

AD-A249 096



**SHORT TERM TASK
FINAL REPORT**

DTIC
ELECT
APR 24 1992
S C D

CUT ORDER PLANNING

Co-Principal Investigator: Dr. Jane C. Ammons
Co-Principal Investigator: Dr. Charlotte Jacobs-Blecha
Research Investigator: Terri Smith
Research Assistant: Avril Baker
Research Assistant: Bill Warden

Georgia Institute of Technology
Apparel Manufacturing Technology Center
Computer Science and Information Technology Laboratory
School of Industrial and Systems Engineering

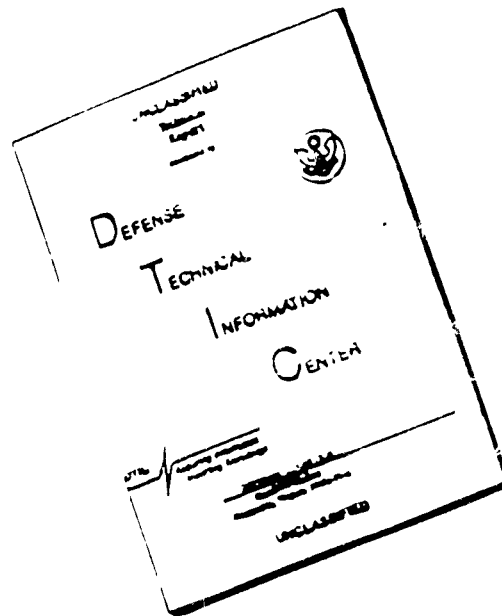
October, 1989 - May, 1991

Research Sponsored by:
U. S. DEFENSE LOGISTICS AGENCY
Contract: DLA900-87-D-0018
Georgia Tech Project No.: A-8496

92-10470



DISCLAIMER NOTICE



THIS DOCUMENT IS BEST
QUALITY AVAILABLE. THE COPY
FURNISHED TO DTIC CONTAINED
A SIGNIFICANT NUMBER OF
PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

May 1991

Final

Oct 1989-May 1991

Cut Order Planning

DLA900-87-D-0018/0012

Short Term Task, Final Rpt.

Jane Ammons and Charlotte Jacobs-Blecha

Georgia Tech Research Institute
215 O'Keefe Building
Atlanta, GA 30332

A-8496

Defense Logistics Agency
Cameron Station (DLA-PRM)
Alexandria, VA 22304-6100

Unclassified/Unlimited

This report details the results of a research project conducted at the Georgia Institute of Technology which investigated methods for improving cut order planning in apparel manufacturing. The project had two complementary objectives. The first objective was to investigate existing solution methodologies for the cut order planning problem. Alternate commercial software packages were examined and their performances comparatively analyzed, using testbed data representative of industrial problems.

The results of this research provide important insights into the state-of-the-art in COP solution methods. A mathematical model of the COP problem was developed to facilitate problem specification and to initiate heuristic development. As a result of the complexity analysis, the COP problem was shown to be sufficiently complex that heuristic methods are the only reasonable means of finding solutions in real time. New methods were developed which perform as well as or better than those used in existing commercial packages. These algorithms have been implemented in a prototype software package for easy incorporation into existing commercial software, and will be transferred to industry through a participating s/w vendor.

Production Planning, Cut Order Planning Software, Apparel
Manufacturing.

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave Blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ..., To be published in When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement.

Denote public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR)

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."

DOE - See authorities

NASA - See Handbook NHB 2200.2.

NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank

DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports

NASA - NASA - Leave blank

NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

ABSTRACT

This report details the results of a research project conducted at the Georgia Institute of Technology which investigated methods for improving cut order planning in apparel manufacturing. The project had two complementary objectives. The first objective was to investigate existing solution methodologies for the cut order planning problem. Alternate commercial software packages were examined and their performances comparatively analyzed, using testbed data representative of industrial problems. The second objective was the theoretical analysis of the cut order planning process and the development of appropriate solution algorithms.

The results of this research provide important insights into the state-of-the-art in COP solution methods. A mathematical model of the COP problem was developed to facilitate problem specification and to initiate heuristic development. As a result of the complexity analysis, the COP problem was shown to be sufficiently complex that heuristic methods are the only reasonable means of finding solutions in real time. New methods were developed which perform as well as or better than those used in existing commercial packages. These algorithms have been implemented in a prototype software package for easy incorporation into existing commercial software, and will be transferred to industry through a participating software vendor.

ACKNOWLEDGEMENTS

We would like to thank the anonymous industrial partners who provided us with so much help and insight on this project. We appreciate those companies who hosted us for site visits and provided many excellent opportunities for interaction. We would like to especially recognize Cynthia Holeridge from Russell Corporation for her assistance.

We are also grateful to several commercial cut order planning software vendors for their interaction and efforts to make this project more successful. Special thanks are due to the two vendors who volunteered their software for testing during the first phase of this project, and for the company who plans to incorporate these results.

Thanks are due to many people who assisted us at Georgia Tech. Leigh McElvany provided excellent support in processing the monthly reports for this project, and Carmella Bell provided outstanding secretarial support. Thanks are also due to Ms. Carol Ring and Dale Stewart, technicians at the AMTC demonstration center, for helping us understand the basics of our problem and for assisting the students; and Mr. Don Eisenstein, a graduate research assistant in the School of Industrial and Systems Engineering, for his help with the complexity proof.

TABLE OF CONTENTS

1.0 Introduction	1
1.1 Overview	1
1.2 Scope of the Project.....	1
1.3 Problem Definition.....	2
2.0 Project Activities	5
2.1 Investigation of Existing Software Packages	5
2.2 Theoretical Analysis of Cut Order Planning.....	5
2.3 Evaluation and Experimentation	6
3.0 Existing Commercial Packages.....	6
3.1 Introduction	6
3.2 Design of Experiment.....	7
3.2.1 Approach	7
3.2.2 Design	8
3.3 Package A Results.....	10
3.3.1 Conclusions for Package A	12
3.3.2 Subjective Observations on Package A	12
3.4 Package B Results.....	13
3.4.1 Conclusions for Package B	14
3.4.2 Subjective Observations on Package B.....	14
3.5 Conclusions on Existing Commercial Software.....	15
4.0 Theoretical Analysis of Cut Order Planning	15
4.1 Mathematical Model	15
4.2 Heuristics for COP.....	22
4.3 Testbed Data	32
4.4 Experimental Results.....	34
4.5 Conclusions for Theoretical Analysis.....	39
5.0 Summary and Conclusions.....	40
5.1. Review of the project	40
5.2. Major contributions of the project	40
5.3 Recommendations for further COP research	41
6.0 References.....	42
Appendix A: Package A Solutions	A1
Appendix B: Package B Solutions.....	B1
Appendix C: Testbed Marker Lengths.....	C1
Appendix D: Prototype Software Computer Codes.....	D1
Appendix E: Computational Results from COP Algorithms	E1

1.0 Introduction

1.1 Overview

To reestablish a competitive position in the international marketplace, the apparel industry is focusing on upgrading its responsiveness to customer needs. Smaller orders are placed in a more dynamic fashion, forcing the efficient production of smaller lots.

Responsive and economical production of apparel products depends upon the interaction of many components, one critical component being an efficient workflow control system. The cut order planning (COP) process is a dynamic one in that the procedure must respond to the ever changing status of many critical factors such as sales, inventory levels, raw materials, and labor and equipment availability. The variety of sizes, styles, fabrics, and colors induces significant complexity in this problem. Adding to the complexity, and thus potentially increasing total production costs, are such considerations as setup, or changeover costs, the question of appropriate lot sizes, the necessity to meet customer demand, and the importance of making competitive delivery promises.

This project has undertaken the study of improving systems for cut order planning to improve the productivity and competitiveness of apparel manufacturers.

1.2 Scope of the Project

The objective of this project has been to investigate appropriate methodologies for cut order planning. First, existing software packages have been examined and their performances comparatively analyzed, utilizing testbed data. Second, a theoretical analysis of the cut order planning process has been performed and appropriate algorithms developed. The approach derived from the theoretical analysis has been implemented in a prototype software package developed for the purposes of experimentation and evaluation of the algorithm.

Two products have resulted from this research. First is an understanding of the relative performance of currently available software for cut order planning and the relative priorities of the cost drivers for the planning decisions. Second is a set of new algorithms implemented in a prototype software package which have been structured for future integration into commercially available software systems.

1.3 Problem Definition

The Cut Order Planning problem can be succinctly described in terms of input, output, and objective:

Given an order to be cut, the *input* to the problem consists of:

- The sizes required for the order,
- The quantity of each size to be cut,
- The total perimeter inches of each pattern piece required for the cut,
- The total area of the pattern pieces required for the cut,
- The front/back sequence (1 or 2 ply per cut), and
- The standards for spreading (marker fixed costs, marker variable costs, cost to copy, minimum and maximum plies, number of sizes per marker, cutting costs, cutting speed, and cutting setup).

The *output* from the cut order planning process then consists of the following:

- The sizes to be combined in each section of the marker,
- The estimated efficiency of the marker (in percent of fabric utilization),
- The cutting cost per unit,
- The total perimeter to be cut, and
- The total area to be cut.

The *objective* of the cut order planning problem is to minimize costs, with a tradeoff to be made between cutting costs and fabric costs. The key decisions to be made are the number of sections required to fill the order, ply height in each section, and the sizes to be cut in each section. The output of which sizes are to be combined in each section of the marker is passed on to the marker making function for actual determination of the marker itself. In Section 3.1, a mathematical model of the cut order planning problem is developed based on this verbal outline of the problem.

The above description of the problem seems to be the traditional one taken by most of the software vendors currently marketing systems for solving the cut order planning problem. Figure 1.1 illustrates the steps of traditional COP. However, it is the view of the researchers that cut order planning should be integrated into a shop floor control setting where a more "systems" view of the world is taken.

Because cut-order planning is the initial step in the release activity of shop floor control, it has significant impact on the performance of downstream production functions. Currently COP is performed independently of subsequent assembly operations. In order to make the entire production system more efficient and more responsive to customer needs, a more comprehensive approach is to integrate COP into the strategic and tactical decision making associated with the overall production system.

As illustrated in Figure 1.2, the COP process should be extended in the future to explicitly consider *tradeoffs* between downstream efficiencies, cutting and fabric costs, and the impact of needed smaller lot sizes. These interactions should capture in both COP an activity itself as well as a better designed production planning and control system which optimizes the work release impact from the COP.

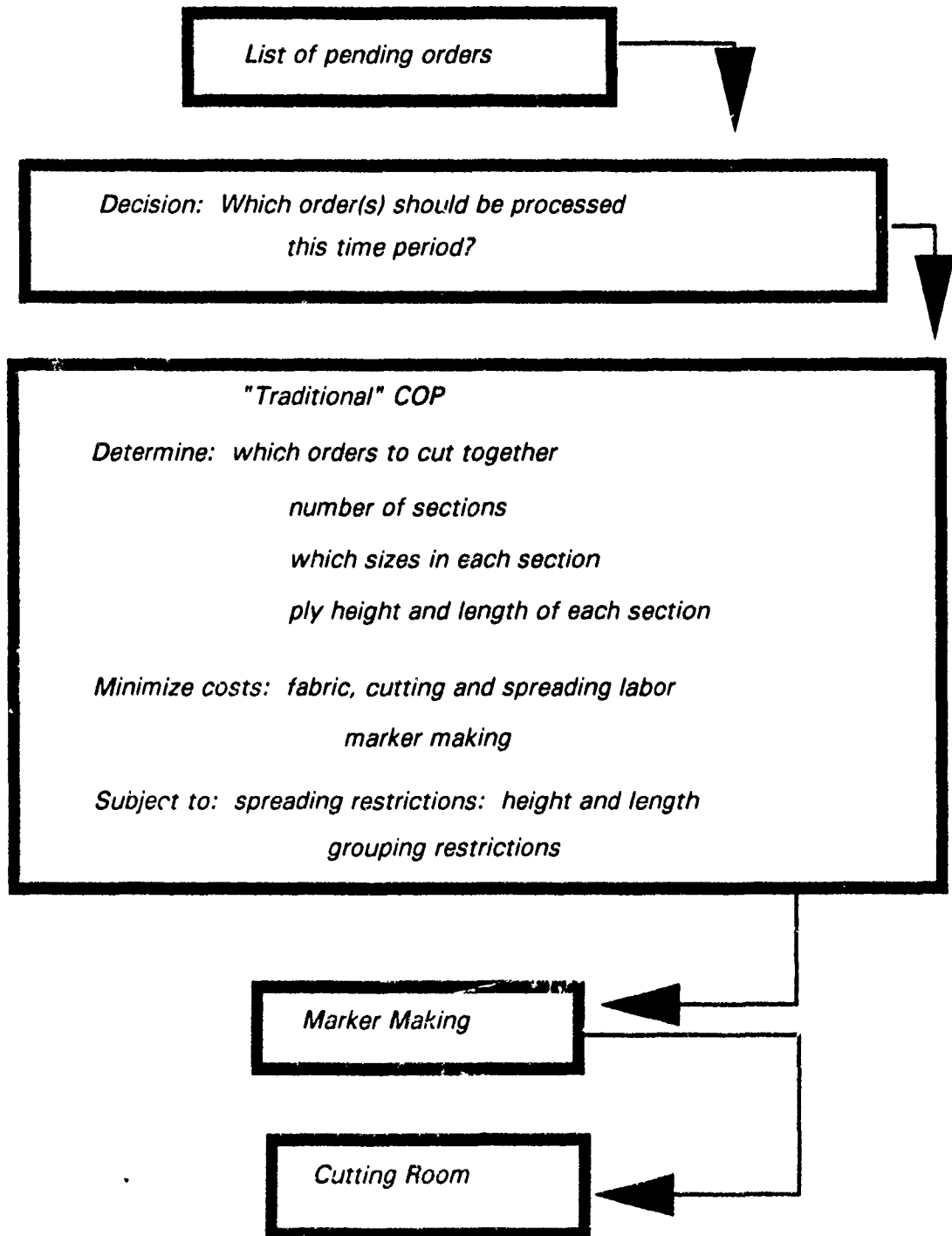


Figure 1.1 Traditional Cut Order Planning

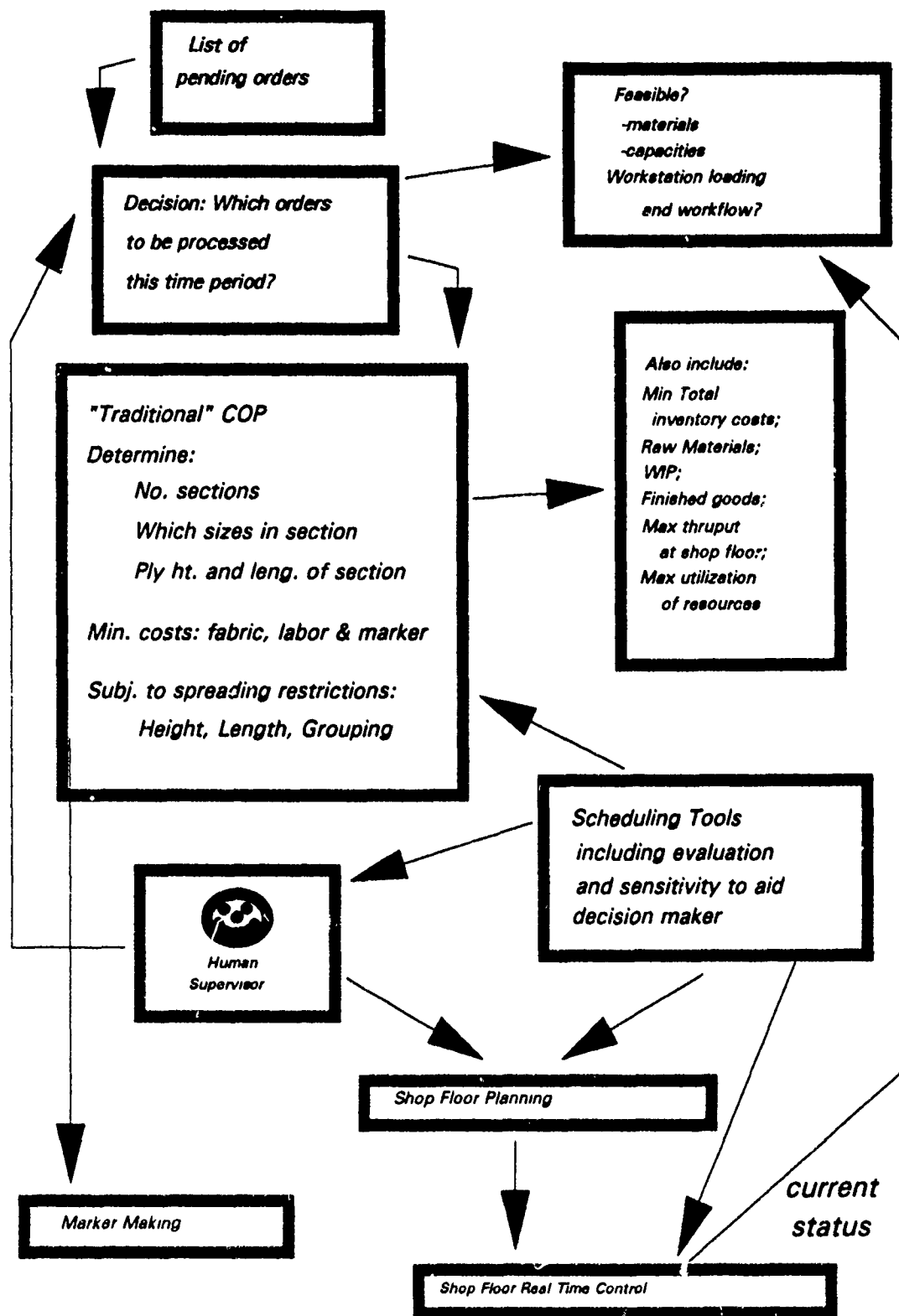


Figure 1.2 The Role of Traditional COP Within a More Comprehensive Planning Approach

2.0 Project Activities

This project has been conducted in two phases. In the first nine months of the project, efforts were concentrated on an investigation of existing software packages. In the second nine months, mathematical models and corresponding algorithms were developed and experimentation was conducted. This project began in September, 1989 and was finished in May, 1991.

2.1 Investigation of Existing Software Packages

Initially thirteen vendors of COP software systems were contacted. A positive response was received from eight of these companies expressing interest in project participation. Vendor software was obtained from two of these 8 companies to solve the problem described in Section 1.3. Each of the obtained packages executed in a similar computing environment (e.g., a DOS-based PC platform). In later sections of the report, these packages will be referred to as Package A and Package B.

Based on representative data supplied by an apparel manufacturer, a tested data set was developed. This data set is described in more detail in Section 3.2 of this report, and listed in Appendices A and B. The relative performance of the software packages were investigated experimentally by running each one against the testbed data. Measures of performance were efficiency and effectiveness. The efficiency was evaluated in terms of computational time and ease of use. The effectiveness was measured in terms of the objective function, a total weighted value of fabric cost and cutting costs.

2.2 Theoretical Analysis of Cut Order Planning

In order to effectively analyze the COP problem, it is necessary to model the problem mathematically. Other problems which seem to have similar structure to the COP problem include the cutting stock problem (e.g., Elsayed and Shetty, 1988; Farley, 1988; Gilmore and Gomory, 1961; and Hinxman, 1980), bin packing problems (e.g., Martello and Toth, 1990; Eilon and Christofides, 1971; and Johnson et al., 1974), the knapsack problem (e.g., Martello and Toth, 1990), and location-allocation problems (e.g., Tompkins and White, 1984; and Francis and White, 1974). In addition to the mathematical basis for the problem, the modeling process relied heavily on discussions with vendors, industrial contacts, and other AMTC personnel who are knowledgeable of the COP problem.

As discussed in more detail in Section 4, the COP problem is a combinatorial one, indicating that heuristic solution methods are appropriate. Based on the problem definition given in Section 1.3, a mathematical model was constructed. This model was evaluated for its accuracy and completeness, and as an indicator of possible solution

methods. As this model was studied and refined, an ongoing review of relevant literature was conducted as the basis for an effective solution algorithm.

Solution methodologies for the developed model were developed and then implemented into prototype software for solving the COP problem. Section 4.2 describes the implementation. This prototype was used for theoretical evaluation of the methodology and for empirical studies utilizing the testbed data, as described in Section 4.3.

2.3 Evaluation and Experimentation

Once the prototype system was constructed, a process of evaluation and refinement was conducted. This process finetuned the effectiveness of the imbedded methodologies and the efficiency of the implementation. Theoretical evaluation investigated several alternative approaches, as indicated in those sections. Empirical evaluation was conducted by running the prototype against the testbed data, as was done for the existing packages. A comparative evaluation of the prototype was executed based on these experiments and the results are described in Section 4.4.

Following discussion of the experimentation, conclusions and recommendations are made in Section 5.0. The participating vendors are being encouraged to incorporate the prototype software into a commercial package so that the results of this research can be effectively assimilated into the apparel industry.

3.0 Existing Commercial Packages

3.1 Introduction

Listed in Section 6.0 are the references which were examined when the existing literature was searched for potential software packages to solve the Cut Order Planning problem. These packages ranged from simple bookkeeping packages to large systems that incorporate planning and execution aspects (e.g., Anon, 1987, Anon, 1988-B, and Kurt Salmon Assoc., 1987). Some packages reviewed were found to be inadequate by the users.

In order to insure a fair comparison in the experimental portion of the research, only packages that run under MS-DOS on IBM-PC compatible computers were considered. There are two packages for solving Cut Order Planning considered in this report. In order to protect the proprietary nature of the commercial software systems, these will be referred to as Package A and Package B. The performance of these packages were compared using an approach based on statistical analysis.

3.2 Design of Experiment

3.2.1 Approach

To investigate the performance of existing commercial packages for cut order planning, a series of "trials" or "experiments" was deemed appropriate on a testbed data set which was representative of actual industry problems. However, since there are literally an infinite number of runs or trials which could be performed, the series of experiments was carefully designed in order to facilitate the maximum possible understanding and insight from the experimental effort expended. Therefore, the powerful tools of statistical analysis were used to design the series of tests that were performed upon the two existing commercial software packages.

Based on the examination of several options, an experimental design scheme was selected to *screen* for the most important factors affecting the cost performance of the packages. By efficiently organizing the variable settings in the set of experiments performed, stronger statements could be made about the objective performance results. Observations only would be taken for subjective performance characteristics such as user friendliness, ease of making changes, and relative run times.

For the screening trials, a frequently used Plackett-Burman (1946) design for 12 runs was employed. Plackett-Burman designs are two-level fractional factorial designs for studying $k = N - 1$ variables in N runs, where N is a multiple of 4. With a mathematical structure which gives them the ability to screen for almost as many main effects as the number of runs performed, these designs are in the family of Resolution III fractional factorial designs (Montgomery, 1984).

For the non-statistician, the terms used in the paragraph above can be readily understood on examination. These terms translate as follows. The term *two-level* means that every factor in the experiment can take on only one of two values. For example, in the experiment described below, the cost of the fabric is a factor. In order to screen to see if fabric cost significantly affects the performance of the package, a "low" fabric cost and a "high" fabric cost were used as alternate settings for this factor. Since the runs were used to screen for the most important, or significant variables, the value settings for each variable were spread greatly apart between "low" and "high" in order to facilitate the detection of differences.

Another important term is *fractional factorial design*. This term denotes that the experiment recognizes the need to run only a portion, or fraction, of the potentially large permutation (or factorial) of factor level settings. In the design employed in this project, a full factorial design to investigate 8 factors, where each factor takes two different values, would require 2^8 or 256 runs. Instead, only a

portion, or fraction, of the complete factorial design was used which only required 12 runs. The 12 runs gave the ability to detect the significant contribution of main factors, at a price of losing the ability to discriminate higher order interactions. The discrimination ability is denoted by the resolution, in this case as indicated by the term *Resolution III*.

3.2.2 Design

The design scheme for the experimental evaluation of each of the two packages of existing commercial software for cut order planning was a twelve run eight factor Plackett-Burman design. This means that twelve runs or trials were performed on Package A and on Package B in order to detect any significant contribution by eight factors of interest. Because a total of eleven factors can be studied in a 12-run Plackett-Burman, and only eight were to be examined, the remaining three possible factors were treated as dummy variables in order to improve the estimate of experimental variation.

The first experimental step involved the brainstorming of all potential factors which might affect the cut order planning solution. Based on this list, an initial pilot study reduced the feasible and practical experimental factors to eight. The eight factors selected are indicated in Table 3.2.1, along with a list of the "low" and "high" values selected for the screening experiment. These values were selected from a set of actual industry data and are considered representative of a typical problem that the commercial software might encounter.

Table 3.2.1 Factors for the Plackett-Burman Design

Factor Label	Factor Description	Values Used
A	Fabric Cost	\$0.50 or \$10.00
B	Number of Pieces in the Order	48 or 1200
C	Distribution of Sizes In Order	uniform or spike
D	Number Sizes Within Order	1 or 6
E	Cutting Labor Cost	\$10 or \$30
F	Spreading Labor Cost	\$8 or \$25
G	Maximum Ply Height	47 m or 108 m
H	Order Filling Requirements	exact or approx.
I,J,K	Estimating Sample Error	

Most of the "low" and "high" values in Table 3.2.1 are self-explanatory. However, the values for Factor C, distribution of sizes in order, and Factor H, order filling requirements, need further explanation. Factor C describes the way the order is distributed over the sizes. When Factor C is set on "low," the order contains the same number of parts needed for each size in the order. The term

uniform is used to denote this condition because the order is uniformly spread over the sizes. On the other hand, when Factor C is set to "high," the order is spread in a very nonuniform way over the sizes (when there is more than one size in the order). When Factor C is set to "high" (denoted *spike*) and there are six sizes in the order (Factor D is set to "high") , then the order profile used in the experiments is shown in Table 3.2.2.

Table 3.2.2 Order Profile with Factor C=high and Factor D=high

	Number of Pieces in Order (Factor B)	
	48 pieces	1200 pieces
Size 30	6	163
Size 32	9	239
Size 34	25	599
Size 36	2	45
Size 38	5	124
Size 40	1	30

The two settings for Factor H in Table 3.2.1 also need further explanation. Factor H describes the conditions under which an order must be filled; more precisely, it determines whether the solution must fill the exact quantities for each size in the order (the "low" setting, denoted *exact*), or if overages or underages are allowed in filling the order (the "high" setting, denoted *approx*). For this test the *approx* setting for Factor H permitted a solution of up to ten units over or under the order specification.

The combinations of these eight factors which made up the twelve runs for the Plackett-Burman design are depicted in Table 3.2.3. This table lists the settings for each factor level using a notation which is standard for experimental design. In this notation, a factor which is to be set at its "high" value for the run is indicated by its capital letter. A factor which is to be set at its "low" value for the run is either omitted altogether, or is indicated by its lower case letter. To see how this notation works, consider Run 1 from the experiment. This run is denoted by the code ABDEFJ, which means for this run, factor A, fabric cost , will be set at its "high" value (\$10 from Table 3.2.1); factor B, number of pieces in the order, will be set at its "high" value (1200); factor C (which does not appear in the notation), distribution of sizes in order, will be at its "low" value (uniform), etc.

In addition to the Plackett-Burman design runs defined in Table 3.2.3, one more trial was run with all factors set at their "high" settings in order to perform validation of the results.

Table 3.2.3 Factor Combinations Observed

Run No.	Factors Observed
1	ABDEFJ
2	ACDEIK
3	BCDHJK
4	ABCGIJ
5	ABFHIK
6	AEGHJK
7	DFGIJK
8	CEFHIJ
9	BDEGHI
10	ACDFGH
11	BCEFGK
12	abcdefghikj

For each of the twelve trials performed on Package A and on Package B, each of the eight factors were preset at the values given in Table 3.2.3. Then the software package was run, and information was collected about the resulting solution. In this case, seven different performance characteristics were measured for each solution obtained. The seven output parameters were:

1. Total Cost (\$)
2. Number of Patterns
3. Number of Sections
4. Total Ply Count
5. Number Units Over Demand Required
6. Number Units Under Demand Required
7. Fabric Utilization

3.3 Package A Results

Appendix A contains the complete computer output from the experimental runs for Package A. The summary of performance measure results of the twelve Plackett-Burman runs and the additional validation run are shown in Table 3.3.1. As shown in the table, for the single run predictive check based on the high settings consistent results were obtained.

Table 3.3.1 Experimental Results for Package A

RUNID	FCost	Pcs	Dst	Szs	Cut Lab	Spd Lab	Max Ply	Delta	i	j	k	A\$ per unit	# Pat	# Sec	Tot Ply	# Ovr	# Und	util% TotFab
ABDEFJ	10.00	1200	0	6	30.00	25.00	47	0	0	1	0	4.00	1	5	200	0	0	0.8400
ACDEIK	10.00	48	1	6	30.00	8.00	47	0	1	0	1	5.14	2	2	7	0	0	0.8300
BCDHJK	0.50	1200	1	6	10.00	8.00	47	10	0	1	1	0.25	4	4	98	0	2	0.8074
ABCGIJ	10.00	1200	1	1	10.00	8.00	108	0	1	1	0	3.70	1	2	200	0	0	0.8400
ABFHIK	10.00	1200	0	1	10.00	25.00	47	10	1	0	1	3.73	1	5	200	0	0	0.8400
AEGHJK	10.00	48	0	1	30.00	8.00	108	10	0	1	1	4.20	1	1	8	0	0	0.8400
DFGIJK	0.50	48	0	6	10.00	25.00	108	0	1	1	1	0.53	1	1	8	0	0	0.8400
CEFHJ	0.50	48	1	1	30.00	25.00	47	10	1	1	0	0.53	1	1	24	0	0	0.7000
BDEGHI	0.50	1200	0	6	30.00	8.00	108	10	1	0	0	0.24	1	2	200	0	0	0.8400
ACDFGH	10.00	48	1	6	10.00	25.00	108	10	0	0	0	4.55	2	2	7	0	0	0.8300
BCEFGK	0.50	1200	1	1	30.00	25.00	108	0	0	0	1	0.24	1	2	200	0	0	0.8400
abcdefgh	0.50	48	0	1	10.00	8.00	47	0	0	0	0	0.38	1	1	24	0	0	0.7000
ABCDEFGH										act ual		3.82	4	4	185	0	8	0.8097
ABCDEFGH										esti ma ted		4.12	2.3	4	183	0	0.17	0.8627

Table 3.3.2 shows the effects of the eight factors on the seven performance variables in value and as a percentage. These effects can be tested for significance by comparing their relative magnitude with the "background noise" or inherent randomness of the system, which is measure using the results from the three dummy variables. This measure of underlying randomness is denoted the Mean Square Error, or MSE. If the effect of a variable is less than or close to the MSE value, then it is interpreted as having unimportant impact on the resulting performance measure.

For example, examine the effects on the total cost (indicated by the row for A\$ in Table 3.3.2). The fabric cost, number of pieces in the order, and the number of sizes in the order were significant contributors to overall price (more than two MSEs in magnitude). Conclusions can be drawn in a similar fashion for the significant factors which affect the number of patterns required (indicated by the "Pattern" row in Table 3.3.2). Distribution of sizes within an order and number of sizes in an order were significant contributors to number of patterns required. For the third performance measure, number of sections required, fabric cost, number of pieces in the order, number of sizes in the order, cost of spreading labor and maximum ply height were significant contributors to determine the number of sections required. Number of pieces in the order was the only significant factor in determining total ply requirements. Fabric cost and maximum ply height contributed to the determination of fabric utilization.

Table 3.3.2 Factor Effects

Factor	A	B	C	D	E	F	G	H	I	J	K		
	FCost	Pcs	Dist	Sizes	Cut Lab	Spd Lab	Max Ply	Delta				Mean	MSE
AS	3.86	-0.53	0.22	0.32	0.20	-0.06	-0.10	-0.08	0.04	-0.18	0.12	3.82	0.11
%	1.01	-0.14	0.06	0.08	0.05	-0.01	-0.02	-0.02	0.01	-0.05	0.03		
Pattern	-0.17	0.17	0.83	0.83	-0.50	-0.50	-0.50	0.50	-0.50	0.17	0.50	0.83	0.39
%	-0.20	0.20	1.00	1.00	-0.60	-0.60	-0.60	0.60	-0.60	0.20	0.60		
Sections	1.00	2.00	-0.33	0.67	-0.33	0.67	-1.33	0.33	-0.33	0.00	0.33	2.67	0.22
%	0.38	0.75	-0.13	0.25	-0.13	0.25	-0.50	0.13	-0.13	0.00	0.13		
Tot Ply	11.33	170.00	-17.33	-22.67	17.00	17.00	11.67	-17.00	17.00	-16.67	-22.33	148.00	18.67
%	0.08	1.15	-0.12	-0.15	0.11	0.11	0.08	-0.11	0.11	-0.11	-0.15		
Over	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
Under	-0.33	0.33	0.33	0.33	-0.33	-0.33	-0.33	0.33	-0.33	0.33	0.33	0.33	0.33
%	-1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	1.00		
Util	0.05	0.04	-0.01	0.04	0.01	0.01	0.05	-0.01	0.01	0.00	0.04	0.22	0.02
%	0.22	0.20	-0.04	0.17	0.02	0.02	0.23	-0.02	0.02	-0.01	0.18		

3.3.1 Conclusions for Package A

The analysis indicated the performance measures were not sensitive to most of the factors. This may be due to higher order interactions blanketing the effects or several other explanations. It may be partially explained by the loss of information associated with converting the information into a form usable by the package. It may also be partially explained by the methodology utilized in package A to solve the problem. **The strongest conclusion which can be drawn appears to be the extremely large impact of fabric cost on total cost.** Not only does the conclusion make logical sense from the problem environment, but is a fact which can be capitalized upon for solution of the cut order planning problem.

3.3.2 Subjective Observations on Package A

The data displays in Package A were well laid out and the menu system kept the user from doing anything detrimental to the data or the package. The package forces the user to describe the problems in terms useful to the package; sometimes these were not in terms useful to the operator. For example, Dynamic Table was used instead of the effects of multiple patterns in a section. Batch runs were awkward because both the order information and the parameter set changed for each run. The system only allowed for running batch orders with the same default parameters. However, Package A finds solutions to the cut order planning problem quickly (on the order of a few seconds) and a counter was incremented on the screen to let the user know that things were progressing.

3.4 Package B Results

An identical protocol was followed for the testing of Package B. Appendix B contains the complete computer output from the experimental runs for Package B. The results of the twelve Plackett-Burman runs and the additional validation run are shown in Table 3.4.1. For the single run predictive check based on the high settings, consistent results were obtained.

For the results obtained for Package B, Table 3.4.2 shows the effects of the eight input variables on the seven output variables in value and as a percentage. As before, comparing the values to the MSE leads to appropriate conclusions concerning the significance of each factor. For Package B, the total cost of a solution is significantly affected by fabric cost and number of pieces in the order. The number of patterns required is significantly affected by the number of sizes in an order and the maximum ply height. The number of sections required is significantly affected by number of pieces in the order, number of sizes in the order, and maximum ply height. Number of pieces in the order was the only significant factor in determining total ply requirements. Fabric cost and maximum ply height contributed to the determination of fabric utilization.

Table 3.4.1 Experimental Results for Package B

					Cut	Spd	Max					AS per	#	#	Tot	#	#	util%
RUNID	FCost	Pcs	Dst	Szs	Lab	Lab	Ply	Dlta	i	j	k	unit	Pat	Sec	Ply	Ovr	Und	TotFab
ABDEFJ	10.00	1200	0	6	30.00	25.00	47	0	0	1	0	3.41	3	5	135	0	0	0.8500
ACDEIK	10.00	48	1	6	30.00	8.00	47	0	1	0	1	4.50	4	4	12	0	0	0.8300
BCDHJK	0.50	1200	1	6	10.00	8.00	47	10	0	1	1	0.21	3	6	213	33	0	0.8400
ABCGIJ	10.00	1200	1	1	10.00	8.00	108	0	1	1	0	3.23	1	2	200	0	0	0.8500
ABFHJK	10.00	1200	0	1	10.00	25.00	47	10	1	0	1	3.24	1	5	201	6	0	0.8500
AEGHIK	10.00	48	0	1	30.00	8.00	108	10	0	1	1	3.55	1	1	9	6	0	0.8500
DFGIJK	0.50	48	0	6	10.00	25.00	108	0	1	1	1	0.52	1	1	8	0	0	0.8500
CEFHJK	0.50	48	1	1	30.00	25.00	47	10	1	1	0	0.45	1	1	29	10	0	0.7500
BDEGHI	0.50	1200	0	6	30.00	8.00	108	10	1	0	0	0.20	1	2	210	60	0	0.8500
ACDFGH	10.00	48	1	6	10.00	25.00	108	10	0	0	0	3.55	1	1	11	40	0	0.8500
BCEFGK	0.50	1200	1	1	30.00	25.00	108	0	0	0	1	0.20	1	2	200	0	0	0.8500
abcdefgh	0.50	48	0	1	10.00	8.00	47	0	0	0	0	0.37	1	2	48	0	0	0.7400
act ual												3.32	2	2	124	17	0	0.8554
est ima ted												3.38	1.6	3	164	22	0	0.9038

Table 3.4.2 Effects of Input on Output for Package B

Factor	A	B	C	D	E	F	G	H	I	J	K		
					Cut	Spd	Max						
Effects	FCost	Pcs	Dst	Szs	Lab	Lab	Ply	Dlta					MSE
AS	3.26	-0.41	0.14	0.23	0.20	-0.12	-0.16	-0.17	0.14	-0.12	0.17	3.17	0.14
%	1.03	-0.13	0.04	0.07	0.06	-0.04	-0.05	-0.05	0.04	-0.04	0.05		
Pattern	0.50	0.17	0.50	1.17	0.50	-0.50	-1.17	-0.50	-0.17	0.17	0.50	1.17	0.28
%	0.43	0.14	0.43	1.00	0.43	-0.43	-1.00	-0.43	-0.14	0.14	0.43		
Sections	0.67	2.00	0.00	1.00	-0.33	-0.33	-2.33	0.00	-0.33	0.00	1.00	1.33	0.44
%	0.50	1.50	0.00	0.75	-0.25	-0.25	-1.75	0.00	-0.25	0.00	0.75		
Tot Ply	-23.33	173.67	9.00	-16.33	-14.33	-18.00	0.00	11.67	7.33	-14.67	1.67	116.67	7.89
%	-0.20	1.49	0.08	-0.14	-0.12	-0.15	0.00	0.10	0.06	-0.13	0.01		
Over	-8.50	7.17	1.83	18.50	-0.50	-7.17	9.50	25.83	-0.50	-9.50	-10.83	25.83	6.94
%	-0.33	0.28	0.07	0.72	-0.02	-0.28	0.37	1.00	-0.02	-0.37	-0.42		
Under	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
Util	0.03	0.04	0.00	0.03	0.00	0.01	0.04	0.00	0.00	0.00	0.03	0.18	0.01
%	0.18	0.21	-0.03	0.17	0.00	0.05	0.22	0.03	-0.01	0.02	0.16		
wt pat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		

3.4.1 Conclusions for Package B

The conclusions for Package B are very similar to those for Package A. Again, the analysis indicated that the performance measures were not sensitive to most of the factors. As for Package A, there are plausible several explanations for this result. However, the most significant result of Package A is again confirmed by the experiments done for Package B: **fabric cost has an overwhelming impact on the total cost.** This conclusion is exploited later in the heuristic development portion of the project.

3.4.2 Subjective Observations on Package B

Package B was fully menu driven, making it easy to use. All necessary commands were given on-screen. The package provides capabilities for storing and retrieving parameters which reduced setup time for each run. The package displayed a summary of markers after it had completed calculations. The summary was brief and provided no detailed information. To read the complete report it was necessary to exit the package and open a text editor. This became time consuming when several runs were made.

3.5 Conclusions on Existing Commercial Software

The objective of the first phase of this project was to investigate existing solution methodologies for the cut order planning problem. Two alternative commercial software packages were examined and their performance analyzed, utilizing testbed data representative of industrial problems.

Given the analysis performed on individual packages, it seems reasonable to compare their relative performance. However, for the data available and the statistical test procedures employed, this was not possible based on the underlying implicit assumptions used in each of the packages. For example, each package had its own method (with different results) for estimating the marker length of a particular size combination within a section. Since this estimate directly determined the amount of fabric required and thus the cost of the section, the "total cost" performance measure of each package was directly dependent on the marker length estimation procedure. Since actual marker making was beyond the scope of this project, estimates had to be used and thus no relative comparisons on the performance measures can be drawn.

One of the most useful results from these empirical studies on existing commercial software is the following: for all packages studied and corresponding marker estimation techniques, **total cost is driven by fabric cost**. By focusing on fabric cost, a direct method to reduce the total cost resulting from the cut planning step is derived.

4.0 Theoretical Analysis of Cut Order Planning

4.1 Mathematical Model

In this section a mathematical model of the cut order planning problem is presented for a more precise statement of the problem. In addition, this model will be helpful in discerning the mathematical complexity of COP and for further algorithmic development.

The model is presented as follows. First, the parameters of the problem are listed. Then, the decision variables are defined. Finally, the objective function and constraints are formulated. Explanation and discussion of the model are given as needed throughout the section.

Problem Description and Parameters: The problem comprises an order to be cut consisting of sizes $s = 1, 2, \dots, S$. The notation d_s will represent the number of units of size s required to fill the order. The marker for the order will contain

sections $j = 1, 2, \dots, J$. The combination and multiples of sizes used in a section of the marker is denoted by index i . For example:

i	size comb.
1	s,m
2	s
3	m,l
4	3m
5	2s,m
... etc.	

If all size combinations are allowed, $i = 1, 2, \dots, 2^{S-1}$ plus all feasible multiples of any size. This number can be reduced by allowing for only a limited number of sizes to be combined in a section. The letter I represents the upper limit on the index i .

- l_i = length of fabric required to cut size combination i (estimate).
- e_i = number of cutting ($\leq$ perimeter) length units in the pattern for size combination i .
- M_i = increased cost of marker making due to size combination i .
- d_{si} = number of units of size s in size combination i .

Other Parameters:

1. c = fabric cost per length unit (normally yards or meters).
2. P = maximum ply height.
3. L = maximum spread length.
4. T = labor cost for time required to spread one length of the table.
5. U = cost per perimeter length unit for cutting.
6. δ_o = number of units allowed to be produced over or under the total units in the order

Decision Variables: There are two sets of integer decision variables representing the ply height of a section and the assignment of sizes to a section.

1. y_j = ply height of section j . $y_j = 0, 1, 2, \dots, P$.
2. x_{ij} = 1 if size combination i is assigned to section j
= 0 otherwise.

Objective:

The objective of the cut order planning problem is to minimize the total cost of cutting the order, including the cost of fabric and labor. Specifically, these costs

are actual fabric costs, spreading costs, cutting costs and the impact on marker making costs.

