A comparative numerical analysis for the guillotine two-dimensional cutting problem

Victor Parada [a,*], Rodrigo Palma [a], Daniel Sales [b] and Arlindo Gómes [c]

[a] *Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Av. Ecuador 3659, Santiago, Chile*
E-mail: vparada@diinf.usach.cl

[b] *Departamento de Ingeniería Eléctrica, Universidad de Santiago de Chile, Av. Ecuador 3659, Santiago, Chile*

[c] *Departamento de Informática, Universidad Federal de Espírito Santo, Campus Goiabeiras s/n, Vitoria, Brazil*
E-mail: agomes@inf.ufes.br

In this work, the behavior of four algorithms in the resolution of the two-dimensional constrained guillotine cutting problem is analyzed. This problem is concerned about the way a set of pieces should be cut from a plate of greater dimensions, considering guillotine cutting and a constrained number of times a piece can be cut from the plate. In this study three combinatorial and two heuristic methods are considered. In the combinatorial methods from the set of pieces, a minimum loss layout is constructively generated based on Wang's algorithm. In addition, an evolutionary and an annealing type approach are considered. All of these models have been implemented on a high performance Silicon Graphics machine. Performance of each algorithm is analyzed both in terms of percentage waste and running time. In order to do that, a set of 1000 instances are classified according to their combinatorial degree and subsequently evaluated.

Keywords: guillotine cutting stock, heuristics, combinatorial optimization

1. Introduction

In this paper we have focused our attention on the industrial area, particularly on a situation known as the Cutting Stock Problem which belongs to the family of NP-hard problems. This problem has been widely studied in the last two decades, due to its great applicability in several industries, such as in steel, paper, glass and wood industries. Both a review and classifications of the several kinds of problems of this family can be found in studies of Hinxman [6] and Dyckhoff [2]. In a cutting problem two aspects should be determined: the position in which cuts should be made in order to obtain the pieces and the resulting trim loss from the cutting process. Typically, the former correspond to a set of constraints and the second to the objective function when the problem is represented by means of a mathematical model. Thus, a solution is given by a cutting pattern.

* Corresponding author.

We are interested in a particular case known as the two-dimensional constrained guillotine cutting problem. Consider a rectangular plate of length L and width W, from which we want to cut a set of smaller pieces $p_1, p_2, \ldots, p_n$, with dimensions (w_i, l_i) such that $w_i \leqslant W$ and $l_i \leqslant L$, $\forall i \in R$, where R is a set of types of pieces. In addition, a limit b_i is considered for the number of times in which a certain piece could be cut from the sheet. Guillotine cuts must be executed orthogonally from side to side of the plate, or a part of it. A pattern specifying the position of each cut to be executed should be generated. The problem can be partially formulated as in (1).

$$\begin{aligned} \text{Min } Z(x) &= WL - \sum w_i l_i x_i \qquad (1)\\ \text{subject to} \quad &\text{(i) } 0 \leqslant x_i \leqslant b_i, \quad \forall i \in R,\\ &\text{(ii) } x_i \text{ is an integer}, \quad \forall i \in R,\\ &\text{(iii) cuts are feasible,} \end{aligned}$$

where, x_i is the number of times that a piece of type i is in the pattern to be cut. In this formulation, two aspects are involved in (iii): all cuts must be guillotine type and no overlapping among pieces is accepted.

Christofides and Whitlock [1] propose an algorithm of resolution for this problem, based on the branch-and-bound method. This method also considers the pioneer work of Gilmore and Gomory [4] that solves the problem by means of dynamic programming. Wang [16] proposed an algorithm that incrementally generates a cutting pattern by analyzing all possible combinations of pieces. Vasko [14] and Oliveira and Ferreira [9] propose an improvement for the algorithm of Wang. On the other hand, Hinxman [5] and later Morabito et al. [8] carry out an approach to the problem using methodologies from intelligent search. Similar ideas are considered by Parada et al. [10] and Viswanathan and Bagchi [15] using a problem based on and/or graphs [13]. Another representation is presented by Parada et al. [11,12] using both genetic algorithms and simulated annealing.

In this work, the computational behavior of a group of algorithms – the method of Wang [16], that of Oliveira and Ferreira [9] and the methods of Parada et al. – is analyzed. Problem instances are grouped in families and the corresponding cutting patterns are generated.

