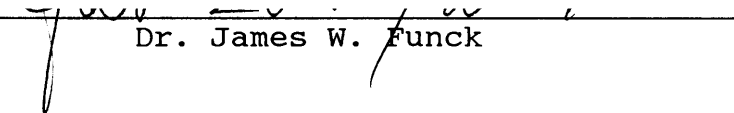


AN ABSTRACT OF THE THESIS OF

Yimin Zeng for the degree of Master of Science in  
Forest Products presented on May 3, 1991.

Title: Log Breakdown Using Dynamic Programming  
and 3-D Log Shape

Signature redacted for privacy.

Abstract approved:   
Dr. James W. Funck

A log breakdown model, SAW3D, was developed to optimize lumber recovery from logs of any shape in 3-dimensional space. The program uses a polygonal cross section representation of logs; three nested dynamic programming (DP) algorithms to optimize log breakdown, flitch and cant edging, and trimming operations; and a simple heuristic search method to find the best position combinations.

After receiving the log scanning data, the program mathematically rotates the log and skews it horizontally. Then the log is sawn into slabs, flitches or cants using the first level DP algorithm. For each piece cut from the log, the program orients it and then edges it into untrimmed lumber using the second level DP algorithm. After that, the program trims all untrimmed lumber to finished dimensions using the third level DP algorithm. This process is repeated until all possible cutting

patterns are searched and all positions/orientations are compared. Upon completion, the program outputs the optimum value of the log; optimum value of each flitch/cant; corresponding sawing, edging and trimming patterns; optimum position of the log; and optimum position of each flitch/cant.

Mathematically generated logs with ellipsoid, horn-down, and 3-D, S-twisted shapes were sawn using SAW3D to analyze how factors such as log, cant, and flitch orientations; sawing intervals; scanning data density; and log size and shape affect both the value and volume of lumber recovered.

Log Breakdown Using Dynamic Programming  
and 3-D Log Shape

by

Yimin Zeng

A THESIS

submitted to

Oregon State University

in partial fulfillment of  
the requirements for the  
degree of

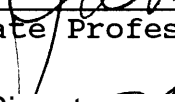
Master of Science

Completed May 3, 1991


Commencement June 1991

APPROVED:

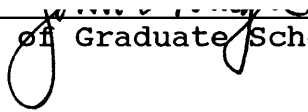
Signature redacted for privacy.

 Associate Professor of Forest Products in charge of major

Signature redacted for privacy.

 Head of Department of Forest Products

Signature redacted for privacy.

 Dean of Graduate School

Date thesis is presented May 3, 1991

Typed by Yimin Zeng for Yimin Zeng

## ACKNOWLEDGEMENTS

I wish to express my deepest gratitude and appreciation to my major professor, Dr. James W. Funck, for his guidance, encouragement and support throughout the course of my graduate study. His tremendous patience and judicious suggestions made possible the successful completion of this thesis.

I wish to thank Dr. Charles C. Brunner and Dr. David A. Butler for contributing their time and expertise while serving on my graduate committee.

My special thanks go to my friends Mr. Johannes B. Forrer for his valuable suggestions and help in many areas; and Mr. Guangchao Zhang and Mr. Jeff Gaber for sharing their computer programming experiences.

I am deeply indebted to my wife, Yuan Zhong, for her love, understanding, encouragement and help in all aspects of my daily life and graduate studies.

I am deeply indebted to my parents for their encouragement and support for my continuous education.

Last, but not least, I would like to thank The Research Institute of Wood Industry, Chinese Academy of Forestry and Ministry of Forestry. Without their agreement, I would not be able to stay in U.S.A. for my continuous education.

## TABLE OF CONTENTS

<b>CHAPTER 1</b>	<b>INTRODUCTION . . . . .</b>	<b>1</b>
<b>CHAPTER 2</b>	<b>OBJECTIVES OF THE PROJECT . . . . .</b>	<b>5</b>
<b>CHAPTER 3</b>	<b>LITERATURE REVIEW . . . . .</b>	<b>7</b>
3.1	Simulation Models . . . . .	8
3.2	Models Using Dynamic Programming . . . . .	16
3.3	Models Developed by Industry . . . . .	18
<b>CHAPTER 4</b>	<b>MODEL DEVELOPMENT . . . . .</b>	<b>30</b>
4.1	Description of the System Procedure . . . . .	31
4.1.1	The Overall Procedure . . . . .	31
4.1.2	Level 1: The Sawing Level . . . . .	33
4.1.3	Level 2: The Edging Level . . . . .	36
4.1.4	Level 3: The Trimming Level . . . . .	37
4.2	Log Modeling . . . . .	37
4.2.1	Cylinder and Truncated-Cone Models . . . . .	38
4.2.2	Cross-Section Models . . . . .	38
4.2.3	Surface Fitting Models . . . . .	39
4.2.4	Log Shape Model Used in This Program . . . . .	40
4.3	Sawing Pattern Optimization . . . . .	42
4.3.1	Log Positioning . . . . .	43
4.3.2	Description of the DP Algorithm . . . . .	51
4.3.3	Mathematical Definition and Formulation . . . . .	59
4.4	Edging Pattern Optimization . . . . .	61
4.4.1	Determining the Two Faces of a Piece . . . . .	61
4.4.2	Pitching the Piece . . . . .	66
4.4.3	Mathematical Formulation for the Edging Problem . . . . .	69
4.5	Trimming Pattern Optimization . . . . .	71
4.5.1	Locating Trimming Zones . . . . .	71
4.5.2	Mathematical Formulation for the Trimming Problem . . . . .	74
4.6	Software Implementation . . . . .	76
4.6.1	Structure of the Program . . . . .	76
4.6.2	Required Information . . . . .	79
4.6.3	Input and Output . . . . .	82
<b>CHAPTER 5</b>	<b>COMPUTATIONAL RESULTS . . . . .</b>	<b>85</b>
5.1	Effect of Log Rotation . . . . .	88
5.1.1	General Observations On All Shapes, Sizes, and Variables . . . . .	88
5.1.2	Effect of Rotation Increment . . . . .	91
5.2	Effect of Log Rotation and Skew . . . . .	93
5.3	Effect of Rotation and Pitch . . . . .	94
5.4	Effect of Rotation, Skew, and Pitch . . . . .	96
5.5	Effect of Positioning On Computing Time . . . . .	97
5.6	Effect of Scanning Data Density . . . . .	99

5.6.1	Effect of Number of Scanning Points At Every Cross Section . . . . .	99
5.6.2	Effect of Interval Between Cross Sections . . . . .	99
5.7	SAW3D vs BOF . . . . .	100
5.8	Effect of Sawing and Edging Increments . .	101
<b>CHAPTER 6</b>	<b>CONCLUSIONS AND FURTHER IMPROVEMENTS . . .</b>	<b>105</b>
<b>Bibliography</b>	. . . . .	<b>108</b>
<b>Appendix A.</b>	<b>The output from sawing a horn-down shaped log 20 inches in diameter and 10 feet long . . . . .</b>	<b>113</b>

## LIST OF FIGURES

Figure 3.1	Procedure followed to determine the initial opening face and subsequent trial opening faces for each log diameter (Hallock et al., 1971). . . .	14
Figure 4.1.	Log and flitch orientation. . . . .	32
Figure 4.2.	Flowchart of log sawing level. A and A' are the links to Figure 4.3. . . . .	34
Figure 4.3.	Flowchart of edging and trimming levels. A and A' are the links to Figure 4.2. . . . .	35
Figure 4.4.	A cross section of the polygonal cross-section model . . . . .	41
Figure 4.5.	A log represented by the polygonal cross-section model. . . . .	42
Figure 4.6.	Sawing level processes. . . . .	44
Figure 4.7.	Projection of the log onto the X-Z plane. . . . .	49
Figure 4.8.	Determination of the skewing range . . .	49
Figure 4.9.	Saw placements on cross sections of a log as seen from one end of the log . .	52
Figure 4.10.	A representation of potential sawing decisions for nodes 0, 1, and 8 . . . .	53
Figure 4.11.	Optimum solution for an illustrative example. . . . .	57
Figure 4.12.	Flowchart of the edging level procedure. . . . .	62
Figure 4.13.	Intersection of a sawing plane with a cross section. . . . .	63
Figure 4.14.	Merging two faces into one for edging .	65
Figure 4.15.	Flowchart of the trimming procedure . .	72
Figure 4.16.	Locating trimming zones . . . . .	73
Figure 4.17.	Hierarchy chart of the program. . . . .	78



Figure 4.18.	A plot of the optimum sawing sequence for a horn-down shaped log 10 inches in diameter and 10 feet long. . . . .	83
Figure 4.19.	Plots of the optimum edging and trimming patterns for the five flitches sawn from the log in Figure 4.18. . . . .	84
Figure 5.1.	Three log shapes tested. . . . .	86
Figure 5.2.	Volume and value effects of log rotation angle on a 3-dimensional, S-twisted shape log 10 inches in diameter and 20 feet in length (s12). . . . .	89
Figure 5.3.	Effects of log rotation on value using the percent increase from the average value at all rotation angles to the optimum value for 12 different logs . . . . .	90
Figure 5.4.	Effects of log rotation on value using the percent increase from the minimum value to the optimum value for 12 different logs. . . . .	91
Figure 5.5.	Effects of rotation angle increment on log s12 using rotation increments of 15 and 5 degrees. . . . .	92
Figure 5.6.	Effects of log rotation angle increment on log s12 using a rotation increment of 24 degrees. . . . .	93
Figure 5.7.	Value increases resulting from using the log skewing operation in addition to log rotation. . . . .	95
Figure 5.8.	Value increases resulting from using the cant/board orientation operation (pitching) in addition to log rotation. . . . .	96
Figure 5.9.	Value increases resulting from using both skewing and pitching operations in addition to log rotation. . . . .	97
Figure 5.10.	Comparison of computing time consumed by various positioning operations . . . . .	98
Figure 5.11.	Effects of the number of log scanning points at each cross section on an S-twisted log 20 inches in diameter and 10 feet long, using 24 and 72 points . . . . .	100

Figure 5.12. A comparison of SAW3D and BOF (Best Opening Face) results on four horn-down shaped logs. . . . .	102
Figure 5.13. Value increases resulting from using a 0.025 inch sawing increment compared to a 0.125 inch increment. . . . .	103
Figure 5.14. Value increases resulting from using a 0.025 inch edging increment compared to a 0.125 inch increment. . . . .	104

## LIST OF TABLES

Table 5.1.	Various combinations of log shapes and sizes used in this study. . . . .	87
Table 5.2.	The base data used to test the model. . .	88

# LOG BREAKDOWN USING DYNAMIC PROGRAMMING AND 3-D LOG SHAPE

## CHAPTER 1 INTRODUCTION

Improving lumber recovery and profitability have always been of great interest to the lumber industry. It is even more essential to use timber resources efficiently in order to meet increasing demands for timber products when those resources are limited. In sawmills, converting logs into finished lumber involves several interrelated operations with the main ones being scanning, transporting, bucking, positioning, sawing, edging and trimming. Improving any one of these can improve recovery. The three portions with greatest potential for increase, however, are in optimizing the sawing, edging, and trimming patterns.

The process of converting logs into lumber is complicated by log geometry, log quality, saw kerf, sawing variation, sawing method, edging method and the mix of possible products. Efforts to improve lumber yield by optimizing sawing patterns and edging methods have been ongoing since the late 1960's. A variety of approaches, which can be classified as computer simulation and mathematical programming methods, have been applied in an attempt to find optimum solutions.

Before the 1980's, the area was almost exclusively dominated by computer simulation methods. During those

years the major achievement was BOF, short for the Best Opening Face System. The program is perhaps the most widely publicized and adopted simulation model. After 1980, some researchers realized that dynamic programming (DP) would be worth exploring since it has some advantages over simulation methods when considering multi-dimension products. However, until recently only a few came up with programs applying the dynamic programming method, but all required greatly simplified log shape models.

The previous research indicates that lumber yield can be increased significantly by choosing proper saw placements and cutting patterns. Today, many computer optimization programs are being successfully used in the industry. However, there is still room to make improvements. These include:

- 1. Real-shape log model.** It is apparent that a prerequisite for obtaining an optimum breakdown solution is to model logs as close to their actual shapes as possible. Up to now, however, most programs use cylinders or truncated cones as log models because they are easy to program and calculate. Unfortunately, the vast majority of logs are not shaped like cylinders or truncated cones. Many logs have eccentric cross sections and sweep or crook. Therefore, any algorithm based on these simplified log models is not able to find real optimum breakdown patterns.

- 2. Orientation of logs and pieces.** If log geometry

and distribution of defects on logs are complex, to achieve an optimum solution there should be a best orientation for any log entering a headrig or any piece entering an edger or trimmer. As some research results have revealed, both grade and yield of lumber can be increased significantly by proper orientation of logs with respect to defects before the first cut is made. Thus, the optimum yield will only be achieved when three-axis control is achieved. Most existing programs, however, do not determine the optimum position of logs or pieces. There are many reasons for this. For any program based on the cylinder or truncated cone log models, log rotation becomes unnecessary. Other reasons include long computing time and the lack of high recovery machinery being able to position logs precisely. These can result in some recovery losses for those irregularly-shaped logs.

**3. 3-D cutting problem.** Roughly speaking, when only the sawing problem is considered, it is called a one-dimensional cutting problem. Sawing and edging together are called a 2-D cutting problem. Sawing, edging and trimming together are called a 3-D cutting problem. Decisions made in the sawing, edging and trimming operations are interrelated. Any decision made in an upper level determines dimensions of pieces in the lower level and the decision made in the lower level determines the value or volume returned to the upper level. Most existing

programs concentrate on one-D or 2-D cutting problems. Among the computer simulation programs, some do include the 3-D cutting problem in their models, such as BOF, but with very limited edging or trimming methods available. For the models using DP, none is able to give an optimum solution for the 3-D problem.

**4. Defects in logs.** Research results have shown that the distribution of defects in logs has substantial effects on both the grades and yields of lumber. Most existing programs, however, give the optimum solution but with the assumption of perfect logs or very simple defect patterns. Therefore, the solution given will not be optimal when logs are not perfect.

Many existing programs have to a limited extent solved these four problems in one aspect or another. However, a program that can solve all four problems without making over simplified assumptions should be able to squeeze out more value and/or volume from logs.

## CHAPTER 2 OBJECTIVES OF THE PROJECT

Objectives of this project were to:

- develop a computer program capable of obtaining optimal sawing, edging, and trimming patterns for logs with any shape in a three-dimensional space.
- analyze the effects of log profile data selection and log, flitch and cant position on value and volume yields.

The program needed to be capable of:

- 1) handling logs in the shape of a cylinder, truncated cone, ellipsoid with or without taper, and twisted solid;
- 2) positioning a log, flitch and cant in three axes;
- 3) both live sawing and cant sawing breakdown of logs;
- 4) considering such parameters as saw kerf, sawing variation and wane allowance;
- 5) providing optimum log position and optimum sawing patterns, ie., the positions of the first cut and each successive cut;
- 6) providing optimum flitch or cant positions and optimum edging patterns for every piece cut from



a log; and .

- 7) providing optimum trimming patterns for every piece cut from flitches or cants.

### CHAPTER 3 LITERATURE REVIEW

As mentioned before, previous log breakdown research can be classified into two categories: (1) models using computer simulation techniques, and (2) models using mathematical programming methods. Early work focused on the use of computer simulation techniques to analyze alternate sawing decisions. Computer simulation models formulate log models using either mathematical descriptions or from empirical data and simulate the sawing process given sawing parameters and sawing methods. Unfortunately, simulation methods do not necessarily locate the optimum. Usually the operator has to inspect the results from simulation runs to obtain a result that approaches optimum. If the program seeks the optimum solution itself, it has to exhaustively try all possibilities and compare all results one by one, which can be very time consuming. If the program wants to arrive at an optimum solution in a reasonable period of time, it has to skip some possibilities, thus risking missing a real optimum point. This led some researchers to explore the application of mathematical programming to the subject. Although many operations research techniques such as linear programming, integer programming, dynamic programming and network techniques have been used for such related areas as log bucking and board cutting, only dynamic programming has

been applied successfully to log sawing. This chapter will trace and discuss literature on simulation models, models using dynamic programming (DP), and proprietary models developed by industry.

### **3.1 Simulation Models**

Models that used the selected sawing procedures and/or obtained the best sawing solution by simply comparing all sawing solutions fall into this category. Compared to other types of models, simulation models had two major advantages: 1) the capabilities of a real sawing process were considered so that the solutions were attainable, and 2) usually they were easier to implement. The disadvantages included: 1) the sawing procedures were limited, and 2) the solutions were usually suboptimum. While selected models are discussed below, this is not a complete literature review but rather an overview.

McAdoo (1969) developed a computer simulation program to compare four small-log sawing methods which were combinations of centered, offset, single taper, and double taper sawing. Logs sawn by the program were represented by truncated cones. Tsolakides (1969) developed a simulation program to study the effects of three alternative sawing methods on grade and volume yields. One of the sawing methods allowed for turning of the log, while the other methods were live and cant sawing. Logs with external and

internal defects were simulated using empirical data from six red oak logs. Each log was sliced into disks and information on the circumference, size and location of the internal defects were recorded manually. The circumference of each log was then rounded in the shape of a cylinder. Reynolds et al. (1969, 1970) presented a computer program (DEFECT) to simulate the log sawing process. Coordinates of the log surface and internal defects were collected from disks sliced from a sample of real logs, and then the log model was represented by a cylinder enclosing the log. Resulting boards, which included defect information on each board for each sawing pattern, were ripped and trimmed by another simulation program called YIELD (Wodzinski et al., 1966).

Richards (1973) simulated log breakdown to study the effects on hardwood lumber yield of such log factors as diameter, length, and taper, and the effects of such sawing factors as board thickness, kerf width, edging method, and sawing method. Six sawing methods, which were combinations of split-taper, full-taper, live, and cant sawing, were used on truncated cone representations of logs. The study indicated that kerf thickness, board thickness, and edging method affected lumber yield significantly, while the sawing method didn't have much effect.

A computer simulation model of log sawing was developed by Airth and Calvert (1973) which simulated a

four-sided sawing method in which four slabs were removed so that a rectangular shape remained. The authors used concepts of solid geometry to generate 3-dimensional shapes such as a cylinder, paraboloid, cone or neiloid, but the cone was chosen as the most appropriate log form in their simulation. The theoretical yield as determined by the program for each log was compared with that obtained by actually sawing the log. The results did not reveal significant differences in lumber yield between the computer and the sawyer's solutions.

Pnevmaticos et al. (1974, 1976) used computer graphics techniques to simulate logs and the process of sawing them into lumber. The log was simulated and displayed as a cylinder or truncated cone by entering the two end diameters and the length of the log. Log positioning operations were not considered since the log representation was simple. Defect locations and dimensions were generated through a random process and displayed as rectangular solids. The process of live sawing the log was also simulated and the resulting boards and slabs displayed.

Wagner and Taylor (1975) developed a simulation program to study the effect of two alternative sawing patterns and log rotation with external and internal defects on lumber value. Data collected from ten southern pine logs were processed by the program, and the logs were theoretically sawn using predetermined sawing patterns

simulating actual sawing planes produced by the saws of a chipping headrig. They concluded that all logs were found to have an optimum log rotation angle that produced the highest value lumber.

Richards (1977) presented a simulation program in which logs were simulated as truncated cones with a taper of 0.3 inch and containing core defects and randomly located knots. The resulting boards were graded and priced by the computer. Both four-sided sawing and live sawing methods with reripping of wide boards for grade and with repeated sawing at 15 degree incremental log rotations were compared in order to study the importance of sawing methods and initial position on value yield. The results indicate that the initial rotational position on the carriage is very important in all sawing methods and that live sawing generally equals or exceeds four-sided sawing in value yield. Later, Richards et al. (1979, 1980) extended the model by including quadrant sawing, cant sawing, decision sawing, live sawing, and live sawing plus reripping for grade at hardwood sawmills, as well as truncated cone logs with a selection of tapers, diameters, core defect diameters, and knot patterns. The results indicated that quadrant, decision, and cant sawing gave similar lumber values. Live sawing was generally better than the other three sawing methods. Live sawing with reripping produced the highest lumber values.

A computer simulation model for sawing hardwood logs was presented by Pnevmticos and Mouland (1978). The log was simulated as a truncated cone with solid rectangular defects either generated randomly or entered from real data. The program was used to compare live, around, and cant sawing employed when sawing sugar maple.

Allekson et al. (1980) discussed three general classes of log models which included the binary array, cross-section, and whole log models. In the binary array model, a three-dimensional binary array is used to represent the log. A particular location in space is represented by a particular bit in memory. In the cross-section models, the log is represented by a series of cross sections which may be circles, ellipses, irregular polygons, or other types of figures. The information is stored in memory as equations or coordinates of polygon vertices. In the whole log model, the surface of the log is represented by a series of points in space, by a single equation or by combinations of equations. The authors developed a computer program which simulated a quad bandsaw operation to optimize the sawing pattern. The program used the binary array model which can closely model logs in any shape from commercial scanners with the assumption that the cross sections are input in terms of grid coordinates provided by either the scanning computer or preprocessing. To compromise between computing time and finding the optimum solution, the program assumed

that two scanners were used. The first one collected the log data and then the optimum position was estimated. After the log was in its stable position, the second scanner recorded the new data which were then stored in the two-dimensional binary array.

Anderson and Reynolds (1981) developed a computer simulation program to calculate the yields that can be obtained from sawing bolts into hardwood squares of various sizes. By analyzing the results from the simulation runs, the operator can determine the best combination of bolt and square sizes for the given bolt diameter range and sawing methods.

Priyasulmana (1983) presented a computer program to simulate live sawing of logs with sweep. Simulated conoid shaped logs with sweep and no other defects were sawn into boards of the same length, thickness, and integer width by four methods of live sawing. The results were compared with that obtained from live sawing of straight logs to study the effects on lumber recovery.

