J. C. Herz

Recursive Computational Procedure for Two-dimensional Stock Cutting

Abstract: A recursive algorithm is implemented to give high computational speeds in the solution of a cutting-stock problem. Optimal edge-to-edge cutting is shown to be achieved more easily by recursive programming than by conventional methods. The technique features preliminary discretization, which lowers the memory requirements in the computational procedure. A comparison is made between this recursive algorithm and two iterative algorithms previously given by Gilmore-Gomory. The limitations of the algorithms are discussed and some numerical results given.

Introduction: basic algorithm

The process of cutting a given rectangle into smaller rectangles of various sizes is often a recursive one. That is, one draws a line orthogonally from one edge to the opposite one and then proceeds the same way with the two resulting rectangles as indicated in Fig. 1. Exceptions are certain applications such as the dissection of a rectangle into squares of different sizes ("perfect rectangles") [1] as shown in Fig. 2.

We consider the following problem, which has been already discussed at length by P. C. Gilmore and R. E. Gomory in Ref. 2: Given a "large" rectangle R and a finite set S of "small" rectangles, find a dissection of the large rectangle (by the process mentioned above) which minimizes the "loss," i.e., the total area of those resulting rectangles which do not belong to the set S. Obviously, this is equivalent to maximizing the total area of "usable" rectangles. More generally, as discussed in Ref. 2, one has to maximize the total "value" of usable rectangles, each member of the set S having been given such a value.

Any optimal solution has the following recursive property: Either the large rectangle is in S, or the first dissection line yields two rectangles, each of which is optimally dissected. Now a simple algorithm consists in trying every possible first dissection line (requiring only a finite number of trials as indicated in Theorem 1) and retaining the one which gives the maximal total value for the two partial dissections.

It should be noted that this algorithm is valid only under the assumption that there is no bound for the number of occurrences of a small rectangle in the solution. Otherwise, some optimal solutions might correspond to non-optimal partial solutions.

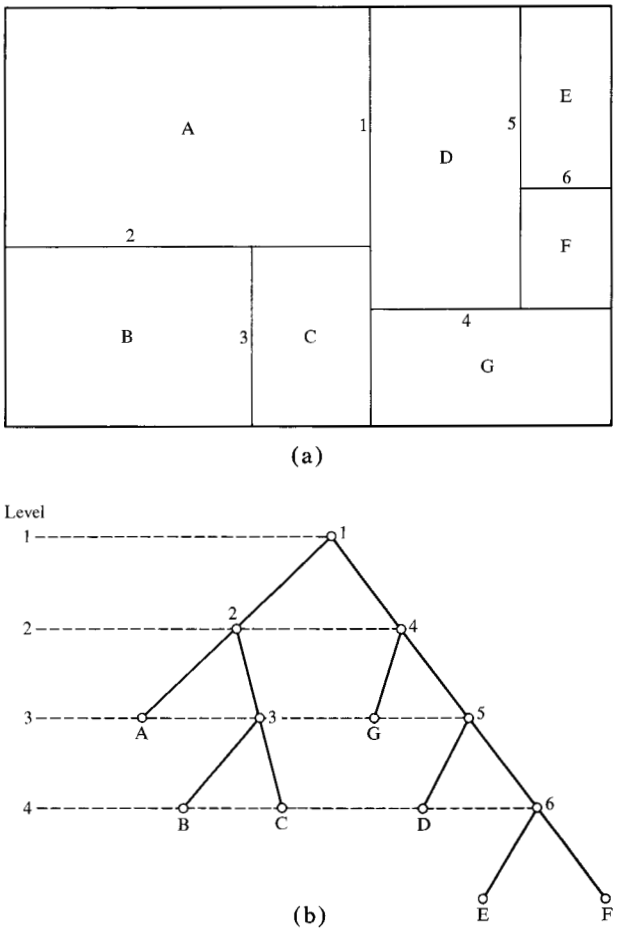


Figure 1 *Recursive cutting.* The graph of Figure 1(b) shows the order in which the large rectangle must be cut. Each small rectangle is obtained by the sequence of cuts about it.

The assumption of no bounds for the number of these occurrences makes it possible also to get rid of "anisotropy" considerations. Suppose some rectangles may be placed only "horizontally," while the others may have either orientation. Then each of the latter type will be represented by two members of S, by exchanging its dimensions, and the whole will be handled "horizontally." Rectangles having either orientation could not be represented in this way if the numbers were bounded.

In the following sections, we present some refinements of this algorithm and discuss the convenience of using a programming language such as PL/1 that features recursiveness.

Discretization

Let a and b be the dimensions ("horizontal" and "vertical") of the given large rectangle R, p_i and q_i those of the ith member of S, and v_i its value $(1 \leq i \leq n)$. Let p and q be the column vectors of coordinates p_i, q_i, respectively.

