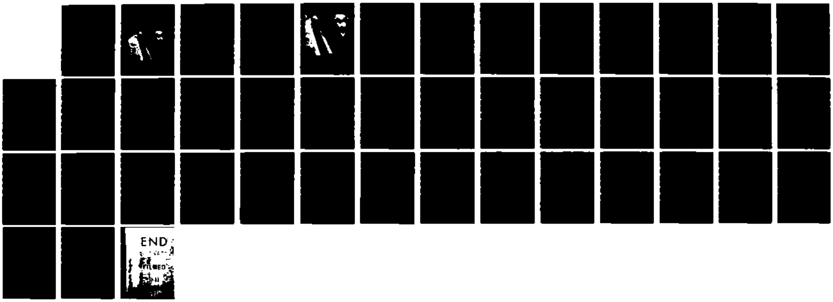
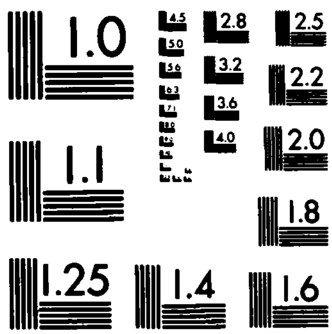
AD-A134 223 CROMAX: A CROSSCUT-FIRST COMPUTER SIMULATION PROGRAM TO DETERMINE CUTTING YIELD(U) FOREST PRODUCTS LAB MADISON WI P J GIESE ET AL. AUG 83 FSGTR-FPL-38 1/1

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A134 223

United States Department of Agriculture

Forest Service

Forest Products Laboratory

General Technical Report FPL-38

CROMAX
A Crosscut-First Computer Simulation Program to Determine Cutting Yield

Pamela J. Giese
and
Jeanne D. Danielson

Abstract

CROMAX simulates crosscut-first, then rip operations as commonly practiced in furniture manufacture. This program calculates cutting yields from individual boards based on board size and defect location. Such information can be useful in predicting yield from various grades and grade mixes thereby allowing for better management decisions in the rough mill.

The computer program CROMAX was written in ASCII FORTRAN on the University of Wisconsin's UNIVAC 1100/80 computer. The complete program listing is included as an appendix.

United States
Department of
Agriculture

Forest Service

Forest
Products
Laboratory[1]

General
Technical
Report
FPL-38

August 1983

CROMAX
A Crosscut-First Computer Simulation Program to Determine Cutting Yield

Pamela J. Giese, Computer Programmer
and
Jeanne D. Danielson, Forest Products Technologist

Introduction

Knowledge of the cutting yields attainable from a given lumber grade is vital to such basic rough mill decisions as ordering raw material and measuring mill performance. However, traditional methods of acquiring this information may be inadequate in light of present high production costs and low product demand. Mill studies are expensive, and valid only for the study day's conditions. Historical records may be biased by changes in within-grade lumber quality among suppliers, or over time, or by changes in cutting bills.

General cut-up models, such as CROMAX, can predict attainable cutting yields without upsetting mill production and can be run for a variety of cutting bills. Use of information from computer simulation models which determine cutting yields offers great benefits to mill operators. An example is the Rough Mill Improvement Program, developed by Huber and Harsh *(3,4,5)*,[2] which offers dimension plants a tool to determine the lowest cost mix of rough lumber grades for a given cutting bill.

Computer-derived cutting yields can also be used as a measure of mill performance, comparing actual mill yield to the highest theoretical yield. This gives the manager a standard which is not influenced by normal variations in production or raw material. To derive this highest theoretical yield, some sort of computer simulation is necessary. The computer program CROMAX was designed to simulate the crosscut-first operation in order to calculate, within the model's constraints, an optimal cutting yield from a given board. CROMAX calculates yield based upon the submitted cutting bill, the value of each size cutting, and the size and location of defects (e.g. knots, splits, checks, etc.) within the board.

[1] Maintained at Madison, Wis., in cooperation with the University of Wisconsin.

[2] Italicized numbers in parentheses refer to literature cited at end of report.

Determining Cutting Yield

Determining the cutting yield from a given board requires (1) accurate description of the unique characteristics of the board—board width, board length, and defect location (e.g. knots, splits), (2) awareness of mill requirements as presented by the cutting bill and, the most difficult to attain, (3) ability to make the best crosscut decision followed by an equally good rip decision.

At first glance, obtaining an accurate description of a board would seem a simple task; the board itself is available to the crosscut operator. What better description would one need? However, lighting, viewing position, and speed of the line may hinder the operator's ability to see the whole board and its defects. Technological improvements to automatically measure the board and locate defects and types of defects would be a great asset in making an accurate picture of the board available to an operator or a computer. In lieu of such technology, board descriptions as used in this study have been hand tallied. Without automatic defect detection equipment, current decision models have no immediate real time on-line possibilities. The hand recording of board data (dimension and defect information) has been used as a method of acquiring this information since the early 1960's *(1,7,8)*. The method used for recording this board information was described by Lucas *(6)* in 1973. Each board is depicted as a rectangle with an X-Y grid superimposed over it. (The grid origin is at the lower left corner of the board.) Defect locations are read from the grid and tallied (fig. 1).

Figure 1.—Boards being tallied for defects. *(122 719)*

To obtain the best cutting yield from a given board, the operator or computer must be supplied information on the quantity of each dimension cutting required to meet mill demand. Thus each cutting takes on a relative value—cuttings which are easy to come by, such as narrow, short cuttings, take on a lesser value, while those which are more difficult to recover, such as wide, long cuttings, have a higher value. It is very important that the computer model be able to incorporate information on the relative value of each cutting dimension to determine the best available cutting yield of a given grade. Therefore, models which only look at the surface area of cuttings as a measure of yield neglect the real possibility that higher valued cuttings are being sacrificed to attain greater surface area.

Once the board data and value of the cuttings have been supplied, the board must be crosscut, then ripped, in such a way as to get the highest total value of cuttings from the board. The decision of where to crosscut is the most difficult decision since the crosscuts could be placed anywhere within the board, limited only by the cutting lengths. In contrast, the location of rip lines is dictated by the crosscut boundaries, the cutting widths required, the location of defects, and the width of the board. Once the crosscut decision for cutting one piece has been made, the yield of the rest of the board is affected. A bad decision may sacrifice overall yield from the board to recover one or two good cuttings.

Program Background

The CROMAX program is a further development in the Forest Products Laboratory's ongoing research program for developing computer models to improve yield in secondary wood processing.

The first of these models was the YIELD program developed in 1966 by Wodzinski and Hahm *(9)*, which has been used in several cutting yield studies *(1,7,8)*. While a great improvement over manual efforts to calculate optimal cutting yields, the program suffered several limitations which prevented it from realistically modeling existing cut-up operations and made it obsolete by today's standards.

The high cost of computer usage at the time necessitated the use of shortcuts which minimized computer time, but which at the same time led to finding less than optimal yields. YIELD searches for the largest clear area between defects and places the longest, widest cutting possible in it. This area is blocked out, and the next largest clear area found and filled, and so on. Given a choice of two cuttings with equal surface areas, the program is biased to the longer cutting. This frequently leads to a situation where the program chooses a long cutting and a very short one over two of medium-length, which in total may be more valuable to the plant.

A mixture of crosscut-first and rip-first operations on different boards results by placing the cuttings in the clear area, then fitting the kerfs around the cuttings. Since most plants are set up for one or the other, either rip-first or crosscut-first, the YIELD program did not accurately model either operation, although it was biased toward the crosscut first.

Efforts to more realistically model the industry led to the development of the OPTYLD program *(2)*, which modeled rip-first operations. The CROMAX program was developed from OPTYLD as the need for a crosscut-first model was recognized.

The Model CROMAX

The CROMAX computer program is the first step in the development of computer models of crosscut-first operations which will be suitable for planning and decisionmaking. CROMAX processes an unlimited number of boards, one board at a time. It retains no memory of previous boards or their solutions. The program represents a board as a rectangle superimposed on a Cartesian coordinate system with the lower left corner at the origin. The description of the board is stored in a binary matrix with each cell of the matrix set to either 1 to represent a defect or 0 (zero) no defect. A sample board is shown in figure 2. Before starting the crosscutting process, the ends of the board are trimmed off. The amount trimmed off each board is specified at run-time and is constant for all boards in the run. CROMAX requires specification of all allowable lengths and widths of cuttings. Cutting yields are generated by repeatedly going through all possible combinations of cutting lengths that will fit within the board.

After the ends of the board are trimmed off, the process of generating cutting-yield solutions is begun. The first solution begins at the left end of the board. Crosscuts are placed such that the distance between two crosscuts is equal to the shortest allowable cutting length. Such an area, where the distance between two crosscuts meets or exceeds the shortest allowable cutting length, will be referred to as a section. Each section is ripped to yield the highest value of cuttings. Figure 3 shows this first combination. The value of the cuttings is summed and stored as total cutting value. No defects are allowed within a cutting.

The next series of cutting yields is obtained by maintaining the same section lengths but varying the location of the beginning of the sections. Defects may lie within some of the sections. Defect coordinates of the board are shown in table 1. If a defect ends within the section, an alternative solution is generated by moving the beginning of the section to the end of the defect. Figures 4 through 8 show the first five alternative solutions to the first crosscutting solution. Positions of crosscut lines are moved first at the right end of the board and gradually to the left. Figure 4 shows the first alternative to figure 3. The beginning crosscut of the 11th cutting length section in figure 3 is relocated to the end of the defect which ends at the X coordinate 428 in figure 4. The next alternative (fig. 5) moves the crosscut to the end of the defect ending at the X coordinate 438. For each of these alternatives no other cutting length section to the left is affected. Since crosscuts had been made at X coordinate 416 in the original crosscutting solution, the alternative involving this defect has already been calculated. The next defect ends at X coordinate 353, so a crosscut is placed at this location for alternative 3 (fig. 6). The two sections to the right of 394 must then be moved; this results in the loss of three cuttings from the two previous solutions. Alternative 4 moves the crosscut to 339 (fig. 7). While this alternative picks up another cutting over the previous solution, the cuttings are narrower, plus no cuttings can be made from the area of 380-420. Alternative 5 places a crosscut at 318 (fig. 8). This results in the same number of cuttings as in the previous alternative, but some of the cuttings made here are wider.

Once the location of a section is moved, all section locations to the right must also be moved to accommodate this change. The sections in the new location are then ripped again and the value of cuttings obtained is summed. Their total is compared with the previous high total cutting value. If the new total is higher than the previous high total, the new total replaces the old. All alternative locations of cutting sections are tried and their values compared with the old high value. After the alternatives to the cutting length solution combination have been tried, the next cutting length solution is tried, then its alternates. In this way, all cutting length combinations and alternates are tried.

After all solutions have been tried, the best solution is printed and the next board is read. The best solution for the sample board is shown in figure 9 and table 2.

Program Description

Computer program CROMAX is divided into 11 modules— the main program, 9 subroutines, and 1 function. A flowchart illustrating the basic structure of the program is shown in figure 10. Table 3 lists these modules and their respective entry points. The complete CROMAX program is presented in appendix A.

Main Program

The main program (MAIN) serves as the input/output center of the program as well as coordinating the processing of the board. Figure 11 describes MAIN. When the program is begun, the run-time options are read. These options control the decisionmaking capabilities of CROMAX throughout the run. The trimming options specify the amount to be trimmed off each end of the board. All allowable cutting lengths and cutting widths must be specified. Table 4 lists these decisionmaking run-time options.

Supplying a table of weighted values for cuttings of different dimensions is optional. If a table is not supplied, the total yield of a crosscutting decision is obtained by summing the surface area of the cuttings available. If a table is used, the total yield of a crosscutting decision is obtained by summing the value (surface area times weight factor) of the cuttings available. The use and derivation of the weighted value table (table 5) is discussed in appendix B.

CROMAX builds a table of the best rip width combinations for a given clear area. This table is built upon and used by all boards within the sample. After the run-time, decisionmaking options are read, WINTL (an entry of WFIND) is called to initialize the possible best rip width combinations for a given clear area.

CROMAX then reads the board information and translates the board into a packed binary matrix where each bit corresponds to the 1/4-inch coordinate grid on the board. A value of 1 is assigned to each grid within a defect while a 0 is assigned to each grid within a clear area. The board is rejected if its length or width exceed the allowable board dimensions. The maximum number of cutting length sections within the board is then found. Yield and cutting length section combinations are then initialized and the first cutting length section combination is generated.

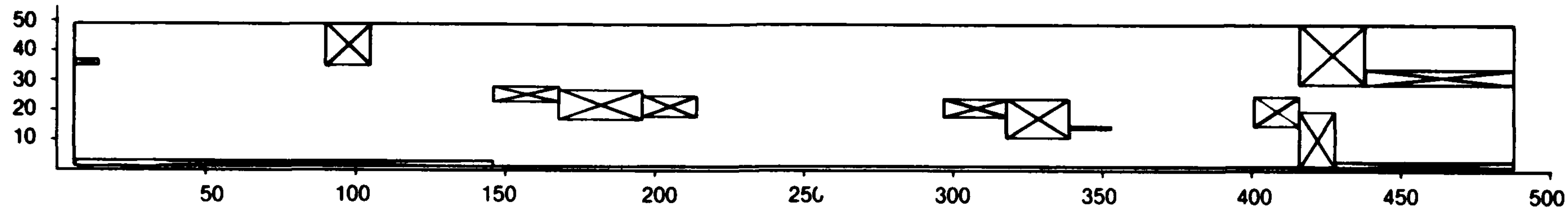


Figure 2.—Sample board. Crossed out boxes represent defects (areas filled with 1's in binary matrix) and remaining areas are clear (filled with O's in matrix). All values are in 1/4-inch units. (ML83 5126)

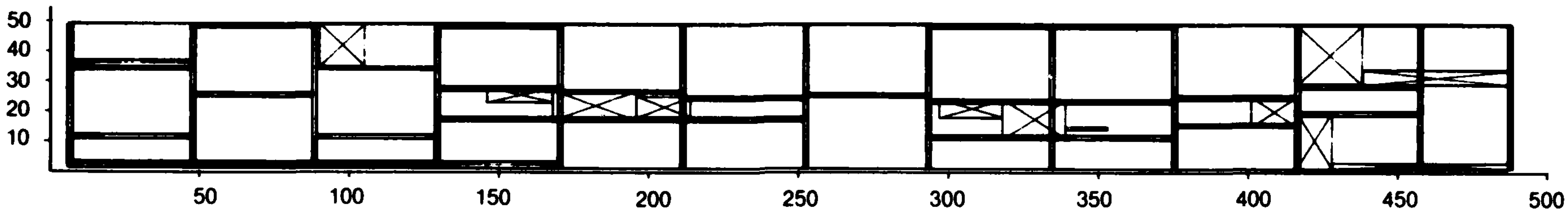


Figure 3.—Board shown in figure 2 after first crosscuts and rips are placed. (ML83 5127)

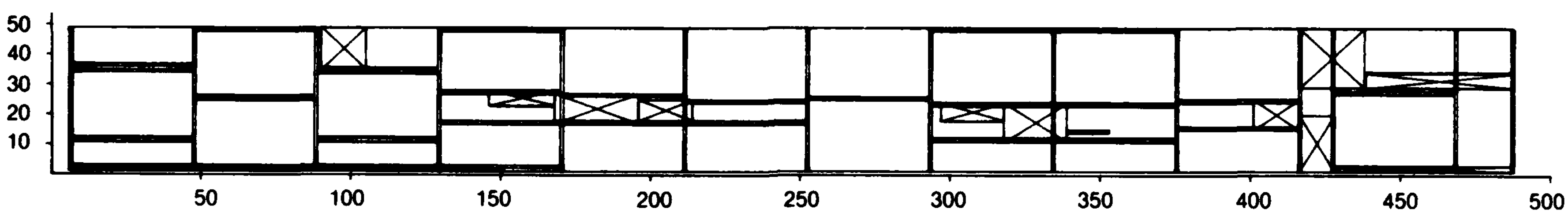


Figure 4.—First alternative to cutting length combination 1 (fig. 3). Note changes in last cutting length section. (ML83 5128)

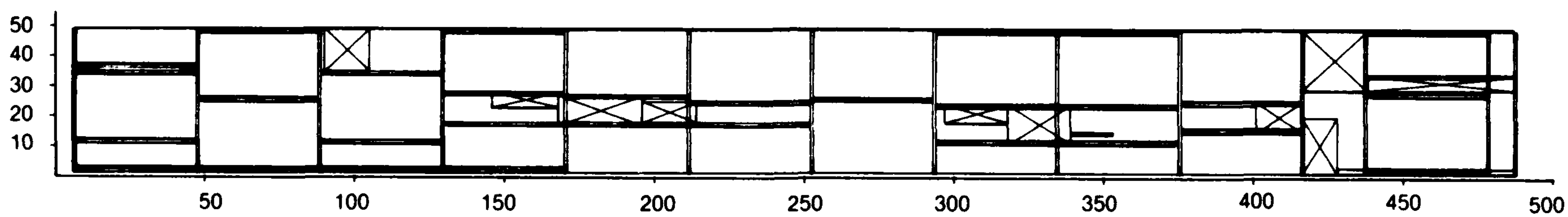


Figure 5.—Second alternative to cutting length combination 1 (fig. 3). (ML83 5129)

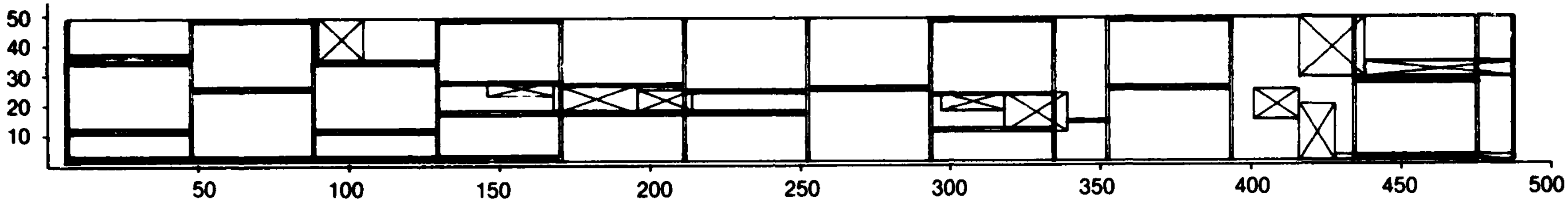


Figure 6.—Third alternative to cutting length combination 1 (fig. 3). Note changes involving last three sections. (ML83 5130)