In section 2, the methods used are briefly described. In section 3, computational results are analyzed. Finally, in section 4, the conclusions of the study are shown.

2. Methods

2.1. Method of Wang (WA)

The algorithm of Wang [16] generates a constructive solution by means of the union of rectangles that represent pieces or combinations of pieces. In order to decrease the number of partial solutions – that is, the number of combinations – previously fixed acceptance parameters are used (algorithm 1).

Algorithm 1.

Begin
Choose a value for β, $0 \leqslant \beta \leqslant 1$;
Define $L^{(0)} = F^{(0)} = \{p_1, p_2, \ldots, p_n\}$
$k = 0$
While $F^{(k)} \neq \phi$
$k = k + 1$;
Determine $F^{(k)}$ which is the set of all rectangles $T = \{R_1, R_2, \ldots, R_n\}$ satisfying:
(i) T is a combination of rectangles belonging to $L^{(k-1)}$, joined in vertical or horizontal form,
(ii) the total waste of each R_i is less than or equal to $\beta_1 HW$ and
(iii) the rectangles R_i belonging to T do not surpass the limits b_i of each piece
Set $L^{(k)} = L^{(k-1)} \cup F^{(k)}$, eliminating equivalent patterns;
Choose the rectangle of $L^{(k)}$ whose total waste is minimum
End.

2.2. Modified Wang method (WM)

Oliveira and Ferreira [9] present an improvement to the Wang method modifying the acceptance criteria used originally by Wang. In addition, an internal loss is incurred when combining two rectangles p_1 and p_2 vertically or horizontally, such as is shown in figure 1(a). An external loss, estimated by assigning pieces in the area still empty, is also defined as in figure 1(b). For this situation, the method of Gilmore and Gomory [4] is used. In this manner, the total loss (internal plus external) is used in order to accept or reject a configuration.

2.3. Additive And/Or method (AAO)*

The *AAO** method carries out a search on an and/or graph representing the problem [10]. At each node of this graph horizontal and vertical rectangle combinations are represented and an evaluation function $f = g + h$ guides the search process on this graph. A value for the function g is obtained by adding the successive cost obtained

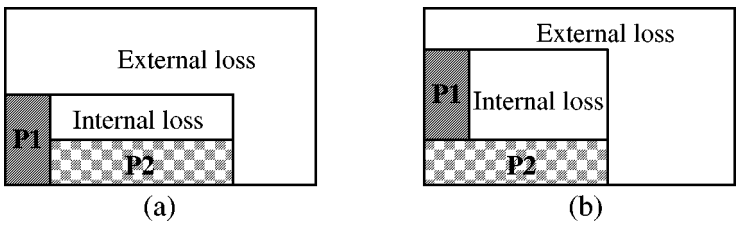

Figure 1. Internal and external loss for horizontal and vertical rectangles.

during the generation of previous nodes, while h is a heuristic function that estimates how much is lacking in order to generate a goal node. Specifically, g is determined by adding the internal loss of the two combined rectangles, while h is associated with the external loss, estimated in several ways.

2.4. Simulated Annealing method (SA)

The *SA* method, proposed by Kirkpatrick et al. [7] (algorithm 2), has been thoroughly used in recent years in several optimization problems [3]. In the study of Parada et al. [12] a solution layout has been represented by means of a binary tree, such a structure not only facilitates the generation of neighbor solutions, but it is also adequate to represent the problem under other methods. The evaluation function corresponds to the loss involved in the corresponding layout. Parameters have been exhaustively calibrated in order to determine a set that provides better solutions. Specifically, several decreasing rates for parameter T have been considered and the number of iterations in the internal cycle has been widely studied as well. The best value for T_0 has also been numerically selected.

Algorithm 2.

Begin
  Find an initial solution i and an initial value for T_0;
  $t = 0$;
  Repeat
    $n = 0$;
    Repeat
      Generate solution j neighbor to i;
      $d = f(j) - f(i)$;
      If $d \leqslant 0$ *then* $i = j$;
        Else If random$(0, 1) < \exp\{-d/T\}$ *then* $i = j$;
      $n = n + 1$;
    Until $n = N(t)$;
    $t = t + 1$; $T = T(t)$;
  Until stop criterion achieved;
End.

