KM 30(1989)
pag 69 - 88

Lumberproduction Optimization

M.P. Reinders[1]

Th.H.B. Hendriks[1]

Abstract.

An algorithm is developed to optimize the conversion of trees into lumber. The algorithm is based on nested dynamic programming sub-algorithms.
Although primarily developed for that purpose, applications in other fields are also possible because arbitrary shapes can be cut into rectangular products in a near to optimal way.

KeyWords: Decomposition, Dynamic Programming, Computational Analysis, Lumber production.

1) Department of Mathematics, Agricultural University Wageningen, De Dreijen 8, 6703 BC WAGENINGEN, tel.: 08370 - 83286/83285.

1. Introduction

Wood is a raw material that is becoming increasingly scarce. Especially large high-quality logs are harder to get every year. So sawmills using trees as an input material for the production of lumber have to produce their products more and more from smaller trees (Hallock et al. 1976, 1979; Christensen 1986).

As a result of this development, the process of producing lumber from trees needs better control in order to obtain a higher recovery. This process control can be improved by heightening the sawing accuracy, (Stern et al. 1979), and pattern optimization (Faalands 1984; Geerts 1984).

The production of lumber can be considered roughly as a two-phase process: firstly the primary breakdown of timber, resulting in assortments cut from the tree, a process called crosscutting; secondly the sawing of assortments into boards of various sizes.

The problem of finding an optimum sawing pattern can be stated as a three-dimensional knapsack problem. Non-guillotine cuts across the tree are not realistic, due to limitations of today's production technology. So a breakdown can be made into the well known one-dimensional knapsack problem representing the primary breakdown, and the two-dimensional knapsack problem modelling the breakdown of an assortment into boards. The latter problem can be stated as fitting rectangles of various dimensions and values into a circle in such a way that the total value of the circle is maximized. Optimizing the sawing patterns has been a subject of interest to various researchers (e.g. Faaland & Briggs 1984; Geerts 1984; Hallock et al. 1979).

Algorithms to optimize the sawing of logs should be examined for both effectiveness and efficiency. Effectiveness is relevant in the long term continuity of the company and the scarcity of the raw material. Efficiency, however, is also relevant because of the on-line application of the algorithm of interest.

In this paper, the emphasis will be on pattern generation and selection. The criterion used to select a typical sawing pattern will be maximization of the value of the timber on the basis of a given list of prices and dimensions of boards to be cut. Because of efficiency, much attention has been paid to speeding up the basic algorithm.

2. Basic assumptions

A tree has to be cut into boards. The boards, in our case called final products, are fully characterized by their dimensions and their value. The quality aspect is not dealt with in this algorithm, although it can also be included by small extensions.

A three-dimensional breakdown pattern has to be found in a way that the tree value is maximized according to the final product pricelist. We performed a decomposition of the problem into three levels. The decisions on each level interact with another to perform a global optimum.

On level one a problem is to find positions to break the tree down into a number of logs. The cuts are perpendicular to the axis of the timber (cross-cutting). The prices of the saw logs are in fact the value of the logs when they are cut into lumber. This is the next step of the optimization. On level two the logs are cut into flitches, mother boards, in a way that the log is maximized according to the flitch values. The flitch values, however, are a result of the final products cut from it. To maximize the value of these flitches, we have to perform a one-dimensional knapsack problem on level three. Figure one shows this way of breaking up a tree.

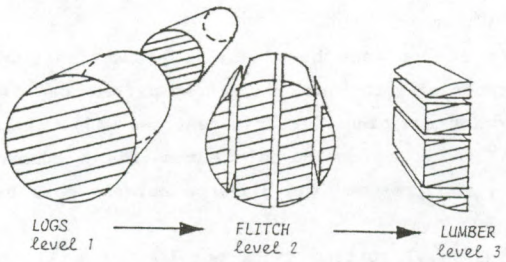

fig. 1. Conversion from tree into lumber.

When breaking up a log in this way, it must be realized that one does not use all degrees of freedom that come with a three-dimensional knapsack problem. When we call the length along the axis the z direction, and along the width, horizontally, the x direction, and the direction perpendicular to these two, the y direction, the following can be concluded. Cuts made perpendicular to the z direction divide the tree into logs. We cannot make end products longer than these logs. The guillotine character of these cuts sets limits on the end products to be cut. Although to our knowledge, a system with non-guillotine cuts in the z direction does not exist at this

moment, it would possibly lead to higher relative recovery.

A second limitation occurs at the next processing step to cut a log into final products. We assume the cuts in the y direction to be of a guillotine type, with respect to the cross-section (we will call this parallel guillotine cuts, see figure 2b). This is a somewhat different approach from that usual in the literature. See for rectangles Gilmore et al. (1969).

