INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 8 (2001) 3–13

Efficient algorithm for the constrained two-dimensional cutting stock problem

André R.S. Amaral[a] and Mike Wright[b]

[a]*Departamento de Informatica, Universidade Federal do Espirito Santo, 29060-900, Vitória, ES, Brazil*
[b]*Department of Management Science, Lancaster University, UK*

Received 7 July 1999; received in revised form 21 October 1999; accepted 7 January 2000

Abstract

This paper considers the constrained two-dimensional cutting stock problem. Some properties of the problem are derived leading to the development of a new algorithm, which uses a very efficient branching strategy for the solution of this problem. This strategy enables the early rejection of partial solutions that cannot lead to optimality. Computational results are given and compared with those produced by a leading alternative method. These results show that the new algorithm is far superior in terms of the computer time needed to solve such problems.

Keywords: Combinatorial optimization, branch-and-bound algorithms, cutting stock problem

1. Introduction

The two-dimensional cutting stock problem can be stated as follows. Given a rectangular sheet of fixed dimensions and a set of permissible types of rectangle (each possibility being defined by its height and width) which can be cut from it, the problem is to cut the sheet into permissible rectangles in such a way as to minimize waste.

There are many practical applications of this problem in a wide variety of industries including metal, glass, wood, leather, etc. Survey papers relating to this problem include those by Dyckhoff et al. (1988), Haessler and Sweeney (1991) and Hinxman (1980).

Usually a guillotine constraint must be observed. This means that each cut that is made must be from one edge of the sheet being cut to the opposite edge. If there is no limit on the number of rectangles available of any type, the problem is known as unconstrained.

The unconstrained guillotine two-dimensional cutting problem has received attention from several researchers, for example Beasley (1985), Gilmore and Gomory (1965), Gilmore and Gomory (1966) and Herz (1972).

The constrained version of the problem has also been studied by many researchers including Christofides and Whitlock (1977), Christofides and Hadjiconstantinou (1995), Daza et al. (1995), Hifi

and Zissimopoulos (1997), Oliveira and Ferreira (1990), Vasko (1989), Viswanathan and Bagchi (1993) and Wang (1983).

This paper presents a new algorithm for the constrained two-dimensional guillotine cutting problem. It is shown that by deriving some properties of the problem, it is possible to develop an algorithm with a very efficient branching strategy.

Given a stock sheet of dimensions L by H and a set of rectangles $\{r_1, r_2, \ldots, r_i, \ldots, r_n\}$, such that each rectangle r_i has length l_i and height h_i, the objective is to minimize the total waste, i.e.:

$$\text{minimize } L.H - \sum_{i=1}^{n} x_i l_i h_i$$

where x_i denotes the number of times a piece i occurs in the stock sheet.

The following constraints apply:

x_i is a non-negative integer ($i = 1, 2, \ldots, n$);
$x_i \leqslant b_i$, ($i = 1, 2, \ldots, n$), where b_i is the upper bound on the number of times a piece may appear in a solution.

There are also constraints which are more difficult to state mathematically. These concern the fact that the rectangles must all fit within the sheet and that all the cuts are of the guillotine type.

Two different approaches have been proposed for the constrained two-dimensional guillotine cutting problem. Christofides and Whitlock (1977) presented a top-down approach, while a bottom-up approach was proposed by Wang (1983).

1.1. Christofides and Whitlock's method

Christofides and Whitlock presented a depth-first tree search algorithm. All possible cuts that can be made on the stock sheet are enumerated by means of a tree in which branching corresponds to guillotine cuts and nodes represent sub-rectangles obtained through the guillotine cut. An upper bound on the maximum value achievable at each node is obtained by using the two-dimensional Knapsack function of Gilmore and Gomory (1966) and a transportation routine devised by Desler and Hakimi (1969).

1.2. Wang's method

Wang's method is based on the observation that any pattern satisfying the guillotine constraint can be obtained through horizontal and vertical builds of rectangles. Given two rectangles x_1 and x_2 of dimensions (l_1, h_1) and (l_2, h_2), the horizontal build between x_1 and x_2 yields the rectangle $(l_1 + l_2, max\{h_1, h_2\})$ and the vertical build between x_1 and x_2 yields the rectangle $(max\{l_1, l_2\}, h_1 + h_2)$.

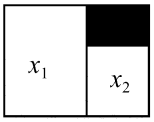

Fig. 1. Horizontal build of two rectangles.