Hallock and Lewis (1971,1973,1978,1985) presented a computer simulation program, the Best Opening Face System (BOF), which perhaps is the most widely publicized and adopted simulation model of log sawing. As early as 1971, Hallock and Lewis proposed the concept of the " Best Opening Face ", which states that the first cut placed on a log determines all successive cuts and, therefore, there



should be a first cut ( opening face ) which is the best in obtaining an optimum yield. They then developed a computer simulation program based on that concept. The first version of the program could only handle cylindrical logs and did not allow any waste on lumber. They kept improving the program, making it more flexible by adding the truncated cone log model, waste allowance, and the ability to simulate most common types of sawmill equipment. Figure 3.1 (1971) gives a brief graphical description of the procedure. In this figure, the narrowest acceptable width of lumber is 4 inches and the opening face increment is 0.2 inch.

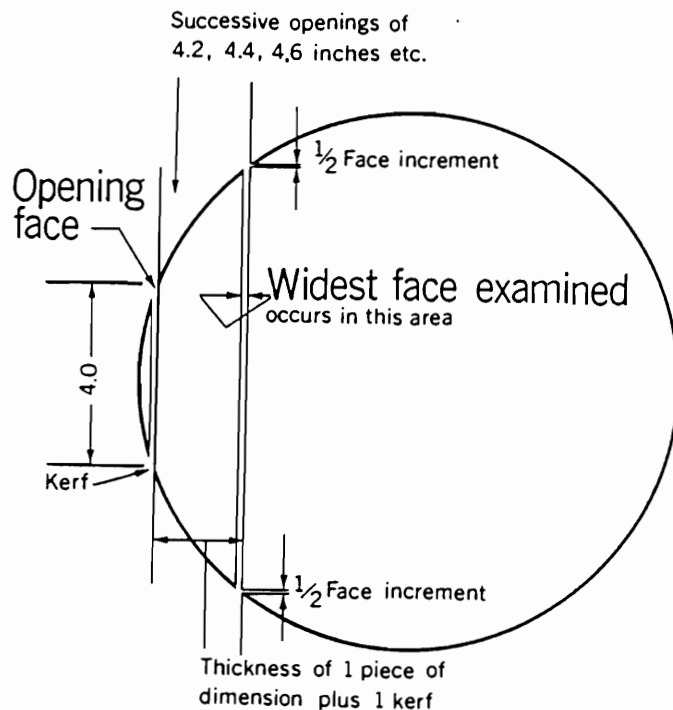


Figure 3.1 Procedure followed to determine the initial opening face and subsequent trial opening faces for each log diameter (Hallock et al., 1971).

Despite the fact that the BOF system gives " the best opening face ", there are two main reasons why there are some opportunities for obtaining even more value or volume from a given log than the solution achieved by the BOF system. First, the final solution is selected from a limited discrete solution set of enumerations. BOF searches for the best opening face by enumerating all possible cutting positions within a range. Obviously, making the opening face increment smaller will increase the chance that the true optimum result is in the solution set. Using Figure 3.1 as an example, suppose the optimum yield would be achieved if the opening face was located somewhere between 4.2 and 4.4 inches. Unfortunately, the procedure only searched for the best opening face among the discrete points of 4.0, 4.2, 4.4, 4.6, etc. The opening face resulting in the optimum yield was not in the solution set, so " the best opening face " given by this procedure was not " the best ". Since the possible number of opening faces is infinite, and it is impossible to search for the best solution over a infinite solution set, a compromise between computing time and solution accuracy must be made by the operator.

The other concern with the BOF model is one shared by most of the other models and that is that the log models are over simplified. The BOF system uses cylindrical and truncated cone log models, yet the real shapes of most

logs are not cylindrical or truncated cone-like. In addition, defects in the logs are also ignored by BOF. While a publication was released describing a version of the BOF system which allows logs with sweep and elliptical cross sections to be modeled (Lewis, 1985), the model itself was never completed<sup>1</sup>.

### **3.2 Models Using Dynamic Programming**

Dynamic programming (DP) can be computationally more efficient than the exhaustive searching method described in the previous section on simulation models. It has been successfully used for the optimization of log bucking since the early 1970's (Pnevmticos and Mann, 1972; Glück and Koch, 1973; Briggs, 1980; Rogler and Canham, 1986.). However, in the area of log sawing optimization, only a few models applying the DP algorithm have been developed and all have used over-simplified log models.

Tejavibulya (1981) presented two dynamic programming models for finding the optimum sawing solution in the live sawing method and in a cant sawing method. These models appear to be some of the earliest models using DP. The major contribution of the study is that, for the first time, the models allowed the flexibility of considering a variety of board thicknesses while still guaranteeing the

---

<sup>1</sup> Danielson, J. 1990. Personal communication. Project Leader, USDA Forest Service, Madison, WI.

optimum yield pattern. Unfortunately, the models were over-simplified by using a perfect cylinder as the log model.

Geerts (1984) presented a mathematical solution for optimizing the sawing pattern of a log with a defect core. Both the log and defect core are assumed to be cylinders. He used a two-level approach which consists of a similar dynamic programming algorithm for each level. The first level maximizes the breakdown of the log into flitches and the second level maximizes the breakdown of each flitch into lumber. The maximum value of each flitch, which is the output of the second level, was returned to the first level. If the profile of log cross sections was available, the algorithm would allow for dealing with an irregular shape defined by polygons, but no taper was taken into account. He developed a computer program in which both logs and defect core were modeled as cylinders to implement his algorithm. Excluding the 3-D cutting problem and over-simplifying the log model are limitations to the program.

Faaland and Briggs (1984) presented a DP formulation that simultaneously optimizes bucking, live saws logs into lumber, and edges lumber into finished dimensions. The model integrates the bucking study of Briggs (1980), the DP sawing study of Tejavibulya (1981), and a knapsack formulation for edging. Logs with taper, sweep, crook and

defects are allowed by the model. Limitations to the log model are that any cross section of the log must be circular and variation along the length of the log should be in a two-dimensional plane. The model bucks a log into segments by using the DP bucking program developed by Briggs (1980), then tries to seek a largest cylinder inscribed inside each segment. Breakdown of the cylinder found from each segment is performed by the DP sawing model and the optimum value of each cylinder is returned to the bucking model. The edging model edges each flitch cut out of the cylinder and returns the optimum value of the flitch to the sawing model. Transforming log irregularity into a cylinder and then performing all operations on the cylinder greatly simplified calculations and significantly reduced computing time. However, these restrictions greatly increase the chance of losing value or volume.

### **3.3 Models Developed by Industry**

Research groups in industry have also developed various computer optimized log bucking, log sawing, or integrated bucking-sawing program packages and systems. Brief reports and advertisements are available in public media such as symposiums, magazines, advertising flyers and research circular papers. However, because relatively little explanation of the computational procedures or other details are available, these models will be discussed in

this separate section. For the same reason, the following discussion on features, advantages, and disadvantages of models is incomplete.

Although the optimum bucking problem is not within the scope of the project being reported in this thesis, it will be discussed because most optimum bucking systems include an optimum sawing solution in obtaining the value or volume of bucked segments.

Advanced Control Technology, Inc.<sup>2</sup> developed an ACT " REAL TIME " LOG BREAKDOWN SYSTEM which is able to process logs through a primary breakdown machine at speeds up to 300 lineal feet per minute. When a log enters the system, log diameters and position are collected on a continuous basis. Length, diameter and position information are assigned to each respective 6 inches of log as it travels through the system. Once the system has collected and built a log model which reflects crook, sweep, ovality and length, the optimization routine uses this information along with lumber sizes, kerf sizes, and lumber values to determine the cutting solution. If a radical log shape, time constraints, or malfunctioning inputs prevent a "REAL TIME" solution from being calculated, an alternate solution will be determined from a predefined table of cutting patterns. The breakdown solution is then presented to a PLC

---

<sup>2</sup> Advanced Control Technology Inc. 1988. Company brochure. Albany, Oregon.

machine controller which issues the machine settings. The system uses a multiple processor VME system. The application programs are written in PASCAL and are in modular format to take advantage of up to three 68000 family CPU's in the VME system. The company claims that this system is the first "real time", "real shape" system capable of operating on close-coupled primary breakdown lines like Chip-N-Saws. However, the real time also means that longitudinal information such as sweep and crook is limited because the cross sections of the log are only cylindrical or elliptical. Another limitation is that the system is not able to position the log in 3-dimensional space.

Applied Theory<sup>3</sup> ( Sullivan, 1987) has developed the "Maximizer" production line for bucking stations, carriages, sharp chains, edgers, gang-saws, resaws and trimmers to obtain the maximum recovery of lumber. The Bucking Maximizer is equipped with curtain-type scanners placed on the X and Y axes for diameter measurement. By scanning on two axes and plotting the position of each cross section in space, the system develops a picture of the stem including sweep, crook, butt swell and the natural elliptical shape. As the stem travels through the scanners, the Maximizer determines the products that can be sawn,

---

<sup>3</sup> Applied Theory, Forest Industries Division of U.S. Natural Resources, Inc. 1991. Company brochure. Corvallis, Oregon.

beginning with the first 8-foot segment passing through the scanners. These products are assigned a priority value by the mill and defined in terms of wane allowance and downstream cutting capability. Up to 100 different boards can be defined and loaded into a board-value table. Whether this 8-foot segment has sweep or crook and is round or elliptical, the Maximizer will fit defined boards from the table into the shape. It moves them until it comes up with the most valuable combination. The segment is then assigned a price. The process is repeated for 10, 12, 14, 16, 18, and 20 foot segments. The Maximizer will then make comparisons between the many combinations to obtain the optimum value for each stem and calculate whether to buck for maximum length, cut out sections with sweep, or to divide a section with sweep into two segments with less sweep. A company test showed a 7.46 percent average increase in volume and 12.76 percent average increase in value, for 16 stems bucked by the Bucking Maximizer when compared to human operators. The other log breakdown optimizers are also equipped with similar scanning systems. Therefore, cross-sectional shape representations in all systems are limited to circles or ovals.

Carrol-Hatch<sup>4</sup> provides SAWSIM, a sawmill simulation program originally developed by Leach in 1973 (Leach,

---

<sup>4</sup> Carrol-Hatch (International) Ltd. 1991. Company brochure. North Vancouver, B.C., Canada.



1973). SAWSIM can simulate straight logs with uniform taper as well as logs with crook, sweep, and variable taper and with circular or elliptical cross sections. It simulates any system used for primary log breakdown, such as bandsaw/carriage, multiple bandsaw and chipper canters. The user can define sawing, edging, resawing and trimming patterns, or select the optional patterns provided by the program. Log rotation is allowed.

Coe<sup>5</sup> provides optimizers for trimmer, edger, and canter/gang systems. Recently the company developed a log scanning system using a ranging-type laser scanner similar to the system originally used for lumber profile scanning, which is able to provide polygonal representations of log cross sections. A log breakdown optimization system equipped with the profile scanning system was recently installed in a sawmill<sup>6</sup>. The benefits of this installation are not yet available from the mill.

Denis Sawmill Equipment Inc.<sup>7</sup> manufactures "Mega-Vision", a bucking optimizer system. Stem diameter, length, sweep, and taper are measured by a dual-axis longitudinal or a transverse scanning system. The built-

---

<sup>5</sup> Coe Manufacturing Company. 1991. Company brochure. Tigard, Oregon.

<sup>6</sup> A. Donald Moen, P.E.. Coe Manufacturing Company. Personal communication. March, 1991.

<sup>7</sup> Denis Sawmill Equipment Inc.. 1991. Company brochure. Boisbriand, Quebec, Canada.

in, user-friendly software allows the user to edit sawing patterns according to the actual production line. For each log cut from the stem, the system evaluates lumber yield by fitting the user-edited sawing pattern into the circumference of each scanning cross section and then using the full-taper, half-taper, or curve sawing method. Then the bucking decision is optimized by comparing values or volumes from different bucking decisions. This system is not able to represent the real shape of log cross sections.

Lloyd Controls, (Thomlinson, 1987) developed an optimum bucking system for optimizing tree-length logs entering a sawmill. The overall system consists of a scanner system, an on-line process control computer and interface system, an off-line simulation computer, and a manual back-up system. Information from the scanner system is fed directly into the on-line process control computer which then interpolates between data generated at the scan locations to determine the diameters and positions of the stem cross sections at the average of all possible bucking locations relative to the end of the stem. The on-line bucking optimization program then determines which combination of permissible bucking and end-stop positions will maximize the value of the stem. It does this by determining the values of all combinations of bucked lengths that could yield an acceptable optimum solution. Values for logs that have crook or sweep are calculated

from values for straight logs that are determined to be of equivalent value, with an adjustment for the value of any additional yield in chips. Values for straight logs of various small-end diameters, lengths and tapers, in excess of their values as chips, are obtained from log price table files pre-generated by the off-line simulation computer. One of the limitations is that the solution is not optimum because the value of a log is determined by approximating a truncated cone rather than the real log shape.

Mel Cruickshank Equipment Ltd. (MCE) (Scaramella, 1987), an exclusive Canadian distributor for KEBA, AIT (two Austrian companies) and LINCK, supplies a "Production Optimizing System" for obtaining the optimum population of logs cut from stems entering a sawmill. The three component system includes a "Production Scheduling System", a "Stem Optimizer", and scanners. The Production Scheduling System (PSS) consists of a Production Scheduler and a Production Optimizer. The Production Scheduler is only good for sawmills that have log sorting and is used to schedule the actual production to minimize work stoppage and to tell the board sorters what material is coming and to which customer the materials belongs. On the other hand, the Production Optimizer can be used by any sawmill. It incorporates both a linear program and a dynamic cutting pattern optimizer. These programs interact dynamically to calculate the most profitable way to break down logs, given

constraints on the log supply, the market demand for lumber, and the existing sawmill equipment. The Stem Optimizer is the second component of the Production Optimizing System. It cuts logs based on the PSS specifications. A curtain-type scanning system is used, which takes snapshot images of a log starting from the butt-end of the stem and at 10 centimeter increments to the small-end and therefore, is able to measure sweep and crook of the log. The values of the largest diameter cylinders that fit inside the stem for all lengths in the order list are calculated and loaded into the crook and sweep table. When the log has been scanned, stem values are optimized using the values from the crook and sweep table. One of the limitations to the system is that the real shape of cross sections of a log is not represented. Possible value loss exists because only a cylinder shape is used to find sawing pattern solutions.

North American Controls<sup>8</sup> produces such sawmill optimization systems as NAC Primary Log Breakdown Systems and Trimmer Optimizer System. Recently the company completed a project which was designed to profile scan large grade logs or stems, up to 48 inches in diameter (Rickford, 1989). Using six CCD solid-state area-array imaging cameras and seven-HeNe 5mW spreadline laser

---

<sup>8</sup> North American Controls Inc. 1991. Company brochure. Portland Oregon.

scanning systems to capture an image of a log circumference, the NAC Log Profiler is able to generate 360 degree contour profiles every 1 to 2 inches along the length of the log, and create from this raw scanner data cross-sectional multiple polygons comprising 36 surface point coordinates at 10 degree intervals from the calculated centroid of each cross section. However, the installation of the log breakdown system using the log profiler has not been reported.

Porter Engineering Ltd.<sup>9</sup> developed "RT2 Software" to optimize dollar or lumber recovery. The system uses curtain scanners or cameras to obtain log diameter, length, sweep and taper data in 1 or 2 axes and generates elliptical slices of the log every 1 inch. The edger optimizer cuts are approximated for a side lumber projection. Three side boards on each side of a cant are assumed and 29 allowed symmetrical and asymmetrical horizontal cutting patterns are evaluated for each cant. The system allows the user to enter either U.S., or metric dimensions or a combination of the two at the same time. The company claims that with dual axis scanning the system can provide an increase in recovery of from 3 to 9 percent over an existing single axis, look-up table system.

---

<sup>9</sup> Porter Engineering Ltd. 1991. Company brochure. Richmond, B.C., Canada.

Warren and Brewster Co.<sup>10</sup> presented a "Maxi Optimizer System" which can generate real time optimal sawing solutions based on actual log geometry from precise scan data, sawing parameters ( saw kerfs, target sizes, wane allowances, etc.), and values for each combination of length, dimension, grade, and species being processed. The system offers a graphic illustration of all saw-lines and the lumber it will produce. The optimizer uses a curtain-type scanner to get log shape information, which includes sweep and crook. Unfortunately, at best the curtain-type scanner can only locate four circumferential points on a cross section of the log, and thus can give only a circular or elliptical cross-sectional representation of the real log.

Weyerhaeuser Company (Cooney,1987) developed three versions of an interactive training system, which includes the VISION Cutting-for-Value System, MicroVISION, and MicroVISION Instructional System, for improving tree-bucking decisions. The VISION Cutting-for-Value System is a package of software and graphics hardware that supports 3-D graphics and interactive database updating. Log values, mill descriptions, type of cut (longitudinal or transverse), stem descriptions, and other operation specific data may be varied by the user. Sweep, crook, knots, and other quality defects may be included in the

---

<sup>10</sup> WB Company. 1991. Company brochure. Albany, Oregon.

stem description. After a set of stems has been selected for analysis, VISION displays a three-dimensional image of the stem which may be rotated or rolled (using a joy-stick) to view the stem from any vantage point. The user then moves a "saw" along the length of the stem, cutting it where desired. The resulting breakdown and product values are displayed. The system then calculates the optimal decision and displays the result for comparison with the user's decision. The other two versions do not permit the user to change the database of stem descriptions, log values, and other log allocation data. Limitations are that the stems are displayed in a 2-D form and can not be reoriented, and the model does not optimize the breakdown process.

Today, various types of optimizing systems have been used extensively in the lumber industry, and those systems have already proven themselves to be great contributors to improved lumber recovery. Most manufacturers of various log breakdown optimizers claim that their equipment is "real time" and "real shape". However, because of processing time and scanning system limitations, most current log breakdown optimization models only work on greatly simplified log shape representations, and they typically use heuristic search rather than mathematical optimization techniques to increase recovery. The "real shape" claimed by those companies actually means that such

longitudinal information as sweep and crook are measured, but cross sections are still represented by a round or elliptical shape. Among all companies cited above, only Applied Theory, Coe, and North American Control developed log scanning systems which are able to provide polygonal shape representations of log cross sections, and only Coe shipped one such system to a sawmill. To squeeze the highest value from every log of any shape, optimization models capable of considering log shapes in 3-dimensional (3-D) space and log scanning systems capable of providing those true shapes need to be developed. Log scanning techniques are evolving from two-point, curtain-type scanners to ones using more scanning points, with even more advanced systems such as ones based on computer tomography on the horizon (Funt et al., 1987). Therefore, this is the appropriate time to undertake research on process control software that will use 3-D log shape scanning information to improve lumber recovery and provide basic analytical data for further development of more sophisticated but economical scanning systems.



#### CHAPTER 4 MODEL DEVELOPMENT

An ideal log breakdown model needs to contain at least the following features:

- (1) utilize 3-dimensional log shape information;
- (2) conduct positioning operations on the log and each cant, flitch, and board cut from the log in order to find the optimum position for each one;
- (3) use mathematically-based algorithms to optimize log breakdown, edging and trimming patterns;
- (4) allow multi-thickness sawing patterns;
- (5) allow any mix of any lumber dimensions (American Lumber Standards or proprietary dimensions, green or dry sizes, rough or finished dimensions, etc.);
- (6) allow live, cant, round, full-taper, and split-taper sawing;
- (7) include parameters such as saw kerfs, sawing variations, and wane allowance;
- (8) run on a microcomputer;
- (9) consider internal defects;
- (10) consider annual ring orientation; and
- (11) provide real-time solutions.

As discussed in Chapter 3, all existing models are limited in one way or another. The program described in this thesis, called SAW3D, was developed as an initial step

towards reaching the "ideal" model. The current version of SAW3D is an advancement in terms of combining 3-dimensional log modeling, optimization/simulation options, positioning operations, and its microcomputer-based environment.

#### **4.1 Description of the System Procedure**

This section introduces a general picture of the procedures used in SAW3D; all processes will be discussed in greater detail later.

##### **4.1.1 The Overall Procedure**

After receiving log profile data from a scanning system, SAW3D mathematically rotates the log counterclockwise and then skews it horizontally (Figure 4.1). Then the program breaks down the log into slabs, flitches, or cants. For each piece cut from the log, the program will mathematically pitch it vertically in the coordinate system, which is equivalent to orienting it horizontally on an edger, and then edge it into untrimmed lumber. After that, the program will trim every untrimmed piece into finished products. This process repeats until all possible cutting patterns are searched and all rotation and skewing positions are compared. The optimum value of the log and optimum value of each slab/flitch/cant are obtained. In addition, corresponding optimum log position, all optimum slab/flitch/cant positions and all optimum

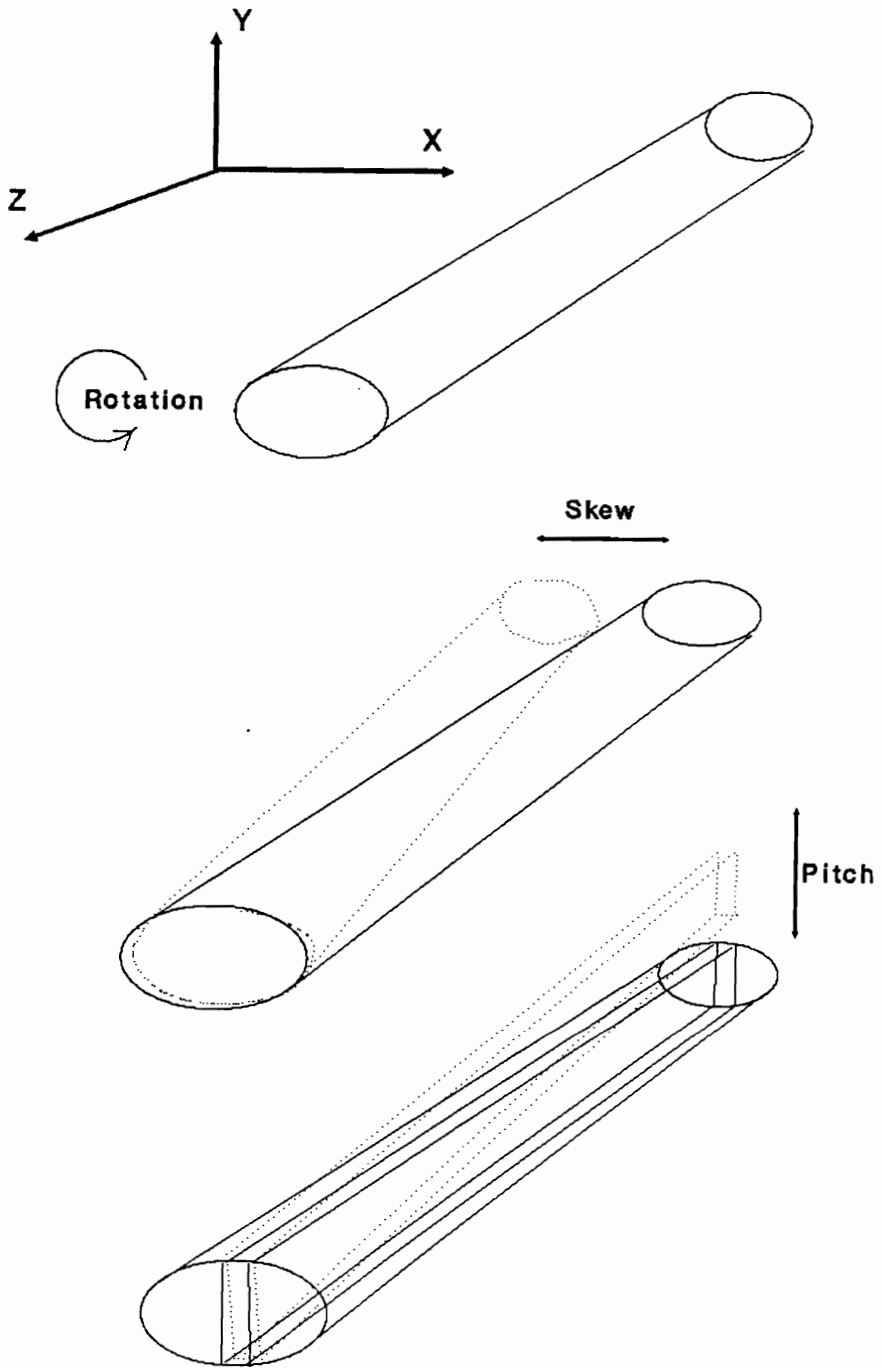


Figure 4.1. Log and flitch orientation.

sawing line, edging line and trimming line positions are provided. Figures 4.2 and 4.3 provide the overall flowchart of the procedure.

