An Improved Sheet Optimization Algorithm

Manfred H. Breede and Peter Pille*

Keywords: Algorithm, Paper Cutting, Automation, Programming, Software

Abstract

This paper provides an improved algorithm for sheet cutting optimization for identical rectangular areas to be cut out of sheets on guillotine type paper cutters. The algorithm provides optimization where grain direction is not a concern. The algorithm was tested and compared to several commercial software optimization programs, some of which are integral to computerized guillotines by a major equipment manufacturer, and all were found to be inferior (the criterion being the number of pieces that fit on a given area) to the algorithm under consideration.

1. Introduction

Printing as a mass communication medium requires great quantities of substrate to disseminate information. By far the most common substrate used for printed products is paper, which is purchased in roll or sheet formats. Maximizing the area of stock sheets when they have to be cut down to a smaller size is an economic concern, because any amount of waste, created in the process of cutting increases raw material costs proportionally.

Depending on the stage to which a printing project has advanced, the responsibility for maximizing stock paper areas could be that of a purchasing agent, estimator or guillotine operator. Prior to the widespread use of computers, people charged with this task have used the so-called diagram method. Many textbooks describe this method (Cogoli, 1973) still widely practiced. It entails the drawing of a diagram by trial and error and it is as such an effective means of optimization as long as the pieces, that have to be cut from a stock sheet, are few in numbers. The problem given in Figure 1.1 could conceivably be solved quite efficiently by this method.

*Ryerson University, School of Graphic Communications Management and School of Information Technology Management.

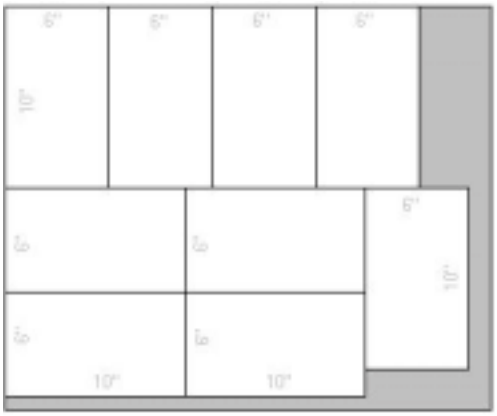

Figure 1.1 Diagram method for a small number of pieces.

If however, a great many pieces have to be cut from stock paper, (Figure 1.2) the diagram method is too cumbersome, because of the vast number of trials that could be required to find the best optimization.

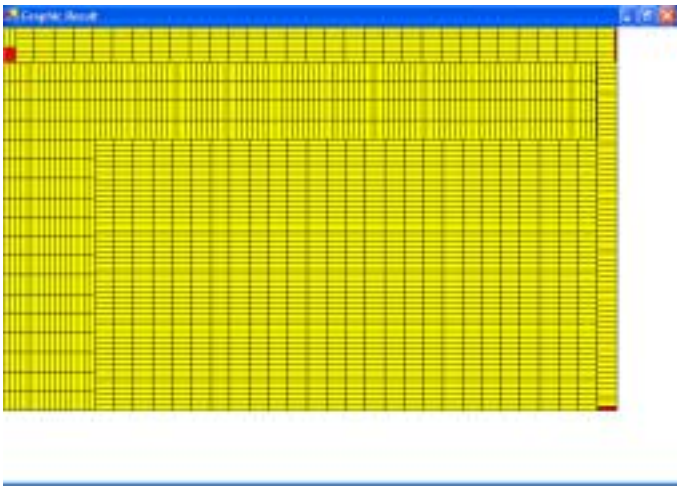

Figure 1.2 Computer generated optimization for a large number of pieces.

Hence the need for algorithms and computers that can perform these iterative operations nearly instantaneously (Figure 1.2).

2. The Significance of Paper Grain Direction

Unlike glass which is an isotropic material, many other materials including wood, leather, sheet metal, and machine made paper produce different effects in their two linear dimensions. In the case of paper we speak of the grain and cross

grain directions. The grain direction of paper is formed on the so called wet end of a papermaking machine where the slurry of pulp is drained of its water content on a moving endless wire called the Fourdrinier (Wilson, 1998). The grain direction is formed because the great majority of cellulose fibers align themselves with the process direction, not unlike logs aligning themselves lengthwise as they float downstream on a river. Consequently, the properties of paper are different in both directions and can have profound effects on the working properties of paper products. For example adhesive bound books should have the grain direction parallel to the spine of the book in order for the book to lay flat when it is opened. Many more examples of the benefits of particular grain directions depending on the end-use of a paper product could be given, but this is not the place.

Pads, post cards, business cards, posters, labels, stamps and many other printed products do not however require a preferred grain direction and can therefore benefit from the greater economy that optimization schemes provide.

This distinction is important because optimizations for preferred grain directions do not require complex algorithms as the results are simply found by dividing each required dimension into the stock dimensions, disregarding the remainders, and then multiplying the integer quotients of both. Paper stock cutting problems without prevailing grain directions are sometimes in the graphic arts industry called "Dutch Cuts", but for its greater clarity, they will henceforth be called "mixed grain" in this paper.

The software output above shows therefore three results; the two prevailing grain direction results, and the mixed grain result that use the optimization algorithm described in this paper.

3. The Cost of Waste in Paper Cutting

Minimizing the cost of paper is usually the most important factor when considering the economic viability of printing projects. The proportional cost of paper in relation to other recurring expenses, such as printing plates, printing ink, chemicals, and labor has been estimated to be somewhere in the order of 23% of printers' cost of sales (Valentino, 2003).

It is therefore of paramount importance that paper wastage is minimized in order to obtain a competitive edge in the communication market place.