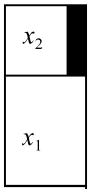

Fig. 2. Vertical build of two rectangles.

Rectangles obtained through a series of vertical and horizontal builds are denominated *guillotine rectangles*. The waste contained inside a guillotine rectangle is called *internal waste*.

When rectangles are combined together this will often cause some internal waste, as in Figure 3. There is also an *external waste* associated with a guillotine rectangle. The external waste is computed as the area of the stock sheet minus the area of a guillotine rectangle, as shown by the area shaded grey in Figure 4.

The sum of the total internal and external waste is the *total waste*, which is the function to be minimized.

Wang successively generates all possible combinations of guillotine rectangles obtaining larger rectangles from smaller ones until no more guillotine patterns can be obtained.

In any stage of this process, in order not to be rejected, any guillotine rectangle generated must satisfy the following criteria:

- it must have dimensions that fit in the stock sheet;
- it must contain not more than b_i pieces of type i;
- it must be combined no more than once with any other guillotine rectangle;
- its total internal waste must not exceed $\beta.H.L$, where $\beta(0 \leqslant \beta \leqslant 1)$ is the maximum allowable waste as a proportion of the whole.

β is a parameter which is set in advance. Increasing β causes the number of partial solutions

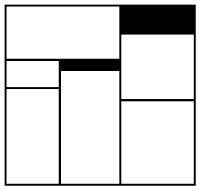

Fig. 3. Internal Waste.

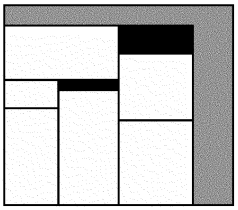

Fig. 4. External Waste.

computed to increase which may then give better solutions. However, the algorithm uses more computer memory and execution time for large values of β.

The algorithm is at its most efficient when β equals β^*, the optimal value of β, i.e. the smallest value of β that allows the generation of at least one optimal guillotine rectangle. Obtaining $\beta = \beta^*$ is usually achieved by incrementing β from zero in steps of 0.01 until its optimal value is reached.

2. The algorithm CSA

Our cutting stock algorithm (CSA) is bottom-up as Wang's but it uses some properties of the problem aiming at early rejection of partial solutions that cannot lead to optimality.

Define **H** and **V** as rectangle operators such that:

$X_1\mathbf{H}X_2$ means the horizontal build between two rectangles X_1 and X_2;
$X_1\mathbf{V}X_2$ means the vertical build between X_1 and X_2.

For any rectangle X, X' is defined as X rotated through 90 degrees (X and X' are regarded as different rectangles in CSA).

Using rotation and horizontal and vertical build gives us eight ways of combining two rectangles.

$$X_1\mathbf{H}X_2 \quad X_1\mathbf{V}X_2$$
$$X_1'\mathbf{H}X_2 \quad X_1'\mathbf{V}X_2$$
$$X_1\mathbf{H}X_2' \quad X_1\mathbf{V}X_2'$$
$$X_1'\mathbf{H}X_2' \quad X_1'\mathbf{V}X_2'$$

However, by noticing that:

$$X_1\mathbf{V}X_2 = (X_1'\mathbf{H}X_2')'$$
$$X_1'\mathbf{V}X_2 = (X_1\mathbf{H}X_2')'$$
$$X_1\mathbf{V}X_2' = (X_1'\mathbf{H}X_2)'$$
$$X_1'\mathbf{V}X_2' = (X_1\mathbf{H}X_2)',$$

it becomes clear that we do not need to use all three operators. Henceforth we shall use operator **H** and the rotation operator only. Although simple, this idea has not apparently been noticed before in the literature.

2.1. Branching strategy

Let $\boldsymbol{R}$ be the total set of all rectangles at any stage of the algorithm. Then for any given rectangle X_1, divide all rectangles in $\boldsymbol{R}$ *whose height is greater than or equal to that of* X_1 into a number of sets $\boldsymbol{B}_0(X_1)$, $\boldsymbol{B}_1(X_1)$, …, $\boldsymbol{B}_k(X_1)$, k being the total number of such sets, in such a way that:

all rectangles in $\boldsymbol{B}_0(X_1)$ have the same height as X_1;
the set $\boldsymbol{B}_i(X_1)$ is not empty;
all rectangles in set $\boldsymbol{B}_i(X_1)$ have the same height as each other, for any i;
rectangles in set $\boldsymbol{B}_{i+1}(X_1)$ have greater height than rectangles in set $\boldsymbol{B}_i(X_1)$, for any i;
within each set $\boldsymbol{B}_i(X_1)$, for any i, the rectangles are ordered by increasing length.