1. Fabric Cost: c_i is the fabric cost of 1 layer of size combination i in any section. Thus, the total fabric cost over all sections j and all size combinations i is:

$$\sum_{j=1}^J \sum_{i=1}^I c_i y_j x_{ij}$$

2. Spreading Cost: l_i is the length of fabric required to cut size combination i in any section. Thus, $\frac{l_i}{L}$ is the fraction of the table length needed in spreading the section for size combination i in any section and $T \frac{l_i}{L}$ is the labor cost for spreading a one-ply section containing size combination i . Hence, $T \frac{l_i}{L} y_j$ is the total cost of spreading size combination i in section j , and the objective function term for spreading cost is:

$$\sum_{j=1}^J \sum_{i=1}^I T \frac{l_i}{L} y_j x_{ij}$$

3. Cutting Cost: e_i = the total number of cutting inches in the pattern for size combination i . Thus, $U \cdot e_i$ is the cost of cutting size combination i , and the objective function term is:

$$\sum_{j=1}^J \sum_{i=1}^I U e_i x_{ij}$$

4. Increased Marker Making Cost: The total increased cost of marker making will be expressed in the objective function by the term:

$$\sum_{j=1}^J \sum_{i=1}^I M_i x_{ij}$$

The complete objective function can then be expressed as follows:

$$\text{Minimize } Z = \sum_{j=1}^J \sum_{i=1}^I [c_i l_i y_j x_{ij} + T \frac{l_i}{L} y_j x_{ij} + M_i x_{ij} + U e_i x_{ij}]$$

$$\text{OR Minimize } Z = \sum_{j=1}^J \sum_{i=1}^I [c_i l_i y_j + T \frac{l_i}{L} y_j + M_i + U e_i] x_{ij}$$

Constraints

1. A demand constraint is required so the order will be filled. That is, the total number of units planned for will be equal to the total number of units ordered.

$$\sum_{j=1}^J \sum_{i=1}^I d_{si} y_j x_{ij} = d_s \quad \forall s.$$

Note that if overages and underages are allowed, alternative constraints must be added. Using the parameter δ_s , the above constraint must be written as two constraints, as follows. The inequality

$$(1A) \quad \sum_{j=1}^J \sum_{i=1}^I d_{si} y_j x_{ij} - \delta_s \leq d_s \quad \forall s$$

restricts the total number of units, including underages, to be less than or equal to the demand. Similarly, the inequality

$$(1B) \quad \sum_{j=1}^J \sum_{i=1}^I d_{si} y_j x_{ij} + \delta_s \geq d_s \quad \forall s$$

restricts the total number of units including overages to be greater than or equal to the demand.

2. Table length constraint: The following constraint restricts the total length of the marker to be less than or equal to the length of the cutting table.

$$\sum_{j=1}^J \sum_{i=1}^I l_i x_{ij} \leq L.$$

3. Forcing Constraints for ply height:

- 3A. Enforce the upper bound on ply height:

$$y_j \leq P, \quad \forall j$$

- 3B. Force the ply height to be the same for different size combinations in the same section.

$$\text{If } x_{ij} + x_{kj} > 1, \text{ then } y_j x_{ij} = y_j x_{kj}, \quad \forall i \neq k, j$$

4. Variable restriction constraints:

$$y_j \in \{0, 1, 2, \dots, P\}, \forall j$$

$$x_{ij} \in \{0, 1\}, \forall i, j$$

The model as presented above is very difficult to solve. Both the objective function and the constraints numbered 1 and 3B have nonlinear terms.

The model can be simplified by making a change of variable to linearize these terms as shown below. The constraints in 3B are also unmanageable from a math programming standpoint because the current form is a logical expression rather than a mathematical one. These will be transformed to a set of linear inequalities as explained later in this section.

$$\begin{aligned} \text{Let } z_{ij} &= y_j x_{ij} \\ &= \text{the number of replicates (layers) of size} \\ &\quad \text{combination } i \text{ which will be cut in} \\ &\quad \text{section } j \\ &= \{0 \text{ or the ply height of section } j\}. \end{aligned}$$

The model can now be rewritten as follows:

$$\text{Minimize } Z = \sum_{j=1}^J \sum_{i=1}^I [c_i l_i z_{ij} + T \frac{l_i}{L} z_{ij} + M_i x_{ij} + U e_i x_{ij}]$$

$$\text{OR Minimize } Z = \sum_{j=1}^J \sum_{i=1}^I [c_i l_i + T \frac{l_i}{L}] z_{ij} + [M_i + U e_i] x_{ij}$$

Subject to:

1. Demand constraint

$$\sum_{j=1}^J \sum_{i=1}^I d_{si} z_{ij} + \delta_s = d_s, \forall s.$$

(or in the case where overages and underages are allowed):

$$\sum_{j=1}^J \sum_{i=1}^I d_{si} y_j x_{ij} - \delta_s \leq d_s, \forall s \text{ and } \sum_{j=1}^J \sum_{i=1}^I d_{si} y_j x_{ij} + \delta_s \geq d_s, \forall s$$

2. Table length constraint

$$\sum_{j=1}^J \sum_{i=1}^I l_i x_{ij} \leq L.$$

3. Forcing constraints for ply height:

$$3A. \quad z_{ij} \leq x_{ij} P$$

$$\begin{aligned} 3B. \quad & x_{ij} + x_{kj} - D_{j1}q_{j1} \leq 1, \quad \forall i, k (i \neq k), \text{ and } \forall j \\ & z_{ij} - z_{kj} - D_{j2}q_{j2} < 0, \quad \forall i, k (i \neq k), \text{ and } \forall j \\ & z_{kj} - z_{ij} - D_{j3}q_{j2} < 0, \quad \forall i, k (i \neq k), \text{ and } \forall j \\ & q_{j1} + q_{j2} \leq 1, \quad \forall j \\ & q_{j1}, q_{j2} = 0 \text{ or } 1, \quad \forall j \end{aligned}$$

4. Variable restriction constraints

$$\begin{aligned} z_{ij} &\in \{0, 1, 2, \dots, P\}, \quad \forall i, j \\ x_{ij} &\in \{0, 1\}, \quad \forall i, j \end{aligned}$$

Complexity of the Cut Order Planning Problem

This problem is very difficult to solve to optimality when the parameters are of realistic size. Intuitively this difficulty can be explained by pointing out that the number of solutions grows exponentially as the size of the problem increases. The fact that the problem is modeled by an integer program with no structure for easy solutions is another clue to the difficulty of the problem. In fact, the Cut Order Planning problem is NP-complete. This means it is almost certain that no algorithm can be found for solving COP in polynomial time. The proof of this conjecture follows. Since exact solutions to COP cannot be produced in real time, the development of heuristic solutions is the next reasonable step.

Proof of NP-completeness: To address the complexity of the COP problem an even simpler problem, in which we only consider cuts in one dimension instead of two, is shown to be computationally hard. That is, for our simplified problem, each size and fabric section has a height dimension only.

Cut Order Planning Problem (COP):

Instance: Finite set U of sizes. Each size has a one dimensional height, $s(u)$, for all $u \in U$. Each fabric section is of height B (the maximum ply height). There are k sizes for which to plan a cut.

Question: Is there a partition of sizes into disjoint sets $U_1, U_2, \dots, U_k$ such that the sum of the heights of the orders in each fabric section U_i is B or less?

The COP is identical to the Bin Packing Problem.

Bin Packing Problem:

Instance: Finite set U of items. Each item has a one dimensional height, $s(u)$, for all $u \in U$, a positive integer bin capacity B , and a positive integer k cardinality of U .

Question: Is there a partition of the items into disjoint sets $U_1, U_2, \dots, U_k$ such that the sum of the sizes of the items in each section U_i is B or less?

Theorem: COP is NP-complete in the strong sense.

Proof: The COP and the Bin Packing Problem are identical and thus a reduction is immediate. Thus, the COP has a solution if and only if the Bin Packing problem has a solution. Since the Bin Packing Problem is NP-Complete in the strong sense (e.g., Garey and Johnson, 1979), the COP is also NP-complete in the strong sense.

Transformation of Inequalities 3B:

It is not obvious how the change of variable from y_j, x_{ij} to z_j produces the set of inequalities shown in line 3B of the last version of the model. This section is presented to explain the derivation of those inequalities.

Constraints for each section j are needed as follows: if more than one size combination will be cut in a section (i.e., $x_{ij} = 1$ for more than one i in section j), then the corresponding z_j 's must all be equal (i.e., the ply height will be the same for each of the size combinations in section j).

Consider size combinations i and k . The condition to be satisfied is if both i and k are to be assigned to the same section j , then it must be true that the variable representing the ply height of section j containing size combination i (z_j) be equal to the variable representing the ply height of section j containing size combination k (z_k). Mathematically, this condition can be written as follows.

$$(P): \quad \text{If } x_{ij} + x_{kj} > 1, \text{ then } z_j = z_k.$$

This condition is called a logical constraint and can be incorporated into the model in a linear way as alternative constraints (e.g. Bradley, Hax and Magnanti, p. 374). Let A represent the premise: $x_{ij} + x_{kj} > 1$, and B will represent the conclusion $z_{ij} = z_{kj}$. (P) is not satisfied *only* when A is true and B is *not* true. Thus, (P) is equivalent to satisfying not A OR B. (Refer to the following truth table):

A	$\sim A$	B	$\sim B$	$A \Rightarrow B$
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	1

Now select large constants D_1 , D_2 , and D_3 , and binary variables q_1 and q_2 .

Set $q_1 = 0$ if not A is true, and 1 if not A is false.

Set $q_2 = 0$ if B is true, and 1 if B is false.

Now add the following constraints to the model $\forall i, k (i \neq k)$, and $\forall j$:

1. $x_{ij} + x_{kj} - D_1 q_1 \leq 1$
2. $z_{ij} - z_{kj} - D_2 q_2 < 0$
3. $z_{kj} - z_{ij} - D_3 q_2 < 0$
4. $q_1 + q_2 \leq 1$
5. $q_1, q_2 = 0 \text{ or } 1$

Note that in (4) if $q_1 = 0$ then q_2 can either be 0 or 1. If $q_1 = 0$, then x_{ij} and x_{kj} are not both equal to 1 and it does not matter if z_{ij} and z_{kj} are equal. Thus, $q_1 = 0$ means (1) is satisfied. Consequently, if $q_2 = 1$, then (2) and (3) are guaranteed; if $q_2 = 0$, (2) and (3) may not be satisfied, but that is irrelevant. On the other hand, if $q_1 = 1$, then (2) and (3) *must* be satisfied. But $q_1 = 1$ means $q_2 = 0$, and the validity of (4) is established.

In the actual model, we will need to add a set of constants for each section j : B_{j1} , B_{j2} , B_{j3} ; and a set of binary variables for each section j : q_{j1} and q_{j2} .

4.2 Heuristics for COP

Since the Cut Order Planning problem is NP-Complete, efficient algorithms for realistic data will necessarily be heuristic in nature. The remainder of section 4 is devoted to a discussion of heuristics developed, the testbed data used in the analysis of the heuristics, and the results of the analysis.

There are three heuristics presented in this section. Two of these algorithms, Savings and Cherry Picking, are constructive in nature. The Savings heuristic assigns size combinations to a section based on the fabric savings achieved by combining them into one section as opposed to having them assigned to separate sections. The Cherry Picking algorithm builds sections by combining certain sizes based on the best utilization of fabric. The algorithm picks the first and second most numerous sizes in the order and places those sections first, then repeats until all sizes are assigned to a section. The third heuristic is an improvement heuristic rather than a constructive one. The Improvement algorithm takes a current solution and tries to improve it by exchanging sizes in different sections or by combining existing sections into one section.

4.2.1. "Savings" Heuristic for COP

- INPUT:
- (1) An order to be cut, consisting of the various sizes required and a quantity desired of each of these sizes.
 - (2) The number of units over or under the demand that will be allowed.
 - (3) The parameter k which determines the number of iterations after which the savings list will be updated.
 - (4) The ply height of each of the initial sections.
 - (5) List of l_i 's (these are the fabric lengths required for cutting a size combination i - like small and large together - in a particular section).
 - (6) Maximum ply height allowed.
 - (7) Maximum number of sizes allowed per section.
 - (8) The cutting cost per inch of fabric.
 - (9) The unit cost of the fabric.

ALGORITHM:

- Step 1. Assign each unit in the order to a separate section of the initial ply height.
- Step 2. Compute a *savings* (see below) achieved for combining any pair of sections into a single section. The maximum size of this list can be set to a specific value. It is best to keep it less than or equal to the input K . The savings list is sorted as each value is calculated and placed in the list.
- Step 3. Start at the top of the savings list and *feasibly* (see below) merge sections according to the best savings. The first two sections that are merged are placed in a temporary section. Each merge thereafter is made only with this temporary section until the number of sizes per section is reached.
- Step 4. Once the temporary section is full it is saved and cannot be used again.

Step 5. After k mergers in step 3 the savings list should be updated and resorted by performing steps 2 and 3 for all newly created actions, then performing step 3. (note: k will be an input parameter)

Step 6. Continue until no more savings can be achieved (i.e. the savings list has been scanned and the list is exhausted, with no mergers possible).

OUTPUT: (1) The number of sections, the sizes assigned to each of those sections, and the ply height of each section.
(2) The total estimated fabric length required to cut the order.
(3) The deviation of the number of units to be cut from the actual number of units required in the order.

*Savings Computations:

Step 2 of the algorithm requires a computation of savings achieved for combining two sections into one. Described below are the details of this computation, based on whether or not the two sections to be combined contain the same sizes or not.

Case A:

The two sections contain exactly the same size(s). The merger can be accomplished in one of two ways:

- (i) Increase ply height by spreading one section on top of the other and making no change to the size combination in the section.

To compute the savings achieved in this situation, the cost savings is essentially based only on the cutting cost. That is, a number is needed to reflect the savings of cutting the size combination in this section once instead of twice. (Note the length of fabric required for the section is the same before and after the merger and hence has no effect on the cost savings for the merger).

Let e represent the number of cutting inches in the pattern for the size combination in the two sections being considered. Then e is also the number of cutting inches required for the merged section as well. Recall that U = cutting cost/inch.

Thus, $Ue + Ue$ = cost of cutting the two unmerged sections, and Ue = cost of cutting the merged sections. Hence, Ue = SAVINGS in cost obtained by merging the two sections. (See Figure 4.2.1A).

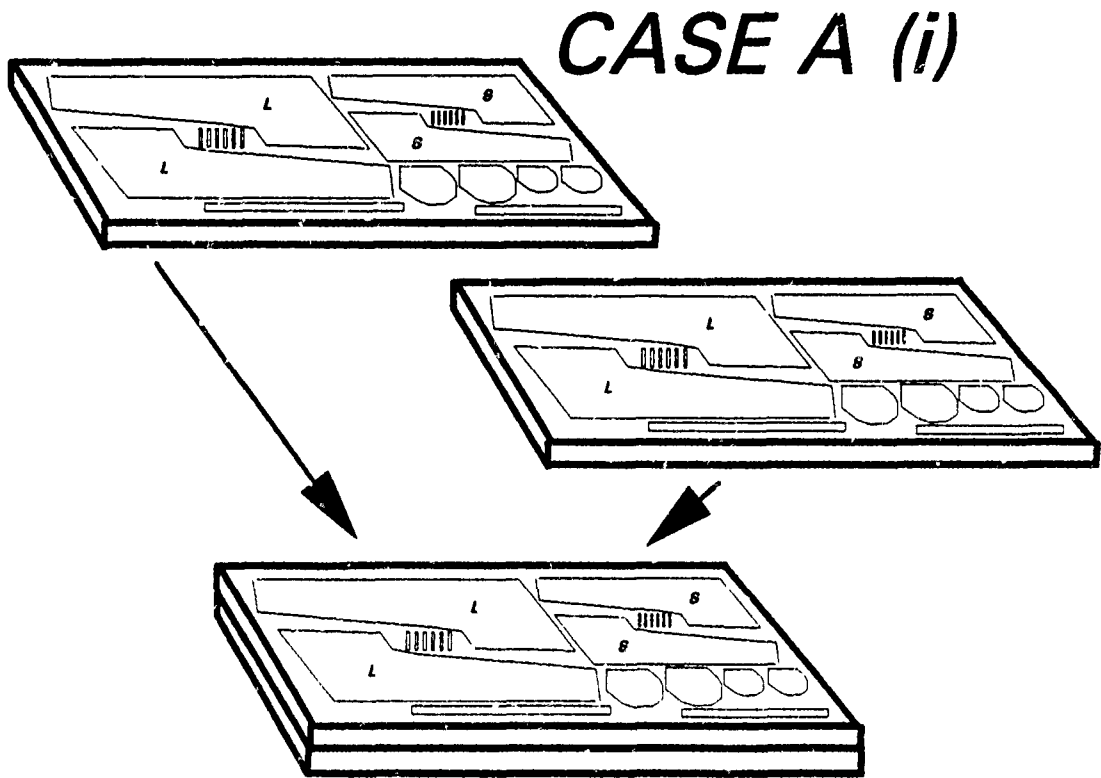


Figure 4.2.1A. Illustration of Case A (i)

However, the merger for case A could also be accomplished by

- (ii) Changing the size combination, leaving the ply height the same.

For example, suppose the two sections both contain sizes 32 and 34. The merged section will then contain the size combination 2-32s and 2-34s. Here the savings will be the decreased cost of fabric required for spreading the merged sections.

Assume the following notation.

l_{11} = length of fabric required to cut one layer of the 1st unmerged section,

l_{12} = length of fabric required to cut one layer of the 2nd unmerged section,

l_{13} = length of fabric required to cut one layer of the 3rd MERGED section, and

p = ply height of the unmerged and merged sections.

Recall that c is the unit cost of fabric

Then, the savings can be computed as $cp(l_{i1} + l_{i2} - l_{i3})$. Case A(ii) is illustrated in Figure 4.2.1B.

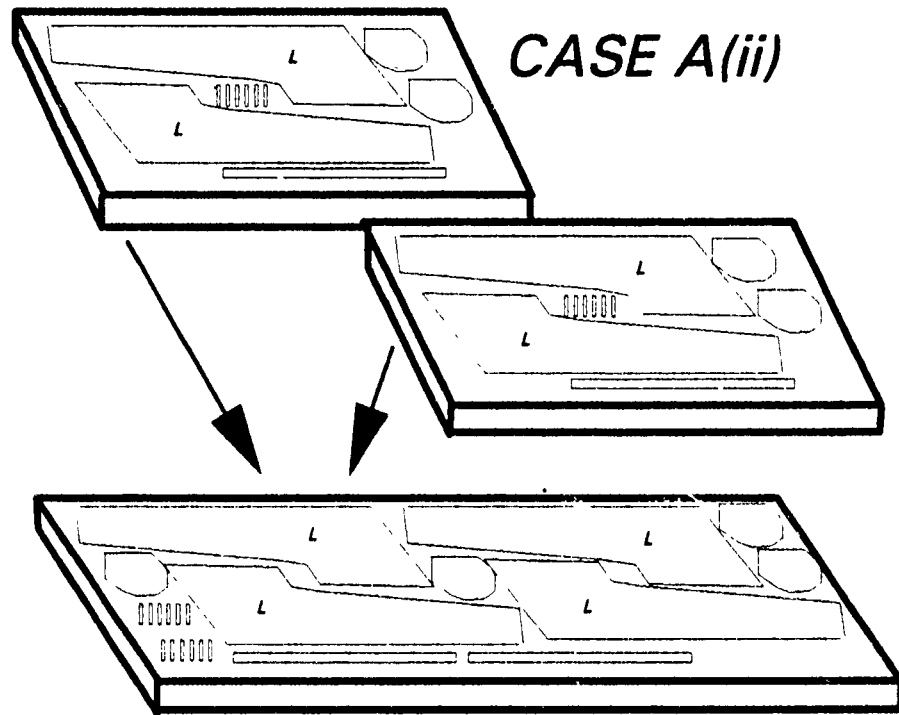


Figure 4.2.1B. Illustration of Case A (ii)

Thus, for case A, the savings is the $\max\{Ue, cp(l_{i1} + l_{i2} - l_{i3})\}$.

If the ply heights of the two section are not equal and the second method of merging the two sections is better the following takes place:

Case B:

The two sections do not contain exactly the same size(s), but are of the same ply height. To maintain consistency, the only possible way to merge two such sections is to merge the size combination, leaving the ply height unchanged. This is precisely the same as case A(ii). Hence the savings computation is $cp(l_{i1} + l_{i2} - l_{i3})$. (See Figure 4.2.1C).

Case C:

The two sections do not contain the same size and have different ply heights. The only way to merge two such sections is to merge the size combination. This is the same as case B. Hence the savings computation is $cp(l_{i1} + l_{i2} - l_{i3})$. (See Figure 4.2.1C).

The ply height of the section being merged is chosen so that the minimum number of overages or underages are created.

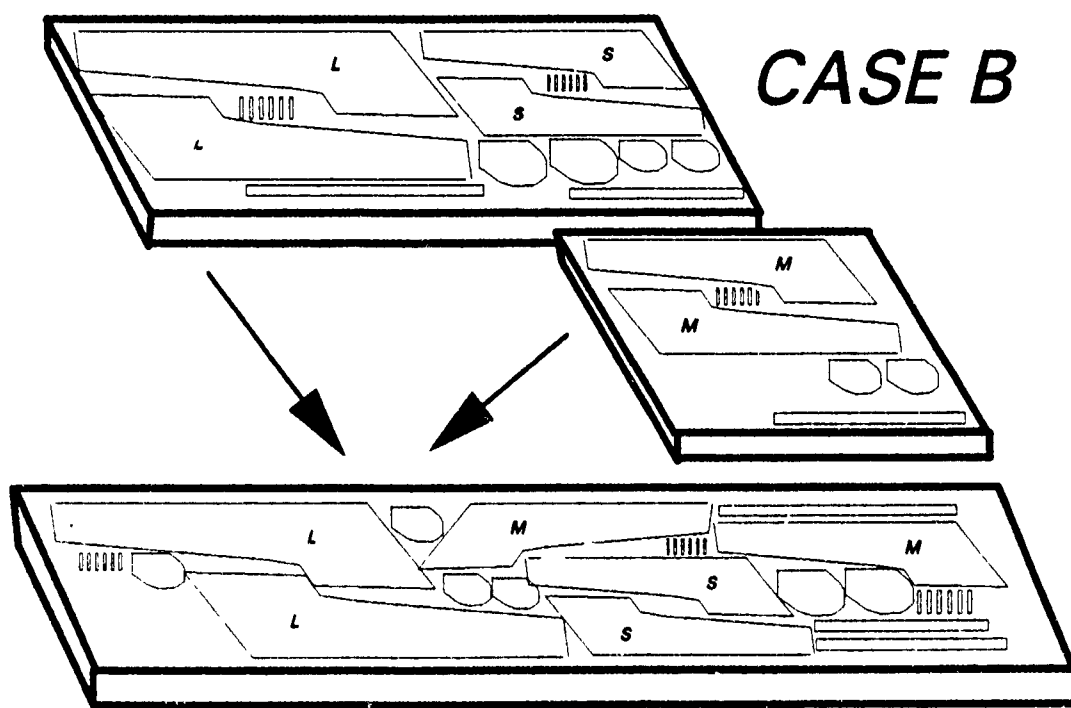


Figure 4.2.1C. Illustration of Cases B and C.

****Feasibility Checks:**

Step 4 of the algorithm states that section mergers should be done only when feasible. The feasibility of such mergers are based on two conditions:

- (1) Will the maximum number of sizes allowed per section be violated? If so, do not merge.
- (2) Will the maximum ply height be violated? If so, do not merge.

4.2.2. Cherry Picking Heuristic

INPUT: (1) An order to be cut, consisting of the various sizes required and a specified demand quantity for each of these sizes.
(2) The number of units over or under the demand that will be allowed.
(3) Maximum ply height allowed.
(4) Maximum number of sizes allowed per section.
(5) List of l_i 's (these are the fabric lengths required for cutting a size combination i - like small and large together - in a particular section).

ALGORITHM:

Step 1. Let q_1 be the largest quantity of any size remaining in the order, and q_2 be the second largest, where $q_2 < q_1$.

(If there is no such q_2 , then one of two cases exists. Case 1: Only one size remains in the order, or Case 2: All sizes remaining have the same order quantity. In either case, set $q_2 = q_1$.)

Form set S by selecting all sizes remaining in the order which have a quantity greater than or equal to q_2 minus the number of units allowed over the specified demand.

Step 2. The next section created will have ply height = $\min\{q_2, \text{max ply height}\}$. Combine the sizes in set S in this section in a way so that a minimal amount of fabric will be required, based on the inputs l_i . For example, if set S contains sizes small and large, it may be necessary to create two sections, one containing size small and the other size large, or only one section may be required which contains both sizes small and large. In the general case, all combinations of the sizes in set S should be considered which do not exceed the maximum number of sizes allowed per section.

Step 3. Reduce the order demand quantities for the sizes in set S by q_2 .

Step 4. If the order contains a size with positive quantity larger than the number of units allowed under the specified demand, go to step 1.

OUTPUT: (1) The number of sections, the sizes assigned to each of those sections, and the ply height of each section.
(2) The total estimated fabric length required to cut the order.
(3) The deviation of the number of units to be cut from the actual number of units required in the order.

4.2.3. Improvement Heuristic

- INPUT: (1) An order to be cut, consisting of the various sizes required and a quantity desired of each of these sizes.
(2) The number of units over or under the demand that will be allowed.
(3) A solution to the problem (see below for details) to be improved upon.
(4) List of l_i 's (these are the fabric lengths required for cutting a size combination i - like small and large together - in a particular section).
(5) Maximum ply height allowed.
(6) Maximum number of sizes allowed per section.
(7) The cutting cost of the fabric per inch.
(8) The unit cost of the fabric.

- OUTPUT: (1) The number of sections, the sizes assigned to each of those sections, and the ply height of each section.
(2) The total estimated fabric length required to cut the order.
(3) The deviation of the number of units to be cut from the actual number of units required in the order.

ALGORITHM:

- Step 0. The iterations of *starting over* must be tracked. If the algorithm starts over and cannot find any improvements after examining all possible exchanges, then the algorithm will terminate.

Each section contains one or more sizes. A portion of a section will consist of only one size. For example, if a section contains sizes M, M and L, the portions to consider are M, L, and MM.

- Step 1. Consider the next portion of one section.
- Step 2. Attempt to reassign the portion from its original section to one or more of the remaining sections so that the reassignment satisfies the *feasibility* checks listed below. If feasible to reassign, compute the *savings* (see below) that would be achieved by making the reassignment.
- Step 3. Attempt to swap the portion from its original section with a portion from one of the remaining sections so that the reassignment satisfies the *feasibility* checks listed below. If feasible compute the *savings* (see below) that would be achieved by making the reassignment.
- Step 4. Perform the reassignment based on the best savings computed.

*Savings Computations:

Case A:

The portion and section contain exactly the same size(s). The merger can be accomplished in one of two ways:

- (i) Increase ply height by spreading one section on top of the other and making no change to the size combination in the section.

To compute the savings achieved in this situation, the cost savings is essentially based only on the cutting cost. That is, a number is needed to reflect the savings of cutting the size combination in this section once instead of twice. (Note the length of fabric required for the section is the same before and after the merger and hence has no effect on the cost savings for the merger).

Let e_i represent the number of cutting inches in the pattern for the size combination in the two sections being considered. Then e_i is also the number of cutting inches required for the merged section as well. Recall that U = cutting cost/inch. Thus, $Ue_i + Ue_i$ = cost of cutting the two unmerged sections, and Ue_i = cost of cutting the merged sections. Hence, Ue_i = SAVINGS in cost obtained by merging the two sections.

However, the merger for case A could also be accomplished by

- (ii) changing the size combination, leaving the ply height the same.

For example, suppose the two sections both contain sizes 32 and 34. The merged section will then contain the size combination 2-32s and 2-34s. Here the savings will be the decreased cost of fabric required for spreading the merged sections.

Assume the following notation:

l_{11} = length of fabric required to cut one layer of the original section from which the portion will be cut (section A),

l_{12} = length of fabric required to cut one layer of the candidate section into which the portion will be added (section B),

l_{13} = length of fabric required to cut one layer of section A after the reassignment of the portion,

l_{i4} = length of fabric required to cut one layer of section B after the reassignment of the portion, and

p = ply height of the unmerged and merged sections.

Recall that c is the unit cost of fabric

Then, the savings can be computed as $cp(l_{i1} + l_{i2} - l_{i3} - l_{i4})$.

Thus, for case A, the savings is the $\max\{Ue_i, cp(l_{i1} + l_{i2} - l_{i3} - l_{i4})\}$.

Case B:

The portion and section do not contain exactly the same size(s).

(i) Same ply height.

To maintain consistency, the only possible way to merge two such sections is to merge the size combination, leaving the ply height unchanged.

(ii) Ply heights not the same.

The merger should take place by combining the size combinations, and choosing the ply height so that the minimum number of overages or underages are created and all other feasibility checks are satisfied.

In either case (i) or (ii), this situation is the same situation as case A(ii). Hence the savings computation is

$$cp(l_{i1} + l_{i2} - l_{i3} - l_{i4}).$$

****Feasibility Checks:**

The feasibility of such mergers are based on two conditions:

- (1) Will the maximum number of sizes allowed per section be violated? If so, do not merge.
- (2) Will the maximum number of units over and under the demand be violated? If so, do not merge.

4.3 Testbed Data

A representative problem was developed to evaluate the performance of the heuristics presented in Section 4.2. The testbed data is composed of a description of an order. The order consists of the sizes required and the quantity of each of those sizes. Section 4.4 details the actual combinations of sizes in an order that were used in the experiment. To plan the cut of the order, it is necessary to estimate marker lengths for all possible combinations of sizes in a marker. Therefore, an important part of the testbed data is the specification of these marker lengths.

For test purposes, estimated lengths for all markers were determined using Package B. All possible combinations of sizes in a marker up to and including six sizes per marker and their associated marker lengths are listed in Appendix C. The majority of the lengths were taken directly from Package B output. However, there were several combinations which could not be obtained directly from Package B. These combinations are marked with an asterisk (*) in Appendix C, and are as follows:

- 4 of each size
- 5 of each size
- 3 of one size, 2 of another
- 4 of one size, 1 of another
- 2 of two sizes, 1 of another
- 2 of two sizes

A brief explanation of how these combinations were estimated is given as follows. The details are presented in Appendix C. It was determined that an order of

1 size 30		
1 size 32	equals	3 size 32 .
1 size 34.		

Using variations of this heuristic, such as

1 size 30		
2 size 32	equals	4 size 32 ,
1 size 34		

lengths were determined for all of the combinations which could not be directly produced. In some cases the order could not be divided in the above fashion, such as:

4 size 30 and 4 size 40 .

Then, the logic followed was to decrease the marker length by a constant amount within each group. An example of this is as follows.

4/0/0/0/0/0					
0/4/0/0/0/0	equals	1/2/1/0/0/0	equals	53.94	
0/0/4/0/0/0	equals	0/1/2/1/0/0	equals	54.95	
0/0/0/4/0/0	equals	0/0/1/2/1/0	equals	55.96	
0/0/0/0/4/0	equals	0/0/0/1/2/1	equals	56.97	
0/0/0/0/0/4					

Since the marker lengths for 4 size 30 and 4 size 40 cannot be divided, they must be estimated using the knowledge that each order is separated by 1.01 inches. This gives the values:

$$\begin{array}{ll}
 4/0/0/0/0/0 & (53.94 - 1.01) = 52.93 \\
 0/0/0/0/0/4 & (56.97 + 1.01) = 57.98
 \end{array}$$

A further detailing of all the estimated markers follows in Appendix C. The next section will describe the results of the implementation of this testbed data with the heuristics and the commercial packages.

4.4 Experimental Results

The testbed data described in Section 4.3 was used to investigate the performance of all the algorithms presented in Section 4.2. For benchmark purposes the commercial packages were run with this data. In this section a summary of the results of this investigation is given.

For the representative problem, several problem instances were examined. Two order types, normal and pathological, were defined. Normal was selected to more typical of industrial problems, and in this case implies that both ply height and order quantities are multiples of twelve. Pathological was selected to test the algorithmic performance on odd numerical combinations, in this case, ply height and/or order quantities which are not multiples of twelve.

Three normal orders and two pathological ones were created. For these five orders, three alternative ply heights were specified for different runs. In addition, the results from the algorithms were recorded before and after the improvement algorithm was applied. These various combinations of the problem parameters resulted in twenty problem instances. Table 4.4.1 summarizes the problem instances.

Table 4.4.1. Experimental Problem Instances

Problem Instance	Normal Order	Pathological Order	Ply Height	Improved?
1	72/144/360/360/144/72		48	No
2	72/144/360/360/144/72		48	Yes
3	72/144/360/360/144/72		108	No
4	72/144/360/360/144/72		108	Yes
5	0/0/0/0/960/240		48	No
6	0/0/0/0/960/240		48	Yes
7	0/0/0/0/960/240		108	No
8	0/0/0/0/960/240		108	Yes
9	0/0/0/0/1200/0		48	No
10	0/0/0/0/1200/0		48	Yes
11	0/0/0/0/1200/0		108	No
12	0/0/0/0/1200/0		108	Yes
13		163/239/599/45/124/30	47	No
14		163/239/599/45/124/30	47	Yes
15		163/239/599/45/124/30	108	No
16		163/239/599/45/124/30	108	Yes
17		200/200/200/200/200/200	47	No
18		200/200/200/200/200/200	47	Yes
19		200/200/200/200/200/200	108	No
20		200/200/200/200/200/200	108	Yes

Based on the conclusions from the experimental results described in Section 3.5, the criterion of **total fabric length** alone was used to compare the performance of the algorithms. The remainder of this section will describe the results of the experimental runs. The graphs given below show these comparisons for all the problem instances in Table 4.4.1. The results of the normal orders with ply height 48 are given in Figures 4.4.1 and 4.4.2.

The results of the normal orders with ply height 48 are given in Figures 4.4.1 and 4.4.2. The solutions marked Improvement are the result of applying the improvement algorithm to an initial solution comprised of each size in the marker in a separate section of the marker. Note that this solution is essentially equivalent to the best of all the others. These solutions are shown for all the problem instances.

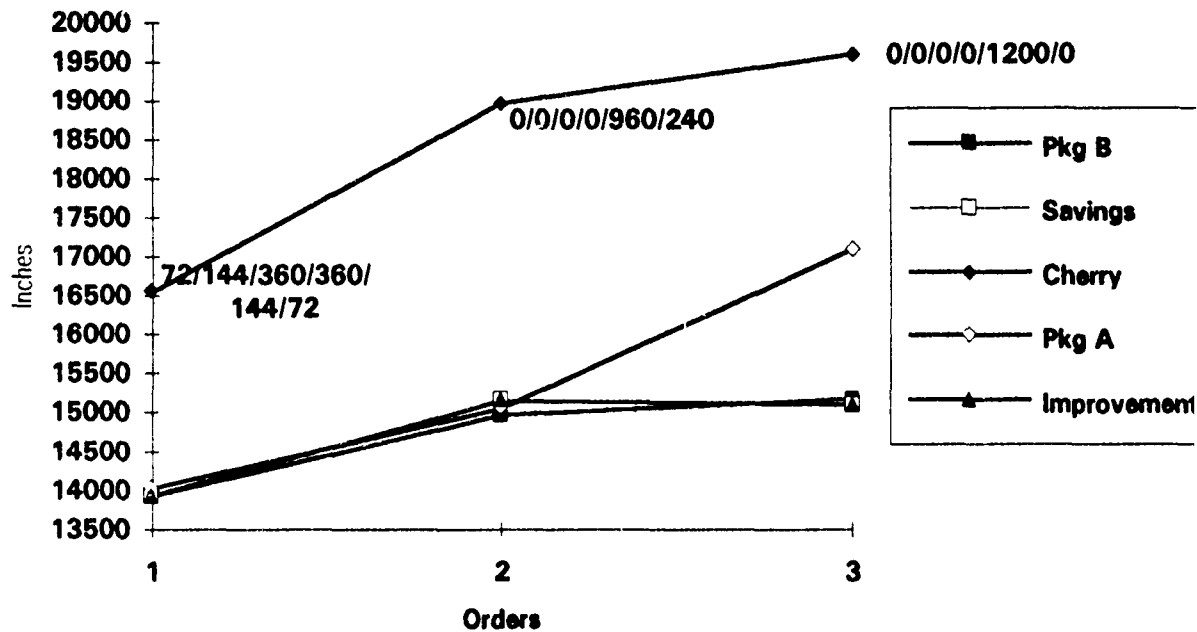


Figure 4.4.1 Total Fabric Inches for Normal Orders with Ply Height 48

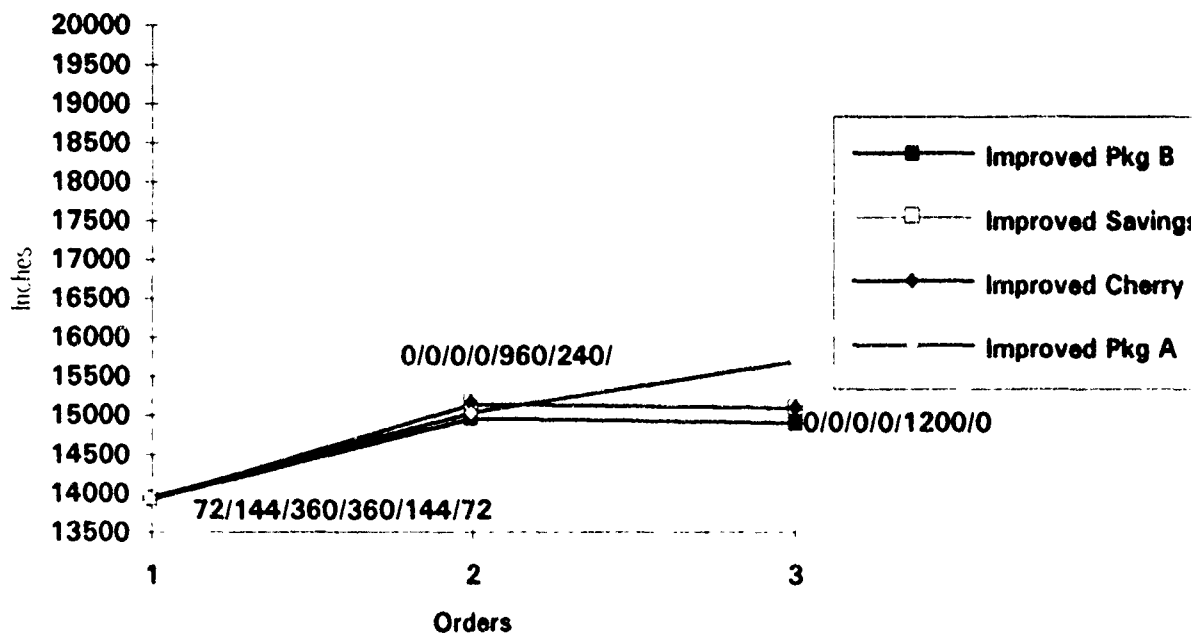


Figure 4.4.2 Total Fabric Inches for Improved Normal Orders with Ply Height 48

From these figures three conclusions can be drawn. First, the Savings algorithm performs significantly better than the Cherry Picking algorithm. The second conclusion is that the Savings algorithm provides solutions which are as good as or better than the commercial packages. Finally, the Improvement algorithm is able to make improvements in all solutions, even the commercial ones. After the improvement algorithm is applied, the solutions are all very similar in fabric length required.

The results for normal orders with ply height 108 are presented in Figures 4.4.3 and 4.4.4. Conclusions are consistent with those for 48-ply problems.

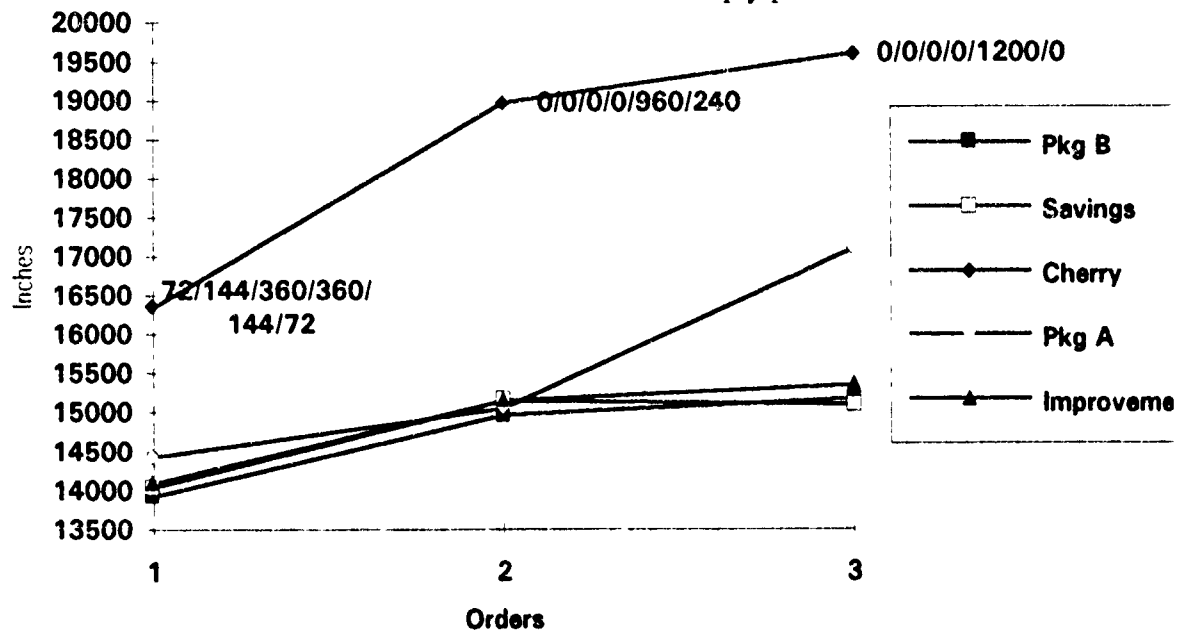


Figure 4.4.3. Total Fabric Inches for Normal Orders with Ply Height 108

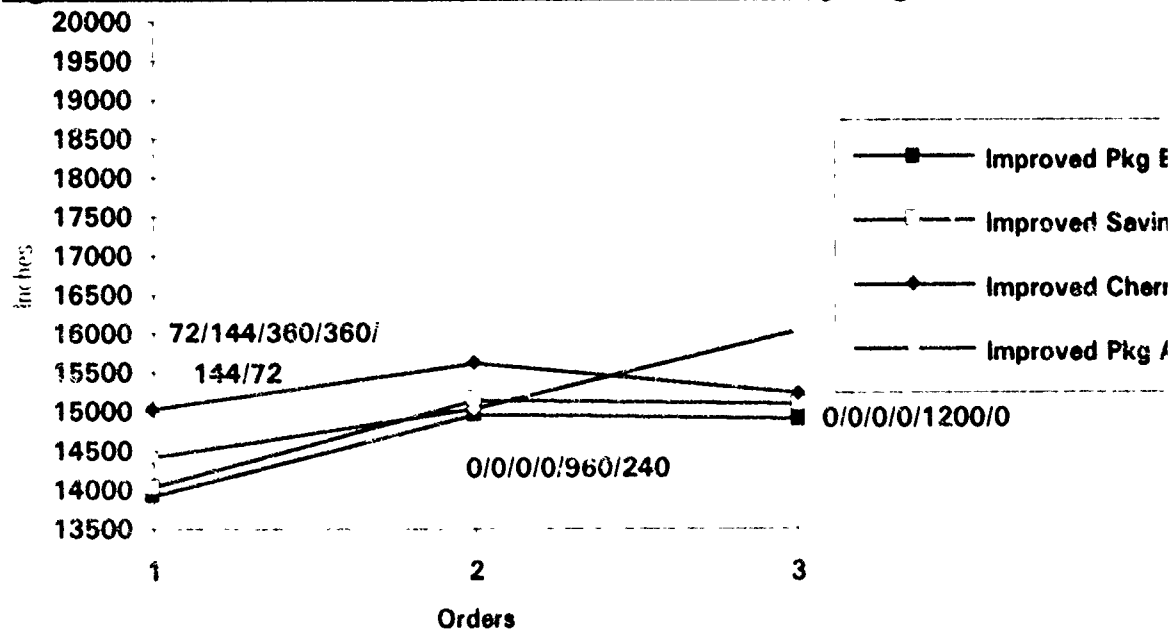


Figure 4.4.4 Total Fabric Inches for Improved Normal Orders, Ply Height 108

Figures 4.4.5 through 4.4.8 present the results of the pathological orders with ply heights 47 and 108. Similar conclusions can be drawn from these graphs as for the normal orders.

