Solving the Two Dimensional Cutting Problem using Evolutionary Algorithms with Penalty Functions

Beraudo V., Orellana A., Alfonso H., Minetti G., Salto C.
Laboratorio de Investigación en Sistemas Inteligentes (LISI)
Facultad de Ingeniería - Universidad Nacional de La Pampa
Calle 110 Esq. 9 (6360) General Pico – La Pampa – Rep. Argentina
Te. / Fax: (02302) 422780/422372, Int. 6302
e-mail: {beraudov, alfonsoh, minettig, saltoc}@ing.unlpam.edu.ar, orellana@cospec.com.ar

ABSTRACT

In this work a solution using evolutionary algorithms with penalty function for the non-guillotine cutting problem is presented. In this particular problem, the rectangular pieces have to be cut from an unique large object, being the goal to maximize the total value of cut pieces. Some chromosomes can hold pieces to be cut, but some pieces cannot be arranged into the object, generating infeasible solutions. A way to deal with this kind of solutions is to use a penalizing strategy. The used penalty functions have been originally developed for the knapsack problem and they are adapted for the cutting problem in this paper. Moreover, the effect on the algorithm performance to combine penalty functions with two different selection methods (binary tournament and roulette wheel) is studied. The algorithm uses a binary representation, one-point crossover, big-creep mutation and in order to evaluated the quality of solutions a placement routine is considered (Heuristic with Efficient Management of Holes). Experimental comparisons of the performance of the resulting algorithms are carried out using publicly available benchmarks to the non-guillotine cutting problem. We report on the high performance of the proposed models at similar (or better) accuracy with respect to existing algorithms.

Keywords: Non-guillotine Cutting Problem, Evolutionary Algorithm, Penalty Function, Selection Methods.

Event: Workshop de Agentes y Sistemas Inteligentes (WASI)

I. INTRODUCTION

The optimization plays a central role in operations research/management science and engineering design problems. There are different techniques to resolve but some problems are very complex in nature and difficult to solve with conventional optimization techniques. One example of them is the Cutting Problem (CP), in which there is an objective function in presence of a set of constraints on a specific domain and the solution that satisfied all the constraints is called a *feasible solution* to the problem. The collection of all such solutions forms the feasible region. The conventional solution methods for this kind of problems are very complex and not very efficient. For them, in the last years there has been a growing effort to apply some meta-heuristics to obtain good solutions in a limited time. In particular the Evolutionary Algorithms are a technique that provide a good quality of solutions and allow to try the infeasible solution with different strategies such as rejecting, repairing or penalizing it.

In the Cutting Problem there are a set of small objects ('products', 'pieces', 'items') that is necessary to allocate in a set of large objects ('material', 'object'), which in general is called containment region. The objective is to obtain the more efficient allocation without overlapping, maximizing the quantity of small objects and/or minimizing the quantity of large object used for them. CP can be found in areas such as computer science, industrial engineering, logistics, manufacturing, etc. A typology of CP given by H. Dyckhoff [6, 7] distinguishes among the dimension of objects (1, 2, 3, …, N), the kind of assignment and the structure of the set of large objects and of small objects.

In many cases there is special requirement on the cutting patterns: only orthogonal guillotine cuts are allowed, i.e., pieces may only be cut horizontally or vertically from one border to the one opposite. Furthermore, the number of the stages of such cuts is often limited in real word applications. In some specific applications the pieces have a fixed orientation (not allow rotation). In same real contexts the rotation of elements (usually by 90º) can be accepted in order to generate a better assignments. For example, rotation is not allowed in newspaper paging or when the items to be cut are decorated or corrugated, whereas orientation is free in the case of plain materials and in other contexts. The guillotine constraint is frequently present in cutting problems, due to technological characteristic of automated cutting machines.

In this paper the two-dimensional cutting problem is tackled, in particular the constrained two-dimensional non-guillotine cutting problem, where rectangular pieces are required to be cut from only one rectangular object of material. Each piece has a defined value, so the objective is to maximize the total value of cut pieces. In order to solve this problem, an evolutionary option to obtain an optimal solution is proposed which is hybridized with our heuristic [5]. The last one made an efficient management of free spaces closed by yet assigned pieces. As infeasible solutions are generated for the EA, penalty technique was added to the objective function. Moreover different methods to select parents were incorporated in this evolutionary algorithm. The main objective of this work is to determine the best combination of penalty strategy and selection methods for the problem under consideration.