The algorithm will involve successive *horizontal* associations of rectangles X_1 with a rectangle X_2 in a set $\boldsymbol{B}_i$, for any *i*. There is no need to consider associating X_1 with any rectangle X_2 less high than X_1, since this association will be covered when X_2 is considered as the given rectangle. There is also no need to consider vertical associations because of the properties proved earlier. Thus all potential associations are covered in this way.

Assume without loss of generality that $L \geqslant H$. Let the length and height of X_1 be (l_1, h_1). Then the horizontal association between X_1 and any rectangle X_2 in $\boldsymbol{B}_i(X_1)$, for any *i*, with length l_2 and height h_2, results in a rectangle of length $(l_1 + l_2)$ and height h_2 (since $h_2 > h_1$). Since rotation is allowed, this combination will fit into the overall rectangle if and only if $MAX(l_1 + l_2, h_2) \leqslant L$ and $MIN(l_1 + l_2, h_2) \leqslant H$. Otherwise the association is infeasible.

Moreover, the association will also be considered infeasible if the extra internal waste involved $(h_2 - h_1).l_1$ means that the new internal waste thus exceeds $\beta.H.L$.

The point behind the definition and ordering of these rectangle sets is that the detection of an infeasibility automatically implies the infeasibility of a large number of other potential associations, thus making the algorithm very efficient.

2.2. The new algorithm

The new CSA works from a given set of rectangular pieces by sequentially combining these pieces to make further rectangles, and so on until no further feasible rectangles can be produced. The power of the algorithm concerns the order in which rectangles are combined, which allows for the early rejection of unproductive partial solutions.

Step 1. Choose a value for β. Go to Step 2.

Step 2. Initialise $\boldsymbol{R}$ as the set of allowable single rectangle types and their rotations. (In our implementation, the rectangles in $\boldsymbol{R}$ are ordered by increasing height. Rectangles of the same height are ordered by increasing length.) Go to Step 3.

Step 3. Choose a rectangle $X_1 \in \boldsymbol{R}$. Let the length and height of X_1 be l_1 and h_1 respectively.

Step 4. Construct the ordered rectangle sets $\boldsymbol{B}_0(X_1)$, $\boldsymbol{B}_1(X_1)$, …, $\boldsymbol{B}_k(X_1)$ as detailed earlier. Initialize the set $\boldsymbol{S}$ to be the empty set. Initialize $z = 0$. Go to Step 5.

Step 5.

5.1 Associate X_1 horizontally in turn with rectangles X_2 in $\boldsymbol{B}_z(X_1)$. Let the length and height of X_2 be l_2 and h_2 respectively.

5.2 If the extra internal waste involved $(h_2 - h_1)^* l_1$ means that the new internal waste exceeds $\beta.H.L$, the association is infeasible. Moreover, because of the way the rectangle sets were constructed, this will also be true for all other rectangles X_2 in $\boldsymbol{B}_y(X_1)$ for all $y \geqslant z$. Therefore go straight to Step 7.

5.3 If $MAX(l_1 + l_2, h_2) > L$ or $MIN(l_1 + l_2, h_2) > H$, the association is infeasible. Moreover, because of the ordering of rectangles within the rectangle sets, this will also be true for all other rectangles X_2 in $\boldsymbol{B}_z(X_1)$. Therefore go straight to Step 6.

5.4 Otherwise, if the association does not break the constraints concerning the maximum number of pieces allowable of each type, add the new rectangle formed into the set $\boldsymbol{S}$.

5.5 If Step 5 does not terminate early in 5.2 or 5.3 above, it continues until there are no more rectangles X_2 in $\boldsymbol{B}_z(X_1)$. We then move to Step 6.

Step 6. If $z < k$, increment z by 1 and go back to Step 5. Otherwise, proceed to Step 7.

Step 7. If X_1 is not the final rectangle in $\boldsymbol{R}$, then let X_1 be the next rectangle in $\boldsymbol{R}$ and return to Step 4. Otherwise proceed to Step 8.

Step 8. If $\boldsymbol{S}$ is not empty, then $\boldsymbol{R} = \boldsymbol{R} \cup \boldsymbol{S}$ and return to Step 3. Otherwise, proceed to Step 9.