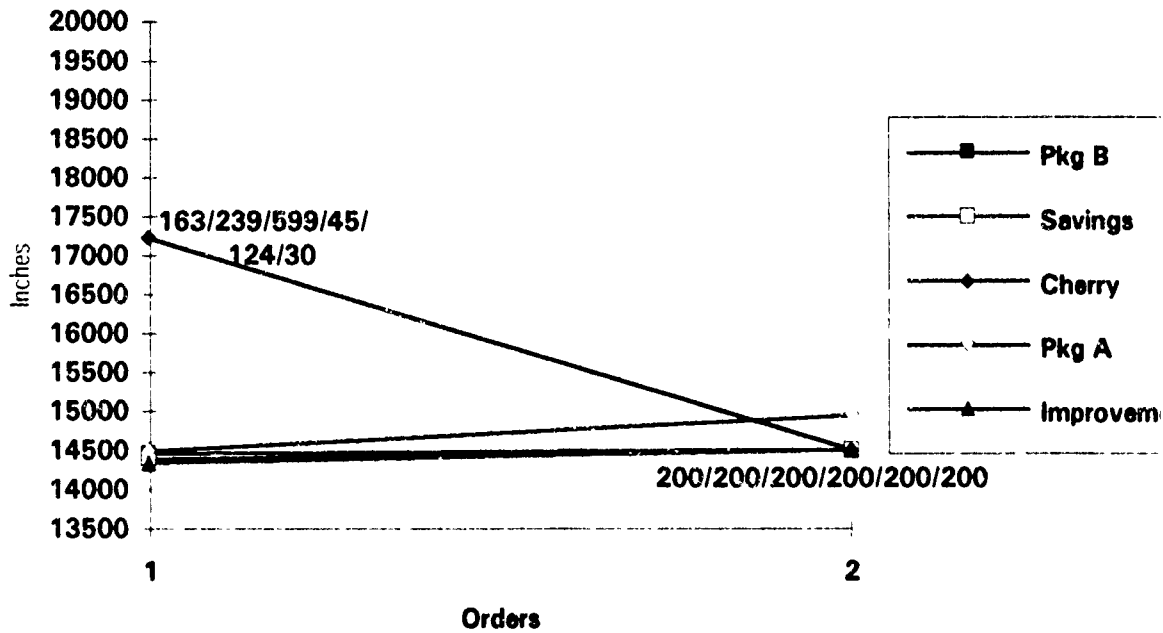


Figure 4.4.5 Total Fabric Inches for Pathological Orders with Ply Height 47

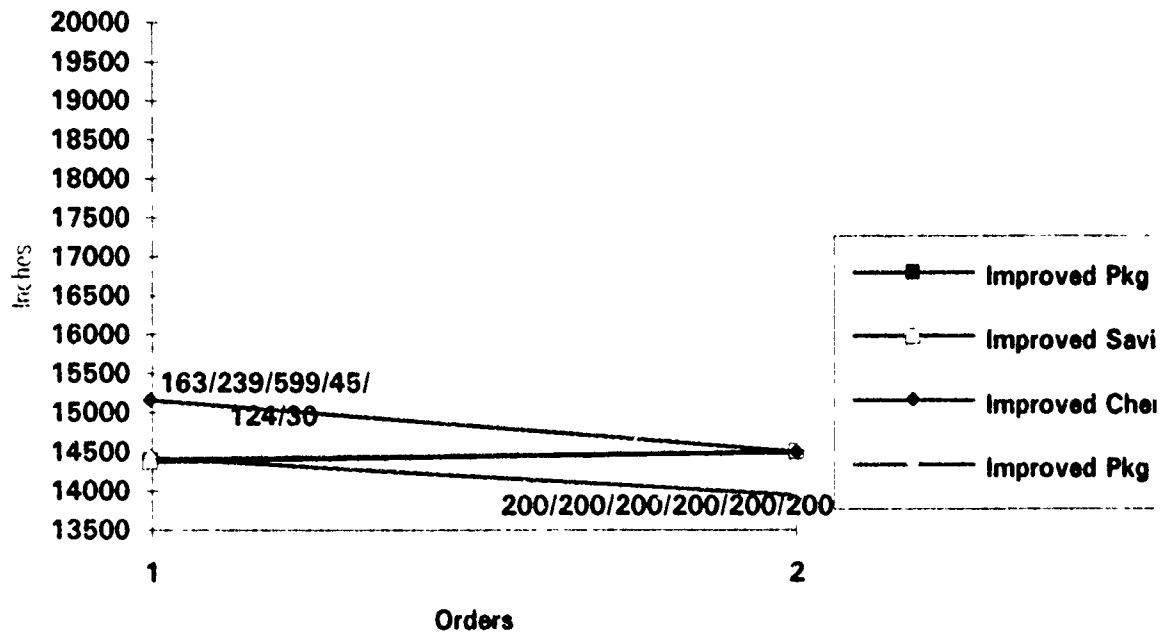


Figure 4.4.6 Total Fabric Inches for Improved Pathological Orders with Ply Height 47

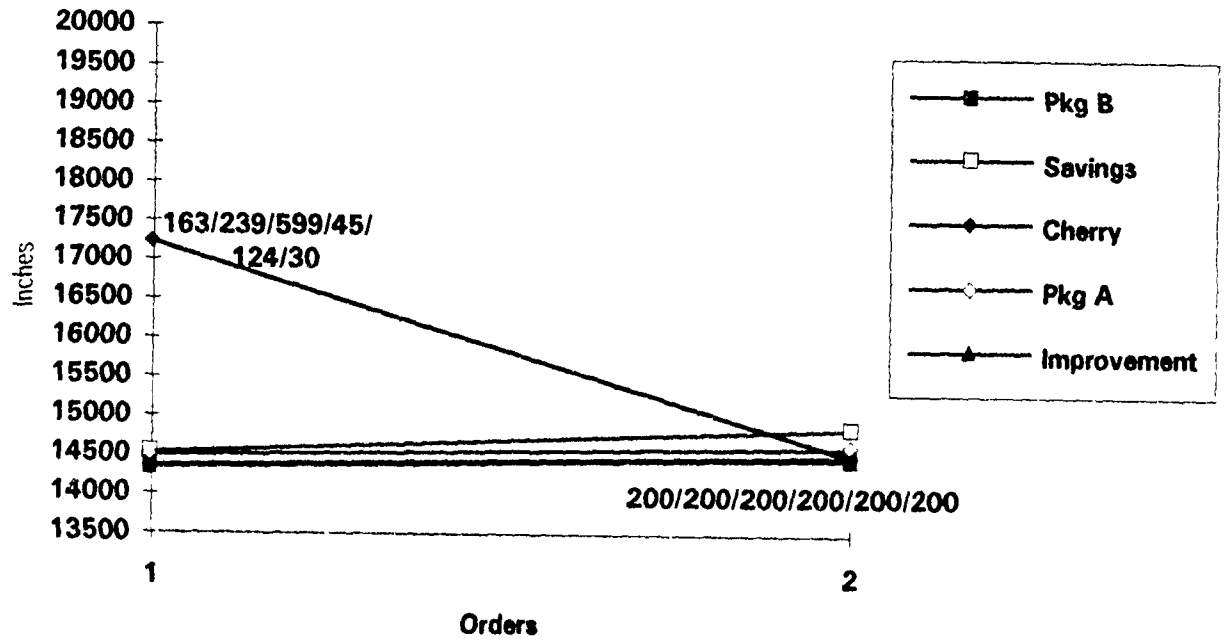


Figure 4.4.7 Total Fabric Inches for Pathological Orders with Ply Height 108

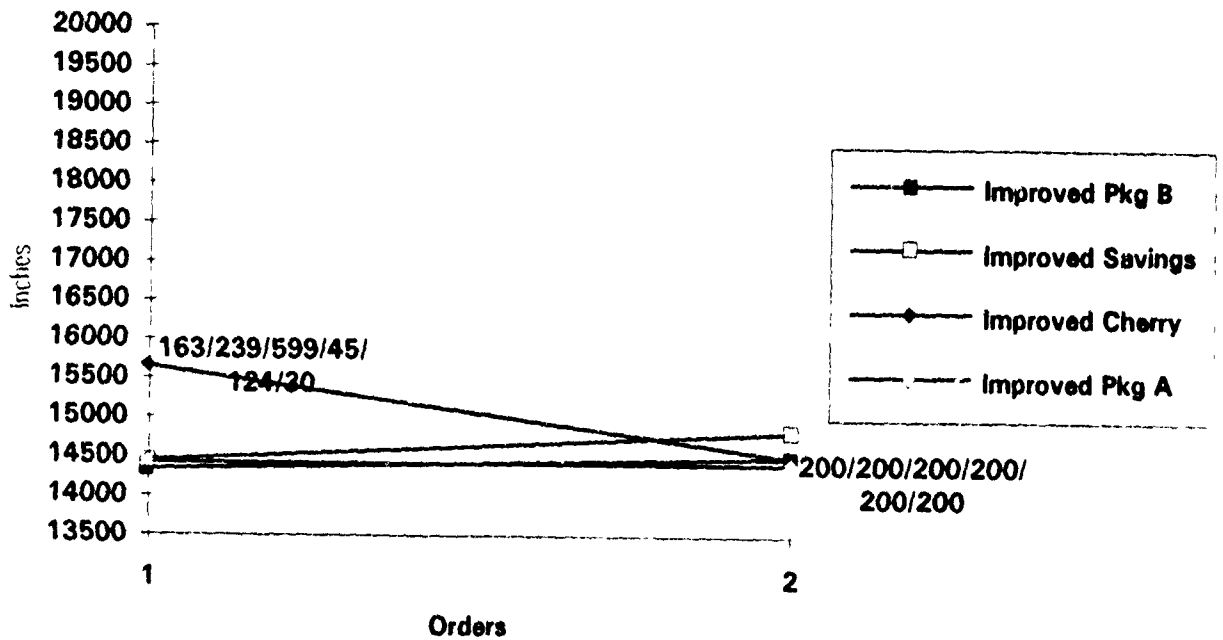


Figure 4.4.8 Total Fabric Inches for Improved Pathological Orders with Ply Height 108

The numerical results from all of the experimental problem instances are tabulated in Appendix E.

4.5 Conclusions for Theoretical Analysis

In the theoretical analysis of the Cut Order Planning problem, several things have been established. First, a mathematical model has been developed which represents the first known efforts to represent precisely the cut order planning problem. This model provides current and future researchers with a basis for extending the results of this work. In this project, the mathematical model was used to facilitate problem specification, to assist in the complexity analysis, and to initiate algorithm development.

Second, complexity analysis established that the cut order planning problem is a very complex one, requiring approximate algorithms for real-time solutions. This result is important because it demonstrates the **futility of efforts to produce an exact algorithm for COP** which is also efficient computationally. Another important insight from this result is no commercial package can solve COP exactly in real time, and therefore must resort to heuristic algorithms for solution. As a result, the heuristics in a particular commercial package are not guaranteed to produce solutions which dominate those in another package for all possible problem instances. This conclusion is useful in understanding why different manufacturing entities may prefer one package over another for their specific problem characteristics.

The third result derives from the second, namely the development of three heuristics for COP. Of these three, two constructive algorithms and one improvement approach were developed. For the representative problems developed in this project, one of the constructive algorithms produced solutions equivalent to those from commercial packages. Also, the improvement algorithm was able to improve all solutions generated for the testbed data, including those produced by the commercial packages. Because of its ability to enhance the constructive methods and the existing commercial packages, in addition to random solutions, the creation of the improvement algorithm is one of the major contributions of this research project.

The development of the testbed problems is another contribution of this project. Current and future researchers will be able to use these problems to benchmark other methodological developments. The testbed problems used to evaluate the various algorithms were very basic. The fabric lengths and other problem parameters were based on the details of numerous conversations with individuals who solve the cut order planning problem daily. Some of the problem instances were deliberately set up as nonstandard to examine the performance of the heuristics for realistic situations which may not occur frequently. No attempt was made to set up problem instances to represent all possible scenario in cut order planning; the number of such problems would be overwhelming. Instead, the testbed data has been created to be representative of common scenarios in COP.

5.0 Summary and Conclusions

5.1. Review of the project

This project has consisted of two major phases. In phase one, commercial packages for solving COP were identified. Two software vendors agreed to allow closer examination of their packages. These packages were comparatively analyzed, using testbed data. During phase two, a theoretical analysis of COP was carried out and solution methods were developed, implemented, and compared to each other and the commercial solution methods.

Results from the first phase led to an understanding of the relative performance of currently available software for cut order planning, and the relative priorities of the cost drivers for the planning decisions. The work in phase two produced a set of new algorithms for solving COP, implemented in a prototype software package.

5.2. Major contributions of the project

In this final section of the paper the major contributions of the research project is itemized and recommendations for further work in cut order planning research is made.

The first major contribution of this work to the knowledge base of cut order planning is the experimental verification of **fabric cost being the dominant determinant** of the cut order planning solution. This drives home the point that heuristic solutions, including those used in commercial packages, are **critically dependent** on the estimation of fabric length required to cut a particular combination of sizes together in a marker. Therefore, it is important that this information be obtained from historical data or correctly estimated by experienced operators.

Major analytical contributions of this work include the development of a mathematical model, the complexity analysis, and the improvement algorithm. The mathematical model of the cut order planning problem described in this report is the only one known to exist for modeling this difficult problem. The complexity analysis establishes that no solution method is likely to be able to produce mathematically optimal COP solutions, including commercial packages. Finally, an improvement algorithm was developed which has been shown to improve solutions produced by commercial methods. This algorithm is extremely versatile as well. It was able to produce solutions from a random starting solution which were essentially equivalent to improved solutions from commercial packages and other simple heuristics. This means that solutions can be found by anyone with a desktop PC-based computer and accurate information for marker lengths.

5.3 Recommendations for further COP research

The cut order planning problem as addressed in this project and solved in many operations is performed independently of downstream production considerations. This is done even though the output of COP is the direct input for marker making and the cutting room. Given the ever changing status of current orders, current work-in-process (WIP) in the system, and production configuration, there is a significant opportunity to make the production system more efficient and responsive by better coordination: dynamic, integrated planning and scheduling of WIP release, WIP movement, and configuration of the associated flexible production capacity. For these reasons, it is recommended that future research for COP extend this work to capitalize on the adaptive capabilities of the improvement algorithm and to explicitly include material flow control considerations. This work will allow for consideration of the current status of the assembly operations and the reflection of competing system objectives. This extension will bring the cut order planning process closer to real integration with the production planning process and implement one more step in the direction of true flexible manufacturing for the apparel industry.

6.0 References

- Anon. (1987), "Apparel Industry-Specific Software Packages," *Bobbin* (October), pp. 72-77.
- Anon. (1988-A), "More Computer Software Designed With Manufacturers in Mind," *Knitting Times* (June), pp. 83-84.
- Anon. (1988-B), "Directory of Software Suppliers," *Bobbin* (July), pp. 80-90.
- Bradley, Hax and Magnanti (1977), *Applied Mathematical Programming*, Addison-Wesley, Reading, Massachusetts.
- Czirik, J. (1989), "An On-line Algorithm for Variable-Sized Bin Packing," *Acta Informatica*, Vol. 26, pp.697-709.
- Eilon, S. and N. Christofides (1971), "The Loading Problem," *Management Science*, Vol. 17, pp. 259-267.
- Elsayed and Shetty (1988), "Computer Simulation of a Recursive Algorithm for Two and Three Dimensional Packing," *Proceedings of the International Conference on Computer Aided Design Graphics*, Vienna.
- Fadigan, Harry (1986), "Your Inventory has a Future", *Bobbin* (October), pp. 91-94.
- Farley (1988), "Mathematical Programming Models for Cutting-Stock Problems in the Clothing Industry," *Journal of the Operational Research Society*, Vol. 39, No. 1, pp. 41- 53.
- Francis, Richard L. and John A. White (1974), *Facility Layout and Location an Analytical Approach*. Prentice Hall, Inc., New Jersey.
- Garey, Michael R. and David S. Johnson (1979), *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York.
- Gilmore and Gomory (1961), "A Linear Programming Approach to the Cutting Stock Problem," *Operations Research*, Vol. 9, pp. 349-359.
- Hinxman (1980), "The Trim-loss and Assortment Problems: A Survey," *European Journal of Operational Research*, Vol. 5, pp. 8-18.
- Johnson, D.S., et al. (1974), "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms," *SIAM Journal of Computing*, Vol. 3, pp. 299-325.

- Kurt Salmon Associates (1987), "87 Software Survey," *Apparel Industry Magazine* (November), pp. 72-77.
- Martello, Silvano and Toth, Paolo (1990), *Knapsack Problems, Algorithms and Computer Implementations*. John Wiley & Sons, New York.
- Mendel, Sy (1987), "The Minicomputer Approach to MRP II," *Bobbin* (March), p. 106, 108, 114, 116.
- Montgomery, Douglas C., (1984), *Design and Analysis of Experiments, Second Edition*, John Wiley & Sons, New York.
- Plackett, R. L. and Burman, J. P. (1946), "Design of Optimal Multi-Factorial Experiments," *Biometrika*, Vol. 3, pp. 305-325.
- Rhee, W.T. and Talagrand, M. (1989), "Optimal Bin Packing with Items of Random Sizes II," *SIAM Journal of Computing* (February), Vol. 18, pp. 139-151.
- Riley, Sid (1988), "Automation Crosses the Line," *Bobbin* (October), pp. 84-87.
- Segal, Glen (1987), "Software Needs Hard Sell," *Bobbin* (October), pp. 58-68.
- Segal, Glen (1988), "Hard News on Software," *Bobbin* (July), pp. 80-90.
- Tang, C.S. and Denardo, E.V. (1988), "Models Arising from a Flexible Manufacturing Machine II, Minimization of the Number of Switching Instants," *Operations Research* (September-October), Vol. 36, pp. 778-784.
- Tompkins, James A. and White, John A. (1984), *Facility Planning*, John Wiley & Sons, New York.
- Tray, Anne I. (1987), "GGT Strives For Even Higher Grad(ing)s," *Bobbin* (June), pp. 111-116.
- White, Robert (1985), "Dividend of Computers is Information Management," *Apparel World* (January), pp. 32, 35.

Appendix A: Package A Solutions

Order: ABFHIK

Color\Sizes	34	
	1200	1200
	1200	1200

Parameter table name: 5

Max. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	10.00	Area of pattern in ins. ² :	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

ABFHIK-OP1

Sizes in marker: 6*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	5	Type of marker:	Open
Number of plies (NOT folds!):	200	Cost per unit:	3.73	Cost per unit (avg.):	3.73
Total costs:	4481.31	Total units in marker:	1200	Accumulated total units:	1200
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 2001

Order: AEGHJK

Color\Sizes	34	
	48	48
	48	48

Parameter table name:: 6

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:
Plus deviation/size/color:	10	Cost of cloth:	10.00	Area of pattern in ins.:
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins:
				Calculated efficiency in %:

1. > * * * * *

AEGHJK-0F1

Sizes in marker: 6*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:
Number of plies (NOT folds!):	8	Cost per unit:	4.20	Cost per unit (avg.):
Total costs:	201.59	Total units in marker:	48	Accumulated total units:
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:
Spreading method:	Zigzag			

: 8|

Order: DF61JK

Color\Sizes	30	32	34	36	38	40	
	8	8	8	8	8	8	48
	8	8	8	8	8	8	48

Parameter table name: 7

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	0.50	Area of pattern in ins.:	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

DF61JK-OP1

Sizes in marker: 1*30 1*32 1*34 1*36 1*38 1*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	8	Cost per unit:	0.53	Cost per unit (avg.):	0.53
Total costs:	25.59	Total units in marker:	48	Accumulated total units:	48
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 81

Order: CEFH1J

Color\Sizes	34	
	48	48
	48	48

Parameter table name: 8

av. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	0.50	Area of pattern in ins.²:	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins.:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

CEFH1J-OP1

zes in marker: 2424

Please enter the effectively produced earlier length here:

Total sizes in marker:	2	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	24	Cost per unit:	0.53	Cost per unit (avg.):	0.53
Total costs:	25.58	Total units in marker:	48	Accumulated total units:	48
Efficiency of marker in %:	70.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 24|

Order: BDEGHI

Color\Sizes	30	32	34	36	38	40	
	200	200	200	200	200	200	1200
	200	200	200	200	200	200	1200

Parameter table name: 9

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	0.50	Area of pattern in ins. ² :	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

BDEGHI-OP1

Sizes in marker: 1*30 1*32 1*34 1*36 1*38 1*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	2	Type of marker:	Open
Number of plies (NOT folds):	200	Cost per unit:	0.24	Cost per unit (avg.):	0.24
Total costs:	284.31	Total units in marker:	1200	Accumulated total units:	1200
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 200|

Order: ACDF6H

Color\Sizes	30	32	34	36	38	40	
	6	9	25	2	5	1	48
	6	9	25	2	5	1	48

Parameter table name: !

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	10.00	Area of pattern in ins. ² :	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins.:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

ACDF6H-OP1

Sizes in marker: 1*30 1*32 4*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	6	Cost per unit:	3.91	Cost per unit (avg.):	3.91
Total costs:	140.86	Total units in marker:	36	Accumulated total units:	36
Efficiency of marker in %:	84.00	% of total units in order:	75.00	Accumulated % of total order:	75.00
Spreading method:	Zigzag				

: 6|

2. + + + + +

ACDF6H-OP2

Sizes in marker: 3*32 1*34 2*36 5*38 1*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	12	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	1	Cost per unit:	6.45	Cost per unit (avg.):	4.55
Total costs:	77.37	Total units in marker:	12	Accumulated total units:	48

Efficiency of marker in %:	80.00	% of total units in order:	25.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 1|

Order: BCEFGK

Color\Sizes	34	
	1200	1200
	1200	1200

Parameter table name:: @

Max. no. of plies (not folds):	106	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	0.50	Area of pattern in ins. ² :	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

BCEFGK-OP1

Sizes in marker: 6*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	2	Type of marker:	Open
Number of plies (NOT folds!):	200	Cost per unit:	0.24	Cost per unit (avg.):	0.24
Total costs:	284.92	Total units in marker:	1200	Accumulated total units:	1200
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 200|

Order: HIGH

Color\Sizes	30	32	34	36	38	40	
	163	239	599	45	124	30	1200
	163	239	599	45	124	30	1200

Parameter table name: \$

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	10.00	Area of pattern in ins. ² :	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. * * * * *

HIGH-OP1

Sizes in marker: 9*30 9*32 9*34 3*36 8*38 2*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	40	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	15	Cost per unit:	3.92	Cost per unit (avg.):	3.92
Total costs:	2254.59	Total units in marker:	600	Accumulated total units:	600
Efficiency of marker in %:	80.00	% of total units in order:	50.00	Accumulated % of total order:	50.00
Spreading method:	210713				

: 15|

2. * * * * *

HIGH-OP2

Sizes in marker: 1*32 5*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	92	Cost per unit:	3.67	Cost per unit (avg.):	3.80
Total costs:	2025.20	Total units in marker:	552	Accumulated total units:	1152

Efficiency of marker in %: 84.00 % of total units in order: 46.00 Accumulated % of total order: 96.00
 Spreading method: Zigzag

: 92|

3. * * * * *

HIGH-OP3

Sizes in marker: 2*30

Please enter the effectively produced marker length here: _____

Total sizes in marker:	2	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	14	Cost per unit:	4.08	Cost per unit (avg.):	3.61
Total costs:	114.18	Total units in marker:	28	Accumulated total units:	1180
Efficiency of marker in %:	70.00	% of total units in order:	2.33	Accumulated % of total order:	98.33
Spreading method:	Zigzag				

: 14|

4. * * * * *

HIGH-OP4

Sizes in marker: 2*32

Please enter the effectively produced marker length here: _____

Total sizes in marker:	2	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	6	Cost per unit:	5.24	Cost per unit (avg.):	3.82
Total costs:	62.82	Total units in marker:	12	Accumulated total units:	1192
Efficiency of marker in %:	70.00	% of total units in order:	1.00	Accumulated % of total order:	99.33
Spreading method:	Zigzag				

: 6|

Order: ABDEFJ

Color\Sizes	30	32	34	36	38	40	
ONE COLOR	200	200	200	200	200	200	1200
	200	200	200	200	200	200	1200

Parameter table name:: 1

Max. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	10.00	Area of pattern in ins. ² :	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. * * * * *

ABDEFJ-0P1

Sizes in marker: 1*30 1*32 1*34 1*36 1*38 1*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	5	Type of marker:	Open
Number of plies (NOT folds!):	200	Cost per unit:	4.00	Cost per unit (avg.):	4.00
Total costs:	4602.40	Total units in marker:	1200	Accumulated total units:	1200
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

ONE COLOR: 2001

Order: BCDHJK

Color\Sizes	30	32	34	36	38	40	
	163	239	599	45	124	30	1200
	163	239	599	45	124	30	1200

Parameter table name:: 3

Max. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	10	Cost of cloth:	0.50	Area of pattern in ins. ² :	780
Minus deviation/size/color:	10	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. * * * * *

BCDHJK-OP1

Sizes in marker: 3*30 5*32 9*34 1*36 2*38

Please enter the effectively produced marker length here: _____

Total sizes in marker:	20	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	45	Cost per unit:	0.23	Cost per unit (avg.):	0.23
Total costs:	206.93	Total units in marker:	900	Accumulated total units:	900
Efficiency of marker in %:	80.00	% of total units in order:	75.00	Accumulated % of total order:	75.00
Spreading method:	Zigzag				

: 45|

2. * * * * *

BCDHJK-OP2

Sizes in marker: 5*34 1*38

Please enter the effectively produced marker length here: _____

Total sizes in marker:	5	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	34	Cost per unit:	0.27	Cost per unit (avg.):	0.24
Total costs:	54.11	Total units in marker:	204	Accumulated total units:	1104

Efficiency of marker in %:	84.00	% of total units in order:	17.00	Accumulated % of total order:	92.00
Spreading method:	Zigzag				

: 34|

3. * * * * *

BCDHJK-OP3

Sizes in marker: 2*30 1*32 1*34 2*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	14	Cost per unit:	0.35	Cost per unit (avg.):	0.24
Total costs:	29.64	Total units in marker:	84	Accumulated total units:	1188
Efficiency of marker in %:	84.00	% of total units in order:	7.00	Accumulated % of total order:	99.00
Spreading method:	Zigzag				

: 14|

4. * * * * *

BCDHJY-OP4

Sizes in marker: 2*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	2	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	5	Cost per unit:	0.90	Cost per unit (avg.):	0.25
Total costs:	9.02	Total units in marker:	10	Accumulated total units:	1198
Efficiency of marker in %:	70.00	% of total units in order:	0.83	Accumulated % of total order:	99.83
Spreading method:	Zigzag				

: 5|

Order: ACDEIK

Color\Sizes	30	32	34	36	38	40	
	6	9	25	2	5	1	48
	6	9	25	2	5	1	48

Parameter table name: 2

Max. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	10.00	Area of pattern in ins.²:	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins.:	19.21
				Calculated efficiency in %:	70.01

1. + + + + +

ACDEIK-OP1

Sizes in marker: 1*30 1*32 4*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	6	Cost per unit:	4.16	Cost per unit (avg.):	4.16
Total costs:	149.65	Total units in marker:	36	Accumulated total units:	36
Efficiency of marker in %:	84.00	% of total units in order:	75.00	Accumulated % of total order:	75.00
Spreading method:	Zigzag				

: 6|

2. + + + + +

ACDEIK-OP2

Sizes in marker: 3*32 1*34 2*36 5*38 1*40

Please enter the effectively produced marker length here: _____

Total sizes in marker:	12	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	1	Cost per unit:	8.08	Cost per unit (avg.):	5.14
Total costs:	97.00	Total units in marker:	12	Accumulated total units:	48

Efficiency of marker in %: 80.00
Spreading method: Zigzag

% of total units in order: 25.00 Accumulated % of total order: 100.00

: 1 |

Order: LOW

Color/Sizes

	48	48
	48	48

Parameter table name:: ABCGIJ

Max. no. of plies (not folds):	47	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	0.50	Area of pattern in ins. ² :	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21

1. * * * * *

LOW-OP1

Sizes in marker: 2*Medium

Please enter the effectively produced marker length here: _____

Total sizes in marker:	2	Number of lays:	1	Type of marker:	Open
Number of plies (NOT folds!):	24	Cost per unit:	0.38	Cost per unit (avg.):	0.38
Total costs:	18.10	Total units in marker:	48	Accumulated total units:	48
Efficiency of marker in %:	70.00	% of total units in order:	100.00	Accumulated % of total order:	100.

: 24|

Order: ABCGIJ

Color\Sizes	34	
	1200	1200
	1200	1200

Parameter table name:: 4

Max. no. of plies (not folds):	108	Max. no. of sizes in marker:	99	Perimeter of pattern in ins.:	317
Plus deviation/size/color:	0	Cost of cloth:	10.00	Area of pattern in ins. ² :	780
Minus deviation/size/color:	0	Usable width of cloth in ins.:	58.00	Calculated usage in ins:	19.21
				Calculated efficiency in %:	70.01

1. * * * * *

ABCGIJ-OP1

Sizes in marker: 6*34

Please enter the effectively produced marker length here: _____

Total sizes in marker:	6	Number of lays:	2	Type of marker:	Open
Number of plies (NOT folds!):	200	Cost per unit:	3.70	Cost per unit (avg.):	3.70
Total costs:	4438.84	Total units in marker:	1200	Accumulated total units:	1200
Efficiency of marker in %:	84.00	% of total units in order:	100.00	Accumulated % of total order:	100.00
Spreading method:	Zigzag				

: 200|

Appendix B: Package B Solutions

Parameters: ABDEFJ

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
-----	-----	-----	-----	-----	-----
1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units
Red	200	200	200	200	200	200	1200
Total	200	200	200	200	200	200	1200

Marker solution..... :

Marker 1 Repeat 3 Plies 100 Units 1000
S1 S2 S3 S4 S5 Sizes

2 2 2 2 2 10

Marker 2 Repeat 1 Plies 33 Units 198
S6 Sizes

6 6

Marker 3 Repeat 1 Plies 2 Units 2
S6 Sizes

1 1

	Marker	Size	Repeat	Plies	Units
Total	3	17	5	135	1200

Cutting Order Report :

Colors: Red

Marker Repeat Plies YD

1	3	100	327-5.91
2	1	33	69-19.32
3	1	2	0-33.17

Total 5 135 397-22.41

Total Plies.....

Marker Repeat Plies YD

1	3	100	327-5.91
2	1	33	69-19.32
3	1	2	0-33.17

Total 5 135 397-22.41

Cutting Order Report :

Marker	1	2	3	Total
Colors Repeat	[3]	[1]	[1]	[5]
=====	=====	=====	=====	=====
Red	100	33	2	135
=====	=====	=====	=====	=====
Total	100	33	2	135

Deviation report.... :

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	0	0	0	0	0	0	0
-----	-----	-----	-----	-----	-----	-----	-----
Total	0	0	0	0	0	0	0

Units produced

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	200	200	200	200	200	200	1200
-----	-----	-----	-----	-----	-----	-----	-----
Total	200	200	200	200	200	200	1200

Unit solution..... :

Marker number.....: 1
Spread length.....:3-9.78

Repeat [1/3]

Colors YD	Plies	S1	S2	S3	S4	S5	Total
Sizes Marked.....		2	2	2	2	2	10
Red 153-27.62	47	94	94	94	94	94	470
Total 153-27.62	47	94	94	94	94	94	470

39.17%

=====

Marker number.....: 1
Spread length.....:3-9.78

Repeat [2/3]

Colors YD	Plies	S1	S2	S3	S4	S5	Total
Sizes Marked.....		2	2	2	2	2	10
Red 153-27.62	47	94	94	94	94	94	470
total 153-27.62	47	94	94	94	94	94	470

39.17%

=====

Marker number.....: 1
Spread length.....:3-9.78

Repeat [3/3]

Colors YD	Plies	S1	S2	S3	S4	S5	Total
Sizes Marked.....		2	2	2	2	2	10
Red 19-22.67	6	12	12	12	12	12	60
Total 19-22.67	6	12	12	12	12	12	60

5.00%

=====

Marker number.....: 2
Spread length.....:2-3.86

Repeat [1/1]

Colors YD	Plies	S6	Total
Sizes Marked.....		6	6
Red 69-19.32	33	198	198

Total 69-19.32 33 198 198

16.50%

=====

Marker number.....: 3
Spread length.....: 0-16.59

Repeat [1/1]

Colors	YD	Plies	56	Total
Sizes Marked.....		1		1

Red	0-33.17	2	2	2

Total	0-33.17	2	2	2

0.17%

=====

Material Consumption

Colors	Length	Rest

Red	391-22.41	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	3	Marker Origin.....	New
Est Eff.....	85.55	Marker width.....IN	58.00	Marker length.....YD	3- 7.28
S1 S2 S3 S4 S5	Sizes				
2 2 2 2 2	10				
Marking cost.....\$	12.05	Total Plies.....	100	Spread length.....YD	3- 9.78
Spreading cost.....\$	12.20	# bundles.....	84	Fabric used.....YD	327-6
Cutting cost/Hour.....\$	42.21	Units/plie yield.....	1	Spreading time.....Hours	0.49
Bundling cost.....\$	8.40	Plies/bundle.....	12	Cutting time.....Hours	1.41
Fabric cost.....\$	3271.64	Fabric cost.....\$ /YD	10.00	total time.....Hours	1.89
total cost.....\$	3346.50	total units.....	1000	Avg Cost/unit.....\$	3.35

Marker number.....	2	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	2- 1.36
S6	Sizes				
6 6					
Marking cost.....\$	7.55	Total Plies.....	33	Spread length.....YD	2- 3.86
Spreading cost.....\$	7.02	# bundles.....	17	Fabric used.....YD	69-19
Cutting cost/Hour.....\$	9.84	Units/plie yield.....	1	Spreading time.....Hours	0.28
Bundling cost.....\$	1.70	Plies/bundle.....	12	Cutting time.....Hours	0.33
Fabric cost.....\$	685.37	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.61
total cost.....\$	721.48	total units.....	198	Avg Cost/unit.....\$	3.64

Marker number.....	3	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	74.05	Marker width.....IN	58.00	Marker length.....YD	0-14.09
S6	Sizes				
1 1					
Marking cost.....\$	2.25	Total Plies.....	2	Spread length.....YD	0-16.59
Spreading cost.....\$	5.09	# bundles.....	1	Fabric used.....YD	0-33
Cutting cost/Hour.....\$	3.72	Units/plie yield.....	1	Spreading time.....Hours	0.20
Bundling cost.....\$	0.10	Plies/bundle.....	12	Cutting time.....Hours	0.12
Fabric cost.....\$	9.21	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.33
total cost.....\$	20.38	total units.....	2	Avg Cost/unit.....\$	10.19

Total cost summary.. :

Total Markers.....	3	AVG eff.....	85.49	Marker width.....	58.00
Marking cost.....\$	21.85	Number of stacks.....	5	Sizes Marked.....	17
Spreading cost.....\$	24.31	# bundles.....	102	Total Plies.....	135
Cutting cost/Hour.....\$	55.78	Units/plie yield.....	1	Spreading time.....Hours	0.97
Bundling cost.....\$	10.20	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	1.86
Fabric cost.....\$	3976.22	Fabric used.....YD	397.62	total time.....Hours	2.83
total cost.....\$	4088.36	total units.....	1200	Avg Cost/unit.....\$	3.41

Graphics Output..... :

Marker number.....: 1 () Repeat [1/3], 1
 SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	47	153-27.6YD
Total	47	

<----- 3- 9.78 ----->

Sizes	S1	S2	S3	S4	S5	Total
Sizes Marked.....	2	2	2	2	2	10
Total Units	94	94	94	94	94	470

Graphics Output..... : _____

Marker number.....: 1 ()

Repeat [2/3], 2

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	47	153-27.6YD
Total	47	

<----- 3- 9.78 ----->

Sizes	S1	S2	S3	S4	S5	Total
Sizes Marked.....	2	2	2	2	2	10

Total	Units	94	94	94	94	94	470

Graphics Output..... : _____

Marker number.....: 1 ()

Repeat [3/3], 3

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	6	19-22.7YD
Total	6	

<----- 3- 9.78 ----->

Sizes	S1	S2	S3	S4	S5	Total
Sizes Marked.....	2	2	2	2	2	10

Total	Units	12	12	12	12	12	60

Graphics Output..... :

Marker number.....: 2 ()

Repeat [1/1], 4

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	33	69-19.3YD
Total	33	

<----- 2- 3.86 ----->

Sizes S6 Total
Sizes Marked..... 6 6

Total Units 198 198

Graphics Output..... :

Marker number.....: 3 ()

Repeat [1/1], 5

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	2	0-33.2YD
Total	2	

<----- 0-16.59 ----->

Sizes S6 Total
Sizes Marked..... 1 1

Total Units 2 2

Marker Solution II.. :

Marker	Sizes						Sizes/Marker	Repeat	Flies	Units
	S1	S2	S3	S4	S5	S6				
	2	2	2	2	2	-	10	3	100	1000
	-	-	-	-	-	6	6	1	33	198
	-	-	-	-	-	1	1	1	2	2
3							17	5	135	1200

Optimized Spreading :

14	Red	
27	Red	
4	Red	
2	Red	
< 3- 7.28 >< 3- 7.28 >< 2- 1.36 >< 3- 7.28 >< 0-14.09 >		

Colors	Plies	StartPoint:	EndPoint: Spread length.....	
Red	2	0- 0.00	12- 2.53	24- 7.56
Red	4	0- 0.00	11-24.45	46-30.78
Red	27	0- 0.00	8-17.17	229-29.24
Red	14	0- 0.00	6-15.81	90-22.81

Parameters: ACDEIK

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	8.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	257
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6

1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10

74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units

Red	6	9	25	2	5	1	48

Total	6	9	25	2	5	1	48

Marker solution..... :

Marker 1 Repeat 1 Plies 6 Units 36
S1 S2 S3 Sizes

1 1 4 6

Marker 2 Repeat 1 Plies 3 Units 6
S2 S5 Sizes

1 1 2

Marker 3 Repeat 1 Plies 2 Units 4
S4 S5 Sizes

1 1 2

Marker 4 Repeat 1 Plies 1 Units 2
S3 S6 Sizes

1 1 2

Marker	Sizes	Repeat	Plies	Units
Total	4	12	4	12

Cutting Order Report :

Colors: Red

Marker Repeat Plies YD

1	1	6	11-31.14
2	1	3	2-15.05
3	1	2	1-23.04
4	1	1	0-29.52

total 4 12 16-26.75

total Plies.....

Marker Repeat Plies YD

1	1	6	11-31.14
2	1	3	2-15.05
3	1	2	1-23.04
4	1	1	0-29.52

total 4 12 16-26.75

Cutting Order Report :

Marker	1	2	3	4	Total
Colors Repeat	[1]	[1]	[1]	[1]	[4]
=====	=====	=====	=====	=====	=====
Red	6	3	2	1	12
=====	=====	=====	=====	=====	=====
Total	6	3	2	1	12

Deviation report.... :

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	0	0	0	0	0	0	0
=====	=====	=====	=====	=====	=====	=====	=====
Total	0	0	0	0	0	0	0

Units produced

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	6	9	25	2	5	1	48
=====	=====	=====	=====	=====	=====	=====	=====
Total	6	9	25	2	5	1	48

Unit solution..... :

Marker number.....: 1
Spread length.....: 1-35.19

Repeat [1/1]

Colors	YD	Plies	S1	S2	S3	Total
Sizes Marked.....			1	1	4	6
Red	11-31.14	6	6	6	24	36
Total	11-31.14	6	6	6	24	36

75.00%

Marker number.....: 2
Spread length.....: 0-29.02

Repeat [1/1]

Colors	YD	Plies	S2	S5	Total
Sizes Marked.....			1	1	2
Red	2-15.05	3	3	3	6
Total	2-15.05	3	3	3	6

12.50%

Marker number.....: 3
Spread length.....: 0-29.52

Repeat [1/1]

Colors	YD	Plies	S4	S5	Total
Sizes Marked.....			1	1	2
Red	1-23.04	2	2	2	4
Total	1-23.04	2	2	2	4

8.33%

Marker number.....: 4
Spread length.....: 0-29.52

Repeat [1/1]

Colors	YD	Plies	S3	S6	Total
Sizes Marked.....		1	1		2

Red	0-29.52	1	1	1	2

Total	0-29.52	1	1	1	2

4.17%

=====

Material Consumption

Colors	Length	Rest

Red	16- 7.55	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-32.69
S1 S2 S3 Sizes					
1 1 4 6					
Marking cost.....\$	7.55	Total Plies.....	6	Spread length.....YD	1-35.19
Spreading cost.....\$	1.72	# bundles.....	4	Fabric used.....YD	11-31
Cutting cost/Hour.....\$	9.38	Units/plie yield.....	1	Spreading time.....Hours	0.21
Bundling cost.....\$	0.40	Plies/bundle.....	12	Cutting time.....Hours	0.31
Fabric cost.....\$	118.65	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.53
total cost.....\$	137.69	total units.....	36	Avg Cost/unit.....\$	3.82

Marker number.....	2	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	75.10	Marker width.....IN	58.00	Marker length.....YD	0-26.52
S2 S5 Sizes					
1 1 2					
Marking cost.....\$	3.25	Total Plies.....	3	Spread length.....YD	0-29.02
Spreading cost.....\$	1.65	# bundles.....	1	Fabric used.....YD	2-15
Cutting cost/Hour.....\$	4.84	Units/plie yield.....	1	Spreading time.....Hours	0.21
Bundling cost.....\$	0.10	Plies/bundle.....	12	Cutting time.....Hours	0.16
Fabric cost.....\$	24.18	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.37
total cost.....\$	34.01	total units.....	6	Avg Cost/unit.....\$	5.67

Marker number.....	3	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	75.10	Marker width.....IN	58.00	Marker length.....YD	0-27.02
S4 S5 Sizes					
1 1 2					
Marking cost.....\$	3.25	Total Plies.....	2	Spread length.....YD	0-29.52
Spreading cost.....\$	1.63	# bundles.....	1	Fabric used.....YD	1-23
Cutting cost/Hour.....\$	4.88	Units/plie yield.....	1	Spreading time.....Hours	0.20
Bundling cost.....\$	0.10	Plies/bundle.....	12	Cutting time.....Hours	0.16
Fabric cost.....\$	16.40	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.37
total cost.....\$	26.26	total units.....	4	Avg Cost/unit.....\$	6.57

Marker number.....	4	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	75.10	Marker width.....IN	58.00	Marker length.....YD	0-27.02
S3 S6 Sizes					
1 1 2					
Marking cost.....\$	3.25	Total Plies.....	1	Spread length.....YD	0-29.52
Spreading cost.....\$	1.62	# bundles.....	1	Fabric used.....YD	0-30
Cutting cost/Hour.....\$	4.88	Units/plie yield.....	1	Spreading time.....Hours	0.20
Bundling cost.....\$	0.10	Plies/bundle.....	12	Cutting time.....Hours	0.16
Fabric cost.....\$	8.20	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.36
total cost.....\$	18.05	total units.....	2	Avg Cost/unit.....\$	9.02

Total cost summary.. :

Total Markers.....	4	AVG eff.....	82.76	Marker width.....	58.00
Marking cost.....\$	17.30	Number of stacks.....	4	Sizes Marked.....	12
Spreading cost.....\$	6.61	# bundles.....	7	Total Plies.....	12
Cutting cost/Hour.....\$	23.97	Units/plie yield.....	1	Spreading time.....Hours	0.83
Bundling cost.....\$	0.70	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	0.80
Fabric cost.....\$	167.43	Fabric used.....YD	16.74	total time.....Hours	1.63
total cost.....\$	216.01	total units.....	48	Avg Cost/unit.....\$	4.50

Graphics Output..... :

Marker number.....: 1 ()

Repeat [1/1], 1

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	6	11-31.1YD
Total	6	

1-35.19

Sizes	S1	S2	S3	Total
Sizes Marked.....	1	1	4	6
Total Units	6	6	24	36

Graphics Output..... :

Marker number.....: 2 ()

Repeat [1/1], 2

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	3	2-15.0YD
Total	3	

< 0-29.02 >

Sizes	S2	S5	Total
Sizes Marked.....	1	1	2

Total	Units	3	3
		6	

Graphics Output..... :

Marker number.....: 3 ()

Repeat [1/1], 3

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	2	1-23.0YD
Total	2	

< 0-29.52 >

Sizes	S4	S5	Total
Sizes Marked.....	1	1	2

Total	Units	2	2
		4	

Graphics Output..... :

Marker number.....: 4 ()

Repeat [1/1], 4

spreadMethod Zig/Zag

Colors Plies Spread length.....