The organization of the paper is as follows. Section II mentions the different approaches proposed in the literature to solve the general two-dimensional cutting problem. Meanwhile Section III describes the constrained two-dimensional non-guillotine cutting problem and the notation adopted in this research. Section IV describes the solution representation that is used. The following section details the heuristic placement routine. In Section VI the penalty techniques are described. Section VII summarizes and extends the results in order to analyze the evolutionary algorithm proposed in this work. Finally, concluding remarks and future works are presented.

II. SOLUTION APPROACHES

A first approach on one dimension cutting problem based on linear programming was proposed by Gilmore and Gomory [10]. In this model each column of the constraint matrix corresponds to a feasible cutting pattern of a single stock length. The total number of columns is very large in practical instances so that only a subset of columns/variables can be handled explicitly. The continuous relaxation of this model was solved using the revised simplex method and heuristic rounding was applied to obtain feasible solutions. Under conditions of cyclic regular production, the solution of the continuous relaxation is optimum for a long-term period because fractional frequencies can be transferred/stored to the next cycle. Heuristic rounding of continuous solutions produces an integer optimum in most cases.

The *unconstrained* two-dimensional *n*-stage cutting problem was introduced in [10], where an exact dynamic programming approach was proposed. The problem has since received growing attention because of its real-world applications. Meanwhile for the *constrained* version, a few approximate and exact approaches are known: [2, 12, 13, 14, 16, 18, 20]. Among exact algorithms, the most successful are linear programming based enumeration without column generation [16] and non-linear programming based enumeration [14].

Combinatorial heuristics are usually called heuristics, which do not use any linear programming relaxation of the problem. Many contributions consider evolutionary algorithms for one-dimensional cutting problem (see [8, 23]) and also for two-dimensional cutting problem [20].

Another group of approaches is based on the structure of the problems. To begin, the various kinds of First-, Next-, and Best-Fit algorithms with worst-case performance analysis are mentioned [22]. The sequential approaches [11, 15, 19] for one-dimensional CP and Wang's combination algorithm [21] and the *extended substrip generation algorithm* (ESGA) of Hifi [13] two-dimensional CP are more sophisticated. Beasley presents a heuristic algorithm for the constrained two-dimensional non-guillotine cutting problem in [4]. This algorithm is a population heuristic, where a population of solutions to the problem is progressively evolved and the heuristic is based on a new, non-linear, formulation of the problem. This last problem is also solved by Beraudo et. al in [5], here an evolutionary algorithm is used and also different placement heuristics are analysed, one of them is considered here with slight modifications.

III. CONSTRAINED TWO-DIMENSIONAL NON-GUILLOTINE CUTTING PROBLEM

The constrained two-dimensional non-guillotine cutting problem is the one that refers to cutting smaller pieces from a single large planar stock rectangle also called object. Each planar stock rectangle has fixed dimensions (L_0 ,W_0) where L_0 is the length and W_0 is the width, whereas each type of pieces i has a length L_i and a width W_i (i=1,...,m), considering m as the number of different types of pieces. Each type of piece i has fixed orientation (i.e. cannot be rotated). It must be cut (by infinitely thin cuts) with its edges parallel to the edges of the stock rectangle (i.e. orthogonal cuts); and the number of pieces of each type i that are cut must lie between P_i and Q_i ($0 \le P_i \le Q_i$). Each type of piece i has an associated value v_i and the objective is to maximize the total value of the cut pieces. It is usual to assume that all dimensions (L_i ,W_i) i=0,1,...,m are integers without entering into a significant loss of generality. To ease the notation in this paper we shall use $M = \sum_{i=1}^{m} Q_i$ [4] which represents the total number of pieces.