Step 9. Find the rectangle in $\boldsymbol{R}$ with the smallest total waste (internal plus external). If this is no greater than $\beta.H.L$ then this is the optimal solution and the algorithm stops. Otherwise increment β and return to Step 2.

The algorithm also contains a time-saving routine to ensure that there is no repetition in Step 5, i.e., that no equivalent combination of rectangles is associated together more than once.

3. Computational results

In order to evaluate the performance of our CSA, some computational tests have been carried out. The enhanced version of Wang's Algorithm One proposed by Vasko (1989) was implemented (denoted EWA below) and used for comparison purposes. EWA ran, on average, more than 25 times as fast as Wang's algorithm for the data set considered by Vasko (1989). Rotation of pieces was considered in both methods.

The algorithms were implemented in C and run on an HP9000/735 workstation. The algorithms differ in that CSA was conceived to generate all patterns up to an acceptable level ($\beta.H.L$) while EWA generates only the best pattern up to that level. Either algorithm could be easily modified to generate all patterns up to $\beta.H.L$ or only the best pattern. CSA was then modified to generate only the best pattern, so that the methods could be directly compared.

First the algorithms were compared over a set of 10 problems with 30 pieces each. These are derived from the data in Table 2 of Beasley (1985); the method to derive these problems is that described by Viswanathan and Bagchi (1993). For these problems, the smallest value of β that allows the algorithms to verify optimality was used. The results are presented in Table 1. Measuring CPU time on multi-user

Table 1
Results for Beasley's set of 10 problems with 30 pieces each

	Time (sec.)		Number of rectangles generated		Total waste	
	CSA	EWA	CSA	EWA	CSA	EWA
B1	0.032	0.068	20	69	1234	1234
B2	0.031	0.207	101	96	1586	1586
B3	0.048	0.428	162	158	2310	2310
B4	0.047	0.287	124	113	1256	1256
B5	0.048	0.338	116	126	1053	1053
B6	0.071	1.149	262	242	1203	1203
B7	0.104	2.514	369	293	571	571
B8	0.022	0.054	20	68	1041	1041
B9	0.549	32.659	1512	1033	410	410
B10	0.100	3.264	346	350	203	203

and multitasking operating systems is problematic and therefore times provided here for the HP9000/735 should be considered only as an approximation, to within about ±20%.

The algorithms have also been compared using a set of 8 problems presented in Daza et al. (1995). For these problems the value of β was increased from 0, in steps of 0.01, until the optimal solution has been reached. Results are presented in Tables 2, 3 and 4.

From these results it can be seen that CSA is much more efficient than EWA, requiring far less computational effort. This is particularly the case for those problems for which EWA took longest.

We considered further tests with a data set of 120 randomly generated problems kindly provided to us by F. Vasko with permission of Bethlehem Steel.

To generate this data set, Vasko (1989) used a method analogous to Christofides and Whitlock (1977). Three master rectangles, with length 100 and widths 150, 200, and 300, respectively, are considered. For each $n \in \{5, 10, 15, 20\}$, $H \in \{150, 200, 300\}$ and $Qmax \in \{5, 10\}$, he generates a set with 5 instances with the following characteristic:

Stock Sheet: $100 \times H$
P1:$\{r_{11}, r_{21}, \ldots, r_{i1}, \ldots, r_{n1}\}$;
P2:$\{r_{12}, r_{22}, \ldots, r_{i2}, \ldots, r_{n2}\}$;
P3:$\{r_{13}, r_{23}, \ldots, r_{i3}, \ldots, r_{n3}\}$;
P4:$\{r_{14}, r_{24}, \ldots, r_{i4}, \ldots, r_{n4}\}$;
P5:$\{r_{15}, r_{25}, \ldots, r_{i5}, \ldots, r_{n5}\}$;
$1 \leqslant b_{ij} \leqslant Qmax$; $i = 1, \ldots, n$; $j = 1, \ldots, 5$.