The stop criterion is defined as a maximum number of iterations, or as a number of iterations that occurs without any improvement in the objective function.

2.5. Evolutionary Algorithm (GAO)

A feasible cut is represented by a syntactic binary tree, ensuring by this means the guillotine requirement for the cut. Each intermediary node in the tree describes a vertical or horizontal cut, while the leaves represent the pieces to be cut. By assigning

characters to the nodes and performing a depth-first search over the tree, the string obtained characterizes a partial solution [11].

With this string as a basis, a population may be built, and thus a set of partial solutions defined. By combining those solutions step by step, an incremental evolutionary process is originated, in the same manner as a genetic algorithm.

Algorithm 3 describes the evolutionary process. Here, $A(t-1) \times B(t-1)$ corresponds to the crossover between elements selected from the subsets A and B, present in step $t-1$.

The selection process is made with consideration to the quality of each individual, in terms of the evaluation function. Every time that the crossover takes place in order to generate a new individual, by concatenating two strings, the generated individual is analyzed to check its feasibility. Thus the growth of the string will not be infinite.

Algorithm 3.

Begin
  $B(0) \leftarrow$ Initial set of individuals;
  $A(0) \leftarrow \varnothing$; $t = 0$;
  Repeat
    $t = t + 1$;
    $C \leftarrow \varnothing$;
    $C \leftarrow$ Selection of the best individuals from the feasible crossover between $A(t-1) \times B(t-1)$ and $B(t-1) \times B(t-1)$;
    $B(t) \leftarrow C$;
    Update $A(t)$;
  Until total loss $= 0$ Or no more possible crossovers exist;
End.

3. Computational results

In order to carry out the analysis, 1000 instances of the guillotine cutting problem were randomly generated by varying plate dimensions, the number of different pieces and their respective dimensions and limits. After that, such instances were classified according to their combinatorial degree defined in (2) and (3).

$$\lambda = 2^{\gamma-1} \ln(n+1). \tag{2}$$

In (2) n is the number of pieces and $\gamma = \lfloor WL/(w_i^* l_i^*) \rfloor$ indicates the number of times the smallest piece with area $w_i^* l_i^*$ fits on the plate of area WL.

$$\kappa = |\log \lambda|. \tag{3}$$

It is clear that the greater κ, the greater the number of feasible combinations among pieces in order to generate a layout. For the generated instances κ varies from

2 to 38 and instances with the same κ are considered as a family of approximately the same degree of difficulty.

Since the three constructive algorithms are sensitive to β, the recommendations given in [9,10,16] were followed. In such studies β was gradually increased to find the optimum solution for each of the 9 instances. The higher found value 0.08 was used in order to solve the 1000 instances.

SA has 3 parameters influencing its behavior, T_0, $N(t)$ and $T(t)$. Parameters used are based on [12] where the same set mentioned above was used. The only change considered here is the number of iterations in which the current solution can not be improved; this number was increased enabling the algorithm to find a better solution.

In the case of *GAO* two values n and $2n$ for the population size upper bound were tested. The best results were obtained when $2n$ was used, therefore this limit was considered to solve the instances.

We have implemented the above described methods on a Silicon Graphics Challenge computer with two processors of 150 MHz and 256 MB in RAM. In figures 2–4 the obtained results in the resolution of the 1000 instances are shown. On the x-axis of each 3D-graph, different families of instances according to κ are represented and on the z-axis in figure 2 are the algorithms *WA*, *WM*, *AAO**, *SA* and *GAO*.

In figure 2, the fraction of solved problems ρ for each algorithm is represented on the y-axis. It is observed that in problems with low κ, the degree of resolution is near 100% for all the algorithms. As the combinatorial degree increases, the resolution capacity of *WA* decreases, moreover from $\kappa = 29$ ($\lambda \approx 10^{29}$) onwards the algorithm was not able to solve the problem. To have an idea about such a number, one of the instances on this range has 17 different sort of pieces, plate dimensions are 61×98 units and the largest and smallest piece are 58×73 and 60×1 units, respectively. On the other hand, results for *WM* show its bigger capacity to solve the problem when compared with *WA* (see, for example, figure 2 with $\kappa \in [5, 11]$), in fact, such