IV. CHROMOSOME REPRESENTATION

The possible solutions to this problem can be encoded inside the chromosome structure. There are many ways to do this, being some better than others. For the constrained two-dimensional non-guillotine cutting problem Beasley [4] proposed a combined binary/real–numbered solution representation. In this work only the binary part is used:

z_{ip} { = 1 if the p'th copy (p=1,…,Q_i) of type of piece i is cut from (L_0, W_0).
= 0 otherwise.

In order to show an example of this representation, the following problem is considered: m = 3, M=5, L_0=10, W_0=10, L_1=3, W_1=7, Q_1=2, L_2=10, W_2=2, Q_2=1, L_3=8, W_3=2, Q_3=2. A possible chromosome conformation is as follows:

Piece (i)	1	1	2	3	3
Copy (p)	1	2	1	1	2
z_{ip}	0	1	1	0	1

In this example the first copy of piece 1 is not cut (z_{11}=0), while the second copy of piece 1 is cut (z_{12}=1). Other pieces to be cut are the only copy available of piece 2 and the second copy of piece 3.
Piece and Copy lists are used as reference to interpret the binary part of all chromosomes. They are maintained unchanged during all the run. The Piece list is built considering all the pieces involved in a particular instance, each piece is numbered and that number is repeated depending on Q. The Copy list enumerates the copies for each piece.

V. HEURISTIC WITH EFFICIENT MANAGEMENT OF HOLES

The quality of the layout depends on the sequence in which the rectangles are presented to the heuristic placement routine. The task of the heuristic is to search for a good ordering of the items. A placement routine is then needed to interpret the chromosome and evaluate its quality.
The Heuristic with Efficient Management of Holes were developed in [5]. At the beginning the pieces, with z_{ip}=1, are ordered by their width. Then, to accommodate a piece, the heuristic acts in the following way. The first piece is put at the bottom-left border (or corner). After that, the following piece is taking into account:

Case 1: The heuristic tries to accommodate this piece over already placed pieces in the left margin. If it is not possible, the heuristic continues with case 2.

Case 2: The heuristic tries to locate the piece in a free hole. This hole must have an adequate size and it must be located in the leftmost. If it is not possible, the heuristic continues with case 3.

Case 3: The heuristic tries to locate the piece in the bottom border of the object at the right of already placed pieces.

Case 4: In other case the piece cannot be cut in any other possible way.

If there are more pieces to accommodate, the heuristic comes back to first case. Note that a hole is a free space on the object, which is delimited by 2 or more pieces. See Figure 1 for an example.
This heuristic, as many others, can generate solutions where pieces with z_{ip}=1 could not be placed on the same object. In this way, infeasible solutions are produced and a penalization of these solutions is necessary. Note that, here the objective is to maximize the total value of cut pieces. Different penalization techniques are described in the next section.

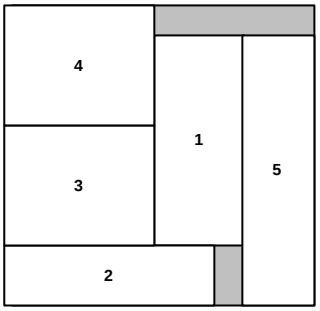

Figure 1. Obtained layout applying the heuristic .

VI. PENALTY FUNCTION

The penalty approach considers infeasible solutions from search space. In an optimization problem, the search space has two parts: a feasible area and an infeasible area. The main idea of the penalty technique is in the design of penalty function. This function should guides the genetic search to promising solution space, in an effective way.

The penalty value corresponds to the amount of its infeasibility under some measurement. There is no general guideline on designing penalty function, and constructing an efficient penalty function is quite problem-dependent [9] . An evaluation function would be written as:

$$Eval(x)=F(x) \pm Pen(x)$$

Where x represents a chromosome, $F(x)$ the objective function of problem, and $Pen(x)$ the penalty term. For maximization problems, the following is considered:

$$\begin{cases} Pen(x)=0 & \text{if } x \text{ is feasible} \\ Pen(x)<0 & \text{otherwise} \end{cases}$$

Michalewicz propose the following function to calculate the fitness:

$$Eval(x)= \sum_{i=1}^{m} z_{ip} v_i \; - Pen(x),$$

where $z_{ip}=1$, represents the chosen piece for cutting; v_i is an associated value for each piece.