Two scenarios that demonstrate the extent of potential savings follow:

Figure 3.1 shows the output of the software algorithm discussed in this paper. It gives three possible solutions to the problem as to how many labels measuring 7 x 4 inches can be cut from a stock sheet measuring 45 x 35 inches.

If the total number of required labels is 500,000, then the required stock sheets listed from the most to least waste scenario is 10,417, 9,090, and 8,929 stock sheets respectively.

A typical paper price list (Breede, 1999) shows the price for a 35 x 45 –215M offset paper to be $232.20 per 1,000 sheets or 2.322 cents per sheet.

Multiplying the individual number of sheet requirements by the cost of one sheet gives from the most expensive to the least expensive $24,188.27, $21,106.98, and $20,733.14.

This constitutes a paper price differential of $3,455.13 between the most and the least efficient scenario for this press run alone. Compounding the potential cost savings is the fact that a typical printing company could easily process three jobs of this run length per 8-hour shift.

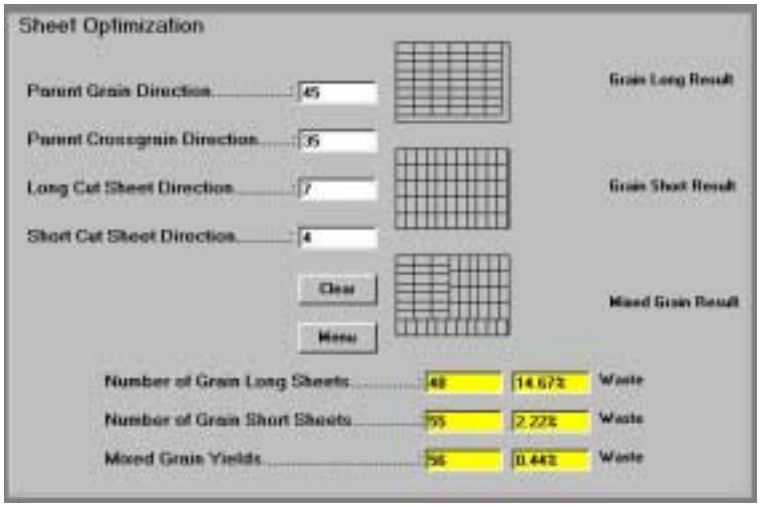

Figure 3.1 Three possible optimization possibilities using stock sheet size 35x45.

Possible cost savings are also achievable by selecting a favorable standard stock size, a variety of which paper merchants offer their customers. Given that the size of required labels is constant, optimization efficiency generally improves with larger standard stock sizes because as the ratio of standard stock size to label size increases optimization possibilities also increase (Agrawal, 1993). All other factors, such as the weight and type of paper, being equal, the cost of paper per unit area is identical for all standard stock sizes. If for example a bindery wanted to produce pads measuring 4 x 7 inches and had to select from two standard stock sizes: 35 x 45 inches (Figure 3.1), or 19 x 25 inches (Figure 3.2),

then the better solution would be with the larger standard stock size shown in Figure 3.1, rather than with the smaller standard stock size shown in Figure 3.2. The dollar value saved is the same as the difference between the two waste percentages, which is 5.68 – 0.44 = 5.24%.

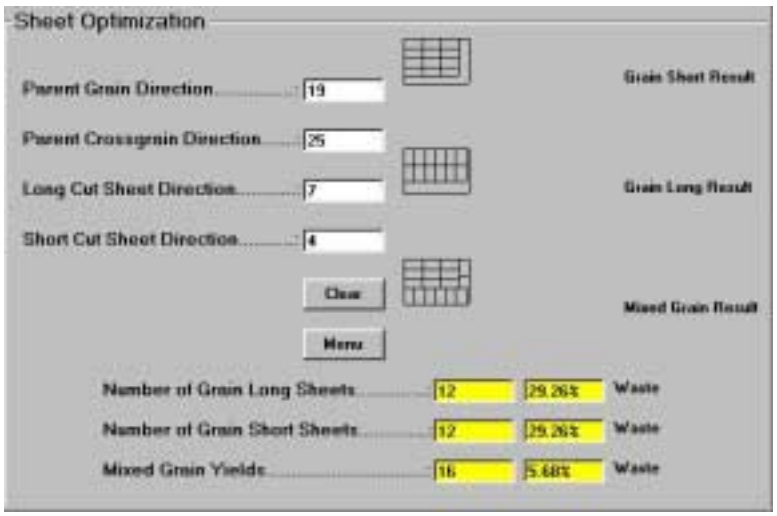

Figure 3.2 Three possible optimization possibilities using stock sheet size 19x25.

If for example 10,000 standard stock size sheets of 45 x 35 were needed to produce the required number of pads, then in the19 x 25 standard size, 5.24% more paper would be required to complete the order. Still using the same price of $232.20 per 1,000 sheets, the total price for the better optimization result would amount to 232.20 x 10 = $2,322.00. Using the inferior optimization result, would result in costs of $2,322.00 + 5.24% = $2,443.66 or an increase of $121.91.

Better optimization results are particularly beneficial when large quantities of paper are consumed, in which case even small optimization improvements can produce significant financial returns.

4. The Guillotine Cutter Principle

The standard type of cutting machine used in the graphic arts industry belongs to a category of cutting machines called guillotine cutters, the working principle of which permits only cuts that form a straight line from one side of the cut to the opposite side.

The cutting pattern shown in Figure 4.1 represents thirty-three 3 x 7 inch pieces cut from a stock sheet measuring 35 x 23 inches and is as such the absolute

maximum optimization possible. However, on a guillotine cutter, this cutting diagram would cause the pieces to be bisected, which is clearly not permissible.