The program consists of three interrelated levels. The first level is the sawing level for the primary breakdown of the log, the second level is the edging level for edging a slab, flitch, or cant, and the third level is the trimming level.

#### **4.1.2 Level 1: The Sawing Level**

Level 1 (Figure 4.2) is used to determine optimum sawing patterns and optimum log positions. Input data include log profile data, production parameters, a lumber price table, and control options. The log is described by a wire-frame model based on all scanning points around the surface of the log. Any log shape can be handled. The process starts by rotating the log. The log can be rotated through the full 360 degrees at any rotating increment angle entered by the operator. For logs in the shape of a cylinder or truncated cone, the rotation operation is skipped. Once a rotation position is determined and the log is rotated to that position, the program will determine a skew range. Skewing the log means to move it horizontally in the coordinate system. The skewing angle increment is also entered by the operator. As in rotation, the skewing operation is skipped for cylinders. After the

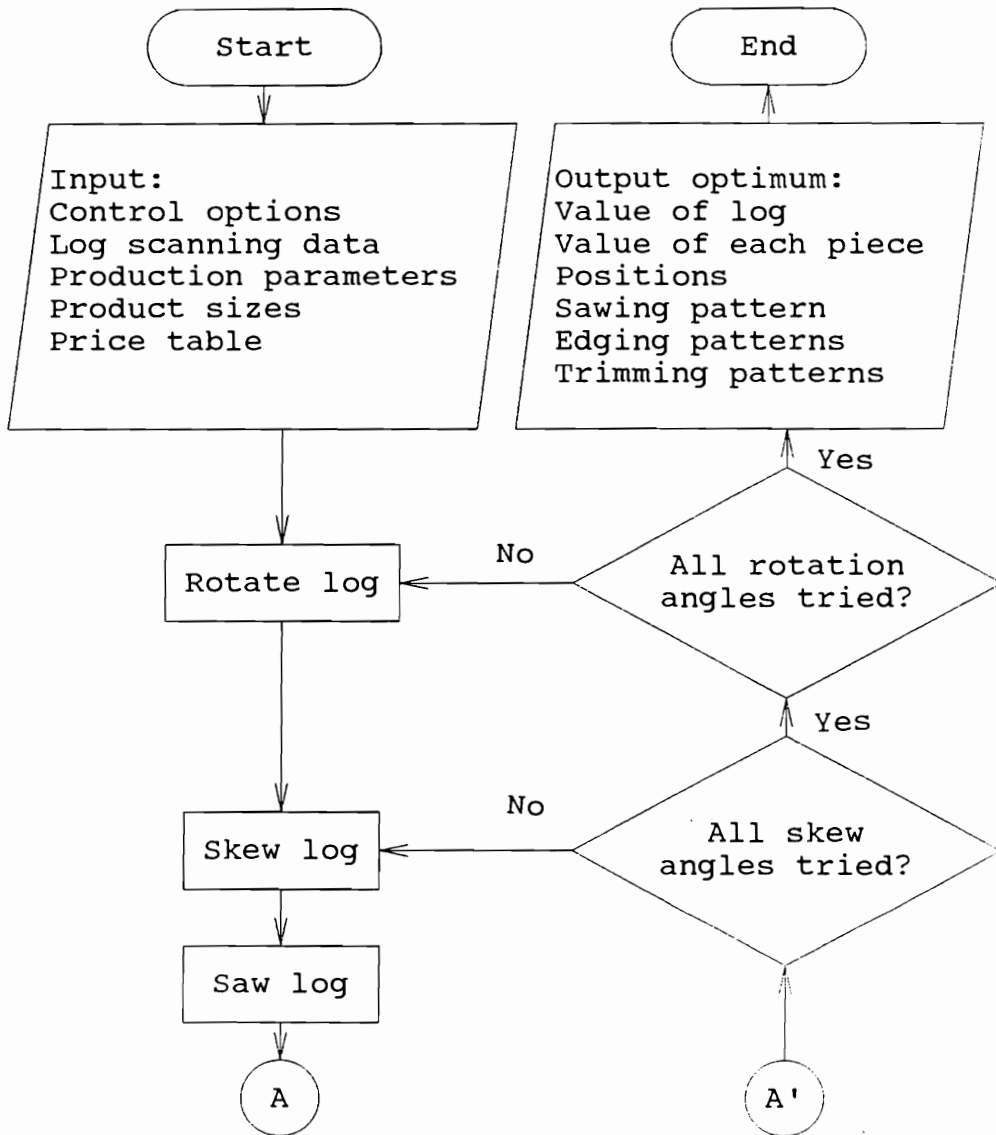


Figure 4.2. Flowchart of log sawing level. A and A' are the links to Figure 4.3.

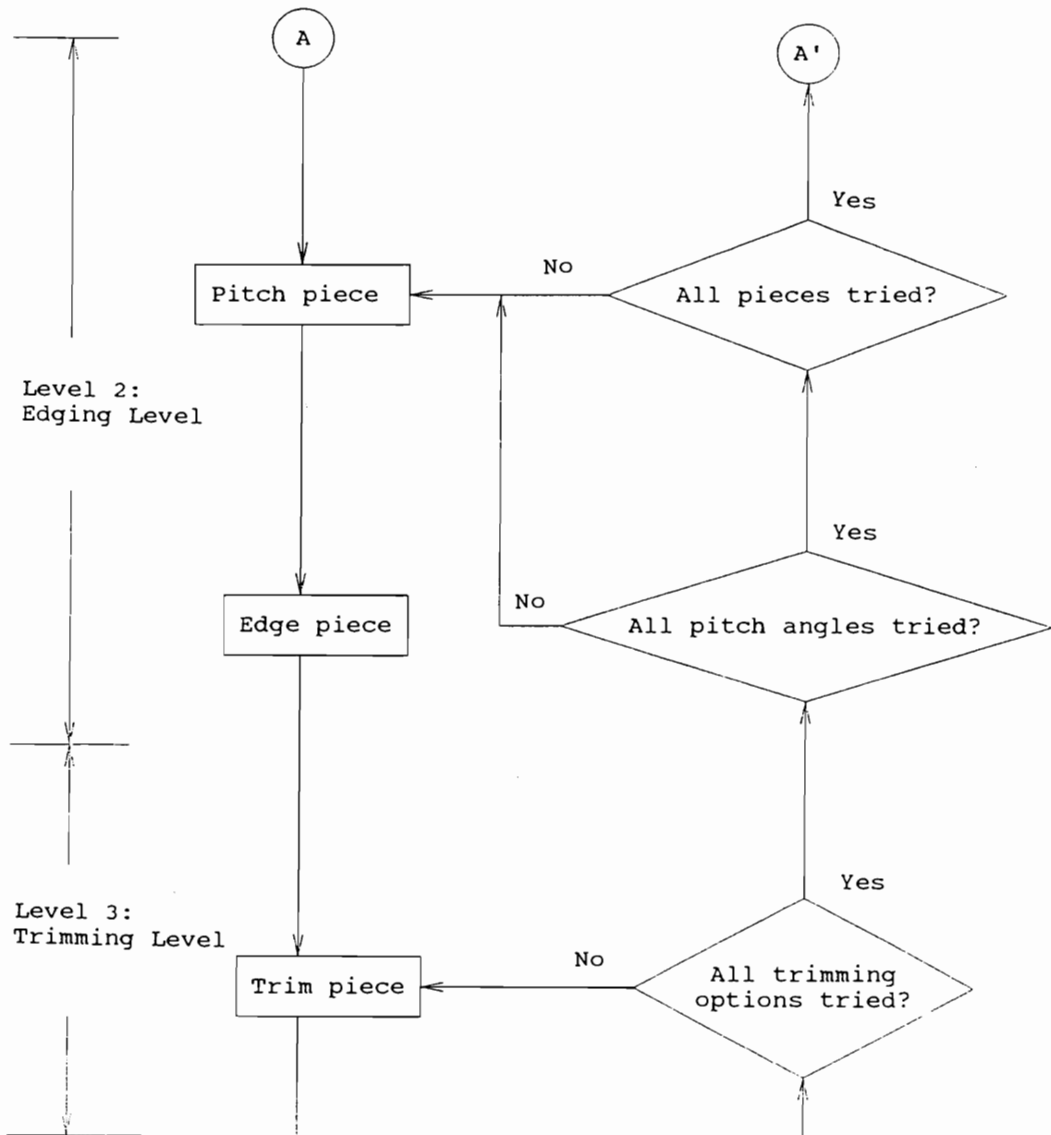


Figure 4.3. Flowchart of edging and trimming levels. A and A' are the links to Figure 4.2.

position of the log has been fixed, the program breaks it down using a dynamic programming algorithm to find the optimum sawing pattern. The monetary value of each trial board or cant is obtained by the edging level. The optimum sawing pattern, which provides the maximum total value or volume of all slabs, flitches, and cants, is determined by the dynamic programming algorithm. This total value or volume is compared with that given by the previous log position and the best one is saved along with all corresponding optimum position information and the sawing pattern. This repeats until all log rotation and skewing positions are enumerated and the final optimum solution is reached.

#### **4.1.3 Level 2: The Edging Level**

The edging level's function (Figure 4.3) is to obtain the optimum value of a slab, flitch or cant. The sawing line positions of the two faces of each piece are given by the first level. The edging level first determines a pitching range. Pitching a piece means to orient it vertically in the coordinate system, and the pitching angle increment is also entered by the operator. Once a pitching position is chosen and the piece is pitched to that position, the edging level will edge the piece using a dynamic programming algorithm similar to that in the first level. The value of each piece of untrimmed lumber edged

from the piece is obtained by the third level. The dynamic programming algorithm determines the optimum edging pattern yielding the optimum value or volume of the piece. Then the value or volume is compared with that given by the previous pitching position, and the best one is saved along with all corresponding optimum position information and the edging pattern. This repeats until all pitching positions in the pitching range have been tried and the final optimum solution is obtained. Then the final optimum value of the piece is returned to the first level.

#### **4.1.4 Level 3: The Trimming Level**

The trimming level (Figure 4.3) is simpler than the other two levels. The positioning operation is not necessary any more since two edges of the untrimmed lumber are now parallel. This level engages a dynamic programming algorithm, which is similar to that in the other two levels, to search for the trimming pattern yielding the optimum value or volume, which is then returned to the second level.

## **4.2 Log Modeling**

How the log shape is described is one of the critical aspects in obtaining an optimum breakdown solution. No matter what kind of optimum searching algorithm is engaged, the final optimum solution is obtained by operating on that



log model. Obviously, the closer to the real log shape the log model is, the better the solution will be.

There are several log modeling methods and each has its own advantages and disadvantages. This section briefly introduces the existing log modeling methods and then focuses on the polygonal cross-section model, which is used in this thesis.

#### **4.2.1 Cylinder and Truncated-Cone Models**

The cylinder model is the simplest log shape model. Given a diameter and length, the log is represented by the equation for a cylinder. Because taper is not considered, no log is really correctly described by this model. The truncated-cone model is better than the cylinder model since taper is included. Given the small-end diameter, taper, and length, the log can be represented by a truncated cone. These two models are easily formulated by mathematical equations and are very easy to solve with any optimum searching algorithm. As a result, most existing programs use either cylinder or truncated-cone log models.

#### **4.2.2 Cross-Section Models**

Cross-section models use a series of cross sections at intervals along the length of the log to represent the log. The log surface is represented by straight lines connecting each pair of intervals and each pair of cross sections.

Since data at each cross section can be different, some irregularity such as crook and sweep can be included in the model. The cross-section models can be in one of three categories according to the shape of the cross sections.

1. Circular cross-section model. In this model, all the cross sections are circular. By giving different diameters to each cross section, the irregularity along the length of the log can be represented to some extent.

2. Elliptical cross-section model. In this model, all the cross sections are elliptical. The long axes and the short axes of all the cross sections are not necessarily parallel. Since this model contains more information on the cross sections, it is closer to the real shape of logs than the circular cross-section model.

3. Polygonal cross-section model. In this model, all the cross sections are represented by polygons. This model is much closer to a real log shape since the irregularities on the cross sections are also included in the model. The program described in this thesis uses the polygonal cross-section model, so it will be discussed in greater detail later.

#### **4.2.3 Surface Fitting Models**

In cross-section models, the log surface is represented by straight lines between each pair of cross sections. Therefore, some information about the real shape

of the log between the cross sections is lost. Surface fitting models pick up some of that information by using curve or surface fitting equations instead of straight lines. There are many curve or surface fitting techniques available, such as cubic spline interpolation, B-spline methods and Bezier methods.

#### **4.2.4 Log Shape Model Used in This Program**

When choosing a log modeling method, two aspects deserve very careful attention, reality and complexity. If a model more closely describes the real shape of the log than other models, it will contain more information on the log. Therefore, the final solution could be better when given the same optimum searching algorithm, but the computing time is also increased. Among all the modeling techniques, the cylinder and truncated cone are very easy to handle, since the log can be described by simple mathematical equations, resulting in shorter computing time. The cross-section models are better models of a real log, but need longer computing times. The surface fitting models are even better, but require the greatest computing times. As a compromise between reality and complexity, the polygonal cross-section model was chosen for use in this program.

In the program discussed in this thesis, the log is represented by a series of polygonal cross sections at

selected intervals along the length of the log. Scanning points representing vertices of polygonal cross sections can be provided by some of the latest log scanning systems, such as the system supplied by Coe<sup>11</sup>. The appropriate number of scanning points (vertices) required for every cross section as well as the distance between two cross sections can be analyzed by executing this sawing program since both are variables entered by the operator. The surface of the log between two adjacent cross sections is described by straight lines connecting corresponding vertices on the two cross sections. Figure 4.4 shows one cross section of the log model, while Figure 4.5 shows the whole log modeled by this method.

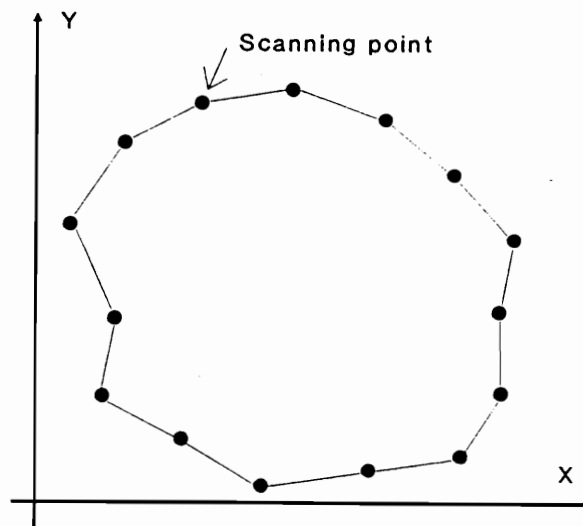


Figure 4.4. A cross section of the polygonal cross-section model.

---

<sup>11</sup> Coe Manufacturing Company. 1991. Company brochure. Tigard, Oregon.

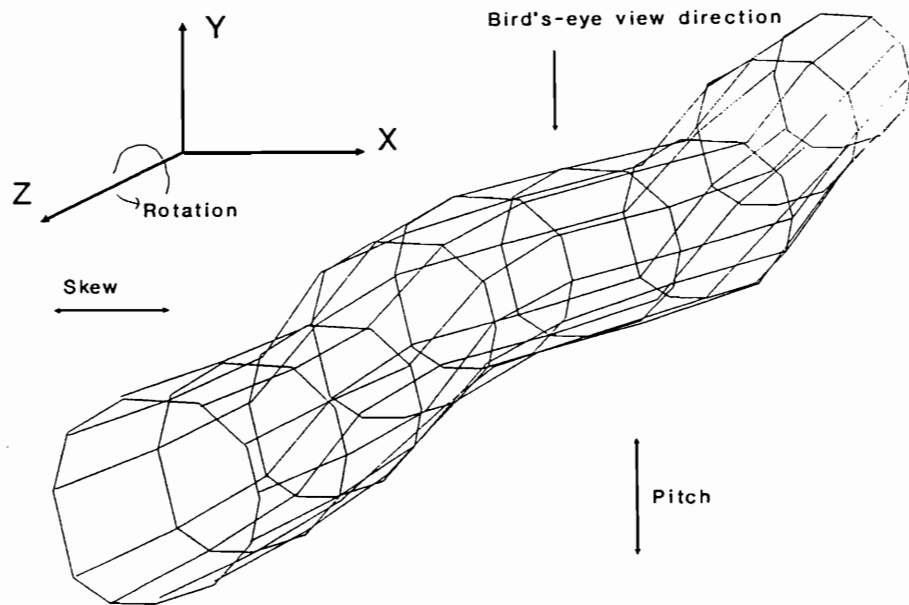


Figure 4.5. A log represented by the polygonal cross-section model.

### 4.3 Sawing Pattern Optimization

To optimize the value or volume yield of a log, the log breakdown program must be able to mathematically search

for the log's optimum position and determine the optimum sawing pattern for that optimum position. The log positioning operation includes two movements: rotating the log through 360 degrees and skewing the log horizontally through a specified range. Skewing the log vertically is not necessary, since all cutting planes are vertical.

The sawing program first mathematically orients the log in a certain position, and then saws it using a dynamic programming algorithm. The solution is compared with that found in the previous position and the better solution is saved. This process repeats until all positions have been searched. Figure 4.6 illustrates the process of the sawing level in greater detail than Figure 4.2. To concentrate on this level, the relationship between this level and the edging level is not shown in Figure 4.6.

#### **4.3.1 Log Positioning**

As previously mentioned, log positioning includes rotating and skewing the log to find the best position. To do this, the program mathematically rotates the log around the Z axis first and then skews it around the Y axis. After rotation and skewing, the dynamic programming sawing algorithm is performed on the log in that position. While this process is repeated until all positions are tried, in fact there are an infinite number of possible log positions. To solve the problem in a reasonable amount of

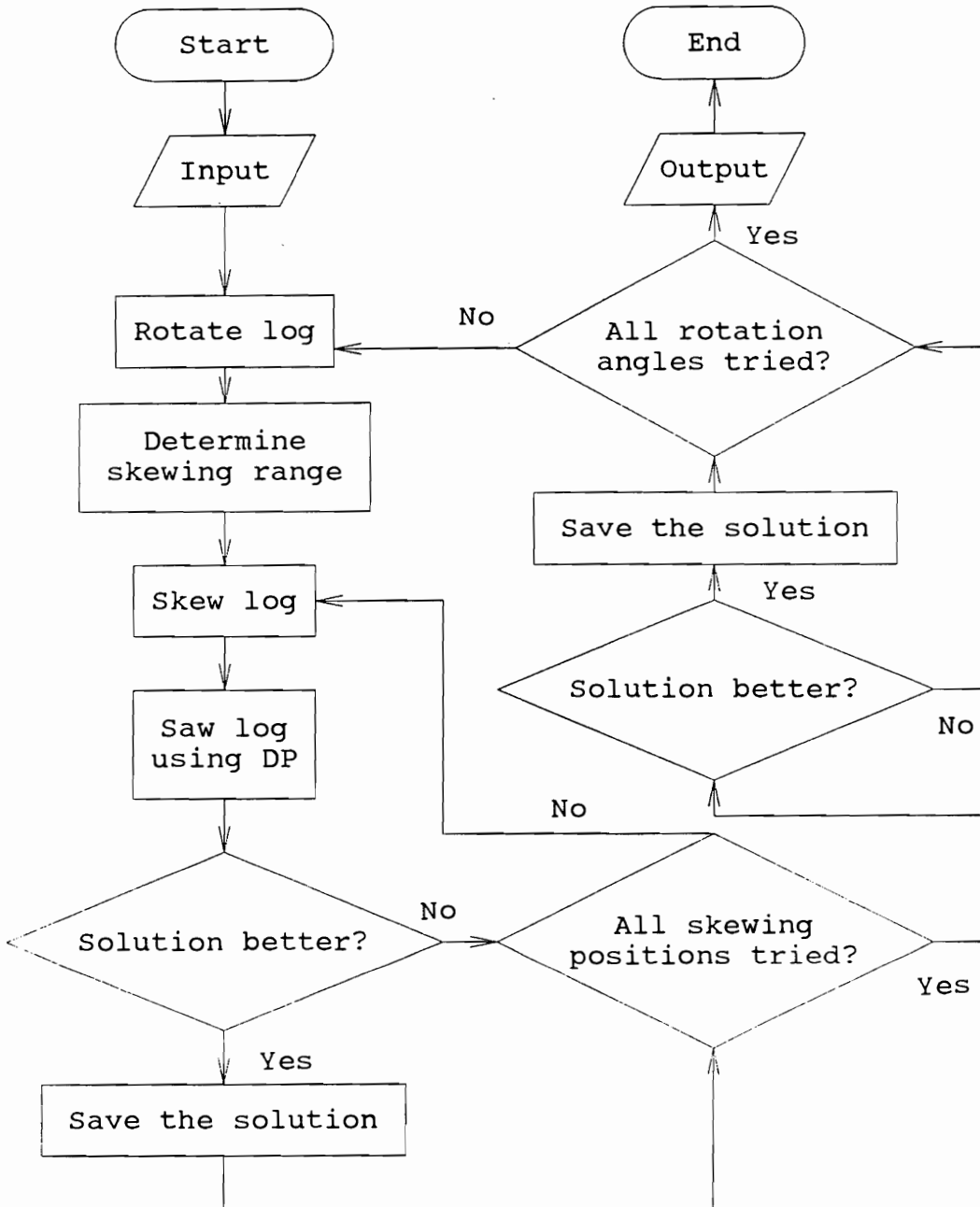


Figure 4.6. Sawing level processes.

time, position searching ranges and position changing increments must be chosen. Therefore, there are two steps involved in carrying out the positioning operations: (1) determining the positioning range and (2) positioning the log through that range.

