IEMS Vol. 3, No. 1, pp. 22-37, April 2004.

A GA-based Heuristic for the Interrelated Container Selection Loading Problems

Anulark Techanitisawad† · Paisitt Tangwiwatwong
Industrial Engineering & Management School of Advanced Technologies
Asian Institute of Technology P.O.Box 4, Klong Luang Patumtani 12120, Thailand
Tel: +66-2-524-5689, E-mail: anulark@ait.ac.th (A. Techanitisawad)

Abstract. An integrated heuristic approach based on genetic algorithms (GAs) is proposed for solving the container selection and loading problems. The GA for container selection solves a two-dimensional knapsack problem, determining a set of containers to minimize the transportation or shipment cost. The GA for container loading solves for the weighted coefficients in the evaluation functions that are applied in selecting loading positions and boxes to be loaded, so that the volume utilization is maximized. Several loading constraints such as box orientation, stack priority, stack stability, and container stability are also incorporated into the algorithm. In general, our computational results based on randomly generated data and problems from the literature suggest that the proposed heuristic provides a good solution in a reasonable amount of computational time.

Keywords: genetic algorithms, container loading, 3-dimensional packing, multi-container loading

1. INTRODUCTION

Logistics is usually regarded as a crucial function in an organization, since for sales to be realized, firms must deliver or distribute their goods to customers, warehouses, or distribution centers. In the present distribution and transportation systems, merchandise is normally loaded into a container for economical movement and ease of handling and transporting in a variety of modes. In manual operations, the container type and the number of each type to be used must be *a priori* determined to minimize total transportation cost for a shipment. However, due to space wastage that is unknown during this preliminary stage, it is quite often difficult in practice to determine a minimum-cost set of containers that can stow all loads. When containers are not well packed, additional containers may be required to transport all boxes and those containers result in a higher shipping cost. In addition, if there is any space remaining in a container as a result of the maximum volume utilization of containers, more merchandise can be added into the container to increase sales volume and revenues to exporters, as currently practiced in the import/export business. The container selection and loading activities, thus, deserve attention for improvement in their cost, effectiveness, and efficiency.

Most papers in the literature only address the container loading problem. Although there are many approaches developed for the container loading problem, according to Bischoff and Ratcliff (1995a), most existing approaches are limited when dealing with practical requirements, for example, the box orientation, load bearing strength, load stability, and shipment priorities. This paper, thus, addresses the interrelated container selection and container loading problems with several practical loading constraints, and also proposes a genetic-algorithm-based heuristic to solve such problems iteratively.

Given a number of rectangular boxes of products or goods in different sizes and weights and different types of rectangular containers, each of which has a specific size with fixed dimensions and weight capacity, the interrelated problems determine a combination of the container types and the number of containers of each type that minimize the total transportation or shipment cost, and a loading pattern of boxes in each container that maximizes the volume utilization of the container subject to the following practical loading constraints:

1. The box orientation constraint specifies for a box any orientation restriction in loading. Some box types may have only one uppermost face but others may have any uppermost face.
2. The stack priority constraint addresses the strength and

† : Corresponding Author

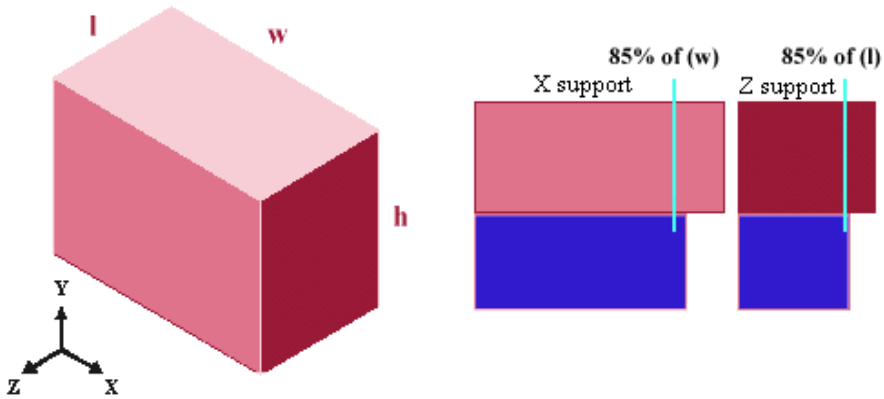

Figure 1. The stack stability constraint

density of the box or the unloading priority. A box with higher strength has a higher priority being loaded first. A box can only be stacked on top of boxes with equal or higher priority.

3. The stack stability characterized in Figure 1 guarantees that each box has enough support under its base and is within the support parameters required by the user. The X support parameter is specified by the percentage of the width dimension and the Z support parameter is specified by the percentage of the length dimension of the box that must be supported by boxes below, assuming that X-Z is the corresponding base area of a box relative to its orientation.
4. The stability of the container is determined by the center of gravity of a fully loaded container. It must be sufficiently stable or balanced in both transverse and longitudinal planes for ease of handling and safety purposes, assuming that the weight of each box is distributed uniformly over the box shape.

In terms of the solution algorithm, the genetic algorithm (GA) concept was formally introduced in 1970s by John Holland at the University of Michigan. It is a search procedure based on the mechanism of natural selection and genetics (Goldberg, 1989), using a process similar to biological evolution to improve a set of feasible solutions called a population or generation through an iterative process based on a fitness or evaluation function. The genetic operators include reproduction, cloning, crossover, and mutation. GAs have been extensively and successfully applied to a variety of combinatorial optimization problems. Motivated by success and flexibility of the GAs, we devised an extended version of Kawakami *et al.*(1991) and proposed an integrated algorithm for iteratively solving the container selection and loading problems.

The remainder of the paper is organized as follows: Section 2 provides a brief literature review mostly on container loading. Section 3 proposes the integrated solution procedure, describing in details the selection and loading modules, while Section 4 presents the computational evaluation and analysis of the procedure and the comparison of its performance with other existing methods. Section 5 gives conclusions and suggestions for future research.

2. LITERATURE REVIEW

The container-loading problem is normally classified as the three-dimensional packing problem found in the existing literature. Dyckhoff (1990) systematically compiled and classified different types of cutting and packing problems. Relative to the large volume of works addressing one or two-dimensional cutting and packing problems, only a limited number of researches have dealt with three dimensions. Most works are concerned with practical, heuristic solution procedures due to the problem's well-known complexity; however not many practical loading constraints have been addressed.

A heuristic procedure for packing boxes into a container using the layer concept was firstly developed by George and Robinson (1980). This heuristic allows the boxes to be stacked with any uppermost face and no restriction on the number of boxes stacked on top, while the boxes of the same type are restricted to be stacked in proximity. The heuristic procedure is based on the concept of filling the container layer by layer, a section of the length over the complete height and width of the container. The layer concept has become the basic idea of some subsequent developments, such as Gehring *et al.* (1990), Bischoff and Marriott (1990), Bischoff and Ratchiff (1995b), Davies and Bischoff (1999), and Pisinger (2002).

Metaheuristics, namely tabu search and genetic algorithms, were also applied to the container loading problem. Bortfeldt and Gehring (1998) applied a tabu

search algorithm in a two-stage procedure, while Bortfeldt and Gehring (2001) proposed a hybrid GA based on the layer concept for solving the container loading problem with several practical constraints. A chromosome representing a stowage plan was generated by a variant of the basic loading heuristic of Gehring *et al*. (1990). The crossover and mutation operators were done on layers, substructures of the stowage plan. The computational results showed quite an improvement in volume utilization, when compared with other algorithms. The practical constraints addressed in Bortfeldt and Gehring (2001) are similar to ours, although differences do exist. Their so-called stability and stacking constraints are quite restrictive, while we allow flexibility for the user to specify the minimum required base support of a box and stack priority. In terms of the container balance, we also check for the stability of an inclined container.

