Three-dimensional Container Loading Using a Co-operative Co-evolutionary Genetic Algorithm

Chaiwat Pimpawat
Department of Industrial Engineering
King Mongkut's Institute of Technology North Bangkok
Bangkok
Thailand 10800

Nachol Chaiyaratana
Research and Development Center for Intelligent Systems
King Mongkut's Institute of Technology North Bangkok
Bangkok
Thailand 10800

Abbreviated Title: 3D Container Loading Using a CCGA

Corresponding Author
Dr. Nachol Chaiyaratana
Research and Development Center for Intelligent Systems,
Faculty of Engineering,
King Mongkut's Institute of Technology North Bangkok,
1518 Piboolsongkram Road, Bangsue,
Bangkok 10800,
THAILAND
Tel: +66 (0)2 9132500 Ext. 8208
Fax: +66 (0)2 5870029
E-mail:nchl@kmitnb.ac.th, nachol1@go.com

Abstract

This paper presents the use of a co-operative co-evolutionary genetic algorithm (CCGA) in conjunction with a heuristic rule for solving a 3D container loading or bin packing problem. Unlike previous works which concentrate on using either a heuristic rule or an optimisation technique to find an optimal sequence of packages which must be loaded into the containers, the proposed heuristic rule is used to partition the entire loading sequence into a number of shorter sequences. Each partitioned sequence is then represented by a species member in the CCGA search. The simulation results indicate that the use of the heuristic rule and the CCGA is highly efficient in terms of the compactness of packages in comparison to the results given by a standard genetic algorithm search. In addition, this helps to confirm that the CCGA is also suitable for use in a sequence-based optimisation problem.

Keywords: Bin Packing Problem, Co-evolution, Container Loading Problem, Co-operative Co-evolutionary Genetic Algorithm

1. Introduction

Generally, once the products have been manufactured and packaged, the processes that follow would be shipment loading and transportation. During one shipment, a number of containers will be used to hold all sorts of packages, depending upon the shipment order. Usually, the packages themselves would be in different sizes and shapes. Hence, it is highly desirable to be able to load the containers with these packages in such a way that there are as little empty spaces as possible. Such a problem is commonly referred to as a container loading problem or a bin packing problem.

A number of research works have been carried out in order to find a procedure that leads to an optimal way of loading containers. The development in this research area can be roughly divided into two main categories: an algorithmic design which leads to a generalised loading heuristic rule (Gehring et al., 1990; Dowsland, 1991; Mohanty et al., 1994; Chen et al., 1995) and a customised search which results in a specific way of loading the containers for one particular shipment order (Corcoran and Wainwright, 1992; Kröger, 1995). In the case of the algorithmic design approach, expert knowledge is usually required in order to build a knowledge base where a heuristic rule can be derived. Although the resulting heuristic rule is proven to be adequate to a certain extent, it is rarely flexible enough to cope with the current trend of the loading problem in which the shipment order is the result of a mass customisation process. On the contrary, a customised search can result in a solution that can be regarded as an optimal solution for one specific shipment order. This will certainly be a suitable method that follows the philosophy of mass customisation.

The techniques that can be used to carry out such customised search for an optimal way of loading containers fall into the category of non-linear optimisation techniques. Although a number of optimisation techniques have been explored in this problem area, in recent years much attention has concentrated on evolutionary computation techniques. For

instance, Kröger (1995) uses a simple genetic algorithm to solve a two-dimensional loading problem. In contrast, Corcoran and Wainwright (1992) have used a simple genetic algorithm to solve a three-dimensional version of the problem. Although some successful results have been reported in Corcoran and Wainwright (1992), it can be clearly seen that as the problem size increases, the efficiency of the genetic algorithm search will sharply decrease. One possible way to reduce the effect of the problem size on the search performance is to divide the loading problem into a number of sub-problems where the search technique can be applied to each sub-problem. One particular evolutionary computation technique which will be explored here in an attempt to find an optimal solution to a partitioned container loading problem is the co-operative co-evolutionary genetic algorithm (CCGA).

The reason that the CCGA is chosen for this particular problem is because the CCGA has a unique characteristic in being able to deal with a problem by assembling a solution to the overall problem from a number of sub-components of the complete solution. For this particular problem, provided that the loaded packages can be categorised into a number of groups according to their sizes and shapes, the process of loading a group of packages can be treated as a sub-problem of the overall container loading problem. With this arrangement, the solution to each sub-problem can be viewed as a sub-component which can be assembled into a complete solution. This treatment on the container loading problem makes the CCGA a suitable optimisation technique for the problem.

The organisation of this paper is as follows. In section 2, the container loading problem considered will be explained. This also includes the description on how the container loading problem is divided into a number of sub-problems. In section 3, the background on the CCGA will be given. In addition, the explanation on the modifications made on the original CCGA for this problem will also be included. The use of the CCGA in solving the container loading problem will be given in section 4. The simulation results obtained from

using the CCGA to solve the problem are shown in section 5. The results from using a standard genetic algorithm to solve the container loading problem are also included in this section in order to compare with the CCGA performances. Discussions on the simulation results are given in section 6. Finally, conclusions are drawn in section 7.

2. Container Loading Problem

As mentioned earlier, a container loading problem arises when a number of packages with different sizes and shapes are required to be packed into a set of containers. In brief, the assumptions regarding the container loading problem which will be explored in this paper can be described as follows.