Denote by z a line vector with n nonnegative integer coordinates. A finite number of such vectors satisfies $zp \leq a$ and the same is true for $zq \leq b$. Let P and Q be the corresponding finite sets of values of zp and zq.

Theorem 1

For any dissection of R, there exists a dissection of equal or higher value with all abscissae in P and all ordinates in Q.

Here we designate the distance of a vertical dissection line to the left edge of R as the abscissa and the distance of a horizontal dissection line to the lower edge as the ordinate.

Proof

We use induction on a and b. If a is less than every p_i, then any dissection gives no usable rectangle and hence is of same value as the "empty" dissection (i.e., the rectangle with no dissection lines). The same is true if b is less than every q_i.

Suppose the theorem is true for every rectangle R' of dimensions a', b' such that $a' \leq a$ and $b' \leq b$ (but $a + a' < b + b'$). Consider a dissection D of R.

We may assume that the first dissection line L is vertical, the other case being quite similar. By the induction hypothesis, there exists a dissection D_1 of the left rectangle R_1 of equal or higher value than the left part of D and with all abscissae in P and all ordinates in Q; there exists a dissection D_2 of the right rectangle R_2 of equal or higher value than the right part of D and with all abscissae (counted from L) in P and all ordinates in Q.

If the abscissa of L is in P, the union of D_1 and D_2 is a dissection satisfying the conclusion, as P is additive. More precisely, $x \in P$, $x' \in P$ and $x + x' \leq a$ implies $x + x' \in P$.

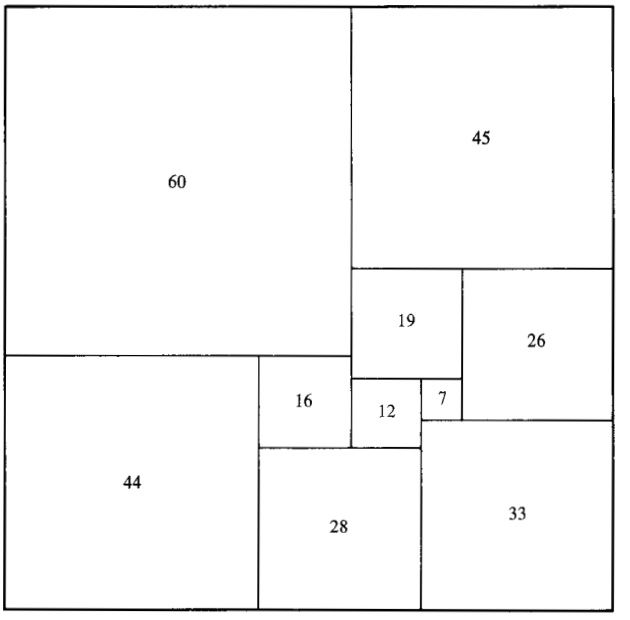
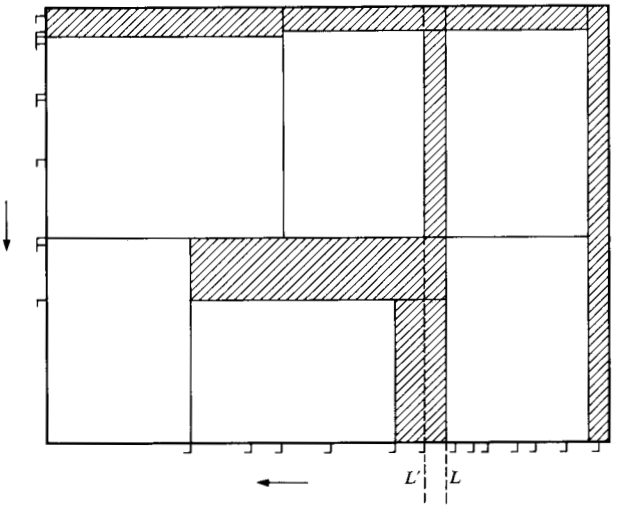


Figure 2 *Nonrecursive cutting.* This figure is taken from Ref. 1. The numbers inside the squares are the edge lengths.

Figure 3 *Getting a canonical dissection.* Nonusable rectangles are shaded. Each of the two main parts is canonically dissected. P and Q are represented along the edges.



If the abscissa of L is not in P, then every rectangle of D_1 with its right edge on L is nonusable; the rightmost abscissa of the corresponding left edges is in P. Now we obtain the sought-for dissection of R by drawing the vertical line L' at that abscissa, suppressing the lines of D_1 on the right of L' and translating R_2 with D_2 from L to L', as shown in Fig. 3. Q.E.D.