Kawakami *et al.* (1991) proposed a mechanism in which the three-dimensional packing rule is automatically adjusted by applying a genetic algorithm. The packing strategies used to select the next loaded position and box are controlled by two evaluation functions, consisting of the weighted sum of the problem characteristics. Their weighted coefficients are prescribed by the genetic algorithm. However, limitations do exist in that all of the boxes can have only one fixed orientation, and that no practical aspects were considered in the packing procedure. Motivated by the flexibility of Kawakami *et al*. (1991), we further modified their GA approach to solve our combined container selection and loading problem, while simultaneously considering several practical loading constraints.

In addition to the heuristics, a few papers proposed the analytical approach, attempting to find exact solutions. Tsai *et al.* (1993) formulated a mixed-integer programming model for the three-dimensional pallet loading problem. As expected, the significant computational time requirement of the model does limit the practical use in real-time palletizing applications, and the computation time was found to increase exponentially as the number of different box sizes increases. Ngoi *et al*. (1994) solved the container-packing problem by applying a spatial representation technique that represents the objects and empty space as a combination of variable orthorhombic cells by means of a simple matrix structure. Chen *et al.* (1995) formulated the problem into a zero-one mixed integer-programming model which incorporates many practical restrictions, such as carton orientations, multiple carton sizes, multiple container sizes, avoidance of carton overlapping, and space utilization. Several aspects of the container selection and loading problem, such as selecting one container from several alternatives, weight distribution of the cartons in the container, and variable container length were also addressed. However, as in other analytical approaches, this model suffered from long computational times and, thus, is not practical for real applications.

For the combined container selection and loading problem, Laotaweesub (1996) proposed an iterative method to determine a set of containers that minimizes the total transportation cost and a loading pattern for each container that maximizes the volume utilization. The container selection was solved by an integer-programming model whose constraints include the volume and payload capacity of the selected containers. The container loading algorithm, on the other hand, was based on the layer concept of Gehring *et al*. (1990), while considering the stack restriction, box orientations, and container stability. Although Laotaweesub (1996) dealt with many practical constraints, weakness still exists in the loading procedure, due to the high dependency of the volume utilization on the choice of the reference box. This paper addresses the combined container selection and loading problem as in Laotaweesub (1996), however, with the additional stack stability constraint, and further improves the effectiveness and efficiency of the integrated solution procedure.

3. THT INTEGRATED HEURISTIC SOLUTION PROCEDURE

The integrated solution procedure summarized in Figure 2 is an iterative procedure combining two main heuristic algorithms, namely the container selection algorithm (Section 3.1) and the container loading algorithm (Section 3.2). The container selection algorithm selects a combination of the available containers for loading all boxes, minimizing the total transportation or shipment cost. The container loading algorithm, on the other hand, loads boxes into one container at a time, resulting in a loading pattern of each selected container. The practical loading constraints and limitations are considered in the loading heuristic. Once all boxes are loaded, the integrated procedure terminates. Otherwise, the container selection algorithm is resolved, after accordingly increasing the minimum required volume (MRV) by the total volume of the remaining boxes and, only if necessary, the minimum required weight (MRW) by the total weight of the remaining boxes. However, in almost all practical cases, MRW normally remains unchanged in all iterations, unless the weight constraint is significantly dominant. Given the updated values of MRV, a new set of containers is determined and the integrated procedure repeats as shown in Figure 2. For the same containers suggested by both old and new sets of containers, their loading solutions are retained, whereas the rest must be unloaded, and those boxes unloaded must be returned to the remaining box list. Consequently, only new containers will be loaded with the remaining boxes.