For this working three different kinds of penalty functions, $Pen(x)$, have been implemented considering the proposed ones by Michalewicz for the *Knapsack Problem* [17]:

Linear Penalty: $Pen(x) = \rho \sum_{i=1}^{m} z'_{ip} \; a_i$

Quadratic Penalty: $Pen(x)= (\rho \sum_{i=1}^{m} z'_{ip} \; a_i)^2$

Logarithmic Penalty: $Pen(x)=\log_2 (1+ \rho \sum_{i=1}^{m} z'_{ip} \; a_i)$

Where $\rho= \max_{i=1..m} [v_i /a_i]$, $z'_{ip} = 1$ when $z_{ip} = 1$ and the piece cannot be cut and a_i is the surface of piece i.

VII. EXPERIMENTAL TESTS AND RESULTS

We used a steady-state evolutionary algorithm, which objective is to maximize the fitness. That is to maximize the total value of cut pieces by applying a penalty function. This algorithm uses: binary solution representation [5], the uniform crossover and big-creep mutation. The created child ever should replace an individual from the population; the individual to be replaced is selected randomly from the ten worst individuals. Two diferent methods to select parents are used, one of them is the Binary Tournament and another is the Roulette Wheel method. For using the last method is necessary scaling the fitness because sometimes the fitness is negative. The used technique for scaling is the Normalizing technique, which was proposed by Cheng and Gen in [9]. For a maximization problem, it takes the following form:

$$f'(x) = (f(x) - f_{min} + g) / (f_{max} - f_{min} + g)$$

where f_{max} is the maximum fitness, f_{min} is the minimum fitness, $f(x)$ is the fitness for individual x, $f'(x)$ is the scaled fitness for individual x and g is a small positive real number which is usually restricted within the open interval (0, 1). For those tests the g value is equal to 0.5, which was obtained after to experiment with different values (for example 0.05, 0.5, 0.9).

The algorithms were tested for 14 instances of the CP from OR-Library [1, 3]. For each instance, a series of fifty runs was performed. The maximum number of steps was fixed at 15000 and probabilities for crossover and mutation were set at 0.6 and 0.1, respectively. The population size was fixed at 120 individuals. Those parameter values were set after many proof trials and they are summarised in Table 1.

Evolutionary Algorithm	Steady State Genetic Algorithm
Trials for each instance	50
Max. # steps	15000
Chromosome Representation	Binary
Crossover Operator	Uniform, Pc=6%
Mutation Operator	Big-creep, Pm=1%
Population Size	120
Selection method for mating	Binary Tournament and Roulette Wheel
Selection method for insert the new child	Random selection from the ten worst individuals

Table 1: parameter value's details of steady state genetic algorithm

In Table 2, for each test problem, the size of the problem, the optimal solution, the best values found in the literature [4, 5], the results found by our proposed algorithms are shown. Last row of Table 2 shows average results over all instances obtained by each algorithm. The boldface values indicate that the algorithm found the optimal for the respective instance. Regarding this, when the Linear and Logarithmic penalty (independently of selection method) are used, the arisen algorithms find the optimum in a bigger number of instances (for example 9 of 14 for Linear and 10 of 14 for Logarithmic using in both cases Roulette Wheel Selection) than the approach applying Quadratic penalty (for example 8 of 14 under Roulette Wheel Selection). Comparing the results obtained by the algorithms proposed here with the ones appeared in the literature the following remarks can be observed: (a) more instances (72% of them) are optimally solved than in Beraudo's paper [5] (36% of them) and (b) the optimum for instance 12 is only reached by the algorithm combining Logarithmic penalty with Roulette Wheel Selection; while others could not find it, even though Beasley's algorithm obtained the optimum for the rest of the instances. In Figure 2, as an example of the found optimal solution, the individual (chromosome and fitness) and the corresponding layout are shown.

Table 3 shows the percentage error of the best individual, found by each algorithm, in relation with the optimum value for the respective instance. Last row, in this table, summarises average results over all instances obtained by each algorithm. In average the algorithms using Linear penalty made smaller errors (1.5%) than others approaches (1.83% for Logarithmic penalty and 2.6% for Quadratic penalty); being the algorithms with Quadratic penalty which produce the biggest error. That means, in average the smallest distance between the best solution and the optimum is caused by the use of linear penalty.