Red	1	0-29.5YD
Total	1	

<----- 0-29.52 ----->

Sizes S3 S6 Total

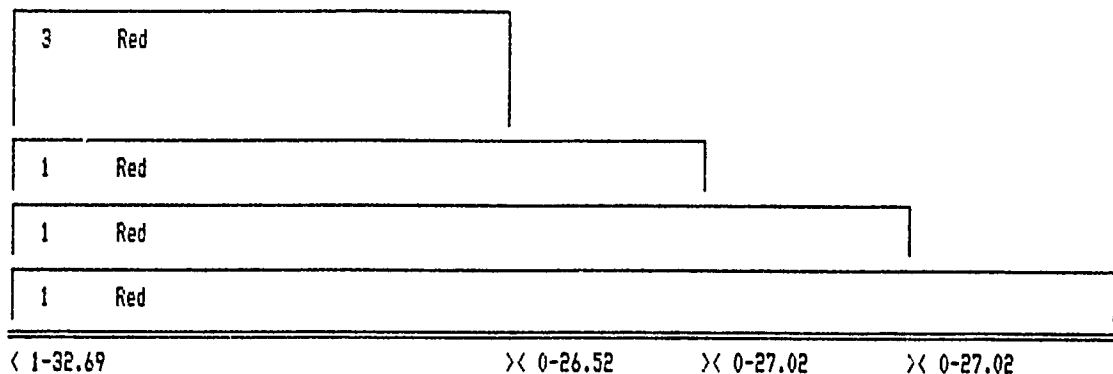
Sizes Marked..... 1 1 2

Total Units 1 1 2

Marker Solution II.. :

Marker	Sizes						Sizes/Marker	Repeat	Plies	Units
	S1	S2	S3	S4	S5	S6				
1	1	1	4	-	-	-	6	1	6	36
2	-	1	-	-	1	-	2	1	3	6
3	-	-	-	1	1	-	2	1	2	4
4	-	-	1	-	-	1	2	1	1	2
4							12	4	12	48

Optimized Spreading :



Colors	Plies	StartPoint:	EndPoint:Spread length.....	
Red	1	0- 0.00	4- 6.50	4- 7.75
Red	1	0- 0.00	3-15.48	3-16.73
Red	1	0- 0.00	2-24.46	2-25.71
Red	3	0- 0.00	1-33.94	5-33.57

Parameters: ACDFGH

problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6

1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10

74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units

Red	6	9	25	2	5	1	46

Total	6	9	25	2	5	1	48

Marker solution..... :

Marker	1	Repeat	1	Plies	11	Units	88
S1	S2	S3	S4	S5	S6	Sizes	

1	1	3	1	1	1	8	

Marker	Sizes	Repeat	Plies	Units

Total	1	8	11	88

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD

1	1	11	29-6.70

Total	1	11	29-6.70

=====

Total Plies.....

Marker	Repeat	Plies	YD

1	1	11	29-6.70

Total	1	11	29-6.70

Cutting Order Report :

Marker	1	Total
Colors	Repeat	[1] [1]

Red	11	11

Total	11	11

Deviation report.... :

=====

Colors	S1	S2	S3	S4	S5	S6	Units

Red	5	2	8	9	6	10	40

Total	5	2	8	9	6	10	40

Units produced

=====

Colors	S1	S2	S3	S4	S5	S6	Units

Red	11	11	33	11	11	11	88

Total	11	11	33	11	11	11	88

Unit solution..... :

Marker number.....: 1

Repeat [1/1]

pread length.....: 2-23.52

Colors	YD	Plies	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....			1	1	3	1	1	1	8
Red 29- 6.70		11	11	11	33	11	11	11	88
Total 29- 6.70		11	11	11	33	11	11	11	88

100.00%

=====

Material Consumption

Colors	Length	Rest
Red	28-25.10	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
st Eff.....	85.43	Marker width.....IN	58.00	Marker length.....YD	2-21.02
S1 S2 S3 S4 S5 S6	Sizes				
1 1 3 1 1 1	8				
arking cost.....\$	9.81	Total Plies.....	11	Spread length.....YD	2-23.52
Spreading cost.....\$	5.73	# bundles.....	8	Fabric used.....YD	29-7
Cutting cost/Hour.....\$	3.94	Units/plie yield.....	1	Spreading time.....Hours	0.23
undling cost.....\$	0.80	Plies/bundle.....	12	Cutting time.....Hours	0.39
Fabric cost.....\$	291.8t	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.62
total cost.....\$	312.14	total units.....	88	Avg Cost/unit.....\$	3.55

Total cost summary.. :

Total Markers.....	1	AVG eff.....	85.43	Marker width.....	58.00
Marking cost.....\$	9.81	Number of stacks.....	1	Sizes Marked.....	8
Spreading cost.....\$	5.73	# bundles.....	8	Total Plies.....	11
Setting cost/Hour.....\$	3.94	Units/plie yield.....	1	Spreading time.....Hours	0.23
Bundling cost.....\$	0.80	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	0.39
Fabric cost.....\$	291.86	Fabric used.....YD	29.19	total time.....Hours	0.62
Total cost.....\$	312.14	total units.....	88	Avg Cost/unit.....\$	3.55

Graphics Output..... :

Marker number.....: 1 ()

Repeat [1/1], 1

spreadMethod Zig/Zag

Colors Plies Spread length.....

Red	11	29- 6.7YD
Total	11	

<----- 2-23.52 ----->

Sizes	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....	1	1	3	1	1	1	8

Total	Units	11	11	33	11	11	11	88

Marker Solution II.. :

Marker	Sizes						Sizes/Marker	Repeat	Plies	Units
	S1	S2	S3	S4	S5	S6				
1	1	1	3	1	1	1	8	1	11	88
1							8	1	11	88

Optimized Spreading : _____

11	Red

< 2-21.02

Colors	Plies	StartPoint:	EndPoint:Spread length.....
Red	11	0- 0.00	2-22.27 29- 6.70

Parameters: BCDHJK

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	8.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units
Red	163	239	599	45	124	30	1200
Total	163	239	599	45	124	30	1200

Marker solution..... :

Marker 1 Repeat 3 Plies 121 Units 1089

S1 S2 S3 S5 Sizes

1 2 5 1 9

Marker 2 Repeat 2 Plies 52 Units 104

S1 S4 Sizes

1 1 2

Marker 3 Repeat 1 Plies 40 Units 40

S6 Sizes

1 1

Marker	Sizes	Repeat	Plies	Units

Total	3	12	6	213	1233

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD

1	3	121	355-31.58

2	2	52	41-6.59

3	1	40	18-15.46

Total	6	213	415-17.62

total Plies.....

Marker	Repeat	Plies	YD

1	3	121	355-31.58

2	2	52	41-6.59

3	1	40	18-15.46

Total	6	213	415-17.62

Cutting Order Report :

Marker	1	2	3	Total
Colors Repeat	[3]	[2]	[1]	[6]
=====	=====	=====	=====	=====
ed	121	52	40	213
=====	=====	=====	=====	=====
Total	121	52	40	213

Deviation report.... :

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	10	3	6	7	-3	10	33
=====	=====	=====	=====	=====	=====	=====	=====
Total	10	3	6	7	-3	10	33

Units produced

Colors	S1	S2	S3	S4	S5	S6	Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	173	242	605	52	121	40	1233
=====	=====	=====	=====	=====	=====	=====	=====
Total	173	242	605	52	121	40	1233

Unit solution..... :

Marker number.....: 1
Spread length.....:2-33.88

Repeat [1/3]

Colors	YD	Plies	S1	S2	S3	S5	Total
Sizes Marked.....			1	2	5	1	9

Red 138-	8.40	47	47	94	235	47	423

Total 138-	8.40	47	47	94	235	47	423

34.31%

=====

Marker number.....: 1
Spread length.....:2-33.88

Repeat [2/3]

Colors	YD	Plies	S1	S2	S3	S5	Total
Sizes Marked.....			1	2	5	1	9

Red 138-	8.40	47	47	94	235	47	423

Total 138-	8.40	47	47	94	235	47	423

34.31%

=====

Marker number.....: 1
Spread length.....:2-33.88

Repeat [3/3]

Colors	YD	Plies	S1	S2	S3	S5	Total
Sizes Marked.....			1	2	5	1	9

Red 79-	14.78	27	27	54	135	27	243

Total 79-	14.78	27	27	54	135	27	243

19.71%

=====

Marker number.....: 2
Spread length.....:0-28.51

Repeat [1/2]

Colors	YD	Plies	S1	S4	Total

Total 37- 8.03 47 47 47 94

7.62%

Marker number.....: 2
Spread length.....:0-28.51

Repeat [12/2]

Colors YD Plies S1 S4 Total
Sizes Marked..... 1 1 2

Red 3-34.56 5 5 5 10

Total 3-34.56 5 5 5 10

0.81%

Marker number.....: 3
Spread length.....:0-16.59

Repeat [1/1]

Colors YD Plies S6 Total
Sizes Marked..... 1 1

Red 18-15.46 40 40 40

Total 18-15.46 40 40 40

3.24%

MaterialConsumption

Colors Length Rest

Red 406- 0.82

Marker cost summary. :

Marker number.....	1	Number of stacks.....	3	Marker Origin.....	New
Est Eff.....	85.49	Marker width.....IN	58.00	Marker length.....YD	2-31.38
S1 S2 S3 S5	Sizes				
1 2 5 1	9				
Marking cost.....\$	10.97	Total Plies.....	121	Spread length.....YD	2-33.88
Spreading cost.....\$	4.27	# bundles.....	91	Fabric used.....YD	355-32
Cutting cost/Hour.....\$	12.87	Units/plie yield.....	1	Spreading time.....Hours	0.53
Bundling cost.....\$	9.10	Plies/bundle.....	12	Cutting time.....Hours	1.29
Fabric cost.....\$	177.94	Fabric cost.....\$ /YD	0.50	total time.....Hours	1.82
total cost.....\$	215.14	total units.....	1089	Avg Cost/unit.....\$	0.20

Marker number.....	2	Number of stacks.....	2	Marker Origin.....	New
Est Eff.....	75.10	Marker width.....IN	58.00	Marker length.....YD	0-26.01
S1 S4	Sizes				
1 1	2				
Marking cost.....\$	3.25	Total Plies.....	52	Spread length.....YD	0-28.51
Spreading cost.....\$	2.42	# bundles.....	9	Fabric used.....YD	41-7
Cutting cost/Hour.....\$	3.19	Units/plie yield.....	1	Spreading time.....Hours	0.30
Bundling cost.....\$	0.90	Plies/bundle.....	12	Cutting time.....Hours	0.32
Fabric cost.....\$	20.59	Fabric cost.....\$ /YD	0.50	total time.....Hours	0.62
total cost.....\$	30.35	total units.....	104	Avg Cost/unit.....\$	0.29

Marker number.....	3	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	74.05	Marker width.....IN	58.00	Marker length.....YD	0-14.09
S6	Sizes				
1	1				
Marking cost.....\$	2.25	Total Plies.....	40	Spread length.....YD	0-16.59
Spreading cost.....\$	2.19	# bundles.....	4	Fabric used.....YD	18-15
Cutting cost/Hour.....\$	1.24	Units/plie yield.....	1	Spreading time.....Hours	0.27
Bundling cost.....\$	0.40	Plies/bundle.....	12	Cutting time.....Hours	0.12
Fabric cost.....\$	9.21	Fabric cost.....\$ /YD	0.50	total time.....Hours	0.40
total cost.....\$	15.29	total units.....	40	Avg Cost/unit.....\$	0.38

Total cost summary.. :

Total Markers.....	3	AVG eff.....	84.24	Marker width.....	58.00
Marking cost.....\$	16.47	Number of stacks.....	6	Sizes Marked.....	12
Spreading cost.....\$	8.87	# bundles.....	104	Total Plies.....	213
Cutting cost/Hour.....\$	17.30	Units/plie yield.....	1	Spreading time.....Hours	1.11
Bundling cost.....\$	10.40	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	1.73
Fabric cost.....\$	207.74	Fabric used.....YD	415.49	total time.....Hours	2.84
total cost.....\$	260.79	total units.....	1233	Avg Cost/unit.....\$	0.21

Graphics Output..... :

Marker number.....: 1 ()
 SpreadMethod Zig/Zag

Repeat [1/3], 1

Colors Plies Spread length.....

Red	47	138- 8.4YD
Total	47	.

<----- 2-33.68 ----->

Sizes	S1	S2	S3	S5	Total
Sizes Marked.....	1	2	5	1	9
Total Units	47	94	235	47	423

Graphics Output..... :

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [2/3], 2

Colors Plies Spread length.....

Red	47	138- 8.4YD
Total	47	.

<----- 2-33.88 ----->

Sizes S1 S2 S3 S5 Total
Sizes Marked..... : 2 5 1 9

Total Units 47 94 235 47 423

Graphics Output..... :

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [3/3], 3

Colors Plies Spread length.....

Red	27	79-14.8YD
Total	27	.

<----- 2-33.88 ----->

Sizes S1 S2 S3 S5 Total
Sizes Marked..... : 1 2 5 1 9

Total Units 27 54 135 27 243

Graphics Output..... :

Repeat [1/2], 4

Marker number.....: 2 ()
SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	47	37- 8.0YD
Total	47	

<----- 0-28.51 ----->

Sizes	S1	S4	Total
Sizes Marked.....	1	1	2
Total	Units	47	47
			94

5 Red

01

< 0-26.01

>> 2-31.38

>< 2-31.38

Colors	Plies	StartPoint:	EndPoint:Spre
Red	5	0- 0.00	10-17.50
Red	22	0- 0.00	9-27.49
Red	13	0- 0.00	6-32.11
Red	7	0- 0.00	6-18.02

Parameters: high

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximue plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimue plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	Mediu	large	xlarg	38X42	40X40	Units
Red	163	239	599	45	124	30	1200
Total	163	239	599	45	124	30	1200

Marker solution..... :

Marker 1 Repeat 1 Plies 101 Units 1010
SI Mediularge 38X42 Sizes

1 2 6 1 10

Marker 2 Repeat 1 Plies 23 Units 207
SI Mediuxlarg 38X42 40X40 Sizes

3 2 2 1 1 9

Marker Sizes Repeat Plies Units

Total 2 19 2 124 1217

Cutting Order Report :

Colors: Red

Marker Repeat Plies YD

1 1 101 329-6.91

2 1 23 67-28.36

Total 2 124 396-35.27

Total Plies.....

Marker Repeat Plies YD

1 1 101 329-6.91

2 1 23 67-28.36

Total 2 124 396-35.27

Cutting Order Report :

Marker	1	2	Total
Colors Repeat	[1]	[1]	[2]
=====	=====	=====	=====
ed	101	23	124
=====	=====	=====	=====
Total	101	23	124

Deviation report.... :

Colors	SI Mediulargexlarg38X4240X40						Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	7	9	7	1	0	-7	17
-----	-----	-----	-----	-----	-----	-----	-----
Total	7	9	7	1	0	-7	17

Units produced

Colors	SI Mediulargexlarg38X4240X40						Units
=====	=====	=====	=====	=====	=====	=====	=====
Red	170	248	606	46	124	23	1217
-----	-----	-----	-----	-----	-----	-----	-----
Total	170	248	606	46	124	23	1217

Unit solution..... :

Marker number.....: 1
Spread length.....:3-9.34

Repeat [1/1]

Colors	YD	Plies	S1	Mediu	large	38X42	Total
Sizes Marked.....			1	2	6	1	10
Red 329-	6.91	101	101	202	606	101	1010
Total 329-	6.91	101	101	202	606	101	1010

82.99%

=====

Marker number.....: 2
Spread length.....:2-34.10

Repeat [1/1]

Colors	YD	Plies	S1	Mediu	xlarg	38X42	40X40	Total
Sizes Marked.....			3	2	2	1	1	9
Red 67-	26.36	23	69	46	46	23	23	207
Total 67-	26.36	23	69	46	46	23	23	207

17.01%

=====

MaterialConsumption

Colors	Length	Rest
Red	391-16.87	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.55	Marker width.....IN	58.00	Marker length.....YD	3- 6.84
S1 Mediu large 38X42		Sizes			
-----		-----			
1	2	6	1	10	
Marking cost.....\$	12.05	Total Plies.....	101	Spread length.....YD	3- 9.34
Spreading cost.....\$	12.26	# bundles.....	85	Fabric used.....YD	325-7
Cutting cost/Hour.....\$	14.02	Units/plie yield.....	1	Spreading time.....Hours	0.49
Bundling cost.....\$	8.50	Plies/bundle.....	12	Cutting time.....Hours	0.47
Fabric cost.....\$	3291.92	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.96
-----		-----			
total cost.....\$	3338.75	total units.....	1010	Avg Cost/unit.....\$	3.31
=====		=====			

Marker number.....	2	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.49	Marker width.....IN	58.00	Marker length.....YD	2-31.60
S1 Mediu xlarge 38X42 40X40		Sizes			
-----		-----			
3	2	2	1	1	9
Marking cost.....\$	10.97	Total Plies.....	23	Spread length.....YD	2-34.10
Spreading cost.....\$	6.59	# bundles.....	18	Fabric used.....YD	67-28
Cutting cost/Hour.....\$	12.85	Units/plie yield.....	1	Spreading time.....Hours	0.26
Bundling cost.....\$	1.80	Plies/bundle.....	12	Cutting time.....Hours	0.43
Fabric cost.....\$	677.85	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.69
-----		-----			
total cost.....\$	710.12	total units.....	207	Avg Cost/unit.....\$	3.43
=====		=====			

Total cost summary. :

Total Markers.....	2	AVG eff.....	85.54	Marker width.....	58.00
Marking cost.....\$	23.02	Number of stacks.....	2	Sizes Marked.....	19
Spreading cost.....\$	18.84	# bundles.....	103	Total Plies.....	124
Cutting cost/Hour.....\$	26.92	Units/plie yield.....	1	Spreading time.....Hours	0.75
Bundling cost.....\$	10.30	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	0.90
Fabric cost.....\$	3969.80	Fabric used.....YD	396.98	total time.....Hours	1.65
-----		-----			
total cost.....\$	4048.88	total units.....	1217	Avg Cost/unit.....\$	3.33

Graphics Output..... :

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [1/1], 1

Colors Plies Spread length.....

Fed	101	329- 6.9YD
Total	101	

<----- 3- 9.34 ----->

	Sizes	S1Mediularge38X42	Total
Sizes Marked.....	1 2 6 1	10	

Total Units	101 202 606 101	1010	

Graphics Output..... :

Marker number.....: 2 ()
SpreadMethod Zig/Zag

Repeat [1/1], 2

Colors Plies Spread length.....

Red	23	67-28.4YD
Total	23	

<----- 2-34.10 ----->

	Sizes	S1Mediuxlarg38X4240X40	Total
Sizes Marked.....	3 2 2 1 1	9	

Total Units	69 46 46 23 23	207	

Marker Solution II.. :

Marker	Sizes							Sizes/Marker	Repeat	Plies	Units
	S1 Mediu large xlarg 38X42 40X40										
	1	1	2	6	-	1	-	10	1	101	1010
	1	3	2	-	2	1	1	9	1	23	207
2								19	2	124	1217

Optimized Spreading :

78 Red

23 Red

< 3- 6.84

> 2-31.60

>

Colors	Plies	StartPoint:	EndPoint:	Spread length.....
Red	23	0- 0.00	6- 3.69	141- 5.58
Red	78	0- 0.00	3- 8.09	254- 8.19

Parameters: BDEGHI

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	8.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6
1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units
Red	200	200	200	200	200	200	1200
Total	200	200	200	200	200	200	1200

Marker solution..... :

Marker	1	Repeat	2	Plies	210	Units	1260
S1	S2	S3	S4	S5	S6	Sizes	
1	1	1	1	1	1	6	

Marker	Sizes	Repeat	Plies	Units	
Total	1	6	2	210	1260

Setting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	2	210	423-2.00
Total	2	210	423-2.00

Local Flies.....

Marker	Repeat	Flies	YD
1	2	210	423-2.00
Total	2	210	423-2.00

Cutting Order Report :

Colors	Marker Repeat	1 [2]	Total [2]
=====		=====	=====
ed		210	210
=====		=====	=====
Total		210	210

Deviation report,... :

Colors	S1	S2	S3	S4	S5	S6	Units
Red	10	10	10	10	10	10	60
Total	10	10	10	10	10	10	60

Units produced

Colors	S1	S2	S3	S4	S5	S6	Units
Red	210	210	210	210	210	210	1260
Total	210	210	210	210	210	210	1260

Univ solution..... :

Marker number.....: 1
Spread length.....:2-0.52

Repeat [1/2]

Colors	YD	Plies	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....			1	1	1	1	1	1	6
Red 217-20.57		108	108	108	108	108	108	108	648
Total 217-20.57		108	108	108	108	108	108	108	648

51.43%

=====

Marker number.....: 1
pread length.....:2-0.52

Repeat [2/2]

Colors	YD	Plies	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....			1	1	1	1	1	1	6
Red 205-17.43		102	102	102	102	102	102	102	612
Total 205-17.43		102	102	102	102	102	102	102	612

48.57%

=====

MaterialConsumption

Colors	Length	Rest
Red	413-26.00	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	2	Marker Origin.....	New
St Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-34.02
S1 S2 S3 S4 S5 S6	Sizes				
1 1 1 1 1 1	6				
Marking cost.....\$	7.55	Total Plies.....	210	Spread length.....YD	2- 0.52
Spreading cost.....\$	5.65	# bundles.....	106	Fabric used.....YD	423-2
Cutting cost/Hour.....\$	19.02	Units/plie yield.....	1	Spreading time.....Hours	0.71
Handling cost.....\$	10.60	Plies/bundle.....	12	Cutting time.....Hours	0.63
Fabric cost.....\$	211.53	Fabric cost.....\$ /YD	0.50	total time.....Hours	1.34
total cost.....\$	254.35	total units.....	1260	Avg Cost/unit.....\$	0.20

total cost summary.. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	2	Sizes Marked.....	6
Spreading cost.....\$	5.65	# bundles.....	106	Total Plies.....	210
Cutting cost/Hour.....\$	19.02	Units/plie yield.....	1	Spreading time.....Hours	0.71
Handling cost.....\$	10.60	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	0.63
Fabric cost.....\$	211.53	Fabric used.....YD	423.06	total time.....Hours	1.34
total cost.....\$	254.35	total units.....	1260	Avg Cost/unit.....\$	0.20

Graphics Output..... :

Marker number.....: 1 () Repeat [1/2], 1
 readMethod Zig/Zag

Colors Plies Spread length.....

Red	108	217-20.6YD
Total	108	

< 2- 0.52 >

Sizes	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....	1	1	1	1	1	1	6
Total	Units	108	108	108	108	108	648

Graphics Output..... :

Marker number.....: 1 ()

Repeat [2/2], 2

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	102	205-17.4YD
Total	102	

← 0.52 →

Sizes	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....	1	1	1	1	1	1	6

Total	Units	102	102	102	102	102	102	612

Marker Solution 11.. :

Marker	Sizes						Sizes/Marker	Repeat	Plies	Units
	S1	S2	S3	S4	S5	S6				
1	1	1	1	1	1	1	6	2	210	1260
1							6	2	210	1260

Optimized Spreading :

6 Red

102 Red

< 1-34.02

>< 1-34.02

>

Colors	Plies	StartPoint:	EndPoint:Spread length.....
Red	102	0- 0.00	3-33.30 403-31.86
Red	6	0- 0.00	1-35.27 12- 3.14

Parameters: AEGHJK

Problem parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	8.00
Max sizes / marker.....	.10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1

1.00

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1
1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	Sl	Units
Red	48	48
Total	48	48

Marker solution..... :

Marker	1	Repeat	1	Plies	9	Units	54
Sl	Sizes						
6	6						

Marker	Sizes	Repeat	Plies	Units	
Total	1	6	1	9	54

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	1	9	17-10.70
Total	1	9	17-10.70

Total Plies.....

Marker	Repeat	Plies	YD
1	1	9	17-10.70
Total	1	9	17-10.70

Cutting Order Report :

Marker	1	Total
Colors Repeat	[1]	[1]
Red	9	9
Total	9	9

viation report.... :

Colors	Sl	Units
Red	6	6
Total	6	6

Units produced

Colors	Sl	Units
Red	54	54
Total	54	54

Unit solution..... :

Marker number.....: 1
Spread length.....:1-33.19

Repeat [1/1]

Colors	YD	Plies	Sl	Total
Sizes Marked.....			6	6

Red	17-10.70	9	54	54

Total	17-10.70	9	54	54

100.00%

=====

Material Consumption

Colors	Length	Rest

Red	16-32.30	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-30.69
SI	Sizes				
6	6				
Marking cost.....\$	7.55	Total Plies.....	9	Spread length.....YD	1-33.19
Spreading cost.....\$	1.77	# bundles.....	5	Fabric used.....YD	17-11
Cutting cost/Hour.....\$	9.18	Units/plie yield.....	1	Spreading time.....Hours	0.22
Bundling cost.....\$	0.50	Plies/bundle.....	12	Cutting time.....Hours	0.31
Fabric cost.....\$	172.97	Fabric cost.....\$ /YD	10.00	total time.....Hours	0.53
total cost.....\$	191.97	total units.....	54	Avg Cost/unit.....\$	3.55

Marker cost summary. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	1	Sizes Marked.....	6
Spreading cost.....\$	1.77	# bundles.....	5	Total Plies.....	9
Cutting cost/Hour.....\$	9.18	Units/plie yield.....	1	Spreading time.....Hours	0.22
Bundling cost.....\$	0.50	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	0.31
Fabric cost.....\$	172.97	Fabric used.....YD	17.30	total time.....Hours	0.53
total cost.....\$	191.97	total units.....	54	Avg Cost/unit.....\$	3.55

Graphics Output..... :

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [1/1], 1

Colors Plies Spread length.....

Red	9	17-10.7YD
Total	9	

<----- 1-33.19 ----->

Sizes	S1	Total
Sizes Marked.....	6	6

Total	Units	54

Marker Solution II.. :

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				

1	6	6	1	9	54

1		6	1	9	54

Optimized Spreading :

9	Red

< 1-30.69

Colors	Plies	StartPoint:	EndPoint:	Spread length.....
Red	9	0- 0.00	1-31.94	17-10.70

Parameters: ABFHIK

Problem parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SFR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1
1.00

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1
1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	95.4	85.4	85.5	85.5

put table..... :

Colors	Sl	Units
Red	1200	1200
Total	1200	1200

Marker solution..... :

Marker	1	Repeat	5	Plies	201	Units	1206
Sl	Sizes						
6	6						

Marker	Sizes	Repeat	Plies	Units	
Total	1	6	5	201	1206

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	5	201	386-11.06
Total	5	201	386-11.06

Total Plies.....

Marker	Repeat	Plies	YD
1	5	201	386-11.06
Total	5	201	386-11.06

Cutting Order Report :

Marker	1	Total
Colors Repeat	[5]	[5]
Red	201	201
Total	201	201

Deviation report.... :

Colors	SI	Units
Red	6	6
Total	6	6

Units produced

Colors	SI	Units
Red	1206	1206
Total	1206	1206

Unit solution..... :

Marker number.....: 1
Spread length.....:1-33.19

Repeat [1/5]

Colors	YD	Plies	Sl	Total
Sizes Marked.....			6	6

Red	90-11.90	47	282	282

Total	90-11.90	47	282	282

23.38%

=====

Marker number.....: 1
Spread length.....:1-33.19

Repeat [2/5]

Colors	YD	Plies	Sl	Total
Sizes Marked.....			6	6

Red	90-11.90	47	282	282

Total	90-11.90	47	282	282

23.38%

=====

Marker number.....: 1
Spread length.....:1-33.19

Repeat [3/5]

Colors	YD	Plies	Sl	Total
Sizes Marked.....			6	6

Red	90-11.90	47	282	282

Total	90-11.90	47	282	282

23.38%

=====

Marker number.....: 1
pread length.....:1-33.19

Repeat [4/5]

Colors	YD	Plies	SI	Total
Sizes Marked.....		6		6

Red	90-11.90	47	282	282

Total	90-11.90	47	282	282

23.38X
=====

Marker number.....: 1
Spread length.....:1-33.19

Repeat [5/5]

Colors	YD	Plies	SI	Total
Sizes Marked.....		6		6

Red	24-35.46	13	78	78

Total	24-35.46	13	78	78

6.47X
=====

Material Consumption

Colors	Length	Rest

Red	377-13.46	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	5	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-30.69
SI	Sizes				
---	---				
6	6				
Marking cost.....\$	7.55	Total Plies.....	201	Spread length.....YD	1-33.19
Spreading cost.....\$	16.95	# bundles.....	101	Fabric used.....YD	386-11
Cutting cost/Hour.....\$	15.29	Units/plie yield.....	1	Spreading time.....Hours	.068
Bundling cost.....\$	10.10	Plies/bundle.....	12	Cutting time.....Hours	1.53
Fabric cost.....\$	3863.07	Fabric cost.....\$ /YD	10.00	total time.....Hours	2.21

total cost.....\$	3912.97	total units.....	1206	Avg Cost/unit.....\$	3.24
=====					

Total cost summary. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	5	Sizes Marked.....	6
Spreading cost.....\$	16.95	# bundles.....	101	Total Plies.....	201
Cutting cost/Hour.....\$	15.29	Units/plie yield.....	1	Spreading time.....Hours	0.68
Bundling cost.....\$	10.10	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	1.53
Fabric cost.....\$	3863.07	Fabric used.....YD	386.31	total time.....Hours	2.21

total cost.....\$	3912.97	total units.....	1206	Avg Cost/unit.....\$	3.24

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [1/5], 1

Colors Plies Spread length.....

Red	47	90-11.9YD
Total	47	

<----- 1-33.19 ----->

Sizes	Sl	Total
Sizes Marked.....	6	6

Total	Units	282

282

Graphics Output..... :

Marker number.....: 1 ()
SpreadMethod Zig/Zag

Repeat [2/5], 2

Colors Plies Spread length.....

Red	47	90-11.9YD
Total	47	

<----- 1-33.19 ----->

Sizes	Sl	Total
Sizes Marked.....	6	6

Total	Units	282

282

Graphics Output..... :

Marker number.....: 1 ()

Repeat [3/5], 3

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	47	90-11.9YD
Total	47	

<----- 1-33.19 ----->

	Sizes	SI	Total
Sizes Marked.....	6	6	
Total	Units	282	282

Graphics Output..... :

Marker number.....: 1 ()

Repeat [4/5], 4

readMethod Zig/Zag

Colors Plies Spread length.....

Red	47	90-11.9YD
Total	47	

<----- 1-33.19 ----->

	Sizes	SI	Total
Sizes Marked.....	6	6	
Total	Units	282	282

Graphics Output..... :

Marker number.....: 1 ()

Repeat [5/5], 5

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	13	24-35.5YD
Total	13	

<----- 1-33.19 ----->

Sizes S1 Total
Sizes Marked..... 6 6

Total Units 78 78

Marker Solution II.. :

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				
1	6	6	5	201	1206
1		6	5	201	1206

Optimized Spreading : _____

34	Red
13	Red

< 1-30.69 >< 1-30.69 >< 1-30.69 >< 1-30.69 >< 1-30.69 >

Colors	Flies	StartPoint:	EndPoint:Spread length.....
Red	13	0- 0.00	9-10.70 121-11.31
Red	34	0- 0.00	7-16.01 254-10.75

Parameters: CEFHJJ

Problem parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	10	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	10	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1

1.00

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1

1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10

74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

=====		
Colors	SI	Units
=====		
Red	48	48

Total	48	48

Marker solution..... :

Marker	1	Repeat	1	Plies	29	Units	58
SI	Sizes						

2	2						

Marker	Sizes	Repeat	Plies	Units

Total	1	2	1	29 58

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	1	29	22-12.86
Total	1	29	22-12.86

Total Plies.....

Marker	Repeat	Plies	YD
1	1	29	22-12.86
Total	1	29	22-12.86

Cutting Order Report :

Colors	Marker	1	Total
	Repeat	[1]	[1]
Fed		29	29
Total		29	29

ev..ation report.... :

Colors	SI	Units
Red	10	10
Total	10	10

Units produced

Colors	SI	Units
Red	58	58
Total	58	58

Unit solution..... :

Marker number.....: 1

Repeat [1/1]

Spread length.....:0-27.75

Colors	YD	Flies	Sl	Total
Sizes Marked.....			2	2

Red	22-12.86	29	58	58

Total	22-12.86	29	58	58

100.00%

=====

Material Consumption

Colors	Length	Rest

Red	21- 2.46	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	75.10	Marker width.....IN	58.00	Marker length.....YD	0-25.25
SI	Sizes				
2	2				
Marking cost.....\$	3.25	Total Plies.....	29	Spread length.....YD	0-27.75
Spreading cost.....\$	6.42	# bundles.....	5	Fabric used.....YD	22-13
Cutting cost/Hour.....\$	4.72	Units/plie yield.....	1	Spreading time.....Hours	0.26
Bundling cost.....\$	0.50	Plies/bundle.....	12	Cutting time.....Hours	0.16
Fabric cost.....\$	11.18	Fabric cost.....\$ /YD	0.50	total time.....Hours	0.41
total cost.....\$	26.07	total units.....	58	Avg Cost/unit.....\$	0.45

Total cost summary.. :

Total Markers.....	1	AVS eff.....	75.10	Marker width.....	58.00
Marking cost.....\$	3.25	Number of stacks.....	1	Sizes Marked.....	2
Spreading cost.....\$	6.42	# bundles.....	5	Total Plies.....	29
Cutting cost/Hour.....\$	4.72	Units/plie yield.....	1	Spreading time.....Hours	0.26
Bundling cost.....\$	0.50	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	0.16
Fabric cost.....\$	11.18	Fabric used.....YD	22.36	total time.....Hours	0.41
total cost.....\$	26.07	total units.....	58	Avg Cost/unit.....\$	0.45

Graphics Output..... :

Marker number.....: 1 ()

Repeat [1/1], 1

readMethod Zig/Zag

Colors Plies Spread length.....

Red	29	22-12.9YD
Total	29	

<----- 0-27.75 ----->

Sizes S1 Total
izes Marked..... 2 2

Total Units 58 58

Marker Solution 11.. :

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				
1	2	2	1	29	58
1		2	1	29	58

Optimized Spreading :

29	Red

< 0-25.25

>

Colors	Plies	StartPoint:	EndPoint:Spread length.....
Red	29	0- 0.00	0-26.50 22-12.86

Parameters: BCEFGK

Machine parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	30.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1

1.00

Problee parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1

1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10

74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	SI	Units

Red	1200	1200

Total	1200	1200

Marker solution..... :

Marker	1	Repeat	2	Plies	200	Units	1200
SI	Sizes						

6	6						

Marker	Sizes	Repeat	Plies	Units

Total	1	6	2	200 1200

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	2	200	384-13.87
Total	2	200	384-13.87

Total Plies.....

Marker	Repeat	Plies	YD
1	2	200	384-13.87
Total	2	200	384-13.87

Cutting Order Report :

Marker	1	Total
Colors Repeat	[2]	[2]
Red	200	200
Total	200	200

Evolution report.... :

Colors	SI	Units
Red	0	0
Total	0	0

Units produced

Colors	SI	Units
Red	1200	1200
Total	1200	1200

Unit solution..... :

Marker number.....: 1
Spread length.....:1-33.19

Repeat [1/2]

Colors	YD	Plies	SI	Total
Sizes Marked.....		6		6
Red 207-20.45		108	648	648
Total 207-20.45		108	648	648

54.00%

=====

Marker number.....: 1
Spread length.....:1-33.19

Repeat [2/2]

Colors	YD	Plies	SI	Total
Sizes Marked.....		6		6
Red 176-29.42		92	552	552
Total 176-29.42		92	552	552

46.00%

=====

Material Consumption

Colors	Length	Rest
Red	375-17.67	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	2	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-30.69
SI	Sizes				
6	6				
Marking cost.....\$	7.55	Total Plies.....	200	Spread length.....YD	1-33.19
Reading cost.....\$	16.89	# bundles.....	101	Fabric used.....YD	384-14
Cutting cost/Hour.....\$	18.35	Units/plie yield.....	1	Spreading time.....Hours	0.58
Bundling cost.....\$	10.10	Plies/bundle.....	12	Cutting time.....Hours	0.61
Fabric cost.....\$	192.19	Fabric cost.....\$ /YD	0.50	total time.....Hours	1.29
total cost.....\$	245.09	total units.....	1200	Avg Cost/unit.....\$	0.20

Total cost summary. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	2	Sizes marked.....	6
Spreading cost.....\$	16.89	# bundles.....	101	Total Plies.....	200
Cutting cost/Hour.....\$	18.35	Units/plie yield.....	1	Spreading time.....Hours	0.65
Bundling cost.....\$	10.10	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	0.61
Fabric cost.....\$	192.19	Fabric used.....YD	384.39	total time.....Hours	1.29
total cost.....\$	245.09	total units.....	1200	Avg Cost/unit.....\$	0.20

Graphics Output..... :

Marker number.....: 1 ()
readMethod Zig/Zag

Repeat [1/2], 1

Colors Plies Spread length.....

Red	108	207-20.4YD
Total	108	

<----- 1-33.19 ----->

Sizes SI Total

Sizes Marked..... 6 6

Total Units 648 648

Graphics Output..... :

Marker number.....: 1 ()

Repeat [2/2], 2

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	92	176-29.4YD
Total	92	

<----- 1-33.19 ----->

Sizes S1 Total

Sizes Marked..... 6 6

Total Units 552 552

Marker Solution 11.. :

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				
1	6	6	2	200	1200
1		6	2	200	1200

Optimized Spreading :

16 Red

92 Red

< 1-30.69

>< 1-30.69

>

Colors	Plies	StartPoint:	EndPoint:	Spread length.....
Red	92	0- 0.00	3-26.63	347- 8.84
Red	16	0- 0.00	1-31.94	30-27.03

Parameters: LOW

able parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	8.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	47	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1
1.00

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1
1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	SI	Units
Red	46	48
Total	46	48

Marker solution..... :

Marker	1	Repeat	2	Plies	48	Unit,	48
SI	Sizes						
1	1						

Marker	Sizes	Repeat	Plies	Units
Total	1	1	2	48

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	2	48	20-14.68
Total	2	48	20-14.68

Total Plies.....

Marker	Repeat	Plies	YD
1	2	48	20-14.68
Total	2	48	20-14.68

Cutting Order Report :

Marker	1	Total
Colors Repeat	[2] [2]	
Red	48	48
Total	48	48

Deviation report.... :

Colors	St	Units
Red	0	0
Total	0	0

Units produced

Colors	St	Units
Red	48	48
Total	48	48

Unit solution..... :

Marker number.....: 1

Repeat [1/2]

Spread length.....:0-15.31

Colors	YD	Plies	SI	Total
Sizes Marked.....			1	1
Red	19-35.39	47	47	47
Total	19-35.38	47	47	47

97.92%

=====

Marker number.....: 1

Repeat [2/2]

Spread length.....:0-15.31

Colors	YD	Plies	SI	Total
Sizes Marked.....			1	1
Red	0-15.31	1	1	1
Total	0-15.31	1	1	1

2.08%

=====

Material Consumption

Colors	Length	Rest
Red	18- 9.86	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	2	Marker Origin.....	New
Est Eff.....	74.05	Marker width.....IN	58.00	Marker length.....YD	0-12.81
SI	Sizes				
1	1				
Marking cost.....\$	2.25	Total Plies.....	48	Spread length.....YD	0-15.31
Spreading cost.....\$	2.30	# bundles.....	4	Fabric used.....YD	20-15
Cutting cost/Hour.....\$	2.41	Units/plie yield.....	1	Spreading time.....Hours	0.29
Bundling cost.....\$	0.40	Plies/bundle.....	12	Cutting time.....Hours	0.24
Fabric cost.....\$	10.20	Fabric cost.....\$ /YD	0.50	total time.....Hours	0.53
total cost.....\$	17.56	total units.....	48	Avg Cost/unit.....\$	0.37

Total cost summary.. :

Total Markers.....	1	AVG eff.....	74.05	Marker width.....	58.00
Marking cost.....\$	2.25	Number of stacks.....	2	Sizes Marked.....	1
Spreading cost.....\$	2.30	# bundles.....	4	Total Plies.....	48
Setting cost/Hour.....\$	2.41	Units/plie yield.....	1	Spreading time.....Hours	0.29
Bundling cost.....\$	0.40	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	0.24
Fabric cost.....\$	10.20	Fabric used.....YD	20.41	total time.....Hours	0.53
total cost.....\$	17.56	total units.....	48	Avg Cost/unit.....\$	0.37

Graphics Output..... :

Marker number.....: 1 ()

Repeat [1/2], 1

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	47	19-35.4YD
Total	47	

< 0-15.31 >

Sizes S1 Total
 Sizes Marked..... 1 1

Total Units 47 47

Graphics Output..... :

Marker number.....: 1 ()

Repeat [2/2], 2

SpreadMethod Zig/Zag

Colors Plies Spread length.....