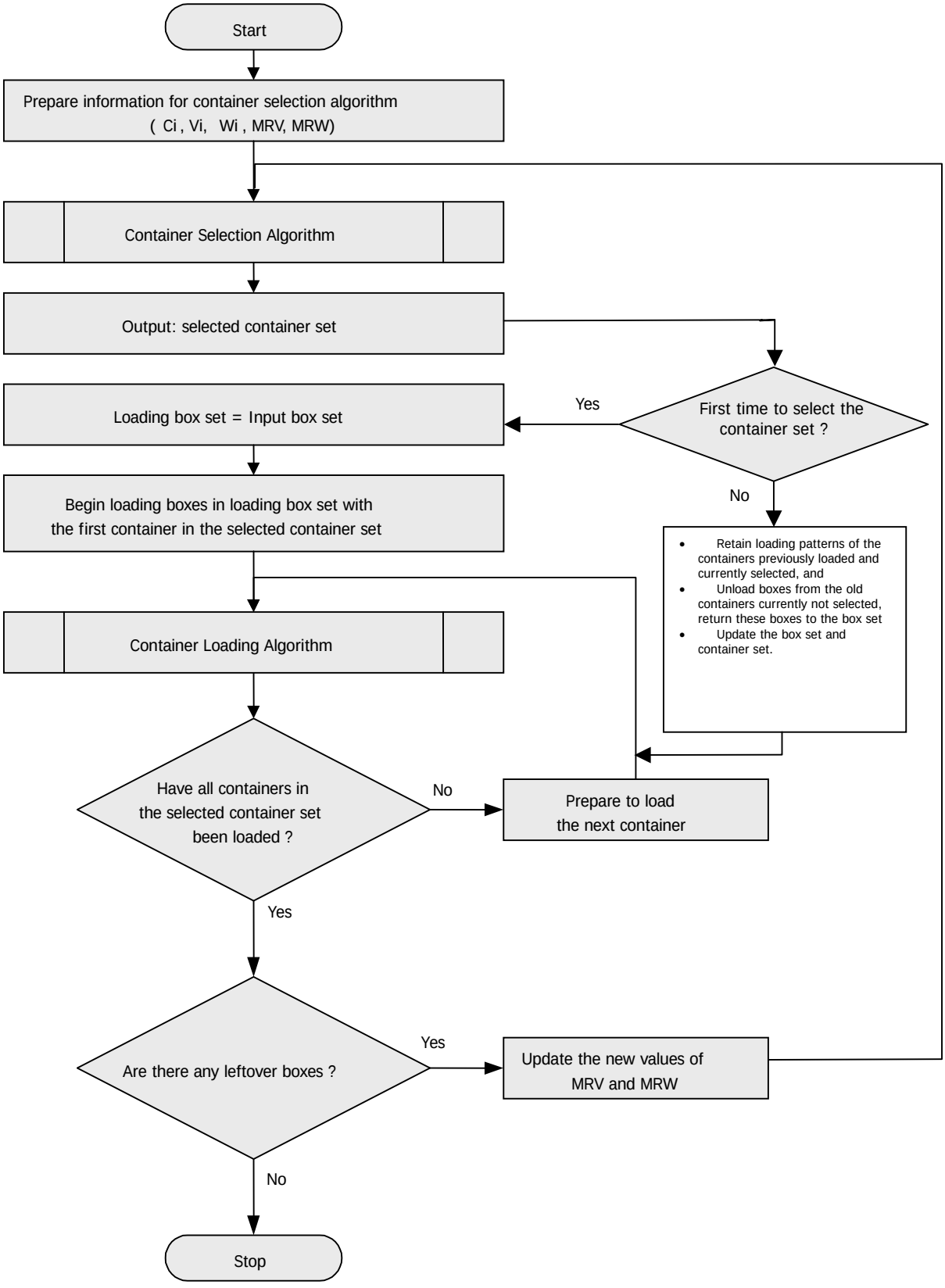

Figure 2. The integrated container selection and loading algorithms

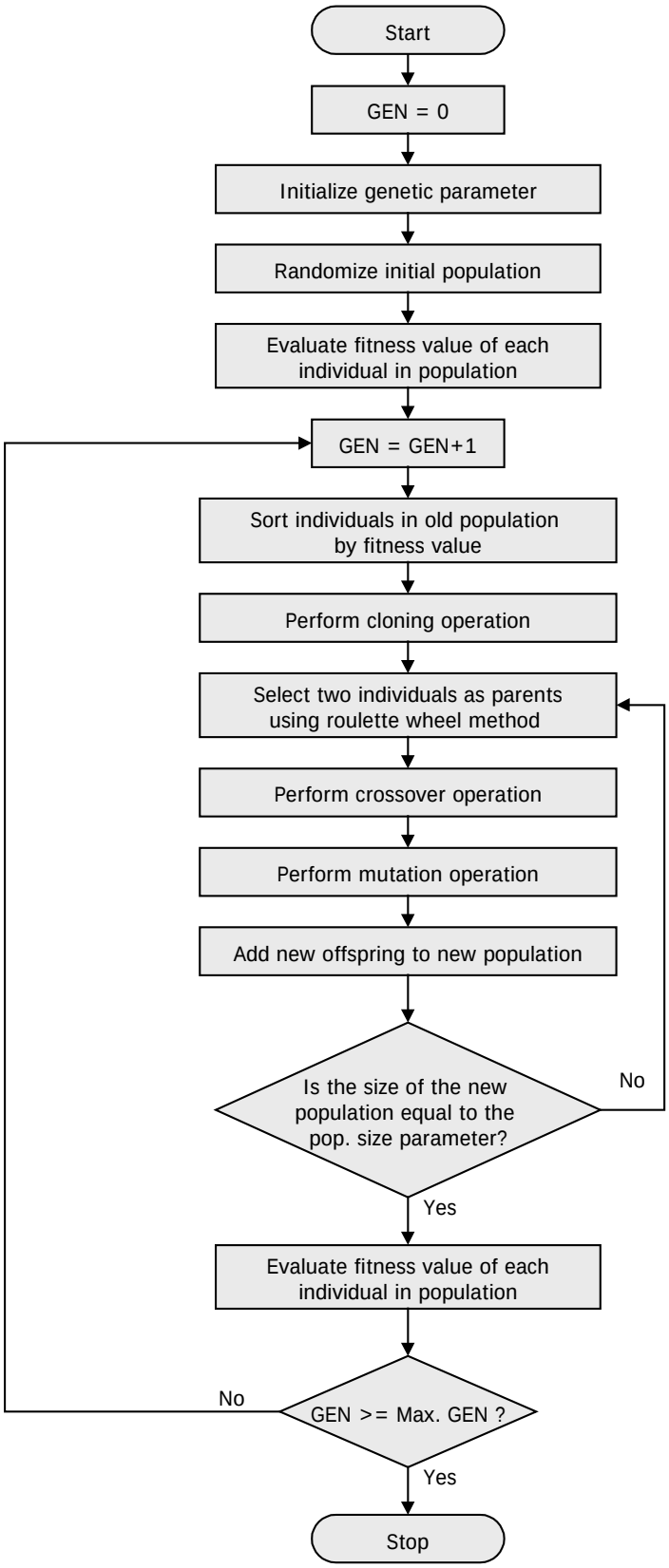

Figure 3. The genetic algorithm for the container selection problem

3.1 Container Selection Problem and Algorithm

Laotaweesub (1996) proposed a simple two-dimensional knapsack model to find an optimal set of containers that minimizes the total shipment cost, subject to the volume capacity and the weight capacity constraints. The model can be written as follows:

$$\textit{Minimize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\textit{subject to} \quad \sum_{i=1}^{n} v_i x_i \geq MRV$$

$$\sum_{i=1}^{n} w_i x_i \geq MRW$$

$$x_i \geq 0, \text{ integer} \quad \textit{for all } i,$$

whereas:

I = container type; $i = 1, 2, 3, \ldots, n$ (positive integer)

x_i = Number of containers of type I

c_i = Freight rate or transportation cost of container type i

v_i = Volume capacity of container type i

w_i = Weight capacity or maximum payload of container type i

MRV = Minimum required volume for loading all boxes

MRW = Minimum required weight for loading all boxes

In this study, we propose a genetic algorithm to solve the two-dimensional knapsack problem above for four reasons. Firstly, the problem is of a small size and the heuristic approach such as GA can often provide good solutions. From our preliminary testing (not reported), the GA can, in almost all cases, obtain optimal solutions quickly. Secondly, due to the space wastage that incurs inevitably after loading, we found that optimality of the solution in this part may not be as critical as that of the container loading. Thirdly, by using the GA, we can be positive that for a larger problem with a higher number of different container types, the runtime will normally be fast. Lastly, in order to minimize the programming effort, the same GA code can be used for both container selection and loading algorithms.

A simple GA for the container selection problem was proposed as follows. A solution is represented by a chromosome which consists of the number of each container type $\{x_1, x_2, x_3, \ldots, x_n\}$. The initial population is randomly generated. Each chromosome is evaluated using the following evaluation function:

Evaluation function = Total transportation cost + Volume penalty + Weight penalty, in which the penalty is M (a big positive number) if the corresponding constraint is violated.

In each generation, a new population is first created by the cloning operator with a specific percentage of cloning using the elitism concept. Two chromosomes or "Parents" are selected from the previous generation using the roulette wheel method. The one-point crossover is then performed on the parents to obtain two new chromosomes or "Offsprings". The crossover is repeated until the number of new chromosomes including cloned ones is equal to the population size. In addition, the mutation operation may be performed with the mutation probability on the new offsprings generated from the crossover operations, selecting a position in the chromosome randomly, and then replacing the value at that position with a new randomly generated value. The solution of the container selection GA is obtained from the best chromosome in the final generation. Its algorithm is illustrated in Figure 3.

3.2 CONTAINER LOADING PROBLEM AND ALGORITHM

3.2.1 Loading Concept

Our container loading algorithm was modified from Kawakami *et al.* (1991), in which boxes are loaded into a container in the two following steps:

Step 1: Select a loading position in a container.

A best loading position is a point or coordinate (x_k, y_k, z_k) that minimizes, among all possible positions at which a box could be located, the point evaluation function:

$$P_k = p_1 \cdot x_k^2 + p_2 \cdot y_k^2 + p_3 \cdot z_k^2 + p_4 \cdot x_k \cdot y_k + p_5 \cdot x_k \cdot z_k + p_6 \cdot y_k \cdot z_k + p_7 \cdot x_k + p_8 \cdot y_k + p_9 \cdot z_k,$$

in which $p_1, p_2, p_3, \ldots, p_9$ are the weighted coefficients.

Step 2: Select a box to be loaded at the selected loaded position.

A best box to be loaded is one that maximizes, among all remaining box i, the box evaluation function:

$$B_i = \sum_{j=1}^{9} b_j \, factor_j,$$

in which b_j is the jth weighted coefficient and $factor_j$ is the jth evaluation factor which is defined to reflect the individual characteristics of box i as follows:

$factor_1$	$= (w_i \cdot h_i \cdot l_i)$	Volume of box i
$factor_2$	$= (w_i \cdot h_i) / (W \cdot H)$	Ratio of the x-y area of box i and the container
$factor_3$	$= (w_i \cdot l_i) / (W \cdot L)$	Ratio of the x-z (base) area of

box i and the container

$factor_4 = (h_i \cdot l_i) / (H \cdot L)$ Ratio of the y-z area of box i and the container

$factor_5 = (h_i)^2 / (w_i \cdot l_i)$ Ratio of y (height) to the x-z (base) area of box i

$factor_6$ = orientation type of box i

0: A box has only one fixed position relative to the container.

1: A box has the fixed height dimension or only one uppermost face, while the width and length dimensions (base dimensions) are interchangeable.

2: A box can be put in any orientation with any uppermost face.

$factor_7$ = stack index of box i

The stack index, a positive integer predetermined by the user, indicates the stackability of a box. A box with a low stack index is of a high density and high strength, and thus, has a high priority to be loaded first. Only a box with a higher stack index can be stowed on top.