Figure 3 shows a percentage of the number of times that each algorithmic approach obtains the optimal value. Note that, this percentage is calculated taking into account fifty runs of the same algorithm for each instance.

	Problem Size			Opt.	Berau-do et.al [5]	Beas-ley [4]	Tournament Selection			Roulette Wheel with Normalizing Scaling		
Inst.	(L_0,W_0)	m	M				Linear Pen.	Logarit. Pen.	Quad. Pen.	Linear Pen.	Logarit. Pen.	Quad. Pen.
1	(10,10)	5	10	164	***164***	***164***	***164.00***	***164.00***	***164.00***	***164.00***	***164.00***	***164.00***
2	(10,10)	7	17	230	***230***	***230***	***230.00***	***230.00***	***230.00***	***230.00***	***230.00***	*227.00*
3	(10,10)	10	21	247	*233*	***247***	*217.00*	*200.00*	*212.00*	*222.00*	*215.64*	*231.00*
4	(15,10)	5	7	268	***268***	***268***	***268.00***	***268.00***	***268.00***	***268.00***	***268.00***	***268.00***
5	(15,10)	7	14	358	***358***	***358***	***358.00***	***358.00***	***358.00***	***358.00***	***358.00***	***358.00***
6	(15,10)	10	15	289	*273*	***289***	*282.00*	*283.49*	*273.00*	*289.00*	*284.46*	*289.00*
7	(20,20)	5	8	430	*426*	***430***	***430.00***	***430.00***	***430.00***	***430.00***	***430.00***	***430.00***
8	(20,20)	7	13	834	*828*	***834***	*828.00*	*828.00*	*828.00*	*828.00*	*828.00*	*828.00*
9	(20,20)	10	18	924	*871*	***924***	*863.00*	*863.00*	*853.00*	*863.00*	*863.00*	*863.00*
10	(30,30)	5	13	1452	*1383*	***1452***	***1452.00***	***1452.00***	*1383.00*	***1452.00***	***1452.00***	*1383.00*
11	(30,30)	7	15	1688	***1688***	***1688***	***1688.00***	***1688.00***	*1570.00*	***1688.00***	***1688.00***	***1688.00***
12	(30,30)	10	22	1865	*1826*	*1801*	*1841.00*	*1842.43*	*1855.00*	*1841.00*	***1865.00***	*1749.00*
13	(30,30)	15	15	1270	-	***1270***	***1270.00***	*1264.87*	*1237.00*	***1270.00***	***1270.00***	*1221.00*
14	(30,30)	7	7	1178	-	***1178***	***1178.00***	***1178.00***	***1178.00***	***1178.00***	***1178.00***	***1178.00***
			Average	799.79	712.33	795.21	790.64	789.27	774.21	791.50	792.44	776.93

Table 2: Comparison among the best values found by other authors and by our proposed algorithms

	Tournament Selection			Roulette Wheel with Normalizing Scaling		
Instance	Linear Pen.	Logarit. Pen.	Quad. Pen.	Linear Pen.	Logarit. Pen.	Quad. Pen.
1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
2	0.00%	0.00%	0.00%	0.00%	0.00%	1.30%
3	12.15%	19.03%	14.17%	10.12%	12.70%	6.48%
4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
6	2.42%	1.91%	5.54%	0.00%	1.57%	0.00%
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
8	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%
9	6.60%	6.60%	7.68%	6.60%	6.60%	6.60%
10	0.00%	0.00%	4.75%	0.00%	0.00%	4.75%
11	0.00%	0.00%	6.99%	0.00%	0.00%	0.00%
12	1.29%	1.21%	0.54%	1.29%	0.00%	6.22%
13	0.00%	0.40%	2.60%	0.00%	0.00%	3.86%
14	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Average	1.66%	2.13%	3.07%	1.34%	1.54%	2.14%

Table 3: percentage error of the best-found solution in comparison with the optimum