Technically a log breakdown pattern with guillotine cuts would be possible, but a serious drawback would be the number of rotations necessary to make the cuts (See figure 2a).

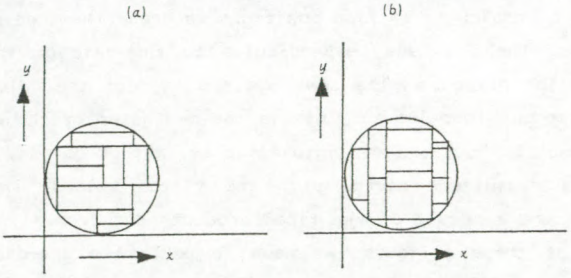

fig. 2. (a) *Guillotinecuts*
(b) *Parallel guillotinecuts*

The first pattern can be made by classical production equipment but only by means of a number of rotations. The second pattern can also be made with classical production equipment. In this text, we will only allow patterns of type two. This is a considerable extension in comparison with Faaland et al. (1985) who presumed the distance between cuts parallel to the y axis fixed.

To solve the three-level cutting stock problem, we will use a model that consists of three nested, one-dimensional dynamic programming routines (DP routines), that interact with each other for their value function and decisions at the various moments of decision, called stages in the dynamic programming literature. (Geerts, 1984). In our research the effect of the waney edge, being the waney part of the crosssection, is fully used in the optimization routine in contrast to the algorithm used by Geerts (1984) that dealt with the waney edge heuristically. A special problem is formed by the shape of the tree, that does not allow us to make all suggestions for speeding up made by Christofides & Whitlock (1977), in optimizing the cutting of rectangular motherboards.

3. Introduction to Effects of Symmetry[2] and equality and similarity

When performing an algorithm based on nested one-dimensional DP routines, we can use effects of symmetry to save time in finding the optimum pattern. At the first level, the z-axis, there is no symmetry whatsoever. On the xy-level there are two major effects.

1) When "slicing" a circle, the value of a typical trial flitch on the left has the same value at the opposite position at the right side, This holds only for uniform quality.
2) A second effect is the fact that a flitch containing a part that has been optimized before, does not need a new calculation. For instance if a flitch contains a rectangle that has been calculated before, we can skip the calculation of the rectangle in the flitch. We will call this an effect of equality and similarity.

4. Geometry

Let Lt be the length of a tree divided into intervals of length Δz along the z-axis. If Lt is not an integer multiple of the unit length, then waste is formed at the top. The usable part of the tree is of length Le. (effective length). The diameter of the tree is a function of the distance from the stump, and can be denoted as $D(z.\Delta z)$ where z is the number of the unit interval.

The moments when decisions can be made are symbolized by z and called the stages in the z direction. The states are the distances from the stump of the processed part of the tree, and can thus be denoted as $z.\Delta z$. The decision is the length to cut off. To evaluate the quality of the decision, one has to know the value of the part of the tree to cut off at stage z. This length will be called λ_z. The value of the length λ_z can be calculated by means of two-dimensional dynamical programming (DP).

At the x level, the stages can be defined as the number of the unit interval, with numbering starting on the left side of the circular cross-section. The state can thus be defined, analogous to the z direction, as

2) Special thanks to the student M.A. Kramer who performed the programming of the effects of symmetry into the algorithm.

the distance from the left of the cross-section. As a result of these definitions again, we have a one-to-one correspondence between stages and states. The decision to be made at stage x is the width of the flitch to cut off, we will denote this flitch width, at stage x, at position z in the shaft as ϕ_{zx} (Figure 3).

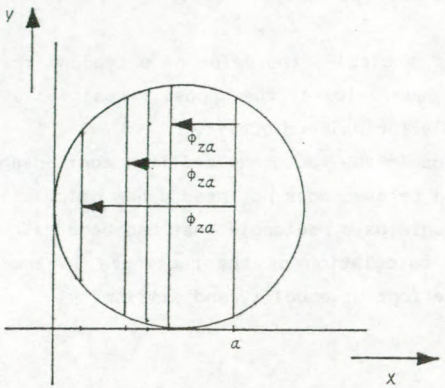

fig. 3. Possible decisions at x = a.

If the diameter is not an integer multiple of the unit interval length Δx, there is a small circle of waste at the exterior, the thickness of which depends on the grid width projected on the cross-section.

