A Genetic Approach for Two Dimensional Packing with Constraints

Wee Sng Khoo, P. Saratchandran, and N. Sundararajan

School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore.
Email: epsarat@ntu.edu.sg

Abstract. In this paper, a new genetic algorithm based method is proposed for packing rectangular cargos of different sizes into a given loading area in a two dimensional framework. A novel penalty function method is proposed for checking the solution strings that have violated the loading area's aspect constraints. This method penalizes the solution strings based on the extent by which they violate the length and breadth constraints. The results from the proposed genetic algorithm are compared with the popular heuristic method of Gehring et. al. to show the advantages of this method.

Keywords: Genetic Algorithm, Two dimensional packing, Penalty function, Sentry point

1. Introduction

The field of the genetic algorithms [1-4] has been growing and has proved successful in many real-world applications like floorplan design in VLSI [5-7], cutting [8-11] and packing [12-20]. The reason for the popular usage of genetic algorithm is its robustness. If an artificial system can be made more robust, costly redesigns can be reduced or eliminated. Other attractive features of genetic algorithm are their ability to interface with other approaches for solving problems and it is very easy to modify for further enhancement as well as adding constraints. In this paper, a genetic algorithm is proposed to maximize the number of cargos that can be loaded into the given area without violating loading area's aspect constraints. The paper is organized as follows. Section 2 describes the representation of the problem and all the genetic operations. Section 3 gives results for cargo loading examples from the proposed method as well the heuristic method of Gehring et. al.[22]. Section 4 gives conclusions for this study.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 291–299, 2001.

2. Proposed Genetic Algorithm for 2-Dimensional Packing

2.1 Model Representation

A slicing tree structure is used to model the relative arrangement of the cargos (which is a solution to the problem). A slicing tree is an oriented rooted binary tree. Each terminal node of the tree is the identification number of the cargo. Each internal node is labeled *v* or *h*. *v* corresponds to a vertical joint or cut and *h* corresponds to a horizontal joint or cut. If the tree is viewed from top to bottom, it specifies how a big rectangle is cut into smaller rectangles. If it is viewed from bottom to top, it indicates how the smaller rectangles are joined together. In this problem, the packing method used will be similar to guillotine cuts [18-19]. That is the tree will be viewed from bottom to top, from left to right. At each internal node where two sub-rectangles are joined, it is considered as a new rectangle to the upper node. In addition, each node will be assigned an orientation symbol of '→' or '↑' except for the topmost node. The '→' symbol indicates a 90 degree turn of the rectangle and the '↑' symbol will indicates that the rectangle is kept upright (that is no change of orientation). As the tree is traversed from bottom to the top and left to right, a post-order polish notation is most suitable to represent the tree. Using a post-order polish notation would make it easier for the definition of objective function and coding of the strings. The string is coded in terms of the nodes and their orientation symbol ('→' or '↑'). In the coded string, the orientation symbol will follow the corresponding node. Hence, for *n* cargos, the strings will have length of (4**n*-3). All these are shown more explicitly in Fig. 1.

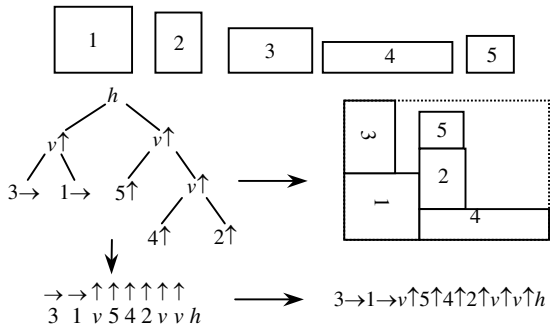


Fig. 1. Tree representation of cargo layout

2.2 Objective Function

After defining the model, an objective function for evaluating the string fitness is required. The density of the packing will be used as the fitness value for this objective function. The packing density is defined as the ratio of the sum of the area of all the cargos to the total area used by the solution string for packing all these cargos. The