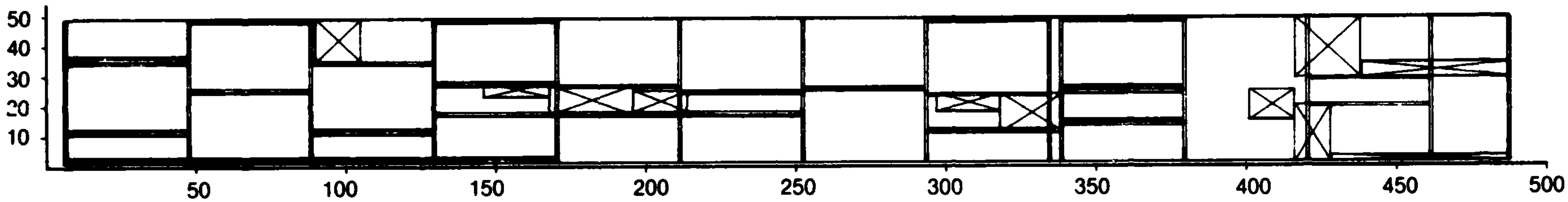


Figure 7.—Fourth alternative to cutting length combination 1. Note changes involving last three sections. (ML83 5131)

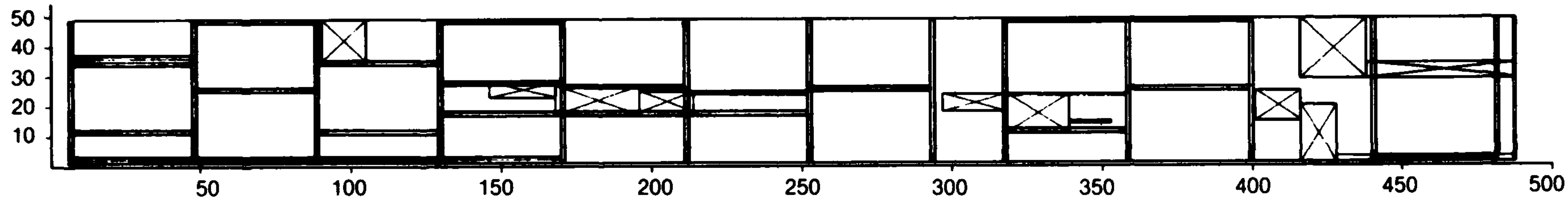


Figure 8.—Fifth alternative to cutting length section combination 1 (fig. 3). Note changes involving last four sections. (ML83 5132)

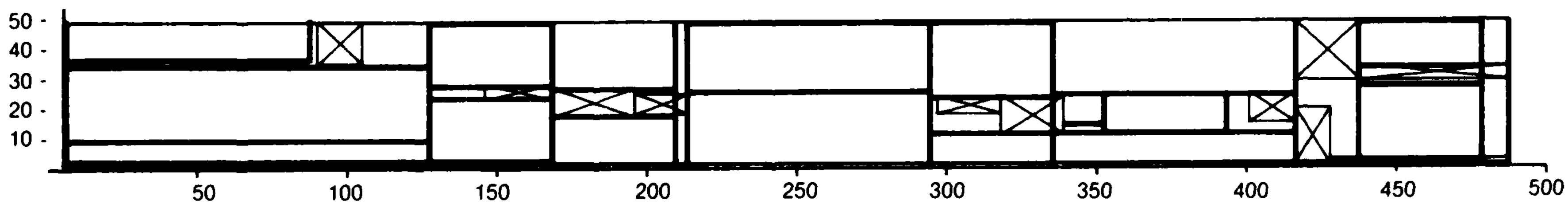


Figure 9.—Best cutting solution for sample board shown in figure 2. (ML83 5133)

Each cutting length section is checked to see if its yield has been calculated before. This is done by calling HOLD. If it has been calculated, its yield is retrieved. The section is also checked by ALTER to see how many defects end within its bounds. These defect coordinates form the alternatives to the cutting length combination which will be attempted; for each defect ending within the section, the beginning of the section is moved to the end of the defect. Subsequent sections are positioned accordingly. CROMAX then calls SAW to cut up all sections that have not yet been calculated. The yields attained from the sections are tallied and totaled, and compared with the previous maximum yield. If the current solution is higher, it and the present combination of cutting lengths are reassigned to be the maximum yield combination. The next alternative position for the cutting length combination is then generated and processed as above. This is repeated for all alternative positions for the cutting length combination. After all alternative positions have been tried, the next cutting length combination is generated and the above cycle is repeated. The coordinates of the cuttings and sawkerfs are not stored, so after all combinations have been calculated, the combination giving the highest yield is rerun and its result printed. The next board is then read. The program stops after all boards have been read and processed.

Subroutine SAW

Subroutine SAW is described by the flowchart in figure 12. Subroutine SAW scans for clear areas within a given cutting length section. When first entered, SAW initializes the yield of the section to zero. If the length of the section exceeds the smallest possible cutting length (this could only occur after the first combination), RANGE is called to set the boundaries of any salvage cuttings. SAW scans the section first by length and then by width in search of defect areas. If a defect is found, the scanning process is stopped and any clear area tested to see if it meets the minimal width. If it does, RIP is called to rip the section. If the whole cutting length section is found to be free of defects, RIP is called to rip the section into cuttings. The whole section is processed in this way; then, if areas remain which have not been utilized, TRIMIT (an entry of RANGE) is called to locate and salvage cuttings. SAW then returns to MAIN.

Subroutine RANGE

Subroutine RANGE contains three routines involved in the salvage cutting process—RANGE, TRIMIT, and STORE. RANGE itself simply initializes to zero the number of actual cuttings found. Entry STORE stores the number of actual cuttings found by RIP and CUTUP. The major routine in RANGE (fig. 13) is TRIMIT, which finds the combination of salvage cuttings giving the highest yield.

Table 1.—Board data for sample board No. 130 (fig. 2)

Grade 2C		Number of defects = 14	
Coordinates			
Lower Y	**Lower X**	**Upper Y**	**Upper X**
BOARD			
1	6	49	488
DEFECTS			
1	6	3	146
35	6	37	14
35	90	49	105
23	146	28	168
17	168	27	196
18	196	25	214
11	318	24	339
14	339	15	353
15	401	25	416
1	416	20	428
29	416	49	438
1	428	3	488
29	438	34	488

Note: All values are in 1/4-inch units.

Table 2.—Best cutting solution for sample board (fig. 2)

2C BOARD NUMBER 130

Cuttings

30.00 × 1.50
30.00 × 6.00
20.00 × 3.00
10.00 × 5.00
10.00 × 5.00
10.00 × 4.00
10.00 × 5.50
20.00 × 6.00
20.00 × 5.50
10.00 × 2.50
10.00 × 6.00
20.00 × 2.50
20.00 × 6.00
10.00 × 3.00
10.00 × 6.00
10.00 × 3.50

Total surface area of board	1,446.00 In.²
Total percentage yield	75.38
Total area of cuttings	1,090.00 In.²

Run options used:

Trim	0.25 In.
Cutting widths	1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0 In.
Cutting lengths	10.00, 20.00, 30.00, 40.00 In.

Weighting based on surface area only

Table 3.—Subprograms and entries of CROMAX

Subprogram name	Subprogram type	Additional entries
MAIN	Main program	—
SAW	Subroutine	—
RANGE	Subroutine	TRIMIT, STORE
RIP	Subroutine	—
AMEND	Subroutine	—
ALTER	Subroutine	REVISE
CUTUP	Subroutine	—
HOLD	Subroutine	INTL, REMEM
WFIND	Subroutine	WINTL
TSTORE	Subroutine	TINTL, RETREV
VALUE	Function	—

Table 4.—Run-time options

Option name	Option action	Card format
Trimming*	Any nonnegative integer	5X, I2
Maximization	Any nonnegative integer where if equal to: 0 means maximize on surface area not 0 means maximize on value	6X, I1
VALUE TABLE (present only if value maximized)		
Number of lengths and widths	Length—positive integer ≤ 8 Width—positive integer ≤ 4	2(5X, I2)
Widths**	Nonnegative integers in increasing order	4I5
Lengths**	Nonnegative integers in increasing order	8I5
Weighted values (4 cards)	Real numbers	8F5.2
Number of cutting lengths and cutting widths	Nonnegative integers ≤ 10	2(5X, I2)
Cutting lengths*	Integers in increasing order	10I5
Cutting widths*	Integers in increasing order	10I5

* Values are in 1/4-inch units.
** Values are in inches.

On entry to TRIMIT, the areas defining potential salvageable areas are found. A potential salvageable area is defined as the area between cuttings already obtained or between a cutting and the edge of the board. These areas are tested to see if they meet minimum width criteria for a cutting. If the area fails this test, it is ignored. All the potential salvageable pieces are checked to eliminate duplicates. TRIMIT then attempts to cut up the salvageable area. For each salvage area, TRIMIT attempts to cut it up first by cutting the length back and then by ripping the piece narrower. The solution of each of these processes is saved by calling TSTORE. After all possible salvage cuttings have been found, RETREV (an entry of TSTORE) is called to retrieve the yield of each cutting. The best (highest yielding) combination of cuttings is chosen.

Subroutine RIP

Subroutine RIP (fig. 14) rips the clear area found in SAW. Upon entry, RIP calls WFIND to find the best combination of cutting widths in that area. For each width RIP calls CUTUP to saw the cuttings. If the area is salvageable (that is, its length exceeds the minimum cutting length), RIP calls STORE (an entry of RANGE) to store the coordinates of the cutting.

Subroutine AMEND

Because only yield per section, not the coordinates of the cuttings within the section, is stored, it is necessary to rerun the maximum combination to determine cutting and sawkerf coordinates. This is the purpose of AMEND (fig. 15). AMEND is called from MAIN after all combinations have been tried and the maximum yield has been found. AMEND takes each cutting length section, defines its bounds, and calls SAW to cut up the section. The coordinates and dimensions of the cuttings and saw of the cut lines are then available to be included in the program output.

Subroutine ALTER

Subroutine ALTER (fig. 16) has two entry points—ALTER and REVISE. The purpose of ALTER is to find any possible alternatives within the cutting length combination. Alternatives consist of changing the beginning of the cutting length section so that the section begins at the end of a defect lying within the original section.

ALTER looks at the given bounds of the cutting length section and tests each defect to see if its end lies within the section's bounds. If such a defect is found, ALTER checks to see if that alternative has already been found. If it has not, the upper X coordinate of the defect is stored. The next defect is then tried. After all defects have been checked, ALTER next returns to MAIN.

Entry REVISE retrieves the X coordinate for a given alternate combination.

Subroutine CUTUP

Using the coordinates sent to it, subroutine CUTUP (fig. 17) defines the cutting and adds the value of the cutting to the section yield total.

Table 5.—Value weighting table. Both lengths and widths are upper bounds of the ranges

Width	Length							
	18.0	23.0	35.0	42.0	59.0	71.0	83.0	95.0
In.				In.				
1.75	0.790	0.851	0.876	0.897	0.936	1.005	1.085	1.105
2.75	.790	.851	.887	.909	.964	1.038	1.083	1.189
3.75	.790	.851	.887	.921	.988	1.055	1.123	1.235
4.75	.817	.875	.897	.933	1.010	1.079	1.235	1.400

Subroutine HOLD

Subroutine HOLD (fig. 18) has three entry points—HOLD, INTL, and REMEM. The purpose of the subroutine is to store the list of coordinates of the cutting length sections tried, and their corresponding yields. The purpose of entry HOLD is to check whether or not a given section has been calculated before. If it has, the yield for that section is retrieved.

Entry INTL simply initializes the number of sections calculated to zero. Entry REMEM stores the yield of a given cutting length section.

Subroutine WFIND

Subroutine WFIND (fig. 19) has two entry points—WFIND and WINTL. WFIND builds the table of best rip width combinations per clear area. This table is used by all boards within the run. When first entered, WFIND checks to see if the best rip width combination for the given clear area has been calculated yet. If it has, the width combination is retrieved and WFIND returns. If the width combination has not been calculated before, it must be solved. WFIND generates the first width combination by ripping the entire clear area with the smallest width of cutting, taking as many rips as will fit in the area. The value of these cuttings is summed and stored. The next combination of cutting widths is then generated. The total value of the cuttings produced by this combination is then compared with the previous high value. If the current value is higher than the previous high, it becomes the new high value. This process of generating width combinations and testing the sum of the values of these cutting(s) is repeated until all width combinations have been generated. The final high yield and high combination are then stored with the clear area in the table of best width combinations. The rip width combination is then returned as a parameter of WFIND.

Entry WINTL initializes the number of clear areas tested to zero.

Subroutine TSTORE and Function VALUE

Subroutine TSTORE (fig. 20) has three entry points—TSTORE, TINTL, and RETREV. TSTORE is a storage location for possible salvage cuttings produced by TRIMIT. Entry TINTL initializes the number of salvage cuttings to zero. Entry TSTORE checks if the salvage cutting is already stored; if it is, TSTORE returns. If not, TSTORE stores the coordinates of the cutting. The value of the cutting is then added to the total value for the cutting process (additional crosscut or rip) from which the cutting was derived. TSTORE then returns. Entry RETREV decides which salvage process (additional crosscut or rip) produces the highest value of cuttings. RETREV then calls CUTUP to saw each of these cuttings and returns. The value of a cutting is determined by referencing the function VALUE (length, width). VALUE (fig. 21) computes the value of a cutting based upon the surface area of the cutting and the weighting factor derived from the value index table.

Program Input

Input to run CROMAX consists of two types: (1) option cards, and (2) board data cards. The option cards list the decisionmaking options to be used while the board data cards describe the individual boards. Table 6 shows the input used to run CROMAX for the board in figures 2 to 9.

Options

Options available in CROMAX allow the user to alter the decisionmaking capabilities of the program. Table 4 lists the options and their respective formats. Briefly:

1. Trimming—The amount of wood trimmed off each end of the board is defined as trimming. CROMAX reads this value in quarter-inch units and trims each board back this amount; no decisions are made as to whether or not a particular board should be trimmed or if more or less wood should be taken off. The amount off is the same for all boards.

2. Maximization—Value of cutting vs. surface area of cutting. CROMAX has the capability of maximizing the yield decision based upon either the sum of the value of the cuttings, or the sum of the surface area of the cuttings. The latter simply maximizes surface area of cuttings alone. The value maximization determines the best cutting solution based upon the surface area of the cuttings and the weighted value. If the total value of cuttings is to be maximized, a value index table must be supplied. Cards are required for (a) the number of lengths and widths to define the table size, (b) the cutting widths to define the row dimension of the table, (c) the cutting lengths to define the columns of the table, and (d) four value cards, one for each width. Entries on this card represent the value index of the corresponding length position for that width. The value index table allows the user great freedom in selecting key cutting dimensions.

Table 6.—Input used to run sample board (fig. 2). All coordinates are listed: Lower Y-Lower X; Upper Y-Upper X. All values are in 1/4-inch units.

Option Cards		TRIM = 1	
		VALUE = 0	
		NLEN = 4NWID = 10	
		40 80 120 160	
		6 8 10 12 14 16 18 20 22 24	
Board Data Cards		Grade 2C Board Number 130 Total Number of Defects 14	
	Board Coordinates	1- 6	49-488
	Defect Coordinates	1- 6	3-146
		35- 6	37- 14
		35- 90	49-105
		23-146	28-168
		17-168	27-196
		18-196	25-214
		18-297	24-318
		11-318	24-339
		14-339	15-353
		15-401	25-416
		1-416	20-428
		29-416	49-438
		1-428	3-488
		29-438	34-488

3. Number of cutting lengths and widths—The number of cutting lengths and the number of cutting widths must be specified.

4. Cutting lengths—The cutting lengths allowed (up to 10) are specified on this card.

5. Cutting widths—The cutting widths allowed (up to 10) are specified on this card.

Board Data

Boards are described as rectangles superimposed on an X-Y grid, with the X direction along the length of the board and Y across its width. Defects are represented as rectangles within the board. Since a rectangle can be defined by two points, only the lower left coordinate and the upper right coordinate of the board or defect are specified. The order of the coordinates is lower Y − lower X, then upper Y − upper X. The input for the board in figure 2 is given in table 1.

The input for each board consists of three record types: (1) a header card defining the lumber grade, the board number, and the number of defects within the board, (2) a board coordinate card defining the coordinates of the board dimensions (lower left and upper right coordinates), and (3) a defect coordinate card for each defect within the board (up to the number specified on the header card) defining the coordinates of the defect (lower left and upper right coordinates). Data are arranged board after board; the sequence for input goes option cards, board 1, board 2, . . ., board n . . . until the end of file.

Discussion

As automatic defect detection and use of computer controls within furniture and other rough mills increase, computer decisionmaking and modeling of these processes will become more and more important. It is hoped this paper will encourage others to investigate models for crosscut-first lumber processing.

The model and program CROMAX are the first generation of a computer program to simulate crosscut-first operations. The major objective was to develop the basic algorithms to maximize cutting yield; however, to do this CROMAX processes a very large number of different combinations of section lengths. The computing time involved in the process is prohibitive (frequently 5 minutes or more per 8-foot board when run on a UNIVAC 1100/80); consequently yield studies such as performed by Schumann *(7,8)* are not economically feasible. The authors are currently investigating algorithms which will decrease the number of combinations without sacrificing accuracy. Heuristics, which will allow CROMAX "to know" if a cutting decision is "good" or "bad" show the most promise.

Literature Cited

1. **Englerth, George H.; Schumann, David R.** Charts for calculating dimension yields from hard maple lumber. USDA For. Serv. Res. Pap. FPL 118. For. Prod. Lab., Madison, Wis.; 1969.

2. **Giese, Pamela J.; McDonald, Kent A.** OPTYLD—A multiple rip-first computer program to maximize cutting yields. USDA For. Serv. Res. Pap. FPL 412. For. Prod. Lab., Madison, Wis.; 1982.

3. **Huber, Henry A.; Harsh, Stephen B.; Pepke, Edward K.** Improving lumber yields in the rough mill. Wood and Wood Products; April 1978.

4. **Huber, Henry A.; Harsh, Stephen B.** Rough-mill improvement program. Woodworking and Furniture Digest; February 1977.

5. **Huber, Henry A.; Harsh, Stephen B.** In the rough mill, should you rip or crosscut first? Woodworking and Furniture Digest; June 1974.

6. **Lucas, Edwin L.; Catron, Leathern R. R.** A comprehensive defect data bank for No. 2 Common oak lumber. USDA For. Serv. Res. Pap. NE-262. Northeastern For. Exp. Stn., Upper Darby, Pa.; 1973.

7. **Schumann, David R.** Dimension yields from alder lumber. USDA For. Serv. Res. Pap. FPL 170. For. Prod. Lab., Madison, Wis.; 1972.

8. **Schumann, David R.; Englerth, George H.** Yields of random-width dimension from 4/4 hard maple lumber. USDA For. Serv. Res. Pap. FPL 81. For. Prod. Lab., Madison, Wis.; 1967.

9. **Wodzinski, Claudia; Hahm, Eldona.** A computer program to determine yields of lumber. USDA For. Serv. FPL unnumbered publ. For. Prod. Lab., Madison, Wis.; 1966.