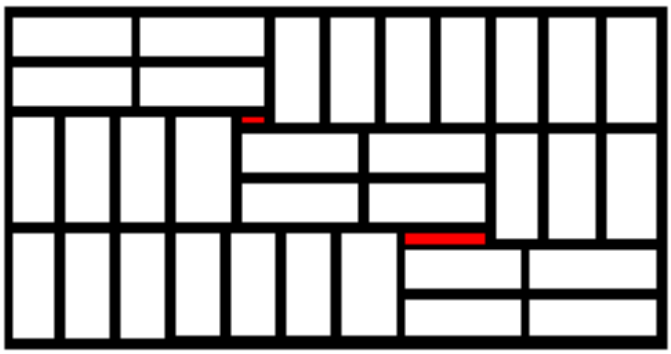

Figure 4.1 Non-guillotineable cutting pattern yielding 33 pieces.

Although, the same stock cutting problem, when run through the software driven by the algorithm at issue in this paper and shown in Figure 4.2 yields one less piece (32) than the solution in Figure 4.1, it is nevertheless the only allowable solution in view of guillotine cutters' technical constrains.

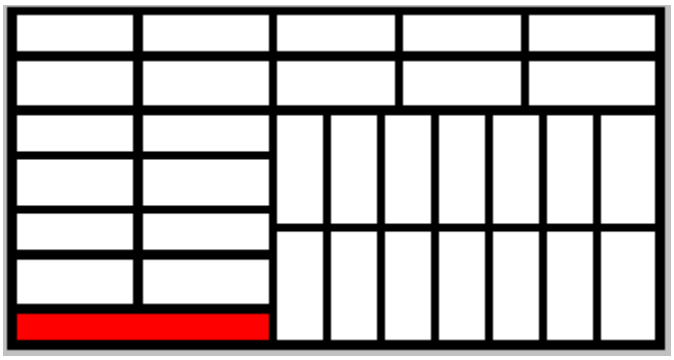

Figure 4.2 Guillotineable cutting pattern yielding 32 pieces.

4.1. Modern Guillotine Cutters

Most printers and virtually all bindery and finishing companies will require a guillotine cutter in their day to day operation, but in companies specializing in product categories such as labels, postcards, securities, pads etc., guillotine cutters occupy a central role. These types of products require extensive and potentially labor intensive cutting sequences, which greater use of automation could rationalize. For this reason highly automated cutting lines are equipped with robotics that reduce manual handling to a minimum, and relevantly in the context of this paper, these guillotine cutters are programmable (Kippan, 2001).

Cutting programs are generated upon the operator's input with regard to production specifications such as stock sizes, required sizes, trims, and cutouts. Optimization is then calculated according to the net area available, cutting sequences are determined and are subsequently carried out automatically.

The algorithm at issue in this paper deals exclusively with the fundamental problem of finding the best optimization solution of a net available area subsequent to input of all other production parameters.

5. The Complexity of Stock Cutting Problems in the Graphic Arts Industry

Dyckhoff (1992) classifies optimization problems according to several conditions, one of which he calls pattern restrictions. Accordingly, there are two types of patterns: orthogonal and non-orthogonal. The cutting problems investigated in this paper belong to the orthogonal category, which means that pieces must be parallel to the edges of the stock paper and may be mutually perpendicular to each other. Furthermore, Dyckhoff distinguishes between nested and guillotine orthogonal patterns, the former being characterized by the example given in Figure 4.1 and the latter shown in Figure 4.2, is applicable to the algorithm discussed here.

In contrast to the sheet metal, leather processing, or textile industries where stock cutting problems of composite sizes and irregular shapes have to be considered (Sharma, R. et al., 1997), typical stock cutting problems in the graphic arts industry usually require only identically sized rectangular stock sheets and pieces. This reduces the complexity of the problems significantly and makes the creation of optimal solutions more likely.

6. Sheet Cutting Optimization

6.1. Problem Formulation and Notations

The objective is to fit as many as possible of identical small rectangular sheets inside a larger rectangular sheet so that the larger sheet can be cut into the small individual sheets with a series of guillotine cuts. All dimensions and areas in the following are positive integers, without loss of generality. Figure 6.1 shows a possible arrangement and where the initial guillotine cut is made.

Basic definitions:

Small sheet dimensions:

horizontal (initially): s_x

vertical (initially): s_y

In the following descriptions and, the small sheet is always initially orientated so that s_x is defined to be measured in the horizontal direction, but when the small sheet is rotated 90 degrees, s_x will measure the vertical dimension.

Large sheet dimensions:

horizontal: l_x

vertical: l_y

For the large sheets, l_x always measures the horizontal direction, including after rotation of the sheet.

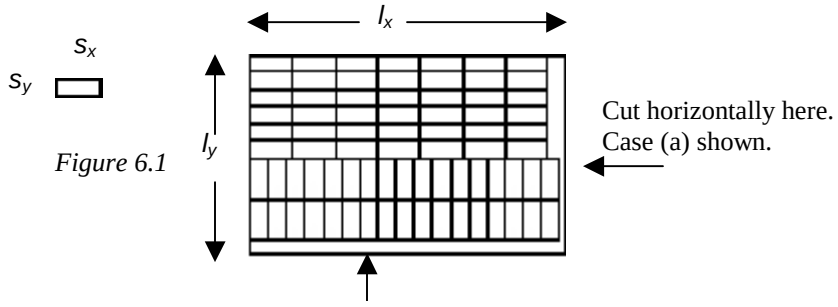

Figure 6.1

Unused horizontal area
Case (a) placed at bottom of large sheet
Case (b) placed between the top and bottom blocks of small sheets as part of the top block of small sheets.