fitness value itself is in normalized form, a value of 1 will indicate 100% efficiency of packing (i.e., no wastage of area). For a given number of cargos, the sum of the area of all cargos is always fixed. The total area used by the solution string for packing is calculated as follows. As the string is in post-order polish form, these calculations need only sum, max and swap operation together with a stack. If a rectangle's identification number is encountered, the length and breadth of this rectangle is read from the rectangle's length and breadth and pushed into the stack. If a '$\rightarrow$' symbol is encountered, the length and the breadth of the rectangle on the top of the stack is swapped. If a '$\uparrow$' symbol is encountered, no swapping is required. If a '*v*' or a '*h*' is encountered, the stack will pop two rectangles' length and breadth. A new meta-rectangle is created from these two rectangles. The length and breadth of the two rectangles are pushed back into the stack. The length of the new meta-rectangle is calculated as follows. If the node has a label '*v*', the length of the meta-rectangle is the maximum of the two rectangles' length, and breadth of the meta-rectangle is the sum of the two rectangles' breadth. If it has a label '*h*', the length of the meta-rectangle is the sum of the two rectangles' length, and breadth of the meta-rectangle is the maximum of the two rectangles' breadth. After the last symbol of the string, only a rectangle's length and breadth will be in the stack. The product of this length and breadth will give the total area required by the solution string for packing the cargos.

2.3 Selection

In this algorithm, Stochastic universal sampling selection [1,3] is used for the selection process. As the population size is small, Stochastic universal sampling is preferred as it would give a better selection [2].

2.4 Crossover

The tree structure with the identification number of the cargos renders normal crossovers like single-point, multi-point and even uniform not applicable. Hence, a new crossover operation has to be defined specifically to suit the tree structure. While forming new crossover, the following constraints have to be maintained so that after crossover the solution string is still a valid one. That is, after crossover, the child strings should have the exact numbers of cargo identification number, there should not be any repeat cargo identification number in the same string and the child strings should still be able to form valid tree (i.e., no missing terminal nodes or broken branches). In order to satisfy these constraints, the proposed crossover will choose only the common terminal nodes of the trees and interchange the nodes with their respective orientation symbol. This can be done by randomly choosing a cargo identification number in one parent string and interchange this cargo identification number and its orientation symbol with the same cargo identification number and its corresponding orientation symbol in the other parent string. This crossover operation will be controlled by the probability of crossover, p_c.

2.5 Mutation

In this genetic algorithm, three types of mutation operations are defined. They are mutation of joint, orientation and interchange.

2.5.1 Joint

Mutation of joint is to invert the joint symbol in the string from '*v*' to '*h*' and vice versa. This operation is controlled by the probability of joint, p_j. It will determine the chances of the mutation of a joint.

2.5.2 Orientation

Mutation of orientation is to invert the orientation symbol in the string from '$\rightarrow$' to '$\uparrow$' and vice versa. This operation is controlled by the probability of orientation, p_o that determines the chances of the mutation of orientations.

2.5.3 Interchange

Interchange mutation is to swap the position of any two random cargo identification numbers in the string. This operation is controlled by the probability of interchange, p_i that will determine the chances of the mutation due to an interchange.

2.6 Elitism

Elitism has been found to improve the genetic algorithm's performance significantly. Hence, in this genetic algorithm, elitism is also implemented to prevent the loss of highly fit strings missed out by the selection process or destroyed by crossover or mutation. The proposed elitism method selects the weakest string in the offspring's generation and replaces it with the fittest string from the parents' generation. The reason for choosing a low replacement is to let the population explore wider in the search domain. At the same time, it does not let the population to converge too early to a local optimum and miss other better solutions.

