to the total volume of the towers.
Pass and remove the tower which has one or more duplicate objects.
Update the set of the objects and add the tower *TNew* to the set *TSet*.
Bfree := *Bfree* / {*TNew*}
TSet := *TSet* *TNew*
Reach the set times?
No
Yes
Retrieve a object *base* from *Bfree*.
Consider all kinds of rotations of *base* to build a tower.(fig. 5)
Choose a tower *T* with the supreme utilization ratio.
Update the set of the objects.
Bfree := *Bfree* / *base*
Add the tower *T* to the set *TNew*.
An ant's job end

The procedure of constructing the tower is in a proper order, and incessant action of running too. The task of an ant is to choose the suitable objects and rotations to stack constantly. Then it feedbacks experience to the ants in the work or the ant of future generation, until one tower set is finished (The procedures are shown in figure 5 and figure 6).

Figure 6: The procedure of the building a tower.

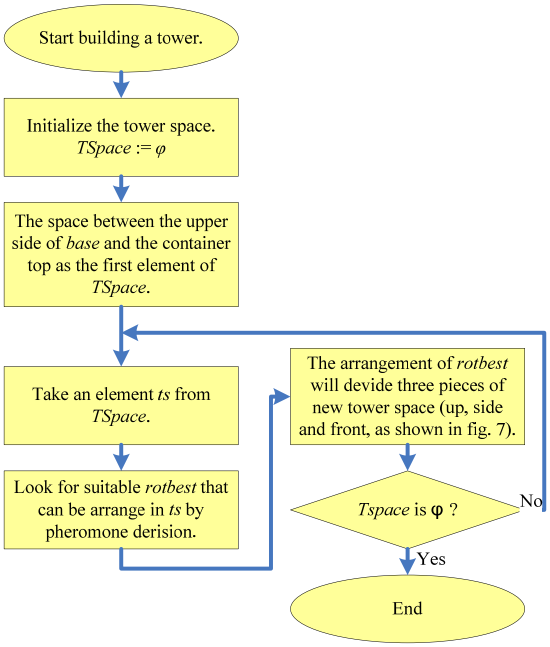
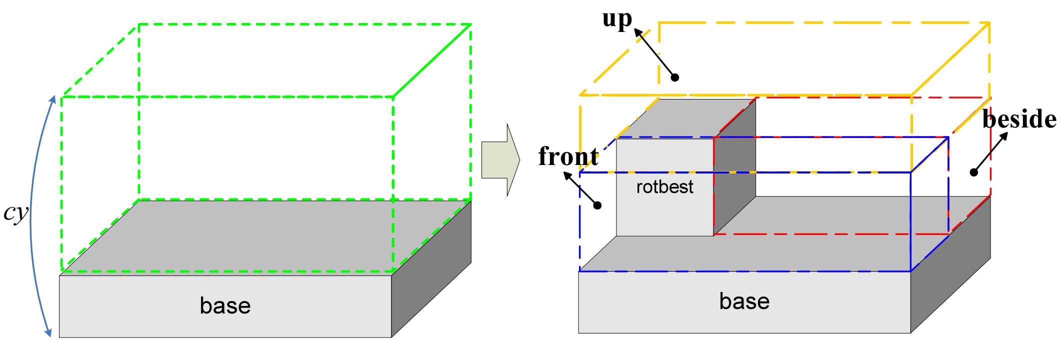

Start building a tower.
Initialize the tower space. *TSpace* := *φ*
The space between the upper side of *base* and the container top as the first element of *TSpace*.
Take an element *ts* from *TSpace*.
Look for suitable *rotbest* that can be arrange in *ts* by pheromone derision.
The arrangement of *rotbest* will devide three pieces of new tower space (up, side and front, as shown in fig. 7).
Tspace is φ ?
No
Yes
End

Figure 7: The tower space diagram.



