An Evolutionary Metaheuristic for the 2D Guillotine Cut Problem

José Ignacio Peláez
Department of Languages & Computer Sciences.
University of Malaga. Spain.
Research of Prometeo Project. University of Guayaquil.
Guayaquil, Ecuador.
jipelaez@uma.es

A.H. Yanez
Department of Mathematics & Physical Sciences.
University of Guayaquil.
Guayaquil, Ecuador.
alcibar.yaneze@ug.edu.ec.

Eduardo A. Santos
Department of Mathematics & Physical Sciences.
University of Guayaquil.
Guayaquil, Ecuador.
eduardo.santosb@ug.edu.ec

L.M. Moncayo
Department of Mathematics & Physical Sciences.
University of Guayaquil.
Guayaquil, Ecuador.
marcelo.moncayot@ug.edu.ec

ABSTRACT

The two-dimension guillotine cut is actually one of the most interesting problems in modern industries like metallurgic, textile, wooden... in which it's needed to cut sheets in pieces with an associated dimensions and benefits in the way to maximize the final benefit. The purpose of this work is to present an evolutionary metaheuristic for the two-dimension cut problem using guillotine, also it's shown how this approach can be applied to solve these types of problems, it's compared with other exact algorithms and finally it's defined an evolutionary representation which may be used with different metaheuristics.

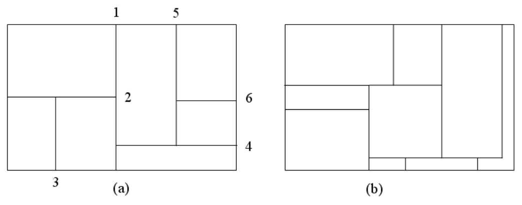
Keywords: Cut Problems, Soft Computing Application, Optimization.

I. INTRODUCTION

The 2 dimensions cut problem using guillotine, from now CG2D, is outstanding due to its transcendence in different industries like metallurgic, wooden, textile, etc., it has been solved from the 60th using different types of algorithms approached with traditional techniques as dynamic or lineal programming [3][6], heuristic methods [1], or even variants that use several types of techniques [4] [7] [9] [10] [11]. To develop efficient algorithms for this problem is very interesting and important for these industries because of the increase in yield and competitiveness when maximizing the benefit of the cuts made.

The CG2D consists on to divide a sheet in rectangular segments where to locate pieces with a given size and value, so that the sum of the values of the cut pieces are maximized. These types of problems use patrons of cut where only is possible to cut uninterruptedly on the sheet, these types of cuts are denominated valid cuts. Figure 1.a shows a pattern of cut in 6 stages. The number of the cut indicates the stage in which the cut has be done. In this example the direction of cut in the first stage is parallel to the axis-y; the second is parallel to the axis-x;

the third parallel to the axis-y; the quarter to the axis-x; the fifth to the axis-y; and finally, the sixth to the axis-x. A sequence of nonvalid cuts is shown in figure 1.b.



Formally the 2D guillotine cut (CG2D) is defined as follow: Be $A_0 = (\alpha_0, \beta_0)$ a rectangular sheet with length α_0 and width β_0 and, be R a set of m smaller rectangular pieces $R_1, R_2, ..., R_m$ with dimensions (α_1, β_1), (α_2, β_2), ..., (α_m, β_m), values $v_1, v_2, ..., v_m$ and max cardinal $b_1, b_2, ..., b_m$. The objective is to make a cut pattern for A_0 with maximum sum of the cuts pieces using no more than $\boldsymbol{b_i}$ copies of each rectangle $\boldsymbol{R_i}$ in the pattern. The followings restrictions must be satisfied:

1. All dimensions (α_i, β_i) for *i = 1, 2,..., m* are integers in the axis-x or axis-y.

2. The directions of the pieces are fixed (a piece with length e and width f is different of other piece with length f and width e).

To solve these types of problems have been applied a great number of different techniques, from tradicional to artifical intelligent techniques, even hibrid algorithms that mix different types of approach [2] [5] [6] [8] [9] [10] [11]. One of these types of solutions is the algorithm proposed by Christofides and Hadjiconstantinou [5]. This algorithm will be used in this work to compare results because it produces good results in small problems, also it presents a search tree to solve the guillotine cut in which, the size of the tree is limited by

pruning derived from relaxed solutions of a formulation of the problem using dynamic programming with a method of upward solutions space to optimize the pruning. The procedure of this algorithm is the following: Be a rectangle A_0 with size (α_0, β_0) and be *m* pieces R_i with size (α_i, β_i) , value v_i and cardinality $b_i \in B_{xy}$.