2.7 Area Constraint

The area constraint is handled as follows. First, the total area of all the given cargos is calculated to see if it is less than or equal to the loading area. If the total area is greater than the loading area, a cargo is randomly selected and withdrawn from the cargo list, so that the number of cargos to be packed is reduced by one. Then the total area of these reduced cargos is checked again. This procedure is repeated until the total cargo area is less than or equal to the loading area. After this condition is satisfied, the genetic algorithm described earlier is applied. After each generation, every string in the population is checked to see whether it satisfies two constraints. First one is that the solution's required packing area should be less than or equal to the loading area. Second one is that the required packing area's length and breadth do not exceed the loading area's length and breadth. If a string does not fulfill these two conditions that means that string has violated the constraints. In every generation, all

the strings in the population are checked against these two constraints. If after n generations, there is still no single string that satisfies these two constraints, then it means that it is not possible to find a valid solution using the current set of cargos. Hence, a cargo is again randomly selected and withdrawn from the cargo list. The genetic algorithm will reset, and start from the first generation with a new cargo list. This process will repeat until a valid string is obtained. The genetic algorithm will then proceed beyond the n^{th} generation and will continue until the stopping criterion is met. The choice of generation 'n' as a 'sentry point' is arbitrary. If the sentry point is set too high, there could be wastage in computation. If it is set to too low, there would not be enough evolutions for a valid solution to come about. There would be a lot of cargos excluded from packing resulting in solution strings that are not the optimum.

2.8 Penalty Function

In the process of handling constraints, whenever a violated string is encountered, the objective function will be modified with a penalty function to suppress the fitness value of that string such that the violated string will have a weak fitness value. Different types of penalty functions could be used to handle the violated strings. One is to give all those violated strings a zero value for their fitness. Another is to reduce the fitness by an arbitrary fraction. Assigning a zero value to the fitness is too drastic. Although the strings may be violating the constraints, it may also have some good schema inside it. Hence assigning a zero value is not a good penalty function. Reducing the fitness by an arbitrary fraction will not differentiate between solution strings of extreme violation and slight violation of the constraints. In the proposed penalty function, if the length of the solution has violated the loading area length, then fitness of the string is reduced by a factor of L_r. If the breadth is violated, the fitness is then reduced by a factor of B_r. If both are violated, then its fitness is reduced by both factors. In this way, the penalized fitness depends on the extent by which a solution string violates the constraints. The penalized fitness is calculated as:

$$\text{Penalized Fitness} = \text{Fitness} \times L_r \times B_r \tag{1}$$

where $$L_r = \begin{cases} CL/SL & \text{if } CL/SL < 1 \\ 1 & \text{if otherwise} \end{cases},$$

and $$B_r = \begin{cases} CB/SB & \text{if } CB/SB < 1 \\ 1 & \text{if otherwise} \end{cases}$$

CL & *CB* are the length and breadth of loading area and *SL* & *SB* are the length and breadth of the solution string.

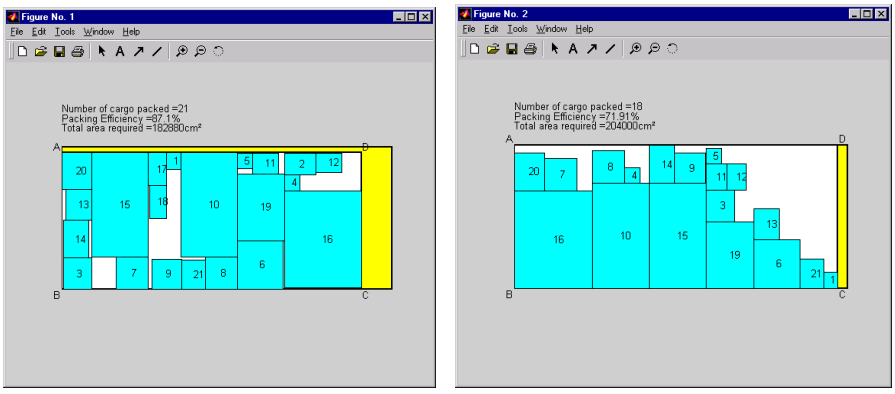
3. Simulation Results

3.1 Optimal Values of Crossover and Mutations Probabilities

Extensive simulations studies [21] were carried out to obtain the optimal values for the crossover and mutations probabilities. These are p_c = 0.99, p_i = 0.51, p_j =0.02 and p_o = 0.01.