Genetic Algorithm

GA (Genetic Algorithm) is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithm is categorized as global search heuristics. GA is a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

In the first phase, we have designed an algorithm to build the tower, and add the community activity of the ant colony. Finally we have used the pheromone feedback technique to help derive search optimization solutions. In second phase, we will utilize GA to find the sequence of placing the tower in the container space, and then look for the combination with the best space utilization ratio with this Corresponding rule. We expect to cooperate with the operator of GA to derive the approximate optimization solution within acceptable time.

The representation of the chromosome

We regard *nt* as the number of the tower in *TSet*, and denote each tower with the only serial number t (*t*=1,2, … , *nt*). The notation *sq* (*sq*=1,2, … , *nt*-1) is the serial number of the gene in the chromosome, it represents the arrangement of the tower *t* in the container bottom plane (figure 8). The feasible chromosome is the combination that does not contain redundant digitals. After that, we can carry out the operation of GA to this chromosome.

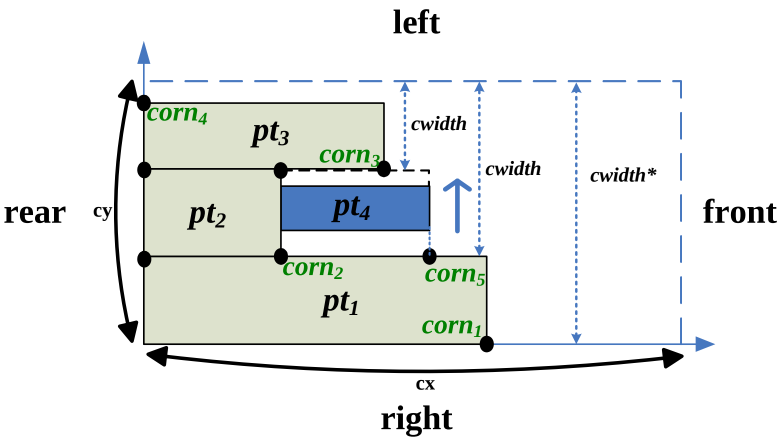
Figure 8: The chromosome diagram.

sq	0	1	2	3	4	5	…	*nt*-2	*nt-1*
t	2	1	*nt*	3	…	4	10	8	7

The decoding procedure and the fitness function

We design a decoding procedure for evaluating chromosome suitable degree, in order to make the genes of the chromosome corresponding to the arrangement of towers in the container's bottom plane. When we want to decode a piece of chromosome, we arrange *t* to the container's bottom plane according to the sequence of the genes in chromosome. The right-rear corner of the plane is regarded as the first placement corner *corn*. Then we determine the corners that each tower *t* placed on, until *t* are all arranged. The tower is pushed left as far as possible (shown as pt_4 in figure 9).

Figure 9: Update the placement corner.



The towers that have been arranged are called placement towers pt_s. After *t* have been arranged, we upgrade and fetch $corn_s$, the method is described as follows:

Each *corn* has an attribute *cwidth* which is the feasible space's width. There is a longest *cwidth* in these $corn_s$, we regard it as *cwidth**, and it is also the width of the container space. Except that the origin (the corner *corn* whose *X* coordinate *cornX* and *Y* coordinate *cornY* all equal 0.) is initial corn, if the following conditions are tenable, we can also call the corner *corn*:

The points at the intersection of the front edge of all pt_s and the adjoin *pt* (shown as $corn_2$ in figure 9).
The points at the intersection of the extended front edge of all pt_s and the opposite of *pt* (shown

as $corn_5$ in figure 9).
The points at the intersection of the left edge and the rear edge of all pt_s.
If the *cornX* of the *corn* equals 0 or the points at the intersection of the extended front edge and the *X* axis (shown as $corn_1$ in figure 9).