1. All packages have rectangular shapes and are roughly divided into three groups where the packages that belong to the same group have similar sizes. These three package groups are small-sized, medium-sized and large-sized packages.
2. The containers also have rectangular shapes where they can hold a number of packages provided that the maximum capacity limit is not exceeded.
3. The packing scheme considered can be described as being orthogonal and guillotineable. A packing arrangement is said to be orthogonal when the sides of a package are always parallel with the sides of other neighbouring packages or those of the container. On the other hand, a packing scheme is said to be guillotineable if and only if it can be created by recursively bi-partitioning the container volume with straight guillotineable cuts. Each of these cuts separates a rectangular volume within the container volume into two likewise rectangular pieces. Figure 1 presents two orthogonal packing schemes: a non-guillotineable one (a) and a guillotineable one (b).
4. It is possible to arrange the packages into a number of layers within each container where each layer of packages will be in a left-bottom justified format. In other words, it is assumed that there is a horizontal boundary hyper-plane between two consecutive layers

of packages. This is illustrated in Figure 2. In Figure 2, a container is filled with packages which have been put into three layers where the packages in each layer are arranged such that the left-most package has one of its sides coinciding with the left wall of the container. There are two horizontal boundary hyper-planes in this case. Notice that the package number 7 is allowed to be placed on top of the package number 6. The maximum allowable height of each package stack is set to the height of the tallest package in the loading sequence that has not yet been loaded into the container. Note that in practice, the gaps between a boundary hyper-plane and packages will be filled with supporting materials.

5. Within the same layer of packages stored inside the container, the packages will be arranged into a front-left justified format. Such a layout format is illustrated in Figure 3. In Figure 3, there are eight packages in the same layer where they are put into three rows. The first package is placed at the front left corner of the container. Then the second and the third packages are placed in the remaining area where one of their sides coinciding with the front wall of the container. Since it is not possible to place the fourth package next to the third one, the fourth package is placed on the new row with one of its sides coinciding with the left wall of the container. This filling process is continued until it is no longer possible to arrange packages into the same layer.
6. The loading process will start from the loading of the large-sized packages. After all large-sized packages have been loaded, the loading process will continue with the loading of medium-sized packages and small-sized packages, respectively. A diagram depicting a cross-sectional view of a container loaded with all three groups of packages is illustrated in Figure 4. In Figure 4, three layers of packages are illustrated. In the bottom layer, all packages are in the large-sized group. In contrast, the second layer of packages is made up from large-sized and medium-sized packages. Similar to the case of the second layer,

packages of different sizes are placed in the same layer at the top of the container; this top layer contains medium-sized and small-sized packages. It can be clearly seen that the packages are loaded in sequence without any interruption by starting from the large-sized packages and followed by the medium-sized and small-sized packages. Note that the loading process will continue until all packages are loaded into the containers. With this form of loading process, the complete loading sequence can be viewed as a series of three loading sequences being joined together. In addition, any attempts on reshuffling the orders of packages from the same package group in a sequence can be viewed as a sequencing problem. This reduces the container loading problem into three sequence-based sub-problems. With this treatment on the loading process, the problem considered has been transformed into a simplified problem which has a structure suitable for the optimisation using the CCGA. The background on the CCGA and how it can be applied to the container loading problem will be given in the following section.

3. Co-operative Co-evolutionary Genetic Algorithm

The co-operative co-evolutionary genetic algorithm (CCGA) was first introduced by Potter and De Jong (1994). The CCGA functions by introducing an explicit notion of modularity to the optimisation process. This is done in order to provide reasonable opportunity for complex solutions to evolve in the form of interacting co-adapted sub-components. In brief, the CCGA explores the search space by utilising a population which contains a number of species or sub-populations. In contrast to other types of genetic algorithms, each species in the CCGA represents a variable or a part of the problem which is optimised. A complete solution to the problem is a combination of individuals, one from each species, whose fitness can be calculated. This value of fitness will be assigned to the individual of interest which participates in the formation of the solution. After the fitness values have been assigned to all individuals, the evolution of each species is then conducted by a standard genetic algorithm.

The CCGA has been successfully used in a number of problem domains including multivariable function optimisation (Potter and De Jong, 1994), sequential rule evolution (De Jong and Potter, 1995; Potter et al., 1995) and design of a cascade neural network (Potter and De Jong, 1995). A comprehensive description of the CCGA and a summary of its applications can be found in Potter and De Jong (2000). Although the CCGA has been successfully used in various applications, its performance can reduce in the circumstance where there are high interdependencies between the optimisation function variables. In order to solve this problem, Potter and De Jong (1994) have suggested a strategy which involves a random combination between individuals from all species during the fitness evaluation process. This helps to increase the chance for each individual to achieve a fitness value which is appropriate to its contribution to the solution produced. In this paper, another strategy for solving the same problem is introduced. The explanation of this strategy and other modifications to the CCGA are given as follows.

Similar to the original CCGA, the modified CCGA used in this paper explores the solution space by means of utilising a number of species or sub-populations where each individual in a species represents a component of a complete solution. In the first generation of the original CCGA, the fitness value of an individual is found from a complete string which is the result of the combination between the individual of interest and other individuals from different species which are being selected at random. In contrast, in the first generation of the modified CCGA, all possible combinations between all individuals from different species are explored. This means that there will be a number of fitness values which are calculated for each individual since an individual will participate in more than one solution. In the modified CCGA, the highest fitness value from this set of fitness values is chosen to be the fitness value assigned to the individual. For example, if the population considered is composed of M sub-populations where each sub-population contains N individuals, the total

number of solutions that can be created is equal to N^M. Among these N^M solutions, an individual will participate in the construction of $N^{(M-1)}$ solutions. The highest fitness value from the set of $N^{(M-1)}$ solutions will be assigned to the individual which is a part of all solutions in this set. After the fitness value has been assigned to each individual, the processes of fitness scaling, selection, crossover and mutation are carried out within each sub-population. These are done in a similar way to the processes taking place in the original CCGA. This is followed by a new set of the complete solutions being created by combining all offspring individuals from different species with the best parent individual previously obtained. At this point, a new fitness assignment strategy will be introduced. Referring to the earlier example, at this point M best parent individuals from different species are available. Each of these parent individuals will be combined with all offspring individuals that belong to different species. This means that there will be a total of $M \times N^{(M-1)}$ solutions being created. Among these solutions, $N^{(M-1)}$ solutions have a common part which contains the same elite parent individual from one species. In contrast, the same part of the chromosomes in the remaining $(M\text{-}1) \times N^{(M-1)}$ solutions will be replaced by an offspring individual that comes from the same species as that particular elite parent individual. In other words, each offspring individual will be a part of $(M\text{-}1) \times N^{(M-2)}$ different solutions in this case. Again, the highest fitness value in the set of $(M\text{-}1) \times N^{(M-2)}$ solutions will be assigned to the individual which contributes to the construction of all these solutions. An example of this new fitness calculation strategy is illustrated in Figure 5. In Figure 5, the entire population is made up from three species where each species contains ten individuals. The total number of fitness evaluations performed in the first generation is equal to $10^3 = 1{,}000$. Among 1,000 possible solutions explored, each individual will participate in the construction of 100 solutions. In contrast, the total number of fitness evaluations performed in each subsequent generation is equal to $3 \times 10^{(3-1)} = 300$. Among these solutions, each one-third portion of the solution set