Although we did not assume the data to be integral, nor even rational, the inductive process is clearly finite. The

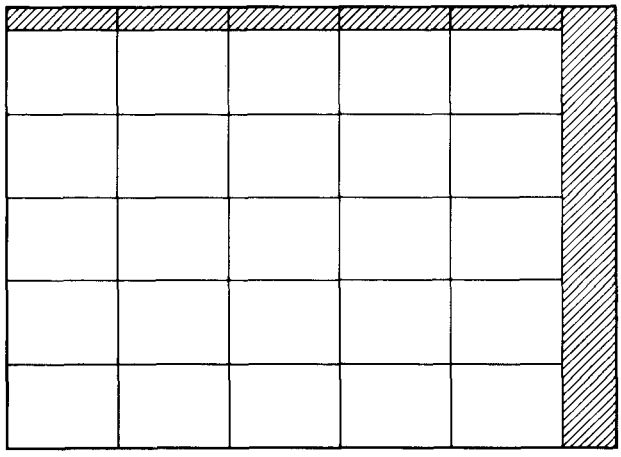


Figure 4 A homogeneous canonical dissection

Figure 5 Symmetrization: first case.

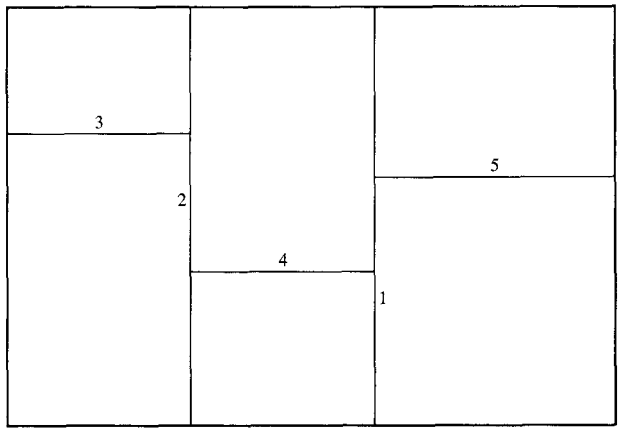
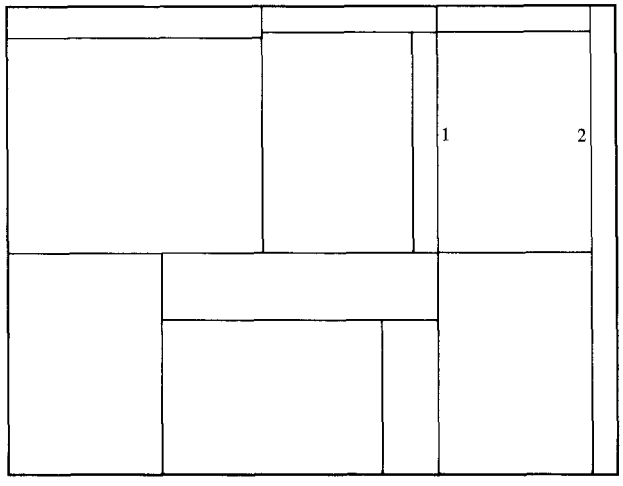


Figure 6 Symmetrization: second case.



procedure consists of forcing the usable rectangles to the left and the bottom.

Theorem 1 reduces the trials involved in the above algorithm to the finite process of scanning P and Q. A dissection with all coordinates in P and Q will be called a *canonical* dissection.

Homogeneous dissections

We call a dissection that involves only one type of small rectangles a *homogeneous* dissection. In a canonical homogeneous dissection, the abscissae are $p_i, 2p_i, 3p_i, \cdots$ and the ordinates are $q_i, 2q_i, 3q_i, \cdots$ for some i. See Fig. 4.

Symmetrization

• *Theorem 2*

For every canonical nonhomogeneous dissection of R there exists a canonical dissection of equal value, the coordinate of the first line of which is at most equal to half of the corresponding dimension of R.

Proof

Suppose the first vertical line of abscissa c is greater than $a/2$. If the second line is vertical at an abscissa less than or equal to $a/2$, we can consider it as the first line (Fig. 5). If the second line is vertical also at an abscissa greater than $a/2$, the rectangle between both lines can be exchanged with the left rectangle (Fig. 6).

If the second line is horizontal, consider the rightmost abscissa c' of usable rectangles; if $c' > c$, one has only to draw the vertical line at c' (as much as $c' < a$) and to exchange the two left vertical parts (Fig. 7).

If $c' = c$, one can extend the second line to the right edge of R and consider it as the first dissection line, which gives rise to the same situation as above, in which coordinates are exchanged.

Finally, only one case cannot be handled, i.e., when there is only one small rectangle in the dissection as illustrated in Fig. 8. But this is not a nonhomogeneous dissection. Q.E.D.

Theorem 2 reduces the scanning of P and Q by half. It applies, of course, to every subrectangle.