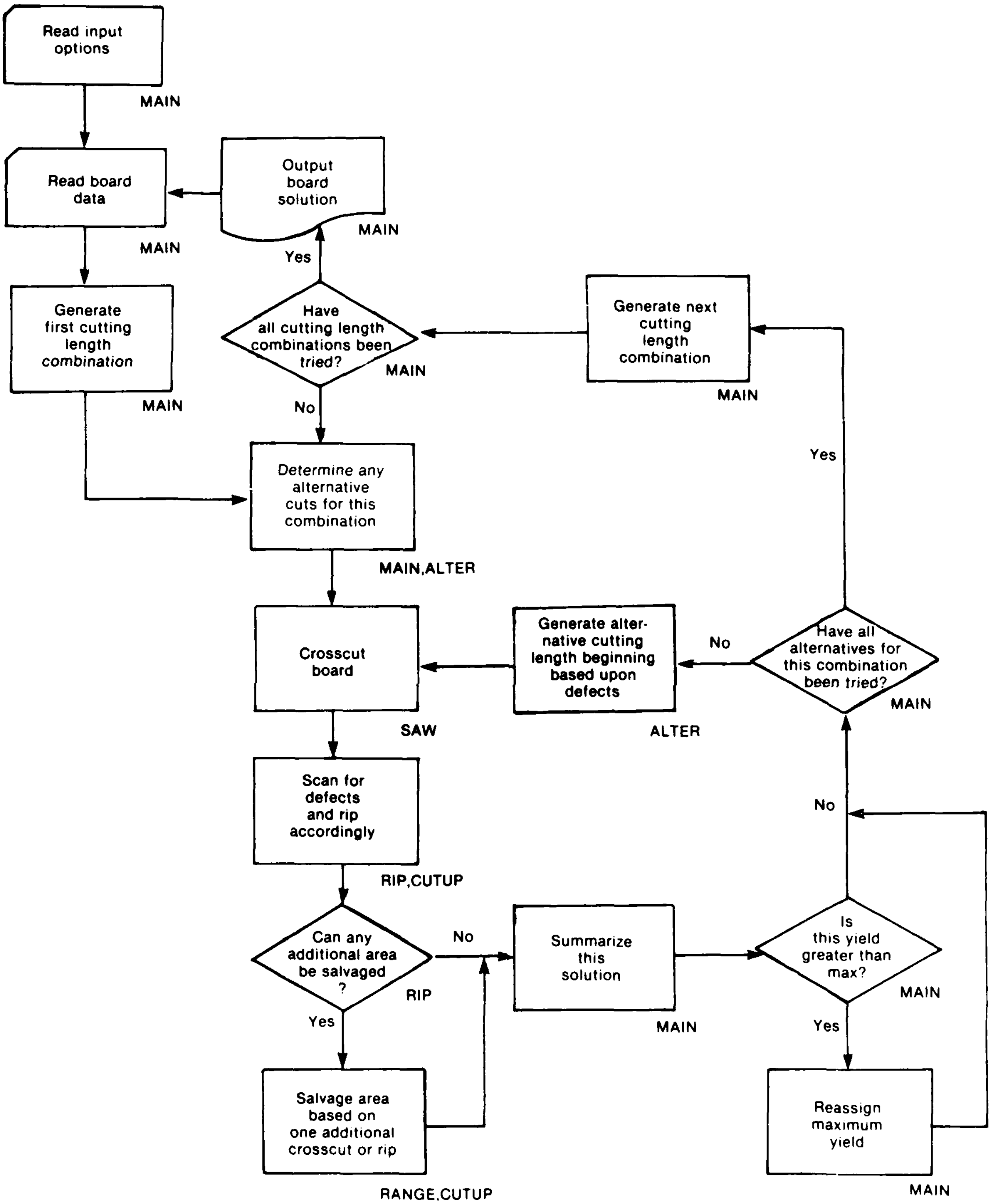


Figure 10.—General flowchart of computer program CROMAX. (ML83 5049)

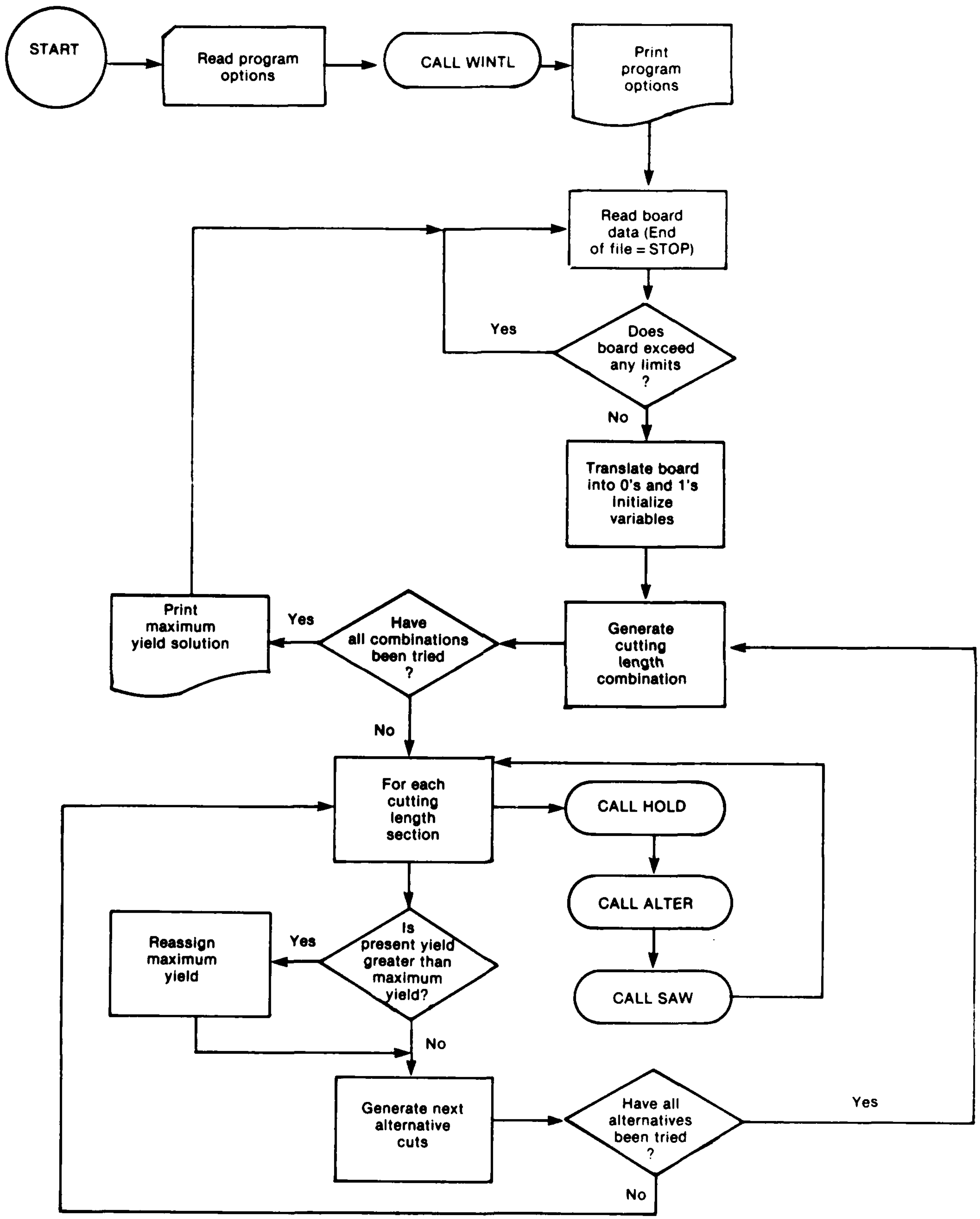


Figure 11.—Flowchart of main program of computer program CROMAX. (ML83 5043)

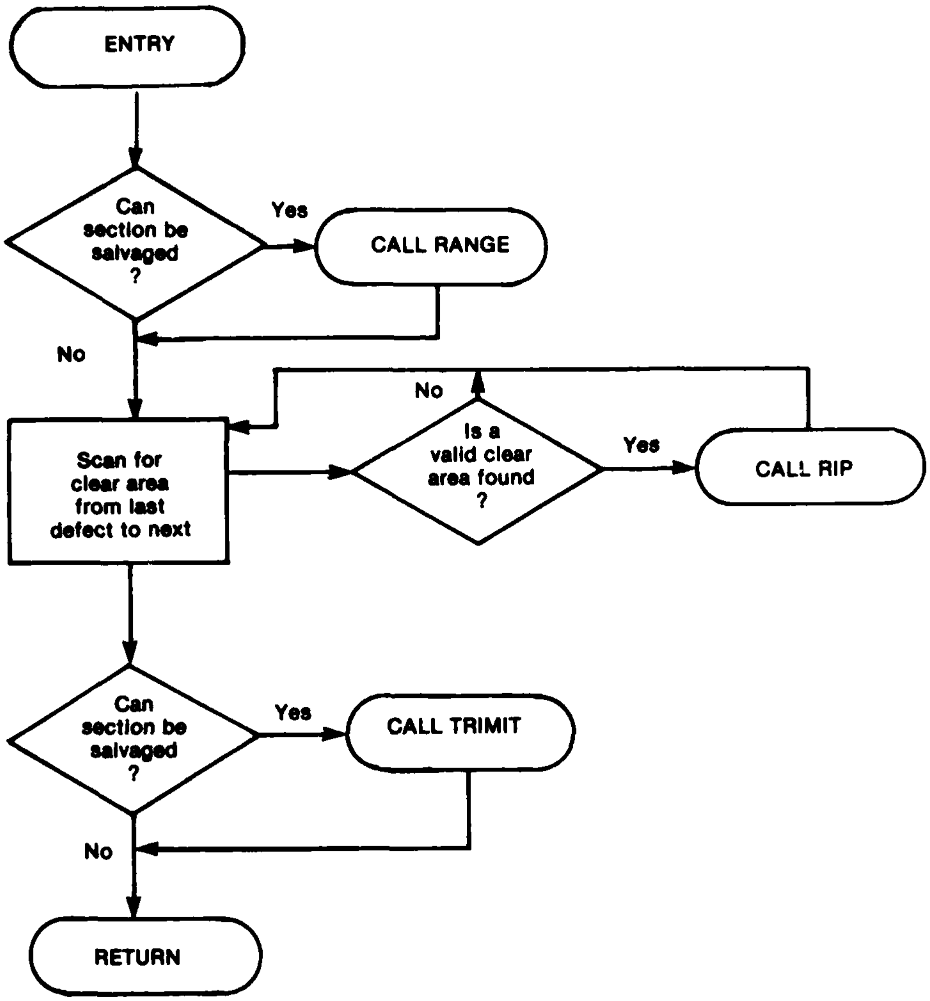


Figure 12.—Flowchart of subroutine SAW. (ML83 5042)

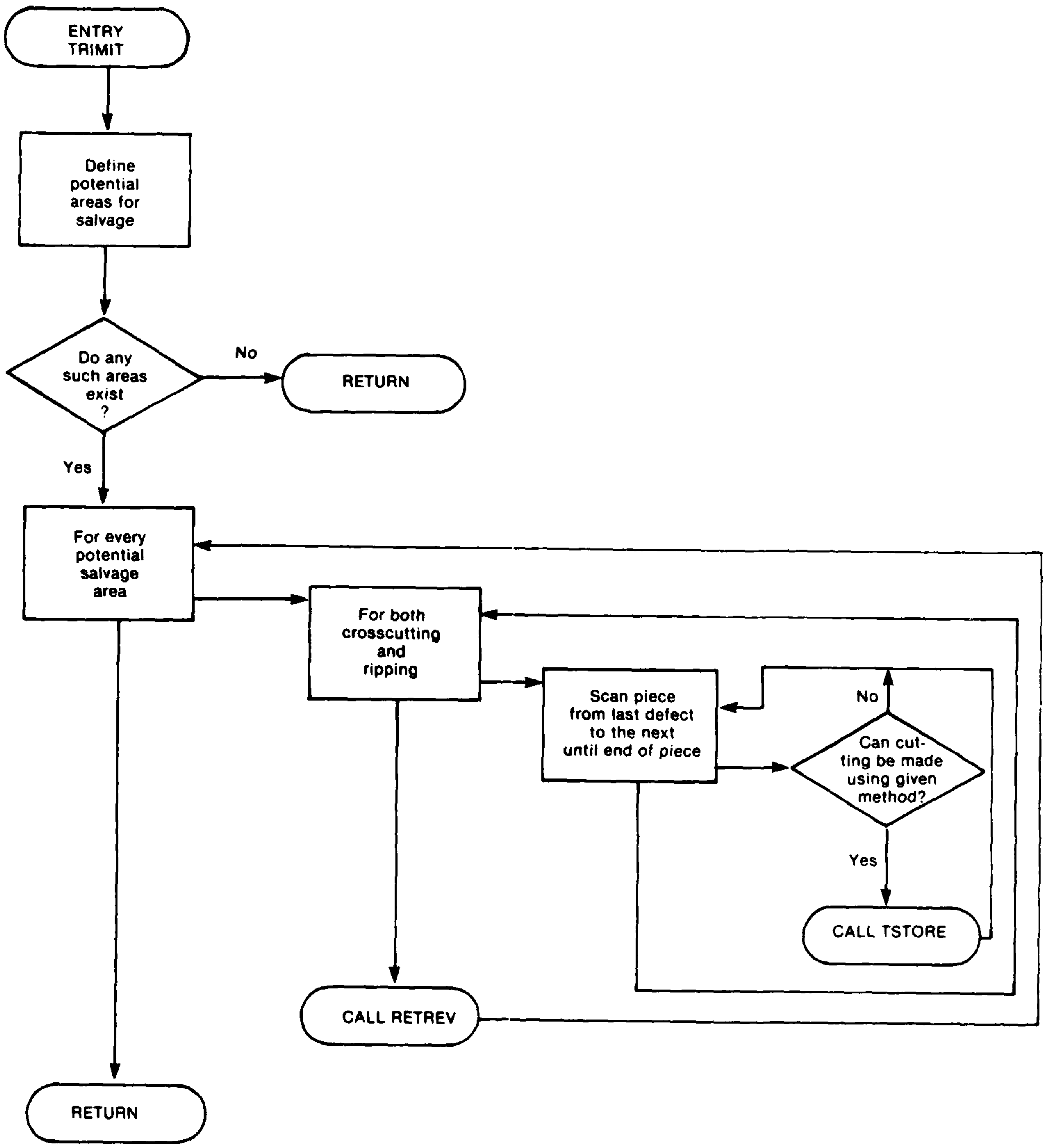


Figure 13.—Flowchart of subroutine RANGE. Entry points are RANGE, TRIMIT, and STORE. (ML83 5044)

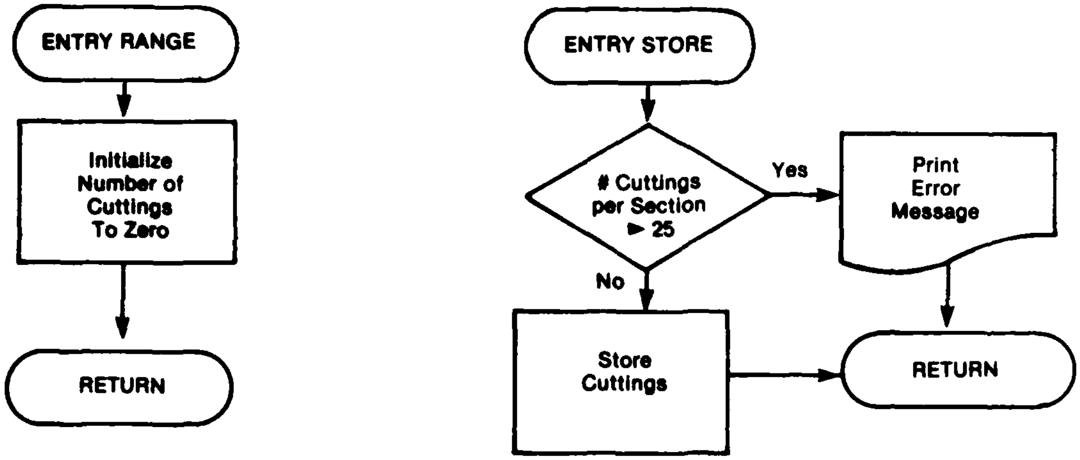


Figure 13.—Flowchart of subroutine RANGE. Entry points are RANGE, TRIMIT and STORE. (Continued) (ML83 5044)

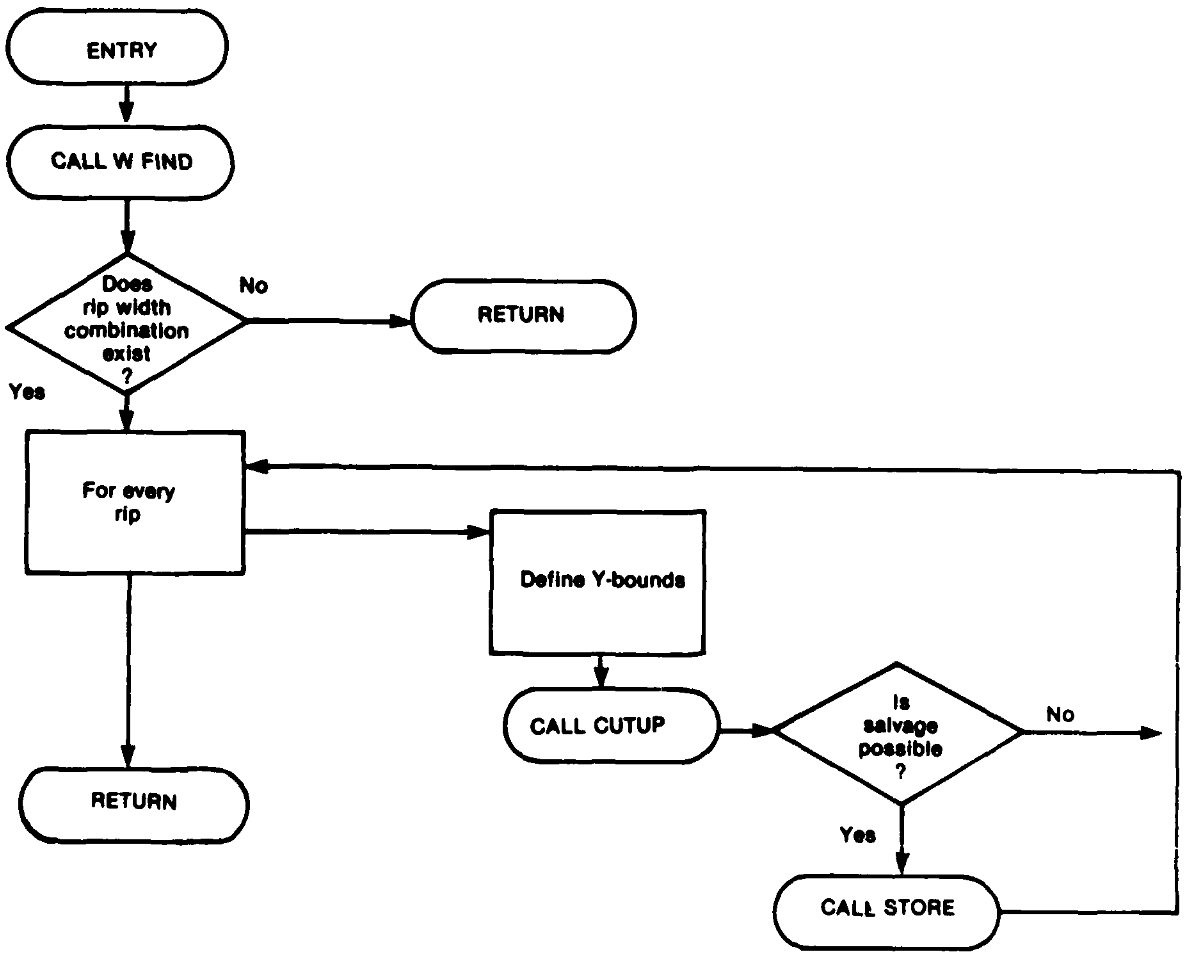


Figure 14.—Flowchart of subroutine RIP. (ML83 5041)