There are two methods of positioning the log. One is to transform the coordinates for all the scanning points, while the other is to transform cutting planes instead of the whole log. The later approach seems to need less transformation operations. Unfortunately, it may not necessarily be efficient when considering the whole program. Determining the intersections of cutting planes with the log is very easy if the cutting planes are perpendicular to the X axis, which is the case when the log is transformed. If the cutting planes are not perpendicular to the X axis, which is the case when the cutting planes are transformed, more operations will be required to find intersection planes. Therefore, this program transforms the whole log instead of cutting planes.

#### **4.3.1.1 Rotating the Log**

Mathematically rotating the log is very straightforward. The rotation range from 0 to 360 degrees and the rotation angle increment are allowed to be entered by the user. Obviously, the smaller the increment, the better the chance that the optimum solution will be found.



However, longer computing times will also be required. Therefore, a compromise should be made by the operator. The log rotation is mathematically performed using the following assignments written in the "C" programming language:

```
sine=sin(RotAngle);
cosine=cos(RotAngle);
for (i=0; i<NoCrossSec; i=i+1) {
    for(j=0; j<NoScanPts; j=j+1) {
        RXY.X[i][j]=XY.X[i][j]*cosine - XY.Y[i][j]*sine;
        RXY.Y[i][j]=XY.X[i][j]*sine + XY.Y[i][j]*cosine;
        RXY.Z[i][j]=XY.Z[i][j];
    }
}
```

Where:

```
RotAngle      = rotation angle;
sine          = sine of rotation angle;
cosine        = cosine of rotation angle;
NoCrossSec    = number of cross sections;
NoScanPts     = number of scanning points on each
                cross section;
XY.X[i][j]    = X coordinate of scanning point j on
                cross section i before any
                positioning;
```

XY.Y[i][j] = Y coordinate of scanning point j on cross section i before any positioning;

XY.Z[i][j] = Z coordinate of scanning point j on cross section i before any positioning;

RXY.X[i][j] = X coordinate of scanning point j on cross section i after rotation;

RXY.Y[i][j] = Y coordinate of scanning point j on cross section i after rotation;

RXY.Z[i][j] = Z coordinate of scanning point j on cross section i after rotation.

#### 4.3.1.2. Skewing the Log

After the log is mathematically rotated to a position, the program skews it horizontally. The first step is to determine the skewing range, which is defined by the maximum angle and minimum angle through which the log is to be skewed. The procedure for determining the skewing range is:

- (1) Find the left and right edges of the log's X-Z projection. These edges correspond to the maximum and minimum X coordinates for each cross section. Figure 4.7 shows the projection of the log onto the X-Z plane as seen from the bird's-eye view.

- (2) Select a group of points on one edge of the log,

i.e., determine how many cross sections are to be included in the group. The number of points selected depends upon the interval between cross sections and the shortest lumber length allowed. The distance between the first and the last points in the group should be greater than or equal to the shortest lumber length. For example, if the interval between cross sections is two feet, and the shortest lumber length is eight feet, then there should be five points selected (Figure 4.8).

(3) Find a straight line fitting this group of points, and then find the slope of this straight line.

(4) Shift the group of selected points one cross section forward, i.e., drop the first point in the group and add another point adjacent to the last point in the group. Repeat steps (3) and (4) until the remaining portion is shorter than the shortest lumber length allowed.

(5) Repeat steps (2), (3) and (4) for the other edge.

(6) Find the maximum slope and the minimum slope from all slopes obtained in the previous steps.

Once the skewing range is determined, the program mathematically skews the log to a position in that skewing range. Since any position in the range could be chosen, there is an infinite number of possible positions. Therefore, a subset of positions in the range must be specified by the operator. If the operator entered two, for instance, the skewing angle increment would be:

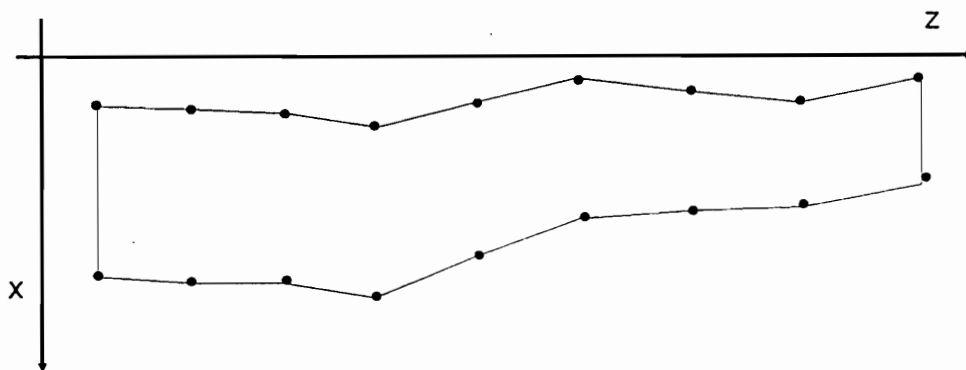
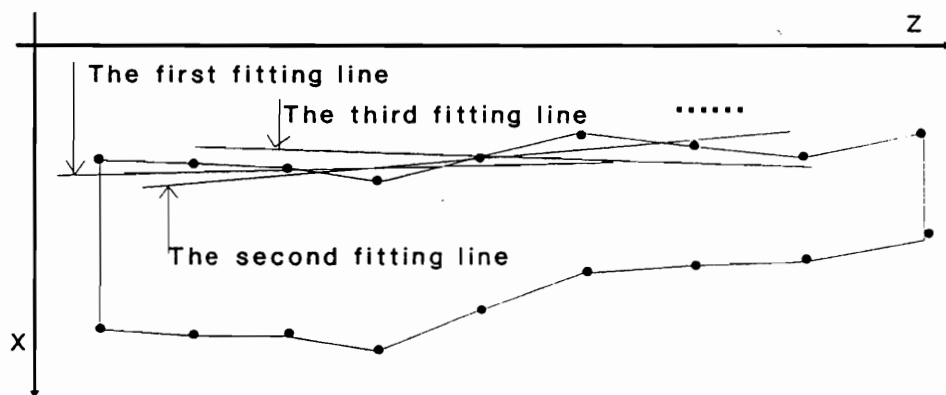
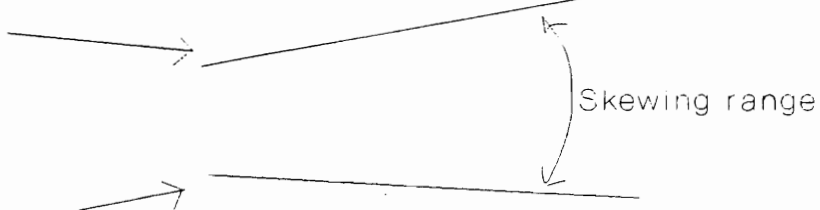


Figure 4.7. Projection of the log onto the X-Z plane.



A line parallel to the fitting line with the maximum slope



A line parallel to the fitting line with the minimum slope

Figure 4.8. Determination of the skewing range.

(maximum skewing angle - minimum skewing angle)/2.  
 Then the program starts with the minimum skewing angle and repeats for each skewing increment until the maximum skewing angle is reached. Note that a very small skew angle can result in a significant log movement at the far end of long logs.

Skewing the log is implemented by the following assignments written in the "C" programming language:

```
sine=sine(SkewAngle);
cosine=cos(SkewAngle);
for ( i=0; i<NoCrossSec; i=i+1) {
    for( j=0; j<NoScanPts; j=j+1) {
        SXY.X[i][j]=RXY.X[i][j]*cosine - RXY.Z[i][j]*sine;
        SXY.Z[i][j]=RXY.X[i][j]*sine + RXY.Z[i][j]*cosine;
        SXY.Y[i][j]=RXY.Y[i][j];
    }
}
```

Where:

```
SkewAngle    =  skewing angle;
sine         =  sine of skewing angle;
cosine       =  cosine of skewing angle;
NoCrossSec   =  number of cross sections;
NoScanPts    =  number of scanning points on each
                cross section;
```

RXY.X[i][j] = X coordinate of scanning point j on  
cross section i after rotation;

RXY.Y[i][j] = Y coordinate of scanning point j on  
cross section i after rotation;

RXY.Z[i][j] = Z coordinate of scanning point j on  
cross section i after rotation;

SXY.X[i][j] = X coordinate of scanning point j on  
cross section i after skewing;

SXY.Y[i][j] = Y coordinate of scanning point j on  
cross section i after skewing;

SXY.Z[i][j] = Z coordinate of scanning point j on  
cross section i after skewing.

#### 4.3.2 Description of the DP Algorithm

Dynamic programming is an optimization procedure that is particularly useful for problems requiring a sequence of interrelated decisions. A sequence of decisions, which in turn results in a sequence of situations, is performed by recursive calculations to maximize overall effectiveness. By proper formulation, the optimization problem of log breakdown can be defined as a dynamic programming problem, and a recursive equation can be established to find the optimum value or volume of the log.

Figure 4.9 shows cross sections as seen from one end of the log. Suppose that points  $X_1, X_2, X_3, \dots, X_n$  on

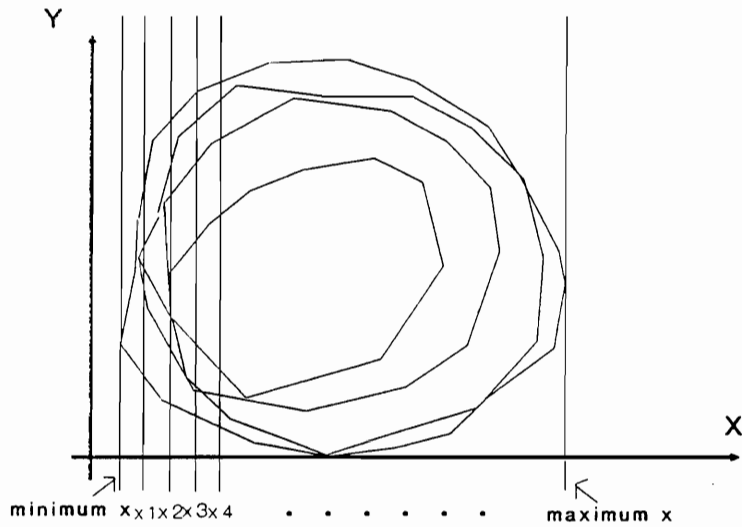


Figure 4.9. Saw placements on cross sections of a log as seen from one end of the log.

the X axis are possible saw placements. In fact, any place on the X axis could be a possible saw placement so there can be an infinite number of possible saw placements. Unfortunately, no dynamic programming algorithm is able to search for the optimum solution from an infinite number of saw placements. To make the problem solvable, a set of discrete points must be chosen from the infinite number of saw placements. To establish the DP recursive formulation, the interval between possible saw placements must be a common denominator of the combined thicknesses of the slabs, flitches, cants, saw kerf, sawing variation, planing allowance, and shrinkage. A smaller interval results in more points in the set, and the better the chance will be

of finding the optimum solution. However, the computing time will be much longer.

To illustrate how the dynamic programming algorithm works, a network representation of sawing decisions is shown in Figure 4.10. Each node corresponds to a possible saw placement and each arc corresponds to a possible decision made with the left end node of the arc representing one saw placement and the other saw placement being represented by the right end node of the arc. Suppose lumber thicknesses of 1, 2, and 3 inches are allowed. Also, assume that the saw kerf is  $1/8$  inch and no other production parameters are considered. The number of possible saw placements depends on the distance between maximum  $X$  and minimum  $X$  and the interval between nodes. For instance, if the distance between maximum  $X$  and minimum  $X$  in Figure 4.10 is 20 inches and the interval between

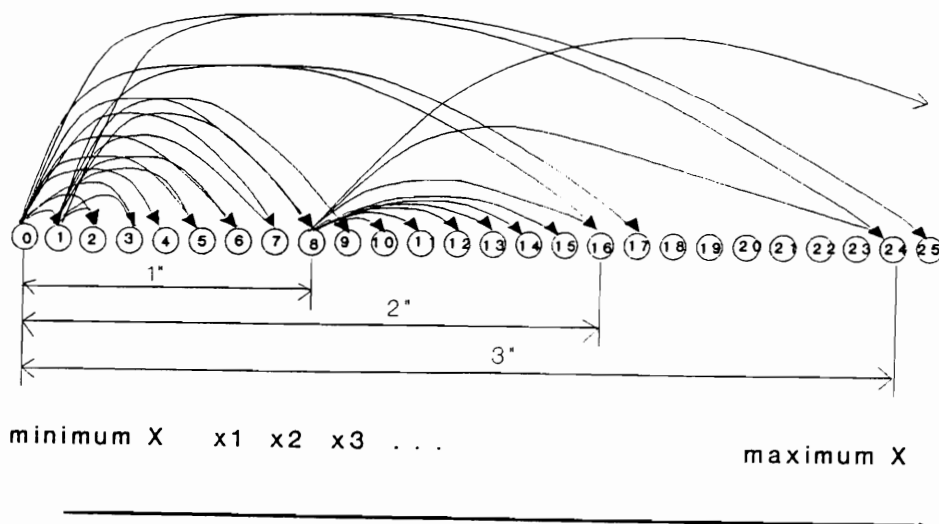


Figure 4.10. A representation of potential sawing decisions for nodes 0, 1, and 8.



possible saw placement points is  $1/8$  inch, then the number of possible saw placements will be:

$$20 \div (1/8) + 1 = 161.$$

At each possible saw placement, the next saw placement will be chosen from one of ten possible saw placements in a decision set. The ten possible saw placements are calculated as the following:

1) If the next saw placement does not make an acceptable board, it must be shifted forward. The possible shifts are  $1/8$ ,  $2/8$ ,  $3/8$ ,  $4/8$ ,  $5/8$ ,  $6/8$ , or  $7/8$  inch. In total, there are seven possible placements which do not make any acceptable piece (narrowest acceptable thickness divided by saw kerf and then minus one).

2) If the next saw placement produces an acceptable slab, flitch or cant, then the number of possible saw placements making an acceptable piece is equal to the number of thicknesses of finished lumber allowed plus the possible number of cant thicknesses. In this example there are three thicknesses and cant sawing is not considered. Therefore, the total number of possible decisions in the decision set is the sum of all possible placements, which equals ten.

If the value of each arc represents the value of the piece between the two nodes connected by the arc, then the problem of log breakdown optimization can be viewed as a longest path problem. If the distance between the two

nodes is less than the narrowest acceptable piece, the value of the piece is zero. Otherwise, the value of the piece is determined by an algorithm similar to one that will be discussed at the edging level. The objective is to find the path representing the largest value from the left-most node (minimum X) to the right-most node (maximum X) by going through some of the nodes between minimum X and maximum X. There are several methods that can be applied to solve the problem. One way is to use an exhaustive enumeration method. This is used by most existing programs using the simulation technique, such as BOF. One shortcoming of such a technique is that the computing time is extremely long since there are huge numbers of possible paths that need to be compared. Therefore, in order to reduce computing time, the thickness assortments and sawing patterns are limited and the saw placement shifting is not allowed in those programs. One other way is to use network techniques. There is no general algorithm to solve the longest path problem in a network. When a network is acyclic (no cycles in the network), the longest path problem happens to possess the same properties as an acyclic network, so the problem can be simply solved by reordering the sequence of the nodes and then performing calculations recursively. Since this concept is the central idea of dynamic programming, the most efficient way to solve the longest path problem of an acyclic network

is to use dynamic programming.

Dynamic programming (DP) can provide the optimum solution with much less effort than an exhaustive search. Dynamic programming divides the original problem into small portions, called stages in DP terminology, with a number of conditions, called states, and with a policy decision to be made at each stage. Dynamic programming finds the optimum solution by making a sequence of interrelated decisions. For the longest path problem, each node is a stage. There is only one state at each stage which is the remaining distance from the node to either minimum X or maximum X, depending upon whether the process is moving "forward" or "backward". At each stage, the decision to be made is to choose an optimum arc which links the node to an optimum path from the starting node to that node.

The decision making process is based on the principle of optimality. For the longest path problem, the principle of optimality can be stated as follows:

The best path from the minimum X to the maximum X has the property that, for any node between maximum X and minimum X, whatever the decisions made to get to that node, the remaining path to the maximum X, starting at the node, must be the best path from that node to the maximum X.

For example, if the optimum path from point X6 to destination maximum X has been decided, whatever the path

from minimum X to X6 is, the path from X6 to maximum X will always be optimum. It is this principle that enables dynamic programming to solve the problem more efficiently. Dynamic programming starts from the first stage (or last stage) and makes a decision resulting in the optimum solution for that stage. It then precedes forwards (or backwards) to the next stage, making the current optimum solution from the preceding one, until the entire problem is solved.

The optimum sawing pattern is obtained when the longest path is known. For example, suppose the dynamic programming algorithm came up with the longest path of minimum X  $\rightarrow$  X32  $\rightarrow$  X40  $\rightarrow$  X56  $\rightarrow$  X80  $\rightarrow$  X82  $\rightarrow$  X98  $\rightarrow$  X114  $\rightarrow$  X130  $\rightarrow$  maximum X. This yields six pieces in the optimum sawing pattern, which are 1, 2, 3, 2, 2, and 2 inches in thickness. The piece between X82 and X98 is shifted  $2/8$  inch towards the right rather than being immediately adjacent to the piece X56  $\rightarrow$  X80. The opening face is at X32 and the right-most face is at X130.

Figure 4.11 shows the optimum solution.

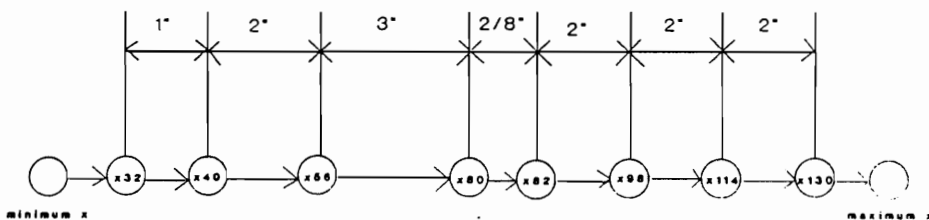


Figure 4.11. Optimum solution for an illustrative example.

In the previous example, the starting point minimum X is at one end of the cross sections and the destination point maximum X is at the other end. In fact, these two boundary points can be in the positions where the narrowest and the shortest acceptable lumber faces are yielded, instead of the two ends. In that case, the sawing range is reduced and the computing time is significantly saved. If the log is a cylinder or truncated cone, the positions can be located easily. However, if the log is S-shaped or any other irregular shape, locating the two points can be very time consuming. Therefore, the program uses the maximum and minimum X coordinates of all cross sections as the two end points.

If there are N stages (nodes) and M decisions (arrival arcs) to be made at each stage, the dynamic programming algorithm must search  $N \cdot M$  combinations to find the optimum solution, which is much more efficient than an exhaustive search. However, if N is large, such as when the log size is large or the interval between stages is small, the dynamic programming algorithm can be very time consuming. One way to reduce computing time is to reduce M. For example, in Figure 4.10 all possible sawing placement shifts, which are arcs less than one inch thickness, were given. In fact, only the 1/8-inch shift is necessary. If two interval shifts are included in an optimum path, like the 2/8-inch shift in Figure 4.11, the algorithm will

guarantee two adjacent 1/8-inch shifts will be in the optimum solution, which is the same as one 2/8-inch shift. Therefore, for the example in Figure 4.10, the number of decisions to be made at each stage can be reduced from 10 to 4, resulting in a 60% reduction in computing time.

#### 4.3.3 Mathematical Definition and Formulation

The following definitions are used to formulate the log breakdown problem as a forward dynamic programming problem.

Let:

- n = stage variable of the DP problem, which is the sequential number of saw placements 0, 1, 2, 3, ..., N, starting from the minimum X coordinate of all cross sections;
- N = number of stages, which is obtained by converting  $(\text{maximum } X - \text{minimum } X)/k$  to the nearest integer; where maximum X is the maximum X coordinate of all cross sections and minimum X is the minimum X coordinate of all cross sections;
- k = interval between saw placements (interval between stages), which must be a common denominator of saw kerf and thickness assortments;

- $S$  =  $n \times k$ , the portion of the log available for sawing at a stage, which is the state variable of the DP problem and is equal to the distance between stage  $n$  and stage 0;
- $A$  = the set of total sawing position variables at all stages, which includes all thickness assortments, cant thickness assortments, intervals and its 2, 3, ..., (thinnest lumber thickness)/ $k$  - 1 times;
- $a_n$  = the decision taken at stage  $n$ ;
- $R_n(a_n)$  = the value obtained if decision  $a_n$  is taken at stage  $n$ ; it is the immediate value of the piece cut from the portion of the log between stage  $n$  and decision  $a_n$ . This value is the output of the edging level.
- $F_n(S)$  = the optimum overall value obtained from the portion between stage 0 and stage  $n$  at state  $S$ . Actually, for this problem there is only one state at each stage.