We assume that a small sheet does indeed fit within the larger sheet, or

one of s_x or $s_y <= l_x$

and the other of s_x or $s_y <= l_y$

In this paper two basic techniques are described to find the maximum number of small sheets that can be fit inside the large sheet. These two techniques use different ways of placing the small sheets, and the best result of the two techniques is the chosen optimum. The first method is labeled Horizontal, described in section 6.2, and the second method is labeled Diagonal, described in section 6.3. The two methods can also be combined into one algorithm. Examples of the methods are given in section 6.4.

Numerous papers have been written about optimizing small pieces of different sizes and shapes: Christofides (1995), Hadjiconstantinou (1995), Nelissen (1995), Herrmann (2001), Delalio (2001), Kroeger (1995), Sharma et al. (1997), Papers dealing with the problem of optimizing identically shaped and sized pieces on rectangular stock sheets are fewer; included are Agrawal (1993) and Arslanov (2000). Agrawal proposes a solution that is similar to our approach, however we add one additional method plus a hybrid method.

6.2. The Horizontal Method

As shown in Figure 6.1, a block of small sheets of one orientation is fit at the top of the large sheet, starting at the left, filling as much as possible along a row, with the s_x dimension parallel to the l_x dimension. A second block of small sheets oriented 90 degrees to the direction of the small sheets in the first block is fit in the bottom portion of the large sheet, in order to maximize the total area utilized of the large sheet. In the large sheet, unused area may result at the right of both the top block and the bottom block, also below the bottom block. The unused area at the bottom may also be placed between the top and bottom blocks of small sheets. It is then necessary to determine the optimal number of rows in each block.

In these procedures, once we chose the orientation of the small sheets as labeled s_x, s_y, that is the orientation we will always chose for the small sheets at the top of the large sheet. Then we rotate the large sheet only, to find other combinations of blocks of small sheets for a best fit. The large sheet is always cut horizontally at the line where the small sheets change orientation. The procedure is repeated for the two cut sheets recursively, until no better fit of small sheets in 90 degree orientations can be found. The concept behind this method is similar to that described in Agrawal's (1993) paper.

Section 6.2.1 describes the determination of the number of rows in each block to maximize the area of the large sheet that is used by the small sheets.

6.2.1 Optimal horizontal placement of two blocks of small sheets

More definitions:

Number of rows of small sheets:

In the top block:	r_t	
In the bottom block:	r_b	

Number of columns of small sheets:

In the top block:	c_t	$= \text{int}(l_x / s_x)$
In the bottom block:	c_b	$= \text{int}(l_x / s_y)$

(where int(n) is the integer portion of n)

Area of the blocks of small sheets:

In the top block:	A_t	$= r_t c_t s_x s_y$
		$= r_t k_t$
In the bottom block:	A_b	$= r_b c_b s_x s_y$
		$= r_b k_b$
Total area used by the small sheets:	A_T	$= A_t + A_b$

Unused area of the large sheet:

Case (a)

To the right of the top block: $U_t = l_x\, r_t\, s_y - A_t$

To the right and bottom of the bottom block: $U_b = l_x\, (l_y - r_t\, s_y) - A_b$

Case (b)

To the right and bottom of the top block: $U_t = l_x\, (l_y - r_b\, s_x) - A_t$

To the right of the bottom block: $U_b = l_x\, r_b\, s_x - A_b$

Total unused area: $U_T = l_x\, l_y - A_t - A_b$

The problem to determine the number of rows (r_t and r_b) in each block of small sheets can be formulated as the maximization of the objective function:

$$A_t + A_b = r_t\, k_t + r_b\, k_b \qquad (1)$$

or minimization of the unused area:

$$U_T = l_x\, l_y - A_t - A_b$$

Subject to:

$$r_t\, s_y + r_b\, s_x <= l_y \qquad (2)$$

where $r_t, r_b >= 0$ (3)

Contraints (2) and (3) can be re-written as:

$$0 <= r_t <= (l_y - r_b\, s_x) / s_y \qquad (4)$$

and

$$0 <= r_t = \text{int}\ ((l_y - r_b\, s_x) / s_y) \qquad (5)$$

since we want the maximum in (1) and the integer portion.

Then (1) becomes:

$$A_t + A_b = \text{int}\ ((l_y - r_b\, s_x) / s_y)\, k_t + r_b\, k_b \qquad (6)$$

Since there is only one variable, r_b, in (6), and in printing we do not have very large numbers of rows, (6) can be solved for the maximum by simple iteration. Thus

$$r_b = 0, 1, 2, 3, \ldots \text{int}\ (l_y / s_y)$$

Once a maximum is obtained, the large sheet can be cut where the small sheets change orientation. However, there is also a need to check whether a 90 degree orientation of the large sheet might give better results. This is described in the next section, as well as the recursive application of the above. Note there may be more than one maximum for (6) for different values of r_b, which implies that we

should follow each of the possible the branches of a tree structure until the path to the best branches are determined. In practice, however, we have found that following every branch at equal maximums did not improve the results over that where we followed only one of the branches leading from one of the maximums.