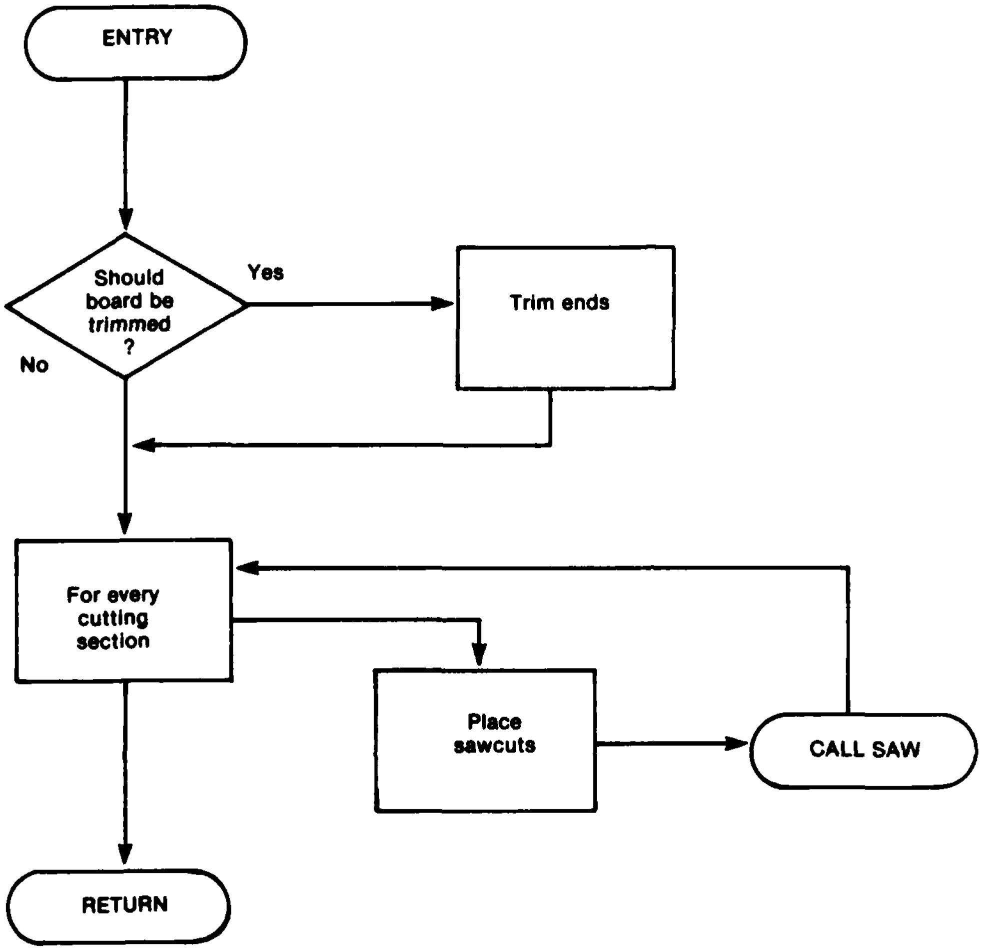


Figure 15.—Flowchart of subroutine AMEND. (ML83 5050)

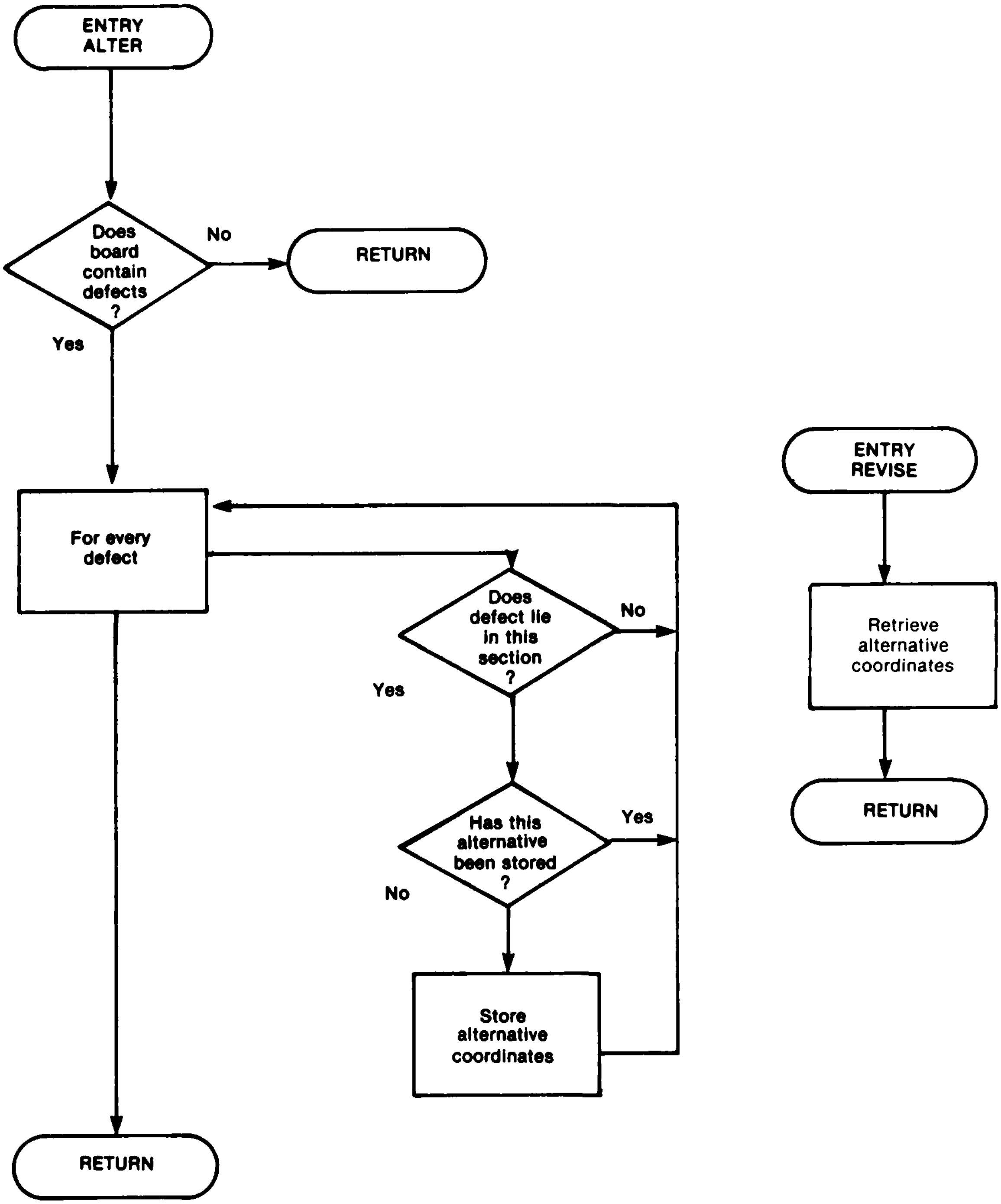


Figure 16.—Flowchart of subroutine ALTER. Entry points are ALTER and REVISE. (ML83 5045)

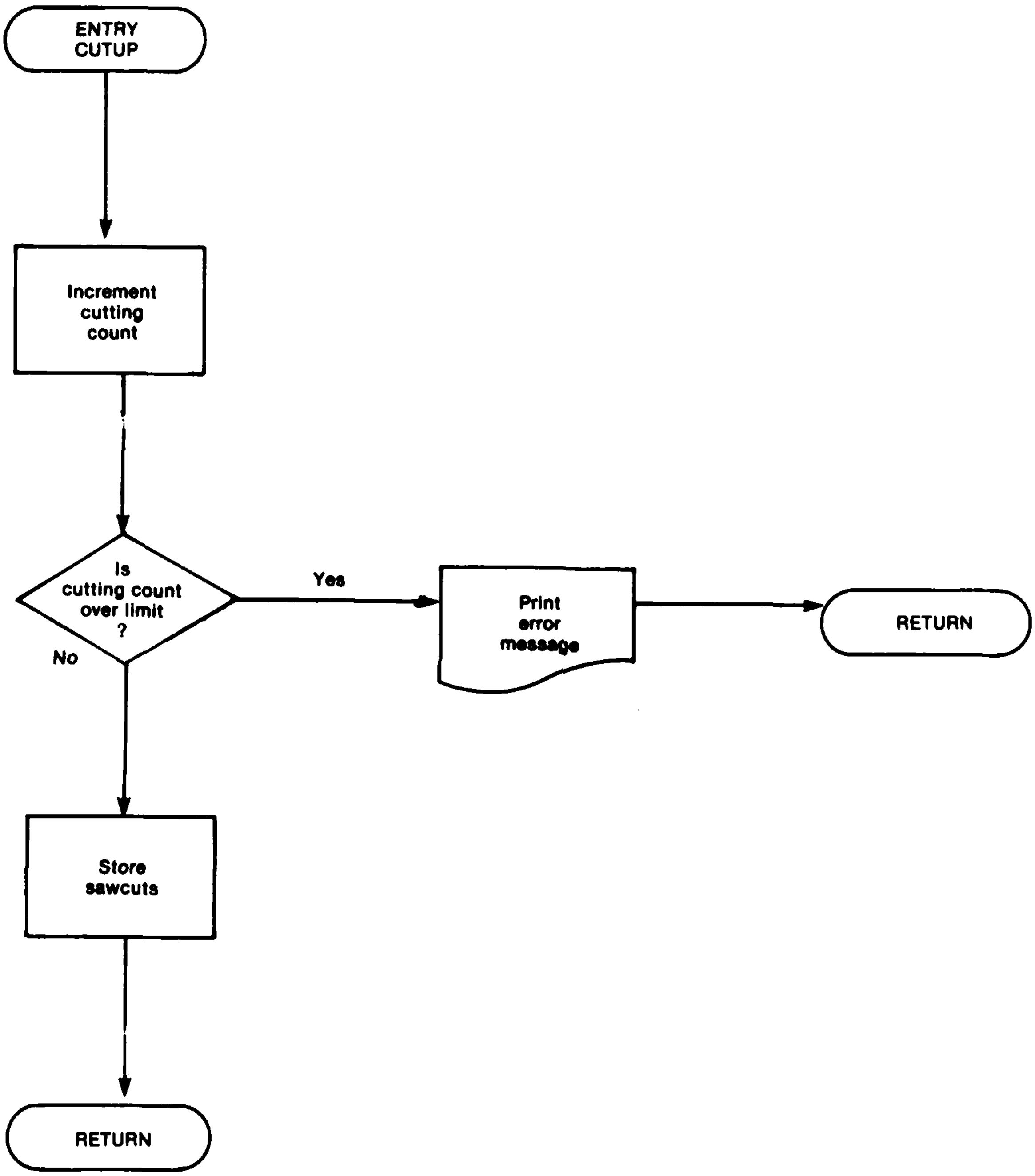


Figure 17.—Flowchart of subroutine CUTUP. (ML83 5046)

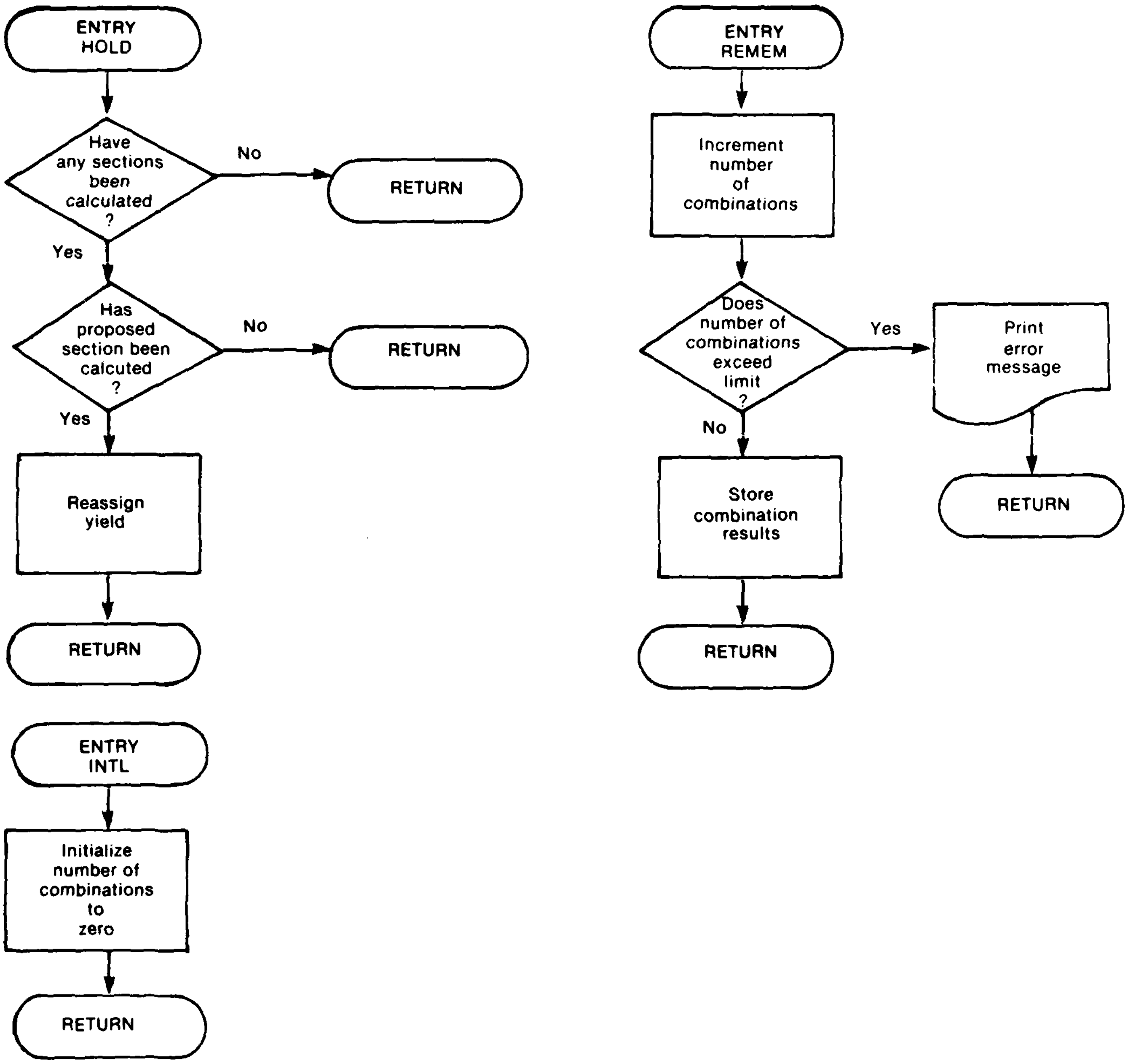


Figure 18.—Flowchart of subroutine HOLD. Entry points are HOLD, INTL, and REMEM. (ML83 5047)

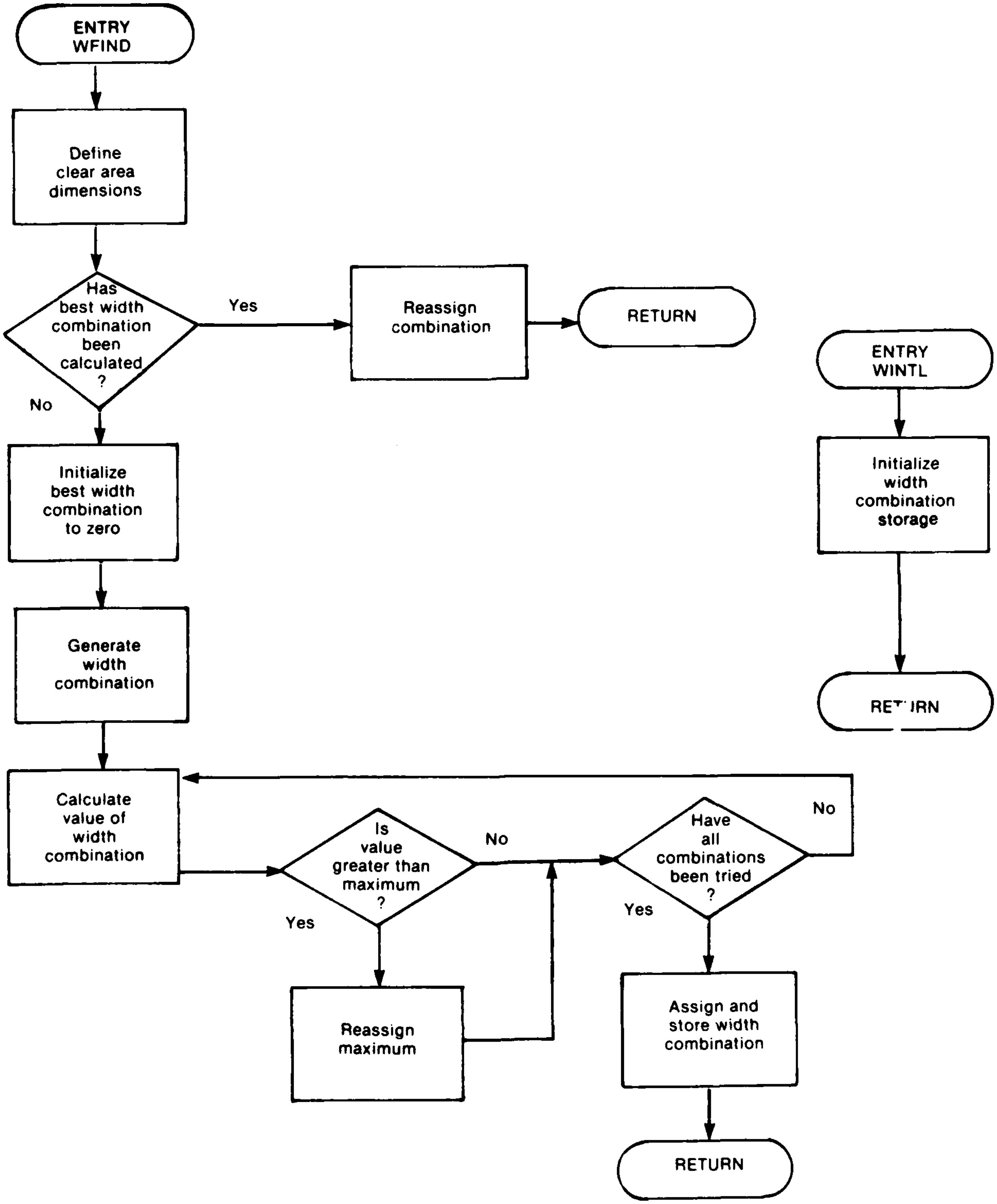


Figure 19.—Flowchart of subroutine WFIND. Entry points are WFIND and WINTL. (ML83 5048)