1. Number of times that each piece can appear. First, the number of repetitions S_{xy} for each piece R_i in the rectangle A_0 are calculated; b_i is determined.

$$S_i < \hat{S}_i = \begin{cases} \min(b_i, \lfloor xy/(\alpha_i\beta_i) \rfloor) & \text{si}\ \alpha_i \le x, \beta_i \le y \\ 0 & \text{en otro caso} \end{cases}$$

2. Cut points. All cut combinations (verticals *L* and horizontals *W*) are determinated.

$$L = \left\{ x / x = \sum_{i=1}^{m} \theta_i \alpha_i, 1 \le x \le \alpha_0, 0 \le \theta_i \le b_i \text{ y } \theta_i \text{ entero}, \forall \text{i} = 1,..., \text{m} \right\}$$

$$W = \left\{ y / y = \sum_{i=1}^{m} \tau_i \beta_i, 1 \le y \le \beta_0, 0 \le \tau_i \le b_i \text{ y } \tau_i \text{ entero}, \forall \text{i} = 1,..., \text{m} \right\}$$

3. Heuristic Solution. A heuristic solution is calculated. It will be used to recalculated the weight of the pieces. This solution is obtained using a greddy algorithm.

4. Calculation of superior level. Using dinamic programming a superior level is calculated, in the way to obtain the cuts for the pieces. The firts level is calculated without any restrictions as follow:

$$Z_{UB} = \max[F_n(\alpha_0, \beta_0, Q), G_n(\alpha_0, \beta_0, Q)]$$

where

- F_k (x, y, q) is the optimal cut for the step *k* in a rectangle with size (x, y) when the direcction of cut in the first stage is parallel to the axis 'y'. This value is calculated as follow:

$$F_k(x, y, q) = \max\left[G_{k-1}(x, y, q), \max_{x' < x, x' \in L, q' = 0,...q} \left[F_k(x', y, q') + G_{k-1}(x - x', y, q - q')\right]\right]$$

The first term corresponds to the case where there is no cut in (*x, y*) parallel to axis-*y* in the first stage, which implies that the second cuts is parallel to the axis-*x*. The second term corresponds to the case in which exist at least a cut parallel axis-*y* in the first stage, then x' $\in$ L will be the cut with greater value in coordinate *x* which produces two rectangles.

- The definition for the cut in the first stage parallel to the axis 'x' is similar:

$$G_k(x, y, q) = \max\left[F_{k-1}(x, y, q), \max_{y' < y, y' \in W, q' = 0,...q} \left[G_k(x, y', q') + F_{k-1}(x, y - y', q - q')\right]\right]$$

- For the base cases (stage 0) the function F and G are calculated as follow:

$$F_0(x, y, q) = \max_i[v_i / \alpha_i \le x, \beta_i \le y, q_i \le q, i = 1,..., m]$$
$$G_0(x, y, q) = F_0(x, y, q).$$

- The function *q* are calculated as follow: $q(S_{xy}) = q \equiv \sum_{s_i \in S_{xy}} s_i q_i$

- In the calculation of F and G is applied a restriction for the dinamic programming solution that consists on assigning a weight to the pieces. The pieces must approve the restriction imposed by Q then they can intervene in the solution. The function Q is calculated using the following expression:

$$Q = \sum_{\hat{s}_i \in \hat{S}_{\alpha_0 \beta_0}} \hat{s}_i q$$

- The stop condition for the dinamic algorithm is determined with the following expresions:

$$F_k(\alpha_0, \beta_0, \hat{S}_{\alpha_0\beta_0}) = F_{k-1}(\alpha_0, \beta_0, \hat{S}_{\alpha_0\beta_0}) \text{ and}$$

$$G_k(\alpha_0, \beta_0, \hat{S}_{\alpha_0\beta_0}) = G_{k-1}(\alpha_0, \beta_0, \hat{S}_{\alpha_0\beta_0})$$

5. Tree search. The solution calculated in the 4th step using branch and boond is tested. If it is feasible, then it is optimal, in other case a readjustment is made in the weight of pieces to delimiting the solution. The tree search look for a valid cut sequence to find a value Z_{UB}, if this sequence is not found, a non-valid sequence will be found which will be used to recalculate the weight q_i. Three types of cut are defined: cut-x, cut-y and cut-0. Both first are vertical and horizontal cuts; and the last one consist on using the cut rectangle to introduce a piece inside. If a cut-0 is made then it will be impossible to cut in this rectangle.