Exclusion

We have implicitly assumed that for every i, $p_i \leq a$ and $q_i \leq b$. Otherwise some small rectangles could never be cut. This assumption, however does not necessarily hold for the partial rectangles involved in the recursive algorithm. For such a rectangle, which is of dimensions $\alpha \times \beta$, we have to scan

$$P_{\alpha\beta} = \{zp \mid (q_i > \beta \Rightarrow z_i = 0) \text{ and } (zp \leq \alpha/2)\}$$

and

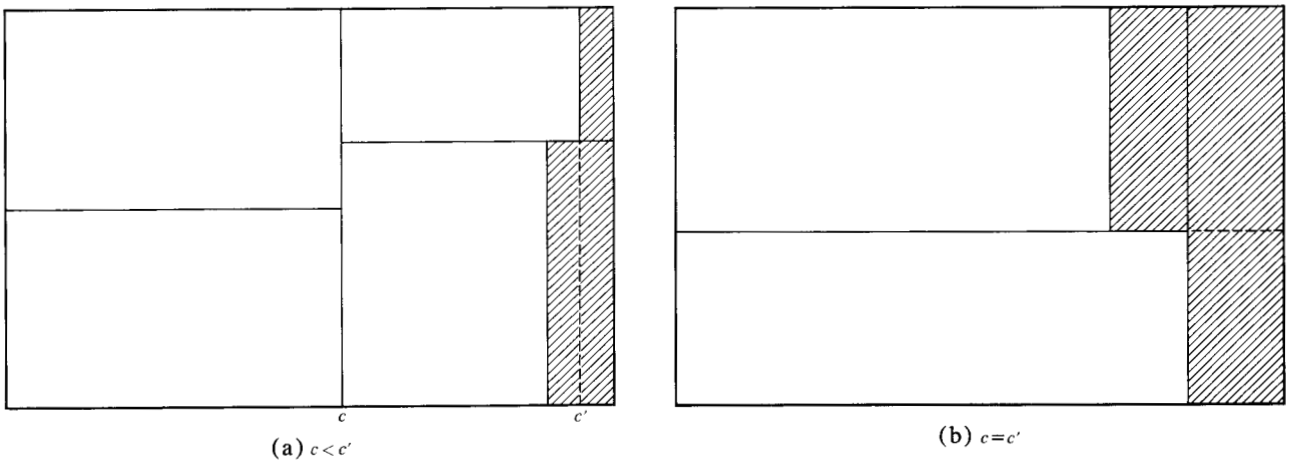


Figure 7 (a) Symmetrization: third case. (b) Symmetrization: fourth case.

$Q_{\alpha\beta} = \{zq \,|\, (p_i > \alpha \Rightarrow z_i = 0) \text{ and } (zq \leq \beta/2)\}$.

A more practical computation of $P_{\alpha\beta}$ is the following: Associate with every z the quantity

$f(z) = \sup \{q_i | z_i > 0\}$

and to every x in P the quantity

$F(x) = \inf \{f(z) | zp = x\}$

Then

$P_{\alpha\beta} = \{x | x \leq \alpha/2 , F(x) \leq \beta\}$.

Similarly, $Q_{\alpha\beta}$ is

$\{y | y \in Q , y \leq \beta/2 , G(y) \leq \alpha\}$,

where

$G(y) = \inf \{g(z) \,|zq = y\}$

and

$g(z) = \sup \{p_i \,| z_i > 0\}$.

Because F and G do not depend on α and β, they can be computed at the outset.

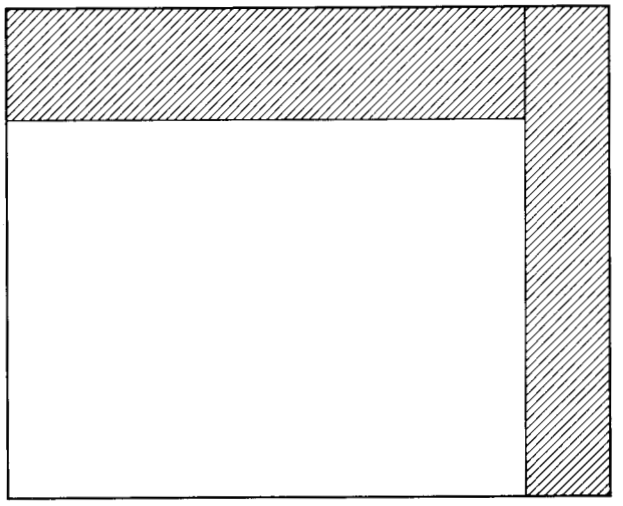
In order to speed up the main computation, P and Q are generated by increasing mappings ϕ and ψ from $[1 , \lambda]$ and $[1 , \mu]$ into **R**, so that F and G are expressed through mappings $F_1 = F \circ \phi$, $G_1 = G \circ \psi$, as now developed. (λ and μ are the cardinals of P and Q.)

• *Computation of F_1 and G_1*

ϕ and ψ are constructed by the following recursive process, which we give only for ϕ. Initially, λ is zero. The current step consists in checking, for every i and every j such that $\phi(j) + p_i \leq a/2$, to see whether $\phi(j) + p_i$ belongs to $\phi([1 , \lambda])$; if not, λ is increased by 1 and the values of ϕ are shifted in order that ϕ keep increasing. The process terminates when λ is not changed in the current step.