$factor_8$ = (total number of boxes of box i type) ÷ (weight of box i type) / total weight of all boxes

$factor_9$ = total number of boxes of box i type / total number of all boxes

The point and box evaluation functions proposed here are extensions of those presented by Kawakami *et al.* (1991) that included only three factors. From our preliminary computational experience, having more factors that reflect different problem characteristics often resulted in better solution quality. The weighted coefficients of the evaluation functions, on the other hand, are determined by a genetic algorithm to maximize the volume utilization or, equivalently, to minimize the space wastage of a container. The GA for container loading is described in details in Section 3.2.3.

3.2.2 The Loading Process Algorithm

Given a container and boxes, the loading process attempts to pack boxes, if not all, into the container, using the point and box evaluation functions described above and checking volume and weight feasibility and practical limitations including stack priority and stability. The loading algorithm is summarized in Figure 4.

Based on the box evaluation function, all boxes are sorted in a nonincreasing order of B_i. The set of loading points is initialized and (0,0,0) is set as the first and currently chosen point or loading position. If there is more than one possible loading position in the point set, the best point k that minimizes P_k is selected as the current point. For each current point, the first box in the box type set or list is considered as the "current box" to be tried for loading. Given all possible orientations of the current box, the loading constraints are then checked. A box can be loaded only if all of the following conditions are met: (1) *size*: it fits into the remaining space based on the current loading point; (2) *overlap*: it does not overlap with any other loaded boxes; (3) *stack priority*: its stack index is higher than that of the box below or that of the current loading point; (4) *stack stability*: it has sufficient base support as required by the user; (5) *weight capacity*: the total weight of the loaded boxes must not exceed the maximum payload of the container. If all checks are passed, the current box is loaded at the current point; both are then removed from their respective sets; the three new points are added to the point set; and the process repeats. Otherwise, the next box in the list is considered. If no remaining box can be loaded at the current point, this point is deleted from the list, and another point is selected. The loading process continues until no box remains or no points can be loaded by any remaining boxes.

3.2.3 The Genetic Algorithm for the Container Loading

To achieve a good loading pattern that maximizes the volume utilization of a container, it is important to prescribe the sets of weighted coefficients p_j and b_j respectively for the point and box evaluation functions that best suit the loading problem at hand. Proposed by Kawakami *et al.* (1991), the GA is flexible and capable in searching the set of weighted coefficients without specific structures or prior information.

The general GA framework and components as described in Section 3.1 are applicable to the container loading as well, except where indicated otherwise. Here, each chromosome is, instead represented by real values of the weighted coefficients $\{p_1, p_2, p_3, \ldots, p_9, b_1, b_2, b_3, \ldots, b_9\}$. Those in the initial population are randomly generated. The random crossover was found in our preliminary experiment to outperform the one- and two-point crossovers, and, thus, implemented.

Since the overall objective is to maximize the volume utilization of a container, each chromosome can only be evaluated after the loading process has been completed, as shown in Figure 5, using the following fitness function:

Fitness function = Container volume utilization* Penalty factor,

in which the penalty factor = 1 if the container stability is satisfied, or

= $1/M$ otherwise.

The container stability is checked after the container has been completely loaded. Given that x_i, y_i, z_i are positions of the center of gravity of loaded box i in x-, y-, and z-axes, respectively, the center of gravity of the loaded container