Red	1	0-15.3YD
Total	1	

< 0-15.31 >

Sizes S1 Total
 Sizes Marked..... 1 1

Marker Solution II.. : _____

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				

1	1	1	2	48	48

1		1	2	48	48

Optimized Spreading : _____

46 Red

1 Red

< 0-12.81 >> 0-12.81 >

Colors	Plies	StartPoint:	EndPoint:Spread length.....
Red	1	0- 0.00	0-26.86 0-28.11
Red	46	0- 0.00	0-14.06 19-20.07

Parameters: ABCGIJ

Problem parameters.. :

Number of sizes.....	1	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HP	8.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead.....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	10.00

Mark size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1
1.00

problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1

1.00

Material Util scale

1	2	3	4	5	6	7	8	9	10
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	Units
-----	-----	-----
Red	1200	1200
-----	-----	-----
Total	1200	1200

Marker solution..... :

Marker	1	Repeat	2	Plies	200	Units	1200
S1	Sizes						
-----	-----						

6	6

Marker	Sizes	Repeat	Plies	Units
-----	-----	-----	-----	-----
Total	1	6	2	200 1200

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD

1	2	200	384-13.87

Total	2	200	384-13.87

total Plies.....

Marker	Repeat	Plies	YD

1	2	200	384-13.87

Total	2	200	384-13.87

Cutting Order Report :

Colors	Marker	Repeat	1	Total

Red			200	200

Total			200	200

Deviation report.... :

Colors	\$1	Units

Red	0	0

Total	0	0

Units produced

Colors	\$1	Units

Red	1200	1200

Total	1200	1200

Unit solution..... :

Marker number.....: 1
Spread length.....: 1-33.19

Repeat [1/2]

Colors	YD	Plies	SI	Total

Sizes Marked.....		6	6	

Red 207-20.45	108	648	648	

Total 207-20.45	108	648	648	

54.00%

=====

Marker number.....: 1
Spread length.....: 1-33.19

Repeat [2/2]

Colors	YD	Plies	SI	Total

Sizes Marked.....		6	6	

Red 176-29.42	92	552	552	

Total 176-29.42	92	552	552	

46.00%

=====

Material Consumption

Colors	Length	Rest

Red	375-17.87	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	2	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-30.69
SI	Sizes				
6	6				
Marking cost.....\$	7.55	Total Plies.....	200	Spread length.....YD	1-33.19
Spreading cost.....\$	5.41	# bundles.....	101	Fabric used.....YD	384-14
Cutting cost/Hour.....\$	6.12	Units/plie yield.....	1	Spreading time.....Hours	0.68
Bundling cost.....\$	10.10	Plies/bundle.....	12	Cutting time.....Hours	0.61
Fabric cost.....\$	3843.85	Fabric cost.....\$ /YD	10.00	total time.....Hours	1.29
total cost.....\$	3673.02	total units.....	1200	Avg Cost/unit.....\$	3.23

Total cost summary.. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	2	Sizes Marked.....	6
Spreading cost.....\$	5.41	# bundles.....	101	Total Plies.....	200
Cutting cost/Hour.....\$	6.12	Units/plie yield.....	1	Spreading time.....Hours	0.68
Bundling cost.....\$	10.10	Fabric cost.....\$ /YD	10.00	Cutting time.....Hours	0.61
Fabric cost.....\$	3843.85	Fabric used.....YD	384.39	total time.....Hours	1.29
total cost.....\$	3673.02	total units.....	1200	Avg Cost/unit.....\$	3.23

Graphics Output..... :

Marker number.....: 1 ()
preadMethod Zig/Zag

Repeat [1/2], 1

Colors Plies Spread length.....

Red	108	207-20.4YD
Total	108	

<----- 1-33.19 ----->

Sizes SI Total
izes Marked..... 6 6

Total Units 648 648

Graphics Output..... :

Marker number.....: 1 ()
preadMethod Zig/Zag

Repeat [2/2], 2

Colors Plies Spread length.....

Red	92	176-29.4YD
Total	92	

<----- 1-33.19 ----->

Sizes SI Total
izes Marked..... 6 6

Total Units 552 552

Marker Solution 11.. :

Marker	Sizes	Sizes/Marker	Repeat	Plies	Units
	S1				
	6	6	2	200	1200
1		6	2	200	1200

Optimized Spreading : _____

16	Red
92	Red

< 1-30.69 >> 1-30.69 >

Colors	Files	StartPoint:	EndPoint:Spread length.....
Red	92	0- 0.00	3-26.63 347- 8.84
Red	16	0- 0.00	1-31.94 30-27.03

Parameters: DFGIJK

Problem parameters.. :

Number of sizes.....	6	Multiple plies.....	1	Spreading overhead...MIN/SPR	6.00
Number of colors.....	1	Max Ply-difference.....	100	Spreading cost.....\$ /HR	25.00
Max sizes / marker.....	10	Units/plie yield.....	1	Spread rate.....YD/MIN	45.00
Min sizes / marker.....	1	SpreadMethod	Zig/Zag	Turn Time.....SEC/END	6
Parity.....	Either	Plies/bundle.....	12	Roll change time.....SEC	360
Overcut %.....	0.00	Max. SpreadLength.....YD	100.00	Cutting overhead....MIN/CUT	5.00
Undercut %.....	0.00	Marker width.....IN	58.00	CuttingTime/Size.....SEC	267
Overcut units.....	0	Pattern area.....SQIN	550.00	Cutting cost/Hour.....\$ /HR	10.00
Undercut units.....	0	End Loss.....IN	2.50	Cost/Bundle.....\$	0.10
Maximum plies.....	108	Fixed Cost/Marker.....\$	1.25	Stack factor %.....	0.00
Minimum plies.....	1	Cost/Size Marker.....\$	1.00	Split factor %.....	0.00
				Fabric cost.....\$ /YD	0.50

Mark size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.02	1.03	1.05	1.05	1.06	1.07	1.08	1.08

Spreading size scale

1	2	3	4	5	6	7	8	9	10

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Cutting time scale

1	2	3	4	5	6

1.00	1.02	1.04	1.06	1.08	1.10

Problem parameters.. :

Common Line factor

1	2	3	4	5	6	7	8	9	10

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Patt siz arr scale

1	2	3	4	5	6

1.00	1.02	1.04	1.06	1.08	1.10

Material Util scale

1	2	3	4	5	6	7	8	9	10

74.0	75.1	75.2	75.2	75.3	85.3	85.4	85.4	85.5	85.5

Input table..... :

Colors	S1	S2	S3	S4	S5	S6	Units
=====							
Red	8	8	8	8	8	8	48

Total	8	8	8	8	8	8	48

Marker solution..... :

Marker	1	Repeat	1	Plies	8	Units	48
S1	S2	S3	S4	S5	S6	Sizes	

1	1	1	1	1	1	6	

Marker	Sizes	Repeat	Plies	Units

Total	1	6	8	48

Cutting Order Report :

Colors: Red

Marker	Repeat	Plies	YD
1	1	8	16-4.19
Total	1	8	16-4.19

tal Plies.....

Marker	Repeat	Plies	YD
1	1	8	16-4.19
Total	1	8	16-4.19

Cutting Order Report :

Marker	1	Total
Colors Repeat	[1]	[1]
	8	8
Total	8	8

Deviation report.... :

Colors	S1	S2	S3	S4	S5	S6	Units
Red	0	0	0	0	0	0	0
Total	0	0	0	0	0	0	0

Units produced

Colors	S1	S2	S3	S4	S5	S6	Units
Red	8	8	8	8	8	8	48
Total	8	8	8	8	8	8	48

Unit solution..... :

Marker number.....: 1
Spread length.....: 2-0.52

Repeat [1/1]

Colors	YD	Plies	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....			1	1	1	1	1	1	6
Red 16- 4.19		8	8	8	8	8	8	8	48
Total 16- 4.19		8	8	8	8	8	8	8	48

100.00%

=====

Material Consumption

Colors	Length	Rest
Red	15-27.39	

Marker cost summary. :

Marker number.....	1	Number of stacks.....	1	Marker Origin.....	New
Est Eff.....	85.32	Marker width.....IN	58.00	Marker length.....YD	1-34.02
S1 S2 S3 S4 S5 S6	Sizes				
1 1 1 1 1 1	6				
Marking cost.....\$	7.55	Total Plies.....	8	Spread length.....YD	2- 0.52
Spreading cost.....\$	5.48	# bundles.....	5	Fabric used.....YD	16-4
Cutting cost/Hour.....\$	3.17	Units/plie yield.....	1	Spreading time.....Hours	0.22
Bundling cost.....\$	0.50	Plies/bundle.....	12	Cutting time.....Hours	0.32
Fabric cost.....\$	8.06	Fabric cost.....\$ /YD	0.50	total time.....Hours	0.54
total cost.....\$	24.76	total units.....	48	Avg Cost/unit.....\$	0.52

Total cost summary.. :

Total Markers.....	1	AVG eff.....	85.32	Marker width.....	58.00
Marking cost.....\$	7.55	Number of stacks.....	1	Sizes Marked.....	6
Spreading cost.....\$	5.48	# bundles.....	5	Total Plies.....	8
Setting cost/Hour.....\$	3.17	Units/plie yield.....	1	Spreading time.....Hours	0.22
Handling cost.....\$	0.50	Fabric cost.....\$ /YD	0.50	Cutting time.....Hours	0.32
Fabric cost.....\$	8.06	Fabric used.....YD	16.12	total time.....Hours	0.54
Total cost.....\$	24.76	total units.....	48	Avg Cost/unit.....\$	0.52

Graphics Output..... :

Marker number.....: 1 ()
 SpreadMethod Zig/Zag

Repeat [1/1], 1

Colors Flies Spread length.....

Red	8	16- 4.2YD
Total	8	

<----- 2- 0.52 ----->

Sizes	S1	S2	S3	S4	S5	S6	Total
Sizes Marked.....	1	1	1	1	1	1	6
Total	Units	8	8	8	8	8	48

Marker Solution II.. :

Marker	Sizes						Sizes/Marker	Repeat	Plies	Units
	S1	S2	S3	S4	S5	S6				
1	1	1	1	1	1	1	6	1	8	48
1							6	1	8	48

Optimized Spreading : _____

8	Red

< 1-34.02

>

Colors	Plies	StartPoint:	EndPoint:	Spread length.....
Red	8.	0- 0.00	1-35.27	16- 4.19

Appendix C: Testbed Marker Lengths

Testbed Marker Lengths

Package B was used to estimate all marker lengths for the development of the testbed data described in Section 4.3. This package was selected because it was easy to force the package to produce a marker containing a particular combination of sizes and the associated fabric length for that size combination. All marker lengths from Package B were generated in one of two ways. First, the marker was requested directly as an order, or, if the solution did not yield a single marker, a request was made for a large order which divided into several markers, one being the desired marker. A set of fixed parameters were used so all markers would be under consistent constraints. One of the constraints was: Maximum number of sizes in a marker equal to six, this parameter guaranteed that an order which was divided into several markers had no more than six sizes in each marker.

The first attempt to generate the marker was done through the input table in Package B. The marker was requested as if it were an order. If this input did not produce the desired marker, a larger order was placed which would be divided into many markers. Using both of these ordering techniques, all lengths which the package could produce were generated.

To calculate the lengths of markers that Package B would not explicitly produce, each marker was calculated by using either an averaging heuristic or the fact that each order decrements the fabric length by a constant amount. Using these two possible calculations a length was determined for each of the following orders:

- (1) 4 of each size
- (2) 5 of each size
- (3) 3 of one size, 2 of another
- (4) 4 of one size, 1 of another
- (5) 2 of two sizes, 1 of another
- (6) 2 of two sizes.

It was determined that all markers, if ranked by increasing length, fell into certain numerical groupings. After calculating the approximate values in the above fashion they were compared to the numerical groupings. In most cases the calculations fell exactly into a numerical grouping. Where they did not the difference was at most .04 inches, in these cases the calculated lengths were altered to fall in the closest grouping.

Detailed calculations for each of the values follows.

1. Four of each size

Markers of four units in one size can be broken into 1-2-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
4/0/0/0/0/0		
0/4/0/0/0/0	1/2/1/0/0/0	53.94
0/0/4/0/0/0	0/1/2/1/0/0	54.95
0/0/0/4/0/0	0/0/1/2/1/0	55.96
0/0/0/0/4/0	0/0/0/1/2/1	56.97
0/0/0/0/0/4		

Since the marker lengths for four size 30 and four size 40 cannot be broken up they must be estimated using the knowledge that each order is separated by 1.01 inches. This gives the values:

<u>Original Marker</u>		<u>Length</u>
4/0/0/0/0/0	(53.94 - 1.01) =	52.93
0/0/0/0/0/4	(56.97 + 1.01) =	57.98

2. Five of each size

This marker can be broken into 1-3-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
5/0/0/0/0/0		
0/5/0/0/0/0	1/3/1/0/0/0	66.76
0/0/5/0/0/0	0/1/3/1/0/0	68.02
0/0/0/5/0/0	0/0/1/3/1/0	69.28
0/0/0/0/5/0	0/0/0/1/3/1	70.54
0/0/0/0/0/5		

The difference between each marker is 1.26 inches which results in the following values for the remaining markers.

<u>Original Marker</u>		<u>Length</u>
5/0/0/0/0/0	(66.76 - 1.26) =	65.50
0/0/0/0/0/5	(70.54 + 1.26) =	71.80

3. Three of one size, two of another

Because the three is on a end, it is impossible to separate, therefore, the two will be separated. The two can be represented as 1-0-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
3/2/0/0/0/0		
3/0/2/0/0/0	3/1/0/1/0/0	66.51
3/0/0/2/0/0	3/0/1/0/1/0	67.01

3/0/0/0/2/0	3/0/0/1/0/1	67.52
3/0/0/0/0/2		
2/0/0/0/0/3		
0/2/0/0/0/3	1/0/1/0/0/3	69.78
0/0/2/0/0/3	0/1/0/1/0/3	70.29
0/0/0/2/0/3	0/0/1/0/1/3	70.79
0/0/0/0/2/3		

The difference between each marker is approximately .51 inches which results in the following values for the remaining markers.

<u>Original Marker</u>		<u>Length</u>
3/2/0/0/0/0	(66.51-.51) =	66.00
3/0/0/0/0/2	(67.52+.51) =	68.03
2/0/0/0/0/3	(69.78-.51) =	69.27
0/0/0/0/2/3	(70.79+.51) =	71.30

It is now possible to break up the three into 1-1-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/3/0/0/0/0	3/1/1/0/0/0	66.26
0/3/2/0/0/0	1/1/3/0/0/0	67.26
0/3/0/2/0/0	1/1/1/2/0/0	67.77
0/3/0/0/2/0	1/1/1/0/2/0	68.27
0/3/0/0/0/2	1/1/1/0/0/2	68.78
2/0/3/0/0/0	2/1/1/1/0/0	67.01
0/2/3/0/0/0	0/3/1/1/0/0	67.52
0/0/3/2/0/0	0/1/1/3/0/0	68.52
0/0/3/0/2/0	0/1/1/1/2/0	69.03
0/0/3/0/0/2	0/1/1/1/0/2	69.53
2/0/0/3/0/0	2/0/1/1/1/0	67.77
0/2/0/3/0/0	0/2/1/1/1/0	68.27
0/0/2/3/0/0	0/0/3/1/1/0	68.78
0/0/0/3/2/0	0/0/1/1/3/0	69.78
0/0/0/3/0/2	0/0/1/1/1/2	70.29
2/0/0/0/3/0	2/0/0/1/1/1	68.52
0/2/0/0/3/0	0/2/0/1/1/1	69.03
0/0/2/0/3/0	0/0/2/1/1/1	69.53
0/0/0/2/3/0	0/0/0/3/1/1	70.03
0/0/0/0/3/2	0/0/0/1/1/3	71.04

4. Four of one size, one of another

In the following examples we will break the four into 1-2-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/4/0/0/0/0		
0/4/1/0/0/0		
0/4/0/1/0/0	1/2/1/1/0/0	67.26
0/4/0/0/1/0	1/2/1/0/1/0	67.52
0/4/0/0/0/1	1/2/1/0/0/1	67.77

The difference between markers in .25 inches.

<u>Original Marker</u>		<u>Length</u>
1/4/0/0/0/0	$(67.26 - .25) =$	67.01
0/4/1/0/0/0	$[67.26 - (3*.25)] =$	66.51

The following markers are impossible to reduce, therefore the following strategy will be employed: moving one size up or down increases or decreases the length of the marker by .25 inches. The marker 0/4/1/0/0/0 which is 66.51 inches long will be used as a base.

<u>Original Marker</u>		<u>Length</u>
4/1/0/0/0/0	$[66.51 - (3*.25)] =$	65.76
4/0/1/0/0/0	$[66.51 - (2*.25)] =$	66.01
4/0/0/1/0/0	$(66.51 - .25) =$	66.26
4/0/0/0/1/0		66.51
4/0/0/0/0/1	$(66.51 + .25) =$	66.76

Once again breaking up the four:

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/4/0/0/0	1/1/2/1/0/0	67.52
0/0/4/0/1/0	0/1/2/1/1/0	68.52
0/0/4/0/0/1	0/1/2/1/0/1	68.78
0/1/4/0/0/0		
0/0/4/1/0/0		

Again the difference between markers is 0.25 inches.

<u>Original Marker</u>		<u>Length</u>
0/1/4/0/0/0	$(67.52 + .25) =$	67.77
0/0/4/1/0/0	$(68.52 - .25) =$	68.27

Following the same patterns, the remainder of these markers are computed as follows.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/0/4/0/0	1/0/1/2/1/0	68.52
0/1/0/4/0/0	0/1/1/2/1/0	68.78
0/0/1/4/0/0		
0/0/0/4/1/0		
0/0/0/4/0/1	0/0/1/2/1/1	69.78

<u>Original Marker</u>		<u>Length</u>
0/0/1/4/0/0	$(68.78 + .25) =$	69.03
0/0/0/4/1/0	$(69.78 - .25) =$	69.53

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/0/0/4/0	1/0/0/1/2/1	69.53
0/1/0/0/4/0	0/1/0/1/2/1	69.78
0/0/1/0/4/0	0/0/1/1/2/1	70.04
0/0/0/1/4/0		
0/0/0/0/4/1		

<u>Original Marker</u>		<u>Length</u>
0/0/0/1/4/0	$(70.04 + .25) =$	70.29
0/0/0/0/4/1	$[70.04 + (3*.25)] =$	70.79

Calculated values for the remaining markers are based on the marker 0/0/0/0/4/1. Each of these markers was corrected to fall in the correct numerical group before the next marker was calculated. The calculations are:

<u>Original Marker</u>	<u>Calculated Length</u>	<u>Corrected Length</u>
1/0/0/0/0/4	$(70.79 - .25) = 70.54$	
0/1/0/0/0/4	70.79	70.75
0/0/1/0/0/4	$(70.75 + .25) = 71.00$	70.97
0/0/0/1/0/4	$(70.97 + .25) = 71.22$	71.19
0/0/0/0/1/4	$(71.19 + .25) = 71.44$	71.41

5. Two of two sizes, one of another

To compute these markers one of the "two of two sizes" was broken into 1-0-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/2/1/0/0/0		
2/2/0/1/0/0	3/0/1/1/0/0	66.76
2/2/0/0/1/0	3/0/1/0/1/0	67.01
2/2/0/0/0/1	3/0/1/0/0/1	67.26

Using the 0.25 decrement we calculate:

<u>Original Marker</u>		<u>Length</u>
2/2/1/0/0/0	$(66.76 - .25) =$	66.51

The remainder of two of two sizes and one of another has been calculated as follows.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/1/2/0/0/0		
2/0/2/1/0/0		
2/0/2/0/1/0	2/1/0/1/1/0	67.52
2/0/2/0/0/1	2/1/0/1/0/1	67.77

<u>Original Marker</u>		<u>Length</u>
2/1/2/0/0/0	$[67.52 - (3*.25)] =$	66.77
2/0/2/1/0/0	$(67.52 - .25) =$	67.27

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/1/0/2/0/0	2/1/1/0/1/0	67.26
2/0/1/2/0/0		
2/0/0/2/1/0		
2/0/0/2/0/1	2/0/1/0/1/1	68.27

<u>Original Marker</u>		<u>Length</u>
2/0/1/2/0/0	$[(68.27 - (3*.25))] =$	67.52
2/0/0/2/1/0	$(68.27 - .25) =$	68.02

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/1/0/0/2/0	2/1/0/1/0/1	67.77
2/0/1/0/2/0	2/0/1/1/0/1	68.02
2/0/0/1/2/0		
2/0/0/0/2/1		
2/1/0/0/0/2		
2/0/1/0/0/2		
2/0/0/1/0/2		
2/0/0/0/1/2		

<u>Original Marker</u>		<u>Length</u>
2/0/0/1/2/0	$(68.02 + .25) =$	68.27
2/0/0/0/2/1	$[68.02 + (3*.25)] =$	68.77
2/1/0/0/0/2	$[67.77 + (2*.25)] =$	68.27
2/0/1/0/0/2	$[68.02 + (2+.25)] =$	68.52
2/0/0/1/0/2	$68.52 + .25 =$	68.77
2/0/0/0/1/2	$[68.52 + (2*.25)] =$	69.02

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/2/2/0/0/0	2/1/1/1/0/0	67.01
0/2/2/1/0/0	1/0/3/1/0/0	67.77
0/2/2/0/1/0	1/0/3/0/1/0	68.02
0/2/2/0/0/1	1/0/3/0/0/1	68.27
1/2/0/2/0/0	1/2/1/0/1/0	67.52
0/2/1/2/0/0	1/0/3/0/1/0	68.02
0/2/0/2/1/0	1/0/1/2/1/0	68.52
0/2/0/2/0/1	1/0/1/2/0/1	68.78
1/2/0/0/2/0	1/2/0/1/0/1	68.02
0/2/1/0/2/0	0/2/1/1/0/1	68.52
0/2/0/1/2/0	1/0/1/1/2/0	68.78
0/2/0/0/2/1	1/0/1/0/2/1	69.28
1/2/0/0/0/2		
0/2/1/0/0/2		
0/2/0/1/0/2	1/0/1/1/0/2	69.28
0/2/0/0/1/2	1/0/1/0/1/2	69.53

<u>Original Marker</u>		<u>Length</u>
1/2/0/0/0/2	$[69.28 - (3 \times .25)] =$	68.53
0/2/1/0/0/2	$(69.28 - .25) =$	69.03

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/2/2/0/0	1/1/0/3/0/0	68.02
0/1/2/2/0/0		
0/0/2/2/1/0	0/1/0/3/1/0	69.03
0/0/2/2/0/1	0/1/0/3/0/1	69.28

<u>Original Marker</u>		<u>Length</u>
0/1/2/2/0/0	$(68.02 + .25) =$	68.27

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/2/0/2/0	1/1/0/1/2/0	68.52
0/1/2/0/2/0	0/1/2/1/0/1	68.78
0/0/2/1/2/0		
0/0/2/0/2/1		

<u>Original Marker</u>		<u>Length</u>
0/0/2/1/2/0	$[68.78 + (2 \times .25)] =$	69.28
0/0/2/0/2/1	$[68.78 + (4 \times .25)] =$	69.78

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/2/0/0/2	1/1/0/1/0/2	69.03
0/1/2/0/0/2		

0/0/2/1/0/2		
0/0/2/0/1/2	0/1/0/1/1/2	70.04

<u>Original Marker</u>		<u>Length</u>
0/1/2/0/0/2	$(69.03 + .25) =$	69.28
0/0/2/1/0/2	$[69.03 + (3*.25)] =$	69.78

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/0/2/2/0	1/0/1/0/3/0	69.03
0/1/0/2/2/0	0/1/1/0/3/0	69.28
0/0/1/2/2/0	0/0/1/3/0/1	69.53
0/0/0/2/2/1	0/0/1/0/3/1	70.29
1/0/0/2/0/2	1/0/1/0/1/2	69.53
0/1/0/2/0/2	0/1/1/0/1/2	69.78
0/0/1/2/0/2		
0/0/0/2/1/2		

<u>Original Marker</u>		<u>Length</u>
0/0/1/2/0/2	$(69.78 + .25) =$	70.03
0/0/0/2/1/2	$[69.78 + (2*.25)] =$	70.28

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
1/0/0/0/2/2	1/0/0/1/0/3	70.04
0/1/0/0/2/2	0/1/0/1/0/3	70.29
0/0/1/0/2/2	0/0/1/1/0/3	70.54
0/0/0/1/2/2		

<u>Original Marker</u>		<u>Length</u>
0/0/0/1/2/2	$(70.54 + .25) =$	70.79

6. Two of two sizes

This marker can be broken as before with two becoming 1-0-1.

<u>Original Marker</u>	<u>New Marker</u>	<u>Length</u>
2/2/0/0/0/0	3/0/1/0/0/0	53.44
2/0/2/0/0/0	2/1/0/1/0/0	53.94
2/0/0/2/0/0	2/0/1/0/1/0	54.45
2/0/0/0/2/0	2/0/0/1/0/1	54.95
2/0/0/0/0/2		
0/2/2/0/0/0	1/0/3/0/0/0	54.45
0/2/0/2/0/0	0/2/1/0/1/0	54.95
0/2/0/0/2/0	1/0/1/0/2/0	55.46
0/2/0/0/0/2	1/0/1/0/0/2	55.96
0/0/2/2/0/0	0/1/0/3/0/0	55.46
0/0/2/0/2/0	0/1/0/1/2/0	55.96

0/0/2/0/0/2	0/1/0/1/0/2	56.46
0/0/0/2/2/0	0/0/1/0/3/0	56.46
0/0/0/2/0/2	0/0/1/0/1/2	56.97
0/0/0/0/2/2	0/0/0/1/0/3	57.47

Original Marker

2/0/0/0/0/2

(54.95 + .51) =

Length

55.45

	A	B	C	D	E	F	G	H
1	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
2	1	1	1	1			54.45	
3	1	1	1		1		54.70	
4	1	1	1			1	54.95	
5	1	1	1	1	1		68.02	
6	1	1	1	1		1	68.27	
7	1	1	1	1	1	1	72.52	
8	1	1	1		1	1	68.52	
9	1	1	1		2		68.27	
10	1	1	1	2			67.77	
11	1	1	1		2		68.27	
12	1	1	1			2	68.78	
13	1	1	1	2	1		72.08	
14	1	1	1	2		1	72.30	
15	1	1	1	1	2		72.30	
16	1	1	1		2	1	72.75	
17	1	1	1	1		2	72.75	
18	1	1	1		1	2	72.97	
19	1	1	1	3			71.86	
20	1	1	1		3		72.52	
21	1	1	1			3	73.19	
22	1	1	2			1	68.02	
23	1	1	2	2			71.63	
24	1	1	2		2		72.08	
25	1	1	2			2	72.52	
26	1	1	2	1			67.52	
27	1	1	2		1		67.77	
28	1	1	2			1	68.02	
29	1	1	2	1	1		71.86	
30	1	1	2	1		1	72.08	
31	1	1	2		1	1	72.30	
32	1	1	2				54.19	
33	1	1	3				67.26	
34	1	1	3	1			71.41	
35	1	1	3		1		71.63	
36	1	1	3			1	71.86	
37	1	1	4				71.19	
38	1	1					28.01	
39	1	1		1			41.36	
40	1	1			1		41.62	
41	1	1				1	41.87	
42	1	1		1	1		54.95	
43	1	1		1		1	55.20	
44	1	1			1	1	55.46	
45	1	1		3			68.02	
46	1	1			3		68.78	
47	1	1				3	69.53	
48	1	1		4			72.08	

	A	B	C	D	E	F	G	H
49	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
50	1	1			4		72.97	
51	1	1				4	73.86	
52	1	1		1	1	1	68.78	
53	1	1			2		55.20	
54	1	1		1	2		68.52	
55	1	1		2	2		72.52	
56	1	1		2		2	72.97	
57	1	1			2	2	73.41	
58	1	1		2	1		68.27	
59	1	1		2		1	68.52	
60	1	1		1	2		68.52	
61	1	1			2	1	69.03	
62	1	1		1		2	69.03	
63	1	1			1	2	69.28	
64	1	1		2	1	1	72.75	
65	1	1		1	2	1	72.97	
66	1	1		1	1	2	73.19	
67	1	1		3	1		72.30	
68	1	1		3		1	72.52	
69	1	1		1	3		72.75	
70	1	1			3	1	73.19	
71	1	1		1		3	73.41	
72	1	1			1	3	73.64	
73	1	1		2			54.70	
74	1	1			2		55.20	
75	1	1				2	55.71	
76	1	2	1	2			71.41	
77	1	2	1		2		71.86	
78	1	2	1			2	72.30	
79	1	2	1	1			67.26	
80	1	2	1		1		67.52	
81	1	2	1			1	67.77	
82	1	2	1	1	1		71.63	
83	1	2	1	1		1	71.86	
84	1	2	1		1	1	72.08	
85	1	2	1				53.94	
86	1	2	2	1			71.19	
87	1	2	2		1		71.41	
88	1	2	2			1	71.63	
89	1	2	2				67.01 *	
90	1	2	3				70.97	
91	1	2					40.86	
92	1	2		3			71.63	
93	1	2			3		72.30	
94	1	2		1			54.19	
95	1	2		2	1		71.86	
96	1	2		2		1	72.08	

	A	B	C	D	E	F	G	H
97	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
98	1	2		1	2		72.08	
99	1	2			2	1	72.52	
100	1	2		1		2	72.52	
101	1	2			1	2	72.75	
102	1	2		1	1		67.77	
103	1	2		1		1	68.02	
104	1	2			1	1	68.27	
105	1	2		1	1	1	72.3	
106	1	2		1			54.19	
107	1	2			1		54.45	
108	1	2				1	54.7	
109	1	2				3	72.97	
110	1	2		2			67.52	*
111	1	2			2		68.03	*
112	1	2				2	68.52	*
113	1	3	1				66.76	
114	1	3	1	1			70.97	
115	1	3	1		1		71.19	
116	1	3	1			1	71.41	
117	1	3	2				70.75	
118	1	3		1			67.01	
119	1	3			1		67.26	
120	1	3				1	67.52	
121	1	3		2			71.19	
122	1	3			2		71.63	
123	1	3				2	72.08	
124	1	3		1	1		71.41	
125	1	3		1		1	71.63	
126	1	3			1	1	71.86	
127	1	3					53.69	
128	1	4	1				70.52	
129	1	4		1			70.75	
130	1	4			1		70.97	
131	1	4				1	71.19	
132	1	4					66.51	*
133	1	5					70.3	
134	1		1				28.26	
135	1		1	1			41.62	
136	1		1		1		41.87	
137	1		1			1	42.12	
138	1		1	3			68.27	
139	1		1	1	1	1	69.03	
140	1		1		3		69.03	
141	1		1			3	69.78	
142	1		1	4			72.3	
143	1		1		4		73.19	
144	1		1			4	74.08	

	A	B	C	D	E	F	G	H
145	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
146	1		1	1	1		55.2	
147	1		1	1		1	55.46	
148	1		1		1	1	55.71	
149	1		1		2		55.46	
150	1		1		2	1	69.28	
151	1		1	1		2	69.28	
152	1		1	1	1	2	73.41	
153	1		1	2	2		72.75	
154	1		1	2		2	73.19	
155	1		1		2	2	73.64	
156	1		1	2	1		68.52	
157	1		1	2		1	68.78	
158	1		1	1	2		68.78	
159	1		1		2	1	69.28	
160	1		1	1		2	69.28	
161	1		1		1	2	69.53	
162	1		1	2	1	1	72.97	
163	1		1	1	2	1	73.19	
164	1		1	1	1	2	73.41	
165	1		1	3	1		72.52	
166	1		1	3		1	72.75	
167	1		1	1	3		72.97	
168	1		1		3	1	73.41	
169	1		1	1		3	73.64	
170	1		1		1	3	73.86	
171	1		1	2			54.95	
172	1		1		2		55.46	
173	1		1			2	55.96	
174	1		2				41.36	
175	1		2	3			72.08	
176	1		2		3		72.75	
177	1		2			1	55.2	
178	1		2	2	1		72.3	
179	1		2	2		1	72.52	
180	1		2	1	2		72.52	
181	1		2		2	1	72.97	
182	1		2	1		2	72.97	
183	1		2		1	2	73.19	
184	1		2	1	1		68.27	
185	1		2	1		1	68.52	
186	1		2		1	1	68.78	
187	1		2	1	1	1	72.75	
188	1		2	1			54.7	
189	1		2		1		54.95	
190	1		2			1	55.2	
191	1		2			3	73.41	
192	1		2	2			68.02 *	

	A	B	C	D	E	F	G	H
193	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
194	1		2		2		68.52	*
195	1		2			2	69.03	*
196	1		3	1			67.77	
197	1		3		1		68.02	
198	1		3			1	68.27	
199	1		3	2			71.86	
200	1		3		2		72.30	
201	1		3			2	72.75	
202	1		3	1	1		72.08	
203	1		3	1		1	72.30	
204	1		3		1	1	72.52	
205	1		3				54.45	
206	1		4	1			71.63	
207	1		4		1		71.86	
208	1		4			1	72.08	
209	1		4				67.52	*
210	1		5				71.41	
211	1						15.31	
212	1			1			28.51	
213	1				1		28.76	
214	1					1	29.02	
215	1			2			41.87	
216	1			1	1		42.12	
217	1				2		42.37	
218	1			1		1	42.37	
219	1				1	1	42.63	
220	1					2	42.88	
221	1			3	1		68.78	
222	1			3		1	69.03	
223	1			1	3		69.28	
224	1				3	1	69.78	
225	1			1		3	70.04	
226	1				1	3	70.29	
227	1			4	1		72.75	
228	1			3	2		72.97	
229	1			4		1	72.97	
230	1			2	3		73.19	
231	1			3		2	73.41	
232	1			1	4		73.41	
233	1				4	1	73.86	
234	1				3	2	74.08	
235	1			1		4	74.30	
236	1				1	4	74.52	
237	1			1	1	1	55.96	
238	1			1	2	1	69.53	
239	1			2	2	1	73.41	
240	1			2	1	2	73.64	

	A	B	C	D	E	F	G	H
241	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
242	1			1	2	2	73.86	
243	1			2	1	1	69.28	
244	1			1	2	1	69.53	
245	1			1	1	2	69.78	
246	1			3	1	1	73.19	
247	1			1	3	1	73.64	
248	1			1	1	3	74.08	
249	1			2	1		55.46	
250	1			2		1	55.71	
251	1			1	2		55.71	
252	1				2	1	56.21	
253	1			1		2	56.21	
254	1				1	2	56.46	
255	1			2		3	73.86	
256	1				2	3	74.3	
257	1			5			72.52	
258	1				5		73.64	
259	1					5	74.75	
260	1			3			55.2	
261	1				3		55.96	
262	1					3	56.72	
263	1			4			68.52	*
264	1				4		69.53	*
265	1					4	70.52	*
266	1			2	2		69.03	*
267	1				2	2	70.04	*
268	1			2		2	69.53	*
269	2	1	1		1		67.26	
270	2	1	1		2		71.63	
271	2	1	1	2			71.19	
272	2	1	1		2		71.63	
273	2	1	1			2	72.08	
274	2	1	1	1			67.01	
275	2	1	1		1		67.26	
276	2	1	1			1	67.52	
277	2	1	1	1	1		71.41	
278	2	1	1	1		1	71.63	
279	2	1	1		1	1	71.86	
280	2	1	1				53.69	
281	2	1	2	1			70.97	
282	2	1	2		1		71.19	
283	2	1	2			1	71.41	
284	2	1	2				66.75	*
285	2	1	3				70.75	
286	2	1					40.61	
287	2	1		3			71.41	
288	2	1			3		72.08	

	A	B	C	D	E	F	G	H
289	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
290	2	1		1	2		71.86	
291	2	1		2	1		71.63	
292	2	1		2		1	71.86	
293	2	1		1	2		71.86	
294	2	1			2	1	72.3	
295	2	1		1		2	72.3	
296	2	1			1	2	72.52	
297	2	1		1	1		67.52	
298	2	1		1		1	67.77	
299	2	1			1	1	68.02	
300	2	1		1	1	1	72.08	
301	2	1		1			53.94	
302	2	1			1		54.19	
303	2	1				1	54.45	
304	2	1				3	72.75	
305	2	1		2			67.26 *	
306	2	1			2		67.77 *	
307	2	1				2	68.27 *	
308	2	2	1	1			70.75	
309	2	2	1		1		70.97	
310	2	2	1			1	71.19	
311	2	2	1				66.51 *	
312	2	2	2				70.52	
313	2	2		1	1		71.19	
314	2	2		1		1	71.41	
315	2	2			1	1	71.63	
316	2	2		2			70.97	
317	2	2			2		71.41	
318	2	2				2	71.86	
319	2	2		1			66.76 *	
320	2	2			1		67.01 *	
321	2	2				1	67.26 *	
322	2	2					53.45 *	
323	2	3	1				70.3	
324	2	3		1			70.52	
325	2	3			1		70.75	
326	2	3				1	70.97	
327	2	3					66.26 *	
328	2	4					70.08	
329	2		1				40.86	
330	2		1	3			71.63	
331	2		1		3		72.3	
332	2		1		2	1	72.52	
333	2		1		1		54.45	
334	2		1	2	1		71.86	
335	2		1	2		1	72.08	
336	2		1	1	2		72.08	

	A	B	C	D	E	F	G	H
337	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
338	2		1		2	1	72.52	
339	2		1	1		2	72.52	
340	2		1		1	2	72.75	
341	2		1	1	1		67.77	
342	2		1	1		1	68.02	
343	2		1		1	1	68.27	
344	2		1	1	1	1	72.3	
345	2		1	1			54.19	
346	2		1		1		54.45	
347	2		1			1	54.7	
348	2		1			3	72.97	
349	2		1	2			67.51	*
350	2		1		2		68.02	*
351	2		1			2	68.52	*
352	2		2	1	1		71.63	
353	2		2	1		1	71.86	
354	2		2		1	1	72.08	
355	2		2	2			71.41	
356	2		2		2		71.86	
357	2		2			2	72.3	
358	2		2	1			67.26	*
359	2		2		1		67.52	*
360	2		2			1	67.77	*
361	2		2				53.93	*
362	2		3	1			71.19	
363	2		3		1		71.41	
364	2		3			1	71.63	
365	2		3				67.01	*
366	2		4				70.97	
367	2			1			41.11	
368	2				1		41.36	
369	2					1	41.62	
370	2			3	1		72.08	
371	2			3		1	72.3	
372	2			1	3		72.52	
373	2				3	1	72.97	
374	2			2	1	1	72.52	
375	2			1	2	1	72.75	
376	2			1	1	2	72.97	
377	2			1	1	1	68.52	
378	2			1	1		54.7	
379	2			1		1	54.95	
380	2				1	1	55.2	
381	2			1		3	73.19	
382	2				1	3	73.41	
383	2						27.75	
384	2			2	2		72.3	

	A	B	C	D	E	F	G	H
385	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
386	2			2		2	72.75	
387	2				2	2	73.19	
388	2			4			71.86	
389	2				4		72.75	
390	2					4	73.64	
391	2			3			67.77 *	
392	2				3		68.52 *	
393	2					3	69.27 *	
394	2			2	1		68.02 *	
395	2			2		1	68.27 *	
396	2			1	2		68.27 *	
397	2				2	1	68.78 *	
398	2			1		2	68.78 *	
399	2				1	2	69.03 *	
400	2			2			54.45 *	
401	2				2		54.94 *	
402	2					2	55.46 *	
403	3	1	1				66.26	
404	3	1	1	1			70.52	
405	3	1	1		1		70.75	
406	3	1	1			1	70.97	
407	3	1	2				70.3	
408	3	1		1			66.51	
409	3	1			1		66.76	
410	3	1				1	67.01	
411	3	1		2			70.75	
412	3	1			2		71.19	
413	3	1				2	71.63	
414	3	1		1	1		70.97	
415	3	1		1		1	71.19	
416	3	1				1	71.41	
417	3	1					53.19	
418	3	2	1				70.08	
419	3	2		1			70.3	
420	3	2			1		70.52	
421	3	2				1	70.75	
422	3	2					65.77 *	
423	3	3					69.86	
424	3		1	1			66.76	
425	3		1		1		67.01	
426	3		1			1	67.26	
427	3		1	2			70.97	
428	3		1		2		71.41	
429	3		1			2	71.86	
430	3		1	1	1		71.19	
431	3		1	1		1	71.41	
432	3		1		1	1	71.63	

	A	B	C	D	E	F	G	H
433	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
434	3		1				53.44	
435	3		2	1			70.75	
436	3		2		1		70.97	
437	3		2			1	71.19	
438	3		2				66.01 *	
439	3		3				70.52	
440	3			1	1		67.26	
441	3			1		1	67.52	
442	3				1	1	69.77	
443	3			2	1		71.41	
444	3			2		1	71.63	
445	3			1	2		71.63	
446	3				2	1	72.08	
447	3			1		2	72.08	
448	3				1	2	72.3	
449	3			1	1	1	71.86	
450	3						40.36	
451	3			3			71.19	
452	3				3		71.86	
453	3					3	72.52	
454	3			1			53.69	
455	3				1		53.94	
456	3					1	54.19	
457	3			2			66.77 *	
458	3				2		67.26 *	
459	3					2	68.03 *	
460	4	1	1				69.26	
461	4	1	1				69.86	
462	4	1		1			70.08	
463	4	1			1		70.3	
464	4	1				1	70.52	
465	4	1					65.76 *	
466	4	2					69.63	
467	4		1	1			70.3	
468	4		1		1		70.52	
469	4		1			1	70.75	
470	4		1				66.01 *	
471	4		2				70.08	
472	4			1	1		70.75	
473	4			1		1	70.97	
474	4				1	1	71.19	
475	4			2			70.52	
476	4				2		70.97	
477	4					2	71.41	
478	4						52.93 *	
479	4			1			66.26 *	
480	4				1		66.51 *	