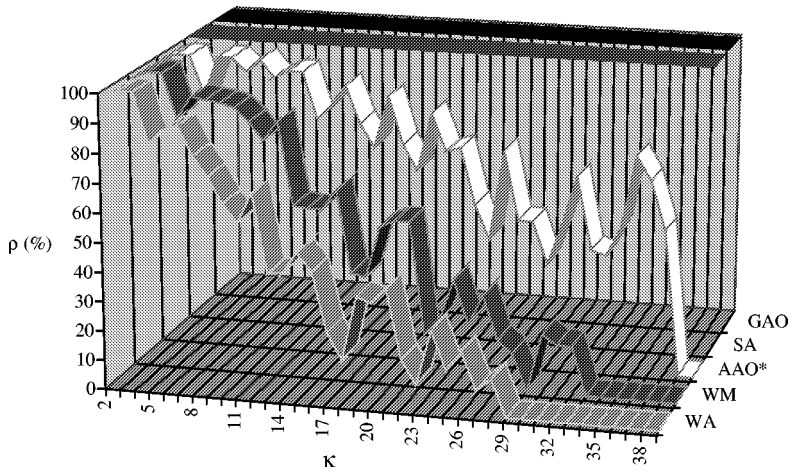

Figure 2. Percentage of solved instances for each algorithm.

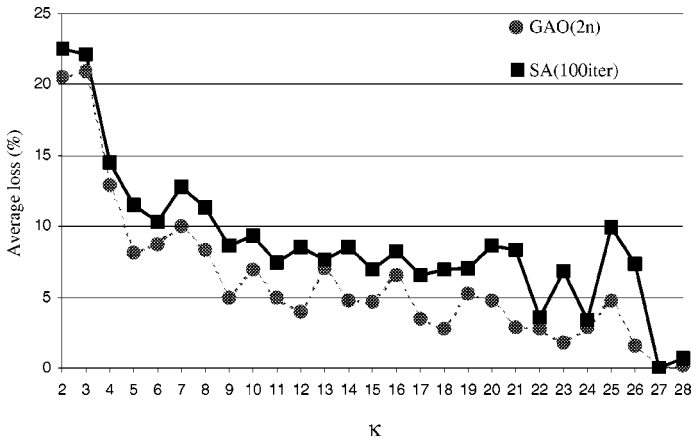

Figure 3. Percentage of loss obtained for each algorithm.

an algorithm is an improvement of *WA*, however, in general terms its behavior is similar. *AAO** has an even better behavior but it is also unable to solve the problem for all the instances considered in this study. From the same figure the general behavior of *SA* and *GAO* can be appreciated and to observe more in detail differences between them we consider figure 3, where the average percentage of loss obtained for each algorithm is shown. The axis y represents the percentage of loss related to the plate area. It can be seen that, upon increasing the combinatorial degree of the family of instances, the loss obtained by both methods decreases. Since instances with small pieces and in a large number have in general lower loss, this tendency was expected. *SA* obtains a slightly higher loss than *GAO* in most of the cases but in some of them (for example: $\kappa = 13, 24, 27, 28$) almost the same quality of solution is found.

In figure 4 the average running time is shown only considering those successfully solved instances for all the algorithms. The running times for the three combinatorial algorithms are represented in figures 4(b), 4(c) and 4(d), respectively and in this same order this running time decreases from the range $(0.081, 330.96)$ seconds for *WA* to the range $(0.072, 10.342)$ seconds for *AAO**. These three algorithms solve almost the same instances so the difference in running time is explained by the different number of generated rectangles during the constructive process of solution.

Since the average running time taken to solve an instance of a given family is less in *SA* than in *GAO* (figures 4(e) and 4(f)) it can be affirmed that *GAO* is better than *SA* in solving the problem but with a bigger computational effort. Specifically for *SA* it can be appreciated that as the combinatorial degree increases, the running time also increases; however, this time should not be an unlimited value for too large problems due to the maximum number of iterations considered as the stop criterion. Even in this case, the running time does not exceed 2 seconds, showing a notable superiority when compared to the other methods. On the other hand, in most of the cases the running time for *GAO* is less than 4 seconds.