The quality of a decision ϕ_{zx} can be examined by the value of the flitch resulting from this decision. The value of the flitch is defined as the sum of the values of the lumber products that are produced from it. To maximize this value, stages in the y direction are defined as the number of the interval unit with length Δy, starting in the x axis at y=0. The decision to be made at each stage y is the number of the product to cut off at stage y, at position z,x, either rotated or not, called $(n,\rho)_{zxy}$, where n is the product number and ρ is the rotation factor, being 0 if no rotation occurs and 1 after rotation through 90°. The quality of this decision depends on the value of the product $v((n,\rho)_{zxy})$, and the value of the part of the flitch left after this decision $(n,\rho)_{zxy}$. Figure 4 shows the relationships.

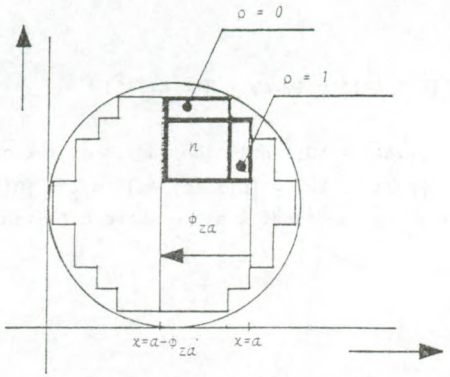

fig. 4. Possible decisions at the y-level.

A tree can be viewed as built up of unit elements Δz, Δx, Δy, as done in figure 5.

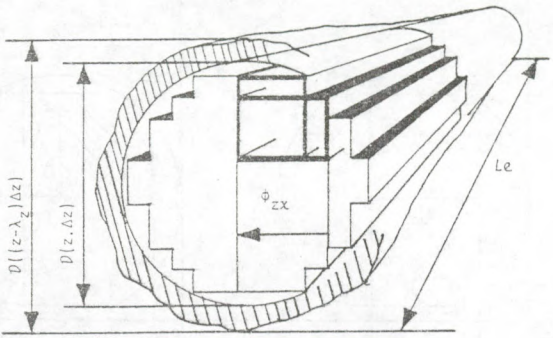

fig. 5. A shaft build up out of unit elements.

Let us now consider the geometric aspects of breaking down a circular cross-section into rectangles. If at a stage x, at z interval distances from the stump, the decision is made to cut off a flitch of width $\phi_{zx}.\Delta x$, then the functions as shown in Figure 6 have to be considered. The g-function is used for the calculation of the available flitch-height on a width x. The h-function has a similar role for calculating the flitch width at a height y.

They can be expressed as:

$$g(z,x): = \sqrt{(R(z.\Delta z)^2 - (x.\Delta x - R(z.\Delta z))^2)} \qquad (4.6)$$

$$h(z,y) := \sqrt{(R(z.\Delta z)^2 - (y.\Delta y - R(z.\Delta z))^2)} \qquad (4.7)$$

In these functions, $R(z.\Delta z) = \frac{1}{2}D(z.\Delta z)$, the radius of the cross-section. Furthermore let $Nz = [Lt/\Delta z]$, $Nx_z = [D(z.\Delta z)/\Delta x]$, $Ny_z = [D(z.\Delta z)/\Delta y]$, where [.] means entier (.) x_z^{α} is the first x stage where a flitch can be produced, x_z^{ω} the last.

The definitions are shown in Figure 6.

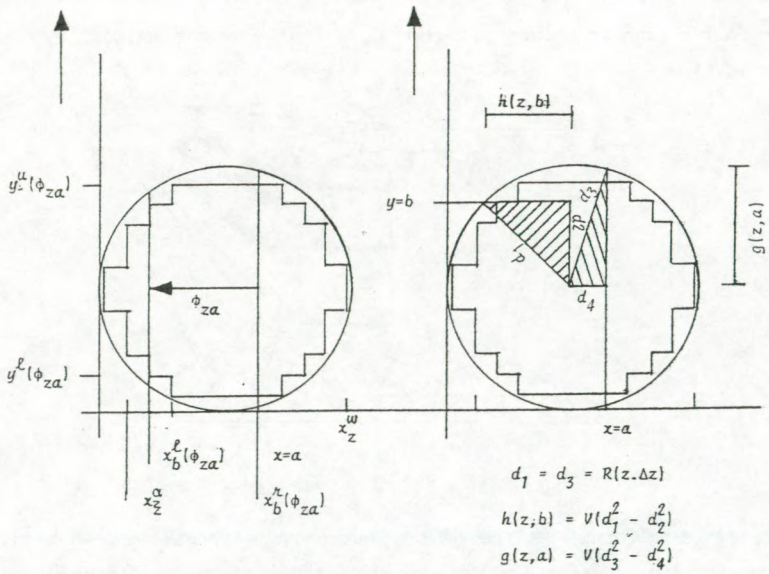

fig. 6 Boundary values for x = a and y = b.