will contain the best individual from sub-populations *A, B* and *C* in the previous generation, respectively. It can be clearly seen that each individual from the current generation will be present in $(3\text{-}1) \times 10^{(3\text{-}2)} = 20$ distinct solutions. For example, 20 solutions that contain the first individual in the current sub-population *C* can be found in the first and the second portions of the entire set where there will be 10 solutions in each portion.

In addition to a new strategy for the fitness assignment procedure, a scheme involving the control of gene pool diversity is also embedded into the modified CCGA; this scheme is explained as follows. Provided that the number of offspring individuals is equal to that of parent individuals, two equal-sized populations where each individual possesses the appropriate fitness value are available at this point in the genetic algorithm run. The parent sub-population and the offspring sub-population of the same species can then be merged together and the appropriate individuals of the new generation can be extracted from this merged sub-population. Note that the number of individuals extracted is equal to the size of the sub-population. The strategy used in this case for the extraction process is based on the mechanisms of a diversity control oriented genetic algorithm or DCGA (Shimodaira, 1997). In brief, the process starts with the elimination of duplicate individuals in the merged sub-population. Since the fitness values among the duplicate individuals may be different from one another because the fitness values obtained are the results from the evaluation in either the present or previous generation, the individuals deleted will be the less fitness ones. The remaining individuals are then sorted according to their fitness values in descending order. Following that the best individual from the remaining individuals is determined and kept for passing on to the next generation. Then a cross-generational probabilistic survival selection (CPSS) method is applied to the remaining non-elite individuals where a probability value is assigned to each individual according to its similarity to the best individual in that particular sub-population. If the genomic structure of an individual is very close to that of the best

individual, the survival probability assigned to this individual will be close to zero. On the other hand, if the chromosome structure of this individual is quite different from that of the best individual, its survival probability will be closed to one. Each individual will then be selected according to the assigned survival probability. If the total number of all selected individuals including the pre-selected elite individual does not reach the required sub-population size after the first survival selection loop, the process of survival selection from the previously unselected individuals will be repeated until the required number is met. Note that the process regarding the diversity control scheme will be repeated for all sub-populations. The strategy described above will help maintaining a high level of gene pool diversity within each sub-population (Shimodaira, 1997). After the extraction process is completed, the selected individuals will later be used as the parent individuals for the evolutionary processes in the next generation.

The pseudo code of the modified CCGA described above is given in Figure 6. From Figure 6, the main differences between the modified CCGA presented here and the original CCGA are the strategy used to maintain the diversity level in the gene pool and the way fitness value is assigned to each individual. In the original CCGA, no considerations regarding the diversity control within the population have been mentioned. For the fitness assignment issue, in the original CCGA fitness of a species member is obtained after either combining it with the current best individuals or the randomly selected individuals from the remaining species depending upon whether which combination yields a higher fitness value. It can be clearly seen that the fitness value of each individual in the original CCGA is the result of a selection between two possible values while in the modified CCGA the fitness value of each individual has to be selected from a much larger set. The original CCGA would have an advantage in the merit of the computational complexity since the strategy used for the fitness evaluation is much simpler than the one in the modified CCGA. The fitness

evaluation strategy used in the modified CCGA would be impractical if the population contains a large number of species. In contrast, the strategy used in the modified CCGA yields an advantage in the merit of the certainty in the fitness value assigned to each individual. Since only two trial fitness values are considered during the fitness evaluation of each individual in the original CCGA, the fitness value assigned to the individual may not be the most appropriate value. On the contrary, in the modified CCGA the fitness value assigned to each individual has to be chosen from a relatively larger set. This may help increasing the level of confidence in the suitability of the fitness value assigned to the individual.

4. Application of the CCGA on the Container Loading Problem

In this section, the use of the modified CCGA explained earlier in section 3 on the container loading problem will be discussed. As mentioned earlier, the packages considered for the loading process can be divided into a number of groups according to the sizes of the packages. In this investigation, the packages will be divided into three categories: small-sized, medium-sized and large-sized packages. Note that it is also assumed that the large-sized packages will be loaded first. Following that, the medium-sized packages will be loaded next. The small-sized packages will be loaded into the containers only when all other large-sized and medium-sized packages have been loaded. By partitioning the entire loading sequence into three shorter sequences, the problem has been formulated in the manner that is suitable for the optimisation using a CCGA. Details on the problem formulation and genetic operators used are discussed as follows.