Figure 10: The coding procedure.

```
========================Decoding Procedure========================
Initialize the corner set Cset. (CSet = empty;)
The right-rear corner of the plane is regarded as the first element of CSet.

For( sq = 0 to nt-1){
        Take the $sq_{th}$ tower t according to the gene sequence of the chromosome.

        While(true){
                Take a corner corn from CSet according to the sequence of CSet.

                t.trot = 0;  //Make the tower t upright.
                             //(The tower's y-dimension exceeds the x-dimension).
                If( t.dimy >= corn.cwidth ){
                        The tower t is feasible to arrange on the corner corn.
                        break;
                }

                t.trot = 1;  //Make the tower t recumbent.
                             //(The tower's x-dimension exceeds the y-dimension).
                If( t.dimx >= corn.cwidth ){
                        The tower t is feasible to arrange on the corner corn.
                        break;
                }
        }

        Update CSet.        //Re-fetch the corners to the CSet according to the
                              placement towers.
        Sort CSet.          //Sort the corners with formula (6).
}
```

After all $corn_s$ are found, check all $corn_s$ and reject the $corn_s$ which are already covered by other pt_s. The remainder $corn_s$ will be utilized later.

Additional, it must comparatively approach the origin when all towers are put on the $corn_s$. So we sort the $corn_s$ after they are retrieved. The sort rule is as follows (6). The $corn_1X$ is represented the *X* coordinate of $corn_1$; and the $corn_2Y$ is represented the *Y* coordinate of $corn_2$. The sequence of the towers arranged is based on the sequence of the $corn_s$. The *corn* whose ranking is smaller will have more priority to be chosen and be arranged.

$$\begin{array}{ll} \text{If (} & (corn_1X < corn_2X) \quad \text{or} \\ & (corn_1X = corn_2X \quad \text{and} \quad corn_1Y < corn_2Y) \quad \text{)} \\ \text{Then} & corn_1 < corn_2 \end{array} \tag{6}$$

We utilize the decoding procedure to make each chromosome in the population corresponding to the arrangement of the towers on the container's bottom plane, and then compute the total space utilization ratio of each chromosome. The ratio represents the fitness value.

THE OPERATORS OF GENETIC ALGORITHM

Crossover

Crossover is an extremely important role in GA. Utilizing crossover; we can keep the advantage of previous generation. In this operator, we used two kinds of schemes, including the single point crossover and the two point crossover.

Mutation

If the chromosomes in the population are homogenized, in order to make them change, we star the mutation procedure under the regular probability. In this operator, we used scheme which are "random two points exchanging" and "reverse".

Selection

After the chromosomes in population have finished crossover, they are selected through a fitness-based procedure (fitness function), where fitter solutions are more likely to be selected. In this paper, we regard container's utilization ratio as the fitness value and filter two better chromosomes from the parent and the sub generation

EXPERIENCE

The experiment uses eight problem sets, including 15 problems in a problem sets and the other 7 problem sets with 100 problems specifically.

The first problem set which includes 15 problems was proposed by Loh and Nee (1992), Bischoff and Ratcliff (1995), and Ngoi and Chua (1994). All of them have been tested by their method. The following are experimental results compared to the different results: (a) represents the results of Loh and Nee (1992); (b) represents the results of Ngoi etc., (c) represents the results of Bischoff and Ratcliff, (d) represents the results of Bischoff etc., (e) represents ours. Each problem is tested 5 times and computed to derive an average value. Only the result in Table 1 (a) is a little exaggerated. When we check the problem set, we find the fact that the utilization ratios of some problems are already unable to increase again. The reason is that the results of some problems have already totally utilized the objects to stack in container space. The utilization ratio has been already limited. So besides (a), compared with the others, the proposed method has better results.

Table 1: LN (Loh and Nee) test problem results.

Problem	Utilization ratio (%)				
no.	(a)	(b)	(c)	(d)	(e)
1	78.1	62.5	62.5	62.5	62.5
2	76.8	80.7	89.7	90.0	89.7
3	69.5	53.4	53.4	53.4	53.4
4	59.2	55.0	55.0	55.0	55.0
5	85.8	77.2	77.2	77.2	77.2
6	88.6	88.7	89.5	83.1	91.4
7	78.2	81.8	83.9	78.7	84.6
8	67.6	59.4	59.4	59.4	59.4
9	84.2	61.9	61.9	61.9	61.9
10	70.1	67.3	67.3	67.3	67.3
11	63.8	62.2	62.2	62.2	62.2

12	79.3	78.5	76.5	78.5	78.5
13	77.0	84.1	82.3	78.1	85.6
14	69.1	62.8	62.8	62.8	62.8
15	65.6	59.6	59.5	59.5	59.5
Avg.	74.2	69.0	69.5	68.6	70.0

The problem sets which include 100 problems separately are provided by the website OR-Library. The following are the result and the comparison in this research, (a) represents Bischoff's experimentally result; (b), (c) and (d) are proposed by Lim etc., which represents the method "packing tree", "packing tree 1" and "packing tree 2" separately; (e) represents our results. Each problem set is tested 5 times and computed to derive an average value.