6. Readjusting the weights. If the search tree doesn't find a feasible solution, a readjustment of the pieces weight must be made. The q_i are modified based on the results obtained with the tree. The pieces that produce non feasible soltions have less possibilities to appear in the new solution. The q_i is calculated as follow

$$q_i = \begin{cases} q_i + \lfloor t\sqrt{\gamma_i - b_i} \rfloor & \text{si } \gamma_i > b_i \\ \max\left[0, q_i - \lfloor t\sqrt{b_i - \gamma_i} \rfloor\right] & \text{si } \gamma_i \le b_i \end{cases}$$

where $t > 0$

$$t = \frac{1}{2}\sqrt{\frac{\pi(Z_{UB} - Z_{LB})}{\sum_{i=1}^{m}(b_i - \gamma_i)^2}}$$

The π parameter is initialised to 1,0 for a number of consecutive iterations, being reduced until it is minor than $\varepsilon = 0.05$.

γ_i is the number of times that a rectangle *i* is used in the *n*-stages for the cut pattern A_0, If $\gamma_i \le b_i$ for all $i = 1,...,m$, then a feasible solution for the CGC problen have been found, this solution is necesarilly optimal; in other case an ascending set of solutions is found to optimize the result of the superior level.

Z_{LB} is the value of the inferior level (for a feasible solution) in the optimal value for the CGC problem, obtained by a heuristic procedure in the 3th step. Z_{UB} is the actual value of the superior level using the dinamic programing of the step 4. If $Z_{UB} = Z_{LB}$, then the algorithm finish with Z_{LB} as the optimal solution for the initial problem.

The procedure used to readjusting the weight of the pieces is based on the following: if $\gamma_i > b_i$ for some piece *i*, then may be reasonable to reduce this number to make it (more) feasible. A procedure to make this is increase the value q_i of a piece *i* with $\gamma_i > b_i$ and, at the same time, decrease the weight for the pieces in which the associated restrictions are satisfied (for example pieces with $\gamma_i \le b_i$).

When the weight q_i are readjusted, the algorithm back to the step 4 to calculate the new Z_{UB} only if π is not less than 0.05, in this case the algorithm is stoped and a superior level is obtained which may be used in other type of algorithm.

This technique is very effective in small problems, but it is ineffective in problems of more size because the cut with guillotine is a NP-complete problem. For this reason, in the last years, the techniques of the artificial intelligence have been applied in the resolution of this type of problems, especially the genetic algorithms, because they are very appropriate to solve problems in complex spaces [8]. In this work a genetic algorithm is presented for the problem CG2D, and it has been organized as follow: in the section 2 a genetic algorithm is presented to solve the cut problem with guillotine; in the section 3 the results of this algorithm are compared with others; and finally the conclusions and future works are presented.

II. PROPOSED GENETIC ALGORITHM

In this section a genetic algorithm for the problem CG2D is proposed and it is shown how it can be applied to solve this type of problem. First the genetic representation given to the problem is shown; second the function of adaptation is presented; third the evolutionary process is described; and finally, the genetic operators are presented.

In this work with the purpose of distinguishing among the pieces given in a set *R* and the rectangles made by the cuts in A_0 with each stage during the cut process, from now on we refer to the first as "pieces" and to the second as "rectangles.

A. Genetic Representation

The genetic representation for this problem is conditioned by the decisions in the cut process. They are the following:

1. What rectangle cut.
2. What type of cut use? Cut parallel to axis-y (cut in coordinate x), cut parallel to axis-x (cut in coordinate y), or to locate a certain piece in a rectangle, this is denominated as cut-0.
3. Where to make the cut or what piece choose. In our algorithm the cut process is made in two steps: first we cut to obtain rectangles and after a process of assignment of pieces to this rectangles is made. This assignment process is made assigning the piece of more benefit. At first we determine the cuts based on the size of the pieces, but after several simulations it was verified that any modification in the individuals produces non valid individuals (insufficient size of area to locate the piece, to overcome the cardinality of the piece) and therefore a very high computational cost that makes that the algorithm is not operative.

Based on these decisions it is established that each individual is formed by two chromosomes, a first denominated Type-Cut which indicates the cut sequence, and a second named Cut which indicates the coordinates where the cut is made.