Then a forward recursive relationship can be set up as:

$$\begin{aligned}
 F_0(0) &= 0; \\
 F_n(n \cdot k) &= \text{MAX}_{a_n \in A} \left\{ R_n(a_n) + F_{n - \frac{a_n}{k}} \left( \left( n - \frac{a_n}{k} \right) \cdot k \right) \right\} \quad n=1, \dots, N
 \end{aligned}$$

#### **4.4 Edging Pattern Optimization**

Level 1 of the program, the log breakdown level, gives X coordinates for the sawing planes creating the left and right faces of the piece being edged. The next step, level 2 or the edging level, determines the optimum edging pattern and the optimum value of the piece. It does this by first finding the two faces of the piece, merging the two faces together, determining the pitching range, pitching the piece, and finally using a dynamic programming algorithm similar to that in the first level to obtain the optimum solution. The optimum value is returned to the first level and the optimum edging pattern is stored. Figure 4.12 shows the detailed flow chart of the edging level, but the relationship between this level and the trimming level is not shown.

##### **4.4.1 Determining the Two Faces of a Piece**

A face of the piece is formed by the intersection of the sawing plane found in Level 1 at a given X coordinate for all cross sections. The procedure used to find intersection points of the sawing plane with all cross sections involves five steps. These steps will be explained in terms of the example cross section shown in Figure 4.13.

Step 1: If the sawing plane passes through a scanning point, the upper (or lower, depending upon whether or not



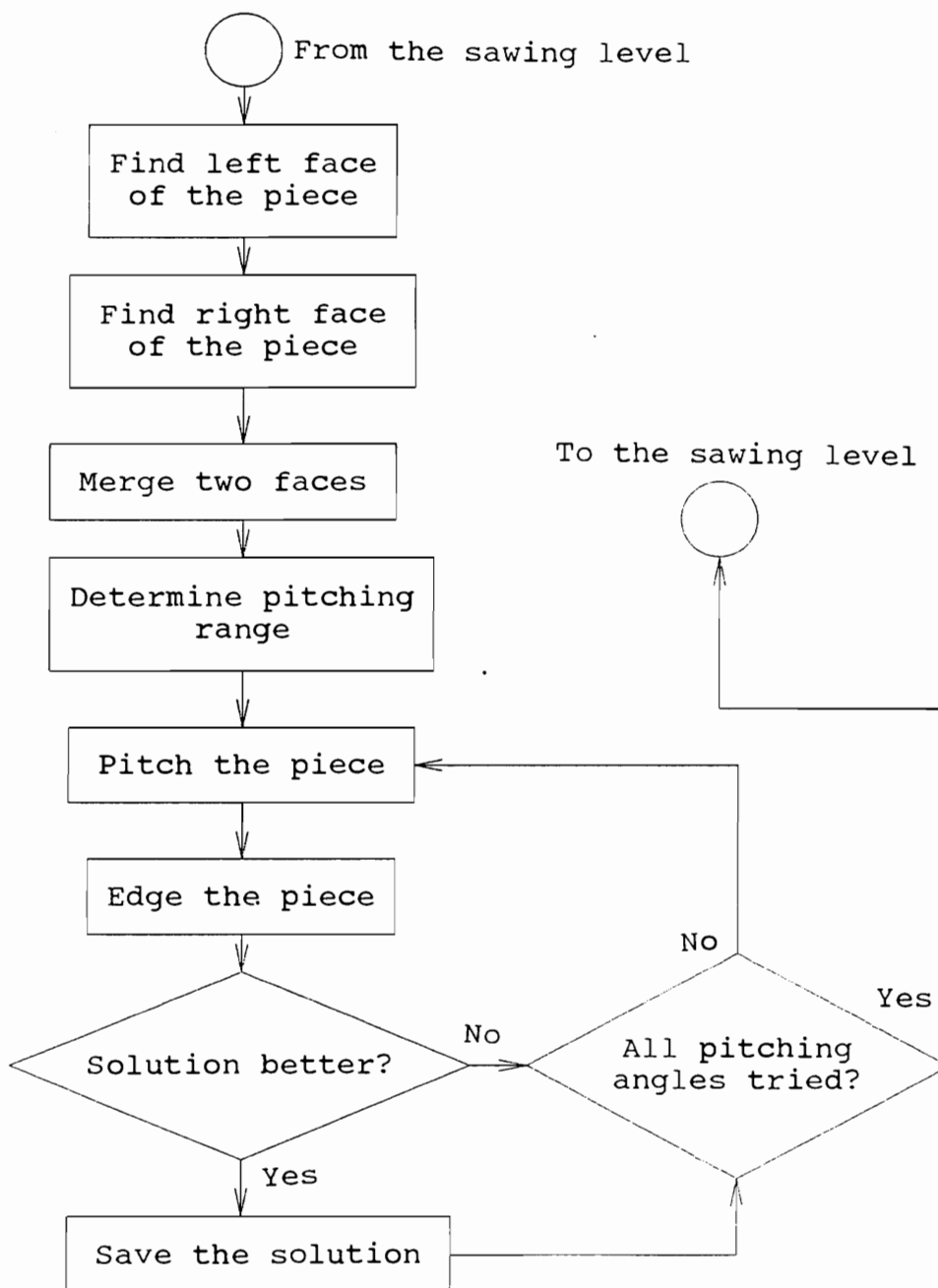


Figure 4.12. Flowchart of the edging level procedure.

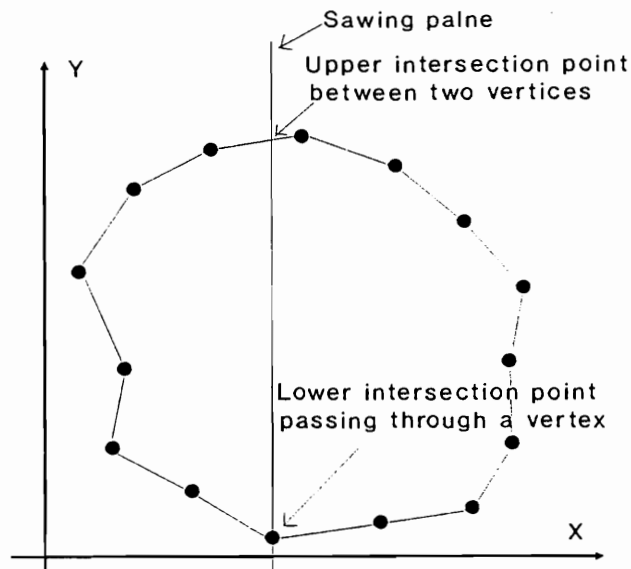


Figure 4.13. Intersection of a sawing plane with a cross section.

an intersection point has been found before) intersection point is equal to the first scanning point. If this is the lower intersection point, go to Step 5. Otherwise, set a flag indicating the upper intersection point has been found.

Step 2: If the sawing plane falls in the gap between two scanning points, the upper or lower intersection point is on the straight line connecting the two scanning points, depending upon whether or not an intersection point has been found before. If this is the lower intersection point, go to Step 5. Otherwise, set a flag indicating the upper intersection point has been found.

Step 3: If neither of the previous two situations happens, do nothing. Examples of logs in which this might occur include ones with significant amounts of taper, sweep, or crook. In these cases, a sawing plane might only intersect a subset of all cross sections.

Step 4: Repeat Step 1 to Step 3 until all scanning points on the cross section have been tried. If no intersection point is found, set a special number such as -9999 indicating no intersection point. If there is only one intersection point, let the upper and the lower ones be the same.

Step 5: Repeat Step 1 to Step 4 until all cross sections have been tried.

To simplify the successive calculations and reduce the time required for data transformation, the two faces are merged into one. When merging the two faces together, wane allowances on both width and thickness are taken into account. In the version of SAW3D used in this thesis, all lumber sizes use only wane allowances based on the narrowest (4 inches) and thinnest (1 inch) lumber sizes typically allowed in the industry. Merging the two faces into one involves five steps. They are explained using the example shown in Figure 4.14.

Step 1: If any one of the four potential intersection points is not on the cross section, i.e., it is not an intersection point, eliminate the segment between this

cross section and the immediately previous cross section by not adding the interval between the two cross sections to the length of the piece. Then go to Step 5.

Step 2: Find the merged upper and lower points, which make: a) both  $\delta X_u$  and  $\delta X_l$  less than or equal to the wane allowance on thickness, and b) the sum of  $\delta Y_u$  and  $\delta Y_l$  less than or equal to the wane allowance on width.

Step 3: If the distance between the merged upper and lower points is greater than or equal to the narrowest acceptable lumber width, add the interval between cross sections to the length of the piece. Otherwise, go to Step 5 if the previous length is greater than or equal to the

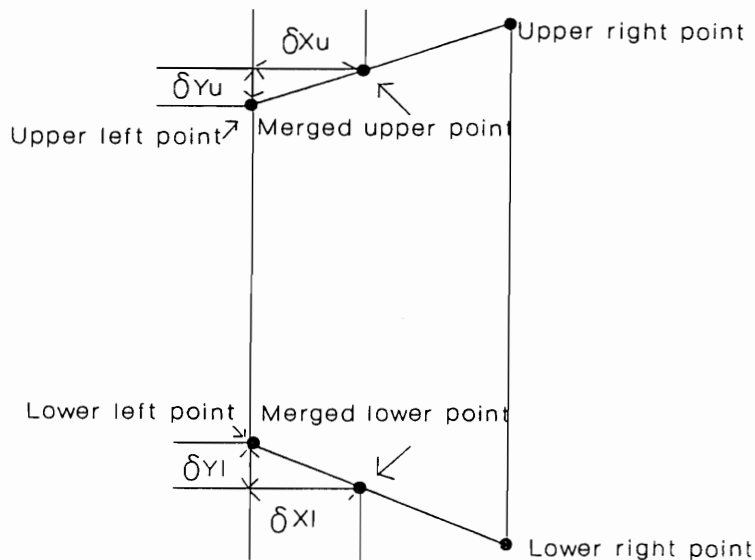


Figure 4.14. Merging two faces into one for edging.

shortest acceptable lumber length, or initialize the length (let it be zero) and go to Step 4 if the previous length is less than the shortest acceptable length.

Step 4: Repeat Step 1 to Step 3 until all cross sections have been tried.

Step 5: End the procedure.

#### **4.4.2 Pitching the Piece**

Pitch refers to orienting the piece so that the edging plane is at an angle to the center line of the piece. There are two steps involved in pitching the piece. The first step is to find the pitching range, and the second step is to pitch the piece.

##### **4.4.2.1 Determining The Pitching Range**

For a cylinder, the edging lines are all parallel to the central axis of the piece. For an irregular shaped piece, however, there could be an infinite number of possible edging positions. Since pitching a piece means tilting the piece vertically into a certain edging position, it is necessary to find a pitching range and then choose a finite number of pitching angles for an irregular shaped piece.

The procedure used to determine the pitching range is similar to that used to determine the skewing range (Figure 4.8). A group of points is taken on an edge of the merged

piece, and then a straight line equation is determined using simple linear regression. The number of points in the group depends on the interval between cross sections and the shortest acceptable lumber length. The distance between the first point and the last point should be greater than or equal to the shortest acceptable lumber length. A set of straight lines is obtained in this way by taking a series of groups of points, with each group of points containing the same number of points but shifted one point towards the other end of the piece until the number of remaining points is less than that needed to make a group. The same procedure is used to find straight lines on the opposite edge. Next, the pitching range is determined by choosing the maximum and minimum slopes among all those straight lines.

#### **4.4.2.2 Pitching The Piece**

There is an infinite number of possible pitching positions between the maximum and minimum pitching angles obtained in the previous sections. Realistically, the program can only use some discrete positions from the pitching range. While a greater number of pitching positions increases the chances of obtaining the optimum solution, the computing time will be longer. Therefore, a pitching angle increment is entered by the operator. Given a pitching angle, the piece is mathematically pitched by

the following assignments written in the "C" programming language:

```
sine = sin(angle);
cosine = cos(angle);
for(i=1; i<=intsecnp.NoZ; i=i+1) {
    intsec.Zu[i]=intsecnp.Zu[i]*cosine - intsecnp.Yu[i]*sine;
    intsec.Zl[i]=intsecnp.Zl[i]*cosine - intsecnp.Yl[i]*sine;
    intsec.Yu[i]=intsecnp.Zu[i]*sine + intsecnp.Yu[i]*cosine;
    intsec.Yl[i]=intsecnp.Zl[i]*sine + intsecnp.Yl[i]*cosine;
}
```

Where:

angle = pitching angle;

sine = sine of the pitching angle;

cosine = cosine of the pitching angle;

intsecnp.NoZ = number of points on one edge of the piece;

intsecnp.Zu[i] = Z coordinate of the ith point on the upper edge of the piece before pitching;

intsecnp.Zl[i] = Z coordinate of the ith point on the lower edge of the piece before pitching;

intsecnp.Yu[i] = Y coordinate of the ith point on the upper edge of the piece before pitching;

intsecnp.Yl[i] = Y coordinate of the ith point on the lower edge of the piece before pitching;

`intsec.Zu[i]` = Z coordinate of the *i*th point on the upper edge of the piece after pitching;  
`intsec.Zl[i]` = Z coordinate of the *i*th point on the lower edge of the piece after pitching;  
`intsec.Yu[i]` = Y coordinate of the *i*th point on the upper edge of the piece after pitching;  
`intsec.Yl[i]` = Y coordinate of the *i*th point on the lower edge of the piece after pitching.

#### 4.4.3 Mathematical Formulation for the Edging Problem

After the position of the piece has been decided, a dynamic programming algorithm similar to that used to optimize the sawing pattern is used to optimize the edging pattern. To make it easier to compare with the DP formulation of the first level, the same notations are adopted here, but their definitions are modified to reflect the edging operations.

Let:

`n` = stage variable of the DP formulation, which is the sequential number of edging saw placements 0, 1, 2, 3, . . ., `N`, starting from the minimum Y coordinate on the piece;  
`N` = number of stages, which is obtained by converting  $(\text{maximum } Y - \text{minimum } Y)/k$  to the nearest integer; where maximum Y is the maximum Y coordinate on the piece and



minimum  $Y$  is the minimum  $Y$  coordinate on the piece;

- $k$  = interval between edging saw placements (interval between stages), which must be a common dominator of edging saw kerf and width assortments (and thickness assortments for cant sawing);
- $S$  = state variable of the DP formulation, which is the portion of the piece being analyzed for edging, equal to  $n \times k$ , which is the distance between stage  $n$  and stage 0;
- $A$  = a set of total edging saw position variables at all stages, which includes all width assortments, intervals between stages, and its 2, 3, 4, . . . , (smallest lumber width)/ $k$  - 1 multiplier (and thickness assortments for cant sawing);
- $a_n$  = decision taken at stage  $n$ ;
- $R_n(a_n)$  = the value obtained if decision  $a_n$  is taken at stage  $n$  and is the immediate value of the untrimmed lumber cut off the portion of the piece between stage  $n$  and decision  $a_n$ ; this value is determined in the trimming level;
- $F_n(S)$  = the optimum overall value obtained from the portion between stage 0 and stage  $n$  at state  $S$ ; for the edging problem, there is only one

state at each stage for the edging problem.

Then the recursive relationship is:

$$F_0(0) = 0;$$

$$F_n(n \cdot k) = \text{MAX}_{a_n \in A} \left\{ R_n(a_n) + F_{n - \frac{a_n}{k}} \left( \left( n - \frac{a_n}{k} \right) \cdot k \right) \right\} \quad n=1, \dots, N$$

#### 4.5 Trimming Pattern Optimization

Given the geometric profile formed by the merging of the two faces of the piece and the upper and lower edging lines which are the edging decisions made at the edging level, the trimming level determines the optimum lumber value by determining the optimum trimming pattern. There are two steps involved. The first step is to locate trimming zones, which means to find all possible rectangles containing acceptable lumber. The second step is then to trim those rectangles into finished lumber using a dynamic programming algorithm similar to those used in the log breakdown and edging levels. Figure 4.15 gives the flow chart of the trimming level.

##### 4.5.1 Locating Trimming Zones

To reduce both the computing time and memory space required by the dynamic programming algorithm, any portions

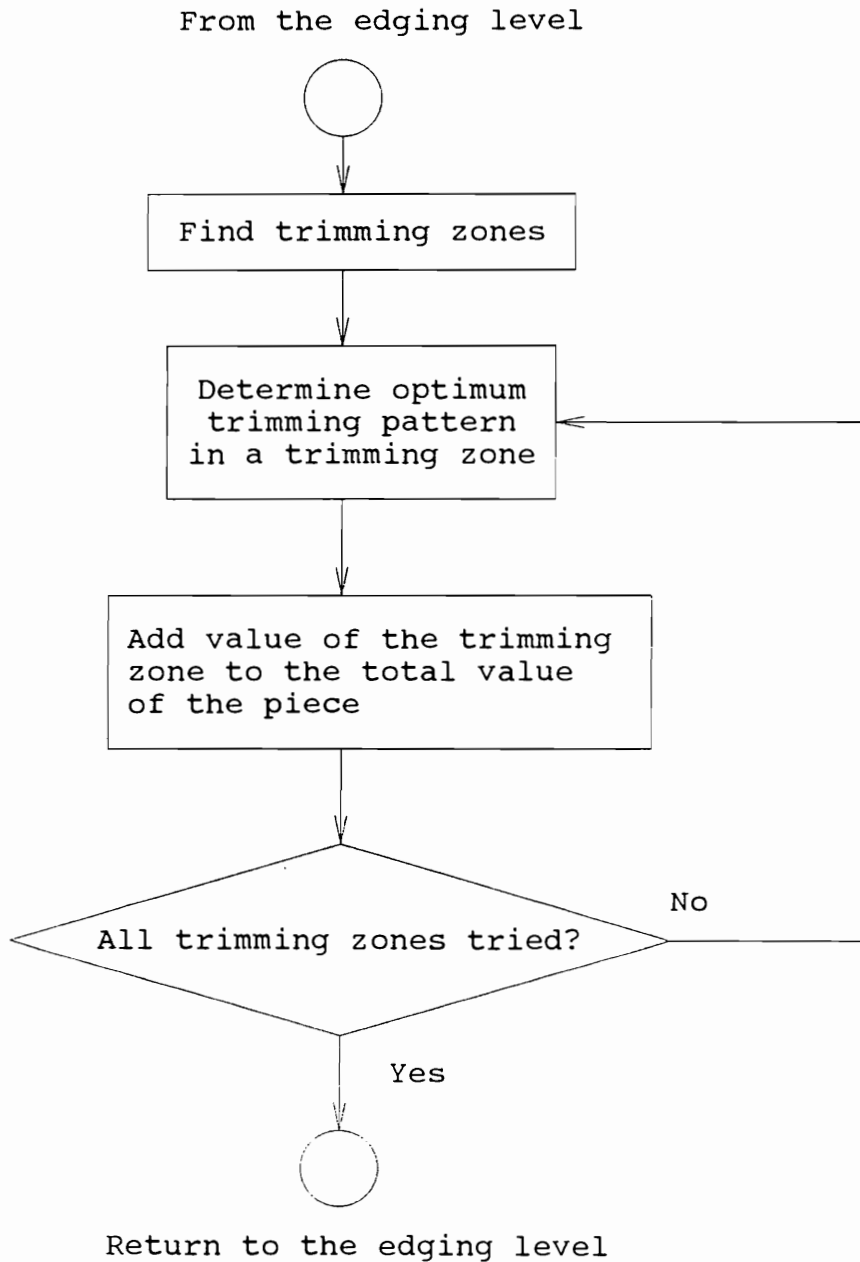


Figure 4.15. Flowchart of the trimming procedure.

of the piece which are less than the shortest lumber allowed should be eliminated before starting DP iterations. This is done by determining acceptable trimming zones (Figure 4.16).

The idea of locating trimming zones is straightforward, although its implementation in code may not be so simple. The idea is to locate one or more rectangles greater than or equal to the shortest acceptable lumber length within the piece profile. The width of a rectangle is the distance between two edging lines, while its length is the distance between two boundaries of the trimming zone. The following is a three-step outline of the procedure used for locating trimming zones.

(1) Find intersection points of the upper edging line with the piece profile. If a segment between two intersection points is within the piece profile, it is

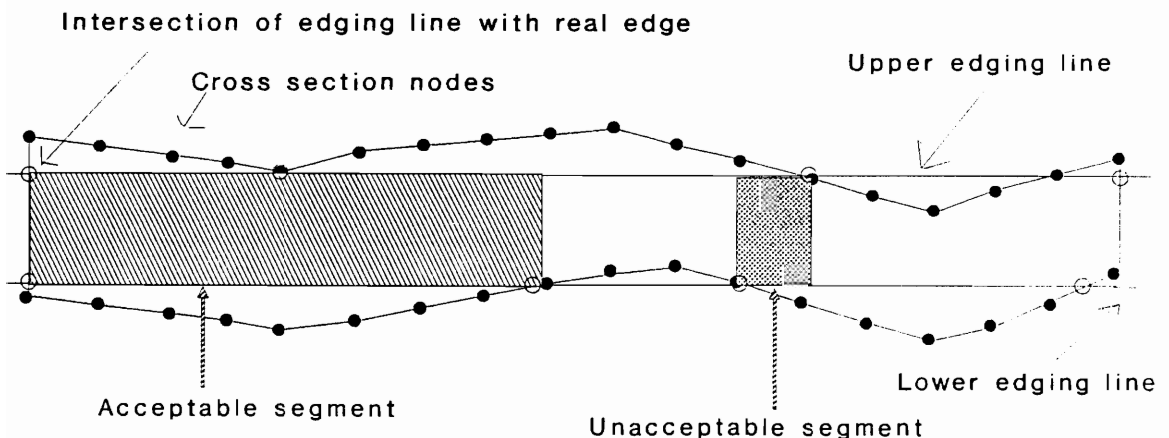


Figure 4.16. Locating trimming zones.

kept; otherwise, it is eliminated. Acceptable segments are called upper segments.

(2) Find the intersection points of the lower edging line with the piece profile. If a segment between two intersection points is within the piece profile, it is kept; otherwise, it is eliminated. Acceptable segments are called lower segments.

(3) Find where the upper and lower segments overlay. If an overlay is greater than or equal to the shortest acceptable lumber length, this overlay is a trimming zone; otherwise it should be eliminated. If there are more than one acceptable overlays, then there is more than one trimming zone. Keep all of them, and assign sequential numbers to these trimming zones.