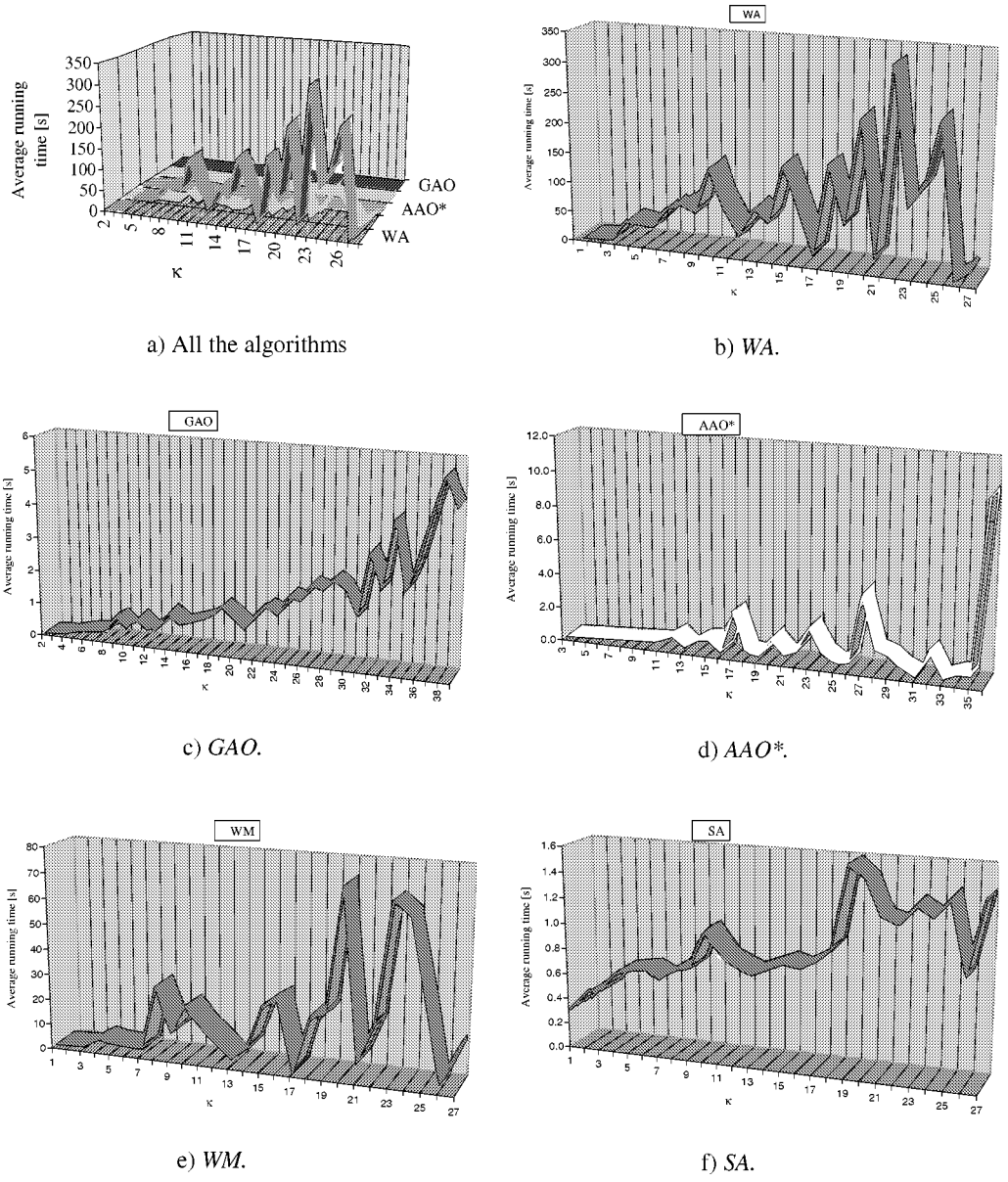

Figure 4. Average running time for each algorithm.

A comparison was also carried out between *GAO* and *SA*, considering all instances, and not only those resolved by all the methods. The results showed that, as the complexity grows, the running time of *GAO* reaches twice that of *SA*; however, the quality of the solutions given by *GAO* are notably better than those given by *SA*.

In order to build a computational system to solve this problem for a practical situation, an also practical running time must be considered, in this sense, in table 1

Table 1
Average running time.