F_1 (and similarly G_1) is computed at the same time; if $\phi(j) + p_i$ is $\phi(k)$, then $F_1(k)$ is replaced by the minimum of $F_1(k)$ and the maximum of $F_1(j)$ and q_i. If $\phi(j) + p_i$ is not in $\phi([1 , \lambda])$, F_1 takes at the inserted index k the value of the maximum of $F_1(j)$ and q_i (and indices following k are shifted).

Figure 8 Symmetrization: fifth case.



Main recursion

We are now ready to describe the search for an optimal dissection of R. The possible dissections of R are

1. The homogeneous dissections.

2. Any dissection, the first line of which is vertical at some abscissa in P_{ab}, the remaining dissection lines being those in any dissection of each of the two partial rectangles. The value is the sum of the values of these two dissections.
3. Any similar dissection in which the first line is horizontal.

The optimal dissection is chosen by comparing the values of all these dissections.

Clearly, recursion occurs in 2) and 3).

Memorization

In the course of the computation, the same partial rectangle happens to be considered many times. It is obviously useful to store its value rather than recompute it each time. Moreover, if we define α_0 as the maximum element of P that is less than or equal to α, and β_0 as the maximum element of Q that is less than or equal to β, then $\alpha \times \beta$ and $\alpha_0 \times \beta_0$ have the same dissection value, so we have to memorize only the values of elements of $P \times Q$. The first step in the algorithm will be to compute α_0 from α, and β_0 from β (or more exactly, the indices of α_0 in P and of β_0 in Q).

Use of bounds

When an upper bound $u(\alpha, \beta)$ of the dissection value of every rectangle $\alpha \times \beta$ is known, one has a means of speeding up the computation. This happens in particular when the value is simply equal to the area; $u(\alpha, \beta)$ is then $\alpha\beta$.

- *Full dissection*

When searching for an optimal dissection of $\alpha \times \beta$, we can stop as soon a value has been found that is equal to $u(\alpha, \beta)$. If the value equals the area, this corresponds to a solution without loss.

- *Unattainable value*

Suppose that, at some moment in the computation, the best dissection value already found for $\alpha \times \beta$ is v, and that, after a cut is made at abscissa γ, the optimal dissection value of the left rectangle $\gamma \times \beta$ is w and has been computed. If

$$v \geq w + u(\alpha - \gamma, \beta), \tag{1}$$

it is useless to investigate the dissections of the right rectangle $(\alpha - \gamma) \times \beta$, and one can proceed to the next step of the algorithm.

Hence we shall add a third parameter to α and β in the recursive routine, i.e., the value v_0 which must be reached ($v - w$ in the above example); if v_0 is not less than $u(\alpha, \beta)$, then the whole routine will be skipped.

- *Indirect bound*

Suppose again that v is the best dissection value already found for $\alpha \times \beta$, but also that v is less than v_0. Then condition (1) must be replaced by

$$v_0 \geq w + u(\alpha - \gamma, \beta), \tag{2}$$

so that, for the dissection of $(\alpha - \gamma) \times \beta$, the goal will be $v_0 - w$.

As a consequence, if $v_0 \geq u(\gamma, \beta) + u(\alpha - \gamma, \beta)$, it is useless even to investigate the dissections of $\gamma \times \beta$.

- *Heuristic search*

The method developed so far yields an optimal solution. In some cases, the computation time may be excessive. Use of arbitrary bounds makes it possible to obtain suboptimal solutions more rapidly. For instance, one can reduce the actual bound u by a fixed factor, i.e., one may accept a fixed percentage of loss. The remainder of the process then remains unchanged.

PL/1 program

The recursion features of PL/1 permit the use of a very concise and elegant program, a flowchart of which is given in Fig. 9. Recursion occurs also in the final *tracing*, which is achieved by a simple recursive routine t with parameters a, b, giving in all cases a, b and the dissection value; if the dissection is homogeneous, it gives also the index i of the used rectangle. If the first line is vertical at abscissa c, it prints "VERTICAL" and c and calls $t(c,b)$ and $t(a - c,b)$. If the first line is horizontal, at ordinate c, it prints "HORIZONTAL" and c and calls $t(a,c)$ and $t(a,b - c)$.

Roundoff errors

The use of floating point computation has been justified by the above discretization theory. In practical situations, fixed-point computation would be quicker. Moreover, because of roundoff errors, the equality of integers may be overlooked. Such errors can be prevented by introducing small quantities in comparison operations.

Comparison with Gilmore-Gomory's algorithms

Gilmore and Gomory give two algorithms [2] for the two-dimensional edge-to-edge cutting problem—a basic algorithm and a refinement of it.

There are a number of differences between these algorithms and the present procedure. A minor one is that the former operate on integers rather than on real numbers. Also, they do not use bounds; they have a larger memory requirement; they do not introduce homogeneous dissections. The main difference is that they are iterative rather than recursive. In addition, the improved algorithm does not lead to a correct solution, as will be seen.