4.1. Decision Variables

The decision variables in this case are made up from the orders of the packages in the loading sequence. Since the entire loading sequence has been partitioned into three separate sequences – one sequence for each group of packages – the decision variables will also be divided into three groups. The decision variables that belong to the same group will

subsequently be coded into an individual in a sub-population. In this paper, the packages within each package group can be further classified into a number of different types. Two test examples are investigated here; the physical dimensions of each type of packages and the quantity of packages used in the first and second examples are displayed in Tables 1 and 2, respectively. From Table 1, there are 205 packages in the loading process considered. These packages are sorted according to their height in descending order and divided into three groups where the numbers of packages in the large-sized, medium-sized and small-sized package groups are 75, 65 and 65, respectively. Similar to the first example, the 220 packages shown in Table 2 are also divided into three package groups. The numbers of large-sized, medium-sized and small-sized packages are equal to 75, 80 and 65, respectively in the second example. Note that the sizes of nearly all packages in the second example are smaller than that in the first example. For each example, all packages will be loaded into a number of containers where each container has a dimension of 1,170 mm (width) $\times$ 2,960 mm (length) $\times$ 1,190 mm (height). The loading process will begin with the sequence of large-sized packages, follow by the sequence of medium-sized packages and terminate with the sequence of small-sized packages. These three sequences will be concatenated together and treated as one uninterrupted sequence. An attempt on loading the packages according to their orders in the complete sequence into a container will be pursued until it is no longer possible to load any more packages into the container. The package at this interrupted point in the sequence will be the first package being loaded into the next container. Note that the loading process will be carried out according to the assumptions given in section 2. The total number of containers required and the waste space within the last container will be used for the calculation of fitness; this will be discussed in the following section.

4.2. Fitness Function

The objective of the search is to find the loading sequence that gives optimal compactness of the packages. This can be achieved by considering two measurement criteria: the number of containers required and the empty volume of the last container. The interesting part of the empty volume in the last container is schematically displayed in Figure 7. In Figure 7, the main portion of the empty volume in the container that can be used for storing more packages resides in the spaces marked A and B. The combined volume of these two spaces is used in the fitness calculation. The fitness function for the search can be defined as

$$f_i = w_1\left[\frac{C_{\max} - C_i}{C_{\max} - C_{\min}}\right] + w_2\left[\frac{V_i - V_{\min}}{V_{\max} - V_{\min}}\right] \quad (1)$$

where f_i is the fitness of individual i, $C_{\max}$ is the maximum possible number of the containers which is set to ten, $C_{\min}$ is the minimum possible number of containers which is set to three, $V_{\max}$ is the maximum possible empty volume in the container which is equal to the volume of the container itself, $V_{\min}$ is the minimum possible empty volume which is set to zero, C_i is the number of containers required in the case of the ith individual, V_i is the empty volume in the case of the ith individual, and w_1 and w_2 denote weighting factors. In this investigation, w_1 is set to 0.9 while w_2 is set to 0.1. These parameter settings are used so that the performance criterion that has a higher priority in this case is the number of containers used.

4.3. Chromosome Coding

The chromosome of each individual represents the orders of the packages in the loading sequence. In this investigation, an integer-based coding scheme is utilised. Each gene on the chromosome denotes the type of the package while the gene locus represents the package order in the sequence. In this paper, the entire population is made up from three sub-populations. By combining an individual from all sub-populations in the first example, the total sequence would have the length of 205. This leads to an overall search space of

approximately 9.67×10^{107} $\left(\left(\frac{75!}{15!20!30!10!}\right)\left(\frac{65!}{15!10!25!15!}\right)\left(\frac{65!}{20!15!25!5!}\right)\right)$ search points. In contrast, the length of loading sequence in the second example is equal to 220 where the corresponding search space contains approximately 4.14×10^{115} $\left(\left(\frac{75!}{15!20!30!10!}\right)\left(\frac{80!}{15!10!25!30!}\right)\left(\frac{65!}{20!15!25!5!}\right)\right)$ search points.

4.4. Fitness Scaling and Selection Methods

A linear fitness scaling technique discussed in Goldberg (1989) is used in the modified CCGA in order to improve the search performance. After the scaling procedure, a stochastic universal sampling (Baker, 1989) is used in the fitness selection. Note that the elitist strategy used is to select one individual with the highest fitness and pass this individual on to the offspring individual set without performing any crossover or mutation operations.

4.5. Crossover and Mutation Methods

Since the problem involves the reordering of the packages in the sequence is a sequencing problem, some standard crossover and mutation techniques which are designed for use in a travelling salesman problem can be utilised. In this paper, a standard OX crossover technique (Davis, 1985) is used in the recombination while a reciprocal exchange operation (Herdy, 1991) is used for the mutation.

4.6. Diversity Control

As mentioned earlier in section 3, after the offspring individuals have been created from the parent individuals, these two sets of individuals will be merged together where one half of the merged set will be selected for use as the parent individual set in the following generation. The survival selection procedure begins with the elimination of all duplicated individuals and the selection of the best individual from the combined set. Then the remaining individuals are selected according to the survival probability assigned to each individual. This survival probability of an individual is calculated via the use of the similarity between the genomic

structure of the individual and that of the best individual. In this paper, a similarity between two individuals is measured by counting the number of common pairs of two adjacent genes between the two individuals. An example for this similarity measurement is illustrated in Figure 8. In Figure 8, there are three common pairs of two adjacent genes between two individuals; they are 1-1, 1-2 and 2-3 pairs. The similarity count obtained in this case is equal to three. In this investigation, the maximum possible value of the similarity count is equal to the chromosome length subtracted by one. The survival probability of an individual is given by

$$p_s = \{(1-c)(L-1-s)/(L-1)+c\}^{\alpha} \quad (2)$$

where p_s denotes the survival probability, s is the similarity count obtained for the individual, L is the chromosome length, c is the shape coefficient and α is the exponent coefficient. With this form of survival probability assignment, an individual which its genomic structure is quite different from that of the best individual will be given a high chance of survival. This will help promoting the diversity within the gene pool of the sub-population. The parameter settings for the modified CCGA are summarised in Table 3.