3.2 Results of Comparison with the Heuristic Method

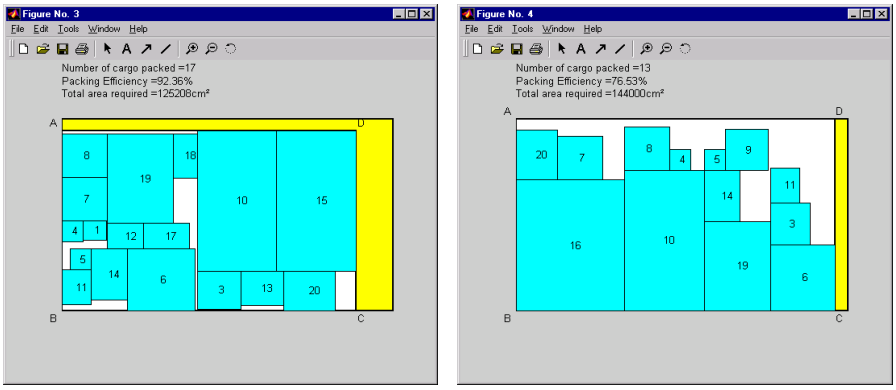
A comparison between our proposed genetic algorithm and the heuristic method [22] proposed by Gehring et.al. is carried out for two examples. In both examples, there are 21 cargos that need to be loaded. The dimensions of the cargos used in the comparison study are listed in Appendix A. In example 1, the loading area is 700 cm x 300 cm. Fig. 2 shows the results of the two methods. It can be seen from the figure that our genetic algorithm is able to pack all the 21 cargos while the heuristic method is able to pack only 18 cargos.



(a) (b)

Fig. 2. Packing results from the two methods for Example 1. (a: proposed GA, b: Heuristic)

In example 2, the given loading area is 500 cm x 300 cm. This is less than the sum of the area of all the 21 cargos. Hence it is impossible to pack all the cargos within the loading area. Fig. 3 shows the results obtained form our genetic algorithm and the heuristic method. It can be seen from the figure that our genetic algorithm is able to pack 17 cargos while the heuristic method is able to pack only 13 cargos.



(a) (b)

Fig. 3. Packing results from the two methods for Example 2. (a: proposed GA, b: Heuristic)

Table 1 summarizes the results obtained for the two examples. The packing efficiency referred in the table is the ratio of sum of the area of the cargos packed to the area required by the algorithms (area ABCD in Fig. 2 & 3) to pack these cargos.

Table 1. Results of the two sets of comparisons

	Number of Cargos Packed		Packing Efficiency	
	Genetic Algorithm	Heuristic Method	Genetic Algorithm	Heuristic Method
Example 1	21	18	87.10 %	71.91 %
Example 2	17	13	92.36 %	76.53 %

4. Conclusion

In this paper, a new genetic algorithm for solving the cargo loading problem in two-dimension is proposed. The algorithm makes use of a novel penalty function method to handle the solution strings that violate the length and breadth constraints imposed by the loading area. Comparisons with a well-known heuristic method have shown that the proposed genetic algorithm is superior in terms of number of cargos packed and packing efficiency.

Reference

[1] David E. Goldberg, "Genetic Algorithm in Search, Optimization & Machines Learning", Addison-Wesley, 1989.

[2] Srinivas, E. and Patnaik, L. M., "Genetic Algorithms: A Survey", *Computer* Vol. 24/6, 17-26, 1994.

[3] Mitchell, M., "An Introduction To Genetic Algorithms", MIT Press, 1997.

[4] Ribeiro Filho, J. L., Treleaven, P. C. and Alippi, C., "Genetic-Algorithms Programming Environments", *Computer* Vol. 24/6, 28-43, 1994.