$$x_z^{\alpha} := \min_{1 \leq x \leq Nx_z} \{x \mid [2g(z,x)/\Delta y] \geq 1\} + 1 \qquad (4.8)$$

$$x_z^{\omega} := \max_{[Nx_z/2] \leq x \leq Nx_z} \{x \mid [2g(z,x)/\Delta y] \geq 1\} \qquad (4.9)$$

The effective height of the flitch is a function of the decision ϕ_{zx} made at stage x. If a flitch width is bigger, the available shape of this

flitch will become less rectangular. y^{ℓ} and y^{u} are the lower and the upper y stages, respectively, that can be used to make a decision which product should be produced from the flitch, dependent on the width of the flitch.

$$y^{\ell}(\phi_{zx}) := \min_{x-\phi_{zx} \leq u \leq x-1} \{[(R(z.\Delta z) - g(z,u))/\Delta y]\} + 2 \qquad (4.10)$$

$$y^{u}(\phi_{zx}) : \max_{x-\phi_{zx} \leq u \leq x-1} \{[(R(z.\Delta z) + g(z,u))/\Delta y]\} \qquad (4.11)$$

Because the flitch for large ϕ_{zx} is often not rectangular, the effective x range is not $(x-\phi_{zx})$ in most y stages. Therefore we define x^{ℓ} and x^{r} as the left and the right usable x coordinates, respectively, in order to determine the resulting flitch width at a position x if an overall width of ϕ_{zx} is cut off.

$$x_y^{\ell}(\phi_{zx}) := \max\{x - \phi_{zx}, [(R(z.\Delta z) - h(z,y))/\Delta x] + 1\} \qquad (4.12)$$

$$x_y^{r}(\phi_{zx}) := \min\{x, [(R(z.\Delta z) - h(z,y))]/\Delta x\} \qquad (4.13)$$

On the y level, a decision is made whether to cut off a product with number n, and whether to rotate the product 90° or not. The decision is two-dimensional and can be stated as $(n,\rho)_{zxy}$. The usable flitch width at a level y, when choosing a product with number n can be stated as:

$$W(\phi_{zx},y,n,\rho) := \min\{x_y^{r}(\phi_{zx}) - x_y^{\ell}(\phi_{zx}),\ x_{y'}^{r}(\phi_{zx}) - x_{y'}^{\ell}(\phi_{zx})\}$$

with

$$y' = y - ((1-\rho)\ell^{y}(n) - \rho\ell^{x}(n))/\Delta y$$

where $\ell^{x}(n)$ and $\ell^{y}(n)$ represent the lengths in the x and y directions of product n.

5. The model

The model is here introduced more formally, based upon the definitions given in the previous sections.

Consider a set of numbered products P. A product p(n) can be characterized as:

$$p(n) := (\ell^z(n),\ \ell^x(n),\ \ell^y(n),\ v(n)) \tag{4.14}$$

The numbers are given to the products in a way that the products are ordered lexicographically with respect to their dimensions.

$$P := \{p(n)\} \tag{4.15}$$

At the z and x level, lengths and widths have to be cut, respectively. These quantities are directly related to the product dimensions.
Two ordered sets Λ and Φ containing the lengths and widths allowed to be cut at the various z and x stages are introduced:

$$\Lambda := \{\lambda \mid \exists n[\ \lambda.\Delta z = \ell^z(n)]\} \tag{4.16}$$

$$\Phi := \{\phi \mid \exists n[\ \phi.\Delta x = \ell^x(n) \text{ or } \phi.\Delta y = \ell^y(n)]\} \tag{4.17}$$

When making a decision in the z direction, at a stage z, lengths $\lambda.\Delta z$ greater than the length $z.\Delta z$ cannot be used. So we introduce

$$(\Lambda)_z := \Lambda \setminus \{\lambda \mid \lambda > z\} \tag{4.18}$$

The same is true for the decision at stage x, at position z from the stump end. The decision has to be made which width ϕ to use, but ϕ values greater than the width of the cross-section at stage x cannot be used. Also widths ϕ that only occur with products that have a length $\ell^z(n)$ greater than the $z.\Delta z$ at this moment of decision are not usable. In conclusion, we can state

$$(\Phi)_{\lambda_z x} := \Phi \setminus (\{\phi \mid \forall n[(\ell^x(n) = \phi.\Delta x \text{ or } \ell^y(n) = \Phi.\Delta y) \text{ and } \ell^z(n) > \lambda_z.\Delta z]\}$$

$$\cup \{\phi \mid \phi > x\}) \tag{4.19}$$

At the y level, we have to decide which product to cut off and whether to rotate the product or not, once decisions λ_z and ϕ_{zx} have been suggested. The set of possible combinations of product number and rotation can be stated as:

$$(NR)_{\lambda_z \phi_{zx} y} := \{(n,\rho) \mid y - (1-\rho)\ell^y(n)/\Delta^y - \rho\ell^x(n) > 0 \text{ and}$$

$$(1-\rho)(W(\phi_{zx},y,n,\rho) - \ell^x(n)/\Delta^x) + \rho(W(\phi_{zx},y,n,\rho) - \ell^y(n)/\Delta x) > 0 \text{ and}$$

$$\ell^z(n) = \lambda_z\} \tag{4.20}$$

Furthermore we define:

$G_z(\lambda_z)$: The maximum reachable value added after a decision λ_z. It is the value of a knotted cone of length λ_z and a top diameter $D(z.\Delta z)$ and foot diameter $D((z.\Delta z)')$ where $(z.\Delta z)' = (z-\lambda_z)\Delta z$

ZF_z : Value function of the system at stage z, if at all relevant previous stages optimum decisions λ_z have been made. It can be considered as the value of the processed part of the tree.

$G_{zx}(\lambda_z, \phi_{zx})$: The value of a part of the cross-section with width ϕ_{zx} starting at position x.

$XF(\lambda_z)_{zx}$: Value function of the section in stage x, if at all relevant previous stages optimum decisions ϕ_{zx} have been made.

$G_{zx}(\lambda_z, \phi_{zx}, (n,\rho)_{zxy})$: The value added after a decision $(n,\rho)_{zxy}$ at stage y. It can be the value v(n) of product n.

$YF(\lambda_z, \phi_{zx})_{zxy}$: The valuefunction of the system at stages y, if at all relevant previous stages optimum decisions $(n,\rho)_{zxy}$ have been made.

With the definition given in this paragraph and the information in the previous section, the model can be summarized as:

$$1)\quad ZF_z = \max_{\lambda_z \in (\Lambda)_z} \{G_z(\lambda_z) + ZF_{z-\lambda_z}\}$$

$$1 \leqslant z \leqslant Nz$$

$$G_z(\lambda_z) = XF(\lambda_z)_{zx_z^{\omega}}$$

$$2)\quad XF(\lambda_z)_{zx} = \max_{\phi_{zx} \in (\Phi)_{\lambda_z x}} \{G_{zx}(\lambda_z, \phi_{zx}) + XF(\lambda_z)_{zx-\phi_{zx}}\}$$

$$x_z^{\alpha} \leqslant x \leqslant x_z^{\omega}$$

$$G_{zx}(\lambda_z, \phi_{zx}) = YF(\lambda_z, \phi_{zx})_{zxy}\ (\phi_{zx})$$

$$3)\quad YF(\lambda_z, \phi_{zx})_{zxy} = \max_{(n,\rho)_{zxy} \in (NR)_{\lambda_z \phi_{zx} y}} \{G_{zxy}(\lambda_z, \phi_{zx}, (n,\rho)_{zxy}) + YF(\lambda_z, \phi_{zx})_{zxy'}\}$$

$$y' = y - (1-\rho)\ell^{y}(n_{zxy})/\Delta y - \rho\ell^{x}(n_{zxy})/\Delta y$$

$$y^{\ell}(\phi_{zx}) \leqslant y \leqslant y^{u}(\phi_{zx})$$

The model described in the previous section can be turned into an algorithm that performs the optimum primary and secondary breakdown. However we can speed up such an algorithm by using the effects of symmetry described in Section 3.

6. Symmetry

At a level x, suppose that different values for ϕ_{zx} are checked. For x greater than $[Nx_z/2]$, a great chance occurs that the trial flitch has already been optimally cut into final products at a stage before x.

If $x - [\phi_{zx}/2]$ is greater than $[Nx_z/2]$, the flitch has already been optimised at stage $Nx_z - x + \phi_{zx}$.

Hence,

$$\forall\phi_{zx}[x-[\phi_{zx}/2] > [Nx_z/2] \rightarrow YF(\lambda_z,\phi_{zx})_{zxy^u(\phi_{zx})} = YF(\lambda_z,\phi_{za})_{zay^u(\phi_{za})}] \quad (4.21)$$

with $a := Nx_z - x + \phi_{zx}$ and $\phi_{za} = \phi_{zx}$.

A second case of symmetry occurs if part of a flitch is rectangular and has been optimized before.

A flitch can only be a rectangle if the following two equations hold; this situation is expressed in figure 6a.