#### **4.5.2 Mathematical Formulation for the Trimming Problem**

For each rectangle identified as a possible trimming zone, a dynamic programming algorithm similar to the previous two DPs is applied to find the rectangle yielding the optimum value. The following definitions are used in the mathematical formulation. Note that the number of stages could be too large since the length of the untrimmed lumber is very long compared with the tenth inch increments, resulting in extremely long iteration times. To reduce the number of stages, the trimming saw kerf is assumed to be zero. Under this assumption, the number of

stages can be reduced dramatically by using two feet as the interval between stages.

Let:

$n$  = stage variable, which is the sequential number of trimming saw placements 0, 1, 2, 3, . . .,  $N$ , starting from the minimum  $Z$  coordinate on the piece;

$N$  = number of stages, which is obtained by converting  $(\text{maximum } Z - \text{minimum } Z)/k$  to the nearest integer; where maximum  $Z$  is the maximum  $Z$  coordinate on the piece and minimum  $Z$  is the minimum  $Z$  coordinate on the piece;

$k$  = interval between trimming saw placements (interval between stages), which is 24 inches;

$S$  = state variable, which is the portion of the piece being analyzed for trimming, and is equal to  $n \times k$ , which is the distance between stage  $n$  and stage 0;

$A$  = a set of total trimming saw position variables at all stages, which includes all lumber length assortments, intervals between stages, and its 2, 3, . . ., (shortest lumber length)/24 - 1 multipliers;

$a_n$  = decision taken at stage  $n$ ;

$R_n(a_n)$  = the value obtained if decision  $a_n$  is taken at stage  $n$  and is the immediate value of the lumber cut from the portion of the untrimmed lumber between stage  $n$  and decision  $a_n$ ;

$F_n(S)$  = the optimum overall value obtained from the portion between stage 0 and stage  $n$  at state  $S$ ; for the trimming problem, there is only one state at each stage.

Then the recursive relationship is:

$$F_0(0) = 0;$$

$$F_n(n \cdot k) = \text{MAX}_{a_n \in A} \left\{ R_n(a_n) + F_{n - \frac{a_n}{k}} \left( \left( n - \frac{a_n}{k} \right) \cdot k \right) \right\} \quad n=1, \dots, N$$

#### 4.6 Software Implementation

A computer program, called SAW3D, was written using the "C" programming language. It implements all algorithms and methods described in the previous chapters, and runs under an MS-DOS operating system. The current version is not menu-driven since the primary task was to implement the algorithms for research purposes rather than commercial use.

##### 4.6.1 Structure of the Program

The program is constructed in a modular fashion to

make modifications and maintenance easier. Each independent module accomplishes its own task and is linked to related modules by transferring arguments and/or global variables. Figure 4.17 shows the hierarchy chart of the program which executes from top to bottom and left to right.

The construction of the program follows the flow charts described in previous chapters. One important thing which has not been discussed before is how to obtain output coordinates of optimum cutting patterns. The dynamic programming algorithms determine optimum values in the sawing, edging, and trimming operations. A trace-back process is needed to find the optimum cutting pattern. To trace back the cutting patterns resulting in the optimum value, the program has to keep all decisions in memory. There is an extremely high number of combinations of positioning, sawing, edging and trimming outputs that need to be searched to find the final solution. It is impossible to keep all those decisions in memory. However, if only those cutting patterns yielding the optimum value at each level under each position are stored in memory, the program has to refresh its memory whenever a greater value is found. This still takes a considerable amount of memory as well as too much time.

For these reasons, the program does not keep any decision made at the edging or trimming levels. After the



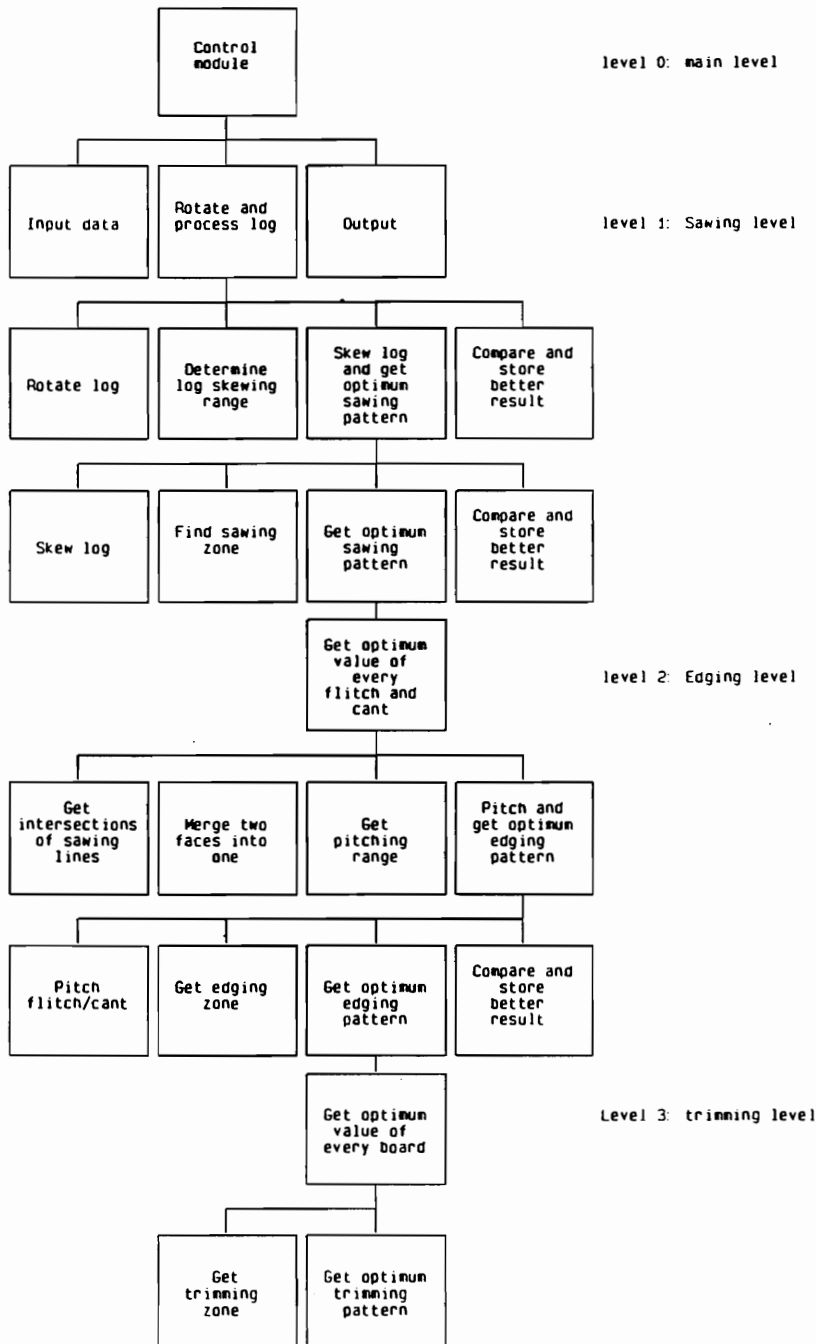


Figure 4.17. Hierarchy chart of the program.

program starts, it will keep only rotation, skewing, and pitching positions as well as sawing decisions yielding the best value returned so far, but not any edging and trimming decisions until all rotation, skewing and pitching positions are searched. When the final optimum value is obtained, the program can trace back the optimum sawing pattern and print coordinates of the optimum sawing pattern to an output file very quickly since all the sawing decisions are kept in memory. For each optimum sawing decision which yields two faces of a piece cut from the log, the program transfers the piece to the optimum pitching position stored in memory and goes through the edging operation once again to find the optimum edging pattern. This time, all edging decisions are kept in memory. Once the optimum edging value is obtained, the program traces back the optimum edging pattern and prints the coordinates of the optimum edging pattern to the output file. For the trimming level, the process is the same as in the edging level. All trimming decisions are kept in memory now, and the program traces back and prints coordinates of the optimum trimming pattern immediately after it is obtained.

#### **4.6.2 Required Information**

##### **4.6.2.1 Log Profile Data**

Log profile data provided by most scanning systems for

logs of any shape can be handled by the program. Acceptable shapes include a cylinder, truncated cone, ellipsoid with or without taper, logs with crook and sweep, 2-dimensional S-shaped logs with any cross-sectional shape, and twisted 3-dimensional S-shaped logs with any cross-sectional shape. There are no limits on diameter and length of logs imposed by the program itself, but RAM availability on the computer running the program may create limits due to memory overflow for very complicated, large, and long logs.

#### **4.6.2.2 Log and Piece Positioning**

The program allows the user to control positioning operations. If the user decides that a positioning operation is not necessary, it can be turned off. For example, a cylinder does not need to be rotated or skewed during log breakdown nor do pieces cut from the cylinder need to be pitched. In this case, the operator would turn off the rotating, skewing and pitching operations. For a log that needs to be rotated, the program allows the user to determine the rotation range and rotation angle increment. For the skewing and pitching operations, however, the positioning ranges will be determined by the program but increments of positioning should be given by the user. There are no limits on log rotation, skewing angle, or pitching angle.

#### **4.6.2.3 Sawing Methods**

The following are sawing options provided by the program:

- live sawing or cant sawing;
- full-taper or split-taper sawing;
- single lumber thickness only or multiple lumber thicknesses;
- only 1 inch or 2 inches jacket board, 2 inches lumber thickness elsewhere;
- any thickness jacket board.

The user provides control option data to tell the program what method(s) to use.

#### **4.6.2.4 Production Parameters**

Wane allowance on lumber thicknesses and widths should be given by the operator. Sawing variations for the headrig saw and edger saws are taken into account by the program, so they also need to be entered by the user. Dressing allowance and shrinkage do not need to be entered separately by the user since green target sizes of lumber are used by the program.

#### **4.6.2.5 Lumber Sizes**

Any sort of lumber size can be handled by the program. Sizes must be green target sizes. A complete size listing, including thickness, width and length of all products,

needs to be entered.

#### **4.6.2.6 Optimum Solution Searching Increments**

The interval between stages in the DP formulation is very important. It should be a common denominator of all sizes handled. For example, at the sawing level it should be a common denominator of all thicknesses, saw kerf and sawing variation. At the edging level it should be a common denominator of all widths, edging saw kerf, and edging sawing variation. The smaller the increment is, the more reliable the solution will be, but at the expense of longer computing times and more memory space.

#### **4.6.2.7 Lumber Price Table**

Prices of all lumber sizes are entered in dollars per thousand board feet (MBF). If a size is not to be produced but has to appear in the price table, the user can assign a very small value to that size so that it is not very likely that the optimum solution will contain lumber in that size.

#### **4.6.3 Input and Output**

At present, there is no interactive interface for the program. Scanning data are written in one file and the rest of the input data are written in a separate file. Those two files are the input of the program.

There are two output options, a short form and a long form. The short form contains the optimum values of the log, each flitch and each cant; the corresponding sawing, edging and trimming patterns; the optimum position of the log; and the optimum orientation of each log and cant. Appendix A gives the output from sawing a horn-down shaped log 20 inches in diameter and 10 feet long. The long form output contains all the information provided in the short form, plus profile data of the log and each piece cut from the log. The profile data allow the user to draw graphics of the output. Figures 4.18 and 4.19 show the output pictures of the horn-down shaped log.

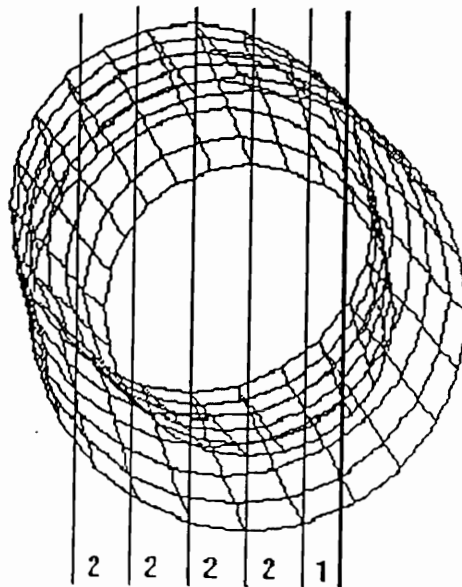


Figure 4.18. A plot of the optimum sawing sequence for a horn-down shaped log 10 inches in diameter and 10 feet long. The optimum rotation angle was 30 degrees from the horn-down position.

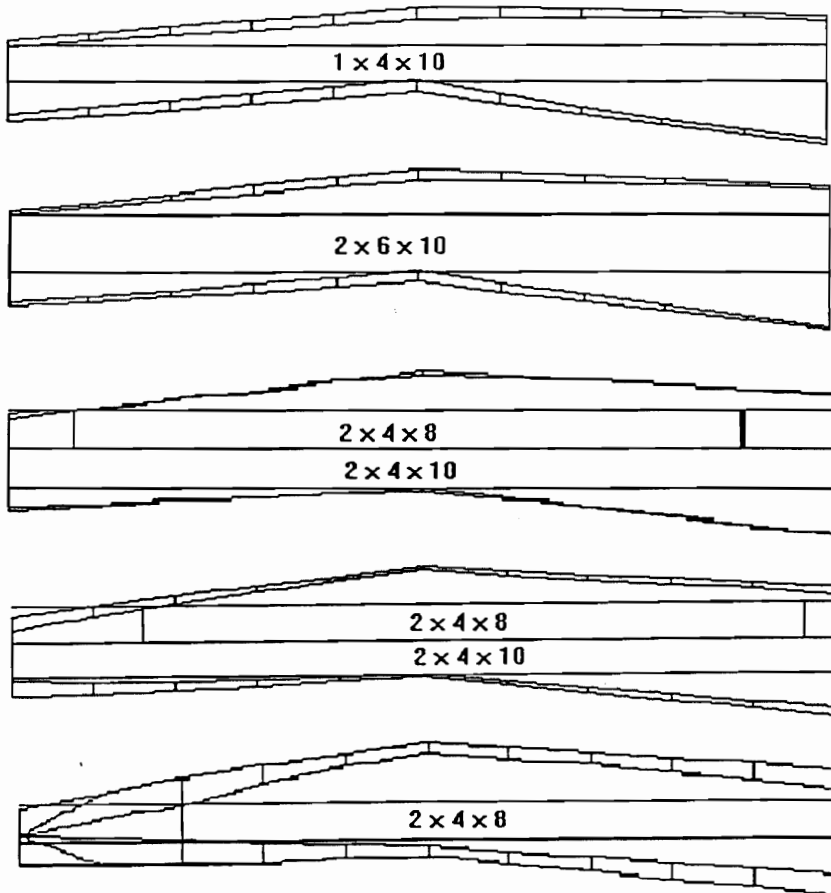


Figure 4.19. Plots of the optimum edging and trimming patterns for the five flitches sawn from the log in Figure 4.18.

## CHAPTER 5 COMPUTATIONAL RESULTS

The program was tested to (1) verify that the program works properly, (2) find primary effects of different input data, and (3) provide guidelines for more extensive tests in the near future to draw more reliable and practical rules-of-thumb.

When this sawing model was first developed, the only scanning systems available to provide the required detail in log shape were used in precision X-Y lathe chargers. This meant that only eight feet long logs could be analyzed. Therefore, twelve logs were theoretically generated to provide a representative group of log shapes and sizes. Three log shapes were chosen: ellipsoid, horn-down shaped, and 3-dimensional S-twisted shaped (Figure 5.1). Four logs of each shape were generated to provide combinations of log shapes and sizes as shown in Table 5.1. All logs have 2 inches of taper per 8 feet of length. Horn-down shaped logs have 4 inches of sweep centered on the log. All S-twisted shaped logs have 4 inches of crook at a point  $1/4$  along the length and 4 inches of crook at a point  $3/4$  of the length in the opposite direction to the first crook, and both crooks are not in the same plane so that a S-twisted shape is formed.

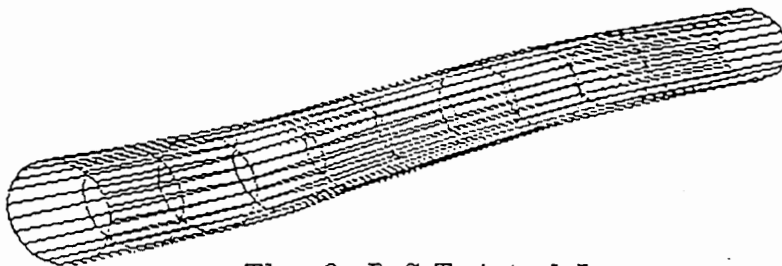




The Ellipsoid Log



The Horn-down Shaped Log



The 3-D S Twisted Log

Figure 5.1. Three log shapes tested.

Table 5.1. Various combinations of log shapes and sizes used in this study.

Legend	Shape	Small-end diameter (inches)	Length (feet)
E11	ellipsoid	10	10
E12	ellipsoid	10	20
E21	ellipsoid	20	10
E22	ellipsoid	20	20
H11	horn-down	10	10
H12	horn-down	10	20
H21	horn-down	20	10
H22	horn-down	20	20
S11	3-D, S-twisted	10	10
S12	3-D, S-twisted	10	20
S21	3-D, S-twisted	20	10
S22	3-D, S-twisted	20	20

To analyze the questions proposed in the objectives of this thesis, logs in different sizes and shapes should be tested using different scanning data, program process control options, sawing methods, and production parameters. An exhaustive test concerning all possible combinations of those issues would be nearly impossible because of time limitations. Therefore, a subset of all variables was chosen. Base data for scanning data density, production parameters and sawing method are given in Table 5.2. When testing effects of a specific variable, that variable in the base was changed to evaluate effects on value and volume recovered. American Lumber Standard sizes were used for all tests. Green target sizes are also shown in Table 5.2. Lumber prices were obtained from RANDOM LENGTHS (July 21, 1989) and are for DOUGLAS FIR, GREEN, in the Portland

area.

Table 5.2. The base data used to test the model.

---

Cross Section Interval:	2 feet
Scanning Points	
On Each Cross Section:	24 (360/15)
Sawing Increment:	0.125 inch
Edging Increment:	0.125 inch
Saw kerf(all saws):	0.125 inch
Saw Variation(all saws):	0.125 inch
Rotation Increment:	15 degree
Skewing Positions:	min., mid., max.
Pitching Positions:	min., mid., max.
Sawing Method:	live saw and cant saw
Nominal ALS size (inches):	
1    2    4    6    8    10    12	
Final Green Size (inches):	
1    1.75    3.75    5.75    7.625    9.625    11.625	
Wane Allowance:	25% on 1" thickness and 4" width

---

## 5.1 Effect of Log Rotation

### 5.1.1 General Observations On All Shapes, Sizes, and Variables

There were significant value differences between different rotation positions. Figure 5.2 gives the optimum value and volume at different rotation positions for a 3-dimensional, S-twisted shape log 10 inches in diameter and

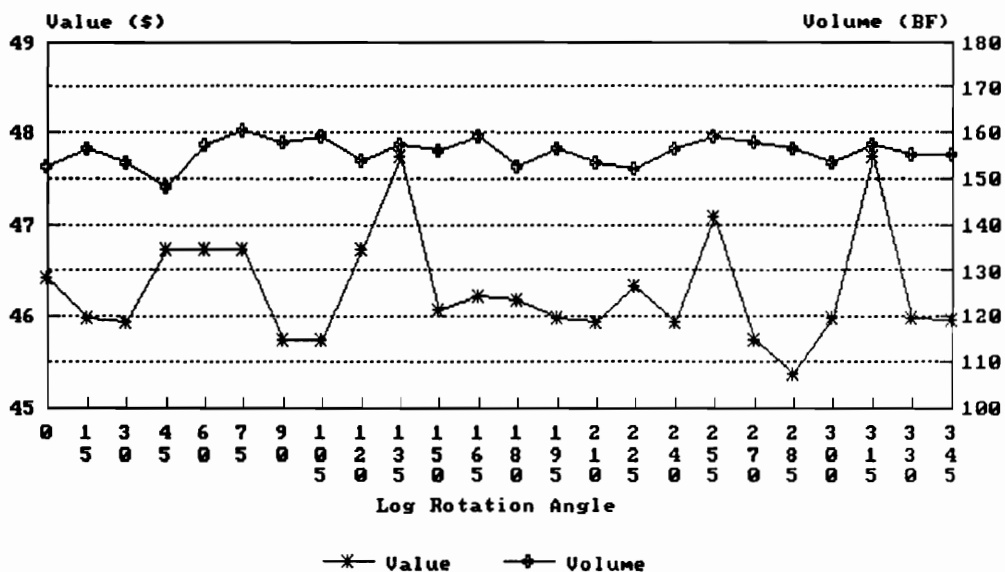


Figure 5.2. Volume and value effects of log rotation angle on a 3-dimensional, S-twisted shape log 10 inches in diameter and 20 feet in length (s12).

20 feet long (s12). The same pattern holds true for all other logs tested. Figures 5.3 and 5.4 show the percentage difference between the optimum value and the average value of all positions, and between the optimum value and the minimum value for all logs tested. The effects were more significant on the smaller, shorter, and more eccentric logs. While there were significant differences between angles, the optimum rotation angle may not be unique. Figure 5.2 shows that there are two optimum rotation angles, one at 135 degrees and another at 315 degrees. It is also interesting to note that the maximum value rotations were not the same as the maximum volume rotations.