	A	B	C	D	E	F	G	H
481	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
482	4					1	66.76	*
483	5	1					69.41	
484	5		1				69.63	
485	5			1			69.86	
486	5				1		70.08	
487	5					1	70.3	
488	5						65.5	*
489	6						69.19	
490		1	1				28.51	
491		1	1	1			41.87	
492		1	1		1		42.12	
493		1	1			1	42.37	
494		1	1	1	1		55.46	
495		1	1	1		1	55.71	
496		1	1	3			68.52	
497		1	1	1	1	1	69.28	
498		1	1		3		69.28	
499		1	1			3	70.04	
500		1	1	4			72.52	
501		1	1		4		73.41	
502		1	1			4	74.3	
503		1	1		1	1	55.96	
504		1	1	2		1	69.03	
505		1	1	1		2	69.53	
506		1	1	2		2	73.41	
507		1	1	2	2		72.97	
508		1	1	2		2	73.41	
509		1	1		2	2	73.86	
510		1	1	2	1		68.78	
511		1	1	2		1	69.03	
512		1	1	1	2		69.03	
513		1	1		2	1	69.53	
514		1	1	1		2	69.53	
515		1	1		1	2	69.78	
516		1	1	2	1	1	73.19	
517		1	1	1	2	1	73.41	
518		1	1	1	1	2	73.64	
519		1	1	3	1		72.75	
520		1	1	3		1	72.97	
521		1	1	1	3		73.19	
522		1	1		3	1	73.64	
523		1	1	1		3	73.86	
524		1	1		1	3	74.08	
525		1	1	2			55.2	
526		1	1		2		55.71	
527		1	1			2	56.21	
528		1	2				41.62	

	A	B	C	D	E	F	G	H
529	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
530		1	2	3			72.3	
531		1	2		3		72.97	
532		1	2	1		2	73.19	
533		1	2	2	1		72.52	
534		1	2	2		1	72.75	
535		1	2	1	2		72.75	
536		1	2		2	1	73.19	
537		1	2	1		2	73.19	
538		1	2		1	2	73.41	
539		1	2	1	1		68.52	
540		1	2	1		1	68.78	
541		1	2		1	1	69.03	
542		1	2	1	1	1	72.97	
543		1	2	1			54.95	
544		1	2		1		55.2	
545		1	2			1	55.46	
546		1	2			3	73.64	
547		1	2	2			68.27 *	
548		1	2		2		68.78 *	
549		1	2			2	69.28 *	
550		1	3	1			68.02	
551		1	3		1		68.27	
552		1	3			1	68.52	
553		1	3	2			72.08	
554		1	3		2		72.52	
555		1	3			2	72.97	
556		1	3	1	1		72.3	
557		1	3	1		1	72.52	
558		1	3		1	1	72.75	
559		1	3				54.7	
560		1	4	1			71.86	
561		1	4		1		72.08	
562		1	4			1	72.3	
563		1	4				67.77 *	
564		1	5				71.63	
565		1					15.56	
566		1		1			28.76	
567		1			1		29.02	
568		1				1	29.27	
569		1		2			42.12	
570		1		1	1		42.37	
571		1			2		42.63	
572		1		1		1	42.63	
573		1				2	43.13	
574		1		1	1	1	56.21	
575		1		3	1		69.03	
576		1		3		1	69.28	

	A	B	C	D	E	F	G	H
577	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
578		1		1	3		69.53	
579		1			3	1	70.04	
580		1		1		3	70.29	
581		1			1	3	70.54	
582		1		4	1		72.97	
583		1		4		1	73.19	
584		1		3	2		73.19	
585		1		2	3		73.41	
586		1		1	4		73.64	
587		1		3		2	73.64	
588		1			4	1	74.08	
589		1			3	2	74.3	
590		1		1		4	74.52	
591		1			1	4	74.75	
592		1			1	1	42.88	
593		1		2		1	55.96	
594		1		2	2	1	73.64	
595		1		2	1	2	73.86	
596		1		1	2	2	74.08	
597		1		2	1	1	69.53	
598		1		1	2	1	69.78	
599		1		1	1	2	70.04	
600		1		3	1	1	73.41	
601		1		1	3	1	73.85	
602		1		1	1	3	74.3	
603		1		2	1		55.71	
604		1		2		1	55.96	
605		1		1	2		55.96	
606		1			2	1	56.46	
607		1		1		2	56.48	
608		1			1	2	56.72	
609		1		2		3	74.08	
610		1			2	3	74.52	
611		1		5			72.75	
612		1			5		73.86	
613		1				5	74.97	
614		1		3			55.46	
615		1			3		56.21	
616		1				3	56.97	
617		1		4			68.78 *	
618		1			4		69.78 *	
619		1				4	70.75 *	
620		1		2	2		69.28 *	
621		1			2	2	70.29 *	
622		1		2		2	69.78 *	
623		2	1				41.36	
624		2	1	3			72.08	

	A	B	C	D	E	F	G	H
625	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
626		2	1		3		72.75	
627		2	1	2		1	72.52	
628		2	1	1			54.70	
629		2	1	2	1		72.30	
630		2	1	2		1	72.52	
631		2	1	1	2		72.52	
632		2	1		2	1	72.97	
633		2	1	1		2	72.97	
634		2	1		1	2	73.19	
635		2	1	1	1		68.27	
636		2	1	1		1	68.52	
637		2	1		1	1	68.78	
638		2	1	1	1	1	72.75	
639		2	1	1			54.70	
640		2	1		1		54.95	
641		2	1			1	55.20	
642		2	1			3	73.41	
643		2	1	2			68.02	*
644		2	1		2		68.52	*
645		2	1			2	69.03	*
646		2	2	1	1		72.08	
647		2	2		1	1	72.52	
648		2	2	1		1	72.30	
649		2	2	2			71.86	
650		2	2		2		72.30	
651		2	2			2	72.75	
652		2	2	1			67.77	*
653		2	2		1		68.02	*
654		2	2			1	68.27	*
655		2	2				54.45	*
656		2	3	1			71.63	
657		2	3		1		71.86	
658		2	3			1	72.08	
659		2	3				67.52	*
660		2	4				71.41	
661		2		1			41.62	
662		2			1		41.87	
663		2				1	42.12	
664		2		3	1		72.52	
665		2		3		1	72.75	
666		2		1	3		72.97	
667		2			3	1	73.41	
668		2		2	1	1	72.97	
669		2		1	2	1	73.19	
670		2		1	1	2	73.41	
671		2		1	1	1	69.03	
672		2		1	1		55.20	

	A	B	C	D	E	F	G	H
673	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
674		2		1		1	55.46	
675		2			1	1	55.71	
676		2		1		3	73.64	
677		2			1	3	73.41	
678		2					28.26	
679		2		2	2		72.75	
680		2		2		2	73.19	
681		2			2	2	73.64	
682		2		4			72.30	
683		2			4		73.19	
684		2				4	74.08	
685		2		3			68.27 *	
686		2			3		69.03 *	
687		2				3	69.84 *	
688		2		2	1		68.52 *	
689		2		2		1	68.78 *	
690		2		1	2		68.78 *	
691		2			2	1	69.28 *	
692		2		1		2	69.28 *	
693		2			1	2	69.53 *	
694		2		2			54.95 *	
695		2			2		55.46 *	
696		2				2	55.96 *	
697		3	1	1			67.52	
698		3	1		1		67.77	
699		3	1			1	68.02	
700		3	1	2			71.63	
701		3	1		2		72.08	
702		3	1			2	72.52	
703		3	1	1	1		71.86	
704		3	1	1		1	72.08	
705		3	1		1	1	72.30	
706		3	1				54.19	
707		3	2	1			71.41	
708		3	2		1		71.63	
709		3	2			1	71.86	
710		3	2				67.26 *	
711		3	3				71.19	
712		3		1	1		68.02	
713		3		1		1	68.27	
714		3			1	1	68.52	
715		3		2	1		72.08	
716		3		2		1	72.30	
717		3		1	2		72.30	
718		3			2	1	72.75	
719		3		1		2	72.75	
720		3			1	2	72.97	

	A	B	C	D	E	F	G	H
	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
721								
722		3		1	1	1	72.52	
723		3					41.11	
724		3		3			71.86	
725		3			3		72.52	
726		3				3	73.19	
727		3		1			54.45	
728		3			1		54.7	
729		3				1	54.95	
730		3		2			67.77 *	
731		3			2		68.27 *	
732		3				2	68.78 *	
733		4	1		1		71.41	
734		4	1			1	71.63	
735		4	1	1			71.19	
736		4	1				67.01 *	
737		4	2				70.97	
738		4		1	1		71.63	
739		4		1		1	71.86	
740		4			1	1	72.08	
741		4		2			71.41	
742		4			2		71.86	
743		4				2	72.3	
744		4					53.94 *	
745		4		1			67.26 *	
746		4			1		67.52 *	
747		4				1	67.77 *	
748		5	1				70.75	
749		5		1			70.97	
750		5			1		71.19	
751		5				1	71.41	
752		5					68.76 *	
753		6					70.52	
754			1				15.82	
755			1	1			29.02	
756			1		1		29.27	
757			1			1	29.52	
758			1	2			42.37	
759			1	1	1		42.63	
760			1		2		42.88	
761			1	1		1	42.88	
762			1		1	1	43.13	
763			1			2	43.38	
764			1	1	1	1	56.46	
765			1	3	1		69.28	
766			1	3		1	69.53	
767			1	1	3		69.78	
768			1		3	1	70.29	

	A	B	C	D	E	F	G	H
769	SIZE 30	SIZE 32	SIZE 34	SIZE 36 -	SIZE 38	SIZE 40	Length (inches)	
770			1	1		3	70.54	
771			1		1	3	70.79	
772			1	4	1		73.19	
773			1	4		1	73.41	
774			1	3	2		73.41	
775			1	2	3		73.64	
776			1	1	4		73.86	
777			1	3		2	73.86	
778			1		4	1	74.30	
779			1		3	2	74.52	
780			1	1		4	74.75	
781			1		1	4	74.97	
782			1	2		1	56.21	
783			1	1		2	56.72	
784			1	1	1	2	70.29	
785			1	2	2	1	73.86	
786			1	2	1	2	74.08	
787			1	1	2	2	74.30	
788			1	2	1	1	69.78	
789			1	1	2	1	70.04	
790			1	1	1	2	70.29	
791			1	3	1	1	73.64	
792			1	1	3	1	74.08	
793			1	1	1	3	74.52	
794			1	2	1		55.96	
795			1	2		1	56.21	
796			1	1	2		56.21	
797			1		2	1	56.72	
798			1	1		2	56.72	
799			1		1	2	56.97	
800			1	2		3	74.30	
801			1		2	3	74.75	
802			1	5			72.97	
803			1		5		74.08	
804			1			5	75.19	
805			1	3			55.71	
806			1		3		56.46	
807			1			3	57.22	
808			1	4			69.03 *	
809			1		4		70.04 *	
810			1			4	71.04 *	
811			1	2	2		69.53 *	
812			1		2	2	70.54 *	
813			1	2		2	70.04 *	
814			2	1			42.12	
815			2		1		42.37	
816			2			1	42.63	

	A	B	C	D	E	F	G	H
817	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
818			2	3	1		72.97	
819			2	3		1	73.19	
820			2	1	3		73.41	
821			2		3	1	73.86	
822			2	1		1	55.96	
823			2	2	1	1	73.41	
824			2	1	2	1	73.64	
825			2	1	1	2	73.86	
826			2	1	1	1	69.53	
827			2	1	1		55.71	
828			2	1		1	55.96	
829			2		1	1	56.21	
830			2	1		3	74.08	
831			2		1	3	74.30	
832			2				23.76	
833			2	2	2		73.19	
834			2	2		2	73.64	
835			2		2	2	74.08	
836			2	4			72.75	
837			2		4		73.64	
838			2			4	74.52	
839			2	3			68.78 *	
840			2		3		69.53 *	
841			2			3	70.35 *	
842			2	2	1		69.03 *	
843			2	2		1	69.28 *	
844			2	1	2		69.28 *	
845			2		2	1	69.78 *	
846			2	1		2	69.78 *	
847			2		1	2	70.04 *	
848			2	2			55.46 *	
849			2		2		55.96 *	
850			2			2	56.46 *	
851			3	1	1		68.78	
852			3	1		1	69.03	
853			3		1	1	69.28	
854			3	2	1		72.75	
855			3	2		1	72.97	
856			3	1	2		72.97	
857			3		2	1	73.41	
858			3	1		2	73.41	
859			3		1	2	73.64	
860			3	1	1	1	73.19	
861			3				41.87	
862			3	3			72.52	
863			3		3		73.19	
864			3			3	73.86	

	A	B	C	D	E	F	G	H
865	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
866			3	1			55.20	
867			3		1		55.46	
868			3			1	55.71	
869			3	2			68.52 *	
870			3		2		69.03 *	
871			3			2	69.53 *	
872			4	1	1		72.52	
873			4	1		1	72.75	
874			4		1	1	72.97	
875			4	2			72.30	
876			4		2		72.75	
877			4			2	73.19	
878			4				54.95 *	
879			4	1			68.27 *	
880			4		1		68.52 *	
881			4			1	68.78 *	
882			5	1			72.08	
883			5		1		72.30	
884			5			1	72.52	
885			5				68.02 *	
886			6				71.86	
887				1	1	1	43.38	
888				1	1	2	57.22	
889				1	1	3	71.04	
890				1	1	4	75.19	
891				1	1		29.52	
892				1	2	1	56.97	
893				1	2	2	70.79 *	
894				1	2	3	74.97	
895				1	2		43.13	
896				1	3	1	70.54	
897				1	3	2	74.75	
898				1	3		56.72	
899				1	4	1	74.52	
900				1	4		70.30 *	
901				1	5		74.30	
902				1		1	29.77	
903				1		2	43.64	
904				1		3	57.47	
905				1		4	71.19 *	
906				1		5	75.41	
907				1			16.07	
908				2	1	1	56.72	
909				2	1	2	70.30 *	
910				2	1	3	74.75	
911				2	1		42.88	
912				2	2	1	70.29 *	

	A	B	C	D	E	F	G	H
913	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
914				2	2	2	74.52	
915				2	2		56.46	*
916				2	3	1	74.30	
917				2	3		70.03	*
918				2	4		74.08	
919				2		1	43.13	
920				2		2	56.97	*
921				2		3	70.86	*
922				2		4	74.97	
923				2			29.27	
924				3	1	1	70.04	
925				3	1	2	74.30	
926				3	1		56.21	
927				3	2	1	74.08	
928				3	2		69.78	*
929				3	3		73.86	
930				3		1	56.46	
931				3		2	70.29	*
932				3		3	74.52	
933				3			42.63	
934				4	1	1	73.86	
935				4	1		69.53	*
936				4	2		73.64	
937				4		1	69.78	*
938				4		2	74.08	
939				4			55.96	*
940				5	1		73.41	
941				5		1	73.64	
942				5			69.28	*
943				6			73.19	
944					1	1	30.03	
945					1	2	43.89	
946					1	3	57.72	
947					1	4	71.41	*
948					1	5	75.64	
949					1		16.33	
950					2	1	43.64	
951					2	2	57.46	*
952					2	3	71.30	*
953					2	4	75.41	
954					2		29.77	
955					3	1	57.22	
956					3	2	71.04	*
957					3	3	75.19	
958					3		43.38	
959					4	1	70.79	*
960					4	2	74.97	

	A	B	C	D	E	F	G	H
961	SIZE 30	SIZE 32	SIZE 34	SIZE 36	SIZE 38	SIZE 40	Length (inches)	
962					4		56.97	*
963					5	1	74.75	
964					5		70.54	*
965					6		74.52	
966						1	16.59	
967						2	30.28	
968						3	44.14	
969						4	57.98	*
970						5	71.8	*
971						6	75.86	

Appendix D: Prototype Software Computer Codes

Savings Algorithm Source Code

```

1  /* -----
2  -- $Header:: D:/cups/src/savings/case_ai.c   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : case_ai.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To compute the savings if ply heights and sizes
12 -           in both sections are the same.
13 -
14 -
15 - MODIFICATION HISTORY-
16 -
17 - -----*/
18 #include <stdio.h>
19 #include <stdlib.h>
20 #include "savedec.h"
21 #include "saveicl.h"
22
23 float case_ai(sect1, sect2, cut_cost)
24
25     section_t sect1;
26     section_t sect2;
27     int      cut_cost;
28
29 {
30     int i;
31     int e = 0;
32     float savings;
33
34     for (i=0; i< num_of_sizes; i++) {
35         e = e + (order.perimeter[i] * sect1.sizes[i]);
36         e = e + (order.perimeter[i] * sect2.sizes[i]);
37     }
38
39     savings = (float) cut_cost * e;
40
41     return(savings);
42
43 }
44

```

```

1  /* -----
2  -- $Header::  D:/cops/src/savings/case_ail.c    January 1991
3  -- -----*/
4
5  /*-----
6  -  FILE NAME    : case_ail.c
7  -  PROGRAMMER   : Terri A. Smith
8  -  DATE WRITTEN : January 1991
9  -  ADDRESS      : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 -  PURPOSE- To compute the savings if the units or ply
12 -            height is not the same in two sections.
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include "savedec.h"
19 #include "savelcl.h"
20
21 float case_ail(sect1, sect2, unit_cost)
22
23     section_t sect1;
24     section_t sect2;
25     int unit_cost;
26
27 {
28     int i;
29     int e = 0;
30     float savings;
31     float sect1_inch;
32     float sect2_inch;
33     float merge_inch;
34     order_t merged_order;
35
36     sect1_inch = find_inches(sect1.sizes);
37     sect2_inch = find_inches(sect2.sizes);
38
39     for (i=0; i< num_of_sizes; i++) {
40         merged_order[i] = 0;
41         merged_order[i] = merged_order[i] + sect1.sizes[i];
42         merged_order[i] = merged_order[i] + sect2.sizes[i];
43     }
44
45     merge_inch = find_inches(merged_order);
46
47     savings = unit_cost * sect1.ply_height * (sect1_inch + sect2_inch - merge_inch);
48
49     return(savings);
50
51 }
52

```

```

1  /* -----
2  -- $Header::  D:/cops/src/savings/compute.c   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : compute.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To determine which method to use to compute
12 -           the savings.
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <memory.h>
19 #include "savedec.h"
20 #include "save1cl.h"
21
22 float compute_savings(sect1, sect2, cut_cost, unit_cost, temp_save,
23                      max_sizes, max_ply)
24
25     section_t sect1;
26     section_t sect2;
27     int cut_cost;
28     int unit_cost;
29     savings_t *temp_save;
30     int max_sizes;
31     int max_ply;
32
33 {
34     int i;
35     int e = 0;
36     float savings = (float) 0.0;
37     float save2 = (float) 0.0;
38     char match = 1;
39     int num_units = 0;
40     int j, k, count;
41     char match2;
42
43     temp_save->ply1 = sect1.ply_height;
44     temp_save->ply2 = sect2.ply_height;
45
46     for (i=0; i<num_of_sizes; i++) {
47         if (sect1.sizes[i] != sect2.sizes[i])
48             match = 0;
49         num_units = num_units + sect1.sizes[i];
50         num_units = num_units + sect2.sizes[i];
51     }
52
53
54     if (match) { /* sizes in sections are the same */
55         if ((sect1.ply_height + sect2.ply_height) <= max_ply) {
56             savings = case_ai(sect1, sect2, cut_cost);
57             temp_save->type= 1;

```

```

58         temp_save->ply_height = sect1.ply_height + sect2.ply_height;
59     }
60
61     else if (num_units <= max_sizes) {
62         save2 = case_a11(sect1, sect2, unit_cost);
63
64         /*      if ((save2 > savings) || (temp_save->ply_height > max_ply)) {*/
65             temp_save->type= 2;
66             savings = save2;
67
68             temp_save->ply_height = sect1.ply_height;
69         /* } */
70     }
71 }
72
73 else if ((sect1.ply_height == sect2.ply_height) && (num_units <= max_sizes)) {
74     savings = case_a11(sect1, sect2, unit_cost);
75     temp_save->type= 3;
76     temp_save->ply_height = sect1.ply_height;
77 }
78
79 else if (num_units <= max_sizes) {
80     temp_save->ply_height = sect1.ply_height;
81     savings = case_a11(sect1, sect2, unit_cost);
82     temp_save->type= 4;
83 }
84
85 temp_save->savings = savings;
86
87 return(savings);
88
89 }
90

```

```

1  /* -----
2  -- $Header::  D:/cops/src/savings/findinch.c   January 1991
3  -- -----*/
4
5  /*-----
6  -  FILE NAME      : Findinch.c
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN  : January 1991
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- To find the length (in inches) in the list of Is
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "savedec.h"
19 #include "savelcl.h"
20
21 float find_inches(sizes)
22
23     order_t sizes;
24
25 {
26     int i, j;
27     char match = 0;
28
29     i = 0;
30     while ((!match) && (i < num_list)) {
31         match = 1;
32         for (j=0; j<num_of_sizes; j++) {
33             if (sizes[j] != list[i].sizes[j])
34                 match = 0;
35         }
36         ++i;
37     }
38
39     if (match)
40         return(list[--i].inches);
41     else {
42         printf(" COULDNT FIND ");
43         for (i=0; i<num_of_sizes; i++) {
44             if (sizes[i] > 0)
45                 printf("%d %s ", sizes[i], order.ch_sizes[i]);
46         }
47         printf("\n");
48         exit(0);
49     }
50 }
51
52

```

```

1  /* -----
2  -- $Header:: D:/cops/src/savings/getparm.c    December 1990
3  -----*/
4
5  /*-----
6  - FILE NAME      : Getparm.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To read in the parameters from a file
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "savedec.h"
19 #include "saveicl.h"
20
21 int get_parameters(ou_units, max_ply, max_sizes,
22                   k, init_ply, q, cut_cost, unit_cost)
23
24     int *ou_units;
25     int *max_ply;
26     int *max_sizes;
27     int *k;
28     int *init_ply;
29     int *q;
30     int *cut_cost;
31     int *unit_cost;
32
33 {
34     int i, j, m, l;
35     FILE *fp = NULL;
36     int quantity;
37     float temp;
38     char match;
39
40     if ((fp = fopen("INPUT", "r")) == NULL) {
41         printf("Cannot open input file - getparm.c");
42         exit(0);
43     }
44
45     /* set order and list values to -1 */
46     for (i = 0; i < MAX_SIZES; i++) {
47         order.number[i] = 0;
48         order.ch_sizes[i][0] = 0;
49         order.perimeter[i] = 0;
50     }
51
52     for (i=0; i<MAX_LIST; i++) {
53         list[i].inches = (float) 0.0;
54
55         for (j = 0; j < MAX_SIZES; j++)
56             list[i].sizes[j] = 0;
57     }

```



```

58
59
60 fscanf(fp,"%d", ou_units);
61 fscanf(fp,"%d", max_ply);
62 fscanf(fp,"%d", max_sizes);
63 fscanf(fp,"%d", init_ply);
64 fscanf(fp,"%d", k);
65 fscanf(fp,"%d", cut_cost);
66 fscanf(fp,"%d", unit_cost);
67 fscanf(fp,"%d", q);
68
69
70 /* Input Order */
71 for (i = 0; i < MAX_SIZES; i++) {
72     fscanf(fp,"%d", &order.number[i]);
73     if (order.number[i] == -1) {
74         order.number[i] == 0;
75         break;
76     }
77
78     fscanf(fp,"%d", &order.perimeter[i]);
79     fscanf(fp,"%s", order.ch_sizes[i]);
80 }
81
82 num_of_sizes = i;
83
84
85 /* Input List */
86 i=0;
87 while(1) {
88
89     fscanf(fp,"%d", &quantity);
90
91     if (quantity == -2)
92         break;
93
94     while (quantity != -1) {
95
96         fscanf(fp,"%d", &m);
97
98         if (m >= num_of_sizes) {
99             printf("ERROR in reading size variable - getparm.c");
100             exit(0);
101         }
102
103         list[i].sizes[m] = quantity;
104
105         fscanf(fp,"%d", &quantity);
106     }
107
108     fscanf(fp,"%f", &list[i].inches);
109
110     ++i;
111 }
112
113 fclose(fp);
114

```

```
115     return(i);  
116 }  
117
```

```

1  /* -----
2  -- $Header:: D:/cops/src/savings/globals.h   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Globals.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To declare all global variables
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include "savedec.h"
17 #include "zavelcl.h"
18
19     ord_var_t order;
20
21     list_t    *list = NULL;
22
23     int       num_of_sizes;
24
25     int       num_list;
26
27     int       total_order = 0;
28
29     int       curr_tot = 0;
30
31     int       num_old_sect = 0;
32
33     section_t *old_sect = NULL;

```

```
1 INCLUDES = savedec.h
2 LIBNAME = savelib
3
4
5 OBJS = \
6     globals.obj \
7     getparm.obj \
8     findinch.obj \
9     case_ai.obj \
10    case_aif.obj \
11    compute.obj
12
13
14 .c.obj:
15     $(CC)
16     $(LIB)
17
18
19 globals.obj : globals.c $(INCLUDES)
20
21 getparm.obj : getparm.c $(INCLUDES)
22
23 findinch.obj : findinch.c $(INCLUDES)
24
25 case_ai.obj : case_ai.c $(INCLUDES)
26
27 case_aif.obj : case_aif.c $(INCLUDES)
28
29 compute.obj : compute.c $(INCLUDES)
30
31 savings.obj : savings.c $(INCLUDES)
32
33 savings.exe : savings.obj $(OBJS)
34     cl savings /link savelib.lib
35
36
37 $(B)\savings.exe : savings.exe
38     $(CP)
39
40 $(I)\savedec.h : savedec.h
41     $(CP)
42
43
```

```

1  /* -----
2  -- $Header:: D:/cops/src/savings/savedec.h    December 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Savedec.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332   (404) 894-8952
10 -
11 - PURPOSE- To define all variables and procedures
12 -
13 - -----*/
14 #ifndef SAVEDEC_H
15 #define SAVEDEC_H
16
17 #define MAX_LIST 1000
18 #define MAX_SIZES 25
19 #define MAX_SAVINGS 2
20
21
22 typedef int order_t(MAX_SIZES);
23
24 typedef char sizes_t(MAX_SIZES)[10];
25
26 typedef struct (
27     order_t  number;
28     sizes_t  ch_sizes;
29     int      perimeter(MAX_SIZES);
30 ) ord_var_t;
31
32 typedef struct (
33     order_t  sizes;
34     float    inches;
35 ) list_t;
36
37 typedef struct (
38     order_t  sizes;
39     int      ply_height;
40     char     merged;
41 ) section_t;
42
43 typedef struct (
44     int sect1;
45     int sect2;
46     int ply_height;
47     float savings;
48     int type;
49     int ply1;
50     int ply2;
51 ) savings_t;
52
53
54 int get_parameters(int *units, int *max_ply, int *max_sizes, int *k,
55                  int *init_ply, int *q, int *cut_cost, int *unit_cost);
56
57 float find_inches(order_t sizes);

```

```
58
59 float case_ai(section_t sect1, section_t sect2, int cut_cost);
60
61 float case_aii(section_t sect1, section_t sect2, int unit_cost);
62
63 float compute_savings(section_t sect1, section_t sect2, int cut_cost,
64                       int unit_cost, savings_t *temp_save, int max_sizes, int max_ply);
65
66
67 #endif
```

```

1  /* -----
2  -- $Header::  D:/cops/src/savings/savedec.h    December 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Savelcl.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332   (404) 894-8952
10 -
11 - PURPOSE- To define all global variables
12 -
13 -
14 - -----*/
15 #ifndef SAVELCL_H
16 #define SAVELCL_H
17
18
19 extern ord_var_t order;
20 extern list_t *list;
21 extern int num_list;
22 extern int num_of_sizes;
23 extern int total_order;
24 extern int curr_tot;
25 extern int num_old_sect;
26 extern section_t *old_sect;
27
28
29 #endif

```

```

1  /* -----
2  -- $Header:: D:/cops/src/savings/savings.c   December 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Savings.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Mai : program which controls execution of other procedures
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <malloc.h>
17 #include <memory.h>
18 #include <stdlib.h>
19 #include <string.h>
20 #include <time.h>
21 #include <math.h>
22 #include "savedec.h"
23 #include "savelcl.h"
24
25
26 #define clock() time(NULL)
27
28 main(argv, argc)
29     int argv;
30     char *argv[];
31
32 {
33     /* Input Variables */
34     int    ou_units;           /* # of units over/under allowed */
35     int    max_ply;           /* max ply height allowed */
36     int    max_sizes;         /* # of sizes allowed / section */
37     int    init_ply;          /* initial ply height */
38     int    k;                 /* # c' merges allowed */
39     int    cut_cost;           /* cutting cost / inch */
40     int    unit_cost;          /* unit cost */
41     int    q;                 /* ply used for initial sections */
42
43     /* Output Variables */
44     float  tot_length;         /* the total amt of fabric needed*/
45     int    unit_dev;           /* deviation of units to cut from order */
46     char    unit_string[10];   /* string for over, under */
47     float  inches;             /* used in output */
48
49     int    i, j, x, y;         /* counters */
50     int    curr_sect = 0;       /* current section */
51     section_t new_sect;         /* new sections */
52     section_t *save_sect = NULL; /* new sections */
53     div_t  n;                   /* quotient and remainder */
54     savings_t temp_save;        /* temp savings - 1 structure */
55     savings_t *save_list = NULL; /* savings list */
56     int    num_savings;         /* # of savings in list */
57     int    m, l, r, s;         /* counters */

```



```

58 char mergers_possible = 1; /* Boolean for loop */
59 int num_new_sect; /* # of total new_sect */
60 int num_save_sect; /* # of total new_sect */
61 int num_units; /* # of units in one section */
62 int unit_count; /* # of units in all sections */
63 int order_count; /* # of units in order */
64 int num_mergers = 0; /* # of mergers */
65 FILE *fp; /* output file pointer */
66 clock_t start_time, end_time; /* times */
67 double total_time; /* total time program runs */
68 float marker; /* inches in marker */
69 float tot_marker; /* total inches in all markers */
70 char match2; /* boolean value */
71 int count; /* counts the sections */
72 int old_ply; /* ply height of older section */
73 section_t temp_sect; /* temporary section */
74 int add_sect; /* # of sections to add */
75 order_t temp_order;
76 order_t hold_sizes;
77 int abs1, abs2;
78
79 start_time = clock();
80
81 if ((fp = fopen("OUTPUT", "w")) == NULL) {
82     printf("CANNOT OPEN OUTPUT FILE savings.c\n");
83     exit(0);
84 }
85
86 /*
87 Allocation of input list
88 */
89 if ((list = (list_t *)malloc(MAX_LIST * sizeof(list_t))) == NULL) {
90     printf("ALLOCATION ERROR FOR LIST savings.c\n");
91     exit(0);
92 }
93
94 /*
95 Get parameters and print initial stuff to output file
96 */
97 num_list = get_parameters(&ou_units, &max_ply, &max_sizes, &k, &init_ply,
98 &q, &cut_cost, &unit_cost);
99
100 fprintf(fp, "SAVINGS ALGORITHM 2\n\n");
101 fprintf(fp, "MAX PLY = %d MAX # OF UNITS PER SECTION = %d\n", max_ply, max_sizes);
102 /* fprintf(fp, "UNIT COST = %d cents CUT COST = %d cents\n", unit_cost, cut_cost);
103 */
104 fprintf(fp, "K = %d INIT PLY = %d Q = %d\n\n", k, init_ply, q);
105 fprintf(fp, "ORDER\n");
106 for (i=0; i<num_of_sizes; i++) {
107     fprintf(fp, "%d SIZE %s\n", order.number[i], order.ch_sizes[i]);
108 }
109
110 /*
111 Allocate space for two sets of sections
112 */
113 for (i=0; i<num_of_sizes; i++)
114     total_order = total_order + order.number[i];

```

```

115
116 if ((old_sect = (section_t *)malloc(total_order * sizeof(section_t))) == NULL) {
117     printf("ALLOCATION ERROR FOR OLD SECTION    savings.c\n");
118     exit(0);
119 }
120
121 if ((save_sect = (section_t *)malloc(total_order * sizeof(section_t))) == NULL) {
122     printf("ALLOCATION ERROR FOR SAVE SECTION    savings.c\n");
123     exit(0);
124 }
125
126 for (i=0; i<num_of_sizes; i++) {
127     temp_order[i] = order.number[i];
128 }
129
130 /*
131     Assign each unit in order to a separate section of initial
132     ply height
133 */
134 for (i =0; i < total_order; i++) {
135     old_sect[i].ply_height = q;
136     old_sect[i].merged = 0;
137
138     for (j=0; j< MAX_SIZES; j++)
139         old_sect[i].sizes[j] = 0;
140 }
141
142 unit_count = 0;
143 for (i=0; i<num_of_sizes; i++) {
144     n = div(order.number[i], q);
145     for (j=0; j<n.quot; j++) {
146         old_sect[curr_sect].sizes[i] = 1;
147         ++curr_sect;
148         unit_count = unit_count + q;
149     }
150 }
151
152 for (i=0; i<num_of_sizes; i++) {
153     n = div(order.number[i], q);
154     for (j=0; j<n.rem; j++) {
155         unit_dev = total_order - unit_count;
156         if ((ou_units - unit_dev) < 0) {
157             old_sect[curr_sect].sizes[i] = 1;
158             ++curr_sect;
159             unit_count = unit_count + q;
160         }
161     }
162 }
163
164 num_old_sect = curr_sect;
165
166 /*
167     Allocate space for savings list and initialize
168 */
169 if ((save_list = (savings_t *)malloc(MAX_SAVINGS * sizeof(savings_t))) == NULL) {
170     printf("ALLOCATION ERROR FOR SAVINGS LIST    savings.c\n");
171     exit(0);

```

```

172     }
173
174     for (i=0; i<MAX_SAVINGS; i++) {
175         save_list[i].savings = (float) 0.0;
176         save_list[i].ply_height = 0;
177         save_list[i].type = 0;
178     }
179
180
181     /*
182     Main loop in the Savings algorithm:
183     - creates a savings list and merges sections one at a time.
184     - a temporary section is created and merged with initial sections
185       until it is completely filled. It is then save in the
186       save_section and a new temporary section is started
187     - When all sections are saved to the save_section then program is
188       terminated
189     */
190
191     num_save_sect = 0;
192     while (mergers_possible) {
193
194         for (i=0; i<MAX_SAVINGS; i++) {
195             save_list[i].savings = (float) 0.0;
196             save_list[i].ply_height = 0;
197             save_list[i].type = 0;
198         }
199
200
201         printf("NUM OLD SECT = %d\n", num_old_sect);
202         if (num_old_sect <= 1)
203             break;
204
205         num_units = 0;
206
207         /*
208         When the max number of units per section is reached, the section
209         is saved in save_section
210         */
211         for (j=0; j < num_of_sizes; j++)
212             num_units = old_sect[0].sizes[j] + num_units;
213
214         if (num_units >= max_sizes) {
215             memcpy(&save_sect[num_save_sect], &old_sect[0], sizeof(section_t));
216
217             for (i = 0; i<num_old_sect-1; i++)
218                 memcpy(&old_sect[i], &old_sect[i+1], sizeof(section_t));
219
220             ++num_save_sect;
221             --num_old_sect;
222         }
223
224
225         mergers_possible = 0;
226         num_savings = 0;
227
228         /*

```

```

229      Create Savings List
230      */
231      i = 0;
232      for (j=i+1; j<num_old_sect; j++) {
233          temp_save.sect1 = i;
234          temp_save.sect2 = j;
235          temp_save.ply_height = 0;
236          temp_save.savings = (float) 0.0;
237          temp_save.type = 0;
238
239          compute_savings(old_sect[i], old_sect[j], cut_cost, unit_cost,
240                          &temp_save, max_sizes, max_ply);
241
242          m = 0;
243          while((m < num_savings) &&
244              (temp_save.savings <= save_list[m].savings))
245              ++m;
246
247          if (m != MAX_SAVINGS) {
248
249              for (l = num_savings; l > m; l--) {
250                  memcpy(&save_list[l], &save_list[l-1], sizeof(savings_t));
251              }
252
253              memcpy(&save_list[l], &temp_save, sizeof(savings_t));
254              if (num_savings < MAX_SAVINGS-1)
255                  ++num_savings;
256          }
257      }
258
259      /*
260      Merge Sections
261      */
262
263      new_sect.ply_height = q;
264      new_sect.merged = 0;
265
266      for (j=0; j< MAX_SIZES; j++)
267          new_sect.sizes[j] = 0;
268
269      num_mergers = 0;
270      m = 0;
271
272      for (i=0; i<num_savings; i++) {
273          r = save_list[i].sect1;
274          s = save_list[i].sect2;
275          num_units = 0;
276
277          for (j=0; j < num_of_sizes; j++) {
278              num_units = old_sect[r].sizes[j] + num_units;
279              if (save_list[i].type != 1)
280                  num_units = old_sect[s].sizes[j] + num_units;
281          }
282
283          if ((save_list[i].ply_height <= max_ply) &&
284              (old_sect[r].merged) &&
285              (old_sect[s].merged) &&

```

```

286      (num_units <= max_sizes) &&
287      (save_list[i].type != 0)) {
288
289      mergers_possible = 1;
290      old_sect[r].merged = 1;
291      old_sect[s].merged = 1;
292
293      new_sect.ply_height = save_list[i].ply_height;
294      for (j=0; j<num_of_sizes; j++) {
295          new_sect.sizes[j] = new_sect.sizes[j] +
296                          old_sect[r].sizes[j];
297          if (save_list[i].type != 1)
298              new_sect.sizes[j] = new_sect.sizes[j] +
299                          old_sect[s].sizes[j];
300      }
301
302      /*
303      If the savings is achieved by rearranging sizes
304      in one section (not by putting plys on top of
305      each other), then the two ply heights of the sections
306      must be manipulated to keep the order correct.
307      e.g. If one section has ply 3 and the other ply 10
308      one section of ply 3 with bothe sizes combinations
309      is made and 7 sections of kept in the list of merging
310      sections
311      */
312      if (save_list[i].type != 1) {
313
314          for (x=0; x<num_of_sizes; x++)
315              hold_sizes[x] = old_sect[s].sizes[x];
316
317          /*
318          Count how many sections in the sections list match
319          the given section to merge
320          */
321          count = 0;
322          for (l=1; l<num_old_sect; l++) {
323              match2 = 1;
324              for (j=0; j<num_of_sizes; j++) {
325                  if (old_sect[s].sizes[j] != old_sect[l].sizes[j])
326                      match2 = 0;
327              }
328
329              if (match2)
330                  ++count;
331          }
332
333          /*
334          If the count is greater than the ply height of
335          the temporary section, combine the two sections
336          with the ply height of temporary section and then
337          delete that number (ply height) of sections from
338          the section list
339          */
340          if ((save_list[i].ply1 / q) <= count) {
341              count = save_list[i].ply1 / q;
342              for (l=1; l<num_old_sect; l++) {

```

```

343         match2 = 1;
344         for (j=0; j<num_of_sizes; j++) {
345             if (hold_sizes[j] != old_sect[l].sizes[j])
346                 match2 = 0;
347         }
348
349         if ((match2) && (count > 0)) {
350             for (m=l; m<num_old_sect-1; m++)
351                 memcpy(&old_sect[m], &old_sect[m+1], sizeof(section_t));
352
353             --num_old_sect;
354             --count;
355             --l;
356         }
357     }
358     } /* save_list[i].ply1 <= count */
359
360     /*
361     else if the count is less than the ply height
362     of the temporary section, then the temp section
363     will have a ply height of count and sections are
364     added back to the section list based on the the
365     old_ply (of temp section) minus the the count
366     */
367     else {
368         if (count > 0) {
369             if (old_sect[0].ply_height > count) {
370                 old_ply = old_sect[0].ply_height;
371                 old_sect[0].ply_height = count;
372                 new_sect.ply_height = count;
373                 for (i=0; i<num_of_sizes; i++) {
374                     if (old_sect[0].sizes[i] > 0) {
375                         add_sect = old_ply - count;
376                         temp_sect.ply_height = q;
377                         temp_sect.merged = 0;
378
379                         for (j=0; j<num_of_sizes; j++)
380                             temp_sect.sizes[j] = 0;
381
382                         temp_sect.sizes[i] = 1;
383
384                         for (l=0; l<old_sect[0].sizes[i]; l++) {
385                             for (j=0; j<add_sect; j++)
386                                 memcpy(&old_sect[num_old_sect++], &temp_sect, sizeof(section_t))
387                         }
388                         } /* if old_sect[0].sizes[i] > 0 */
389                     } /* for i=0 etc */
390                 } /* old_sect[0].ply > 0 */
391
392                 for (l=1; l<num_old_sect; l++) {
393                     match2 = 1;
394                     for (j=0; j<num_of_sizes; j++) {
395                         if (hold_sizes[j] != old_sect[l].sizes[j])
396                             match2 = 0;
397                     }
398
399                     if ((match2) && (count > 0)) {

```

```

400         for (m=l; m<num_old_sect-1; m++)
401             memcpy(&old_sect[m], &old_sect[m+1], sizeof(section_t));
402
403         --num_old_sect;
404         --count;
405         --l;
406     }
407     } /* for l=1 etc */
408
409     } /* count > 0 */
410     } /* else */
411     } /* if type != 1 */
412
413     ++m;
414     if (++num_mergers >= k)
415         break;
416 }
417
418 memcpy(&old_sect[0], &new_sect, sizeof(section_t));
419
420 /*
421  Merges Complete
422  */
423
424 if (save_list[i].type == 1) {
425     for (i=s; i< num_old_sect-1; i++)
426         memcpy(&old_sect[i], &old_sect[i+1], sizeof(section_t));
427
428     --num_old_sect;
429 }
430
431 num_savings = 0;
432
433
434
435 } /* End of While (1) */
436
437 if (num_old_sect > 0) {
438     for (i=0; i<num_old_sect; i++) {
439         memcpy(&save_sect[num_save_sect++], &old_sect[i], sizeof(section_t));
440     }
441 }
442
443 /*
444  put final information in output file
445  */
446 end_time = clock();
447 total_time = ((double) end_time - start_time) / CLK_TCK;
448
449 fprintf(fp, "\n\n*****\n\n");
450 tot_length = (float) 0.0;
451 unit_dev = 0;
452 order_count = 0;
453 unit_count = 0;
454 tot_marker = (float) 0.0;
455
456 fprintf(fp, "THE # OF FINAL SECTIONS ARE : %d\n", num_save_sect);

```

```

457     for (i=0; i<num_save_sect; i++) {
458         fprintf(fp, "SECTION %d HAS PLY = %d\n", i, save_sect[i].ply_height);
459         for (j=0; j<num_of_sizes; j++) {
460             if (save_sect[i].sizes[j] > 0) {
461                 fprintf(fp, "                AND %d SIZE %s\n", save_sect[i].sizes[j], order.ch_sizes[
462                     unit_count = unit_count + (save_sect[i].sizes[j] * save_sect[i].ply_height);
463             }
464         }
465         marker = find_inches(save_sect[i].sizes);
466         inches = marker * save_sect[i].ply_height;
467         fprintf(fp, "MARKER LENGTH = %7.2f THE TOTAL LENGTH = %7.2f\n\n",
468             marker, inches);
469         tot_length = tot_length + inches;
470         tot_marker = tot_marker + marker;
471     }
472
473     for (j=0; j<num_of_sizes; j++)
474         order_count = order_count + order.number[j];
475
476     unit_dev = order_count - unit_count;
477     if (unit_dev > 0)
478         strcpy(unit_string, "UNDER");
479     else if (unit_dev == 0)
480         strcpy(unit_string, "\0");
481     else {
482         unit_dev = unit_dev * -1;
483         strcpy(unit_string, "OVER");
484     }
485
486     fprintf(fp, "TOT MARKER = %7.2f TOT LENGTH = %7.2f, UNIT OVER/UNDER = %d %s\n\n",
487         tot_marker, tot_length, unit_dev, unit_string);
488     fprintf(fp, "TOTAL TIME = %f SECONDS\n", total_time);
489
490
491     /*
492     Free all space and close output file
493     */
494     if (list != NULL)
495         free(list);
496
497     if (save_list != NULL)
498         free(save_list);
499
500     if (old_sect != NULL)
501         free(old_sect);
502
503
504     fclose(fp);
505
506     return(0);
507 }