For the purpose of comparison, two approaches using a single population genetic algorithm to solve the loading problem are also investigated. In the first approach, the restriction involving the loading process where all large-sized packages must be loaded prior to the loading of the medium-sized packages and the small-sized package group is the last loading group is no longer held. This means that the chromosome of an individual in the standard genetic algorithm will represent a loading sequence of the packages where each package is being a part of the same package group. Without the use of the restriction imposed earlier, there will be a difference between the two search spaces that each algorithm has to explore. In this case, the size of the search space in the first example has changed to

approximately $1.56 \times 10^{203} \left(\frac{205!}{15!20!30!10!15!10!25!15!20!15!25!5!} \right)$ search points while the search space in the second example has changed to approximately $6.45 \times 10^{217} \left(\frac{220!}{15!20!30!10!15!10!25!30!20!15!25!5!} \right)$ search points. Nonetheless, this is done in order to prove the validity of the heuristic rule used to create such restriction. In contrast, the loading restriction is used in the second approach. Hence, the search spaces for the standard genetic algorithm and the modified CCGA searches are the same in this case. The comparison between the search results will be used to identify the co-operative co-evolutionary effect on the algorithm performance. Note that the parameter settings for both approaches of the standard genetic algorithm search are summarised in Table 4. The simulation results from using the modified CCGA and the standard genetic algorithm to find a suitable loading sequence will be presented in the following section.

5. Simulation Results

As mentioned earlier, two test examples are investigated in this paper. For each example, four loading formats are explored; these formats are illustrated in Figure 9. From Figure 9, in the LL and WL formats the front and the back of the container are the sides that have the edges along the length and the height of the container. In the LL format the packages are loaded into the container in the way that the edge along the length of each package and that of the container are parallel while in the WL format the edge along the width of each package is parallel to the edge along the length of the container. In contrast, in the LW and WW formats the front and the back of the container are the sides that have the edges along the width and the height of the container. The packages are arranged such that the edge along the length of each package is parallel to the edge along the width of the container in the LW format while the edge along the width of each package and that of the container are parallel in the WW format. Note that in the simulation, the layout of packages in the container must conform to

the assumptions given in section 2. Within each case study involving a loading format, there are three sets of simulation results: one set from using the modified CCGA and the other two sets from using the standard genetic algorithm. The reported results are the best results obtained after repeating each simulation run ten times using different initial populations. The simulation results in terms of the fitness value, the number of containers and the empty volume in the last container from all case studies in first example are displayed in Table 5 while the simulation results for the second example are shown in Table 6.

6. Discussions

From all case studies in both test examples, the use of the modified CCGA has been proven to be a more efficient method in locating a suitable loading sequence. It can be clearly seen that the best loading sequence found by the modified CCGA either requires the use of less number of containers or can generate more empty volume in the last container than that from both cases of the standard genetic algorithm. In other words, the loading sequence generated by the CCGA is more compacted. This superiority in the search performance of the modified CCGA stems from two factors: the heuristic rule used to creating a restriction in the loading sequence and the co-operative co-evolutionary mechanism. The heuristic rule utilised here imposes the requirement on categorising the packages into three distinct groups. In addition, it is also enforced that the loading of all large-sized packages have to take place prior to the loading of any medium-sized packages and all small-sized packages must be in the last loading set. The fact that the performances of both the genetic algorithm with a loading restriction and the modified CCGA are higher than that of the genetic algorithm without a loading restriction is a strong supporting evidence for the validity of the heuristic rule used. In addition to the suitability of the heuristic rule, the co-operative co-evolutionary effect also promotes the emergence of a fitter solution. This is concluded from the results in all case studies which clearly indicate that the solution found by the modified CCGA is better than

that found by the genetic algorithm with a loading restriction. It is also noticeable that the best loading formats in the first example are the LL and WL formats while the loading format has a lesser effect on the results in the second example. This is to be expected since most of the packages used in the second example are smaller than that in the first example. In other words, the effect of the package orientation on the layout will be less significant in the case that the package size is relatively small in comparison to the container size.

In order to validate the simulation results, all results from the repeated simulation runs covering all case studies in both test examples are subjected to a two-way ANOVA test. The sources of variation are the search algorithms and the loading formats. The ANOVA test results are illustrated in Table 7. From Table 7, the test results can be interpreted as follows.

1. The differences between the results obtained from the searches using the modified CCGA, the genetic algorithm with a loading restriction and the genetic algorithm without a loading restriction are statistically significant.
2. The differences between the results obtained using the LL, WL, LW and WW loading formats are also statistically significant.
3. There are no interactions between the two sources of variation: the search algorithm and the loading format. This reconfirms the observation that the modified CCGA produces the best loading sequence regardless of the loading format used in all case studies explored.

7. Conclusions

In this paper, a procedure for solving a 3D container loading problem has been proposed. The procedure involves the use of a heuristic rule and a genetic algorithm search. The heuristic rule used covers a scheme which involves a classification of packages into three distinct groups: large-sized, medium-sized and small-sized package groups. The introduction of this heuristic rule results in the partitioning of the entire loading sequence into three sub-sequences where each sub-sequence consists of the loading orders of packages from the same

group. With the use of this sequence partitioning scheme, a search for the best combination of three sub-sequences is then performed by utilising a co-operative co-evolutionary genetic algorithm (CCGA). The search population in this case is composed of three sub-populations where an individual in each sub-population represents a loading sequence of packages from the same group. The simulation results indicate that the search performance of the CCGA is higher than that of a standard genetic algorithm. This helps to confirm the suitability and effectiveness of the proposed heuristic rule when it is used in conjunction with a co-operative co-evolutionary search technique. In addition, this also proves that the CCGA is also suitable for use as an optimisation tool for a sequencing problem.

References

Baker, J. E. 1989. *An analysis of the effects of selection in genetic algorithms*, Ph.D. Thesis, Computer Science Department, Vanderbilt University, Nashville, TN.

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

Corcoran, A. L. III, and R. L. Wainwright. 1992. A genetic algorithm for packing in three dimensions. *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing (SAC'92)*, Kansas City, KS, 1021-1030.

Davis, L. 1985. Applying adaptive algorithms to epistatic domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, 162-164.

De Jong, K. A., and M. A. Potter. 1995. Evolving complex structures via cooperative coevolution. *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, San Diego, CA, 307-318.

Dowsland, W. B. 1991. Three-dimensional packing - Solution approaches and heuristic development. *International Journal of Production Research* 29(8): 1673-1685.