- Type-Cut Chromosome: {x|y|0} [{x|y|0}]
- Cut Chromosome: {cut_coordinate} [{cut_coordinate}]

Example of individual:

- Type_Cut: “x,y,0,0,0”.
- Cut : “20,30”

In figure 2, the cut sequence for an individual is shown. It’s made a cut in the axis x in coor 20; a cut in axis y in coor 30; and three cut-0 in which three pieces are located.

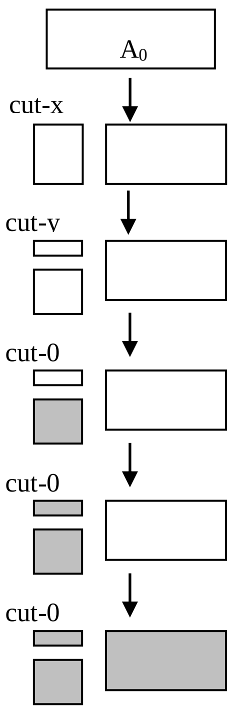


Figure 2. Cut sequence representation

An important aspect in the genetic representation is how to determine in an easy and quick way if an individual represents a sequence of valid cuts. We have been defined the following representation rules to facilitate the computational aspect of the algorithm:

1. **The parentheses rule**

The Parentheses rule allows us to determine in a simple way if an individual describes a valid cut sequence. This rule consists in the following: If cut-x and cut-y are represented as open parenthesis and court-0 as closed parenthesis, and we add an opened initial parenthesis, always we must obtain a correct structure of parenthesis. Following an example is shown:

Valid chromosome:

Type_Cut chromosome: "x, y, 0, x, 0, 0, 0"
Parentheses rule: **(** (( ) ( )))

Non valid chromosome:

Type_Cut chromosome: "x, 0, 0, x, 0, 0, 0"
Parentheses rule: **(** ( )) ( )))

In this example two cut in the axis x have been done, with these we obtain three rectangles, and five cuts type 0 have been made too, that would be impossible to carry out because we have three rectangles where to locate three pieces.

2. **The cut rule**

This rule avoids that the cut sequence surpass the size of the rectangle; the cuts of the cut coordinate must be smaller than the size of the axis where the selected rectangle is cut.

3. **The lenght rule**

This rule relates the size of the chromosomes, in way that an individual that make *k* cuts (cut_x, cut_y) will have a type_cut chromosome with ((k+1) 2) + 1 allele, and a cut chromosome of longitude (k+1) allele.

B. Adaptation Function

The adaptation function used is very simple, this represents the value of the pieces that have been located.

C. Evolutionary Process

The evolutionary process used in this algorithm is based on the concept of cut stages. A cut stage represents the number of cuts that have been done to obtain a solution, for example, those individuals that have a cut will belong to the population of cut stage 1, the individuals with two cuts will belong to the population of cut stage 2., and so on. In this way, k+1 populations are generated to represent cuts from the stage 0 to the stage k, reducing the initial problem in smaller problems. It is needed because it is not feasible from a point of view of computational efficiency to evolve with individuals belonging to different stages.

Once shown the stage concept, the evolutionary process is the following: first a population is generated for each cut stage; next each population evolves in an independent way; when the populations have evolved, the mean of the best individual of each population is determined and those stages or populations that have not overcome the mean are eliminated; this process is done until we obtain an unique population.

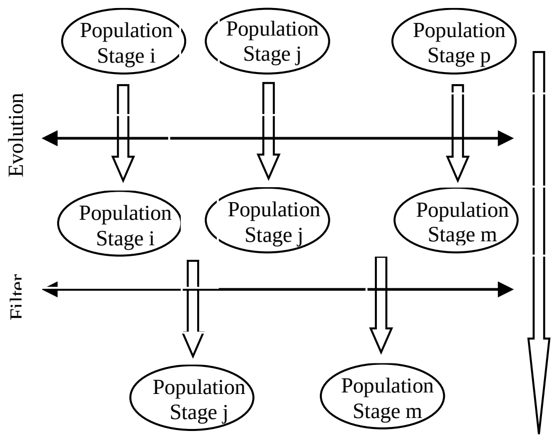


Figure 3. Evolution process

D. Genetic operators

The genetic operators that have been developed are the following:

- **Selection**: The roulette algorithm has been used.
- **Crossing**: The crossing operator works at level of cut and type-cut, then we have two types of crossing:

a) **Crossing cut.** An individual can cross with other compatible individual, a cut-x or cut-y with other, or a cut-0 with other one, because in another case the parenthesis rule would not be fulfilled.

b) **Crossing type-cut.** The process is similar with the previous one. When the cross is done in the *type-cut* chromosome, also the *cut* chromosome is interchanged.