6.2.2. The Sheet Cutting Horizontal Algorithm

A common method for solving problems with multiple possible outcomes is the tree search procedure, also described by Agrawal (1993), and Christofides et al. (1995). The branches of the tree are followed to find the maximum used area of the large sheet. At each branch, the large sheet may be cut horizontally to create two new large sheets and the procedure is repeated for the two new large sheets for the top and bottom sections. An example is shown in section 6.4.1 and Figure 6.3. The horizontal cut may be made with the unused horizontal area placed at the bottom of the bottom block (Case (a)), or at the bottom of the top block (Case (b)) where the cut is made below the unused area. Both branches of the tree must then be followed.

1. Select an orientation for the small sheet for the top block of small sheets. Which orientation is chosen is arbitrary, unless of course one of the small sheet dimensions is larger than one of the large sheet dimensions.

2. The large sheet can initially also be oriented in one of two ways, with a 90 degree rotation (clockwise or counter clockwise). This forms the initial branching of the tree. The orientation that ultimately results in the least unused area of the large sheet provides the solution. Thus the following steps are repeated for each orientation.

Set the initial $A_{ti} = 0$, $A_{bi} = 0$; $i = 0$.

3. For the values of l_x, l_y, s_x, and s_y:

The maximum area of the small sheets in the top and bottom blocks
$A_T = A_t + A_b$ in (1) is determined by iterating r_b

$r_b = 0, 1, 2, 3, \ldots \text{int}\ (l_y / s_y)$

Thus $r_t = \text{int}\ ((l_y - r_b\, s_x) / s_y)$ (from (5)

and this establishes where the horizontal cut of the large sheet would be made. However, since there may be more than one combination of r_t and r_b at the maximum $A_T = A_t + A_b$, we would expect that we should follow each combination as a branch in the tree. However, if there is more than one combination of r_t and r_b that give the same area in this algorithm we take the first maximum only.

4. For the current iteration i the procedure is complete in the current branch if no improvement is found in the area utilized by the small sheets:

- $A_{Ti} \sim> A_{ti-1}$ or $A_{Ti} \sim> A_{bi-1}$

depending on whether we are following the top or bottom branch.

If both r_{ti} and r_{bi} are greater than zero at a maximum of $A_{ti} + A_{bi}$, the large sheet may be cut horizontally to create two new large sheets where the small sheet blocks change orientation. This creates four possible branches to follow: top and bottom for Case (a), and top and bottom for Case (b).

5. Each of the top and bottom blocks is then considered for further processing for the next iteration .

For the top block:
If the total unused area associated with the top block $U_{ti} = 0$ then the procedure is complete for that block. Otherwise it becomes a new large sheet but is rotated 90 degrees. The rotation is necessary to force consideration of both orientations of the small sheets as possible solutions for the next iteration i +1:

- s_x and s_y remain the same as before.

Case (a):	Case(b):
• $l_{xi+1} = r_{ti}\, s_y$	$l_{xi+1} = l_{yi} - r_{bi}\, s_x$
• $l_{yi+1} = l_{xi}$	$l_{yi+1} = l_{xi}$

- Repeat from Step 3

For the bottom block:
If the total unused area associated with the bottom block $U_{bi} = 0$ then the procedure is complete for that block. Otherwise it becomes a new large sheet but is rotated 90 degrees. The rotation is necessary to force consideration of both orientations of the small sheets as possible solutions for the next iteration i +1:

- s_x and s_y remain the same as before.

Case (a):	Case (b):
• $l_{xi+1} = l_{yi} - r_{ti}\, s_y$	$l_{xi+1} = r_{bi}\, s_x$
• $l_{yi+1} = l_{xi}$	$l_{yi+1} = l_{xi}$

- Repeat from Step 3

The branches at the end of the tree that sum to the least unused area U_T provide the optimal solution. What remains after the above steps are completed is to continue cutting the small sheets from each large sheet, which is not described further here.

6.3. The Diagonal Method

The method in this section involves placing an orientation of small sheets at the top and left sides of the larger sheet (the “top” section – an inverted L shape),

and abutting a block of small sheets oriented 90 degrees to the previous small sheets, below and to the right (the "bottom" block), as shown in Figure 6.2. For the maximum utilization of the large sheet, the optimum values of rows and columns of each orientation must be determined.

For the small sheet dimensions, s_x is the dimension parallel to l_x in the "top" section; s_y is the dimension parallel to l_y in the "top" section.

The maximum number of columns c along at the top of the "top" section is a constant:

$$c = \text{int}\,(l_x / s_x)$$

The maximum number of rows r along the left of the "top" section is a constant:

$$r = \text{int}\,(l_y / s_y)$$

The number of rows at the top of the "top" section is r_t. The number of columns at the left of the "top" section is c_t. The number of rows of the "bottom" block is r_b; the number of columns is c_b.

Areas of the sections of small sheets:

In the "top" block: $A_t = r_t\, c\, s_x s_y + (r - r_t)\, c_t\, s_x s_y$

In the "bottom" block: $A_b = r_b\, c_b\, s_x s_y$

The unused area of the large sheet is thus:

$$U_T = l_x\, l_y - A_t - A_b$$

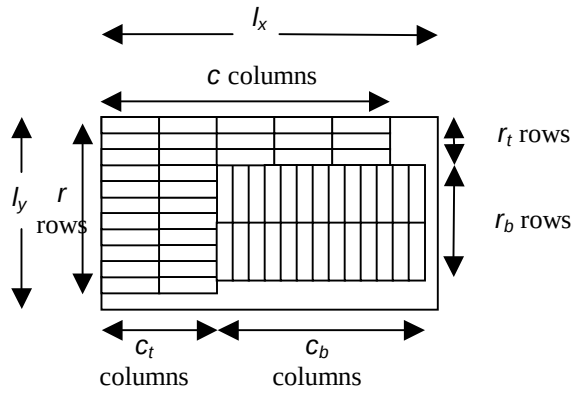

Figure 6.2

The problem to determine the number of rows and columns of small sheets (r_t, c_t, r_b, c_b) can be formulated as the maximization of the objective function:

$$A_T = A_t + A_b = r_t c\, s_x s_y + (r - r_t)\, c_t\, s_x s_y + r_b\, c_b\, s_x s_y \qquad (7)$$