Before discussing these two algorithms in more detail, let us consider another simple iterative algorithm. Sup-

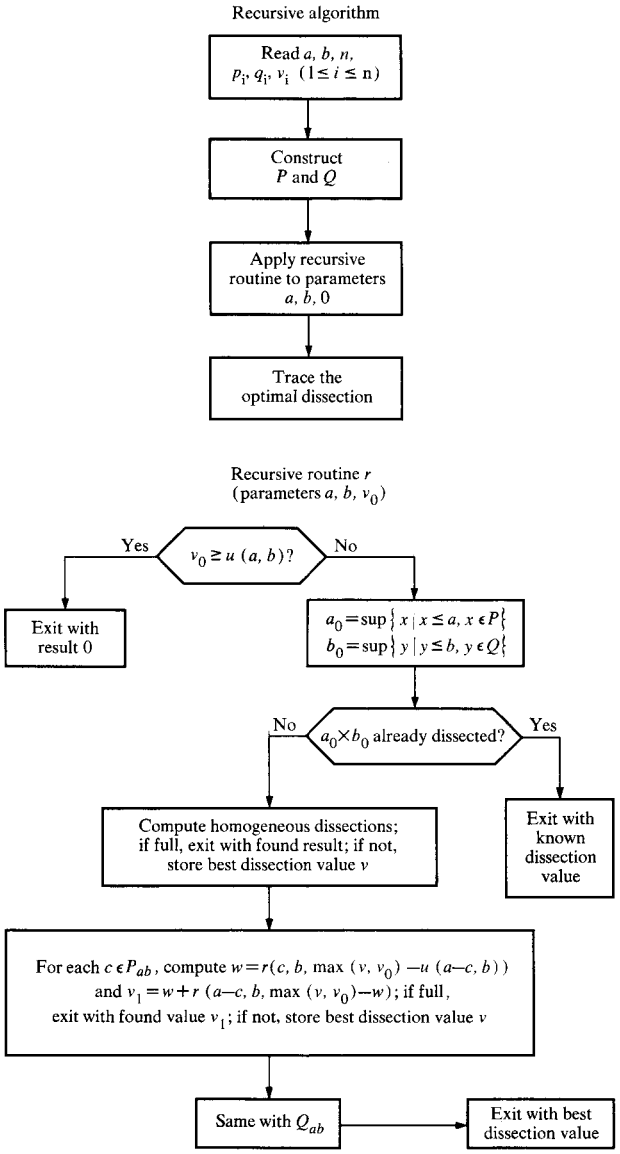


Figure 9 Recursive algorithm.

pose a and b, as well as the p_i's and q_i's are integers. We can compute the best dissection value $H(x, y)$ of every subrectangle $x \times y$ $(1 \leq x \leq a, 1 \leq y \leq b)$ by increasing values of y, and for a given value of y by increasing values of x. For a fixed pair (x, y) we compute all sums $H(x_1, y) + H(x - x_1, y)$ with $1 \leq x_1 \leq x/2$, then all sums $H(x, y_1) + H(x, y - y_1)$ with $1 \leq y_1 \leq y/2$, and retain the maximum sum. It is easily seen that the number of inner loops is about $ab(a + b)/8$.

Gilmore-Gomory's basic algorithm differs from this scheme only in the order of the computation, the outer loops being on values of $y - y_1$ and $x - x_1$. The inner loops are entered the same number of times as above. With the example displayed in Appendix A and Fig. 10,

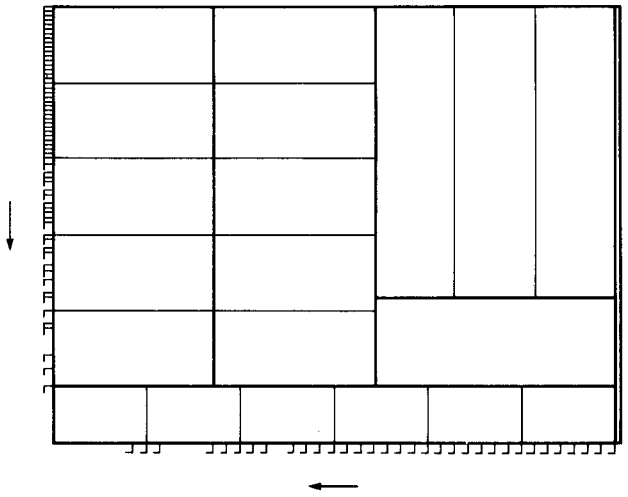


Figure 10 Numerical example. See Table 1. Discretization sets P and Q are represented along the edges.

this number is about 350,000. (It can be reduced to about 70,000 by taking into account the fact that all lengths p_i are multiples of 3.)