Gehring, H., K. Menschner, and M. Meyer. 1990. A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research* 44: 277-288.

Goldberg, D. E. 1989. *Genetic algorithms: In search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Herdy, M. 1991. Application of the evolution strategy to discrete optimisation problems. *Proceedings of the First International Conference on Parallel Problem Solving from Nature (PPSN)*, Dortmund, West Germany, 188-192.

Kröger, B. 1995. Guillotineable bin packing: A genetic approach, *European Journal of Operational Research* 84: 645-661.

Mohanty, B. B., K. Mathur, and N. J. Ivancic. 1994. Value considerations in three-dimensional packing - A heuristic procedure using the fractional knapsack problem. *European Journal of Operational Research* 74: 143-151.

Potter, M. A., and K. A. De Jong. 1994. A cooperative coevolutionary approach to function optimization. *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSNIII)*, Jerusalem, Israel, 249-257.

Potter, M. A., and K. A. De Jong. 1995. Evolving neural networks with collaborative species. *Proceedings of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 340-345.

Potter, M. A., and K. A. De Jong. 2000. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation* 8(1): 1-29.

Potter, M. A., K. A. De Jong, and J. J. Grefenstette. 1995. A coevolutionary approach to learning sequential decision rules. *Proceedings of the Sixth International Conference on Genetic Algorithms*, Pittsburgh, PA, 366-372.

Shimodaira, H. 1997. DCGA: A diversity control oriented genetic algorithm. *The Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, Glasgow, UK, 444-449.

List of Tables

Table 1 Classification of packages and the package quantity used in the first test example.

Type	Group	Dimension (unit: mm)			Quantity
		Height	Length	Width	
50	L	613	420	295	15
11	L	242	417	380	20
17	L	238	465	360	30
21	L	227	415	350	10
9	M	220	437	290	15
8	M	215	440	285	10
49	M	215	535	295	25
15	M	202	475	370	15
36	S	201	530	315	20
16	S	177	510	360	15
46	S	175	435	300	25
1	S	80	645	523	5
Total					205

Table 2 Classification of packages and the package quantity used in the second test example.

Type	Group	Dimension (unit: mm)			Quantity
		Height	Length	Width	
50	L	613	420	295	15
54	L	238	354	274	20
42	L	220	370	305	30
37	L	200	415	315	10
33	M	190	515	320	15
34	M	190	520	320	10
38	M	189	413	312	25
45	M	185	550	300	30
30	S	176	418	338	20
31	S	175	410	335	15
13	S	146	338	338	25
5	S	70	330	500	5
Total					220

Table 3 Parameter settings for the modified CCGA.

Parameter	Value	
	Example 1	Example 2
Number of sub-populations	3	3
Sub-population size	10	10
Chromosome length		
L sub-population (Large-sized package)	75	75
M sub-population (Medium-sized package)	65	80
S sub-population (Small-sized package)	65	65
Number of elitist individuals	1	1
Crossover probability	0.8	0.8
Mutation probability	0.1	0.1
Diversity control		
Shape coefficient, c	0.075	0.075
Exponent coefficient, α	0.195	0.195
Number of generations	100	100
Maximum possible solutions explored	31,000	31,000
Number of solutions in the search space	9.67×10^{107}	4.14×10^{115}

Table 4 Parameter settings for the standard genetic algorithm.

Parameter	Value	
	Example 1	Example 2
Population size	300	300
Chromosome length	205	220
Number of elitist individuals	1	1
Crossover probability	0.8	0.8
Mutation probability	0.1	0.1
Number of generations	103	103
Maximum possible solutions explored	31,200	31,200
Number of solutions in the search space		
Without a loading restriction	1.56×10^{203}	6.45×10^{217}
With a loading restriction	9.67×10^{107}	4.14×10^{115}

Table 5 Summary of results from all case studies in the first test example.

Case Study	Number of Containers	Empty Volume	Fitness
Case I – LL format			
GA without a loading restriction	3	$0.1927V_{\max}$	0.91927
GA with a loading restriction	3	$0.3691V_{\max}$	0.93691
Modified CCGA	3	$0.3831V_{\max}$	0.93831
Case II – WL format			
GA without a loading restriction	4	$0.8400V_{\max}$	0.85543
GA with a loading restriction	3	$0.2924V_{\max}$	0.92924
Modified CCGA	3	$0.3118V_{\max}$	0.93118
Case III – LW format			
GA without a loading restriction	4	$0.8400V_{\max}$	0.85543
GA with a loading restriction	4	$0.9577V_{\max}$	0.86719
Modified CCGA	4	$0.9592V_{\max}$	0.86735
Case IV – WW format			
GA without a loading restriction	4	$0.7939V_{\max}$	0.85082
GA with a loading restriction	3	$0.3724V_{\max}$	0.93724
Modified CCGA	3	$0.3823V_{\max}$	0.93830

Table 6 Summary of results from all case studies in the second test example.

Case Study	Number of Containers	Empty Volume	Fitness
Case I – LL format			
GA without a loading restriction	3	$0.3545V_{\max}$	0.93545
GA with a loading restriction	3	$0.6212V_{\max}$	0.96212
Modified CCGA	3	$0.6248V_{\max}$	0.96248
Case II – WL format			
GA without a loading restriction	3	$0.2812V_{\max}$	0.92812
GA with a loading restriction	3	$0.6007V_{\max}$	0.96007
Modified CCGA	3	$0.7498V_{\max}$	0.97498
Case III – LW format			
GA without a loading restriction	3	$0.3780V_{\max}$	0.93780
GA with a loading restriction	3	$0.7900V_{\max}$	0.97900
Modified CCGA	3	$0.8080V_{\max}$	0.98080
Case IV – WW format			
GA without a loading restriction	3	$0.3122V_{\max}$	0.93122
GA with a loading restriction	3	$0.7225V_{\max}$	0.97225
Modified CCGA	3	$0.7334V_{\max}$	0.97334

Table 7 ANOVA test on all results from the repeated simulation runs covering all case studies in both test examples.