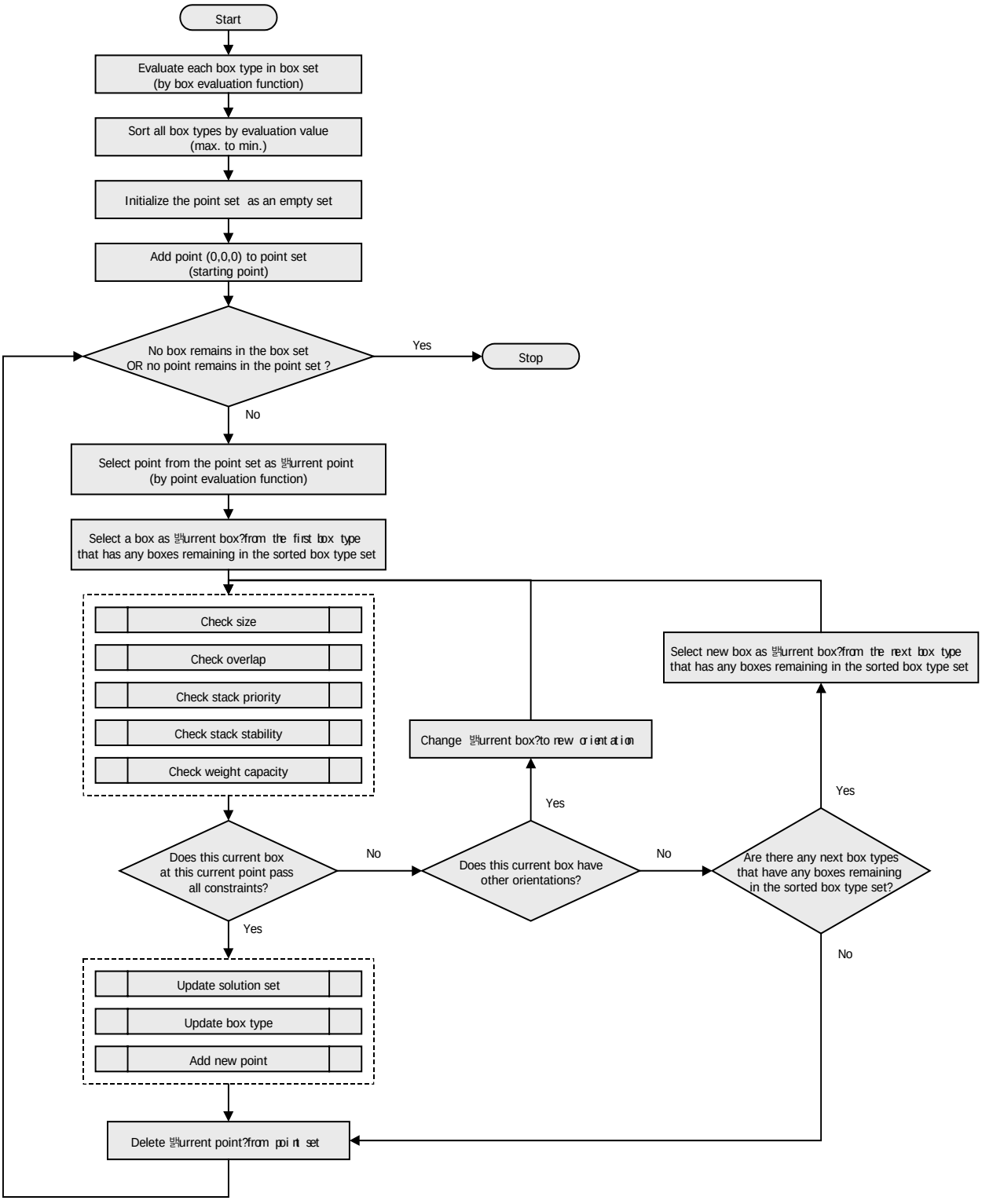

Figure 4. The loading process algorithm

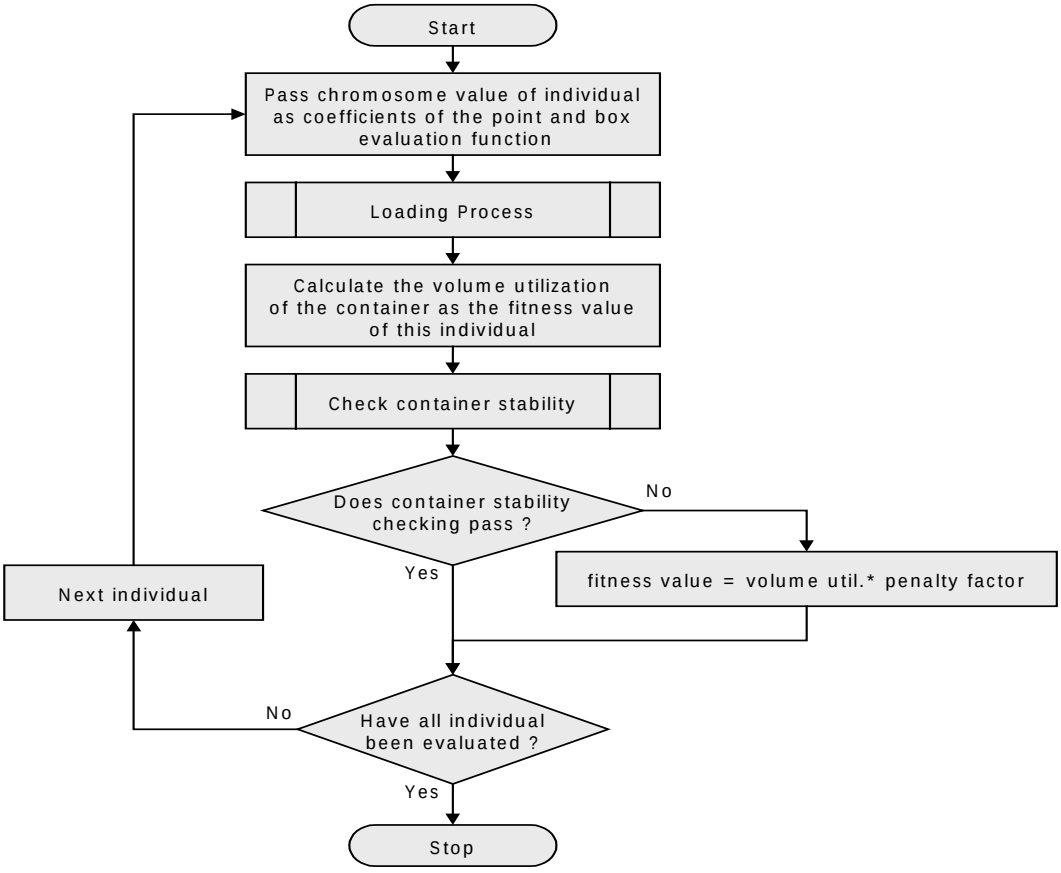

Figure 5. The evaluation process of each chromosome in the container loading

$$(X,Y,Z) = \left(\frac{\sum_i m_i x_i}{\sum_i m_i}, \frac{\sum_i m_i y_i}{\sum_i m_i}, \frac{\sum_i m_i z_i}{\sum_i m_i} \right),$$

where m_i is the mass of the box i.

Figure 6a characterizes two different methods in checking for the container stability. One is by the horizontal range limit, within which the center of gravity (X, Y, Z) of the container must fall in both transverse and longitudinal planes, so that the container is considered stable for handling. The other is by the maximum inclined angle, with which (X, Y, Z) must falls within its base in both transverse and longitudinal planes. In this case, the following conditions are checked:

$\text{Arctan}(X/Y) \geq \theta_T$ and $\text{Arctan}((W-X)/Y) \geq \theta_T$ in the transverse plane (see Figure 6b);

$\text{Arctan}(Z/Y) \geq \theta_L$ and $\text{Arctan}((L-Z)/Y) \geq \theta_L$ in the longitudinal plane (see Figure 6c);

where:

θ_T is the maximum inclined angle in the transverse plane.

θ_L is the maximum inclined angle in the longitudinal plane.

W is the width of the container.

L is the length of the container.

4. COMPUTATIONAL RESULTS AND ANALYSIS

The integrated container selection and loading system was implemented in Microsoft Visual Basic 6.0, providing a user interface for input data and results including a graphical loading pattern. All experiments were conducted on an IBM-PC compatible with the AMD Athlon 700 MHz CPU and 64 MB RAM. From a preliminary experiment (not reported), we found that the optimality of the set of containers selected in the container selection algorithm has very little effect on the quality of the overall solution, which depends rather heavily on the performance of the container loading algorithm. All of our experiments, except indicated otherwise, are, therefore, focused on the effectiveness and efficiency of the container loading problem. Genetic parameters were also tested and recommended. In addition, computational times and comparisons with results from the literature are reported.

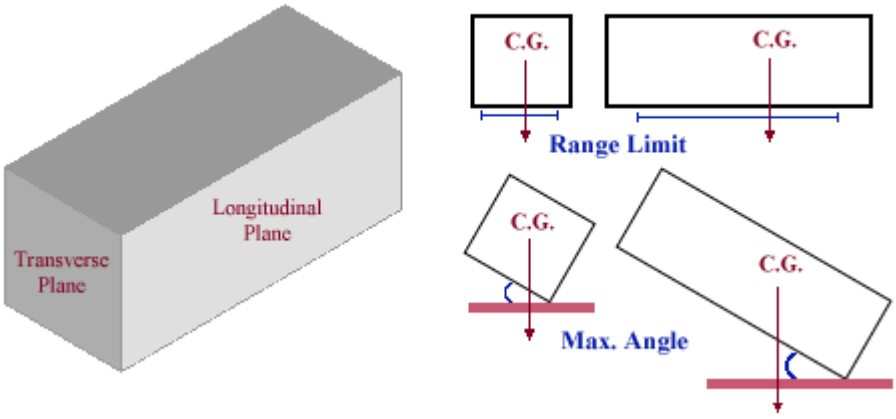

Figure 6a. Conditions for the container stability

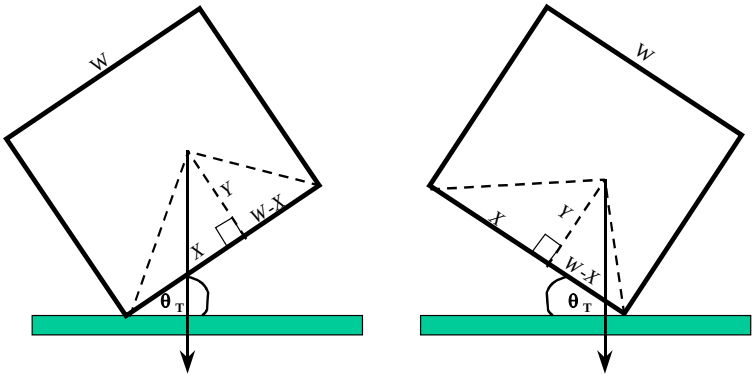

Figure 6b. Container stability with maximum inclined angle in the transverse plane

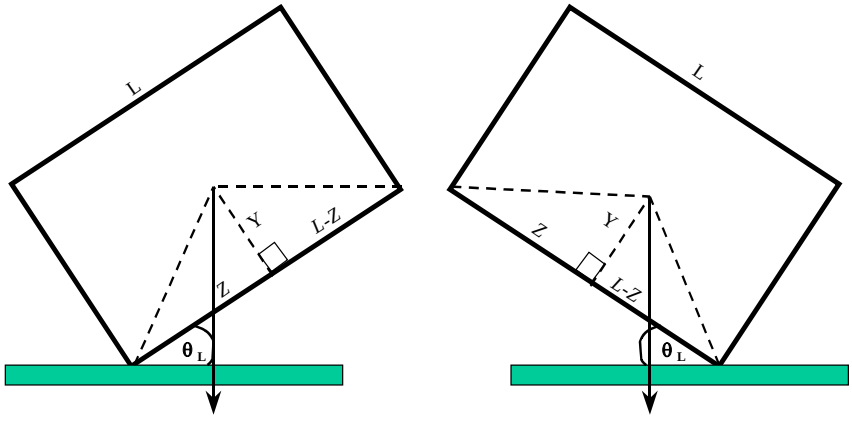

Figure 6c. Container stability with maximum inclined angle in the longitudinal plane

4.1 Randomly Generated Container Loading Test Problems

Thirty container loading test problems were randomly generated using the uniform distribution. Experimental factors include two levels of box dimensions (30-90 cm and 30-180 cm.) and four levels of the number of box types (1-5, 6-20, 21-50, and 51-100 types). The range of the number of boxes per type consequently depends upon the level of number of box types. The differences in box dimensions, on the other hand, prescribe in each level the range of box dimensions. Based on these factors and ranges specified in Table 1, other box data such as dimensions of boxes and restrictions on orientation, stack index, and weight were randomly generated. Table 2 summarizes the levels of the experimental factors for each test problem.