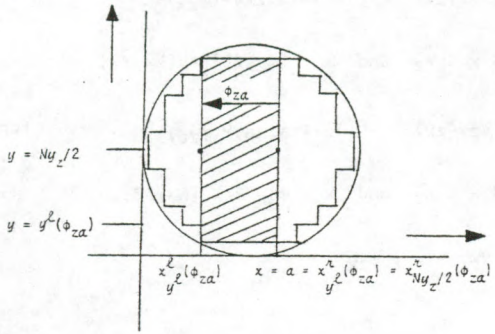

fig. 6a. A rectangular flitch.

$$R(1) : x^{\ell}_{y^{\ell}(\phi_{zx})} = x^{\ell}_{Ny_z/2}(\phi_{zx}) \quad (4.22)$$

$$R(2) : x^{r}_{y^{\ell}(\phi_{zx})} = x^{r}_{Ny_z/2}(\phi_{zx}) \quad (4.23)$$

Suppose x_1 and x_2 with $x_2 > x_1$ are both stages, at which a decision $\phi_{zx_1} = \phi_{zx_2}$ can be made, for which R(1) and R(2) hold, then at stage x_2 one can skip the calculations until $y - y^{\ell}(\phi_{zx_2})$ becomes greater than $y^u(\phi_{zx_1}) - y^{\ell}(\phi_{zx_1})$.

Hence,

$$\forall x_1\forall x_2{>}x_1[\ \phi_{zx_1} = \phi_{zx_2} \rightarrow \forall y{\leq}y^{\ell}(\phi_{zx_2})+y^u(\phi_{zx_1})-y^{\ell}(\phi_{zx_1})\ [YF(\lambda_z,\phi_{zx_2})_{zx_2y} =$$

$$YF(\lambda_z,\phi_{zx_1})_{zx_1y^{\ell}(\phi_{zx_1})} + y]] \quad (4.24)$$

Implementing the effects of both symmetry and equality and similarity, the adjusted model is as follows.

1) $ZF_z = \max_{\lambda_z \epsilon (\Lambda)_z} \{G_z(\lambda_z) + ZF_{z-\lambda_z}\}$

$1 \leqslant z \leqslant Nz$

$G_z(\lambda_z) = XF(\lambda_z)_{zx_z^{\omega}}$

2) $XF(\lambda_z)_{zx} = \max_{\phi_{zx} \epsilon (\Phi)_{\lambda zx}} \{G_{zx}(\lambda_z, \phi_{zx}) + XF(\lambda_z)_{zx-\phi_{zx}}\}$

a) $G_{zx}(\lambda_z, \phi_{zx}) = YF(\lambda_z, \phi_{zx})_{zxy^u(\phi_{zx})}$ for

$x_z^{\alpha} \leqslant x \leqslant x_z^{\omega}$ and $x - [\phi_{zx}/2] < [Nx_z/2]$

b) $G_{zx}(\lambda_z, \phi_{zx}) = YF(\lambda_z, \phi_{za})_{zay^u(\phi_{za})}$ for

$x_z^{\alpha} \leqslant x \leqslant x_z^{\omega}$ and $x - \phi_{zx}/2 \geqslant [Nx_z/2]$, with

$a = Nx_z - x + \phi_{zx}$

3) $YF(\lambda_z, \phi_{zx})_{zxy} = \max_{(n,\rho)_{zxy} \epsilon (NR)_{\lambda_z \phi_{zxy}}} \{G_{zxy}(\lambda_z, \phi_{zx}, (n,\rho)_{zxy}) + YF_{zxy'}$ for

$y^{\ell}(\phi_{zx}) \leqslant y \leqslant y^u(\phi_{zx})$ and

not (R(1) and R(2)),

$y' = y-(1-\rho)\ell^y(n_{zxy})/\Delta y-\rho\ell^x(n_{zxy})/\Delta y$

$YF(\lambda_z, \phi_{zx_2})_{zx_2y} = YF(\lambda_z, \phi_{zx_1})_{zx_1y^{\ell}(\phi_{zx_1})} + y$ for,

$y^{\ell}(\phi_{zx_2}) \leqslant y \leqslant y^u(\phi_{zx_1}) - y^{\ell}(\phi_{zx_1})$ and,

(R(1) and R(2)) for x_1 and x_2

$\phi_{zx_1} = \phi_{zx_2}$

7. Computational results

The algorithm including the symmetry aspects, has been used to develop software for a VAX/8600 computer. The code for the algorithm was written in FORTRAN 77. Furthermore a SAS-procedure was developed in order to display the patterns graphically (see fig. 7).