Table 2: The thpack test problem result.

Problem	Utilization ratio (%)				
set name	(a)	(b)	(c)	(d)	(e)
thpack1.txt	85.4	74.9	79.9	87.4	88.7
thpack2.txt	86.25	75.6	79.4	88.7	90.7
thpack3.txt	85.86	76.9	78.9	89.3	91.6
thpack4.txt	85.08	77.5	79.2	89.7	91.8
thpack5.txt	85.21	76.7	78.6	89.7	91.7
thpack6.txt	83.84	77.0	78.8	89.7	91.3
thpack7.txt	82.95	76.8	78.3	89.4	90.8

CONCLUSION

This paper resolved the problem about combination optimization. It is difficult to find the optimal solution in the limited time. Therefore, in order to increase efficiency and decrease complexity, the problem was decomposed and solved by phased method. First we employed the ACO based on essence of problem to get useful information that could support the GA. For the problem, we changed the pheromone structure that used pheromone feedback to investigate the different surface of rotation among objects with objects. We hoped that the ways that objects were stacked could be reacted more directly. Finally, GA was utilized. The purpose was to improve the utilization ratio of the tower sets first before the second phase. Once the utilization ratio of the tower was raised, the whole utilization ratio would increase too.

We observed from the experimental results, our method was generally better than others It showed that utilized feedback of pheromone to help build the tower that could increase utilization ratio, but we knew from the experimental process that the evolution of ant colony would take a lot of time, and the convergence time could hardly be reduced. The further research direction can focus on the Evaluation of ACO, and employ some different methods such as the updating strategy of pheromone and the evaluation... etc. This will make the evolution procedure more efficient and further increase utilization rate.

REFERENCE

Bischoff, E. E., & Ratcliff, M. S. W., (1995). Issues in the Development of Approaches to Container Loading, Omega, *International Journal of Management Science*, 23 (3), 377-390.

Bischoff, E. E., Janetz, F., & W.Ratcliff, M. S., (1995). “Loading pallets with non-identical items,” *European Journal of Operational Research*.

Bortfeldt, A., & Gehring, H., (2001). A hybrid genetic algorithm for the container loading problem,” *European Journal of Operational Research*, 131 (1), 143-161.

Dorigo, M., & Stutzle, T., (2001). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. *Technical Report IRIDIA-2000-32*.

Hehring, H., & Bortfeldt, A., (1997). A Genetic Algorithm for Solving the Container Loading Problem, *International Transactions of Operational Research*, 4,.401-418.

Lim, A., Rodrigues, B., & Yang, Y., (2005). 3-D Container Packing Heuristics, *Applied Intelligence,* 22, 125-134.

Loh, T. H., & Nee, A.Y.C., (1992). A packing algorithm for hexahedral boxes, in *Proceedings of the Conference of Industrial Automation,Singapore*, 115–126.

McCall, J., (2005). Genetic algorithms for modeling and optimization, School of Computing, Robert Gordon University, JCA, 205-222.

Moura, A., & Oliveira, J.F., (2005). A GRASP approach to the Container Loading Problem, *IEEE Intelligent Systems and Their Applications*, 20 (4), 50-57.

Ngoi, B. K. A., Tay, M.L., & Chua, E.S., (1994). Applying spatial representation techniques to the container packing problem, *International Journal of Product Research.*

Renner, G., & Ekart, A., (2003). Genetic algorithms in computer aided design, *Computer-Aided Design* 3, 709-726.

http://people.brunel.ac.uk/~mastjjb/jeb/info.html.