	Small instance	Medium instance	Large instance
WA	Low	High	High
MW	Low	Medium	Medium
AAO*	Low	Low	Low
SA	Low	Low	Low
GAO	Low	Low	Low

the running time is considered as *low*, when it is less than 5 seconds; *medium*, when it is between 5 seconds and 1 minute and *high* when it is greater than or equal to 1 minute. On the other hand, the problem size is considered as *small* when $\kappa \leqslant 4$; *medium* when $5 \leqslant \kappa \leqslant 15$ and large when $\kappa \geqslant 16$. According to this definition for small problems Wang's algorithm using $\beta = 0.08$ is recommended. For medium problems *AAO** using the admissible heuristic function given in [10] is preferable since the optimum solution is guaranteed. Finally, for large practical problems *GAO* is the best algorithm.

4. Conclusions

In this study, several methods have been implemented in order to solve the two-dimensional constrained guillotine cutting problem. Such methods have been applied to a large number of problem instances classified according to their combinatorial degree. In order to do that an index κ has been defined. The evolutionary approach and the simulated annealing method are those whose behavior is more predictable, since they solve all of the instances in a short time with a loss not greater than 20%. In addition, it has been determined that Wang's algorithm is practical in obtaining the optimum solution for problems with a low combinatorial degree. On the other hand, although the resolution degree for instances solved by means of *AAO** and modified Wang is better than that obtained with Wang's algorithm, the percentage of loss for instances with low combinatorial degree is greater in the first two. If they are compared with *GAO* and *SA*, these two are better in terms of the resolution degree.

Finally, comparing *SA* and *GAO*, the only two methods that give solutions for all the instances, the former has a better performance in terms of running time, while *GAO* is better in terms of the quality of the solution and obtaining the optimum loss in most of the studied instances.

References

[1] N. Christofides and C. Whitlock, An algorithm for two-dimensional cutting problems, Operations Research 25 (1977) 30–44.

[2] H. Dyckhoff, A typology of cutting and packing problems, European Journal of Operational Research 44 (1990) 145–159.
[3] E.W. Eglese, Simulated annealing: A tool for operational research, European Journal of Operational Research 46 (1990) 271–281.
[4] P.C. Gilmore and R.E. Gomory, The theory and computation of knapsack functions, Operations Research (1966) 1045–1074.
[5] A.I. Hinxman, Problem reduction and the two-dimensional trim-loss problem, in: *Artificial Intelligence and Simulation,* Summer Conference, Univ. of Edinburgh (1976) pp. 158–165.
[6] A.I. Hinxman, The trim loss and assortment problems: A survey, European Journal of Operational Research 5 (1980) 8–18.
[7] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[8] R.N. Morabito, M.N. Arenales and V.F. Arcaro, An and-or-graph approach for two-dimensional cutting problems, European Journal of Operational Research 58 (1992) 263–271.
[9] J.F. Oliveira and J.S. Ferreira, An improved version of Wang's algorithm for two-dimensional cutting problems, European Journal of Operational Research 44 (1990) 256–266.
[10] V. Parada, A. Gómes de Alvarenga and J. De Diego, Exact solutions for constrained two-dimensional cutting problems, European Journal of Operational Research 84 (1995) 633–644.
[11] V. Parada, R. Muñoz and A. Gómes, *An Hybrid Genetic Algorithm for the Two-dimensional Cutting Problem in Evolutionary Algorithms in Management Applications*, eds. J. Biethahn and V. Nissen (Springer, Berlin, 1995).
[12] V. Parada, M. Sepúlveda, M. Solar and A. Gómes, Solution for the constrained guillotine cutting problem by simulated annealing, Computers Oper. Res. 25 (1998) 37–47.
[13] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving* (Addison-Wesley, 1984).
[14] F.J. Vasko, A computational improvement to Wang's two-dimensional cutting stock algorithm, Computers and Industrial Engineering 16 (1989) 109–115.
[15] K.V. Viswanathan and A. Bagchi, Best-first search methods for constrained two-dimensional cutting stock problems, Operations Research 41 (1993) 768–776.
[16] P. Wang, Two algorithms for constrained two-dimensional cutting stock problems, Operations Research 31 (1983) 573–586.