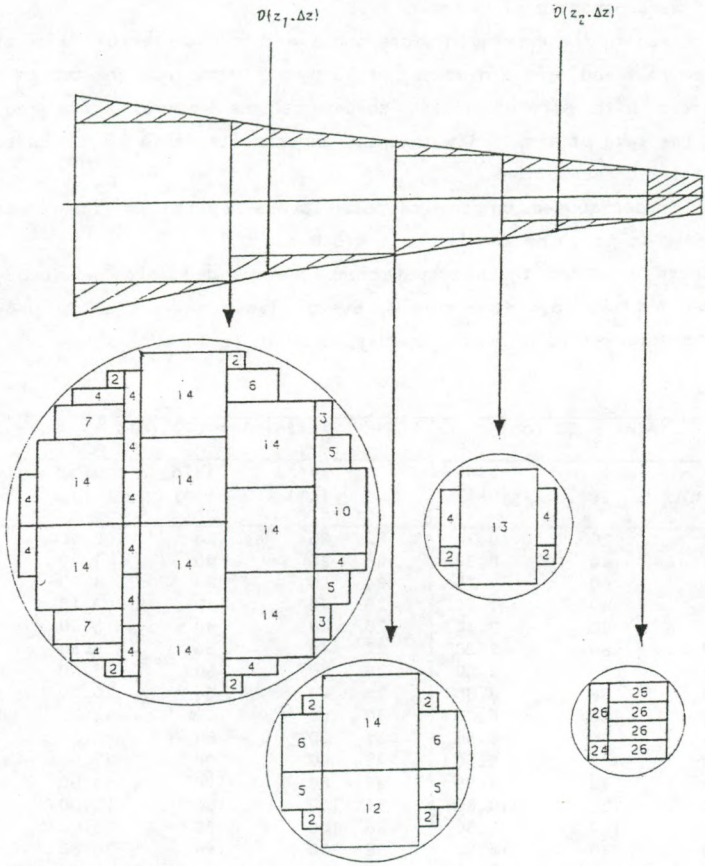

Fig. 7. An example of possible output of the algorithm.

Sawing patterns are influenced by many factors, such as size of log and products, product prices, saw kerf, log quality, size tolerances. The software did not include all these factors. Saw kerf, quality and dimension tolerances have not been taken into account. Main goal of this research was to analyse the possibilities for on-line optimization of sawing patterns, and thus "details" like sawkerf have not been taken into account.

Timber with a length from 200 up to 1400 cm has been evaluated with different software parameters.

The trees used in the experiments are cones with a diameter of 30 cm at 10 cm from the foot end, and a diameter of 10 cm at 10 cm from the top end. Of course trees with more realistic shapes can be optimally cut too, however, for the sake of simplicity in these experiments it is of no use to make more realistic assumptions.

The grid, superimposed on the circular cross-section of the trees varies from 5 mm up to 20 mm for its unit width.

The products used for these computations are divided into two length classes of 50 and 100 cm, respectively. Every class consists of 21 products, the characteristics of which are displayed in Table 1.

$\ell^z(n) = 50$ (cm)				$\ell^z(n) = 100$ (cm)			
n	$\ell^x(n)$ (cm)	$\ell^y(n)$ (cm)	v(n) (Dfl)	n	$\ell^x(n)$ (cm)	$\ell^y(n)$ (cm)	v(n) (Dfl)
1	20	20	0.30	22	20	20	0.45
2	20	40	0.60	23	20	40	1.10
3	20	60	1.15	24	20	60	3.10
4	40	40	1.55	25	40	40	3.10
5	40	60	2.40	26	40	60	5.00
6	40	80	3.20	27	40	80	6.50
7	60	60	3.60	28	60	60	7.50
8	60	80	4.80	29	60	80	10.00
9	60	100	6.20	30	60	100	12.70
10	80	80	6.50	31	80	80	13.50
11	80	100	8.50	32	80	100	17.00
12	80	120	9.80	33	80	120	18.00
13	100	100	10.80	34	100	100	20.00
14	100	120	12.50	35	100	120	25.00
15	100	140	14.50	36	100	140	30.00
16	120	120	15.50	37	120	120	32.00
17	120	140	17.50	38	120	140	35.00
18	120	160	20.50	39	120	160	42.00
19	140	140	21.00	40	140	140	45.00
20	140	160	24.00	41	140	160	50.00
21	160	160	27.00	42	160	160	55.00

Table 1. Characteristics for the products used in the computations.

The results of these experiments are displayed in Table 2.

Le (cm)	Δx (mm)	Δy (mm)	profit (Dfl.)	Cpu (sec.)	effectiveness (%)	efficiency (%)
400	5	5	207.65	22.53	100	1.9
400	10	10	207.60	2.88	100	15
400	20	20	199.90	0.43	96	100
800	5	5	484.35	47.31	100	1.7
800	10	10	478.05	5.95	99	13
800	20	20	458.15	0.80	95	100
1200	5	5	763.95	73.91	100	1.9
1200	10	10	759.90	9.89	99	14
1200	20	20	727.85	1.41	95	100
1400	5	5	908.35	88.71	100	2
1400	10	10	891.70	11.40	98	15
1400	20	20	854.70	1.76	94	100

Table 2. Effectiveness and efficiency as functions of tree length and grid width.