Piece (i)	1	1	1	2	3	4	4	4	5	5	5	6	6	6	7	7	8	8	8	9	10	10
Copy (p)	1	2	3	1	1	1	2	3	1	2	3	1	2	3	1	2	1	2	3	1	1	2
z_{ip}	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**

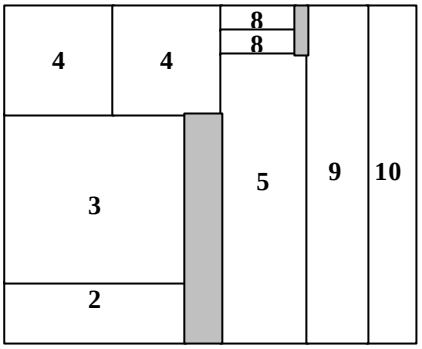

Figure 2: Chromosome of the optimum for instance 12 and its corresponding layout.

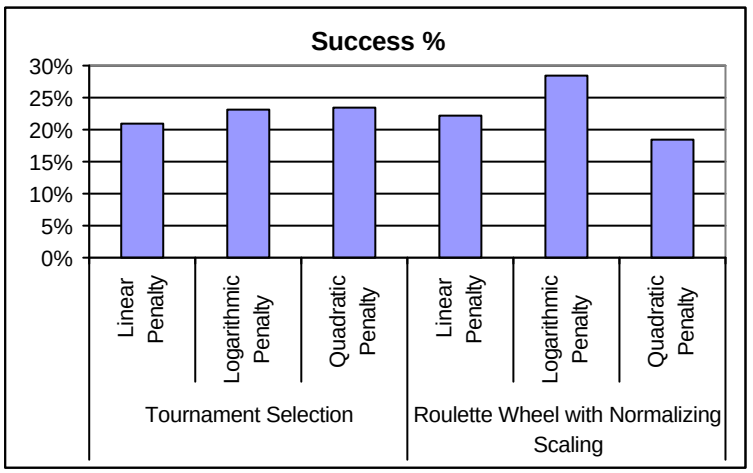

Figure 3: Success percentage of our proposed algorithms.

In general, comparing the results obtained by our algorithms the following remarks can be seen. The selection methods to form the mating pool do not present significant differences when the best solutions are analysed. Now regarding the penalty functions, the results obtained by linear and logarithmic penalties are better than the results reached by the quadratic one. Analysing both tables, the logarithmic option finds the optimum more times but if this option cannot do it, the commit percentage error is bigger than in the linear option.

Being the objective to study if there are or not differences in the quality of distinct set of solutions given by approaches here used, an analysis of variance of nonparametric classification was made. Since the assumption on the normality and the homogeneity of the variances was impossible to verify, the Kruskal-Wallis ANOVA by Ranks test was used. The null hypothesis (H0) is: there are not any differences among average final population values obtained by all the algorithms with α=0.05. As the calculated *p-value* is equal to 0 (minor than α), the H0 is rejected. As the average quality of these sets is different, a multiple comparison was done concluding that three different algorithm groups are detected regarding the penalty function independently of the selection method.

VIII. CONCLUDING REMARKS

For the non-guillotine cutting problem, we presented a solution using evolutionary algorithms with penalty function in order to manipulate infeasible solutions during the evolution process. Also the influence of different selection methods was analyzed in the EA behaviour. Three penalty functions were proposed for this problem originally developed for the knapsack problem: linear, quadratic and logarithmic. The last one presented a better behaviour, which can be seen for the good quality of solutions achieved for the algorithm. About the selection methods used they did not show a significant differences on its behaviour on the same penalty function. Furthermore we can remark that the improvement in the result was only associated to the application of penalty functions independently of the selection methods used.
Our next steps will consist in analyze the applications of others techniques to handle the infeasible solutions present in all the constraints problems such as repairing or rejecting those solutions. Also analyzing the algorithms performances on instances of largest dimensions is an important issue to tackle in the future.

ACKNOWLEDGEMENTS

We acknowledge the cooperation of the project group for providing new ideas and constructive criticisms. Also to the Universidad Nacional de La Pampa, and the ANPCYT from which we receive continuous support.