```


Cherry Algorithm Source Code

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/cherdec.h    December 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Cherdec.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN   : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To define all variables and procedures
12 -
13 -
14 - -----*/
15 #ifndef CHERDEC_H
16 #define CHERDEC_H
17
18 #define MAX_LIST 1000
19 #define MAX_SIZES 25
20
21 typedef int order_t[MAX_SIZES];
22
23 typedef char sizes_t[MAX_SIZES][10];
24
25 typedef struct (
26     order_t number;
27     sizes_t ch_sizes;
28 ) ord_var_t;
29
30 typedef struct (
31     order_t sizes;
32     float inches;
33 ) list_t;
34
35 typedef struct (
36     order_t sizes;
37     int    ply_height;
38 ) section_t;
39
40 int get_parameters(int *units, int *max_ply, int *max_sizes);
41
42 float find_inches(order_t sizes);
43
44 float combine_inches(order_t set_s);
45
46 void check_inches(section_t *temp_secs, int *num_temp_secs);
47
48 void clear_temp(section_t *temp_secs, int *num_temp_secs);
49
50 void copy_hold_to_sections();
51
52 void ones(order_t set_s, section_t *temp_secs, int *num_temp_secs);
53
54 void twos(order_t set_s, section_t *temp_secs, int *num_temp_secs);
55
56 void threes(order_t set_s, section_t *temp_secs, int *num_temp_secs);
57

```

```
58 void fours(order_t set_s, section_t *temp_secs, int *num_temp_secs);  
59  
60 void fives(order_t set_s, section_t *temp_secs, int *num_temp_secs);  
61  
62 void sixes(order_t set_s, section_t *temp_secs, int *num_temp_secs);  
63  
64 #endif
```

```

1  /* -----
2  -- $Header::  D:/cops/src/savings/savedec.h    December 1990
3  -- -----*/
4
5  /*-----
6  -  FILE NAME      : Savelcl.h
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN   : December 1990
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 -  PURPOSE- To define all global variables
12 -
13 -
14 - -----*/
15 #ifndef CHERLCL_H
16 #define CHERLCL_H
17
18
19 extern ord_var_t order;
20 extern list_t *list;
21 extern int num_list;
22 extern int num_of_sizes;
23 extern order_t temp_order;
24 extern int num_sections;
25 extern float total_inches;
26 extern float prev_inch;
27 extern section_t *sections;
28 extern int num_hold_secs;
29 extern section_t *hold_secs;
30 extern int ply_height;
31
32
33 #endif

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/cherry.c  December 1990
3  ----- */
4
5  /*-----
6  -  FILE NAME    : Cherry.c
7  -  PROGRAMMER   : Terri A. Smith
8  -  DATE WRITTEN : December 1990
9  -  ADDRESS      : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- The main program which executes all other procedures
12 -
13 -
14 ----- */
15 #include <stdio.h>
16 #include <malloc.h>
17 #include <stdlib.h>
18 #include <string.h>
19 #include <memory.h>
20 #include <time.h>
21 #include "cherdec.h"
22 #include "cherlcl.h"
23
24 #define clock() time(NULL)
25
26 main(argv, argc)
27     int argv;
28     char *argv[];
29
30 {
31     /* Input Variables */
32     int    ou_units;      /* # of units over/under allowed */
33     int    max_ply;       /* max ply height */
34     int    max_sizes;     /* # of sizes allowed / section */
35
36     /* Output Variables */
37     int    num_temp_secs; /* # of total sections */
38     float  tot_length;    /* total length of fabric */
39     float  tot_marker;    /* total length of fabric */
40     section_t *temp_secs=NULL; /* each section description */
41     int    unit_dev = 0;  /* deviation of # of units from order */
42     char    unit_string[10]; /* string to print over, under */
43
44     int    q1;            /* largest quantity in order */
45     int    q2;            /* 2nd largest quantity in order */
46     int    s_var;         /* q2 minus ou_units */
47     order_t set_s;        /* set S of sizes */
48     int    max_sections=0; /* max # of sections to allocate */
49     char    repeat_loop;  /* boolean to loop again or not */
50     int    i, j, k, l,m,n; /* counters */
51     float  inches;        /* used for printing results */
52     float  marker;        /* used for printing results */
53     int    sets_cnt;      /* # of sizes in set S */
54     FILE    *fp;          /* output file pointer */
55     int    unit_count = 0; /* # of units in all sections */
56     int    order_count = 0; /* # of units in order */
57     int    ou_count = 0;  /* count to determine if repeat loop */

```

```

58     clock_t start_time, end_time;
59     double total_time;
60
61     start_time = clock();
62
63     /*
64     Open output file
65     */
66     if ((fp = fopen ("OUTPUT", "w")) == NULL ) {
67         printf("CANNOT OPEN OUTPUT FILE  cherry.c\n");
68         exit(0);
69     }
70
71     /*
72     Allocate space for the list of Is
73     */
74     if ((list = (list_t *)malloc(MAX_LIST * sizeof(list_t))) == NULL) {
75         printf("ALLOCATION ERROR FOR LIST  cherry.c\n");
76         exit(0);
77     }
78
79     num_list = get_parameters(&ou_units, &max_ply, &max_sizes);
80
81     fprintf(fp, "CHERRY ALGORITHM\n\n");
82     fprintf(fp, "MAX PLY = %d MAX # OF UNITS PER SECTION = %d\n", max_ply, max_sizes);
83     fprintf(fp, "\n ORDER\n");
84     for (i=0; i<num_of_sizes; i++) {
85         fprintf(fp, "%d SIZE %s\n", order.number[i], order.ch_sizes[i]);
86         order_count = order_count + order.number[i];
87     }
88
89     /*
90     Allocate space for the max number of sections
91     for the three list of sections
92     */
93     for (i=0; i< MAX_SIZES; i++) {
94         max_sections = max_sections + order.number[i];
95     }
96
97     if ((sections = (section_t *)malloc(max_sections * sizeof(section_t))) == NULL) {
98         printf("ALLOCATION ERROR FOR SECTIONS  cherry.c\n");
99         exit(0);
100     }
101
102     if ((temp_secs = (section_t *)malloc(max_sections * sizeof(section_t))) == NULL) {
103         printf("ALLOCATION ERROR FOR SECTIONS  cherry.c\n");
104         exit(0);
105     }
106
107     if ((hold_secs = (section_t *)malloc(max_sections * sizeof(section_t))) == NULL) {
108         printf("ALLOCATION ERROR FOR SECTIONS  cherry.c\n");
109         exit(0);
110     }
111
112
113     for (i=0; i<max_sections; i++) {
114         sections[i].ply_height = 0;

```

```

115     for (j=0; j<MAX_SIZES; j++)
116         sections[i].sizes[j] = 0;
117     }
118
119     num_sections = 0;
120
121     /*
122     Main Loop of program
123     */
124     while (1) {
125
126         for (i=0; i<max_sections; i++) {
127             temp_secs[i].ply_height = 0;
128             for (j=0; j<MAX_SIZES; j++)
129                 temp_secs[i].sizes[j] = 0;
130         }
131
132         repeat_loop = 0;
133
134         /*
135         Choose Q1 and Q2
136         */
137         q1 = 0;
138         q2 = 0;
139
140         for (i=1; i<num_of_sizes; i++) {
141             if (order.number[i] > order.number[q1])
142                 q1 = i;
143         }
144
145         q2 = 0;
146         for (i=0; i<num_of_sizes; i++) {
147             if (i != q1) {
148                 if (order.number[i] >= 0) {
149                     q2 = i;
150                     break;
151                 }
152             }
153         }
154
155         for (i=0; i<num_of_sizes; i++) {
156             if (i != q1)
157                 if (order.number[i] >= order.number[q2])
158                     q2 = i;
159         }
160
161         if (order.number[q2] <= 0)
162             q2 = q1;
163
164         /*
165         Form set S with all the sizes remaining in the order
166         which have a quantity greater than or equal to q2 - the number
167         of units allowed over the specified demand
168         */
169         s_var = order.number[q2] - ou_units;
170
171

```

```

172     sets_cnt = 0;
173     for (i=0; i<MAX_SIZES; i++) {
174         if ((order.number[i] >= s_var) && (order.number > 0)) {
175             set_s[i] = 1;
176             ++sets_cnt;
177         }
178         else
179             set_s[i] = 0;
180     }
181
182     /*
183     Set ply height of next section to the min(q2, max ply)
184     */
185     ply_height = order.number[q2];
186     if (max_ply < order.number[q2])
187         ply_height = max_ply;
188
189
190     /*
191     Combine all possibilities of sections up to 5 units
192     per section
193     */
194     inches = (float) 9999.0;
195     for (i=0; i<MAX_SIZES; i++)
196         temp_order[i] = 0;
197     num_temp_secs = 0;
198
199     total_inches = (float) 0.0;
200
201
202     ones(set_s, temp_secs, &num_temp_secs);
203     check_inches(temp_secs, &num_temp_secs);
204     clear_temp(temp_secs, &num_temp_secs);
205
206     if ((sets_cnt > 1) && (max_sizes > 1)) {
207         twos(set_s, temp_secs, &num_temp_secs);
208         check_inches(temp_secs, &num_temp_secs);
209         clear_temp(temp_secs, &num_temp_secs);
210     }
211
212     if ((sets_cnt > 2) && (max_sizes > 2)) {
213         threes(set_s, temp_secs, &num_temp_secs);
214         check_inches(temp_secs, &num_temp_secs);
215         clear_temp(temp_secs, &num_temp_secs);
216     }
217
218     if ((sets_cnt > 3) && (max_sizes > 3)) {
219         fours(set_s, temp_secs, &num_temp_secs);
220         check_inches(temp_secs, &num_temp_secs);
221         clear_temp(temp_secs, &num_temp_secs);
222     }
223
224     if ((sets_cnt > 4) && (max_sizes > 4)) {
225         fives(set_s, temp_secs, &num_temp_secs);
226         check_inches(temp_secs, &num_temp_secs);
227         clear_temp(temp_secs, &num_temp_secs);
228     }

```



```

229
230     if ((sets_cnt > 5) && (max_sizes > 5)) {
231         sixes(set_s, temp_secs, &num_temp_secs);
232         check_inches(temp_secs, &num_temp_secs);
233         clear_temp(temp_secs, &num_temp_secs);
234     }
235
236     copy_hold_to_sections();
237
238     /*
239     Reduce the order demand
240     */
241     for (m=(num_sections - num_hold_secs); m<num_sections; m++) {
242         for (n=0; n< num_of_sizes; n++) {
243             if (sections[m].sizes[n] == 1) {
244                 order.number[n] = order.number[n] - ply_height;
245                 set_s[n] = 0;
246             }
247         }
248     }
249
250     /*
251     Repeat loop if the order contains a size w/ positive
252     quantity greater than the number of units allowed under the
253     specified demand, else break out of loop
254     */
255
256     ou_count = 0;
257     for (i=0; i<num_of_sizes; i++) {
258         ou_count = ou_count + order.number[i];
259         if (ou_count > ou_units)
260             repeat_loop = 1;
261     }
262
263     if (!repeat_loop)
264         break;
265
266     } /* END of While (1) */
267
268     end_time = clock();
269     total_time = ((double) end_time - start_time) / CLK_TCK;
270
271     /*
272     Print Out Results
273     */
274     fprintf(fp, "\n\n*****\n\n");
275     fprintf(fp, "THE NUMBER OF FINAL SECTIONS = %d\n", num_sections);
276
277     for (i=0; i<num_sections; i++) {
278         marker = find_inches(sections[i].sizes);
279         inches = marker * sections[i].ply_height;
280         total_inches = total_inches + inches;
281         tot_marker = tot_marker + marker;
282         fprintf(fp, "\nSECTION %d HAS PLY = %d\n", i, sections[i].ply_height);
283         for (j=0; j<num_of_sizes; j++) {
284             if (sections[i].sizes[j] > 0) {
285                 fprintf(fp, "        HAS %d SIZE %s\n", sections[i].sizes[j], order.ch_sizes[j]);

```

```

286         unit_count = unit_count + (sections[i].sizes[j] * sections[i].ply_height);
287     }
288 }
289 fprintf(fp, "MARKER INCHES = %7.2f and TOTAL INCHES %7.2f\n", marker, inches);
290 }
291 fprintf(fp, "\nTOTAL MARKER INCHES = %7.2f TOTAL INCHES = %7.2f\n", tot_marker, total_inches);
292
293
294 unit_dev = order_count - unit_count;
295 if (unit_dev > 0)
296     strcpy(unit_string, "UNDER");
297 else if (unit_dev == 0)
298     strcpy(unit_string, "\0");
299 else {
300     unit_dev = unit_dev * -1;
301     strcpy(unit_string, "OVER");
302 }
303
304 fprintf(fp, "UNIT OVER/UNDER = %d %s\n\n", unit_dev, unit_string);
305 fprintf(fp, "TOTAL_TIME = %f\n", total_time);
306
307
308 if (list != NULL)
309     free(list);
310
311 if (sections != NULL)
312     free(sections);
313
314 if (temp_secs != NULL)
315     free(temp_secs);
316
317 if (hold_secs != NULL)
318     free(hold_secs);
319
320 fclose(fp);
321
322 return(0);
323 }

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/chkinch.c    December 1990
3  -----*/
4
5  /*-----
6  - FILE NAME      : Chkinch.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To determine if the total inches calculated from
12 -           the last grouping of sections is less than any previous
13 -           grouping. If so the sections are saved in the hold
14 -           sections.
15 -
16 -
17 - -----*/
18 #include <stdio.h>
19 #include <malloc.h>
20 #include <stdlib.h>
21 #include <memory.h>
22 #include "cherdec.h"
23 #include "chericl.h"
24
25 void check_inches(temp_secs, num_temp_secs)
26     section_t *temp_secs;
27     int *num_temp_secs;
28
29 {
30
31     int m, i, j;
32
33     if ((total_inches < prev_inch) && (total_inches > (float) 0.0)) {
34         num_hold_secs = 0;
35         for (m=0; m<*num_temp_secs; m++) {
36             memcpy(&hold_secs[num_hold_secs], &temp_secs[m], sizeof(section_t));
37             hold_secs[num_hold_secs].ply_height = ply_height;
38             ++num_hold_secs;
39         }
40         prev_inch = total_inches;
41     }
42
43 }

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/clrtemp.c   December 1990
3  -----*/
4
5  /*-----
6  - FILE NAME   : Clrtemp.c
7  - PROGRAMMER  : Terri A. Smith
8  - DATE WRITTEN : January 1990
9  - ADDRESS     : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE-    Initializes the temp sections
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <malloc.h>
17 #include <stdlib.h>
18 #include <memory.h>
19 #include "cherdec.h"
20 #include "cherlcl.h"
21
22 void clear_temp(temp_secs, num_temp_secs)
23     section_t *temp_secs;
24     int *num_temp_secs;
25
26 {
27     int i, j;
28
29     total_inches = (float) 0.0;
30     for (i=0; i< *num_temp_secs; i++) {
31         for (j=0; j< num_of_sizes; j++) {
32             temp_secs[i].sizes[j] = 0;
33         }
34     }
35     *num_temp_secs = 0;
36
37 }

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/combine.c   January 1991
3  -- -----*/
4
5  /*-----
6  -  FILE NAME    : Combine.c
7  -  PROGRAMMER   : Terri A. Smith
8  -  DATE WRITTEN : January 1991
9  -  ADDRESS      : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- Finds the length (in inches) of the combine units
12 -             in one section.
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <string.h>
19 #include "cherdec.h"
20 #include "cherlcl.h"
21
22 float combine_inches(temp_order)
23
24     order_t temp_order;
25
26 {
27     float inches;
28
29     inches = find_inches(temp_order);
30
31     return(inches);
32 }
33

```

```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/cphold.c   December 1990
3  -----*/
4
5  /*-----
6  - FILE NAME      : Cphold.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Copies the temp sections into the hold sections.
12 -
13 -----*/
14 #include <stdio.h>
15 #include <malloc.h>
16 #include <stdlib.h>
17 #include <memory.h>
18 #include "cherdec.h"
19 #include "cherlcl.h"
20
21 void copy_hold_to_sections()
22 {
23     int m;
24
25     for (m=0; m<num_hold_secs; m++) {
26         memcpy(&sections[num_sections], &hold_secs[m], sizeof(section_t));
27         sections[num_sections].ply_height = ply_height;
28         ++num_sections;
29     }
30
31     prev_inch = (float) 9999.0;
32 }
33
34

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/findinch.c   January 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : Findinch.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE-   Finds the current unit grouping in the list of Is.
12 -             If it is not found program is exited.
13 -
14 -
15 - MODIFICATION HISTORY-
16 -
17 -----*/
18 #include <stdio.h>
19 #include <stdlib.h>
20 #include <string.h>
21 #include "cherdec.h"
22 #include "chericl.h"
23
24 float find_inches(sizes)
25
26     order_t sizes;
27
28 {
29     int i, j;
30     char match = 0;
31
32     i = 0;
33     while ((!match) && (i < num_list)) {
34         match = 1;
35         for (j=0; j<num_of_sizes; j++) {
36             if (sizes[j] != list[i].sizes[j])
37                 match = 0;
38         }
39         ++i;
40     }
41
42     if (match)
43         return(list[--i].inches);
44     else {
45         printf("\nCOULDNT FIND ");
46         for (i=0; i<num_of_sizes; i++) {
47             if (sizes[i] > 0)
48                 printf("%d %s ", sizes[i], order.ch_sizes[i]);
49             .
50         }
51         printf("\n");
52         exit(0);
53     }
54 }
55
56

```

```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/fives.c   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : fives.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Recursive procedure to group units in fives.
12 -
13 - -----*/
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include "cherdec.h"
18 #include "chericl.h"
19
20 void fives(set_s, temp_secs, num_temp_secs)
21
22     order_t set_s;
23     section_t *temp_secs;
24     int *num_temp_secs;
25 {
26     float inches;
27     int j, i, k, l, m, n;
28     order_t temp_order;
29     float hold_inches1;
30     float hold_inches2;
31     int hold_temp_num;
32
33     hold_temp_num = *num_temp_secs;
34     hold_inches2 = total_inches;
35
36
37     for (i=0; i<num_of_sizes; i++) {
38         for (j=i+1; j<num_of_sizes; j++) {
39             for (k=j+1; k<num_of_sizes; k++) {
40                 for (l=k+1; l<num_of_sizes; l++) {
41                     for (m=l+1; m<num_of_sizes; m++) {
42
43                         for (n=0; n<num_of_sizes; n++)
44                             temp_order[n] = 0;
45
46                         if ((set_s[i] == 1) && (set_s[j] == 1) &&
47                             (set_s[k] == 1) && (set_s[l] == 1) &&
48                             (set_s[m] == 1)) {
49                             temp_order[i] = 1;
50                             temp_order[j] = 1;
51                             temp_order[k] = 1;
52                             temp_order[l] = 1;
53                             temp_order[m] = 1;
54                             inches = combine_inches(temp_order);
55                             if (inches != (float) 0.0) {
56                                 for (n=0; n< num_of_sizes; n++)
57                                     temp_secs[*num_temp_secs].sizes[n] = 0;

```



```

58         total_inches = total_inches + inches;
59         temp_secs[*num_temp_secs].sizes[i] = 1;
60         temp_secs[*num_temp_secs].sizes[j] = 1;
61         temp_secs[*num_temp_secs].sizes[k] = 1;
62         temp_secs[*num_temp_secs].sizes[l] = 1;
63         temp_secs[*num_temp_secs].sizes[m] = 1;
64         ++*num_temp_secs;
65     }
66     temp_order[i] = 0;
67     temp_order[j] = 0;
68     temp_order[k] = 0;
69     temp_order[l] = 0;
70     temp_order[m] = 0;
71
72     for (n=0; n<num_of_sizes; n++) {
73         if ((n != i) && (n != j) && (n != k) &&
74             (n != l) && (n != m) && (set_s[n] == 1)) {
75             temp_order[n] = 1;
76         }
77     }
78
79     hold_inches1 = total_inches;
80     ones(temp_order, temp_secs, num_temp_secs);
81     check_inches(temp_secs, num_temp_secs);
82
83     for (n=0; n<num_of_sizes; n++) {
84         if ((n != i) && (n != j) && (n != k) &&
85             (n != l) && (set_s[n] == 1)) {
86             --*num_temp_secs;
87         }
88     }
89
90
91     total_inches = hold_inches1;
92     twos(temp_order, temp_secs, num_temp_secs);
93
94     total_inches = hold_inches1;
95     threes(temp_order, temp_secs, num_temp_secs);
96
97     total_inches = hold_inches1;
98     fours(temp_order, temp_secs, num_temp_secs);
99
100    total_inches = hold_inches1;
101    fives(temp_order, temp_secs, num_temp_secs);
102
103    *num_temp_secs = hold_temp_num;
104    total_inches = hold_inches2;
105
106    }
107 }
108 }
109 }
110 }
111 }
112 }
113

```

```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/fours.c   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : fours.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Recursive procedure to group units in fours
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "cherdec.h"
19 #include "cherlcl.h"
20
21 void fours(set_s, temp_secs, num_temp_secs)
22
23     order_t set_s;
24     section_t *temp_secs;
25     int *num_temp_secs;
26 {
27     float inches;
28     int j, i, k, l, m;
29     order_t temp_order;
30     float hold_inches1;
31     float hold_inches2;
32     int hold_temp_num;
33
34     hold_temp_num = *num_temp_secs;
35     hold_inches2 = total_inches;
36
37
38     for (i=0; i<num_of_sizes; i++) {
39         for (j=i+1; j<num_of_sizes; j++) {
40             for (k=j+1; k<num_of_sizes; k++) {
41                 for (l=k+1; l<num_of_sizes; l++) {
42
43                     for (m=0; m<num_of_sizes; m++)
44                         temp_order[m] = 0;
45
46                     if ((set_s[i] == 1) && (set_s[j] == 1) &&
47                         (set_s[k] == 1) && (set_s[l] == 1)) {
48                         temp_order[i] = 1;
49                         temp_order[j] = 1;
50                         temp_order[k] = 1;
51                         temp_order[l] = 1;
52                         inches = combine_inches(temp_order);
53                         if (inches != (float) 0.0) {
54                             for (m=0; m< num_of_sizes; m++)
55                                 temp_secs[*num_temp_secs].sizes[m] = 0;
56                             total_inches = total_inches + inches;
57                             temp_secs[*num_temp_secs].sizes[i] = 1;

```

```

58         temp_secs[*num_temp_secs].sizes[j] = 1;
59         temp_secs[*num_temp_secs].sizes[k] = 1;
60         temp_secs[*num_temp_secs].sizes[l] = 1;
61         ++*num_temp_secs;
62     }
63     temp_order[i] = 0;
64     temp_order[j] = 0;
65     temp_order[k] = 0;
66     temp_order[l] = 0;
67
68     for (m=0; m<num_of_sizes; m++) {
69         if ((m != i) && (m != j) && (m != k) &&
70             (m != l) && (set_s[m] == 1)) {
71             temp_order[m] = 1;
72         }
73     }
74
75     hold_inches1 = total_inches;
76     ones(temp_order, temp_secs, num_temp_secs);
77     check_inches(temp_secs, num_temp_secs);
78
79     for (m=0; m<num_of_sizes; m++) {
80         if ((m != i) && (m != j) && (m != k) &&
81             (m != l) && (set_s[m] == 1)) {
82             --*num_temp_secs;
83         }
84     }
85
86
87     total_inches = hold_inches1;
88     twos(temp_order, temp_secs, num_temp_secs);
89
90     total_inches = hold_inches1;
91     threes(temp_order, temp_secs, num_temp_secs);
92
93     total_inches = hold_inches1;
94     fours(temp_order, temp_secs, num_temp_secs);
95
96     *num_temp_secs = hold_temp_num;
97     total_inches = hold_inches2;
98
99     }
100 }
101 }
102 }
103 }
104 }
105

```

```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/getparm.c    December 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Getparm.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : December 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To read in the parameters from a file
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "cherdec.h"
19 #include "chericl.h"
20
21 int get_parameters(ou_units, max_ply, max_sizes)
22
23     int *ou_units;
24     int *max_ply;
25     int *max_sizes;
26
27 {
28     int i, j;
29     FILE *fp = NULL;
30     int quantity;
31     int m;
32     float temp;
33
34     if ((fp = fopen("INPUT", "r")) == NULL) {
35         printf("Cannot open input file - getparm.c");
36         exit(0);
37     }
38
39
40     /* set order and list values to -1 */
41     for (i = 0; i < MAX_SIZES; i++) {
42         order.number[i] = 0;
43         order.ch_sizes[i][0] = 0;
44     }
45
46     for (i=0; i<MAX_LIST; i++) {
47         list[i].inches = (float) 0.0;
48
49         for (j = 0; j < MAX_SIZES; j++)
50             list[i].sizes[j] = 0;
51     }
52
53     /* Input Units */
54     fscanf(fp,"%d", ou_units);
55     fscanf(fp,"%d", max_ply);
56     fscanf(fp,"%d", max_sizes);
57

```

```

58
59  /* Input Order */
60  for (i = 0; i < MAX_SIZES; i++) {
61      fscanf(fp,"%hd", &order.number[i]);
62      if (order.number[i] == -1) {
63          order.number[i] = 0;
64          break;
65      }
66
67      fscanf(fp,"%s", order.ch_sizes[i]);
68  }
69
70  num_of_sizes = i;
71
72
73  /* Input List */
74  i=0;
75  while(1) {
76
77      fscanf(fp,"%d", &quantity);
78
79      if (quantity == -2)
80          break;
81
82      while (quantity != -1) {
83
84          fscanf(fp,"%d", &m);
85
86          if (m >= num_of_sizes) {
87              printf("ERROR in reading size variable - getparm.c");
88              exit(0);
89          }
90
91          list[i].sizes[m] = quantity;
92
93          fscanf(fp,"%d", &quantity);
94      }
95
96      fscanf(fp,"%f", &list[i].inches);
97
98      ++i;
99  }
100
101  fclose(fp);
102
103  return(i);
104  }

```

```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/globals.h   January 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : Globals.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To declare all global variables
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include "cherdec.h"
17 #include "cherlcl.h"
18
19     ord_var_t order;
20
21     list_t    *list = NULL;
22
23     int      num_of_sizes;
24
25     int      num_list;
26
27     order_t temp_order;
28
29     section_t *sections = NULL;
30
31     int num_sections;
32
33     float total_inches = (float) 0.0;
34
35     float prev_inch = (float) 9999.0;
36
37     int num_hold_secs;
38
39     section_t *hold_secs;
40
41     int ply_height;

```

```

1 INCLUDES = cherdec.h chercl.h
2 LIBNAME = cherlib
3
4
5 OBJS = \
6     globals.obj \
7     getparm.obj \
8     findinch.obj \
9     combine.obj \
10    ones.obj \
11    chkinch.obj \
12    cphold.obj \
13    clrtemp.obj \
14    twos.obj \
15    threes.obj \
16    fours.obj \
17    fives.obj \
18    sixes.obj
19
20
21 .c.obj:
22     $(CC)
23     $(LIB)
24
25 globals.obj : globals.c $(INCLUDES)
26
27 getparm.obj : getparm.c $(INCLUDES)
28
29 findinch.obj : findinch.c $(INCLUDES)
30
31 combine.obj : combine.c $(INCLUDES)
32
33 ones.obj : ones.c $(INCLUDES)
34
35 twos.obj : twos.c $(INCLUDES)
36
37 threes.obj : threes.c $(INCLUDES)
38
39 fours.obj : fours.c $(INCLUDES)
40
41 fives.obj : fives.c $(INCLUDES)
42
43 sixes.obj : sixes.c $(INCLUDES)
44
45 chkinch.obj : chkinch.c $(INCLUDES)
46
47 cphold.obj : cphold.c $(INCLUDES)
48
49 clrtemp.obj : clrtemp.c $(INCLUDES)
50
51 cherry.obj : cherry.c $(INCLUDES)
52
53 cherry.exe : cher.y.obj $(OBJS)
54     cl cherry /link cherlib.lib
55
56
57 $(B)\cherry.exe : cherry.exe

```

58 \$(CP)
59
60 \$(I)\cherdec.h : cherdec.h
61 \$(CP)
62
63
64
65
66


```

1  /* -----
2  -- $Header:: D:/cops/src/cherry/ones.c   January 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : Ones.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Groups units in ones and find inches
12 -
13 -
14 - MODIFICATION HISTORY-
15 -
16 -----*/
17 #include <stdio.h>
18 #include <stdlib.h>
19 #include <string.h>
20 #include "cherdec.h"
21 #include "cherlcl.h"
22
23 void ones(set_s, temp_secs, num_temp_secs)
24
25     order_t set_s;
26     section_t *temp_secs;
27     int *num_temp_secs;
28 {
29     float inches;
30     int j, i, m;
31
32     j = *num_temp_secs;
33
34     for (i=0; i<num_of_sizes; i++) {
35         if (set_s[i] == 1) {
36
37             for (m=0; m<num_of_sizes; m++)
38                 temp_secs[j].sizes[m] = 0;
39
40             temp_secs[j].sizes[i] = 1;
41             inches = find_inches(temp_secs[j].sizes);
42             if (inches != (float) 0.0) {
43                 total_inches = total_inches + inches;
44                 ++j;
45             }
46             else
47                 temp_secs[j].sizes[i] = 0;
48         }
49     }
50
51     *num_temp_secs = j;
52
53 }
54

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/sixes.c   January 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : sixes.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN   : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Recursive procedure to group units in sixes.
12 -
13 - -----*/
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include "cherdec.h"
18 #include "chericl.h"
19
20 void sixes(set_s, temp_secs, num_temp_secs)
21
22     order_t set_s;
23     section_t *temp_secs;
24     int *num_temp_secs;
25 {
26     float inches;
27     int j, i, k, l, m, n, o;
28     order_t temp_order;
29     float hold_inches1;
30     float hold_inches2;
31     int hold_temp_num;
32
33     hold_temp_num = *num_temp_secs;
34     hold_inches2 = total_inches;
35
36
37     for (i=0; i<num_of_sizes; i++) {
38         for (j=i+1; j<num_of_sizes; j++) {
39             for (k=j+1; k<num_of_sizes; k++) {
40                 for (l=k+1; l<num_of_sizes; l++) {
41                     for (m=l+1; m<num_of_sizes; m++) {
42                         for (n=m+1; n<num_of_sizes; n++) {
43
44                             for (o=0; o<num_of_sizes; o++)
45                                 temp_order[o] = 0;
46
47                             if ((set_s[i] == 1) && (set_s[j] == 1) &&
48                                 (set_s[k] == 1) && (set_s[l] == 1) &&
49                                 (set_s[m] == 1) && (set_s[n] == 1)) {
50                                 temp_order[i] = 1;
51                                 temp_order[j] = 1;
52                                 temp_order[k] = 1;
53                                 temp_order[l] = 1;
54                                 temp_order[m] = 1;
55                                 temp_order[n] = 1;
56                                 inches = combine_inches(temp_order);
57                                 if (inches != (float) 0.0) {

```

```

58         for (o=0; o< num_of_sizes; o++)
59             temp_secs[*num_temp_secs].sizes[o] = 0;
60         total_inches = total_inches + inches;
61         temp_secs[*num_temp_secs].sizes[i] = 1;
62         temp_secs[*num_temp_secs].sizes[j] = 1;
63         temp_secs[*num_temp_secs].sizes[k] = 1;
64         temp_secs[*num_temp_secs].sizes[l] = 1;
65         temp_secs[*num_temp_secs].sizes[m] = 1;
66         temp_secs[*num_temp_secs].sizes[n] = 1;
67         ++*num_temp_secs;
68     }
69     temp_order[i] = 0;
70     temp_order[j] = 0;
71     temp_order[k] = 0;
72     temp_order[l] = 0;
73     temp_order[m] = 0;
74     temp_order[n] = 0;
75
76     for (o=0; o<num_of_sizes; o++) {
77         if ((o != i) && (o != j) && (o != k) &&
78             (o != l) && (o != m) && (o != n) && (set_s[o] == 1)) {
79             temp_order[o] = 1;
80         }
81     }
82
83     hold_inches1 = total_inches;
84     ones(temp_order, temp_secs, num_temp_secs);
85     check_inches(temp_secs, num_temp_secs);
86
87     for (o=0; o<num_of_sizes; o++) {
88         if ((o != i) && (o != j) && (o != k) &&
89             (o != l) && (o != n) && (set_s[o] == 1)) {
90             --*num_temp_secs;
91         }
92     }
93
94
95     total_inches = hold_inches1;
96     twos(temp_order, temp_secs, num_temp_secs);
97
98     total_inches = hold_inches1;
99     threes(temp_order, temp_secs, num_temp_secs);
100
101     total_inches = hold_inches1;
102     fours(temp_order, temp_secs, num_temp_secs);
103
104     total_inches = hold_inches1;
105     fives(temp_order, temp_secs, num_temp_secs);
106
107     total_inches = hold_inches1;
108     sixes(temp_order, temp_secs, num_temp_secs);
109
110     *num_temp_secs = hold_temp_num;
111     total_inches = hold_inches2;
112
113     }
114

```

115
116
117
118
119
120
121

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/threes.c   January 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : Threes.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN   : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Recursive procedure to group units in threes
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "cherdec.h"
19 #include "chericl.h"
20
21 void threes(set_s, temp_secs, num_temp_secs)
22
23     order_t set_s;
24     section_t *temp_secs;
25     int *num_temp_secs;
26 {
27     float inches;
28     int j, i, k, l;
29     order_t temp_order;
30     float hold_inches1;
31     float hold_inches2;
32     int hold_temp_num;
33
34     hold_temp_num = *num_temp_secs;
35     hold_inches2 = total_inches;
36
37     for (i=0; i<num_of_sizes; i++) {
38         for (j=i+1; j<num_of_sizes; j++) {
39             for (k=j+1; k<num_of_sizes; k++) {
40
41                 for (l=0; l<num_of_sizes; l++)
42                     temp_order[l] = 0;
43
44                 if ((set_s[i] == 1) && (set_s[j] == 1) && (set_s[k] == 1)) {
45                     temp_order[i] = 1;
46                     temp_order[j] = 1;
47                     temp_order[k] = 1;
48                     inches = combine_inches(temp_order);
49                     if (inches != (float) 0.0) {
50                         for (l=0; l< num_of_sizes; l++)
51                             temp_secs[*num_temp_secs].sizes[l] = 0;
52                         total_inches = total_inches + inches;
53                         temp_secs[*num_temp_secs].sizes[i] = 1;
54                         temp_secs[*num_temp_secs].sizes[j] = 1;
55                         temp_secs[*num_temp_secs].sizes[k] = 1;
56                         ++*num_temp_secs;
57                     }

```

```

58     temp_order[i] = 0;
59     temp_order[j] = 0;
60     temp_order[k] = 0;
61
62     for (l=0; l<num_of_sizes; l++) {
63         if ((l != i) && (l != j) && (l != k) && (set_s[l] == 1)) {
64             temp_order[l] = 1;
65         }
66     }
67
68     hold_inches1 = total_inches;
69     ones(temp_order, temp_secs, num_temp_secs);
70     check_inches(temp_secs, num_temp_secs);
71
72     for (l=0; l<num_of_sizes; l++) {
73         if ((l != i) && (l != j) && (l != k) && (set_s[l] == 1)) {
74             --*num_temp_secs;
75         }
76     }
77
78
79     total_inches = hold_inches1;
80     twos(temp_order, temp_secs, num_temp_secs);
81
82
83     total_inches = hold_inches1;
84
85     /*         for (l=0; l<num_of_sizes; l++) {
86                 if ((l != i) && (l != j) && (l != k) && (set_s[l] == 1)) {
87                     --*num_temp_secs;
88                 }
89             */
90
91     total_inches = hold_inches1;
92     threes(temp_order, temp_secs, num_temp_secs);
93
94     *num_temp_secs = hold_temp_num;
95     total_inches = hold_inches2;
96
97     }
98     }
99     }
100 }
101
102

```

```

1  /* -----
2  -- $Header::  D:/cops/src/cherry/twos.c   January 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : Twos.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : January 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- Recursive procedure to group units in twos
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include "cherdec.h"
19 #include "cherlcl.h"
20
21 void twos(set_s, temp_secs, num_temp_secs)
22
23     order_t set_s;
24     section_t *temp_secs;
25     int *num_temp_secs;
26 {
27     float inches;
28     int j, i, k, m;
29     order_t temp_order;
30     float hold_inches1;
31     float hold_inches2;
32     int hold_temp_num;
33
34     hold_temp_num = *num_temp_secs;
35     hold_inches2 = total_inches;
36
37
38     for (i=0; i<num_of_sizes; i++) {
39         for (j=i+1; j<num_of_sizes; j++) {
40
41             for (k=0; k<num_of_sizes; k++)
42                 temp_order[k] = 0;
43
44             if ((set_s[i] == 1) && (set_s[j] == 1)) {
45                 temp_order[i] = 1;
46                 temp_order[j] = 1;
47                 inches = combine_inches(temp_order);
48                 if (inches != (float) 0.0) {
49                     for(m=0; m<num_of_sizes; m++)
50                         temp_secs[*num_temp_secs].sizes[m] = 0;
51                     total_inches = total_inches + inches;
52                     temp_secs[*num_temp_secs].sizes[i] = 1;
53                     temp_secs[*num_temp_secs].sizes[j] = 1;
54                     ++*num_temp_secs;
55                 }
56                 printf(" WITH TOTAL = %d\n", total_inches);
57                 temp_order[i] = 0;

```

```

58         temp_order[j] = 0;
59
60         for (k=0; k<num_of_sizes; k++) {
61             if ((k != i) && (k != j) && (set_s[k] == 1)) {
62                 temp_order[k] = 1;
63             }
64         }
65
66         hold_inches1 = total_inches;
67         ones(temp_order, temp_secs, num_temp_secs);
68         check_inches(temp_secs, num_temp_secs);
69
70         for (k=0; k<num_of_sizes; k++) {
71             if ((k != i) && (k != j) && (set_s[k] == 1)) {
72                 --*num_temp_secs;
73             }
74         }
75
76         total_inches = hold_inches1;
77         twos(temp_order, temp_secs, num_temp_secs);
78         *num_temp_secs = hold_temp_num;
79         total_inches = hold_inches2;
80
81     }
82 }
83
84 }
85
86 }
87

```


Improvement Algorithm Source Code

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/case_ai.c   February 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : case_ai.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : February 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To determine savings if sizes in two sections are
12 -           the same
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include "impdec.h"
19 #include "implcl.h"
20
21 float case_ai(sect1, portion, cut_cost)
22
23     section_t *sect1;
24     section_t *portion;
25     int      cut_cost;
26
27 (
28     int i;
29     int e = 0;
30     float savings;
31
32     for (i=0; i< num_of_sizes; i++) {
33         e = e + (order.perimeter[i] * sect1->sizes[i]);
34         e = e + (order.perimeter[i] * portion->sizes[i]);
35     }
36
37     savings = (float) cut_cost * e;
38
39     return(savings);
40
41 )
42

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/case_ail.c    February 1991
3  -- -----*/
4
5  /*-----
6  -  FILE NAME    : case_ail.c
7  -  PROGRAMMER   : Terri A. Smith
8  -  DATE WRITTEN : February 1991
9  -  ADDRESS      : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 -  PURPOSE- To determine savings by lying sizes next to each
12 -             other instead of on top.
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include "impec.h"
19 #include "implcl.h"
20
21 float case_ail(i, l, unit_cost)
22
23     int i;
24     int l;
25     int unit_cost;
26
27 {
28     float savings = (float) 0.0;
29     float sect1_inch;
30     float sect2_inch;
31     float sect3_inch;
32     float sect4_inch;
33
34     sect1_inch = find_inches(in_section[i].sizes);
35     sect2_inch = find_inches(in_section[l].sizes);
36     sect3_inch = find_inches(sect3.sizes);
37     sect4_inch = find_inches(sect4.sizes);
38
39     savings = unit_cost * in_section[i].ply_height * (sect1_inch + sect2_inch -
40                                                         sect3_inch - sect4_inch);
41
42     return(savings);
43
44 }
45

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/combply.c    February 1990
3  ----- */
4
5  /*-----
6  -  FILE NAME      : Combply.c
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN  : April 1990
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- This procedure combines sections which have the same sizes
12 -             and the ply height of the new section does not exceed the max_ply
13 -
14 -
15 ----- */
16 #include <stdio.h>
17 #include <malloc.h>
18 #include <stdlib.h>
19 #include <memory.h>
20 #include "impdec.h"
21 #include "implcl.h"
22
23 void combine_ply(max_ply)
24     int max_ply;
25
26 {
27
28     int i,j,l,k;
29     char match;
30
31     for (i=0; i<num_in_sec; i++) {
32         for (j=i+1; j<num_in_sec; j++) {
33             match = 1;
34             for (k=0; k<num_of_sizes; k++) {
35                 if (in_section[i].sizes[k] != in_section[j].sizes[k])
36                     match = 0;
37             }
38             if ((match) &&
39                 ((in_section[i].ply_height + in_section[j].ply_height) <= max_ply)) {
40                 in_section[i].ply_height = in_section[i].ply_height + in_section[j].ply_height;
41                 for (l=j; l<num_in_sec-1; l++)
42                     memcpy(&in_section[l+1], &in_section[l], sizeof(section_t));
43                 --j;
44                 --num_in_sec;
45             }
46         }
47     }
48
49     num_temp_sec = num_in_sec;
50
51     return;
52 }
53