Subject to:

$$c_t s_x + c_b s_y <= l_x \qquad (8)$$

$$r_t s_y + r_b s_x <= l_y \qquad (9)$$

where $r_t, r_b\, c_t, c_b >= 0$ (10)

Contraint (8) can be re-written as:

$$0 <= c_b <= (l_x - c_t s_x) / s_y \qquad (11)$$

and

$$0 <= c_b = \text{int}\,((l_x - c_t s_x) / s_y) \qquad (12)$$

since we want the maximum in (7) and the integer portion.

Similarly, constraint (9) can be re-written as:

$$0 <= r_b <= (l_y - r_t s_y) / s_x \qquad (13)$$

and

$$0 <= r_b = \text{int}\,((l_y - r_t s_y) / s_x) \qquad (14)$$

Thus the objective function (7) contains two variables r_t and c_t and becomes:

$$A_t + A_b = r_t c\, s_x s_y + (r - r_t)\, c_t\, s_x s_y + \text{int}\,((l_y - r_t s_y) / s_x)\ \text{int}\,((l_x - c_t s_x) / s_y)\ \ s_x s_y \qquad (15)$$

The maximum of (15) can be obtained by iterating with all possible combination of values of r_t and c_t:

$$r_t = 0, 1, 2, \ldots \text{int}\,(l_y / s_y) \quad (16)$$

$$c_t = 0, 1, 2, \ldots \text{int}\,(l_x / s_x) \quad (17)$$

Note that there may be multiple solutions for r_t and c_t at the maximum of (15). If at the maximum of (15) r_t or c_t are at the limits in (16) or (17), then the problem reverts to the Horizontal case (no L shape for the "top" section).

6.4. Examples

In these three examples, the Horizontal and Diagonal methods are applied. In the first two examples, each method is used separately and the results are compared – in one the Horizontal method is superior, in the other the Diagonal method is superior. In the third example, the methods are mixed, with the Diagonal method also used for each of the top and bottom sections determined by the Horizontal method at each branch.

The examples illustrate that there may be many solutions with the same area utilization of the large sheet. A mix of the Horizontal and Diagonal methods, with the Diagonal method providing another branch of the tree to follow, provides a greater chance of obtaining a better solution.

6.4.1. Example One - Horizontal Method Superior

The small sheet is 7 x 3 = 21. The large sheet is 51 x 32 = 1632. The solution is shown in Figure 6.3, leaving 15 units of unused area, which is less than the area of a small sheet, as determined by the Horizontal Method in section 6.2.2. Three guillotine cuts are required to separate the blocks of small sheets, which would then be followed by cutting out the individual small sheets. The Diagonal method in this case provides multiple solutions with a greater unused area of U_T = 79 and U_T = 36 for the two orientations of the large sheet, as shown in Table 6.1.

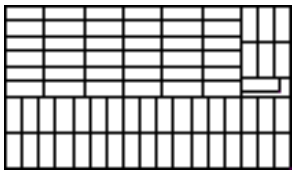

Figure 6.3

Table 6.1. Example 1 - Diagonal Method for large sheet 51 x 32 and small sheets (s_x x s_y,) = (7x 3)

	(l_x x l_y) = (51 x 32) A_T = 1553 U_T = 79				(l_x x l_y) = (32 x 51) A_T = 1596 U_T = 36	
c_t	3	1	2	3	2	2
r_t	1	6	6	6	3	0
c_b	10	14	12	10	6	6
r_b	4	2	2	2	6	7

Table 6.2 shows the iteration and branching pathways for the Horizontal Method of section 6.2.2. Only Case (a) is shown for simplicity, and only for one initial orientation of the large sheet (l_x x l_y) = (51 x 32). Four iterations are shown, and in this simple example, each of the branches is part of the optimal solution (other optimal solutions or branches are not shown). At each iteration the large sheet is cut into the Top and Bottom blocks, into successively smaller sections where the branching occurs. The final solution for the total area used by the small sheets is given by the sum of the end branches (21 + 126 + 756 + 714 = 1617).

<table>
<tr><td colspan="6">Table 6.2 Example 1 – Horizontal Method for large sheet (l_x x l_y) = (51 x 32) and small sheets (s_x x s_y,) = (7x 3). Final A_T = 1617 U_T = 15</td></tr>
<tr><td>i</td><td colspan="5"></td></tr>
<tr><td rowspan="2">1</td><td colspan="5">(l_x x l_y) = (51 x 32)
A_T = 1596 U_T = 36</td></tr>
<tr><td colspan="4">Top
(l_x x l_y) = (51 x 18)
r_t = 6
A_t = 882 U_t = 36</td><td>Bottom
(l_x x l_y) = (51x 14)
r_b = 2
A_b = 714
U_b = 0</td></tr>
<tr><td rowspan="2">2</td><td colspan="4">(l_x x l_y) = (18 x 51)
A_T = 882 U_T = 36</td><td rowspan="2"></td></tr>
<tr><td colspan="3">Top
(l_x x l_y) = (18 x 9)
r_t = 3
A_t = 126 U_t = 36</td><td>Bottom
(l_x x l_y) = (18 x 42)
r_b = 6
A_b = 756 U_b = 0</td></tr>
<tr><td rowspan="3">3</td><td colspan="3">(l_x x l_y) = (9 x 18)
A_t = 147 U_t = 15</td><td colspan="2" rowspan="4"></td></tr>
<tr><td colspan="2">Top
(l_x x l_y) =(9x3)
r_t = 1
A_t = 21
U_t = 6</td><td colspan="2">Bottom
(l_x x l_y) =(9x15)
r_b = 2
A_b = 126
U_b = 9</td></tr>
<tr><td rowspan="2">4</td><td colspan="2">(l_x x l_y) =(3x9)
A_T = 21
U_T = 6</td><td colspan="2">(l_x x l_y) =(15x9)
A_b = 126
U_b = 9</td></tr>
<tr><td>Top
(l_xx l_y)
=(0x0)
r_t = 0
A_t = 0
U_t = 0</td><td>Bottom
(l_x x l_y)
=(3x9)
r_b = 1
A_b = 21
U_b = 6</td><td>Top
(l_xx l_y)
=(15x9)
r_t = 3
A_t = 126
U_t = 9</td><td>Bottom
(l_x x l_y)
=(0x0)
r_b = 0
A_b = 0
U_b = 0</td></tr>
</table>