[5] Wong, D. F. and Liu, C. L., "A New Algorithm for Floorplan Design", *Proc. 23th ACM/IEEE Design Automation Conference*, 101-107, 1986.

[6] Wong, D. F. and Sakhamuri, P. S., "Efficient Floorplan Area Optimization", *Proc. 26th ACM/IEEE Design Automation Conference*, 586-589, 1989.

[7] Wang, T. C. and Wong, D. F., "An Optimal Algorithm for Floorplan Area Optimization", *Proc. 27th ACM/IEEE Design Automation Conference*, 180-186, 1990.

[8] Grinde, R. B. and Cavalier, T. M., "A new algorithm for the minimal-area convex enclosure problem.", *European Journal of Operation Research* 84, 522-538, 1995.

[9] Li, Z. and Milenkovic, V., "Compaction and separation algorithms for non-convex polygons and their applications.", *European Journal of Operation Research* 84, 539-561, 1995.

[10] Valério de Carvalho, J. M. and Guimarães Rodrigues, A. J., "An LP-based approach to a two-stage cutting stock problem.", *European Journal of Operation Research* 84, 580-589, 1995.

[11] Arenales, M. and Morabito, R., "An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems.", *European Journal of Operation Research* 84, 599-617, 1995.

[12] Bischoff, E. E. and Wäscher, G., "Cutting and Packing", *European Journal of Operational Research* 84, 503-505, 1995.

[13] Darrell Whitley, V Scott Gordan and A. P. Willem Böhm, "Knapsack problems", *Handbook of Evolutionary Computation* 97/1, G9.7:1-G9.7:7, 1997.

[14] Dowsland, K. A. and Dowsland, W. B., "Packing problem.", *European Journal of Operation Research* 56, 2-14, 1992.

[15] Neolißen, J., "How to use structural constraints to compute an upper bound for the pallet loading problem.", *European Journal of Operation Research* 84, 662-680, 1995.

[16] Bischoff, E. E., Janetz, F. and Ratcliff, M. S. W., "Loading pallet with non-identical items.", *European Journal of Operation Research* 84, 681-692, 1995.

[17] George, J. A., George, J. M. and Lamar, B. W., "Packing different-sized circles into a rectangular container.", *European Journal of Operation Research* 84, 693-712, 1995.

[18] Kröger, B., "Guillotineable bin packing: A genetic approach.", *European Journal of Operation Research* 84, 645-661, 1995.

[19] Hwang, S.-M., Kao, C.-Y. and Horng, J.-T., "On Solving Rectangle Bin Packing Problems Using Genetic Algorithms.", *IEE Transactions on Systems, Man, and Cybernetics,* Vol. 2, 1583-1590, 1994.

[20] Lin, J.-L., Footie, B., Pulat, S., Chang, C.-H. and Cheung J. Y., "Hybrid Genetic Algorithm for Container Packing in Three Dimensions.", *Proc. 9th Conference on Artificial Intelligence for Applications*, 353-359, 1993.

[21] Khoo, W. S., "Genetic Algorithms Based Resource Allocation Methods", Technical Report EEE4/038/00, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Dec 2000.

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

Appendix A Cargo Details

Dimensions of the 21 cargos are listed in Table 2 (Length by Breadth in cm).

Table 2. Dimensions of the cargos used in Examples 1 and 2

Cargo Number.	Length x Breadth(cm)	Cargo Number.	Length x Breadth(cm)	Cargo Number.	Length x Breadth(cm)
1	30 x 35	8	68 x 68	15	120 x 220
2	46 x 66	9	64 x 64	16	163 x 205
3	60 x 66	10	120 x 220	17	40 x 69
4	32 x 33	11	44 x 55	18	36 x 69
5	32 x 33	12	40 x 55	19	100 x 140
6	97 x 103	13	54 x 64	20	62 x 78
7	68 x 68	14	54 x 80	21	50 x 62