The random generation procedure for each test problem is as follows:

Step 1 : Generate the level of differences in box dimensions, the level of the number of box types, and correspondingly the number of box types for each box type.

Step 2 : For each box type, generate the parameters in this particular order: (i) number of boxes, (ii) box dimensions based on the range specified by the level of differences in box dimensions, (iii) box orientation, (iv) stack index, (v) weight per volume

Step 3 : Generate the container dimensions and container weight capacity per volume.

Step 4 : Accept the generated problem only if the volume of the container is less than the total volume of all boxes and the maximum payload of the container is greater than the total weight of all boxes. Otherwise, repeat all steps.

Generally, difficulty in the container loading problem has been known to be associated with the volume maximization, while the maximum payload of the container is, normally, not the constraint in practice. To obtain meaningful and practical results, we imposed the above conditions on the generated problems.

4.2 Genetic Algorithm Performance and Control Parameters

In a genetic algorithm, several control parameters need to be set appropriately for each application to optimize its performance. We conducted an extensive preliminary experiment to determine suitable parameter values, and noted that the GAs for both container selection and loading problems performed well and resulted in good solution quality, with a high crossover rate of 0.85-0.95, the cloning rate of 0.05-0.15, and the mutation rate of 0.05-0.15. The experiment also showed that the GA normally found better solutions when the population size and the number of populations are large. Realizing the trade-off with the computational time, the recommended values of the control parameters for both GAs are summarized in Table 3. Note that specifically for the container loading GA, the maximum number of generations is set, based on the variable optimization level indicated by the user. The optimization level trades off between the computational time and solution quality. In our implemented system, we allow the optimization level to vary in ten levels each of which prescribes a different number of generations as specified in Table 3. The use may choose a low optimization level, resulting in a short computational time, but may sacrifice the quality of loading solutions obtained, and vice versa. The recommended values were subsequently used in all experiment that follows.

4.3 Effect of Loading Constraints

The stack stability of boxes is controlled by the required base support parameters supplied by the user. The effect of the stack stability was studied for its impact on the quality of solutions. From the experimental results, it is interesting to note that to relax or tighten this constraint does not directly or consistently increase the volume utilization, but the solution quality is rather problem dependent. However, we strongly recommend that this constraint be included in the loading process for a more realistic result, and that high values of the parameters be set to guarantee the stack stability.

We also conducted the experiment to study the effect of other loading constraints. The population size and the maximum number of generations were set to be 60 and 10, respectively, while other genetic algorithm parameters were set as recommended. The box orientation and the stack priority constraints follows the descriptions of test problems, whereas the stack stability was set at the support of 0.85 for both X and Z axes. The container stability was considered with the maximum angle of 7.29° for the transverse plane and of 45° for the longitudinal plane, as suggested by Laotaweesub (1996).

The results showed that, in general, the loading solutions achieved lower volume utilization when the loading constraints exist. The explanation for this is that the loading constraints, except the stack stability constraint, reduce the number of feasible solutions and the solution space, resulting in less volume utilization of the container. However, from our thirty experimental problems, the existence of the loading constraints reduced, on the average, by 3.13% of the volume utilization without loading constraints.

Table 1. Parameters of the experimental problems

Parameter	Range	Unit	Remarks
Box Parameters			
Differences in Box dimension (w, h, l)	30 ~ 90	cm	level 1st
	30 ~ 180	cm	level 2nd
Number of box types	1 ~ 5	types	level 1st
	6 ~ 20	types	level 2nd
	21 ~ 50	types	level 3rd
	51 ~ 100	types	level 4th
Number of boxes	1 ~ 100	boxes/type	for number of box types (1st level)
	1 ~ 25	boxes/type	for number of box types (2nd level)
	1 ~ 10	boxes/type	for number of box types (3rd level)
	1 ~ 5	boxes/type	for number of box types (4th level)
Orientation	0, 1, 2		
Stack index	1 ~ 10		
Box weight per volume	451 ~ 500	kg/m^3	for stack index #1
	401 ~ 450	kg/m^3	for stack index #2
	351 ~ 400	kg/m^3	for stack index #3
	301 ~ 350	kg/m^3	for stack index #4
	251 ~ 300	kg/m^3	for stack index #5
	201 ~ 250	kg/m^3	for stack index #6
	151 ~ 200	kg/m^3	for stack index #7
	101 ~ 150	kg/m^3	for stack index #8
	51 ~ 100	kg/m^3	for stack index #9
	1 ~ 50	kg/m^3	for stack index #10
Container Parameters			
Container width (W)	2000 ~ 4000	mm	
Container height (H)	2000 ~ 4000	mm	
Container length (L)	5000 ~ 13000	mm	
Container weight capacity per volume	300 ~ 600	kg/m^3	

Table 2. The thirty randomly generated test problems

Experimental problem	Level of difference in box dimensions	Level of number of box types	Number of box types	Experimental problem	Level of difference in box dimensions	Level of number of box types	Number of box types
Exp_01	2	3	22	Exp_16	2	1	3
Exp_02	1	3	48	Exp_17	1	2	13
Exp_03	2	3	27	Exp_18	1	4	68
Exp_04	2	2	11	Exp_19	1	1	2
Exp_05	1	1	3	Exp_20	1	1	4
Exp_06	1	2	7	Exp_21	2	4	68
Exp_07	1	2	7	Exp_22	1	3	33
Exp_08	2	1	4	Exp_23	2	1	2
Exp_09	1	1	4	Exp_24	1	2	10
Exp_10	2	4	54	Exp_25	1	1	5
Exp_11	1	3	36	Exp_26	2	2	18
Exp_12	2	1	3	Exp_27	2	4	82
Exp_13	2	2	10	Exp_28	2	3	23
Exp_14	1	4	89	Exp_29	2	1	5
Exp_15	1	2	17	Exp_30	1	2	8

Table 3. Recommended GA parameter settings

Parameter	Container Selection	Container Loading
Representation	0 to 10 (Integer)	-1000 to 1000 (Real Number)
Chromosome Length	1 to 10	18
Population Size	50	60
Genetic Operators	Cloning, One-Point Crossover, Mutation	Cloning, Random Crossover, Mutation
Percent Cloning	0.10	0.10
Crossover Rate	0.90	0.90
Mutation Rate	0.10	0.10
Maximum Generation	1000	10, 20, 30, 40, 50, 100, 200, 300, 500, 1000

4.4 Analysis of Computational Time of the Container Loading GA

The computational time of the container loading GA can be effected by the efficiency of both the GA itself and the loading process. Using the thirty test problems, we conducted an experiment to test for the factors that affect the computation time of the loading process. Applying the previously recommended values of the GA parameters at the optimization level 10, Table 4 presents the computational time of each test run.

We found that the computational times are quite reasonable and practically acceptable, and that they are significantly affected by the number of the different box types (shown in the second column) rather than by the number of loaded boxes as shown by the high runtime in "Exp_02", "Exp_14", "Exp_18", and "Exp_27." For the number of boxes, no significant effect can be found in the experiment.

The existence of loading constraints, except the stack stability constraint, did reduce the runtime of the algorithm. An explanation of this result could be that the box orientation constraint reduces the possible orientations of boxes in consideration, while the stack priority constraint reduces the number of loading points in the point set and thus, in turn, the computational time.