The area of each smaller rectangle is given by $k.H.L$, with k randomly generated in the interval (0.05,

Table 2
Execution time (sec.) for Daza et al.'s set of 8 problems

β		0.00	0.01	0.02
D1	CSA	0.154	–	–
	EWA	12.051	–	–
D2	CSA	0.044	0.133	0.431
	EWA	0.319	3.053	37.335
D3	CSA	0.041	0.140	0.507
	EWA	0.098	1.303	23.594
D4	CSA	0.044	0.108	0.388
	EWA	0.173	1.447	23.745
D5	CSA	0.030	–	–
	EWA	0.143	–	–
D6	CSA	0.337	–	–
	EWA	145.961	–	–
D7	CSA	0.056	0.118	–
	EWA	0.445	3.032	–
D8	CSA	0.041	0.068	0.218
	EWA	0.178	1.244	14.226

Table 3
Number of rectangles generated for Daza et al.'s set of problems

β		0.00	0.01	0.02
D1	CSA	573	–	–
	EWA	1758	–	–
D2	CSA	212	573	1677
	EWA	153	499	1690
D3	CSA	110	497	1918
	EWA	98	406	1447
D4	CSA	148	445	1543
	EWA	133	398	1365
D5	CSA	101	–	–
	EWA	162	–	–
D6	CSA	1256	–	–
	EWA	6081	–	–
D7	CSA	239	658	–
	EWA	222	580	–
D8	CSA	131	358	1044
	EWA	131	358	1045

Table 4
Total waste generated by the algorithms for Daza et al.'s set of problems

β		0.00	0.01	0.02
D1	CSA	0	–	–
	EWA	0	–	–
D2	CSA	520	379	29
	EWA	520	59	29
D3	CSA	556	242	43
	EWA	556	421	43
D4	CSA	280	280	31
	EWA	280	280	31
D5	CSA	0	–	–
	EWA	0	–	–
D6	CSA	0	–	–
	EWA	0	–	–
D7	CSA	288	8	–
	EWA	96	8	–
D8	CSA	638	616	34
	EWA	110	110	34

0.25) and the aspect ratio (length/width) randomly generated in the range (1, 5). The quantity of replicates of each smaller rectangle is a random number between 1 and *Qmax*. After this procedure has been completed, there will be 24 sets of 5 instances each. These tests were carried out on a Pentium II 350MHz microcomputer. Results are presented in Tables 5 and 6.

The optimal solution for these problems was found by CSA on setting β to a large value such as 25%. CSA took an average of 10.855 seconds per problem to provide their exact solution. Then, we ran EW and CSA using the optimal value of β for each problem.

CSA found the optimal layouts taking an average of 0.217 seconds per problem while EW took an average of 7.46 seconds.

Table 5
Average number of rectangles generated for Vasko's set of problems

	$n =$	5		10		15		20		Average Total	Average Total
H	b_i	EW	CSA	EW	CSA	EW	CSA	EW	CSA	EW	CSA
150	1–5	384.80	532.00	1860.40	2332.60	2188.80	2754.80	3481.00	3854.20	1978.75	2368.40
	1–10	253.00	330.00	2582.00	2864.60	1535.60	2106.40	1628.00	1415.20	1565.26	1750.05
200	1–5	666.00	1138.60	917.20	2093.00	996.80	1565.60	1800.60	2694.00	1095.15	1872.80
	1–10	543.20	848.20	745.00	1708.60	1703.20	2360.80	2052.40	3395.00	1260.95	2078.15
300	1–5	409.00	458.20	675.60	1367.60	2655.00	4951.00	964.20	1739.80	1175.95	2129.15
	1–10	306.40	451.40	584.60	1120.40	6062.60	7655.80	1147.80	1987.40	2025.35	2803.75
Average Total		433.07	636.62	1227.47	1914.47	2523.67	3565.73	1845.67	2514.27	1516.50	2170.55

Table 6
Average execution time (seconds) for Vasko's set of problems

	$n =$	5		10		15		20		Average Total	Average Total
H	Quant.	EW	CSA	EW	CSA	EW	CSA	EW	CSA	EW	CSA
150	1–5	0.29	0.04	6.99	0.17	10.58	0.20	28.27	0.30	11.53	0.18
	1–10	0.14	0.02	11.49	0.24	6.05	0.14	4.49	0.09	5.83	0.13
200	1–5	1.15	0.10	1.49	0.22	2.79	0.13	6.47	0.24	2.97	0.17
	1–10	0.72	0.06	1.65	0.15	6.55	0.21	9.68	0.31	4.65	0.18
300	1–5	0.21	0.04	1.16	0.11	14.00	0.77	5.06	0.16	5.11	0.27
	1–10	0.17	0.03	0.56	0.10	53.36	1.17	4.49	0.18	14.65	0.37
Average Total		0.46	0.05	3.89	0.16	15.55	0.44	9.74	0.21	7.47	0.22

4. Conclusion

In our paper we present new properties for the two-dimensional guillotine cutting stock problem and then an effective heuristic algorithm that makes use of these properties along with a clever representation of the main list. Together, both the list representation and the problem properties in which that

representation is founded provide a sophisticated branching strategy for the algorithm. In addition, a procedure to reject duplicate patterns (which is not the core of the paper) is also used.