Source of Variation	Sum of Squared Errors	Degree of Freedom	Mean Squared Error	F_0	p-value
(1) Algorithm	0.12926	2	0.064631	41.92	0.000
(2) Format	0.03590	3	0.010198	6.61	0.000
(1)*(2)	0.01253	6	0.002089	1.35	0.234
Error	0.35149	228	0.001540	-	
Total	0.52387	239			

List of Figures

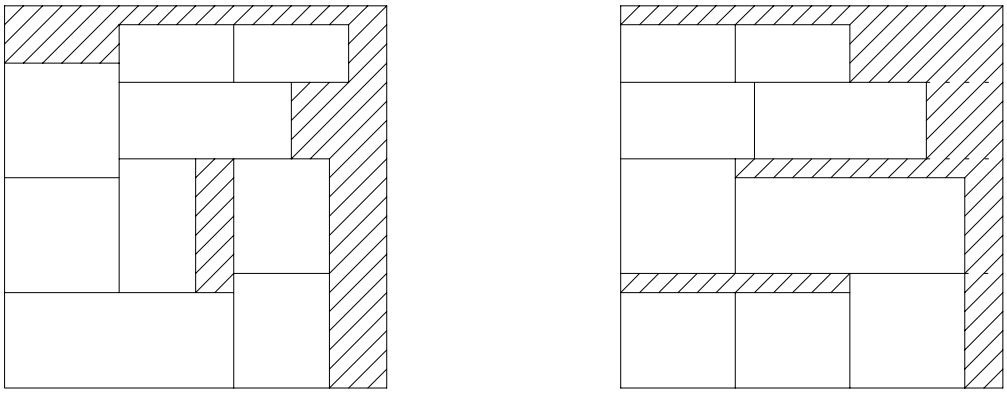

Figure 1 (a) Non-guillotineable packing scheme
(b) Guillotineable packing scheme.

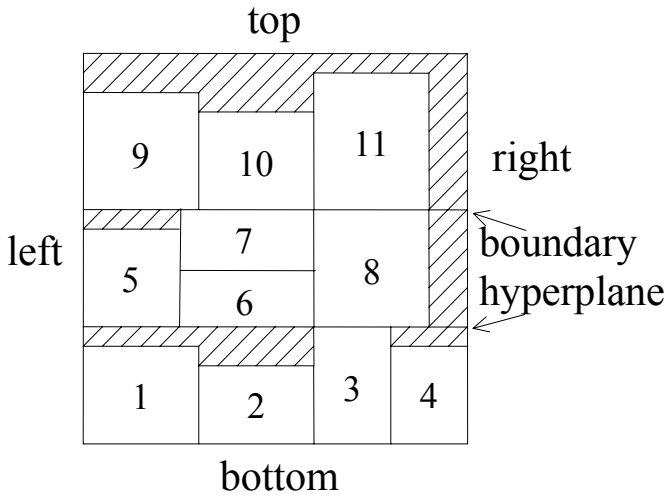

Figure 2 A container filled with three layers of packages.

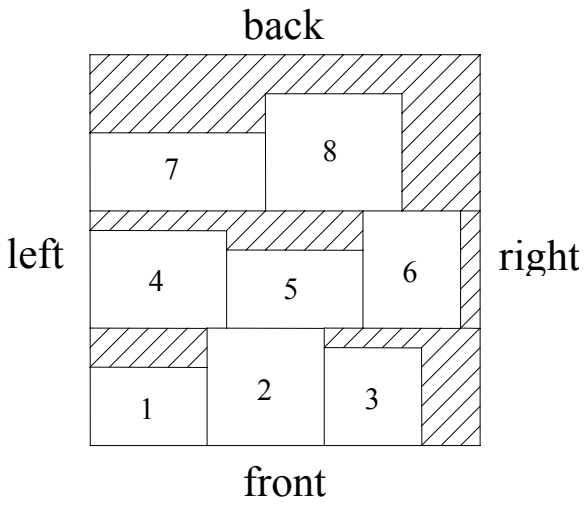

Figure 3 Packages arranged in the front-left justified format.

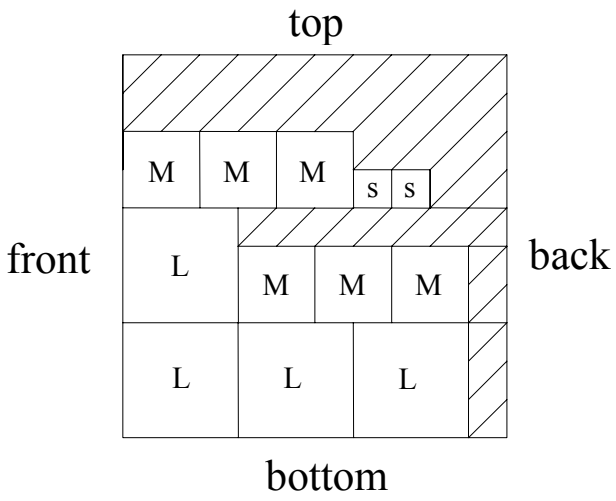

Figure 4 A container loaded with three groups of packages.

sub-population *A*		sub-population *B*		sub-population *C*
----------1---------	+	----------1---------	+	----------1---------
----------2---------		----------2---------		----------2---------
⋮		⋮		⋮
---------10---------		---------10---------		---------10---------

First generation

	sub-population *A*		sub-population *B*		sub-population *C*
1st portion	----------b---------	+	----------1---------	+	----------1---------
	best individual		----------2---------		----------2---------
	from the previous		⋮		⋮
	generation		---------10---------		---------10---------
2nd portion	----------1---------	+	----------b---------	+	----------1---------
	----------2---------		best individual		----------2---------
	⋮		from the previous		⋮
	---------10---------		generation		---------10---------
3rd portion	----------1---------	+	----------1---------	+	----------b---------
	----------2---------		----------2---------		best individual
	⋮		⋮		from the previous
	---------10---------		---------10---------		generation