6.4.2. Example Two - Diagonal Method Superior

The small sheet is 7 x 3, the large sheet is 50 x 40. The Diagonal Method solution is shown in figure 6.4 and Table 6.3, with the unused area U_T = 26 for the (50 x 40) orientation, and U_T = 47 for the (40 x 50) orientation. The best obtained with the Horizontal Method is U_T = 47.

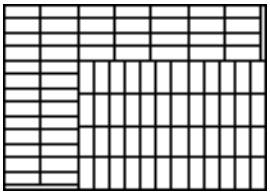

Figure 6.4.

Table 6.3. Example 2 - Diagonal Method for large sheet 50 x 40 and small sheets 7 x 3. (s_x x s_y,) = (7 x 3)		
	(l_x x l_y) = (50 x 40) A_T = 1974 U_T = 26	(l_x x l_y) = (40 x 50) A_T = 1953 U_T = 47
c_t	2	1
r_t	4	0
c_b	12	11
r_b	4	7

6.4.3 Hybrid Horizontal and Diagonal Methods

The algorithm that incorporates the Horizontal and Diagonal methods is diagramed in Figure 6.5. An example of the result is shown in Figure 1.2 for larger sheets 9,885 x 6,165 and small sheets 312 x 91. Starting with the large sheet (x x y), the Horizontal method is called twice, once for orientation (x x y), and once for the reversed orientation (y x x), shown by the circle with the letter H. Similarly, the Diagonal method is called, once for (x x y), and once for the reversed orientation (y x x), shown by the circle with the letter D. These four form the initial branches of the tree to search for the highest utilization of the area of the large sheet by the small sheets. The Diagonal method branches do not branch further.

The Horizontal branches, however, split into another four further branches. The current "large sheet" is cut horizontally forming the top and bottom sections, each of which become new "large sheets"with the x and y dimensions reversed. There are two top sections (case (a) and case (b)), and two bottom sections (case (a) and case (b)). Recall that in case (a) any unused horizontal area is kept with the bottom section, and in case (b) any unused horizontal area is kept with the top section.

Each of the four Horizontal branches is further split into three branches. The Horizontal method is called recursively, and the Diagonal method is called twice, reversing the current "large sheet" x and y dimensions.

The process continues in this manner, with the Horizontal branches continuing to create more branches, while the Diagonal branches terminate. The Horizontal branches terminate when the condition stated in section 6.2.2, step 4 is met. The information from a terminating node is passed back up to the previous node, where the total area utilized by the small sheets is compared. The data from the branch with the highest area is passed further up to the higher node, while the other results are discarded. If there is more than one solution at a node with the same total area utilized, only one is chosen. Note that solutions with different resulting configurations (but the same total area utilization) may be obtained by reversing the initial x and y dimensions provided to the algorithm (for both the large and small sheets).