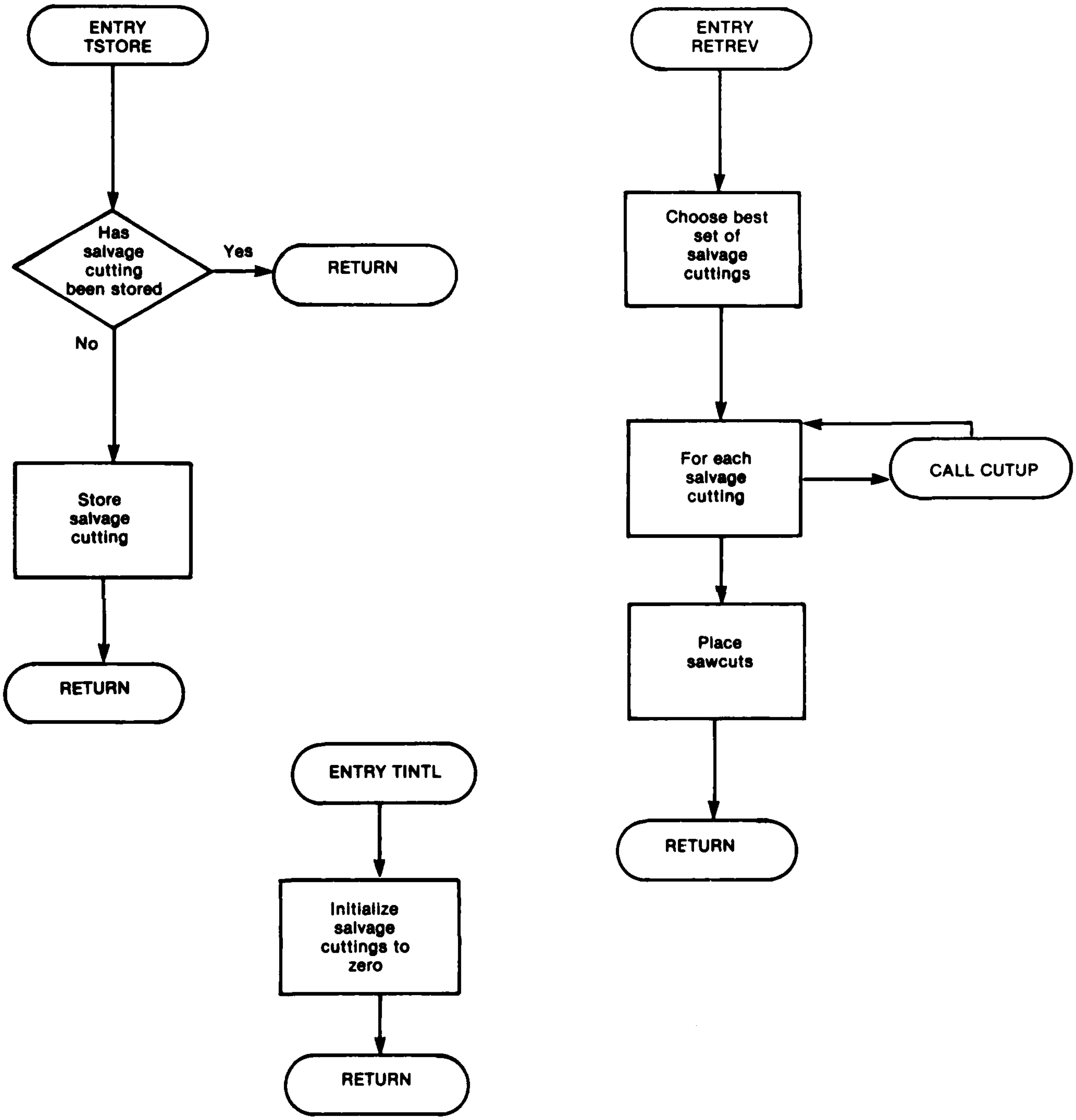


Figure 20.—Flowchart of subroutine TSTORE. Entry points are TSTORE, TINTL, and RETREV (ML83 5051)

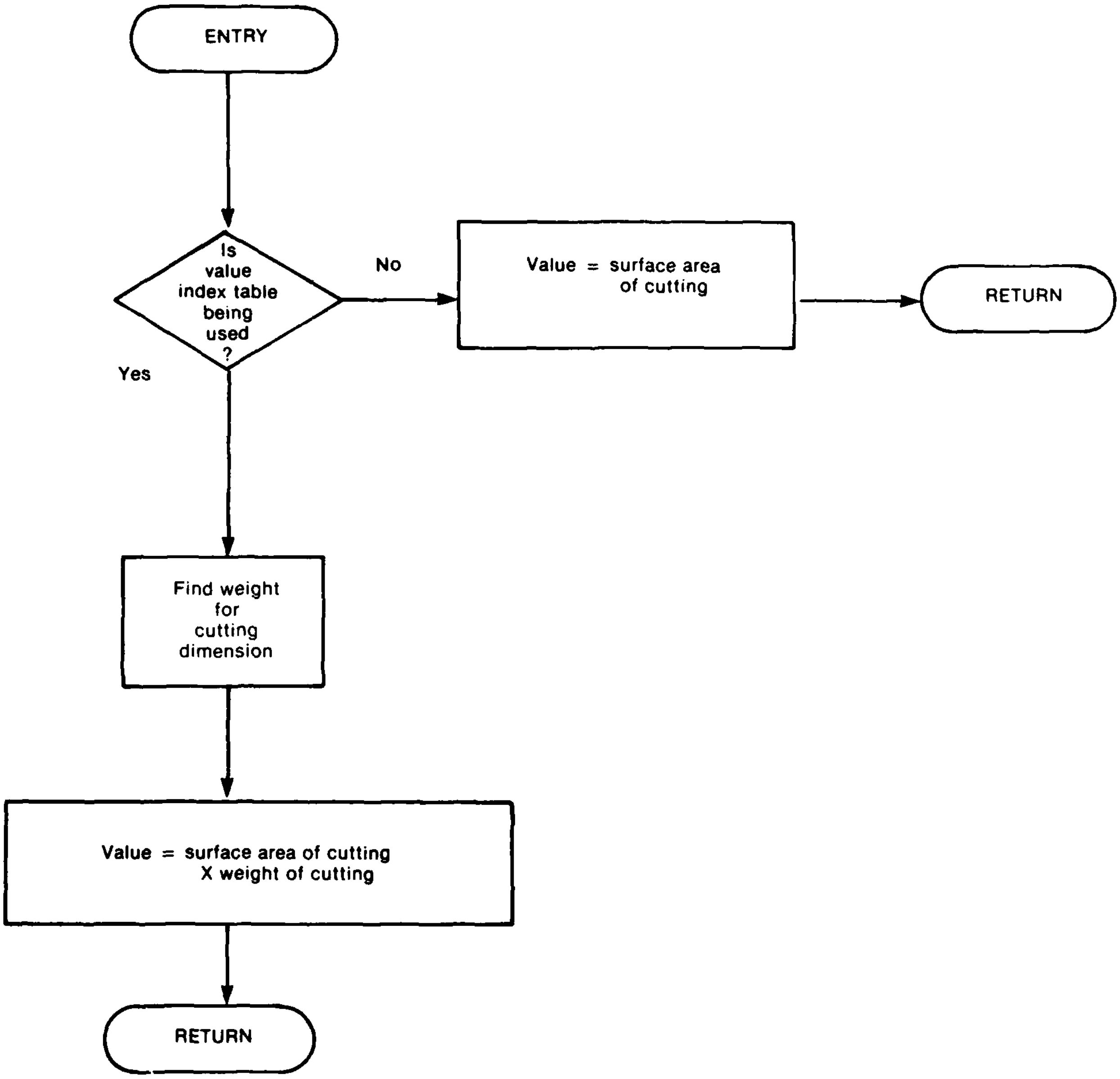


Figure 21.—Flowchart of function VALUE. (ML83 5040)

Appendix A: CROMAX Program Listing

```
C *** CROMAX ***
C *** PROGRAM TO MAXIMIZE CROSSCUTTING
C *** U S FOREST PRODUCTS LABORATORY
      IMPLICIT INTEGER(A-Z)
      REAL LTEMP,GRID,AREA,AREAC,WTEMP,INDEX,VTEMP,VALUE
      LOGICAL BAD,DONE,LAST,PURE,TRIM,KNOW(25),OK,REJECT
      CHARACTER*3 GRADE
      CHARACTER*5 NBOARD
      DIMENSION  (2),ALTCOM(25)
      COMMON  ALL NBOARD,BAD,BDLX,BDLY,BDUX,BDUY
      COMMON MS BOARD(100,24)
      COMMON MP LENGTH(10),NLEN,WIDTH(10),NWID
      COMMON MPA NCUTS,SAWCUT(200,4),LAST,PIECE(100,2),NPIECE
      COMMON  MSF YIELD(25)
      COMMON  MAFG PIECOM(2,25),LOW(25,2),HI(25,2)
      COMMON  MATFG NSEC
      COMMON  NC NI,TC (100),DV (100),NDEF(25),DU (100)
      COMMON  MV INDE (4,8),VLEN, WID,LINDE (8),WINDE (4),NV
      DATA INDE/ 32*1.
      DATA ACTIVE 1 ,MA  2 ,UNIT 11 ,UNIT2 10
      GRID=.25
C *** READ IN OPTIONS
      READ(5,5)EDGE
    5 FORMAT(5 ,I2)
      READ(5,6)NV
    6 FORMAT(5 ,I1)
      IF(NV.EQ.0)GO TO 9
      READ(5,10)VLEN,VWID
      READ(5,15)(WINDE (I),I=1,VWID)
      READ(5,15)(LINDE (I),I=1,VLEN)
      DO   I=1,4
        READ(5,7)(INDE (I,J),J=1,8)
    7   FORMAT(8F5.2)
    8 CONTINUE
    9 READ ...
...
      IF(BAD)GO TO 75
      DLY=DLY+1
      DO 65 K=DLX(I),DUX(I)
        K2=K  36
        IBIT=IABS(K-K2*36)+1
        K2=K2+1
        DO 60 K1=DLY,DUY(I)
          BITS(BOARD(K1,K2),IBIT,1)=1
   60   CONTINUE
   65 CONTINUE
   75 CONTINUE
      IF(.NOT.BAD)GO TO 78
      WRITE(6,77)NBOARD
   77 FORMAT(5X,'BOARD NUMBER',A5,'EXCEEDS BOUNDS')
      GO TO 30
C *** ORGANIZE PROCESSING OF BOARD + ASSIGN INITIAL VALUES
   78 NSEC=(BDUX-BDLX)/LENGTH(1)+1)
      IF(NSEC.GT.25)NSEC=25
C *** INITIALIZE FIRST COMBINATION
      IPOSEC=1
      YMAX =0.
      DONE=.FALSE.
      DO 80 I=1,2
        DO 80 J=1,NSEC
          YIELD(J)=0
          ALTCOM(J)=0.
          KNOW(J)=.FALSE.
   80     PIECOM(I,J)=1
      COMB=1
      CALL INTL
      START=BDLX
      PURE=.TRUE.
      OLDSEC=NSEC
      MAXSEC=NSEC
C *** DEFINE BOUNDS
   85 MAXSEC=NSEC
      NCUTS=0
      DO 90 I=IPOSEC,NSEC
        IF(MAXSEC.LT.I)GO TO 90
        KNOW(I)=.FALSE.
        YIELD(I)=0.
        LOW(I,ACTIVE)=START
        HI(I,ACTIVE)=LOW(I,ACTIVE)+LENGTH(PIECOM(1,I))
        IF(HI(I,ACTIVE).GT.BDUX)THEN
          MAXSEC=I-1
        ELSE
          START=HI(I,ACTIVE)+1
          IF(LONE(I))GO TO 87
          CALL ...(LOW(I,ACTIVE),HI(I,ACTIVE),YIELD(I),KNOW(I)
   87     CALL ...
        ENDIF
   90 CONTINUE
C *** CALCULATE YIELD
...
```

```
      DO 403 P=1,MAXSEC
        TY=TY+(RIPCOM(1,P)**2)
  403 CONTINUE
      IF(TIME.LE.1)GO TO 404
      IF(... ACTIVE ... .GE..01)GO TO 404
      IF(T...LT.M) GO TO 120
  404 [illegible] ACTIVE)
      [illegible]
      [illegible] SEC
      DO 115 I=1,MAXSEC
        [illegible] = LOW(I,ACTIVE)
        [illegible] = HI(I,ACTIVE)
  115 [illegible] RIPCOM(1,I)
C *** ARE OTHER ALTERNATES AVAILIBLE FOR PROCESSING?
  120 IF [illegible] .TRUE.
      [illegible]
      IF [illegible] TO 124
  119 [illegible]
      DO 117 [illegible]
        IF [illegible] TO 117
        IF [illegible] .EQ.0) DONE=.TRUE.
        IF [illegible] TO 117
        [illegible] +1
        IF [illegible] THEN
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        ENDIF
      [illegible]
      [illegible] 124
      [illegible]
      [illegible] TRYSEC+1,ACTIVE)+1
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible] (ISTART,ALTCOM ...
      [illegible]
      [illegible] REPROT=.TRUE.
  127 [illegible]
      [illegible] LENGTH(RIPCOM ...
      [illegible]
      [illegible] THEN
      [illegible]
      [illegible] TRUE.
      [illegible]
      [illegible] HI(I,ACTIVE) ... KNOW(I))

      DO 126 I=1,NSEC
  126 ALTCOM(I)=0
      START =BIG
      IF(TRYSEC.EQ.1 GO TO 126
      DO 130 I=1,TRYSEC-1
        LOW(I,ACTIVE) =START
        HI(I,ACTIVE) = LOW(I,ACTIVE)+LENGTH(RIPCOM(1,I))
        CALL ALTER (LOW(I,ACTIVE),HI(I,ACTIVE),I,BIG)
        CALL ... (LOW(I,ACTIVE),HI(I,ACTIVE), ... (I),KNOW(I))
        START = HI(I,ACTIVE) +1
  130 CONTINUE
  126 TIME=TIME+1
      GO TO 45
  125 [illegible] TRYSEC +1
      TRYSEC=TRYSEC-1
      IF(TRYSEC.GT.0 GO TO 127
C *** ALL COMBINATIONS HAVE BEEN TRIED
C *** PRINT OUTPUT
  200 [illegible]=0
      CALL [illegible] DELETE
  205 IF(BAD) WRITE ... ARE
      IF(BAD) [illegible]=1
      IF(BAD) WRITE [illegible]
      IF(BAD) [illegible]
      WRITE [illegible]
  210 FORMAT [illegible]
      IF [illegible]
      [illegible]
      DO [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
  215 [illegible]
      [illegible]
  230 CONTINUE
      [illegible]
      WRITE [illegible]
  240 FORMAT [illegible]
     *      [illegible]
     *      [illegible]
      IF [illegible]
  245 FORMAT [illegible]
  250 WRITE [illegible]
      WRITE [illegible]
      IF [illegible]
      WRITE [illegible]
  255 FORMAT [illegible]
      [illegible]
  257 FORMAT [illegible]
      [illegible]
      WRITE [illegible]
  260 FORMAT [illegible]
      [illegible]
  300 [illegible]
  270 [illegible]
      [illegible]
  300 [illegible]
      END
```

```
***  STATEMENT NUMBERS  ***

    5     23   *24
    6     25   *26
    7     32   *33
    8     31   *34
    9     27   *35
   10     28    35   *36
   15     29    30    37   *38    39
   20     40   *41
   25     42   *43
   30    *47    85   238   260   271
   35     48   *49
   40     50   *51    69    70
   43     55   *57
   45     54   *58
   50     64   *65
   55     53   *67
   60     77   *79
   65    *80
   75     68    71   *81
   77     83   *84   235
   78     67    82   *87
   80     93    94   *98
   85   *106   227
   87    119  *120
   90    109   109  *122
   93   *12[illegible]
   95   *1[illegible]   207
  100    127   1[illegible]   129  *133
  110    137  *1[illegible]
  113    159   160   162  *171
  115    15[illegible]  *157
  116    1[illegible]   180   181   183  *197
  117    [illegible]  *[illegible]
  118   *15[illegible]   1[illegible]   179   199   205
  120    139   146  *155
  124    125   126   157   172  *209
  125    214  *2[illegible]
  126    219  *[illegible]
  127   *211   [illegible]
  128    215  *21[illegible]
  130    219  *[illegible]
  131    201  *[illegible]
  132    207  *[illegible]
  200   *[illegible]
  205    174  *235
  210    239  *240
  225    247  *248
  230    245  *250
  240    [illegible]  *[illegible]
  245    [illegible]  *[illegible]
  250    241  *[illegible]
  255    [illegible]  *[illegible]
  2[illegible]    [illegible]  *[illegible]
  2[illegible]    [illegible]  *[illegible]
  270    [illegible]  *[illegible]
  2[illegible]    [illegible]  *[illegible]
  300     [illegible]  *[illegible]
  403    141  *143
  404    144   145  *147
  5[illegible]     [illegible]   *60
```

```
***  VARIABLES  ***

ACTIVE    *20   112   113   114   117   119   120   131   132   135   138   139
          145   147   151   152   174   186   187   188   190   194   195   220
          221   222   223   224
ALTCOM      9   *96  *163   164  *168   184   202  *216
ALTER     120   222
ALTSUM   *200  *202   203
AMEND     234
AREA        5   *63   251   252   258
AREAC       5  *251   252   258
BAD         6    10   *47  *52    53    71    82   128   134   235   236   237
          238
BDLX       10   *50    59    61    63    87   101   177   217
BDLY       10   *50    59    62    63   *66   120   227
BDUX       10   *50    52    59    61    63    87   114   190
BDUY       10   *50    52    59    62    63
BITS      *78
BOARD      11   *56    78
BREC     *149   234
COMB      *99   118   144  *226
DLX        17   *69    70    73
DLY       *69    70   *72    77
DOESKX    *73
DONE        6   *92  *161  *169   172  *213
DUX        17   *69    70    73
DUY        17   *69    70    77
EDGE      *23    66   234
GRADE     *48    64
GRID        5   *21    61    62    63   244   245
HOLD      119   195   223
I         *29   *30   *31   *32   *37   *39   *40    41    42   *54    56   *68
          *65    70    73    77   *93    96    97  *102   109   110   111   112
          113   114   115   117   119   120  *127   129   131   132  *137   138
         *150   151   152   153  *159   161   162   163   164   166   168   169
         *179   180   182   183   184   185   186   187   188   190   191   194
          195  *215   216  *219   220   221   222   223   224  *243   244   245
          246   259  *265   266   268
IABS       75
IBIT      *75    78
INDEX       5    18   *19   *32
INTL      100
J         *32   *55    56   *94    95    98   259   266   268
K        *201   202
K1        *77    78
K2        *74    75   *76    78
KNOW        6   *97  *110   119   129  *182   195   2[illegible]
KX         74    75
LAST        6    13
LENGTH     12   *37   *41    42    87   113   188   2[illegible]
LINDEX     18   *30
LTEMP       5   *61    64  *244   247   249
MAX       *20    91   139   145   147   151   152   256
MAXSEC   *104  *106   109  *115   125   126   127   1[illegible]   137   141   149   150
          159   176   177   178   180  *191   192  *1[illegible]   201  *204  *212
MY        146  *148
NBOARD     10   *46    64    83   235
NCUTS      13  *107  *206   260   263   265
ND         17   *48    67    69   157
NDEF       17   161   162   164
NLEN       12   *35    40    42   214
NPIECE     13  *130  *233  *236   237   241   247   255   259
NSEF       16   *87   *88    94   103   104   10[illegible]   1[illegible]   2[illegible]   215
NV         18   *25    27   256
NWID       12   *35    42
OK          6  *158   160  *165
OPTSEC   *103  *155   209
PIECE      13   244   245   246   259
PURF        6  *102   155  *156  *210
R        *141   142
REJECT      6  *175   181  *186   189  *192   198   199
REMEM     132
RE[illegible]F    184
```

```
RIPCOM      15   *98   113   142  *153   188  *211   214   221  *228
SAW        131
SAWCUT      13   266   268
STARTX    *101   112  *117  *173  *174   184   186   187  *194  *217   220  *224
TRIM         6   131
TRMSEC     *90   108   126   127  *136   155  *166   173   174   178   179  *209
           211   214   218   219   228  *229   230
TSEC      *176   179   198   204
TX        *140  *142   146   148
UNIT       *20   237   258   259
UNIT2      *20   263   268
VALUE        5   246
VLEN        16   *28   *30
VTEMP        5  *246   247
VWID        18   *28   *29
WIDTH       1.   *30    42
WINDEX      18   *29
WINT        45
WTEMP        5   *62    64  *245   247   249
XHI         15  *113   114   117   119   120   131   132  *152   174   186  *188
           190   194   195  *221   222   223   224
XLOW        15  *112   113   119   120   131   132  *151  *187   188   195  *220
           221   222   223
Y            9   *91  *135  *138   139   145  *147   256
YIELD       14   *95  *111   119   132   138  *183   195   223
YTEMP     *212  *249   251   252
```