• **Mutation:** The mutation operator also works at cut and cut-type level, with two types of mutation:

a) **Mutation type-cut.** In this case, it is only possible to mutate cut-x in cut-y and vice versa, in another case parenthesis rule would be broken.

b) **Mutation cut.** In this case only changes the cut value

III. COMPARISON OF THE ALGORITHM

The genetic algorithm developed in this work has been compared with the algorithm proposed in [5], due to this algorithm produce in most of the cases exact solutions in problems with small size. The 18.31% of the simulations done with this algorithm produced nonexact solutions, being these of the heuristic type. The next table show the population data.

	Datas
Total Simulations	10.000
Poblation Size	150 to 200.
Number of Pieces	Max 25
Number of Iterations	80 to 100 by poblation/iteration

Table 1. Population data.

The generation of valid individuals in the evolutionary process has forced to have populations of sizes between 150 and 200 individuals. The use of inferior populations produces a diminution remarkable in the computational efficiency due to the difficulty to generate individuals valued. Finished the simulations the obtained results are showing in table 2.

Table 2. Simulation Results.

	Datas
Total Simulations	10.000
Best solutions with genetic algorithm	1831
Same Solution	6368
Worst solution with genetic algorithm	1801

IV. CONCLUSIONS AND FUTURE WORKS

In this work an evolutionary metaheurística for the 2D cut problem using guillotine is presented. Also it has been demonstrated how it can be applied to solve this type of problem and finally its results have been compared with another algorithm that presents exact solutions in problems of small size.

Also a new genetic representation for this problem and the rules of validation of chromosomes (parenthesis rule, cut rule and length rule) that increase the computational efficiency of the algorithm are developed.

Finally, this algorithm has been compared with another which produces exact results, obtaining very satisfactory results that animate to us to follow this line and to compare their results with other algorithms and heuristics as Fans techniques.

ACKNOWLEDGEMENTS. This work is supported by PROMETEO Project. University of Guayaquil. Government of Ecuador.

REFERENCES

[1] Enrico Faggioli y Carlo Alberto Bentivoglio, “Heuristic and exact methods for the cutting sequencing problem**.”** **European Journal of Operational Research**, 110. pp. 564-575. 1998.
[2] E. Falkenauer, “Genetic Algorithms and Grouping Problems”. John Wiley and Sons. 1998.
[3] Julien Antonio, Fabrice Chauvet, Chengbin Chu y Jean Marie Proth. “The cutting stock problem with mixed objectives: Two heuristics based on dynamic programming”. **European Journal of Operational Research**. 114. pp. 395-402. 1999.
[4] Mikuel Rönuqvist, “A method for the cutting stock problem with different quañities.” **European Journal of Operational Reserach** 83/1. pp. 57-68. 1995.
[5] Nicos Christofides y Eleni Hadjiconstantinou, “An exact algorithm for ortogonal 2-D cutting problems using guillotine cuts.” **European Journal of Operational Research**. 83. pp. 21-38. 1995.
[6] Ortmann, F. G., Ntene, N., & Vuuren, J. H. (2010). New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. **European Journal of Operational Research**, 203, 306-315.
[7] P.C. Gilmore, R.E. Gomory, “Linear problem.” **Journal of Operation Research**. 9. 1961.
[8] R. Sharma, T. Balacahander, S. and, Q. Zhang, “A genetic algorithm for the non-convex cutting stock problem.” **Transactions of the North American Manufacturing Research Institute**. XXV. pp.281-286. 1997.
[9] Wäscher, G., Haussner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. **European Journal of Operational Research**, 183, 1109-1130.
[10] Wei, Lijun, Tian, Tian, Zhu, Wenbin and Lim, Andrew, (2014), A block-based layer building approach for the 2D guillotine strip packing problem, **European Journal of Operational Research**, 239, issue 1, p. 58-69.
[11] Wong, L., & Lee, L. (2009). Heuristic placement routines for two-dimensional bin packing problem. **Journal of Mathematics and Statistics** 5 (4), Páginas 334-341. approach to the cutting stock.