4.5 Comparative Evaluation

To evaluate the performance of the proposed algorithm, we also conducted a comparison test using the test data available in the literature. For the container loading GA, we used one test problem found in Gehring *et al.* (1990), which proposed an effective algorithm based on the layer loading concept, and the same fifteen test problems found in Loh and Nee (1992) and Ngoi *et al.* (1994). For the combined container selection and loading, we used the two test problems of Laotaweesub (1996).

Gehring *et al.* (1990)'s problem consists of 21 boxes of 19 different box types required to be loaded in a 20-foot standard container. Each box can have any orientation. No loading constraints were considered, except that the stack stability for our algorithm was specified to 85% for both X and Z support parameters. Since the GA is stochastic by nature, and each run may not produce the same result, we ran the algorithm ten times and reported the best, the average, and the worst results, together with the average runtime. At the optimization level 5, i.e. population size of 60 and maximum number of generations of 50, the obtained results are shown in Table 5. It is interesting to note that,

Table 4. Analysis of the computational time

Experimental problems	No. of box types	Total No. of boxes	With no loading constraint			With loading constraints		
			Vol.Util.(%)	Loaded boxes	CPU time(sec)	Vol.Util.(%)	Loaded boxes	CPU time(sec)
Exp_01	22	97	83.24	51	11.51	80.04	42	8.55
Exp_02	48	280	73.72	208	191.27	72.28	200	111.74
Exp_03	27	140	85.57	80	24.84	81.38	88	14.34
Exp_04	11	89	83.49	53	5.65	77.90	54	3.88
Exp_05	3	196	82.04	127	11.19	82.04	127	8.33
Exp_06	7	97	79.99	78	5.85	80.81	73	4.32
Exp_07	7	121	80.46	96	10.50	75.24	104	8.39
Exp_08	4	174	84.25	120	14.67	79.10	134	9.10
Exp_09	4	266	86.58	205	29.35	84.85	222	21.73
Exp_10	54	162	85.81	86	44.54	82.14	64	32.45
Exp_11	36	211	75.91	138	93.61	72.13	138	60.21
Exp_12	3	170	91.74	89	6.28	90.14	86	5.69
Exp_13	10	140	85.00	100	15.57	84.72	98	7.60
Exp_14	89	252	74.21	190	283.20	71.31	179	158.20
Exp_15	17	296	79.67	227	68.40	76.41	226	47.61
Exp_16	3	175	89.45	101	9.04	88.97	104	7.63
Exp_17	13	226	83.32	189	50.08	82.24	173	36.50
Exp_18	68	197	76.46	134	131.06	73.18	125	80.73
Exp_19	2	192	83.75	171	14.75	74.29	134	8.42
Exp_20	4	291	85.96	242	39.21	85.82	233	26.21
Exp_21	68	185	84.99	78	82.38	83.05	77	50.30
Exp_22	33	205	91.11	21	7.96	87.92	28	5.52
Exp_23	2	103	79.87	73	5.85	79.57	96	4.11
Exp_24	10	162	79.97	100	19.21	78.05	108	15.20
Exp_25	5	277	86.07	158	28.30	84.38	186	23.03
Exp_26	18	193	84.08	74	23.91	82.92	86	14.06
Exp_27	82	252	83.57	118	147.29	82.21	92	93.85
Exp_28	23	116	82.34	50	15.08	77.83	44	9.23
Exp_29	5	207	86.11	84	13.17	84.23	78	7.10
Exp_30	8	140	83.39	81	14.74	79.06	106	9.55

when the stack stability was increased to 95%, all boxes could still be loaded.

Each test problem of Loh and Nee (1992) or Ngoi *et al*. (1994) consists of 100-250 boxes with six-ten different box types. The box orientation constraint was considered. Our system was run with the orientation constraint and stack stability constraint with X and Z support parameters of 85% at the optimization level 5—the population size of 60 with maximum number of generations of 50. Note that, in Table 6, we report the results and comparisons of the volume utilization of the container in both Ngoi *et al*. (1994) and our proposed system, but the packing density as published in Loh and Nee (1992). The packing density was defined as the total volume of loaded boxes expressed as the percentage of the volume of the "smallest rectangular envelope" enclosing those boxes.

From the results shown in Table 6, our Container loading GA performs as well as or even better than the references. For those problems in which no boxes remain, the GA loading also achieves the same results. We paid special attention to the test data sets 2, 6, 7, and 13, in which there were some leftover boxes after loading. It can be noted that our proposed algorithm provided better results both in terms of the volume utilization and the number of remaining boxes, except in test problem 6, in which we could achieve only better volume utilization. This situation is normal, due to the different box sizes. When we even further challenged our GA, by reducing the optimization level to 1—the population size of 60 with maximum number of generations of 10, our system could still perform best among the three, as shown in Table 7.

Table 5. A Comparison with Gehring *et al*. (1990)

Comparison problems	Gehring *et al*. (1990)		The proposed algorithm					
	Vol. Util. (%)	Leftover boxes	Vol. Util.(%)			Leftover boxes*	CPU time**	CPU time per generation
			Best	Avergae	Worst			
Gehring *et al*. (1990)	82.38	0	82.38	82.38	82.38	.	5.82	0.12

Notes: * Leftover boxes are of the best solution.
** CPU time is the average of 10 runs. (Unit: sec.)

Table 6. Comparisons with Ngoi et al. (1994) and Loh and Nee (1992) at the Optimization Level 5

Comparison problems	Ngoi *et al*. (1994)		Loh and Nee (1992)		The proposed algorithm					
	Vol. Util. (%)	Leftover boxes	Packing Density(%)	Leftover boxes	Vol. Util. (%)			Leftover boxes*	CPU time**	CPU time per generation
					Best	Avergae	Worst			
Test data set # 1	62.50	0	78.12	0	62.50	62.50	62.50	0	28.49	0.57
Test data set # 2	80.73	54	76.77	32	91.32	89.08	86.14	31	77.35	1.55
Test data set # 3	53.43	0	69.46	0	53.43	53.43	53.43	0	103.90	2.08
Test data set # 4	54.96	0	77.57	0	54.96	54.96	54.96	0	28.53	0.57
Test data set # 5	77.19	0	85.79	1	77.19	77.19	77.19	0	36.93	0.74
Test data set # 6	88.72	48	88.55	45	90.50	89.22	87.76	52	66.91	1.34
Test data set # 7	81.81	10	78.17	21	84.66	83.99	82.98	0	102.52	2.05
Test data set # 8	59.42	0	67.58	7	59.42	59.42	59.42	0	49.19	0.98
Test data set # 9	61.89	0	84.22	0	61.89	61.89	61.89	0	116.08	2.32
Test data set # 10	67.29	0	70.10	0	67.29	67.29	67.29	0	162.61	3.25
Test data set # 11	62.16	0	65.44	0	62.16	62.16	62.16	0	27.94	0.56
Test data set # 12	78.52	0	79.33	0	78.52	78.52	78.52	0	31.69	0.63
Test data set # 13	84.14	2	77.03	15	85.61	84.90	84.04	2	38.39	0.77
Test data set # 14	62.81	0	69.09	0	62.81	62.81	62.81	0	39.96	0.80
Test data set # 15	59.46	0	73.56	0	59.46	59.46	59.46	0	169.79	3.40

Notes: * Leftover boxes are of the best solution.
** CPU time is an average of 10 runs. (Unit: sec.)