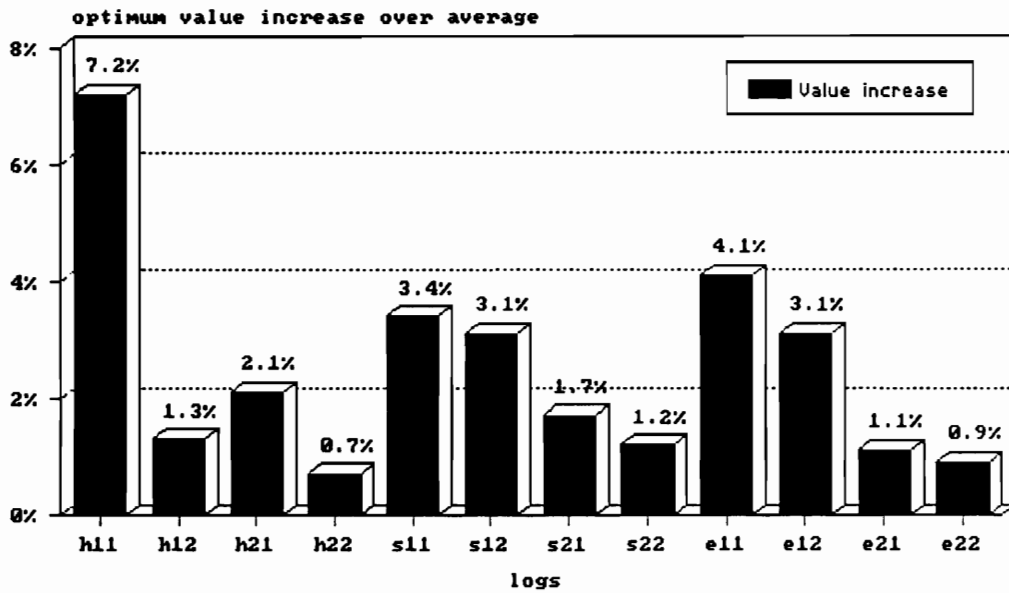


Figure 5.3. Effects of log rotation on value using the percent increase from the average value at all rotation angles to the optimum value for 12 different logs.

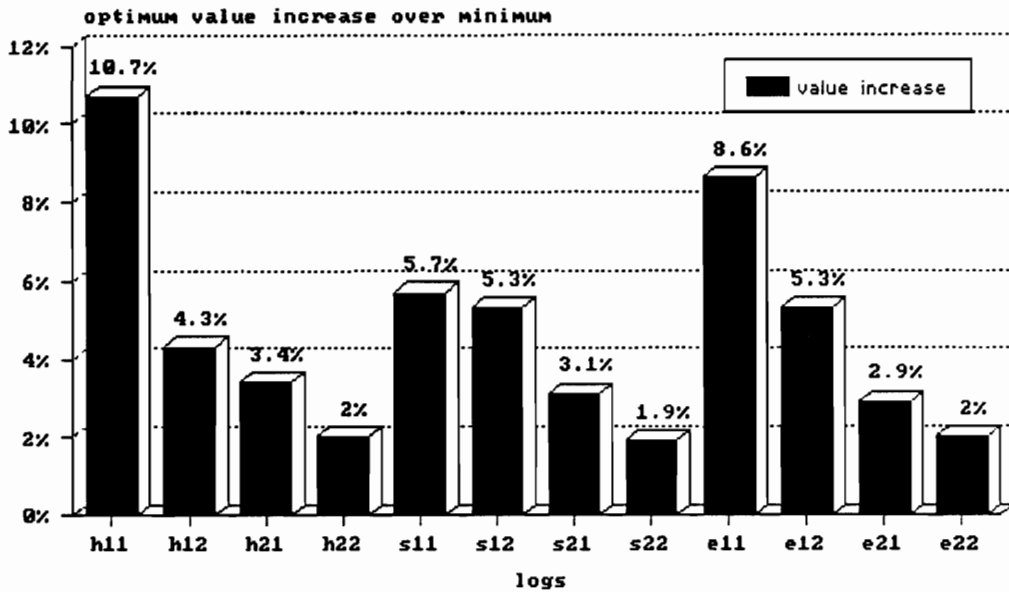


Figure 5.4. Effects of log rotation on value using the percent increase from the minimum value to the optimum value for 12 different logs. The differences were greater for the smaller, shorter logs.

### 5.1.2 Effect of Rotation Increment

Intuitively, using a smaller rotation increment increases the chance of finding a better value, but the computing time will be longer. To analyze the effect of rotation increment, rotation increments of 5 degrees, 15 degrees and 24 degrees were used to break down horn-down shaped logs and S-twisted shaped logs. These increments correspond to 72, 24, and 15 scanning points around the log's circumference, respectively.

Figures 5.5 and 5.6 show the results of sawing an S-twisted shape log 10 inches in diameter and 20 feet long (s12), using 5, 15 and 24 degrees of rotation angle

increment. The variation of values at different positions was more drastic when the rotation increment was smaller, but the trend was approximately the same (Figure 5.5). The smaller rotation increment had more chances to find a better value where the two increments had a common denominator.

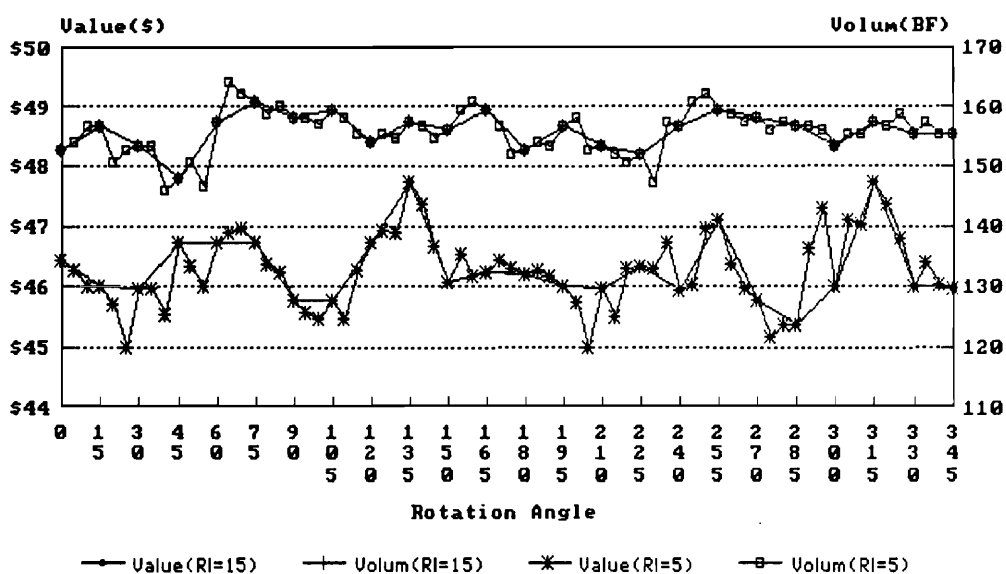


Figure 5.5. Effects of rotation angle increment on log s12 using rotation increments of 15 and 5 degrees.

When using different rotation increments which have no common denominator, such as 15 and 24 degrees, the larger increment also had a chance to find a greater optimum value than when using a smaller increment since they picked up different rotation positions. In most cases, the maximum, minimum, and average value differences resulting from using

different increments were around one percent. This implies that the rotation increment has only a minimal effect on either volume or value recovery.

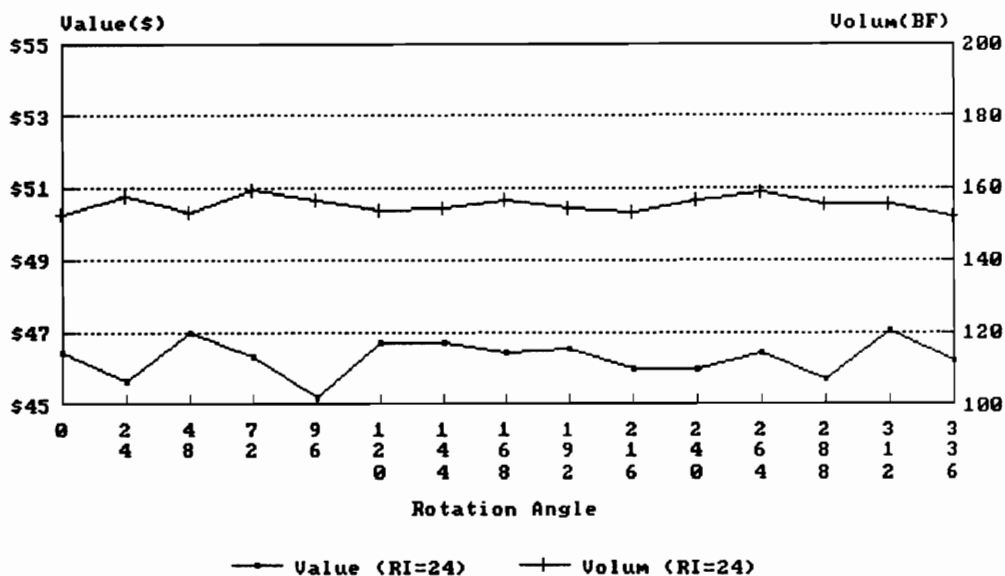


Figure 5.6. Effects of log rotation angle increment on log s12 using a rotation increment of 24 degrees.

## 5.2 Effect of Log Rotation and Skew

When testing the effect of log skew, logs were rotated to an angle and then were skewed. Comparisons were made between rotation only and rotation-skew operations. To save computing time, however, the rotation range was only from 0 to 90 degrees.

When doing log rotation and skewing operations combined, in most cases the resulting value was larger than when doing log rotation alone, especially for the maximum values (Figure 5.7). For some logs, s11 for instance, both



maximum and average value yields of the log were increased when using rotation-skew operations compared to using the rotation only operation. For some other logs, s21 for instance, the average value was increased but the maximum value did not change at all. This case occurred when values at some rotation angles were increased, but none of them was larger than the maximum value when using only the rotation operation.

### **5.3 Effect of Rotation and Pitch**

To test the effect of pitching, every piece cut from the log was pitched to find an optimum pitching position. As was the case in skewing, the rotation range was only from 0 to 90 degrees. The log skewing operation was not used here. Figure 5.8 shows the value increases when doing rotation-pitch operations for all logs tested. It can be seen that value yields of some logs were significantly increased, while others did not change at all.

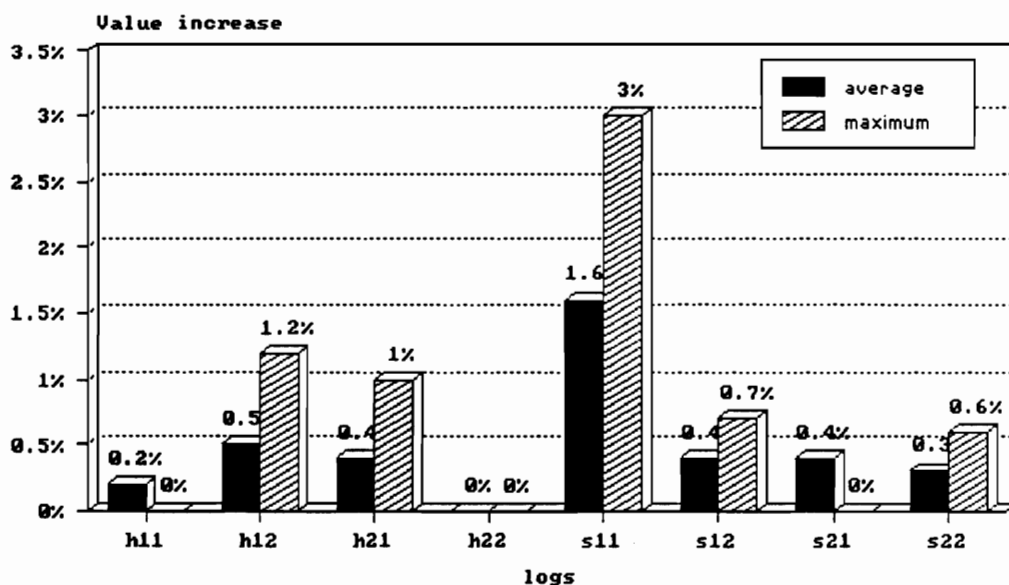


Figure 5.7. Value increases resulting from using the log skewing operation in addition to log rotation. "Average" signifies the percent increase in the average value for all rotation angles when doing both rotation and skew compared to the average value when doing rotation only. "Maximum" signifies the percent increase in the maximum value when doing both rotation and skew compared to the maximum value when doing rotation only.

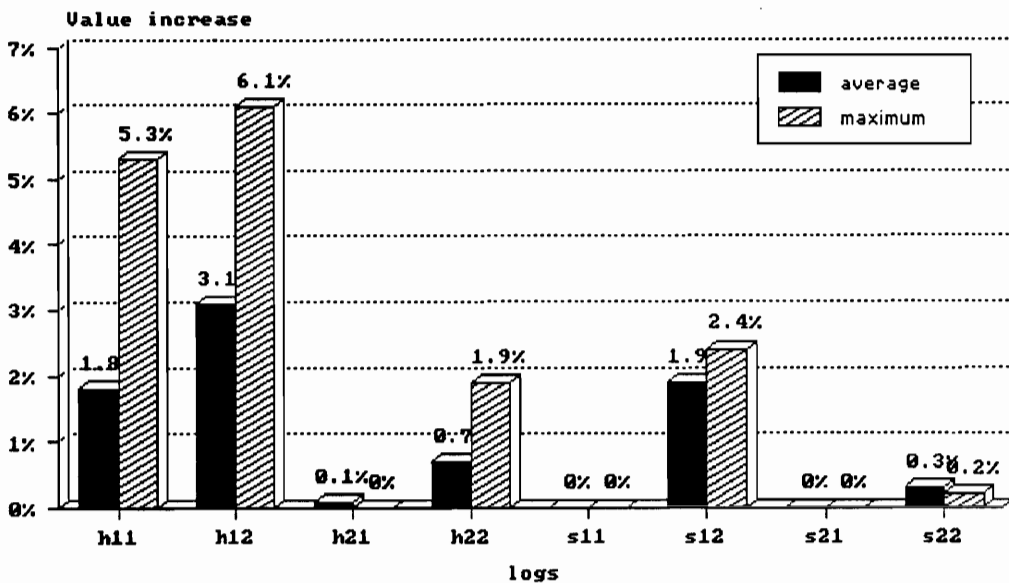


Figure 5.8. Value increases resulting from using the cant/board orientation operation (pitching) in addition to log rotation. "Average" and "maximum" have the same meanings as given in Figure 5.7. In many cases, the pitching operation increased value recovery, and usually to a greater degree than the log skewing results shown in Figure 5.7.

#### 5.4 Effect of Rotation, Skew, and Pitch

When doing all three positioning operations, i.e., rotation-skew-pitch, there will be more chances to achieve a better value, but the computing time will be much longer. Horn-down shaped logs were tested using all three positioning operations. To shorten the processing time, the rotation range was still from 0 to 90 degrees. The results showed that in all cases average and maximum values were larger than when only the rotation operation was allowed (Figure 5.9). Doing all positioning operations

also resulted in the best value yields when compared to the rotation-skew and rotation-pitch operations (Figures 5.7 and 5.8). The results in Figure 5.9 also show that the combined effects of rotation, skewing, and pitching are not simply additive.

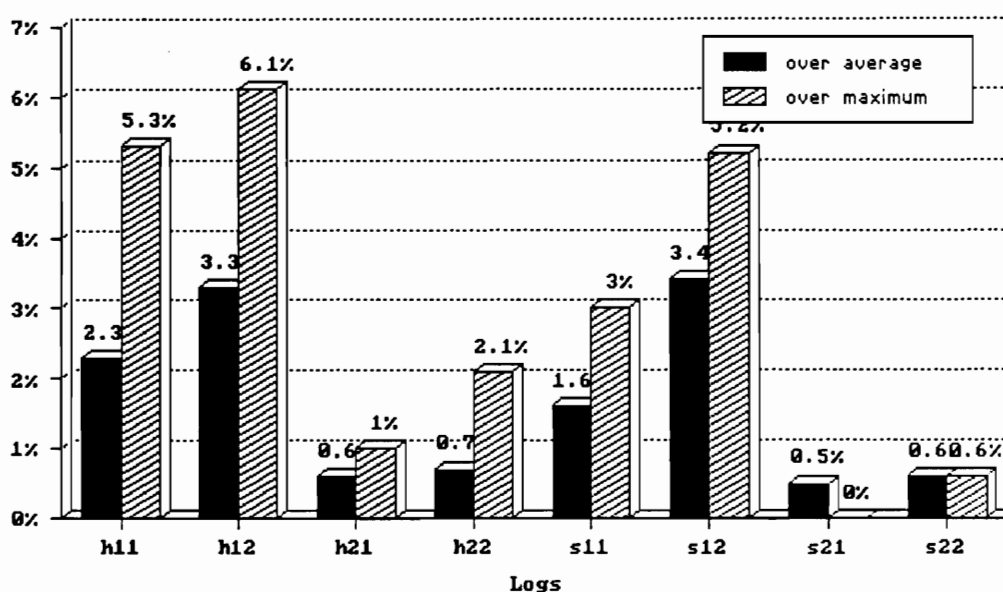


Figure 5.9. Value increases resulting from using both skewing and pitching operations in addition to log rotation. When compared to Figures 5.7 and 5.8, these values show the combined effects of the skewing and pitching operations are not merely additive.

### 5.5 Effect of Positioning On Computing Time

SAW3D was developed as a research tool rather than an on-line optimizer. Therefore, only relative times are presented in Figure 5.10. On a 386 20MHZ IBM compatible

computer, logs tested so far have execution times ranging from tens of seconds to several hours, depending upon log size and shape as well as positioning operation options. A number of techniques that promise to increase the processing speed will be presented in Chapter 6.

When more positioning operations are used, there is a greater chance of getting a better value because more positions are searched. However, more positioning operations definitely result in longer computing times. Figure 5.10 compares time differences when using different positioning operations.

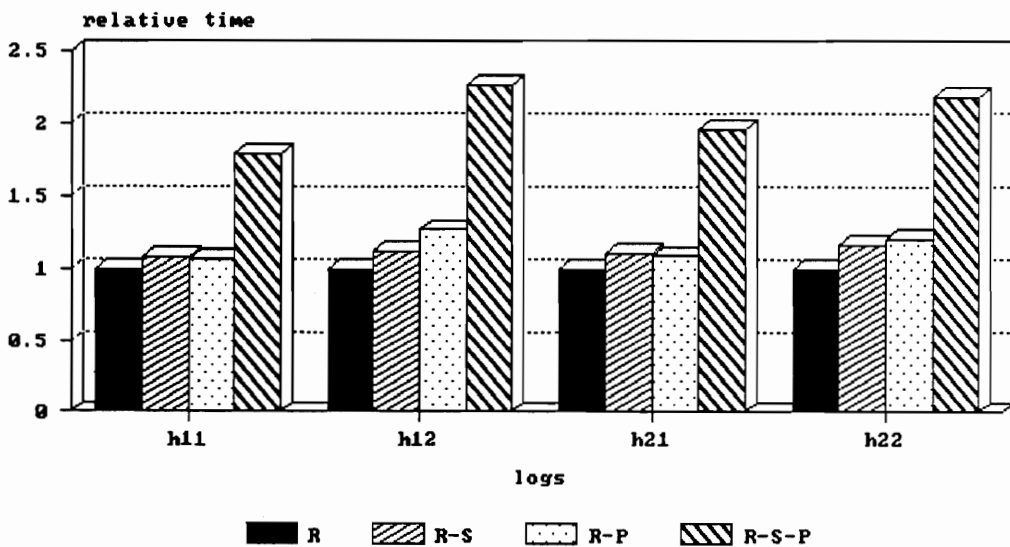


Figure 5.10. Comparison of computing time consumed by various positioning operations. R means rotation only, R-S means rotation and skewing, R-P means rotation and pitching, and R-S-P means all three operations combined.

## **5.6 Effect of Scanning Data Density**

Scanning data density is determined by the number of scanning points around the circumference of the log at every cross section as well as the interval between cross sections. Obviously, the wire-frame model used in the program is closer to the real geometry of the log if more scanning data are obtained regarding the surface of the log.

### **5.6.1 Effect of Number of Scanning Points At Every Cross Section**

Figure 5.11 shows the results of sawing log s21 using both 24 points (every 15 degrees) and 72 points (every 5 degrees) at every cross section. It indicates that there were significant value increases when more scanning data at each cross section were used. This is true for all other logs tested except h11, especially for larger, longer logs.

### **5.6.2 Effect of Interval Between Cross Sections**

Intervals of one and two feet were used to compare data density differences along the length of the log. The results showed that, in most cases, the larger cross-section interval causes the optimum value to be overestimated, especially for smaller and shorter logs.



diameter, length and taper as the truncated cones but with sweep of 4 inches. Since the logs were not perfect truncated cones, the values obtained when the BOF solutions were applied to real shapes were significantly overestimated. When sawing the log using the SAW3D algorithm without doing any positioning operations, the values were higher than when using BOF's sawing pattern; that is to say, SAW3D gives better solutions for the real log shape. When using SAW3D's algorithm and allowing all positioning operations, the values were even higher (Figure 5.12).

### **5.8 Effect of Sawing and Edging Increments**

Sawing and edging increments, which are the intervals between stages of the dynamic programming algorithms, are the most important factors affecting computing time. Using horn-down shaped logs to test the effect, it was found that the smaller increments gave more chances to find better solutions, but the computing times were much longer, especially for large logs. Values, however, did not increase as radically as the computing time (Figures 5.13 and 5.14).



## Log Size

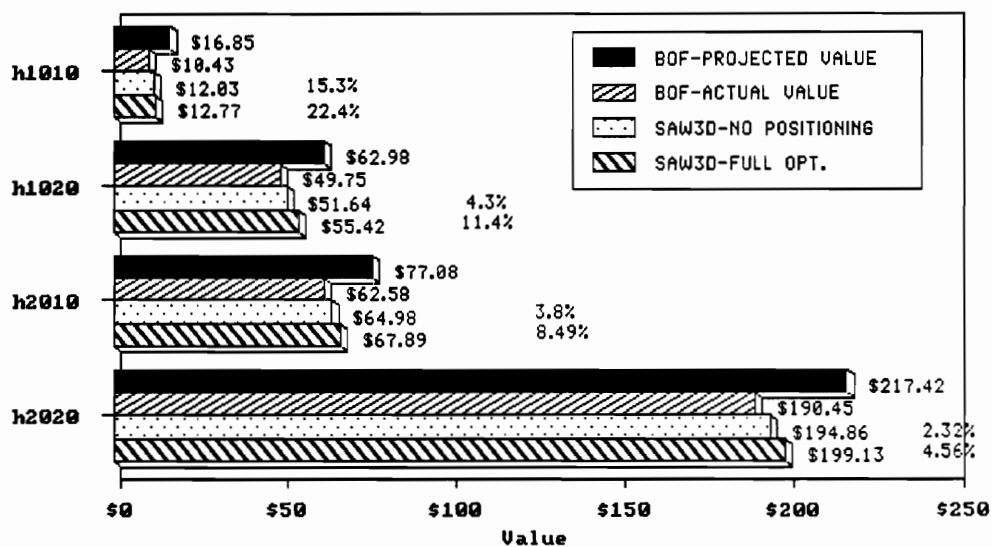


Figure 5.12. A comparison of SAW3D and BOF (Best Opening Face) results on four horn-down shaped logs. First, the logs were treated as truncated cones and sawn using BOF. Then those BOF sawing solutions were analyzed using the real shape. Since the logs were not perfect truncated cones, the values were significantly less than those originally predicted by BOF. Next, the logs were sawn using SAW3D, but no positioning operations were allowed. Finally, SAW3D was allowed to fully optimize all positioning operations.