Effectiveness: = % largest value
Efficiency : = (% smallest cpu time)$^{-1}$

Although not developed for the purpose, the algorithm can also be used to cut rectangular plates of material, such as steel and plastics. We programmed the algorithm suggested by Gilmore & Gomory (1969) including the speed up suggestions made by Christofides & Witlock (1976).

The algorithm based on dynamical programming as suggested in the previous sections can only make parallel guillotine, based on Gilmore & Gomory. While the algorithm can make guillotine cuts in a general way, it thus gives upperlimits to effectiveness for our algorithm. We performed many computational tests in order to compare the algorithms, some of them are included in this text. Table 3 shows the products used are displayed and in Table 4 the computational results.

Product	length (mm)	width (mm)	value (Dfl)
1	10	40	0.60
2	20	20	0.40
3	20	50	1.60
4	20	110	3.50
5	30	30	1.10
6	30	50	2.50
7	40	20	0.90
8	40	30	1.50
9	40	40	2.45
10	40	50	3.20
11	40	130	8.00
12	50	50	3.25
13	60	50	4.40
14	70	20	2.05
15	80	40	5.10
16	80	80	10.75
17	90	50	2.80
18	110	60	9.50
19	120	130	12.00
20	160	110	15.00

Table 3. Product dimensions and values.

	GG				DPB			
Board (cm)	profit (Dfl)	cpu (s)	effec. (%)	effic. (%)	profit (Dfl)	cpu (s)	effec. (%)	effic. (%)
10 x 10	16,40	0,03	100	67	16,40	0,02	100	100
18 x 18	53,95	0,09	100	50	53,60	0,05	99	100
28 x 28	130,80	0,25	100	32	130,15	0,08	100	100
48 x 48	387,00	0,94	100	10	387,00	0,09	100	100
70 x 70	821,80	2,49	100	7	816,40	0,17	99	100
100 x 100	1676,60	6,39	100	3	1675,10	0,19	100	100

Table 4. Computational results.

Effectiveness: = % maximum value
Efficiency : = (% minumum cpu time)$^{-1}$
GG : = Gilmore and Gomary
DPB : = Algoritm based on Dynamical Programming

It is very hard to draw general conclusions from these results, because the shape of the initial rectangle and the product characteristics all influence the patterns in a specific way. In all tests, our algorithm beats the GG algorithm in efficiency, specially with large initial rectangles, but of course the GG algorithm was always more effective, on average about 1 % more. The fact that our algorithm reaches a higher efficiency than the GG algorithm was predictable because of the fact that it needs less computations.

8. Conclusions

The algorithm developed has great potential in optimizing the conversion of trees into lumber, when using a technique based on cross-cutting and sawing with a numerically controlled headrig-saw. The software developed is very fast and very effective. In fact, when using a programmable headrig saw, the resulting patterns are optimum. In our examples we converted whole trees. In practice often the conversionprocess is divided in a primary and secondary breakdown. Of course our algorithm will also function in this situation although a maxim recovery percentage is only attainable when integrating the two breakdown processes.

Both efficiency and effectiveness make this algorithm and the corresponding software useful for on-line application. If the need for efficiency predominates over the need for effectiveness, the algorithm can be used for the cutting-stock problem as defined for rectangular plates.

Literature

Christensen, E., Centralised woodprocessing, Paper presented at the 18th IUFRO World Congress in Ljubljana, Yugoslavia, 1986.

Christofides, N., Whitlock, C. An algorithm for two-dimensional cutting problems, Operations Research, vol. 25 (1977) pp. 30-44.

Faaland, B. and Briggs, D., Log bucking and lumbermanufacturing using dynamic programming, Management science, vol. 30 (1984) pp. 245-257.

Geerts, J.M.P., Mathematical solution for optimising the sawing pattern of a log given its dimensions and its defect core, New Zealand Journal of Forestry Science, vol. 14 (1984) pp. 124-134.

Gilmore, P.C., Gomory, R.E., The theory and computation of knapsack func-

tions, Operations Research, vol. 15 (1969) pp. 1044-1074.

Hallock, H., Stern, A.R., Lewis, D.W., A look at centered vs offset sawing, U.S. department of agriculture, forestscience, forestproducts laboratory, Madison, FPL 321 (1979).

Hallock, H., Stern, A.R., Lewis, D.W., Is there a best sawing method, U.S., department of agriculture, forestscience, forestproducts laboratory, Madison, FPL 321 (1979).

Ontvangen: 03-03-1988
Geaccepteerd: 25-07-1988