Subroutine SAW

```
      SUBROUTINE SAW(YLOW,YHI,SECT)                                    SAW
C *** SCAN FOR CUTTINGS                                                 SAW
      IMPLICIT INTEGER(A-Z)                                            SAW
      LOGICAL BAD,TRIM                                                 SAW
      REAL YIELD                                                       SAW
      CHARACTER*5 NBOARD                                               SAW
      COMMON /ALL/ NBOARD,BAD,BDLX,BDLY,BDUX,BDUY                      SAW
      COMMON /MS/ BOARD(100,4)                                         SAW
      COMMON /MR/ LENGTH(10),NLEN,WIDTH(10),NWID                       SAW
      COMMON /RR/ YIELD(25)                                            SAW
      YIELD(SECT)=0                                                    SAW
      CLRKNT=0                                                         SAW
      YLOW=BDLY                                                        SAW
      XHI=BDUX                                                         SAW
      TRIM=.FALSE.                                                     SAW
      IF((XHI-XLOW).GT.LENGTH(1))TRIM=.TRUE.                           SAW
      IF(TRIM)CALL RANGE(XLOW,YLOW,XHI,YHI,SECT)                       SAW
      DO 100 J=(BDLY+1),BDUY                                           SAW
        DO 50 I=(XLOW+1),(XHI-1)                                       SAW
          K=(I-1)/36                                                   SAW
          IBIT=IABS(I-(K*36))+1                                        SAW
          L=K+1                                                        SAW
          IF(BITS(BOARD(J,L)),IBIT,1).EQ.0)GO TO 50                    SAW
          GO TO 75                                                     SAW
   50   CONTINUE                                                       SAW
        CLRKNT=CLRKNT+1                                                SAW
        GO TO 100                                                      SAW
   75   IF(CLRKNT.LT.WIDTH(1))GO TO 80                                 SAW
      YHI=J-1                                                          SAW
        CALL RIP(XLOW,XLOW,XHI,YHI,CLRKNT,SECT,TRIM)                   SAW
        IF(BAD)RETURN                                                  SAW
   80   YLOW=J                                                         SAW
        YHI=J                                                          SAW
        CLRKNT=0                                                       SAW
  100 CONTINUE                                                         SAW
      IF(CLRKNT.GE.WIDTH(1))THEN                                       SAW
        YHI=BDUY                                                       SAW
        CALL RIP(XLOW,YLOW,XHI,YHI,CLRKNT,SECT,TRIM)                   SAW
      ENDIF                                                            SAW
      IF(TRIM)CALL TRIMIT                                              SAW
      RETURN                                                           SAW
      END                                                              SAW
```

*** STATEMENT NUMBERS ***

```
  50     23   *25
  75     24   *28
  80     28   *32
 100     27   *35
```

*** VARIABLES ***

```
BAD        4     7    31
BDLX       7
BDLY       7    13    18
BDUX       7
BDUY       7    14    18    37
BITS      23
BOARD      8    23
```

CLRKNT	*12	*26	28	30	*34	36	38
DO100I	*18						
DO50IX	*19						
IABS	21						
IBIT	*21	23					
IX	20	21					
IY	23	29	32	33			
K2	*20	21	*22	23			
LENGTH	9	16					
NBOARD	7						
NLEN	9						
NWID	9						
RANGE	17						
RIP	30	38					
SAW	1						
SECT	1	11	17	30	38		
TRIM	4	*15	*16	17	30	38	40
TRIMIT	40						
WIDTH	9	28	36				
UX	1	16	17	19	30	38	
UY	1	16	17	19	30	39	
YHI	*14	17	*29	30	*33	*37	38
YIELD	5	10	*11				
YLOW	*13	17	30	*32	38		

Subroutine RIP

```
      SUBROUTINE RIP(LX,LY,UX,UY,CLRKNT,SECT,TRIM)                     RIP
C *** SAW CUTTINGS                                                      RIP
      IMPLICIT INTEGER(A-Z)                                            RIP
      REAL VALUE                                                       RIP
      LOGICAL OK,LAST,DONE,TRIM                                        RIP
      DIMENSION WCOM(25)                                               RIP
      COMMON /MSR/LENGTH(10),NLEN,WIDTH(10),NWID                       RIP
      COMMON /MRA/NCUTS,SAWCUT(200,4),LAST,PIECE(100,2),NPIECE         RIP
      DATA ACTIVE/1/,MAX/2/                                            RIP
C *** CALCULATE BEST WIDTH COMBINATION TO USE                          RIP
      NW=0                                                             RIP
      CALL WFIND(CLRKNT,WCOM,NW,(UX-LX))                               RIP
      IF(NW.EQ.0)RETURN                                                RIP
      DO 50 I=1,NW                                                     RIP
        YHI=LY+WIDTH(WCOM(I))                                          RIP
        CALL CUTUP(LX,LY,UX,YHI,SECT)                                  RIP
        IF(TRIM)CALL STORE(LX,LY,UX,YHI)                               RIP
        LY=YHI+1                                                       RIP
   50 CONTINUE                                                         RIP
      RETURN                                                           RIP
      END                                                              RIP
```

*** STATEMENT NUMBERS ***

50	14	*19

*** VARIABLES ***

ACTIVE	*9				
CLRKNT	1	12			
CUTUP	16				
DONE	5				
I	*14	15			
LAST	5	8			
LENGTH	7				
LX	1	12	16	17	
LY	1	15	16	17	*18
[illegible]	[illegible]				
NCUTS	[illegible]				
NLEN	[illegible]				
NPIECE	[illegible]				
NW	[illegible]	[illegible]	[illegible]	[illegible]	
NWID	[illegible]				
[illegible]	[illegible]				
PIECE	[illegible]				
RIP	[illegible]				
[illegible]	[illegible]				
[illegible]	[illegible]	[illegible]			
[illegible]	[illegible]				
TRIM	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]				
[illegible]	[illegible]				
[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]				
[illegible]		[illegible]			
YHI	[illegible]	[illegible]	[illegible]	[illegible]	

Subroutine RANGE

```
      SUBROUTINE RANGE(PLX,PLY,PUX,PUY,SECT)
      IMPLICIT INTEGER(A-Z)
      LOGICAL BAD,TRIM,LAST,NOGOOD,WHOLE,TWICE,PIP
      DIMENSION COORD(25,4),TPIECE(25,4),ULOW(25)
      CHARACTER*5 NBOARD
      COMMON ALL NBOARD,BAD,BDLX,BDLY,BDUX,BDUY
      COMMON MS BOARD(100,24)
      COMMON NPA NCUTS,SAVCUT(200,4),LAST,PIECE(100,2),NPIECE
      COMMON MAPG PIPCOM(2,25),IDUM1(25,2),IDUM2(25,2)
      COMMON USR LENGTH(10),NLEN,WIDTH(10),NWID
      DATA LX/1/,LY/2/,UX/3/,UY/4/
      NP=0
      LMA=PIPCOM(1,SECT)
      IF(LAST)LMA=PIPCOM(2,SECT)
      RETURN


      ENTRY TRIMIT
C *** CHECK IF POSSIBLE TRIM PIECES EXIST
C     WRITE(6,200)
  200 FORMAT(25X,'TRIMIT ENTERED')
      WHOLE=.TRUE.
      IF(NP.GT.0)WHOLE=.FALSE.
      LIMIT=NP
      IF(WHOLE)LIMIT=1
      TP=1
      DO 20 I=1,LIMIT
        IF(BAD)GO TO 20
        TWICE=.FALSE.
        TPIECE(TP,LX)=PLX
        TPIECE(TP,UX)=PUX
        IF(.NOT.WHOLE)GO TO 10
        TPIECE(TP,LY)=BDLY
        TPIECE(TP,UY)=BDUY
        TP=TP+1
        GO TO 20
   10   TPIECE(TP,LY)=COORD(I,UY)+1
        IF(TPIECE(TP,LY).GT.BDUY)GO TO 19
   15   IF(I.LT.LIMIT)THEN
        TPIECE(TP,UY)=COORD(I+1,LY)-1
        ELSE
        TPIECE(TP,UY)=BDUY
        ENDIF
   17   [illegible] TPIECE(TP,LY)).GE.WIDTH(1))TP=TP+1
        [illegible] GO TO 19
        [illegible]
        [illegible]
   1?   [illegible] OVERFLOW IN TRIMIT',I5,20(1H*))
        [illegible]
   19   [illegible]
        [illegible]
        [illegible] COORD(I,LY)-1
        [illegible] GO TO 20
        [illegible] THEN
        [illegible]
        [illegible]
        [illegible] COORD(I,UY)+1
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
        [illegible]
C ***   [illegible] ARE DUPLICATED
        [illegible]
      DO 180 I=1,TP
        IF(ICOPY.GT.0)GO TO 180
        IF(I.EQ.TEST)GO TO 180
        IF(TPIECE(TEST,LX).NE.TPIECE(I,LX))GO TO 180
        IF(TPIECE(TEST,LY).NE.TPIECE(I,LY))GO TO 180
        IF(TPIECE(TEST,UX).NE.TPIECE(I,UX))GO TO 180
        IF(TPIECE(TEST,UY).EQ.TPIECE(I,UY))ICOPY=1
  180 CONTINUE
      IF(ICOPY.GT.0)GO TO 185
  183 TEST=TEST+1
      IF(TEST.GT.TP)GO TO 24
C *** REMOVE TEST
  185 IF(TEST.GE.TP)GO TO 24
      DO 190 K=TEST,TP
        TPIECE(K,LX)=TPIECE(K+1,LX)
        TPIECE(K,LY)=TPIECE(K+1,LY)
        TPIECE(K,UX)=TPIECE(K+1,UX)
        TPIECE(K,UY)=TPIECE(K+1,UY)
  190 CONTINUE
      GO TO 183
C *** CHECK EACH POTENTIAL PIECE FOR GOOD TRIMMEND PIECE
C *** INFINITE ADDITIONAL CC ALLOWED
   24 DO 150 I=1,TP
       CALL TINTL
       PIP=.FALSE.
       DO 140 TRIAL=1,2
       NOGOOD=.FALSE.
       IF(TRIAL.EQ.2)PIP=.TRUE.
       IF(PIP.AND.(TPIECE(I,UY)-TPIECE(I,LY)).LE.WIDTH(1))GO TO 140
       CLRPNT=0
       XLOW=TPIECE(I,LX)
       XHI=TPIECE(I,UX)
       YLOW=TPIECE(I,LY)
       YHI=TPIECE(I,UY)
       DO 80 IX=(TPIECE(I,LX)+1),TPIECE(I,UX)
         IF(NOGOOD)GO TO 80
         K2=IX/36
         IBIT=IABS(IX-K2*36)+1
         K2=K2+1
   25    DO 30 IY=(YLOW+1),YHI
           IF(BITS(BOARD(IY,K2),IBIT,1).EQ.0)GO TO 30
            GO TO 35
   30    CONTINUE
         CLRPNT=CLRPNT+1
       IF(IX.EQ.TPIECE(I,UX))GO TO 35
         GO TO 80
C *** DEFECT DETECTED
   35    IF(PIP)GO TO 45
         IF(CLRPNT.GE.LENGTH(1))GO TO 40
          XLOW=IX
         CLRPNT=0
         YLOW=TPIECE(I,LY)
         YHI=TPIECE(I,UY)
         IF((TPIECE(I,UX)-XLOW).LT.LENGTH(1))NOGOOD=.TRUE.
         GO TO 80
   40    XHI=IX
         ALEN=XHI-XLOW
         IL=0
         DO 55 K=1,LMAX
           IF(LENGTH(K).LE.ALEN)IL=K
   55    CONTINUE
         IF(IL.EQ.0)GO TO 75
       XHI=XLOW+LENGTH(IL)
         CALL WFIND(XHI,XLOW,WCNT,UCNT,LENGTH(IL))
         IF(WCNT.EQ.0)GO TO 75
         [illegible]
         DO 60 K=1,WCNT
           [illegible]
           [illegible] (XLOW,XHI,YHI,TRIAL,SECT)
           [illegible]
   60    CONTINUE
         [illegible]
        CLRPNT=0
        [illegible]
```

```
145          YLOW=TYLOW                                               RANGE
146        IF((XHI-XLOW).LT.LENGTH(I))NOGOOD=.TRUE.                   RANGE
147            GO TO 80                                               RANGE
148     75     XLOW=IX                                                RANGE
149            YLOW=TPIECE(I,LY)                                      RANGE
150            YHI=TPIECE(I,UY)                                       RANGE
151            CLRKNT=0                                               RANGE
152            IF((XHI-XLOW).LT.LENGTH(I))NOGOOD=.TRUE.               RANGE
153            GO TO 80                                               RANGE
154 C ***    CHECK IF RIPPING IS POSSIBLE                             RANGE
155     45     CLRWID=0                                               RANGE
156            DO 47 M=(YLOW+1),YHI                                   RANGE
157              CLRWID=CLRWID+BITS(BOARD(M,K2),IBIT,1)               RANGE
158     47     CONTINUE                                               RANGE
159            IF(((YHI-YLOW)-CLRWID).LT.WIDTH(I))GO TO 78            RANGE
160            IF(IY-YLOW.LE.WIDTH(I))GO TO 77                        RANGE
161            YHI=IY-1                                               RANGE
162            CLRKNT=CLRKNT+1                                        RANGE
163            GO TO 80                                               RANGE
164     77     IF((YHI-IY).LT.WIDTH(I))GO TO 78                       RANGE
165            YLOW=IY                                                RANGE
166            CLRKNT=CLRKNT+1                                        RANGE
167            GO TO 80                                               RANGE
168     78     IF(CLRKNT.GE.LENGTH(I))GO TO 40                        RANGE
169            XLOW=IX                                                RANGE
170            CLRKNT=0                                               RANGE
171            YLOW=TPIECE(I,LY)                                      RANGE
172            YHI=TPIECE(I,UY)                                       RANGE
173            IF((XHI-XLOW).LT.LENGTH(I))NOGOOD=.TRUE.               RANGE
174     80   CONTINUE                                                 RANGE
175    140   CONTINUE                                                 RANGE
176          CALL RETREV                                              RANGE
177    150 CONTINUE                                                   RANGE
178        RETURN                                                     RANGE
179        ENTRY STORE(TLX,TLY,TUX,TUY)                               RANGE
180        NP=NP+1                                                    RANGE
181        IF(NP.LE.25)GO TO 160                                      RANGE
182        BAD=.TRUE.                                                 RANGE
183        WRITE(6,155)NBOARD                                         RANGE
184    155 FORMAT(25X,15(1H*),' OVERFLOW IN STORE',A5,15(1H*))        RANGE
185        RETURN                                                     RANGE
186    160 COORD(NP,LX)=TLX                                           RANGE
187        COORD(NP,LY)=TLY                                           RANGE
188        COORD(NP,UX)=TUX                                           RANGE
189        COORD(NP,UY)=TUY                                           RANGE
190        RETURN                                                     RANGE
191        END                                                        RANGE
```

```
   77     160  *164
   78     159   164  *168
   80     106   116   125   147   153   163   167  *174
  140      96    99  *175
  150      93  *177
  155     183  *184
  160     181  *186
  170     *70
  180      71    72    73    74    75    76   *78
  183     *80    90
  185      79   *83
  190      84   *89
  200     *21
```

```
                    *** VARIABLES ***

ALEN     *127   130
BAD         3     6    28   *46    64  *182
BDLX        6
BDLY        6    33    54    56
BDUX        6
BDUY        6    34    38    42
BITS      111   157
BOARD       7   111   157
CLRKNT   *100  *114   119  *121  *143  *151  *162  *166   168  *170
CLRWID   *155  *157   159
COORD       4    37    40    53    58  *186  *187  *188  *189
DO301Y   *110
DO47M    *156
DO80IX   *105
I         *27    37    39    40    53    55    58   *71    73    74    75    76
           77   *93    99   101   102   103   104   105   115   122   123   124
          144   149   150   171   172
IABS      108
IBIT     *108   111   157
ICOPY     *70    72   *77    79
IL       *128  *130   132   133   134
IX        107   108   115   120   126   142   148   169
IY        111   160   161   164   165
K         *84    85    86    87    88  *129   130  *137   138
K2       *107   108  *109   111   157
LAST        3     8    14
LENGTH     10   119   124   130   133   134   146   152   168   173
LIMIT     *24   *25    27    39
LMAX      *13   *14   129
LX        *11    30    60    74    85   101   105   186
LY        *11    33    37    38    40    44    53    56    58    75    85    93
          103   122   149   171   187
M         157
NBOARD      6    47   183
NCUTS       8
NLEN       10
NOGOOD      3   *97   106  *124  *146  *152  *173
NP        *12    23    24  *180   181   186   187   188   189
NPIECE      8
NWID       10
PIECE       8
PLX         1    30    60
PLY         1
PUX         1    31    61
PUY         1
RANGE       1
RETREV    176
RIP         3   *95   *98    99   118
RIPCOM      9    13    14
SAWCUT      8
SECT        1    13    14   139
STORE     179
TEST      *69    73    74    75    76    77   *80    81    83    84
TINTL      94
TLX       179   186
```

```
                    *** STATEMENT NUMBERS ***

10     32   *37
15    *33
17    *14    62
18     47   *48
19     33    45   *51
20     27    28    36    50    51    54   *63
24     67    81    83   *93
25   *110
30    111  *113
35    112   115  *118
40     11  *126   168
45    118  *155
47   *158
55    124  *131
60    137  *141
75    132   135  *148
```

```
TLY         179   187
TP          *26    30    31    33    34   *35    37    38    40    42   *44    45
            *49    53    54    56    58    60    61   *65    66    67    71    81
             83    84    93
TPIECE        4   *30   *31   *33   *34   *37    38   *40   *42    44   *53    54
            *56   *58   *60   *61    74    75    76    77   *85   *86   *87   *88
             93   101   102   103   104   105   115   122   123   124   144   149
            150   171   172
TRIAL       *96    98   139
TRIM          3
TRIMIT       18
TSTORE      139
TUX         179   188
TUY         179   189
TWICE         3   *29    51   *52
TXLOW      *136   145
UX          *11    31    61    76    87   102   105   115   124   144   188
UY          *11    34    37    40    42    44    53    54    58    77    88    99
            104   123   150   172   189
WCNT        134   135   137
WCOM          4   134   138
WFIND       134
WHOLE         3   *22   *23    25    32
WIDTH        10    44    99   138   159   160   164
XDUM1         9
XDUM2         9
XHI        *102  *126   127  *133   139  *144   146   152   173
XLOW       *101  *120   124   127   133   139  *142   146  *148   152  *169   173
YHI        *104   110  *123   134  *138   139   140  *150   156   159  *161   164
           *172
YLOW       *103   110  *122   134   136   138   139  *140  *145  *149   156   159
            160  *165  *171
```