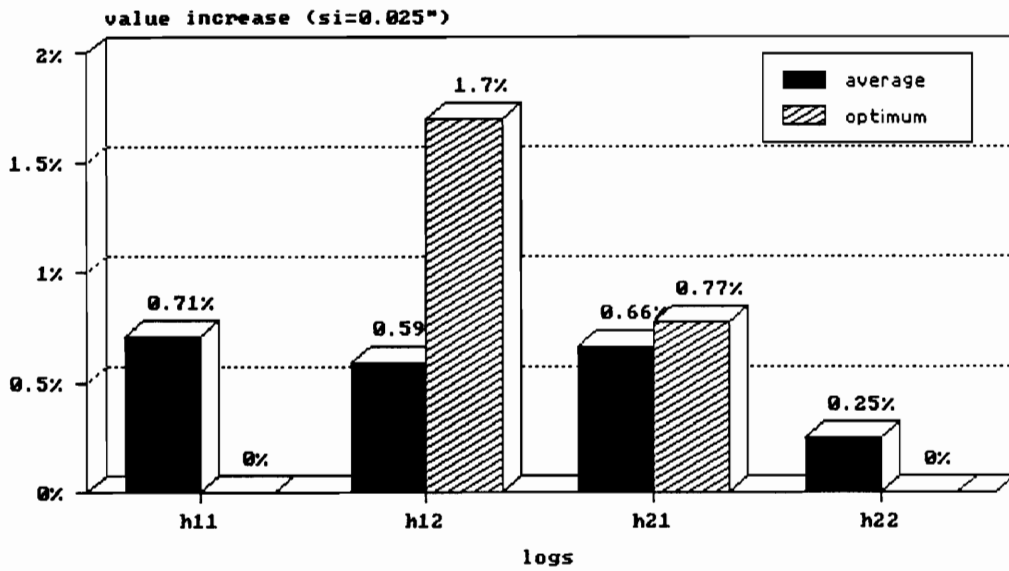


Figure 5.13. Value increases resulting from using a 0.025 inch sawing increment compared to a 0.125 inch increment. "Average" signifies the percent increase in the average value for all rotation angles when doing rotation only. "Optimum" signifies the percent increase in the optimum value when doing rotation only.

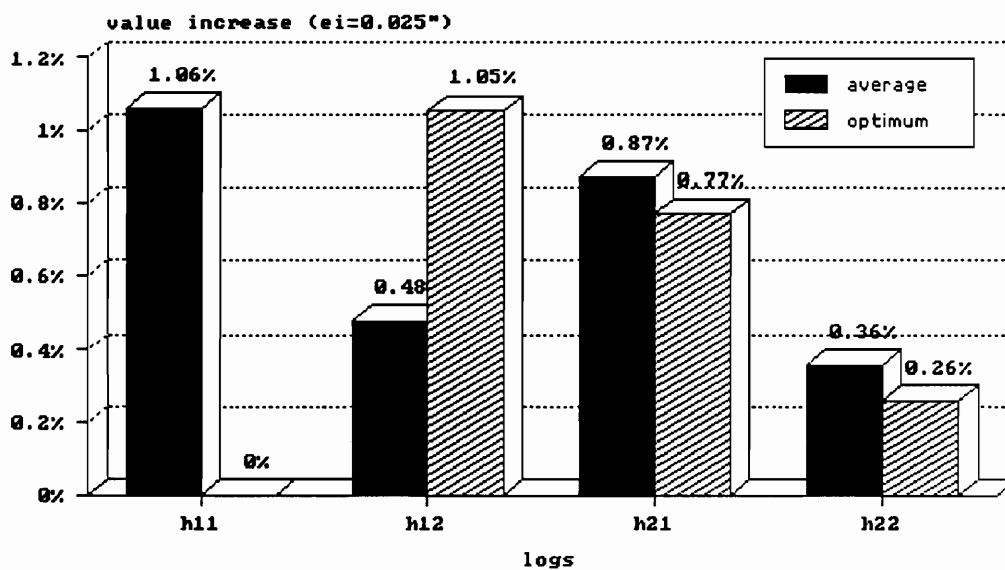


Figure 5.14. Value increases resulting from using a 0.025 inch edging increment compared to a 0.125 inch increment. "Average" signifies the percent increase in the average value for all rotation angles when doing rotation only. "Optimum" signifies the percent increase in the optimum value when doing rotation only.

## CHAPTER 6 CONCLUSIONS AND FURTHER IMPROVEMENTS

The capability of handling logs in any shape in 3-D space and doing positioning operations on the log and every piece cut from the log are appealing features in a log breakdown program. In addition, the three levels of nested dynamic programming algorithms used in SAW3D are more efficient than the exhaustive searching methods used in most existing simulation programs, and they provide optimum sawing, edging and trimming patterns. The capability of providing either a simulation or optimization approach makes it a flexible analytical tool to determine the effects of such factors as scanning data density, sawing methods, log shapes, and log/cant/flitch orientations on lumber recovery both in terms of value and volume. It can be used to evaluate the existing log breakdown process and to explore some practical rules-of-thumb regarding sawing methods and log orientations for different categories of log shapes and sizes.

However, like any other existing optimization program, there is room for improvement. Presently, this program can not consider internal defects or annual ring orientation nor can it be used to do on-line optimization. On a 386 20MHZ IBM compatible computer, logs tested so far have execution times ranging from tens of seconds to several hours, depending upon log size and shape as well as

positioning operation options. There are several avenues for increasing the speed of the program, including:

1) Using more efficient algorithms. For example, a revised dynamic programming algorithm, such as the one by Gilmore and Gomory (1966), could be used in the trimming level. The Gilmore and Gomory algorithm accelerates the searching process by eliminating some of the arcs from the network of the longest path problem.

2) Using information generated by the current version of SAW3D. For example, now SAW3D locates the best positions of a log and each piece by a simple exhaustive search. If some rules-of-thumb can be found for different categories of log shapes and sizes using the current version of SAW3D, then those rules can be used as guidelines to reduce the search range.

3) Using multiprocessors. The dynamic programming algorithm and the positioning operations lend themselves well to parallel processing.

4) Using better scanning data. If a log profile scanning system is developed that is able to provide a series of direct measurements at each of the opening face widths, then all operations of finding widths of opening faces in SAW3D could be eliminated. This would greatly speed up SAW3D.

As log scanning technology advances and log breakdown programs approach the ideal model, significant value and

volume recovery gains will be realized. So far, SAW3D has provided advanced capability in terms of the combination of 3-dimensional log shape, both simulation and optimization options, positioning operations, and microcomputer compatibility.

### Bibliography

1. Adkins, W.K., D.B. Richards, D.W. Lewis, and E.H. Bulgrin. 1980. Programs for computer simulation of hardwood log sawing. USDA Forest Service Res. Paper FPL-357. Forest Products Laboratory, Madison, WI.
2. Airth, J.M., and W.W. Calvert. 1973. Adapting computer simulation techniques to the sawmill. Canadian Forest Industries 93(9):48-52.
3. Airth, J.M., and W.W. Calvert. 1973. Computer simulation of log sawing. Info. Rept. OP-X-66. Forintek Canada Corp., Vancouver, B.C. Canada.
4. Alleckson, T.D., H.B. Sanders, A.J. Koivo, and T.J. Williams. 1980. Studies on the optimum production of lumber by computer positioning of logs in sawmills. Purdue Lab. for Applied Industrial Control Rept. No. 79. Purdue University, West Lafayette, IN.
5. Anderson, R.B., and H.W. Reynolds. 1981. Simulated sawing of squares: a tool to improve wood utilization. USDA Forest Service Research Paper NE-473. Northeastern Experiment Station, Upper Darby, PA.
6. Aune, J.E. 1982. Application and benefits of simulation models in the sawmilling industry. Pages 9-18 in Proceeding of the Process Control In The Forest Products Industry Symposium. Society of Wood Science and Technology, Madison, WI.
7. Bricka, V. 1990. Growth ring pattern consideration in computer sawing simulation. Unpublished M.S. thesis. Oregon State University, Corvallis, OR.
8. Briggs, D.G., 1980. A dynamic programming approach to optimizing stem conversion. Unpublished Ph.D. dissertation. Univ. of Washington, Seattle, WA.
9. Burger, P., and D. Gillies. 1989. Interactive computer graphics. Addison-Wesley Publishing Company, Inc, Reading, MA.
10. Chang, S.J. 1989. Economic feasibility analysis of the NMR fast imaging scanner. Pages VII-1 to VII-6 in Proceedings of the Third International Conference On Scanning Technology In Sawmilling. Miller-Freeman Publications, San Francisco, CA.
11. Chassen, L.H. 1987. Application of "snapshot"

- scanning and optimization to log bucking. Pages 39-47 in Proceedings of the Log Bucking Technology Symposium. Special Publication No. SP-29. Forintek Canada Corp., Vancouver, B.C., Canada.
12. Cooney, T. 1987. Bucking decisions: computer programs help loggers increase revenues. *Journal of Forestry* 85(1):13-14.
  13. Cummins, L.K., and D.D. Culbertson. 1972. Sawmill simulation model for maximizing log yield values. *Forest Products Journal* 22(10):34-40.
  14. Faaland, B., and D. Briggs. 1984. Log bucking and lumber manufacturing using dynamic programming. *Management Science* 30(2):245-257.
  15. Funt, B.V., and E.C. Bryan. 1987. Detection of internal log defects by automatic interpretation of computer tomography images. *Forest Products Journal* 37(1):56-62.
  16. Geerts, J.M. 1984. Mathematical solution for optimising the sawing pattern of a log given its dimensions and its defect core. *New Zealand Journal of Forestry Science* 14(1):124-34.
  17. Gilmore, P.C., and R.E. Gomory. 1966. The theory and computation of knapsack functions. *Operations Research* 14(6):1045-1074.
  18. Glück, P., and W. Kock. 1973. Die Optimale Rohholzaustormung. *Centralblatt für des gesampfte forstwesen* 90(4): 193-228.
  19. Hallock, H., and D.W. Lewis. 1971. Increasing softwood dimension yield from small logs -- best opening face. USDA Forest Service Res. Paper FPL-166. Forest Products Laboratory, Madison, WI.
  20. Hallock, H., A.R. Stern, and D.W. Lewis. 1976. Is there a "best" sawing method? USDA Forest Service Res. Paper FPL-280. Forest Products Laboratory, Madison, WI.
  21. Leach, H.A. 1973. Computer program for log bucking and sawing, user's manual. Unpublished manual. Carrol-Hatch Ltd., Vancouver, B.C., Canada.
  22. Leach, H.A. 1979. SAWSIM: sawmill simulation program. H.A. Leach & Co., Ltd. Vancouver, B.C., Canada.



23. Leban, J.M., F. Colin, and F. Houllier. 1990. SIMQUA - a wood quality simulation software program. SIMQUA user's manual. French Wood Product Quality Station (INRA). Champenoux, France.
24. Lewis, D.W. 1985. Sawmill simulation and the best opening face system: a user's guide. USDA Forest Service Gen. Tech. Report FPL-48. Forest Products Laboratory, Madison, WI.
25. Lewis, D.W. 1985. Best opening face system for sweepy, eccentric logs: a user's guide. USDA Forest Service Gen. Tech. Report FPL-49. Forest Products Lab., Madison, WI.
26. Lewis, D.W., and H. Hallock. 1974. Best opening face programme. Australian Forest Industries Journal 11:21-30.
27. McAdoo, J.C. 1969. Computer simulation of small-log mill processing. Forest Products Journal 19(4):34-35.
28. Nakata, K. 1986. Simulation of softwood-log sawing. Journal of The Hokkaido Forest Products Research Institute 410(3):15-22.
29. Occena, L.G., and J.M.A. Tanchoco. 1988. Computer graphics simulation of hardwood log sawing. Forest Products Journal 38(10):72-76.
30. Peter, R.K. 1967. Influence of sawing methods on lumber grade yield from yellow-poplar. Forest Products Journal 17(11):19-24.
31. Peter, R., and J.H. Bamping. 1962. Theoretical sawing of pine logs. Forest Products Journal 12(11):549-557.
32. Pnevmticos, S. M., and S. H. Mann. 1972. Dynamic programming in tree bucking. Forest Products Journal 22(2):26-30.
33. Pnevmticos, S.M., P.E. Dress, and F.R. Stocker. 1974. Log and sawing simulation through computer graphics. Forest Products Journal 24(3):53-55.
34. Pnevmticos, S.M., H.G. Lama, and M. R. Milot. 1976. Application of computer graphics in simulating sawmilling operations. Pages 1-13 in Proceedings of 9th Annual Simulation Symposium. IEEE, Tampa, FL.
35. Pnevmticos, S.M., and P. Mouland. 1978. Hardwood

- sawing simulation techniques. Forest Products Journal 28(4):51-55.
36. Priasukmana, S. 1984. Sawing of sweepy logs using a live-sawing simulation model. Unpublished M.S. thesis. Univ. of Washington, Seattle, WA.
  37. Reynolds, H.W., and C.J. Gatchell. 1969. Sawmill simulation: concepts and computer use. USDA Forest Service Res. Note NE-100. Northeastern Forest Experiment Station, Upper Darby, PA.
  38. Reynolds, H.W. 1970. Sawmill simulation: data instructions and computer programs. USDA Forest Service Res. Paper NE-152. Northeastern Forest Exp. Station, Upper Darby, PA.
  39. Richards, D.B. 1973. Hardwood lumber yield by various simulated sawing methods. Forest Products Journal 23(10):50-58.
  40. Richards, D.B. 1977. Value yield from simulated hardwood log sawing. Forest Products Journal 27(12):47-50.
  41. Richards, D.B., W.K. Adkins, H. Hallock, and E.H. Bulgrin. 1979. Simulation of hardwood log sawing. USDA Forest Service Res. Paper FPL-355. Forest Products Laboratory, Madison, WI.
  42. Richards, D.B., W.K. Adkins, H. Hallock, and E.H. Bulgrin. 1980. Lumber values from computerized simulation of hardwood log sawing. USDA Forest Service Res. Paper FPL-356. Forest Products Laboratory, Madison, WI.
  43. Rickford, E.N. 1989. Scanning for true shape. Pages II-1 to II-40 in Third International Conference on Scanning Technology in Sawmilling. Miller Freeman Publications, San Francisco, CA.
  44. Rogler, R. K., and H. O. Canham. 1986. An optimal log bucking program for microcomputers. The Compiler 4(2):27-30.
  45. Rogers, D.F., and J.A. Adams. 1990. Mathematical elements for computer graphics, 2/ed. McGraw-Hill Publishing Company, New York, NY.
  46. Scaramella, G. 1987. Stem optimizing: a critical link in sawmill production control. Pages 49-63 in Proceedings of the Log Bucking Technology Symposium.

- Special Publication No. SP-29. Forintek Canada Corp., Vancouver, B.C., Canada.
47. Sullivan, P. 1987. Bucking optimization with "real shape" scanning. Pages 64-75 in Proceedings of the Log Bucking Technology Symposium. Special Publication No. SP-29. Forintek Canada Corp., Vancouver, B.C., Canada.
  48. Tejavibulya, S. 1981. Dynamic programming sawing models for optimizing lumber recovery. Unpublished Ph.D. Dissertation. University of Washington, Seattle, WA.
  49. Tochigi, T., K. Onose, H. Satoh, and K. Sakatsume. 1983. A computer simulation of grain patterns on sawn surfaces. Mokuzaï Gakkaishi 29(12):845-852.
  50. Todoroki, C.L. 1988. Seesaw: a visual sawing simulator, as developed in version 3.0. New Zealand Journal of Forestry Science 18(1):116-123.
  51. Thomlinson, W.W. 1987. True-shape scanning: Lloyd Controls' optimum log value system. Pages 5-15 in Proceedings of the Log Bucking Technology Symposium. Special Publication No. SP-29. Forintek Canada Corp., Vancouver, B.C., Canada.
  52. Tsolakides, J.A. 1969. A simulation model for log yield study. Forest Products Journal 19(7):21-26.
  53. Wagner, F.G., and F.W. Taylor. 1975. Simulated sawing with a chipping headrig. Forest Products Journal 25(10):24-28.
  54. Wagner, F.G., F.W. Taylor, P. Steele, and P.E.G. Harless. 1989. Benefit of internal log scanning. Pages V-1 to V-17 in Proceedings of the Third International Conference On Scanning Technology In Sawmilling. Miller-Freeman Publications, San Francisco, CA.
  55. Williston, E.M. 1981. Small log sawmills. Miller-Freeman Publications, San Francisco, CA.
  56. Wodzinski, C., and E. Hahn. 1966. A computer program to determine yields of lumber. USDA Forest Service Misc. Pub. Forest Products Lab., Madison, Wis..
  57. Zheng, Y., F.G. Wagner, P.H. Steele, and Z. Ji. 1989. Two-dimensional geometric theory for maximizing lumber yield from logs. Wood and Fiber Science 21(1):91-100.

## **APPENDIX**

**Appendix A. The output from sawing a horn-down shaped log  
20 inches in diameter and 10 feet long.**

log scanning data file: h21.dat  
 sawkerf = 0.125000 in  
 edging kerf = 0.125000 in  
 interval between sawing stages = 0.125000  
 interval between edging stages = 0.125000

Optimum rotation angle = 90.00  
 Optimum skew angle = -2.15  
 pitching angle increment = pitch-range/2.000000

saw placement position :  
 16.884132 ( 6 in )10.884132 don't cut !

saw placement position :  
 10.884132 ( 1 in )9.634132 don't cut !

saw placement position :  
 9.259132 ( 1 in )8.009132  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 11.518715 ( 10 in )1.643715  
 don't cut  
 edging positions:  
 1.643715 ( 4 in )-2.356285  
 trimming positions:  
 -0.300342 (10ft) 119.699661  
 edging positions:  
 -2.356285 ( 8 in )-10.231285  
 don't cut  
 edging value = \$0.81

saw placement position :  
 8.009132 ( 2 in )6.009132  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 12.061402 ( 6 in )6.061402  
 don't cut  
 edging positions:  
 5.936402 ( 12 in )-5.938598  
 trimming positions:  
 -0.225342 (10ft) 119.774658  
 edging positions:  
 -5.938598 ( 6 in )-11.938598  
 don't cut  
 edging value = \$6.70

saw placement position :  
 6.009132 ( 2 in )4.009132  
 optimum pitching angle = -0.000000 (degree)

edging positions:  
 12.369861 ( 4 in )8.369861  
 don't cut  
 edging positions:  
 7.869861 ( 12 in )-4.005139  
 trimming positions:  
 -0.150342 (10ft) 119.849655  
 edging positions:  
 -4.005139 ( 4 in )-8.005139  
 trimming positions:  
 -0.150343 (10ft) 119.849655  
 edging positions:  
 -8.005139 ( 4 in )-12.005139  
 don't cut  
 edging value = \$8.53

saw placement position :  
 4.009132 ( 2 in )2.009132  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 8.732307 ( 12 in )-3.142693  
 trimming positions:  
 -0.075342 (10ft) 119.924660  
 edging positions:  
 -3.142693 ( 6 in )-9.142693  
 trimming positions:  
 -0.075342 (10ft) 119.924660  
 edging value = \$9.30

saw placement position :  
 2.009132 ( 2 in )0.009132  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 9.723623 ( 12 in )-2.151377  
 trimming positions:  
 -0.000342 (10ft) 119.999657  
 edging positions:  
 -2.151377 ( 4 in )-6.151377  
 trimming positions:  
 -0.000342 (10ft) 119.999657  
 edging positions:  
 -6.151377 ( 4 in )-10.151377  
 trimming positions:  
 18.806078 (8ft) 114.806076  
 edging value = \$9.89

saw placement position :  
 0.009132 ( 2 in )-1.990868  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 9.515594 ( 12 in )-2.359406  
 trimming positions:  
 0.074658 (10ft) 120.074661  
 edging positions:  
 -2.359406 ( 4 in )-6.359406  
 trimming positions:  
 0.074658 (10ft) 120.074661  
 edging positions:  
 -6.359406 ( 4 in )-10.359406  
 trimming positions:  
 22.295547 (8ft) 118.295547  
 edging value = \$9.89

saw placement position :  
 -1.990868 ( 2 in )-3.990868  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 8.699958 ( 12 in )-3.175042  
 trimming positions:  
 0.149658 (10ft) 120.149658  
 edging positions:  
 -3.175042 ( 6 in )-9.175042  
 trimming positions:  
 0.149658 (10ft) 120.149658  
 edging value = \$9.30

saw placement position :  
 -3.990868 ( 2 in )-5.990868  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 6.918109 ( 10 in )-2.956891  
 trimming positions:  
 0.224658 (10ft) 120.224655  
 edging positions:  
 -2.956891 ( 4 in )-6.956891  
 trimming positions:  
 0.224658 (10ft) 120.224655  
 edging value = \$6.83

saw placement position :  
 -5.990868 ( 2 in )-7.990868  
 optimum pitching angle = 0.000000 (degree)

edging positions:  
 5.540544 ( 12 in )-6.334456

trimming positions:  
2.518005 (8ft) 98.518005  
edging value = \$5.36

saw placement position :  
-7.990868 ( 1 in )-9.240868  
optimum pitching angle = 0.246111 (degree)

edging positions:  
8.698943 ( 4 in )4.698943  
don't cut  
edging positions:  
3.823943 ( 8 in )-4.051057  
trimming positions:  
0.363937 (8ft) 96.363937  
edging positions:  
-4.051057 ( 4 in )-8.051057  
don't cut  
edging value = \$1.26

saw placement position :  
-9.240868 ( 2 in )-11.240868don't cut !

saw placement position :  
-11.240868( 1 in )-12.490868don't cut !

sawing sequence = 1 2 2 2 2 2 2 2 1  
optimum value of the log = \$67.89  
lumber tally = 218.666673