Table 7. Comparisons with Ngoi *et al*. (1994) and Loh and Nee (1992) at the Optimization Level 1

Comparison problems	Ngoi *et al*. (1994)		Loh and Nee (1992)		The proposed algorithm					
	Vol. Util. (%)	Leftover boxes	Packing Density(%)	Leftover boxes	Vol. Util.(%)			Leftover boxes*	CPU time**	CPU time per generation
					Best	Avergae	Worst			
Test data set # 1	62.50	0	78.12	0	62.50	62.50	62.50	0	6.07	0.61
Test data set # 2	80.73	54	76.77	32	90.42	86.55	84.61	34	14.29	1.43
Test data set # 3	53.43	0	69.46	0	53.43	53.43	53.43	0	20.86	2.09
Test data set # 4	54.96	0	77.57	0	54.96	54.96	54.96	0	6.53	0.65
Test data set # 5	77.19	0	85.79	1	77.19	77.19	77.19	0	7.09	0.71
Test data set # 6	88.72	48	88.55	45	89.44	88.23	86.75	46	14.46	1.45
Test data set # 7	81.81	10	78.17	21	84.00	83.16	81.24	2	19.10	1.91
Test data set # 8	59.42	0	67.58	7	59.42	59.42	59.42	0	9.39	0.94
Test data set # 9	61.89	0	84.22	0	61.89	61.89	61.89	0	24.02	2.40
Test data set # 10	67.29	0	70.10	0	67.29	67.29	67.29	0	32.95	3.30
Test data set # 11	62.16	0	65.44	0	62.16	62.16	62.16	0	5.75	0.57
Test data set # 12	78.52	0	79.33	0	78.52	78.52	78.52	0	6.28	0.63
Test data set # 13	84.14	2	77.03	15	85.09	84.51	84.04	2	7.80	0.78
Test data set # 14	62.81	0	69.09	0	62.81	62.81	62.81	0	8.76	0.88
Test data set # 15	59.46	0	73.56	0	59.46	59.46	59.46	0	37.59	3.76

Notes: * Leftover boxes are of the best solution.
** CPU time is an average of 10 runs. (Unit: sec.)

Table 8. Comparisons with Laotaweesub (1996)

Comparison Problems	Laotaweesub (1996)			"The Proposed System			
	Selected container	Vol. Util. (%)	Freight rate (US$)	Selected container	Vol. Util. (%)	Freight rate (US$)	CPU time (sec.)
Problem #1	40 FCL	85.09	650.00	40 FCL	86.20	650.00	
	40 FCL	72.46	650.00	40 FCL	73.13	650.00	
	20 FCL	54.49	450.00	20 FCL	50.92	450.00	
Total			1750.00			1750.00	7.10

Comparison Problems	Laotaweesub (1996)			"The Proposed System			
	Selected container	Vol. Util. (%)	Freight rate (US$)	Selected container	Vol. Util. (%)	Freight rate (US$)	CPU time (xec.)
Problem #2	40 FCL	80.60	3500.00	40 FCL	86.19	3500.00	
	40 FCL	77.25	3500.00	40 FCL	86.91	3500.00	
	40 FCL	66.36	3500.00	40 FCL	84.68	3500.00	
	40 FCL	66.16	3500.00	40 FCL	52.29	3500.00	
	20 FCL	48.12	3500.00				
	20 FCL	12.74	3500.00				
Total			18000.00			14000.00	88.67

The two real-world problems found in Laotaweesub (1996), on the other hand, are different from the container loading problems above, in that they also dealt with the container selection problem, minimizing the total transportation cost. The first problem has 95 boxes of 21 different types, which need to be transported to a destination using two types of containers. The freight rate of each container type was given and considered in the container selection problem, while the container stability and one uppermost face for every box were considered in the container loading algorithm. The second problem has 150 boxes of four types of materials that imply stack priority and two types of containers. Other problem characteristics are the same as those of the first problem.

The results of comparison as summarized in Table 8 showed that our integrated system provided the solution at least as good as Laotaweesub (1996). In Problem 1, the same set of 3 containers summarized in the upper table (2 of 40FCL type and 1 of 20FCL) was selected, quoting the same amount of transportation cost. However, our proposed algorithm prescribed higher volume utilizations in the first two containers (86.20% and 73.13%), leaving more space in the last container for the exporter to stow additional merchandise, if desired. In Problem 2, on the other hand, we could achieve a lower transportation cost summarized in the lower table ($14,000 compared with $18,000), using only 4 containers of 40FCL. The lower transportation cost was achieved mainly due to a higher volume utilization of each loaded container.

5. CONCLUSIONS AND FURTHER RESEARCH

We propose an integrated heuristic approach based on genetic algorithms to solve the interrelated container selection and loading problems. The overall procedure is iterative, first selecting a set of containers that minimizes the shipment cost in the GA container selection algorithm, and subsequently loading boxes into each container one at a time by the Container loading GA algorithm. If the volume (weight) of containers is found insufficient after loading, the minimum required volume (weight) is increased by that of the remaining boxes, and supplied into the container selection module to solve for another set of containers. Some containers may be unloaded, while reloading begins only for the newly selected containers. Practical aspects namely the box orientation, the stack priority, the stack stability and the container stability may be included in the loading process if the user so desires.

When compared with other works, our container loading GA as well as the integrated system provides solutions that are at least as good, and better in several cases in terms of volume utilization and cost. In general, the proposed integrated system was proved experimentally to be superior to other comparative approaches, and practical for real-world applications in dealing with other loading restrictions.

Further research may attempt to improve this algorithm by a more appropriate container sequence and initial loading position. The effect of the container sequence to load on the number of containers used and, in turn, the total transportation cost may be studied, leading to an algorithm prescribing an optimal container sequence. A different initial loading position may be studied to improve the volume utilization. Further research may also include special restrictions that specifically exist in other container loading applications such as air cargo, in which, due to the non-rectangular shape of the container, boxes may be loaded without full-base support called wing packing. In addition, due to the inertia that exists during the take off or landing of a plane, the gap behind boxes is usually not allowed or may be allowed with some minor tolerances, since the boxes may be tilted and relocated, destroying goods inside. Other research may focus on the non-rectangular shapes of containers.

REFERENCES

Bischoff, E.E., Marriott, M.D. (1990) A comparative evaluation of heuristics for container loading, *European Journal of Operational Research* **44**, 267-276.

Bischoff, E.E., Ratcliff, M.S.W. (1995a) Issues in the development of approaches to container loading. Omega **23**, 377–390.

Bischoff, E.E., Ratcliff, M.S.W. (1995b) Loading multiple pallets. *Journal of the Operational Research Society* **46,** 1322–1336.

Bortfeldt, A., Gehring H. (1998) Applying tabu search to container loading problems. In: Operations Research Proceedings 1997, Springer, Berlin, 533–538.

Bortfeldt, A., Gehring H. (2001) A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research* **131,** 141-161.

Chen, C.S., Lee, S.M., Shen, Q.S. (1995) An analytical model for the container loading problem, *European Journal of Operational Research* **80**, 68-76.

Davies, A.P., Bischoff, E.E. (1999) Weight distribution considerations in container loading, *European Journal of Operational Research* **114**, 509-527.

Dyckhoff H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research* **44**, 145–159.

Gehring, H., Menschner, K., Meyer, M. (1990) A computer-based heuristic for packing pooled shipment containers, *European Journal of Operational Research* **44**, 277-289.

George, J.A., Robinson, D.F. (1980). A heuristic for packing boxes into a container, *Computers and Operational Research* **7**, 147-156.

Goldberg, D. (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading, MA.

Kawakami, T., Minagawa, M., Kakazu, Y. (1991) Auto tuning of 3D packing rules using genetic algorithms, IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS'91, Nov. 3-5, 1319-1324.

Laotaweesub, S. (1996) The development of a container selection and loading system, Thesis: No. ISE-96-18, Asian Institute of Technology, Thailand.

Ngoi, B.K.A., Tay, M.L., Chua, E.S. (1994) Applying spatial representation techniques to the container packing problem, *International Journal of Production Research* **32**, 111-123.

Pisinger, D. (2002) Heuristics for the container loading problem, European Journal of Operational Research **141**, 382-392.

Tsai, R.D., Malstrom, E.M., Kuo, W. (1993) Three dimensional palletization of mixed box sizes, *IIE Transactions* **25**, 64-75