Subroutine AMEND

```
 1          SUBROUTINE AMEND(BSEC,EDGE)                                    AMEND
 2 C ***    SMOOTH OUT ANSWERS                                             AMEND
 3          IMPLICIT INTEGER(A-Z)                                          AMEND
 4          LOGICAL BAD,LAST,TRIM                                          AMEND
 5          CHARACTER*5 NBOARD                                             AMEND
 6          COMMON /ALL/NBOARD,BAD,BDLX,BDLY,BDUX,BDUY                     AMEND
 7          COMMON /MSR/LENGTH(10),NLEN,WIDTH(10),NWID                     AMEND
 8          COMMON /MRA/NCUTS,SAWCUT(200,4),LAST,PIECE(100,2),NPIECE       AMEND
 9          COMMON /MAFG/RIPCOM(2,25),XLOW(25,2),XHI(25,2)                 AMEND
10          COMMON /MATFG/NSEC                                             AMEND
11          DATA ACTIVE/1/,MAX/2/                                          AMEND
12          NCUTS=0                                                        AMEND
13          IF(EDGE.LE.0)GO TO 30                                          AMEND
14          NCUTS=NCUTS+1                                                  AMEND
15          SAWCUT(NCUTS,1)=BDLX                                           AMEND
16          SAWCUT(NCUTS,2)=BDLY-EDGE                                      AMEND
17          SAWCUT(NCUTS,3)=BDUX                                           AMEND
18          SAWCUT(NCUTS,4)=BDLY                                           AMEND
19    30    LAST=.TRUE.                                                    AMEND
20          DO 100 I=1,BSEC                                                AMEND
21            IF(BAD)GO TO 100                                             AMEND
22           IF((XLOW(I,MAX)-1).LT.BDLX)GO TO 65                           AMEND
23            NCUTS=NCUTS+1                                                AMEND
24    45      IF(NCUTS.LE.200)GO TO 60                                     AMEND
25              WRITE(6,50)NBOARD                                          AMEND
26    50        FORMAT(10X,'TOO MANY SAWCUTS',A5)                          AMEND
27              BAD=.TRUE.                                                 AMEND
28            GO TO 100                                                    AMEND
29    60      SAWCUT(NCUTS,1)=XLOW(I,MAX)-1                                AMEND
30            SAWCUT(NCUTS,2)=BDLY                                         AMEND
31            SAWCUT(NCUTS,3)=XLOW(I,MAX)                                  AMEND
32            SAWCUT(NCUTS,4)=BDUY                                         AMEND
33    65     IF((XHI(I,MAX)+1).GT.BDUX)GO TO 75                            AMEND
34           NCUTS=NCUTS+1                                                 AMEND
35            IF(NCUTS.GT.200)GO TO 45                                     AMEND
36            SAWCUT(NCUTS,1)=XHI(I,MAX)                                   AMEND
37            SAWCUT(NCUTS,2)=BDLY                                         AMEND
38            SAWCUT(NCUTS,3)=XHI(I,MAX)+1                                 AMEND
39            SAWCUT(NCUTS,4)=BDUY                                         AMEND
40    75      CALL SAW(XLOW(I,MAX),XHI(I,MAX),I,TRIM)                      AMEND
41   100    CONTINUE                                                       AMEND
42          RETURN                                                         AMEND
43          END                                                            AMEND
```

```
                   *** STATEMENT NUMBERS ***

  30       13   *19
  45      *24    35
  50       25   *26
  60       24   *29
  65       22   *33
  75       33   *40
 100       20    21    28   *41

                   *** VARIABLES ***

ACTIVE    *11
AMEND       1
BAD         4     6    21   *27
```

```
BDLX        6     15    22
BDLY        6     16    18    30    37
BDUX        6     17    33
BDUY        6     32    39
BSEC        1     20
EDGE        1     13    16
I         *20     22    29    31    33    36    38    40
LAST        4      8   *19
LENGTH      7
MAX       *11     22    29    31    33    36    38    40
NBOARD      6     25
NCUTS       8    *12   *14    15    16    17    18   *23    24    29    30    31
           32    *34    35    36    37    38    39
NLEN        7
NPIECE      8
NSEC       10
NWID        7
PIECE       8
PIPCOM      9
SAW        40
SAWCUT      8    *15   *16   *17   *18   *29   *30   *31   *32   *36   *37   *38
          *39
TRIM        4     40
WIDTH       7
XHI         9     33    36    38    40
XLOW        9     22    29    31    40
```

Subroutine ALTER

```
      SUBROUTINE ALTER(XLOW,XHI,SECT,BDLY)                              ALTER
      IMPLICIT INTEGER(A-Z)                                             ALTER
      DIMENSION CHG(25,50)                                              ALTER
      LOGICAL HAVEIT                                                    ALTER
      COMMON /MC/ND,DLX(100),DUX(100),NDEF(25),DUY(100)                 ALTER
      IF(ND.EQ.0)RETURN                                                 ALTER
      NDEF(SECT)=0                                                      ALTER
      DO 25 I=1,ND                                                      ALTER
        IF(DUY(I).LE.BDLY)GO TO 25                                      ALTER
        IF(DLX(I).GE.XHI)GO TO 25                                       ALTER
        IF(DUX(I).LE.XLOW)GO TO 25                                      ALTER
        IF(DUX(I).GE.XHI)GO TO 25                                       ALTER
        IF(NDEF(SECT).EQ.0)GO TO 15                                     ALTER
        HAVEIT=.FALSE.                                                  ALTER
        DO 10 K=1,NDEF(SECT)                                            ALTER
          IF(HAVEIT)GO TO 10                                            ALTER
          IF(DUX(I).EQ.CHG(SECT,K))HAVEIT=.TRUE.                        ALTER
   10   CONTINUE                                                        ALTER
        IF(HAVEIT)GO TO 25                                              ALTER
   15   NDEF(SECT)=NDEF(SECT)+1                                         ALTER
        CHG(SECT,NDEF(SECT))=DUX(I)                                     ALTER
   25 CONTINUE                                                          ALTER
      RETURN                                                            ALTER
      ENTRY REVISE(SECT,NEWLX,ALTCOM)                                   ALTER
      NEWLX=CHG(SECT,ALTCOM)                                            ALTER
      RETURN                                                            ALTER
      END                                                               ALTER
```

```
                    ***  STATEMENT NUMBERS  ***

 10        15    16   *18
 15        13   *20
 25         8     9    10    11    12    19   *22

                    ***  VARIABLES  ***

ALTCOM     24    25
ALTER       1
BDLY        1     9
CHG         3    17   *21    25
DLX         5    10
DUX         5    11    12    17    21
DUY         5     9
HAVEIT      4   *14    16   *17    19
I          *8     9    10    11    12    17    21
K         *15    17
ND          5     6     8
NDEF        5    *7    13    15   *20    21
NEWLX      24   *25
REVISE     24
SECT        1     7    13    15    17    20    21    24    25
XHI         1    10    12
XLOW        1    11
```

Subroutine CUTUP

```
 1       SUBROUTINE CUTUP(LX,LY,UX,UY,SECT)                                  CUTUP
 2       IMPLICIT INTEGER(A-Z)                                                CUTUP
 3       LOGICAL BAD,LAST                                                     CUTUP
 4       REAL YIELD,VALUE                                                     CUTUP
 5       CHARACTER*5 NBOARD                                                   CUTUP
 6       COMMON /ALL/NBOARD,BAD,BDLX,BDLY,BDUX,BDUY                           CUTUP
 7       COMMON /MRA/NCUTS,SAWCUT(200,4),LAST,PIECE(100,2),NPIECE             CUTUP
 8       COMMON MRF/YIELD(25)                                                 CUTUP
 9       NPIECE=NPIECE+1                                                      CUTUP
10       IF(NPIECE.LE.100)GO TO 50                                            CUTUP
11         WRITE(6,75)SECT                                                    CUTUP
12   75    FORMAT(10X,'SECTION',I3,'EXCEEDS CUTTING LIMIT')                   CUTUP
13         BAD=.TRUE.                                                         CUTUP
14       RETURN                                                               CUTUP
15   50    PIECE(NPIECE,1)=UX-LX                                              CUTUP
16         PIECE(NPIECE,2)=UY-LY                                              CUTUP
17         YIELD(SECT)=YIELD(SECT)+VALUE(PIECE(NPIECE,1),PIECE(NPIECE,2))     CUTUP
18       IF(.NOT.LAST)GO TO 90                                                CUTUP
19         IF((LY.EQ.BDLY))GO TO 85                                           CUTUP
20         NCUTS=NCUTS+1                                                      CUTUP
21   77    IF(NCUTS.GT.200)THEN                                               CUTUP
22         WRITE(6,80)NBOARD                                                  CUTUP
23         BAD=.TRUE.                                                         CUTUP
24       ELSE                                                                 CUTUP
25   80    FORMAT(10X,'TOO MANY SAWCUTS',A5)                                  CUTUP
26         IF(NCUTS.GT.100)BAD=.TRUE.                                         CUTUP
27         SAWCUT(NCUTS,1)=LX                                                 CUTUP
28         SAWCUT(NCUTS,2)=LY-1                                               CUTUP
29         SAWCUT(NCUTS,3)=UX                                                 CUTUP
30         SAWCUT(NCUTS,4)=LY                                                 CUTUP
31         ENDIF                                                              CUTUP
32   85    IF(BDUY.EQ.UY)GO TO 90                                             CUTUP
33         NCUTS=NCUTS+1                                                      CUTUP
34         IF(NCUTS.GT.200)THEN                                               CUTUP
35         WRITE(6,80)NBOARD                                                  CUTUP
36         BAD=.TRUE.                                                         CUTUP
37         ELSE                                                               CUTUP
38         SAWCUT(NCUTS,1)=LX                                                 CUTUP
39         SAWCUT(NCUTS,2)=UY                                                 CUTUP
40         SAWCUT(NCUTS,3)=UX                                                 CUTUP
41         SAWCUT(NCUTS,4)=UY+1                                               CUTUP
42         ENDIF                                                              CUTUP
43   90 RETURN                                                                CUTUP
44       END                                                                  CUTUP
```

```
                    ***  STATEMENT NUMBERS  ***

  50        10   *15
  75        11   *12
  77       *21
  80        22   *25   35
  85        19   *32
  90        18    32   *43

                    ***  VARIABLES  ***

BAD          7    6   *13   *23   *26   *36
BDLX         6
BDLY         6   19
BDUX         6
BDUY         6   32
CUTUP        1
LAST         3    7    18
LX           1   15    27    38
LY           1   16    19    28    30
NBOARD       6   22    35
NCUTS        7  *20    21    26    27    28    29    30   *33   34   38   39
            40   41
NPIECE       7   *9    10    15    16    17
PIECE        7  *15   *16    17
SAWCUT       7  *27   *28   *29   *30   *30   *39   *40   *41
SECT         1   11    17
UX           1   15    29    40
UY           1   10    32    39    41
VALUE        4   17
YIELD        4    8   *17
```

Subroutine HOLD

```
 1        SUBROUTINE HOLD(NEWLX,NEWUX,YIELD,HERE)                    HOLD
 2 C ***  STORE RESULTS OF SECTIONS                                  HOLD
 3        IMPLICIT INTEGER(A-Z)                                      HOLD
 4        CHARACTER*5 NBOARD                                         HOLD
 5        LOGICAL HERE,BAD                                           HOLD
 6        DIMENSION INDEX(900,2),YSTORE(900)                         HOLD
 7        COMMON ALL NBOARD,BAD,BDLX,BDLY,BDUX,BDUY                  HOLD
 8        DATA LX 1 ,UX 2                                            HOLD
 9        HERE=.FALSE.                                               HOLD
10        IF(NCOMB.LE.0)RETURN                                       HOLD
11        DO 25 I=1,NCOMB                                            HOLD
12          IF(HERE)GO TO 25                                         HOLD
13          IF(INDEX(I,LX).NE.NEWLX)GO TO 25                         HOLD
14          IF(INDEX(I,UX).NE.NEWUX)GO TO 25                         HOLD
15          HERE=.TRUE.                                              HOLD
16          YIELD=YSTORE(I)                                          HOLD
17     25 CONTINUE                                                   HOLD
18        RETURN                                                     HOLD
19                                                                   HOLD
20        ENTRY INTL                                                 HOLD
21        NCOMB=0                                                    HOLD
22        RETURN                                                     HOLD
23                                                                   HOLD
24        ENTRY REMEM(YLOW,YHI,YLD)                                  HOLD
25        NCOMB=NCOMB+1                                              HOLD
26        IF(NCOMB.GT.900)THEN                                       HOLD
27          WRITE(6,50)NBOARD                                        HOLD
28     50   FORMAT(1X,25(1H*),'TOO MANY COMBINATION IN',A5,25(1H*))  HOLD
29          BAD=.TRUE.                                               HOLD
30        ELSE                                                       HOLD
31          INDEX(NCOMB,LX)=YLOW                                     HOLD
32          INDEX(NCOMB,UX)=YHI                                      HOLD
33          YSTORE(NCOMB)=YLD                                        HOLD
34        ENDIF                                                      HOLD
35        RETURN                                                     HOLD
36        END                                                        HOLD
```

```
                  ***  STATEMENT NUMBERS  ***

  25       11    12    13    14   *17
  50       27   *28

                  ***  VARIABLES  ***

BAD         5     7   *29
BDLX        7
BDLY        7
BDUX        7
BDUY        7
HERE        1     5    *9    12   *15
HOLD        1
I         *11    13    14    16
INDEX       6    13    14   *31   *32
INTL       20
LX         *8    13    31
NBOARD      7    27
NCOMB      10    11   *21   *25    26    31    32    33
NEWLX       1    13
NEWUX       1    14
REMEM      24
UX         *8    14    32
YHI        24    32
YLOW       24    31
YIELD       1   *16
YLD        24    33
YSTORE      6    16
```