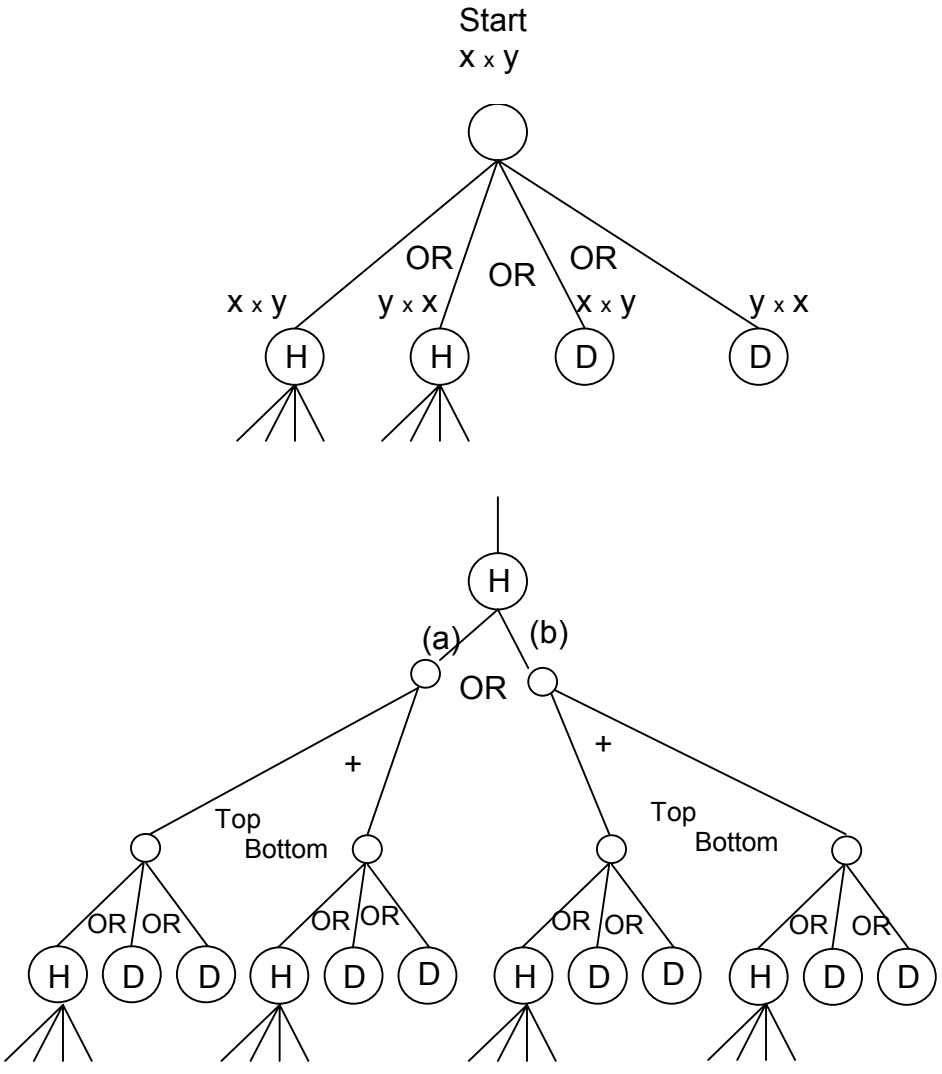

Figure 6.5

7. Conclusion

1. There are often multiple solutions for optimum arrangement of identical small rectangular sheets within a large rectangular sheet. The Horizontal Method involves repeated division of the large sheet into smaller sheets, making horizontal cuts each time, where each of the two resulting sheets has orthogonal orientations of rectangular blocks of the smaller sheets. The Diagonal Method also involves orthogonal arrangement of sections of the smaller sheets, but one of the sections is “L” shaped, while the other is rectangular.

2. The Horizontal Method branches to as many levels as necessary, with each branch or iteration generating more orthogonal blocks. It was found that the best optimization results required in the great majority of cases only two or three blocks, and only in a very small number of cases were four or more blocks required. This trend indicates that although even better optimization results should be possible beyond the fourth orthogonal block, the probability of this occurring diminishes with each additional block. Furthermore, additional orthogonal blocks reduce the possible unused area to a point where the resulting solutions would become too small for practical applications.

3. In some cases the Horizontal Method by itself provides one or more optimum arrangements, in other cases the Diagonal Method by itself provides one or more optimum, or both methods find an optimum which may differ, but utilize the same area of the large sheet. In general, however, a combination of the two methods, where each step of the Horizontal Method also has the Diagonal Method applied to each of the two sub-sheets, may provide other optimum arrangements or a better solution than either method alone.

4. The Horizontal and Diagonal algorithms were implemented in Visual Basic .Net code and its performance was tested against several commercial optimisation programs. In no instance were the results obtained inferior and often the results were superior. Processing time on an Intel II processor was under one second with the most complex problems. As a fast and robust algorithm it could therefore find application as an efficient cutting pattern generator for programmable guillotines.

References

Agrawal, P.K., *"Minimizing trim loss in rectangular blanks of a single size from a Rectangular sheet using orthogonal guillotine cuts"*, European Journal of Operational Research 64, pp. 410-422, 1993.

Arslanov, M.Z., *"Continued fractions in optimal cutting of rectangular sheet into equal small rectangles"*, European Journal of Operational Research 125, pp. 239-248, 2000.

Breede, M.H., *"Handbook of Graphic Arts Equations"*, Sewickley, PA: Graphic Arts Technical Foundation, p. 22, 1999.

Christofides, N., Hadjiconstantinou, E., *"An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts"*, European Journal of Operational Research 83, pp. 21-37, 1995.

Cogoli, J.E., *"Photo-Offset Fundamentals"*, Bloomington, Ill.: McKnight Publishing Company, p. 304, 1973.

Dyckhoff, H., Finke U., *"Cutting and Packing in Production and Distribution: A Typology and Bibliography"*, Heidelberg, Germany: Physica-Verlag, 1992.

Herrmann, J., Delalio, D., *"Algorithms for Sheet Metal Nesting"*, IEEE Transactions on Robotics and Automation, Vol. XX, pp. 100-107, 2001.

Kippan H., *"Handbook of Print Media"* , Heidelberg, Germany: Springer-Verlag, pp. 782-789, 2001.

Kroeger, B., *"Guillotineable bin packing: A genetic approach"* , European Journal of Operational Research 84, pp. 645-661, 1995.

Nelissen, J., *"How to use structural constraints to compute an upper bound for the pallet loading problem"*, European Journal of Operational Research 84, 662-680, 1995.

Sharma R., Balachander, T., McCord, C., Anand, S., Zhang, Q., *"Genetic Algorithms For the Single-Sheet and Multi-Sheet Non-Convex Cutting Stock Problem"*, Cincinnati, OH: University of Cincinnati, 1997.

Valentino, C., *"Saving Paper Money."* Print Profit, - Volume 4 – Number 2, Paramus, NJ: National Association for Printing Leadership, Summer 2003.

Wilson L.A., *"What The Printer Should Know About Paper"*, third edition, Sewickley, PA: Graphic Arts Technical Foundation, pp. 232-235, 1998.