We compare our algorithm (CSA) with Wang's and Vasko's (EW). (Comparison with Wang's and Vasko's methods rather than those of other researchers is fairest because of the similarity between their implementations and ours, including the use of the parameter β.) For such comparison, we used medium-sized problem instances available from the literature with 30 pieces to be cut.

We also ran the algorithms on a PC (Pentium II 350MHz) for the set of 120 problems used by Vasko (1989). (This data set was kindly provided to us by Vasko with permission of Bethlehem Steel.) Using, for example, the optimal value of β for each problem, CSA found the optimal layouts taking an average of 0.217 seconds per problem while EW took an average of 7.46 seconds. Also, for all the instances, CSA took less than 1.2 seconds per problem while, for example, for the 10 problems characterized by ($L = 100$, $H = 300$, $n = 15$), EW took 33.68 seconds per problem.

Finally, we would point out that our approach is useful, practical and very easy to be implemented by the reader—easier than most good algorithms in the literature, e.g. Morabito and Arenales (1996) and Fayard et al. (1998).

Acknowledgements

The authors would like to thank the anonymous referees for valuable comments. The first author was supported by CAPES (grant #0162/95-8).

References

Beasley, J.E. 1985. Algorithms for unconstrained two-dimensional guillotine cutting, *Journal of the Operational Research Society*, 36, 297–306.

Christofides, N., Whitlock, C. 1977. An algorithm for two-dimensional cutting problems, *Operations Research*, 25, 31–44.

Christofides, N., Hadjiconstantinou, E. 1995. An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts, *European Journal of Operational Research*, 83, 21–38.

Daza, V.P., Alvarenga, A.G., de Diego, J. 1995. Exact solutions for constrained two-dimensional cutting problems, *European Journal of Operational Research*, 84, 633–644.

Desler, J.F., Hakimi, S.L. 1969. A graph theoretic approach to a class of integer programming problems, *Operations Research*, 17, 1017–1033.

Dyckhoff, H., Kruse, H.-J., Abel, D., Gal, T. 1988. Classification of real world trim loss problems. In: Fandel, G., Dyckhoff, H., Reese, J. (Eds.), *Essays on Production Theory and Planning*, Springer-Verlag, Berlin, pp. 191–208.

Fayard, D., Hifi, M., Zissimopoulos, V. 1998. An efficient approach for large scale two-dimensional guillotine cutting stock problems, *Journal of the Operational Research Society*, 49, 1270–1277.

Gilmore, P.C., Gomory, R.E. 1965. Multistage cutting problems of two and more dimensions, *Operations Research*, 13, 94–120.

Gilmore, P.C., Gomory, R.E. 1966. The theory and computation of knapsack functions, *Operations Research*, 14, 1045–1074.

Haessler, R.W., Sweeney, P.E. 1991. Cutting stock problem and solution procedures, *European Journal of Operational Research*, 54, 141–150.

Herz, J.C. 1972. Recursive computational procedures for two-dimensional stock cutting, *IBM Journal of Research and Development*, 16, 462–469.

Hifi, M., Zissimopoulos, V. 1997. Constrained two-dimensional cutting: an improvement of Christofides and Whitlock's exact algorithm, *European Journal of Operational Research*, 5, 8–18.

Hinxman, A.I. 1980. The trim loss and assortment problems: a survey, *European Journal of Operational Research*, 5, 8–18.

Morabito, R., Arenales, M.N. 1996. Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach, *European Journal of Operational Research*, 94, 548–560.

Oliveira, J.F., Ferreira, J.S. 1990. An improved version of Wang's algorithm for two-dimensional cutting problems, *European Journal of Operational Research*, 44, 256–266.

Vasko, F.J. 1989. A computational improvement to Wang's two-dimensional cutting stock algorithm, *Computers and Industrial Engineering*, 16, 109–115.

Viswanathan, K.V., Bagchi, A. 1993. Best-first search methods for constrained two-dimensional cutting stock problems, *Operations Research*, 41, 768–776.

Wang, P.Y. 1983. Two algorithms for constrained two-dimensional cutting stock problems, *Operations Research*, 31, 573–586.