Subroutine WFIND

```
1          SUBROUTINE WFIND(CLRKNT,DCOM,DN,LEN)                              WFIND
2          IMPLICIT INTEGER(A-Z)                                             WFIND
3            LOGICAL DONE,OK,BAD                                             WFIND
4          [illegible]                                                       WFIND
5          [illegible]                                                       WFIND
6          [illegible] NDONE(10),CLRWID(10,100),                             WFIND
7        *              [illegible](10,100,2)                                WFIND
8             COMMON [illegible] BAD,BDLX,BDLY,BDUX,BDUY                     WFIND
9             [illegible] LENGTH(10),NLEN,WIDTH(10),NWID                     WFIND
10         [illegible]                                                       WFIND
11         [illegible]                                                       WFIND
12    ***  [illegible] HAS BEEN CALCULATED ALREADY                           WFIND
13         [illegible]                                                       WFIND
14         DO 100 I=1,NLEN                                                   WFIND
15           IF([illegible])GO TO 100                                        WFIND
16           IF([illegible]LENGTH(I))ILEN=I                                  WFIND
17    100  CONTINUE                                                          WFIND
18         IF([illegible].LE.0)THEN                                          WFIND
19           BAD=.TRUE.                                                      WFIND
20           WRITE(6,110)NBOARD                                              WFIND
21    110    FORMAT([illegible] MESSED UP IN WFIND. BOARD NUMBER ',A5)       WFIND
22         ENDIF                                                             WFIND
23         IF(BAD)RETURN                                                     WFIND
24         IF(NDONE(ILEN).EQ.0)GO TO 5                                       WFIND
25         N=0                                                               WFIND
26         DO 130 I=1,NDONE(ILEN)                                            WFIND
27           IF(N.GT.0)GO TO 130                                             WFIND
28           IF(CLRKNT.EQ.CLRWID(ILEN,I))N=I                                 WFIND
29    130  CONTINUE                                                          WFIND
30         IF(N.EQ.0)GO TO 5                                                 WFIND
31         DN=CLRCOM(ILEN,N,1)                                               WFIND
32         IF(DN.EQ.0)RETURN                                                 WFIND
33         DO 150 I=1,DN                                                     WFIND
34           DCOM(I)=CLRCOM(ILEN,N,I+1)                                      WFIND
35    150  CONTINUE                                                          WFIND
36         RETURN                                                            WFIND
37      5  MAXV=0                                                            WFIND
38         MAXW=(CLRKNT+1)/(WIDTH(1)+1)                                      WFIND
39         DO 10 I=1,MAXW                                                    WFIND
40     10    WCOM(I,ACTIVE)=1                                                WFIND
41         NW(ACTIVE)=MAXW                                                   WFIND
42     20  V=0                                                               WFIND
43         IF(NW(ACTIVE).LE.0)GO TO 35                                       WFIND
44         DO 25 I=1,NW(ACTIVE)                                              WFIND
45           V=V+VALUE(LENGTH(ILEN),WIDTH(WCOM(I,ACTIVE)))                   WFIND
46     25  CONTINUE                                                          WFIND
47         IF([illegible].LT.0)GO TO 35                                      WFIND
48         IF([illegible])GO TO 35                                           WFIND
49         [illegible]NW(ACTIVE)                                             WFIND
50         DO 30 I=1,[illegible]                                             WFIND
51     30    WCOM(I,MAX)=WCOM(I,ACTIVE)                                      WFIND
52         [illegible]                                                       WFIND
53     35  OK=.FALSE.                                                        WFIND
54         DO 40 I=MAXW,1,-1                                                 WFIND
55           IF(OK)GO TO 40                                                  WFIND
56           WCOM(I,ACTIVE)=WCOM(I,ACTIVE)+1                                 WFIND
57           IF(WCOM(I,ACTIVE).GT.NWID)THEN                                  WFIND
58             WCOM(I,ACTIVE)=1                                              WFIND
59             IF(I.EQ.1)DONE=.TRUE.                                         WFIND
60           ELSE                                                            WFIND
61             SUM=0                                                         WFIND
62             OK=.TRUE.                                                     WFIND
63             NW(ACTIVE)=0                                                  WFIND
64             DO 38 K=1,MAXW                                                WFIND
65           SUM=SUM+WIDTH(WCOM(K,ACTIVE))                                   WFIND
66           IF(K.GT.1)SUM=SUM+1                                             WFIND
67               IF(SUM.LE.CLRKNT)NW(ACTIVE)=NW(ACTIVE)+1                    WFIND
68     38      CONTINUE                                                      WFIND
69           ENDIF                                                           WFIND
70     40  CONTINUE                                                          WFIND
71         MAXW=NW(ACTIVE)                                                   WFIND
72         IF(MAXW.EQ.0)DONE=.TRUE.                                          WFIND
73         IF(.NOT.DONE)GO TO 20                                             WFIND
74         DN=NW(MAX)                                                        WFIND
75         NDONE(ILEN)=NDONE(ILEN)+1                                         WFIND
76         IF(NDONE(ILEN).GT.100)THEN                                        WFIND
77           WRITE(6,45)NBOARD                                               WFIND
78     45    FORMAT(25X,'ATTEMPT TO STORE TOO MANY WCOMS IN ',A5)            WFIND
79           BAD=.TRUE.                                                      WFIND
80         ENDIF                                                             WFIND
81         IF(BAD)RETURN                                                     WFIND
82         CLRWID(ILEN,NDONE(ILEN))=CLRKNT                                   WFIND
83         CLRCOM(ILEN,NDONE(ILEN),1)=NW(MAX)                                WFIND
84         IF(DN.EQ.0)RETURN                                                 WFIND
85         DO 50 I=1,DN                                                      WFIND
86           DCOM(I)=WCOM(I,MAX)                                             WFIND
87           CLRCOM(ILEN,NDONE(ILEN),I+1)=WCOM(I,MAX)                        WFIND
88     50  CONTINUE                                                          WFIND
89         RETURN                                                            WFIND
90         ENTRY WINTL                                                       WFIND
91         DO 200 I=1,10                                                     WFIND
92    200  NDONE(I)=0                                                        WFIND
93         RETURN                                                            WFIND
94         END                                                               WFIND
```

*** STATEMENT NUMBERS ***

5	24	30	*37	
10	39	*40		
20	*42	73		
25	44	*46		
30	50	*51		
35	43	47	48	*53
38	64	*68		
40	54	55	*70	
45	77	*78		
50	85	*88		
100	14	15	*17	
110	20	*21		
130	26	27	*29	
150	33	*35		
200	91	*92		

*** VARIABLES ***

ACTIVE	*10	40	41	43	44	45	49	51	56	57	58	63
	65	67	71									
BAD	3	8	*19	23	*79	81						
BDLX	8											
BDLY	8											
BDUX	8											
BDUY	8											
CLRCOM	6	31	34	*83	*87							
CLRKNT	1	28	38	67	82							
CLRWID	6	28	*82									
DCOM	1	6	*34	*86								
DN	1	*31	32	33	*74	84	85					
DONE	3	*11	*59	*72	73							
I	*14	16	*26	28	*33	34	*39	40	*44	45	*50	51
	*54	56	57	58	59	*85	86	87	*91	92		
ILEN	*13	15	*16	18	24	26	28	31	34	45	75	76
	82	83	87									
K	*64	65	66									

LEN	1	16									
LENGTH	9	16	45								
[illegible]	*10	49	50	51	74	83	86	87			
[illegible] V	4	*37	48	*52							
[illegible] W	*38	34	41	54	64	*71	72				
N	*25	27	*28	30	31	34					
NBOARD	2	20	22								
NDONE	[illegible]	24	26	*25	26	82	83	87	*92		
NLEN	3	14									
NO	6	*41	43	44	*45	50	*63	*67	71	74	83
NUNIT	4	57									
[illegible]	3	*53	55	*62							
SUM	*61	*65	*66	67							
[illegible]	4	*42	*45	47	48	52					
VALUE	4	45									
WORD	6	*40	45	*51	*56	57	*58	65	86	87	
WPINC	1										
WIDTH	3	38	45	65							
WINTC	90										

Subroutine TSTORE

```
      SUBROUTINE TSTORE(LX,LY,UX,UY,TRIAL,SECT)
      IMPLICIT INTEGER(A-U)
      LOGICAL BAD,LAST,HAVEIT
      CHARACTER*5 NBOARD
      DIMENSION NP(2),V(2),COORD(50,4,2)
      COMMON ALL NBOARD,BAD,BDL,BIL,EDGE,BIG
      COMMON NEW NOUT,SAVOUT(200,4),LAST,PIECE(100,2),NPIECE
C *** CHECK IF COORDINATES ARE ALREADY STORED
      HAVEIT=.FALSE.
      IF(NP(TRIAL).EQ.0)GO TO 8
      DO 5 I=1,NP(TRIAL)
        IF(HAVEIT)GO TO 5
        IF(COORD(I,1,TRIAL).NE.LX)GO TO 5
        IF(COORD(I,2,TRIAL).NE.LY)GO TO 5
        IF(COORD(I,3,TRIAL).NE.UX)GO TO 5
        IF(COORD(I,4,TRIAL).EQ.UY)HAVEIT=.TRUE.
    5 CONTINUE
      IF(HAVEIT)RETURN
    8 NP(TRIAL)=NP(TRIAL)+1
      IF(NP(TRIAL).LE.50)GO TO 20
      BAD=.TRUE.
      WRITE(6,10)NBOARD
   10 FORMAT(1X,'[illegible] OVERFLOW IN TSTORE',A5,[illegible])
      RETURN
   20 V(TRIAL)=V(TRIAL)+VALUE([illegible])
      COORD(NP(TRIAL),1,TRIAL)=LX
      COORD(NP(TRIAL),2,TRIAL)=LY
      COORD(NP(TRIAL),3,TRIAL)=UX
      COORD(NP(TRIAL),4,TRIAL)=UY
      RETURN
      ENTRY TINTL
      DO 30 I=1,2
        NP(I)=0
        V(I)=0.
   30 CONTINUE
      RETURN
      ENTRY RETREV
      BEST=0
      IF((NP(1)+NP(2)).EQ.0)RETURN
      IF((V(1)-V(2)).GT..0001)BEST=1
      IF((V(2)-V(1)).GT..0001)BEST=2
      IF((BEST.EQ.0).AND.(V(1).LT..0001))RETURN
      IF(BEST.EQ.0)BEST=1
      DO [illegible] I=1,NP(BEST)
        IF(BAD)GO TO 50
        CALL [illegible](COORD(I,1,BEST),COORD(I,2,BEST),COORD(I,3,BEST),
     *                COORD(I,4,BEST),SECT)
          TIME=1
          IF(COORD(I,1,BEST).EQ.[illegible])GO TO 43
   35   NOUTS=NOUTS+1
        IF(NOUTS.LE.200)GO TO 42
          WRITE(6,40)NBOARD
   40     FORMAT([illegible] TOO MANY [illegible] IN [illegible])
          [illegible]=.TRUE.
        GO TO 50
   42       SAVOUT(NOUTS,[illegible])=COORD(I,[illegible],BEST)[illegible]
            SAVOUT(NOUTS,[illegible])=COORD(I,4,BEST)+1
          IF(TIME.[illegible])GO TO 41
            [illegible]
            [illegible]
          TIME=TIME+1
          IF(TIME.[illegible])GO TO [illegible]
   47     [illegible]
            [illegible]
   41     [illegible]
            [illegible]
   [illegible] CONTINUE
      RETURN
      END
```

```
***  STATEMENT NUMBERS  ***

   5      11    12    13    14    15   *17
   8      10   *19
  10      22   *23
  20      20   *25
  30      32   *35
  35     *50    64
  40      52   *53
  42      51   *56
  43      49   *63
  44      58   *65
  50      44    45    55    62    63   *67


***  VARIABLES  ***

BAD        3     6   *21    45   *54
BDLX       6    49
BDLY       6
BDUX       6    63
BDUY       6
BEST     *38   *40   *41    42   *43    44    46    49    56    57    59    60
          63    65    66
COORD      5    13    14    15    16   *26   *27   *28   *29    46    49    56
          57    59    60    63    65    66
CUTUP     46
HAVEIT     3    *9    12   *16    18
I        *11    13    14    15    16   *32    33    34   *44    46    49    56
          57    59    60    63    65    66
LAST       3     7
LX         1    13    25    26
LY         1    14    25    27
NRCORD     [illegible]    22    52
[illegible]    [illegible]   [illegible]    51    56    57    59    60    65    66
[illegible]    [illegible]    10    11   *13    20    26    27    28    29   *33    39    44
[illegible]
[illegible]
[illegible]
[illegible]    [illegible]   [illegible]   *57   *59   *62   *65   *66
[illegible]
[illegible]   [illegible]   [illegible]   *61    62
[illegible]
[illegible]    [illegible]    10    11    13    14    15    16    19    20    25    26    27
[illegible]
[illegible]
[illegible]    [illegible]    25    23
[illegible]    [illegible]    25    24
               *[illegible]   *34    40    41    42
[illegible]
```

Function VALUE

```
      FUNCTION VALUE(X,Y)                                          VALUE
      IMPLICIT INTEGER(A-Z)                                        VALUE
      REAL INDEX,VALUE                                             VALUE
      COMMON /MV/ INDEX(4,8),VLEN,VWID,LINDEX(8),WINDEX(4),NV      VALUE
      COMMON /MSR/ LENGTH(10),NLEN,WIDTH(10),NWID                  VALUE
      IF(NV.NE.0)GO TO 5                                           VALUE
      VALUE=FLOAT(X*Y)/(16.*144.)                                  VALUE
      RETURN                                                       VALUE
    5 IW=0                                                         VALUE
      DO 10 I=1,VWID                                               VALUE
        IF(IW.GT.0)GO TO 10                                        VALUE
        IF(Y.LE.WINDEX(I))IW=I                                     VALUE
   10 CONTINUE                                                     VALUE
      IF(IW.EQ.0)IW=VWID                                           VALUE
      IL=0                                                         VALUE
      DO 15 I=1,VLEN                                               VALUE
        IF(IL.GT.0)GO TO 15                                        VALUE
        IF(X.LE.LINDEX(I))IL=I                                     VALUE
   15 CONTINUE                                                     VALUE
      IF(IL.EQ.0)IL=VLEN                                           VALUE
      VALUE=INDEX(IW,IL)*(X*Y)/(16*144)                            VALUE
      RETURN                                                       VALUE
      END                                                          VALUE


***  STATEMENT NUMBERS  ***

   5       6    *9
  10      10    11   *13
  15      16    17   *19


***  VARIABLES  ***

FLOAT      7
I        *10    12   *16    18
IL       *15    17   *18   *20    21
INDEX      3    *4    21
IW        *9    11   *12   *14    21
LENGTH     5
LINDEX     4    18
NLEN       5
NV         4     6
NWID       5
VALUE      1     3    *7   *21
VLEN       4    16    20
VWID       4    10    14
WIDTH      5
WINDEX     4    12
X          1     7    18    21
Y          1     7    12    21
```

Appendix B: Use and Derivation of Value Weighting Table

The best decision on crosscutting a board is dependent not only upon what clear areas exist within the board but what types of cuttings are required for the end products. The highest yield of total surface area of cuttings may be attained by sawing the boards into short, narrow cuttings; however, if each of these cuttings require additional processing such as edge gluing or fingerjointing, the value of the decision is diminished by the additional steps required between initial crosscutting and a finished end product. The desirability as well as availability of types of cuttings must be considered in the decision. Cuttings which are easy to get, such as short and narrow cuttings, take on a relatively low value when weighting the value of cutting dimensions. Cuttings which are more difficult to recover such as long, wide cuttings take on a high value. Also, cuttings which have high demand may take on relatively high values. In summary, cuttings of different dimensions are available in different proportions and are required in different proportions. Since these proportions may not be the same, some weighting as to desirability of cuttings should be considered.

The value weighting table used by CROMAX is a matrix dimensioned four rows by eight columns. The rows correspond to upper limits of rip widths while the columns correspond to upper limits of cutting lengths. Each cell specifies the weighting value for a cutting of given dimensions. So if the data in table 5 were used, the weighting value of 0.921 would be assigned to any cutting with a length greater than 35.0 but less than 42.0 inches and a width greater than 2.75 but less than 3.75 inches. Thus, for a cutting of dimension 3.75 × 40 inches, and given value weighting from table 5, CROMAX would calculate the value:

$$\text{Value} = \frac{(\text{weighting factor}) \times (\text{length of cutting}) \times (\text{width of cutting})}{144}$$

so substituting a cutting of dimension 3.75 × 40.0 inches and table 5 factor

$$\text{Value} = \frac{0.921 \times 40.0 \times 3.75}{144}$$

$$\text{Value} = 0.959$$

This value does not represent the dollar value of the cutting but rather the weighting factor to be used in comparisons with the weighting factor of other cuttings. The quantity is divided by 144 in order to make a conversion to square feet for convenience.

Use of Value Weighting Table

The primary use of the value weighting table is to place a weighting factor on the desirability of a cutting. Without such a factor the program would be unable to discriminate between alternative decisions when surface areas were equal. For example, if surface area only of cuttings is considered, four 1.75- by 9.00-inch cuttings would have the same desirability as one 1.75- by 36.00 inches. A greater weighting factor on the 1.75- by 36.00-inch cutting would ensure that it would be chosen over the smaller cuttings. Using table 5, the sum of the values of the four 1.75- by 9.00-inch cuttings is 0.346, while the value of a 1.75- by 36.00-inch cutting is 0.392.

The value weighting table can also be used to insure recovery of certain size cuttings. This could be done by placing a very high value on the highly desirable dimensions while placing a very low or zero value on the other sizes. A weighting value of zero would still yield allowable cuttings since CROMAX never discards allowable cuttings, but these would be salvage cuttings saved intead of wasting clear wood.

Derivation of a Value Weighting Table

Developing a value weighting table can be a major analysis in itself. The weighting factors are a function of the type of processes used in the mill operation (i.e., edge gluing or no edge gluing), the demand for cuttings of various dimensions, and the availability of the cuttings within the grade mix. Since a purpose of running CROMAX is to determine the yield of cuttings within a lumber grade, it may seem recursive to use the same component in developing the weighting table. However, some experimental idea of how hard cuttings are to get should be conveyed within the table; if all cuttings occur at similar frequency, this factor may not be needed.

In developing the value weighting table, let W be the 4 × 8 matrix of weighting factors where W_{ij} is the weighting factor for a cutting whose width is between width_{i-1} to width_i and whose length is between length_{j-1} to length_j ($i \leq 4$ and $j \leq 8$). If i or j is 1, the lower bound is zero.

D_{ij} is the demand for cutting $_{ij}$. This may just be the number of pieces of that dimension needed (minus discards) for production. However, if edge gluing or fingerjointing is used, the value of the potential demand for the cutting being used in this process should be included.

H_{ij} is the "difficulty" rating for the cutting—how "hard to get" the piece actually is in comparison to its demand. This could be a proportion rating where 1.0 would equal the most difficult piece or could be a general 1 to 10 scaling of difficulty. About any consistent schema would do.

Putting this information together, a reasonable equation for weighting factor would be:

$$W_{ij} = H_{ij} \times \frac{D_{ij}}{\sum_{k=1}^{4} \sum_{m=1}^{6} D_{km}}$$

Other alternative ways of developing a value weighting table exist. It would be possible also to develop a table based upon the actual dollar value of a finished end product and the cost of the components within the product. Such a method would give at least as good a result as the above method. Another method would be to translate the present cutting bill into a value weighting table and then use CROMAX to give feedback as to where surplus or deficiency exist between CROMAX projections and the cutting bill.

☆U.S. GOVERNMENT PRINTING OFFICE: 1983 654 025 4013

2.5-39-8/83

U.S. Forest Products Laboratory.

CROMAX—A Crosscut-First Computer Simulation Program
to Determine Cutting Yields, by Pamela J. Giese and
Jeanne D. Danielson, Madison, Wis., FPL 1983.
39 p. (USDA For. Serv. Gen. Tech. Rep. FPL-38)

The program CROMAX was designed to simulate crosscut-first, then rip operations as commonly practiced in furniture manufacture. It also calculates cutting yields from individual boards based on board size and defect location.

Keywords: Crosscut, rip, cutting yields, defect location, lumber grades

END

FILMED

11-83

DTIC