Tracing is achieved by Gilmore and Gomory in a very similar way to that mentioned earlier: $l(x, y)$ is defined as the smallest positive x_1 such that $H(x, y) = H(x_1, y) + H(x - x_1, y)$ if it exists, otherwise as x; $w(x, y)$ is defined as the smallest positive y_1 such that $H(x, y) = H(x, y_1) + H(x, y - y_1)$ if it exists, otherwise defined as y.

The improvement to Gilmore-Gomory's basic algorithm consists in skipping for some values of $x_2 = x - x_1$ and $y_2 = y - y_1$ some or all values of x_1 and/or y_1 in the basic process, according to the following rules. A new definition of functions l and w is used: $l(x, y) = 0$ if $H(x, y) = H(x - 1, y)$, $w(x, y) = 0$ if $H(x, y) = H(x, y - 1)$. Now

1. the loop on x_1 is skipped if $l(x_2, y_2) = 0$;
2. the loop on y_1 is skipped if $w(x_2, y_2) = 0$;
3. the value of x_1 is limited to $l(x_2, y_2)$ and skipped if $w(x_1, y_2) = 0$;
4. the value of y_1 is limited to $w(x_2, y_2)$ and skipped if $l(x_2, y_1) = 0$.

This "improved" algorithm does not work with the following example: $n = 2$, $a = b = 3$, $p_1 = 1$, $q_1 = 2$, $p_2 = 2$, $q_2 = 3$, $v_2 = 6$ (see Fig. 11). We have $H(1, 3) = H(1, 2)$, hence $w(1, 3) = 0$. When $H(3, 3)$ is being computed, the correct optimal dissection $H(3, 3) = H(1, 3) + H(2, 3)$ is overlooked by virtue of (3). The same example can be used for invalidating Theorem 7 (p. 1072 of [2]).

Comparison with an iterative algorithm.

We are interested in comparing two algorithms that are as similar as possible, one being recursive and the other

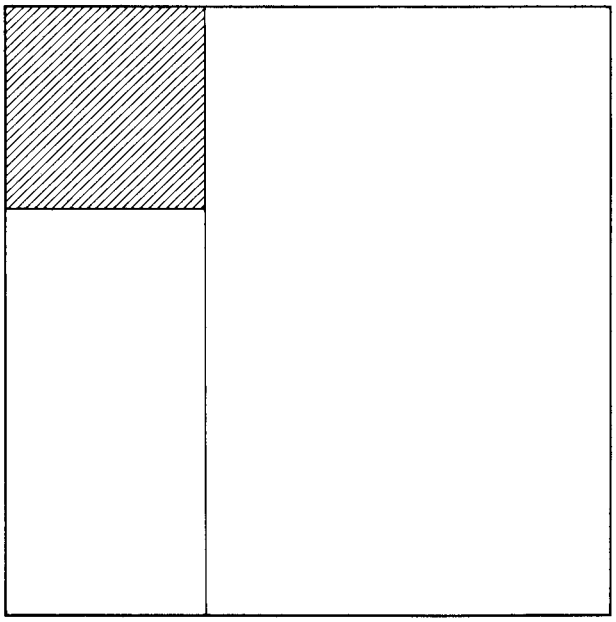


Figure 11 A counterexample to the Gilmore-Gomory theorem.

iterative. Actually, very little is required for transforming our recursive procedure into an iterative one (see Fig. 12). We use the simple algorithm above combined with discretization, use of homogeneous dissections, symmetrization and exclusion. We use also the "full dissection" feature, but neither the "unattainable value", nor the "indirect bound," which are of little interest, since the partial dissection values have already been computed. That was not the case in the recursive procedure.

Computation times show a difference of about 20% in favor of the recursive procedure, as shown in Appendices A and B. On the other hand, the iterative procedure gives the optimal dissection of every subrectangle; Table 1, Examples A and C, shows that only 462 subrectangles out of 1865 have been actually dissected by the recursive procedure.

Extensions

• *Higher dimension*

Our algorithm applies obviously to any dimensionality, provided that the cut-in-two condition holds. If the space is r-dimensional, the r functions $F_1, G_1, \cdots$ are $(r-1)$-vector functions, and the r sets $P_{\alpha\beta}, Q_{\alpha\beta}, \cdots$ are defined by conditions similar to the two-dimensional ones. The program presents r recursive loops.

• *Handling flaws*

The important practical case where the large rectangle has nonusable spots cannot be easily discretized; nevertheless, it can be, in principle, handled by a recursive procedure, the parameters of which would be, in addition to dimensions α, β, coordinates x, y of the lower left vertex.