Other generation

Figure 5 Schematic diagram illustrating combinations of individuals.

```
initialise M sub-populations where each sub-population has N individuals;
calculate fitness of N^M solutions;
assign the best fitness from the set of N^(M-1) solutions to each individual;
for (i = 0; i < MAX_GENERATION; ++i) {
    for (each sub-population) {
        fitness scaling;
        selection;
        crossover;
        mutation;
    } /* for */
    calculate fitness of M × N^(M-1) solutions created by combining offspring
      individuals with the best parent individuals;
    assign the best fitness from the set of (M-1) × N^(M-2) solutions to each
      individual;
    for (each sub-population) {
        merge the offspring individual set with the parent individual set;
        delete duplicate individuals;
        sort individuals
        select the best individual;
        assign a probability to each remaining individual according to the
          similarity between itself and the best individual;
        select individuals according to their survival probabilities;
        while (sub-population size < N) {
            select an individual according to its survival probability from
              the unselected individual list;
        } /* while */
    } /* for */
} /* for */
```

Figure 6 Pseudo code of the modified CCGA.

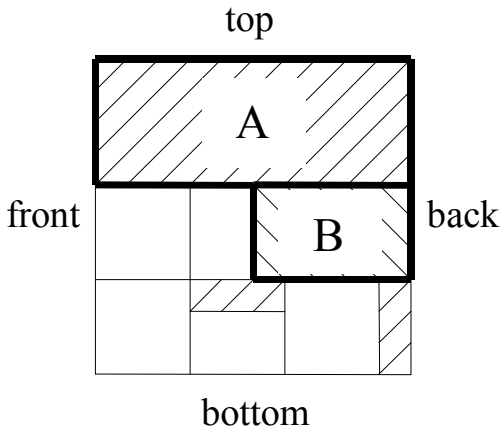

Figure 7 Diagram displaying the empty volume of the last container.

Individual 1:	***1***	**-**	***1***	**-**	***2***	**-**	***3***		2		1
Individual 2:	***1***	**-**	***1***		***1***	**-**	***2***		***2***	**-**	***3***

a - b: common pair of genes between two individuals

Figure 8 Example of the similarity measurement between two individuals.

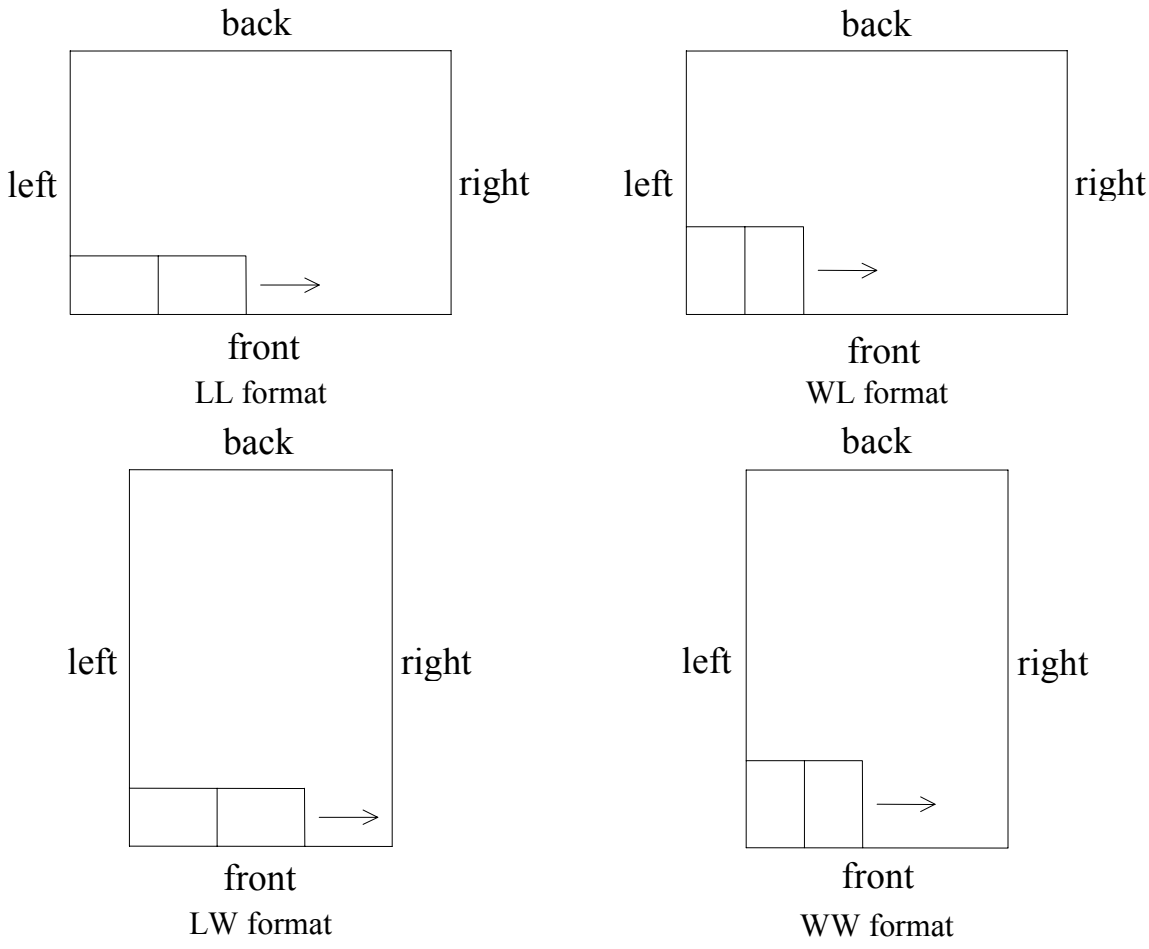

Figure 9 Four loading formats in each test example.