REFERENCES

[1] Beasley J.E., "An exact two-dimensional non-guillotine cutting tree search procedure", *Journal of Operations Research* Vol 33, pp. 49-64, 1985.

[2] Beasley J. E., "Algorithms for unconstrained two–dimensional guillotine cutting", *Journal on Operation Research Society*, Vol 36, pp. 297–306, 1985.

[3] Beasley J.E., "OR-Library: distributing test problems by electronic mail", *Journal of the Operational Research Society*, Vol. 41, pp. 1069-1072, 1990.

[4] Beasley J.E., "A Population Heuristic for Constrained Two Dimensional Non Guillotine Cutting". *European Journal of Operational Research*, Vol. 156, pp. 601-627, 2004.

[5]Beraudo V., Alfonso H., Minetti G., Salto C., Constrained Two-Dimensional Non-Guillotine Cutting Problem: An Evolutionary Approach, *Proc. of XXIV International Conference of the Chilean Computer Science Societty (SCCC2004)*, IEEE Press, pp. 84-92, 2004.

[6] H. Dyckhoff, "A typology of cutting and packing problems", *European Journal of Operational Research* Vol. 44, pp. 145–160, 1990.

[7] H. Dyckhoff and U. Finke, *Cutting and packing in production and distribution*, Physica Verlag, Heidelberg, 1992.

[8] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", *Journal of Heuristics* Vol. 2, pp. 5–30, 1996.

[9] Gen M., Cheng R., Genetic Algorithms & Engineering Desing, EDA, 1997.

[10] Gilmore P. C. and Gomory R. E., "The theory and computation of knapsack functions", *Journal on Operation Research*, Vol. 14, pp. 1045–1075, 1966.

[11] Haessler R. W., "Controlling cutting pattern changes in one-dimensional trim problems", *Journal on Operations Research***,** Vol. 23, pp. 483–493, 1975.

[12] Hifi M., "Exact algorithms for large-scale unconstrained two and three staged cutting problems", *Computational Optimization and Applications* Vol. 18, pp. 63–88, 2001.

[13] Hifi M. and R. M'Hallah, "Strip generation algorithms for constrained two-dimensional two-staged cutting problems", working paper, 2003.

[14] Hifi M. and Roucairol C., "Approximate and exact algorithm for constrained (un)weighted two-dimensional two-staged cutting stock problems", *Journal of Combinatorial Optimization*, Vol. 5, pp. 465–494, 2001.

[15] Kupke J., *Lösung von ganzzahligen Verschnittproblemen mit Branch-and-Pric*e, Diplomarbeit, Universit at zu Köln, 1998.

[16] Lodi A. and Monaci M., "Integer linear programming models for 2-staged two-dimensional knapsack problems", *Mathematical Programming*, Ser. B Vol. 94, pp. 257–278, 2002.

[17] Michalewicz Z., Genetic Algorithms + Data Structure = Evolution Programs, Springer, 3º edition, 1999.

[18] Morabito R. and Arenales M., "Staged and constrained two-dimensional guillotine cutting problems: A new approach", Tech. report, Notas do ICMSC 126, Universidale de Sao Paulo, Sao Carlos, S.P., Brazil, 1992.

[19] Mukhacheva E. A. and Zalgaller V. A., "Linear programming for cutting problems", *International Journal of Software Engineering and Knowledge Engineering*, Vol. 3, pp. 463–477, 1993.

[20] Parada V., Palma R., Sales D., and Gómes A., "A comparative numerical analysis for the guillotine two-dimensional cutting problem", *Annals of Operations Research* Vol. 96, pp. 245–254, 2000.

[21] Wang P. Y., "Two algorithms for constrained two-dimensional cutting stock problems", *Journal on Operation. Research*, Vol. 31, pp. 573–586, 1983.

[22] Yue M., "A simple proof of the inequality, for the FFD bin-packing algorithm", *Acta Math. App. Sinica* Vol. 7, pp. 321–331, 1991.

[23] Zelle C. and Burkard R. E., "A local search heuristic for the reel cutting problem in paper production", SFB Report No. 257, Institute of Mathematics B, Technical University Graz, 2003.