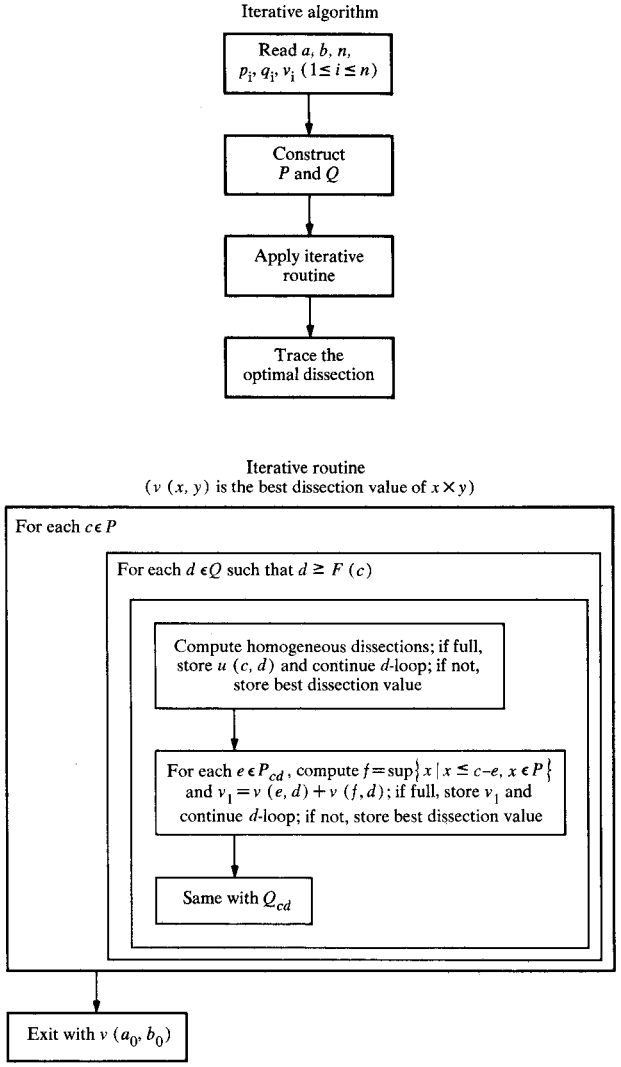


Figure 12 Iterative algorithm.

Conclusion

We have shown that recursive procedures for solving the stock cutting problem are not only simpler than conventional techniques but are also more efficient. Programming language theorists will be interested by the new example shown, which is of a more practical nature than classical examples of recursive operations like factorial function and greatest common divisor.

Acknowledgment

The author is indebted to Mlle. Hélène Valabrègue for recursively improving the style of his paper.

Table 1 Three examples of processing: A, with bounds; B, without bounds; and C, with iterative procedure. *Parameters*: $n = 5$, $a = 127$, $b = 98$.

	Discretization set		
I	p_i	q_i	v_i
1	18	65	1170
2	24	27	648
3	21	13	273
4	36	17	612
5	54	20	1080

Dimensions	*Level*	*Value*	*Line*	*At*	*Type*
127 × 98	1	12348	Horizontal	13	
127 × 13	2	1638			3
127 × 85	2	10710	Vertical	36	
36 × 85	3	3060			4
91 × 85	3	7650	Vertical	36	
36 × 85	4	3060			4
55 × 85	4	4590	Horizontal	20	
55 × 20	5	1080			5
55 × 65	5	3510			1

Example A: LAM = 33 MU = 60 NB = 4861 NB0 = 462
NB1 = 2643 NB2 = 38 NB3 = 4 NB4 = 4.

Example B: LAM = 33 MU = 60 NB = 9859 NB0 = 698

Example C: LAM = 33 MU = 60 NB = 1865 NB0 = 8420
NB1 = 12221 NB2 = 86 NB3 = 21 NB4 = 23.

Appendix A. An example

The same example was processed with and without use of bounds (see Table 1, Examples A and B, and Fig. 10). The value equals the area. The NB_i have the following meanings:

NB: number of calls of routine DIS.
NB0: number of calls with value not found in memory.
NB1: number of calls with attainable value.
NB2: number of full homogeneous dissections.
NB3: number of dissections with first line vertical.
NB4: number of dissections with first line horizontal.

The "level" is the depth of recursion, as in Fig. 1(b).

Appendix B. Results of the iterative program

The example of Appendix A was processed with the iterative routine (see Table 1, Example C, and Fig. 10). The value equals the area. The numbers NB_i have the following meanings:

NB: number of entries in outer loops.
NB0: number of entries in loop on x_1.
NB1: number of entries in loop on y_1.
NB2, NB3, NB4: as in Appendix A.

References

1. W. T. Tutte, "The Quest of the Perfect Square," *The American Mathematical Monthly*, **72,** No. 2, part II, 29–35 (1965).
2. P. C. Gilmore and R. E. Gomory, "The Theory and Computation of Knapsack Functions," *Operations Research*, **14,** 1045–1074 (1966).
3. D. W. Barron, *Recursive Techniques in Programming*, Macdonald Computer Monographs No. 3, 1968.

Received November 3, 1971

The author is located at the IBM Paris Scientific Center, 21 Avenue Gambetta, 92 Courbevoie, France.