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/combsize.c   February 1990
3  -- -----*/
4
5  /*-----
6  -  FILE NAME      : Combsize.c
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN  : April 1990
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- This procedure combines all sections in which the
12 -            the number of sizes in the new section does not excede
13 -            the max_sizes allowed per section
14 -
15 - -----*/
16
17 #include <stdio.h>
18 #include <malloc.h>
19 #include <stdlib.h>
20 #include <memory.h>
21 #include "impdec.h"
22 #include "implcl.h"
23
24
25 void combine_sizes(max_sizes)
26     int max_sizes;
27
28 {
29     int num_units;
30     int i, j, l;
31
32     for (i=0; i<num_in_sec; i++) {
33         for (j=i+1; j<num_in_sec; j++) {
34             num_units = 0;
35             for (l=0; l<num_of_sizes; l++)
36                 num_units = num_units + in_section[i].sizes[l] +
37                                     in_section[j].sizes[l];
38
39             if ((num_units <= max_sizes) &&
40                 (in_section[i].ply_height == in_section[j].ply_height)) {
41                 for (l=0; l<num_of_sizes; l++)
42                     in_section[i].sizes[l] = in_section[i].sizes[l] +
43                                             in_section[j].sizes[l];
44
45                 for (l=j; l<num_in_sec-1; l++)
46                     memcpy(&in_section[l], &in_section[l+1], sizeof(section_t));
47
48                 --j;
49                 --num_in_sec;
50             }
51         }
52     }
53
54     num_temp_sec = num_in_sec;
55
56     return;
57 }

```

58
59

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/compswap.c   February 1991
3  -----*/
4
5  /*-----
6  - FILE NAME      : compswap.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : February 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To determine which method to ue to compute
12 -           the savings
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include "impdec.h"
18 #include "implcl.h"
19
20 float compute_swap_savings(i, l, cut_cost, unit_cost, max_sizes)
21
22     int i;
23     int l;
24     int cut_cost;
25     int unit_cost;
26     int max_sizes;
27
28 {
29     float savings;
30
31     if (in_section[i].ply_height == in_section[l].ply_height) {
32         savings = case_a11(i, l, unit_cost);
33         temp_save.type= 3;
34         temp_save.cand_ply_height = in_section[i].ply_height;
35         temp_save.org_ply_height = in_section[i].ply_height;
36     }
37
38     else {
39         temp_save.cand_ply_height = in_section[l].ply_height;
40         temp_save.org_ply_height = in_section[i].ply_height;
41         savings = case_a11(i, l, unit_cost);
42         temp_save.type= 4;
43     }
44
45     temp_save.savings = savings;
46
47     return(savings);
48
49 }
50

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/compute.c    February 1991
3  -- -----*/
4
5  /*-----
6  -  FILE NAME      : compute.c
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN  : February 1991
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332  (404) 894-8952
10 -
11 -  PURPOSE- To determine which method to ue to compute
12 -             the savings
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include "impdec.h"
18 #include "implcl.h"
19
20 float compute_savings(i, l, cut_cost, unit_cost, max_sizes)
21
22     int i;
23     int l;
24     int cut_cost;
25     int unit_cost;
26     int max_sizes;
27
28 {
29     int j;
30     float savings = (float) 0.0;
31     float save2;
32     char match = 1;
33     int num_units = 0;
34
35     for (j=0; j<num_of_sizes; j++) {
36         if (portion.sizes[j] != in_section[l].sizes[j])
37             match = 0;
38         num_units = num_units + sect4.sizes[j];
39     }
40
41
42     if (match) { /* sizes in sections are the same */
43         if (num_units <= max_sizes) {
44             save2 = case_all(i, l, unit_cost);
45
46             if (save2 > savings) {
47                 temp_save.type= 2;
48                 savings = save2;
49                 if (in_section[i].ply_height != in_section[l].ply_height)
50                     temp_save.cand_ply_height = in_section[l].ply_height;
51                 else temp_save.cand_ply_height = in_section[i].ply_height;
52                 temp_save.org_ply_height = in_section[i].ply_height;
53             }
54         }
55     }
56
57     else if ((in_section[i].ply_height == in_section[l].ply_height) && (num_units <= max_sizes)) {

```



```

58         savings = case_aii(i, l, unit_cost);
59         temp_save.type= 3;
60         temp_save.cand_ply_height = in_section[i].ply_height;
61         temp_save.org_ply_height = in_section[i].ply_height;
62     }
63
64     else if (num_units <= max_sizes) {
65         if (in_section[i].ply_height != in_section[l].ply_height)
66             temp_save.cand_ply_height = in_section[l].ply_height;
67         else temp_save.cand_ply_height = in_section[i].ply_height;
68         temp_save.org_ply_height = in_section[i].ply_height;
69
70         savings = case_aii(i, l, unit_cost);
71         temp_save.type= 4;
72     }
73
74     temp_save.savings = savings;
75
76     return(savings);
77 }
78
79

```

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/findinch.c   February 1991
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Findinch.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : February 1991
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To determine the number of inches in a section based
12 -           on the input list is
13 -
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <string.h>
19 #include "impdec.h"
20 #include "implcl.h"
21
22 float find_inches(sizes)
23
24     order_t sizes;
25
26 {
27     int i, j;
28     char match = 0;
29     char empty = 0;
30
31     i = 0;
32     while ((!match) && (i < num_list)) {
33         empty = 1;
34         match = 1;
35         for (j=0; j<num_of_sizes; j++) {
36             if (sizes[j] != list[i].sizes[j])
37                 match = 0;
38             if (sizes[j] != 0)
39                 empty = 0;
40         }
41         ++i;
42     }
43
44     if (empty)
45         return((float) 0.0);
46
47     if (match)
48         return(list[--i].inches);
49     else {
50         printf(" COULDNT FIND ");
51         for (i=0; i<num_of_sizes; i++) {
52             if (sizes[i] > 0)
53                 printf("%d %s ", sizes[i], order.ch_sizes[i]);
54             }
55         printf("\n");
56         exit(0);
57     }

```

58
59

)

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/getparm.c    February 1990
3  -----*/
4
5  /*-----
6  - FILE NAME      : Getparm.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN   : February 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To read the input parameters from a file
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <string.h>
18 #include <malloc.h>
19 #include "impdec.h"
20 #include "implcl.h"
21
22 int get_parameters(ou_units, max_ply, max_sizes,
23                  cut_cost, unit_cost, old_ou_units)
24
25     int *ou_units;
26     int *max_ply;
27     int *max_sizes;
28     int *cut_cost;
29     int *unit_cost;
30     int *old_ou_units;
31
32 {
33     int i, j;
34     FILE *fp = NULL;
35     int quantity;
36     int m;
37
38     if ((fp = fopen("INPUT", "r")) == NULL) {
39         printf("Cannot open input file - getparm.c");
40         exit(0);
41     }
42
43     /* set order and list values to -1 */
44     for (i = 0; i < MAX_SIZES; i++) {
45         order.number[i] = 0;
46         order.ch_sizes[i][0] = 0;
47         order.perimeter[i] = 0;
48     }
49
50     for (i=0; i<MAX_LIST; i++) {
51         list[i].inches = (float) 0.0;
52
53         for (j = 0; j < MAX_SIZES; j++)
54             list[i].sizes[j] = 0;
55     }
56
57

```

```

58     fscanf(fp,"%d", ou_units);
59     fscanf(fp,"%d", max_ply);
60     fscanf(fp,"%d", max_sizes);
61     fscanf(fp,"%d", cut_cost);
62     fscanf(fp,"%d", unit_cost);
63
64
65     /* Input Order */
66     for (i = 0; i < MAX_SIZES; i++) {
67         fscanf(fp,"%d", &order.number[i]);
68         if (order.number[i] == -1) {
69             order.number[i] == 0;
70             break;
71         }
72
73         fscanf(fp,"%d", &order.perimeter[i]);
74         fscanf(fp,"%s", order.ch_sizes[i]);
75     }
76
77     num_of_sizes = i;
78
79     fscanf(fp,"%d", &num_in_sec);
80
81     if ((in_section = (section_t *)malloc(num_in_sec * sizeof(section_t))) == NULL) {
82         printf("ALLOCATION ERROR - SECTIONS  getparm.c\n");
83         exit(0);
84     }
85
86     for (i=0; i<num_in_sec; i++) {
87         in_section[i].ply_height = 0;
88         for (m=0; m<num_of_sizes; m++) {
89             in_section[i].sizes[m] = 0;
90         }
91     }
92
93     i = 0;
94     /* Input Sections */
95     while(i < num_in_sec) {
96
97         fscanf(fp,"%d", &quantity);
98
99         while (quantity != -1) {
100
101             fscanf(fp,"%d", &m);
102
103             if (m >= num_of_sizes) {
104                 printf("ERROR in reading size variable - getparm.c");
105                 exit(0);
106             }
107
108             in_section[i].sizes[m] = quantity;
109
110             fscanf(fp,"%d", &quantity);
111         }
112         fscanf(fp, "%d", &in_section[i].ply_height);
113
114         ++i;

```

```

115     }
116
117     fscanf(fp,"%d", old_ou_units);
118
119
120     /* Input List */
121     i=0;
122     while(1) {
123
124         fscanf(fp,"%d", &quantity);
125
126         if (quantity == -2)
127             break;
128
129         while (quantity != -1) {
130
131             fscanf(fp,"%d", &m);
132
133             if (m >= num_of_sizes) {
134                 printf("ERROR in reading size variable - getparm.c");
135                 exit(0);
136             }
137
138             list[i].sizes[m] = quantity;
139
140             fscanf(fp,"%d", &quantity);
141         }
142
143         fscanf(fp,"%f", &list[i].inches);
144
145         ++i;
146     }
147
148     fclose(fp);
149
150     return(i);
151 }
152

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/globals.h    February 1991
3  -----*/
4
5  /*-----
6  -  FILE NAME    : Globals.h
7  -  PROGRAMMER   : Terri A. Smith
8  -  DATE WRITTEN : February 1991
9  -  ADDRESS      : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 -  PURPOSE- To declare all global variables
12 -
13 -
14 -----*/
15 #include <stdio.h>
16 #include "impdec.h"
17 #include "implcl.h"
18
19     ord_var_t order;
20
21     list_t    *list = NULL;
22
23     int      num_of_sizes;
24
25     int      num_list;
26
27     section_t *in_section = NULL;
28
29     int      num_in_sec;
30
31     int      num_temp_sec;
32
33     section_t sect3;
34
35     section_t sect4;
36
37     section_t portion;
38
39     savings_t temp_save;
40
41     savings_t save;

```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/impdec.h    February 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : impdec.h
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : February 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332   (404) 894-8952
10 -
11 - PURPOSE- To define all structures and procedures
12 -
13 -
14 - -----*/
15 #ifndef IMPDEC_H
16 #define IMPDEC_H
17
18 #define MAX_LIST 1000
19 #define MAX_SIZES 25
20 #define MAX_SAVINGS 400
21
22
23 typedef int order_t[MAX_SIZES];
24
25 typedef char sizes_t[MAX_SIZES][10];
26
27 typedef struct {
28     order_t  number;
29     sizes_t  ch_sizes;
30     int      perimeter[MAX_SIZES];
31 } ord_var_t;
32
33 typedef struct {
34     order_t  sizes;
35     float    inches;
36 } list_t;
37
38 typedef struct {
39     order_t  sizes;
40     int      ply_height;
41     char     merged;
42 } section_t;
43
44 typedef struct {
45     int      sect1;
46     int      sect2;
47     int      org_ply_height;
48     int      cand_ply_height;
49     float    savings;
50     int      type;
51     order_t  org;
52     order_t  cand;
53     order_t  in_sect1;
54     order_t  in_sect2;
55     } savings_t;
56
57

```



```
58  int get_parameters(int *units, int *max_ply, int *max_sizes,  
59                      int *cut_cost, int *unit_cost, int* old_ou_units);  
60  
61  float find_inches(order_t sizes);  
62  
63  float case_a11(int i, int j, int unit_cost);  
64  
65  float compute_savings(int i, int j, int cut_cost, int unit_cost, int max_sizes);  
66  
67  float compute_swap_savings(int i, int j, int cut_cost, int unit_cost, int max_sizes);  
68  
69  void combine_ply(int max_ply);  
70  
71  void combine_sizes(int max_sizes);  
72  
73  void transfer_forward(int i, int j, int l,  
74                      int cut_cost, int unit_cost, int max_sizes, int max_ply);  
75  
76  void transfer_backwards(int i, int j, int l,  
77                        int cut_cost, int unit_cost, int max_sizes, int max_ply);  
78  
79  void swap_forward(int i, int j, int l,  
80                  int cut_cost, int unit_cost, int max_sizes, int max_ply);  
81  
82  void swap_backwards(int i, int j, int l,  
83                    int cut_cost, int unit_cost, int max_sizes, int max_ply);  
84  
85  #endif
```

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/Impdec.h    February 1990
3  -----*/
4
5  /*-----
6  - FILE NAME    : Implcl.h
7  - PROGRAMMER   : Terri A. Smith
8  - DATE WRITTEN : February 1990
9  - ADDRESS      : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- To define all global variables
12 -
13 -
14 -----*/
15 #ifndef IMPLCL_H
16 #define IMPLCL_H
17
18
19 extern ord_var_t order;
20 extern list_t *list;
21 extern int num_list;
22 extern int num_of_sizes;
23 extern section_t *in_section;
24 extern int num_in_sec;
25 extern int num_temp_sec;
26 extern section_t sect3;
27 extern section_t sect4;
28 extern section_t portion;
29 extern savings_t temp_save;
30 extern savings_t save;
31
32
33 #endif

```

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/improve.c    February 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Improve.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : February 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- The main program which controls flow of execution
12 -
13 -
14 - -----*/
15 #include <stdio.h>
16 #include <malloc.h>
17 #include <memory.h>
18 #include <stdlib.h>
19 #include <string.h>
20 #include <time.h>
21 #include "impec.h"
22 #include "implcl.h"
23
24 #define clock() time(NULL)
25
26 main(argv, argc)
27     int argv;
28     char *argv[];
29
30 {
31     /* Input Variables */
32     int  ou_units;          /* # of units over/under allowed */
33     int  old_ou_units;      /* # of units over/under allowed */
34     int  max_ply;           /* max ply height allowed */
35     int  max_sizes;         /* # of sizes allowed / section */
36     int  init_ply;          /* initial ply height */
37     int  cut_cost;          /* cutting cost */
38     int  unit_cost;         /* unit cost */
39
40     /* Output Variables */
41     float tot_length;       /* the total amt of fabric needed*/
42     float tot_marker;       /* total fabric between markers */
43     int  unit_dev;          /* deviation of units to cut from order */
44     int  unit_count;        /* units in all sections */
45     int  order_count;       /* # of units in order */
46     char unit_string[10];   /* OVER or UNDER */
47
48     int  i,j, k, l, r, s,m,n; /* counters */
49     int  total_order = 0;    /* total # in order */
50     float inches;           /* inches in sections * ply */
51     float marker;           /* inches between markers */
52     FILE *fp;               /* file pointer for output */
53     order_t temp_order;     /* temp order */
54     clock_t start_time;     /* used for timing alg */
55     clock_t end_time;       /* used for timing alg */
56     double total_time;      /* total execution time */
57     char  mergers_possible = 1; /* while loop boolean */

```

```

58     section_t temp_sec;           /* temporary section          */
59
60     start_time = clock();
61
62     if ((fp = fopen("OUTPUT", "w")) == NULL) {
63         printf("CANNOT OPEN OUTPUT FILE    savings.c\n");
64         exit(0);
65     }
66
67     if ((list = (list_t *)malloc(MAX_LIST * sizeof(list_t))) == NULL) {
68         printf("ALLOCATION ERROR FOR LIST    savings.c\n");
69         exit(0);
70     }
71
72     /*
73      Get parameters and print out first solution
74     */
75
76     num_list = get_parameters(&ou_units, &max_ply, &max_sizes,
77                             &cut_cost, &unit_cost, &old_ou_units);
78
79     tot_length = (float) 0.0;
80     tot_marker = (float) 0.0;
81     fprintf(fp, "MAX PLY = %d MAX # OF UNITS PER SECTION = %d\n", max_ply, max_sizes);
82     fprintf(fp, "UNIT COST = %d cents CUT COST = %d cents\n", unit_cost, cut_cost);
83     fprintf(fp, "ORDER\n");
84     for (i=0; i<num_of_sizes; i++) {
85         fprintf(fp, "%d SIZE %s\n", order.number[i], order.ch_sizes[i]);
86     }
87
88     fprintf(fp, "\n FIRST SOLUTION \n");
89     for (i=0; i<num_in_sec; i++) {
90         fprintf(fp, "SECTION %d HAS PLY = %d\n", i, in_section[i].ply_height);
91         for (j=0; j<num_of_sizes; j++) {
92             if (in_section[i].sizes[j] > 0) {
93                 fprintf(fp, "        AND %d SIZE %s\n", in_section[i].sizes[j], order.ch_sizes[j]);
94             }
95         }
96         marker = find_inches(in_section[i].sizes);
97         inches = marker * in_section[i].ply_height;
98         fprintf(fp, "MARKER LENGTH = %7.2f TOTAL LENGTH = %7.2f\n\n", marker, inches);
99         tot_length = tot_length + inches;
100        tot_marker = tot_marker + marker;
101    }
102    fprintf(fp, "TOTAL MARKER = %7.2f TOTAL LENGTH = %7.2f\n\n", tot_marker, tot_length);
103
104    /*
105     Initialize savings structures
106    */
107    for (i=0; i<num_of_sizes; i++) {
108        save.org[i] = 0;
109        save.cand[i] = 0;
110        temp_save.org[i] = 0;
111        temp_save.cand[i] = 0;
112    }
113
114    /*

```

```

115     combine any sections with a combination of sizes <= max_sizes
116 */
117
118 combine_sizes(max_sizes);
119 /*
120     Main Loop of program -
121     The loop begins by trying to place one sizes form one section
122     into another section. Once all possible transferred are tested,
123     then the program tries swapping two sizes from two different
124     sections. The loop begins with the first section. The best
125     transfer or swap from this section is made and the next section
126     goes through the same tests etc. Once all sections have been
127     exhausted then the same is repeated but backwards (starting
128     with the last section. This whole process is repeated twice.
129 */
130 mergers_possible = 2;
131 while (mergers_possible > 0) {
132
133     /* combine any sections with same sizes by putting on
134     top of each other if it doesn't violate max ply height
135     */
136
137     combine_ply(max_ply);
138
139     /*
140     Attempt to reaassign one portion from original section
141     to a new section and calculate savings. Merge only
142     the one with the greatest savings
143     */
144     for (i=0; i<num_in_sec; i++) {
145         for (j=0; j<num_of_sizes; j++) {
146             save.sect1 = -1;
147             save.sect2 = -1;
148             save.type = 0;
149             save.org_ply_height = 0;
150             save.cand_ply_height = 0;
151             save.savings = (float) 0.0;
152
153             for (m=0; m<num_of_sizes; m++)
154                 portion.sizes[m] = 0;
155             portion.ply_height = 0;
156
157             for (l=i+1; l<num_in_sec; l++) {
158
159                 transfer_forward(i, j, l, cut_cost, unit_cost, max_sizes, max_ply);
160
161                 swap_forward(i, j, l, cut_cost, unit_cost, max_sizes, max_ply);
162
163             }
164
165             /*
166             Place portion into section. If the two sections have
167             different ply heights then the smallest ply height is
168             given to both sections and the section with the larger
169             ply height is added to the end of the section list with
170             a ply height equal to larger ply minus the smaller ply
171             */

```

```

172     r = save.sect1;
173     s = save.sect2;
174     if (save.savings != (float) 0.0) {
175         printf("REPLACING PORTION %d %d\n", r, s);
176
177         in_section[r].ply_height = save.org_ply_height;
178         in_section[s].ply_height = save.cand_ply_height;
179
180         if (save.org_ply_height < save.cand_ply_height) {
181             in_section[s].ply_height = save.org_ply_height;
182             temp_sec.ply_height = save.cand_ply_height -
183                 save.org_ply_height;
184
185             for(m=0; m<num_of_sizes; m++)
186                 temp_sec.sizes[m] = save.in_sect2[m];
187
188             if ((in_section = realloc(in_section, ((num_temp_sec + 1)
189                 * sizeof(section_t)))) == NULL) {
190                 printf("REALLOCATION ERROR FOR INSECTION    improve2.c");
191                 exit(0);
192             }
193
194             memcpy(&in_section[num_temp_sec++], &temp_sec, sizeof(section_t));
195         }
196
197         else if (save.org_ply_height > save.cand_ply_height) {
198             in_section[r].ply_height = save.cand_ply_height;
199             temp_sec.ply_height = save.org_ply_height -
200                 save.cand_ply_height;
201
202             for(m=0; m<num_of_sizes; m++)
203                 temp_sec.sizes[m] = save.in_sect1[m];
204
205             if ((in_section = realloc(in_section, ((num_temp_sec + 1)
206                 * sizeof(section_t)))) == NULL) {
207                 printf("REALLOCATION ERROR FOR INSECTION    improve2.c");
208                 exit(0);
209             }
210
211             memcpy(&in_section[num_temp_sec++], &temp_sec, sizeof(section_t));
212         }
213
214         for(m=0; m<num_of_sizes; m++) {
215             in_section[r].sizes[m] = save.org[m];
216             in_section[s].sizes[m] = save.cand[m];
217         }
218     }
219     } /* for j */
220 } /* for i */
221
222
223 /*
224 Perform the same sequence of events to transfer and swap
225 sizes but start at end of list and go backwards
226
227 Attempt to reaassign one portion from original section
228 to a new section and calculate savings. Merge only

```

```

229         the one with the greatest savings
230     */
231
232     num_in_sec = num_temp_sec;
233
234     for (i=num_in_sec-1; i>=0; i--) {
235         for (j=0; j<num_of_sizes; j++) {
236             save.sect1 = -1;
237             save.sect2 = -1;
238             save.type = 0;
239             save.org_ply_height = 0;
240             save.cand_ply_height = 0;
241             save.savings = (float) 0.0;
242
243             for (m=0; m<num_of_sizes; m++)
244                 portion.sizes[m] = 0;
245             portion.ply_height = 0;
246
247             for (l=i-1; l>=0; l--) {
248
249                 transfer_backwards(i, j, l, cut_cost, unit_cost, max_sizes, max_ply);
250
251                 swap_backwards(i, j, l, cut_cost, unit_cost, max_sizes, max_ply);
252             }
253
254             r = save.sect1;
255             s = save.sect2;
256             if (save.savings != (float) 0.0) {
257                 printf("REPLACING PORTION %d %d\n", r, s);
258
259                 in_section[r].ply_height = save.org_ply_height;
260                 in_section[s].ply_height = save.cand_ply_height;
261
262                 if (save.org_ply_height < save.cand_ply_height) {
263                     in_section[s].ply_height = save.org_ply_height;
264                     temp_sec.ply_height = save.cand_ply_height -
265                         save.org_ply_height;
266
267                     for(m=0; m<num_of_sizes; m++) {
268                         temp_sec.sizes[m] = save.in_sect2[m];
269                     }
270                     if ((in_section = realloc(in_section, ((num_temp_sec + 1)
271                         * sizeof(section_t)))) == NULL) {
272                         printf("REALLOCATION ERROR FOR INSECTION      improve2.c");
273                         exit(0);
274                     }
275
276                     memcpy(&in_section[num_temp_sec++], &temp_sec, sizeof(section_t));
277                 }
278                 else if (save.org_ply_height > save.cand_ply_height) {
279                     in_section[r].ply_height = save.cand_ply_height;
280                     temp_sec.ply_height = save.org_ply_height -
281                         save.cand_ply_height;
282
283                     for(m=0; m<num_of_sizes; m++) {
284                         temp_sec.sizes[m] = save.in_sect1[m];
285                     }

```

```

286
287         if ((in_section = realloc(in_section, ((num_temp_sec + 1)
288                                     * sizeof(section_t)))) == NULL) {
289             printf("REALLOCATION ERROR FOR INSECTION    improve2.c");
290             exit(0);
291         }
292         memcpy(&in_section[num_temp_sec++], &temp_sec, sizeof(section_t));
293     }
294
295     for(m=0; m<num_of_sizes; m++) {
296         in_section[r].sizes[m] = save.org[m];
297         in_section[s].sizes[m] = save.cand[m];
298     }
299 }
300 } /* for j */
301 } /* for i */
302
303
304     num_in_sec = num_temp_sec;
305     --mergers_possible;
306     /* while */
307
308     /*
309     Remove sections that are empty
310     */
311
312     for (i=0; i<num_in_sec; i++) {
313         order_count = 0;
314         for (j=0; j<num_of_sizes; j++) {
315             order_count = order_count + in_section[i].sizes[j];
316         }
317         if (order_count == 0) {
318             for (j=i; j<num_in_sec-1; j++) {
319                 memcpy(&in_section[j], &in_section[j+1], sizeof(section_t));
320             }
321             num_in_sec = num_in_sec - 1;
322         }
323     }
324
325     end_time = clock();
326     total_time = ((double) end_time - start_time) / CLK_TCK;
327
328     fprintf(fp, "\n\n*****\n\n");
329     tot_length = (float) 0.0;
330     tot_marker = (float) 0.0;
331     unit_dev = 0;
332     order_count = 0;
333     unit_count = 0;
334
335     fprintf(fp, "THE # OF FINAL SECTIONS ARE : %d\n", num_in_sec);
336     for (i=0; i<num_in_sec; i++) {
337         fprintf(fp, "SECTION %d HAS PLY = %d\n", i, in_section[i].ply_height);
338         for (j=0; j<num_of_sizes; j++) {
339             if (in_section[i].sizes[j] > 0) {
340                 fprintf(fp, "                AND %d SIZE %s\n", in_section[i].sizes[j], order.ch_sizes[j]);
341                 unit_count = unit_count + (in_section[i].sizes[j] * in_section[i].ply_height);
342             }

```



```

343     }
344     marker = find_inches(in_section[i].sizes);
345     inches = marker * in_section[i].ply_height;
346     fprintf(fp, "MARKER LENGTH = %7.2f  TOTAL LENGTH = %7.2f\n\n", marker, inches);
347     tot_length = tot_length + inches;
348     tot_marker = tot_marker + marker;
349 }
350
351 for (j=0; j<num_of_sizes; j++)
352     order_count = order_count + order.number[j];
353
354 unit_dev = order_count - unit_count;
355 if (unit_dev > 0)
356     strcpy(unit_string, "UNDER");
357 else if (unit_dev == 0)
358     strcpy(unit_string, "\0");
359 else {
360     unit_dev = unit_dev * -1;
361     strcpy(unit_string, "OVER");
362 }
363
364 fprintf(fp, "TOTAL MARKER = %7.2f TOTAL LENGTH = %7.2f\n\n", tot_marker, tot_length);
365 fprintf(fp, "UNIT OVER/UNDER = %d %s", unit_dev, unit_string);
366 fprintf(fp, "\n\nTOTAL TIME = %f\n", total_time);
367
368
369 if (list != NULL)
370     free(list);
371
372
373 fclose(fp);
374
375 return(0);
376 }

```

```

1  INCLUDES = impdec.h implcl.h
2  LIBNAME = implib
3
4
5  OBJS = \
6      globals.obj \
7      getparm.obj \
8      findinch.obj \
9      case_ail.obj \
10     compute.obj \
11     compswap.obj \
12     combsize.obj \
13     combply.obj \
14     tranfrwd.obj \
15     swapfrwd.obj \
16     tranbkwd.obj \
17     swapbkwd.obj
18
19
20  .c.obj:
21      $(CC)
22      $(LIB)
23
24
25  globals.obj : globals.c $(INCLUDES)
26
27  getparm.obj : getparm.c $(INCLUDES)
28
29  findinch.obj : findinch.c $(INCLUDES)
30
31  case_ail.obj : case_ail.c $(INCLUDES)
32
33  compute.obj : compute.c $(INCLUDES)
34
35  compswap.obj : compswap.c $(INCLUDES)
36
37  combply.obj : combply.c $(INCLUDES)
38
39  combsize.obj : combsize.c $(INCLUDES)
40
41  tranfrwd.obj : tranfrwd.c $(INCLUDES)
42
43  swapfrwd.obj : swapfrwd.c $(INCLUDES)
44
45  tranbkwd.obj : tranbkwd.c $(INCLUDES)
46
47  swapbkwd.obj : swapbkwd.c $(INCLUDES)
48
49  improve.obj : improve.c $(INCLUDES)
50
51  improve.exe : improve.obj $(OBJS)
52      cl improve /link implib.lib
53
54
55  $(B)\improve.exe : improve.exe
56      $(CP)
57

```

58 \$(I)\impdec.h : impdec.h
59 \$(CP)
60
61

```

1  /* -----
2  -- $Header::  D:/cops/src/improv/swapbkwd.c    February 1990
3  -- -----*/
4
5  /*-----
6  -  FILE NAME      : Swapkwd.c
7  -  PROGRAMMER     : Terri A. Smith
8  -  DATE WRITTEN  : April 1990
9  -  ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 -  PURPOSE- This procedure attempts to swap one size from one
12 -            with another size in a different section if feasible. It
13 -            works from the end of the section list to the start.
14 -
15 -
16 - -----*/
17 #include <stdio.h>
18 #include <malloc.h>
19 #include <memory.h>
20 #include <stdlib.h>
21 #include <string.h>
22 #include "impdec.h"
23 #include "implcl.h"
24
25 void swap_backwards(i, j, l, cut_cost, unit_cost, max_sizes, max_ply)
26     int i;
27     int j;
28     int l;
29     int cut_cost;
30     int unit_cost;
31     int max_sizes;
32     int max_ply;
33
34 {
35
36     int k, m, n;          /* counters */
37     int num_units;        /* num_units in one section */
38
39
40     for (n=0; n<num_of_sizes; n++) {
41         if ((in_section[i].sizes[j] > 0) &&
42             (in_section[l].sizes[n] > 0)) {
43
44             for (m=0; m<num_of_sizes; m++) {
45                 sect3.sizes[m] = in_section[i].sizes[m];
46                 sect4.sizes[m] = in_section[l].sizes[m];
47             }
48
49             sect3.sizes[j] = sect3.sizes[j] - 1;
50             sect3.sizes[n] = sect3.sizes[n] + 1;
51             sect4.sizes[j] = sect4.sizes[j] + 1;
52             sect4.sizes[n] = sect4.sizes[n] - 1;
53
54             temp_save.sect1 = i;
55             temp_save.sect2 = l;
56             temp_save.type = 0;
57             temp_save.org_ply_height = 0;

```

```

58     temp_save.cand_ply_height = 0;
59     temp_save.savings = (float) 0.0;
60
61     compute_swap_savings(i, l, cut_cost, unit_cost, max_sizes);
62
63     num_units = 0;
64     for (m=0; m<num_of_sizes; m++)
65         num_units = num_units + sect4.sizes[m];
66
67     if ((temp_save.savings > save.savings) &&
68         (num_units <= max_sizes) &&
69         (temp_save.type > 0) &&
70         (temp_save.cand_ply_height <= max_ply)) {
71         memcpy(&save, &temp_save, sizeof(savings_t));
72
73         for (m=0; m<num_of_sizes; m++) {
74             if (temp_save.type != 1) {
75                 save.org[m] = sect3.sizes[m];
76                 save.cand[m] = sect4.sizes[m];
77                 save.in_sect1[m] = in_section[i].sizes[m];
78                 save.in_sect2[m] = in_section[l].sizes[m];
79             }
80             else
81                 save.cand[m] = in_section[i].sizes[m];
82             } /* for m */
83         } /* if */
84     } /* if */
85 } /* for n */
86 return;
87 )

```

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/swapfrwd.c   February 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Swapfrwd.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : April 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10
11 - PURPOSE- This procedure attempts to swap one size from one
12 -           with another size in a different section if feasible. It
13 -           works from the start of the section list to the end.
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <malloc.h>
18 #include <memory.h>
19 #include <stdlib.h>
20 #include <string.h>
21 #include "impec.h"
22 #include "implcl.h"
23
24
25
26 void swap_forward(i, j, l, cut_cost, unit_cost, max_sizes, max_ply)
27     int i;
28     int j;
29     int l;
30     int cut_cost;
31     int unit_cost;
32     int max_sizes;
33     int max_ply;
34
35
36 (
37     int k, m, n;           /* counters */
38     int num_units;
39
40     for (n=0; n<num_of_sizes; n++) {
41         if ((in_section[i].sizes[j] > 0) &&
42             (in_section[l].sizes[n] > 0)) {
43
44             for (m=0; m<num_of_sizes; m++) {
45                 sect3.sizes[m] = in_section[i].sizes[m];
46                 sect4.sizes[m] = in_section[l].sizes[m];
47             }
48
49             sect3.sizes[j] = sect3.sizes[j] - 1;
50             sect3.sizes[n] = sect3.sizes[n] + 1;
51             sect4.sizes[j] = sect4.sizes[j] + 1;
52             sect4.sizes[n] = sect4.sizes[n] - 1;
53
54             temp_save.sect1 = i;
55             temp_save.sect2 = l;
56             temp_save.type = 0;
57             temp_save.org_ply_height = 0;

```

```

58     temp_save.cand_ply_height = 0;
59     temp_save.savings = (float) 0.0;
60
61     compute_swap_savings(i, l, cut_cost, unit_cost, max_sizes);
62
63     num_units = 0;
64     for (m=0; m<num_of_sizes; m++)
65         num_units = num_units + sect4.sizes[m];
66
67     if ((temp_save.savings > save.savings) &&
68         (num_units <= max_sizes) &&
69         (temp_save.type > 0) &&
70         (temp_save.cand_ply_height <= max_ply)) {
71         memcpy(&save, &temp_save, sizeof(savings_t));
72
73         for (m=0; m<num_of_sizes; m++) {
74             if (temp_save.type != 1) {
75                 save.org[m] = sect3.sizes[m];
76                 save.cand[m] = sect4.sizes[m];
77                 save.in_sect1[m] = in_section[i].sizes[m];
78                 save.in_sect2[m] = in_section[l].sizes[m];
79             }
80             else
81                 save.cand[m] = in_section[i].sizes[m];
82             } /* for m */
83         } /* if */
84     } /* if */
85 } /* for n */
86
87
88     return;
89 }

```

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/tranbkwd.c   February 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : Tranbkwd.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : April 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- This procedure attempts to transfer one size from one
12 -           section into another section if feasible. It works from
13 -           the end of the section list to the start.
14 -
15 -
16 - -----*/
17 #include <stdio.h>
18 #include <malloc.h>
19 #include <memory.h>
20 #include <stdlib.h>
21 #include <string.h>
22 #include "impdec.h"
23 #include "implcl.h"
24
25 void transfer_backwards(i, j, l, cut_cost, unit_cost, max_sizes, max_ply)
26     int i;
27     int j;
28     int l;
29     int cut_cost;
30     int unit_cost;
31     int max_sizes;
32     int max_ply;
33
34 {
35
36     int k, m;           /* counters */
37     int num_units;      /* num_units in one section */
38
39     if (in_section[i].sizes[j] > 0) {
40         for (m=0; m<num_of_sizes; m++) {
41             sect3.sizes[m] = in_section[i].sizes[m];
42             sect4.sizes[m] = in_section[l].sizes[m];
43         }
44
45         sect3.sizes[j] = sect3.sizes[j] - 1;
46         sect4.sizes[j] = sect4.sizes[j] + 1;
47         portion.sizes[j] = 1;
48         portion.ply_height = in_section[i].ply_height;
49
50         temp_save.sect1 = i;
51         temp_save.sect2 = l;
52         temp_save.type = 0;
53         temp_save.org_ply_height = 0;
54         temp_save.cand_ply_height = 0;
55         temp_save.savings = (float) 0.0;
56
57         compute_savings(i, l, cut_cost, unit_cost, max_sizes);

```



```

58
59     num_units = 0;
60     for (m=0; m<num_of_sizes; m++)
61         num_units = num_units + sect4.sizes[m];
62
63     if ((temp_save.savings > save.savings) &&
64         (num_units <= max_sizes) &&
65         (temp_save.type > 0) &&
66         (temp_save.cand_ply_height <= max_ply)) {
67         memcpy(&save, &temp_save, sizeof(savings_t));
68
69         for (m=0; m<num_of_sizes; m++) {
70             if (temp_save.type != 1) {
71                 save.org[m] = sect3.sizes[m];
72                 save.cand[m] = sect4.sizes[m];
73                 save.in_sect1[m] = in_section[i].sizes[m];
74                 save.in_sect2[m] = in_section[l].sizes[m];
75             }
76             else
77                 save.cand[m] = in_section[i].sizes[m];
78
79         }
80     }
81 }
82
83 return;
84 }

```

```

1  /* -----
2  -- $Header:: D:/cops/src/improv/tranfrwd.c   February 1990
3  -- -----*/
4
5  /*-----
6  - FILE NAME      : tranfrwd.c
7  - PROGRAMMER     : Terri A. Smith
8  - DATE WRITTEN  : April 1990
9  - ADDRESS        : GTRI/CSITL Atlanta GA 30332 (404) 894-8952
10 -
11 - PURPOSE- This procedure attempts to transfer one size from one
12 -           section into another section if feasible. It works from
13 -           the beginning of the section list to the end.
14 -
15 - -----*/
16 #include <stdio.h>
17 #include <malloc.h>
18 #include <memory.h>
19 #include <stdlib.h>
20 #include <string.h>
21 #include <time.h>
22 #include "impdec.h"
23 #include "implcl.h"
24
25 void transfer_forward(i, j, l, cut_cost, unit_cost, max_sizes, max_ply)
26     int i;
27     int j;
28     int l;
29     int cut_cost;
30     int unit_cost;
31     int max_sizes;
32     int max_ply;
33
34 {
35
36     int k, m;           /* counters */
37     int num_units;      /* num_units in one section */
38
39     if (in_section[i].sizes[j] > 0) {
40
41         for (m=0; m<num_of_sizes; m++) {
42             sect3.sizes[m] = in_section[i].sizes[m];
43             sect4.sizes[m] = in_section[l].sizes[m];
44         }
45
46         sect3.sizes[j] = sect3.sizes[j] - 1;
47         sect4.sizes[j] = sect4.sizes[j] + 1;
48         portion.sizes[j] = 1;
49         portion.ply_height = in_section[i].ply_height;
50
51         temp_save.sect1 = i;
52         temp_save.sect2 = l;
53         temp_save.type = 0;
54         temp_save.org_ply_height = 0;
55         temp_save.cand_ply_height = 0;
56         temp_save.savings = (float) 0.0;
57

```

```

58     compute_savings(i, l, cut_cost, unit_cost, max_sizes);
59
60     num_units = 0;
61     for (m=0; m<num_of_sizes; m++)
62         num_units = num_units + sect4.sizes[m];
63
64     if ((temp_save.savings > save.savings) &&
65         (num_units <= max_sizes) &&
66         (temp_save.type > 0) &&
67         (temp_save.cand_ply_height <= max_ply)) {
68         memcpy(&save, &temp_save, sizeof(savings_t));
69
70         for (m=0; m<num_of_sizes; m++) {
71             if (temp_save.type != 1) {
72                 save.org[m] = sect3.sizes[m];
73                 save.cand[m] = sect4.sizes[m];
74                 save.in_sect1[m] = in_section[i].sizes[m];
75                 save.in_sect2[m] = in_section[l].sizes[m];
76             }
77             else
78                 save.cand[m] = in_section[i].sizes[m];
79             /* for m */
80         }
81     } /* if */
82
83
84     return;
85 }

```

Appendix E: Computational Results from COP Algorithms

Savings													
Order	Ply		Patterns in						Pattern		Total inches		
	Height		order						Length	Ply	in pattern	in order	
		Size	30	32	34	36	38	40					
6/9/25/2/5/1	47		1				4	1	73.86	1	73.86		
				1	2	2	1		72.52	1	72.52		
			1	1	4				71.19	5	355.95		
				1	1				28.52	3	85.56		
													587.89
		Size	30	32	34	36	38	40					
	108		1				4	1	73.86	1	73.86		
				1	2	2	1		72.52	1	72.52		
			1	1	4				71.19	5	355.95		
				1	1				28.52	3	85.56		
													587.89
		Size	30	32	34	36	38	40					
200/200/200/200/200/200	47		1				1	4	74.52	47	3502.44		
			1				4	1	73.86	12	886.32		
			1		1		4		73.19	25	1829.75		
			1		3	1	1		72.08	5	360.40		
				1	3	2			72.08	47	3387.76		
				1	1	4			72.52	19	1377.88		
			1	4		1			70.75	25	1768.75		
			3	3					69.86	7	489.02		
			5	1					69.41	13	902.33		
													14304.65
		Size	30	32	34	36	38	40					
	108		1				1	4	74.52	32	2384.64		
			1				3	2	74.08	24	1777.92		
			1				4	1	73.86	24	1772.64		
				1	3	2			72.08	18	1297.44		
				1	2	3			72.30	54	3904.20		
				1	4	1			71.86	2	143.72		
			4	1	1				69.26	30	2077.80		
				1					15.56	96	1493.76		
													14852.12
		Size	30	32	34	36	38	40					
163/239/599/45/124/30	47		1				4	1	73.86	30	2215.80		
			1		3	1	1		72.08	4	288.32		
			3		2	1			70.75	35	2476.25		
				1	4	1			71.86	6	431.16		
				3	3				71.19	47	3345.93		
			1	2	3				70.97	24	1703.28		
					6				71.86	45	3233.70		
				5	1				70.75	4	283.00		
				4	1				67.01	6	402.06		
													14379.50

Savings (Con't)													
Order	Ply	Patterns in							Pattern		Total inches		
	Height	order							Length	Ply	in pattern	in order	
	Size	30	32	34	36	38	40						
	108	1				4	1	73.86	30		2215.80		
		1		3	1	1		72.08	4		288.32		
			1	4	1			71.86	41		2946.26		
		1	1	4				71.19	98		6976.62		
		1	4	1				70.52	7		493.64		
		1	3	1				66.76	24		1602.24		
													14522.88
	Size	30	32	34	36	38	40						
/ / / /960/240	48					3	3	75.19	48		3609.12		
						4	2	74.97	48		3598.56		
						6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						1		16.33	48		783.84		
													15145.44
	Size	30	32	34	36	38	40						
	108					3	3	75.19	72		5413.68		
						5	1	74.75	24		1794.00		
						6		74.52	96		7153.92		
						1		16.33	48		783.84		
													15145.44
	Size	30	32	34	36	38	40						
/ / / /1200/	48					6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						1		16.33	48		783.84		
													15091.68
	Size	30	32	34	36	38	40						
	108					6		74.52	108		8048.16		
						6		74.52	84		6259.68		
						1		16.33	48		783.84		
													15091.68
	Size	30	32	34	36	38	40						
72/144/360/360/144/72	48	1				2	3	74.30	24		1783.20		
		1		2	1	2		72.52	48		3480.96		
			1	3	2			72.08	48		3459.84		
			1	3	2			72.08	24		1729.92		
			1	1	4			72.52	24		1740.48		
			2	1	3			72.08	24		1729.92		
													13924.32

[illegible]

[illegible]

Improved Savings (con't)												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							length	ply	in pattern	in order
	Size	30	32	34	36	38	40					
	108					1	4	74.52	4		298.08	
			2			1	3	73.41	8		587.28	
						6		74.52	6		447.12	
					2	2	2	74.52	2		149.04	
				3	3			72.52	2		145.04	
				4	1	1		72.52	2		145.04	
		4	1	1				69.26	30		2077.80	
		3						40.36	2		80.72	
				1		3	2	74.52	8		596.16	
			1			1	4	74.75	2		149.50	
					1	4	1	74.52	4		298.08	
			1	2	2	1		72.52	4		290.08	
		2		2	1	1		71.63	4		286.52	
				3	3			72.52	4		290.08	
			1			2	3	74.52	4		298.08	
			1					15.56	86		1338.16	
						3	3	75.19	10		751.90	
		1	1	2	2			71.63	8		573.04	
			1	2	3			72.30	48		3470.40	
				1		4	1	74.30	8		594.40	
		2				1	3	73.40	26		1908.40	
		1	1					28.01	2		56.02	
												14830.94
	Size	30	32	34	36	38	40					
/ / / 1960/240	48					3	3	75.19	48		3609.12	
						4	2	74.97	48		3598.56	
						6		74.52	48		3576.96	
						6		74.52	48		3576.96	
						1		16.33	48		783.84	
											0.00	15145.44
	Size	30	32	34	36	38	40					
	108					3	3	75.19	72		5413.68	
						5	1	74.75	24		1794.00	
						6		74.52	96		7153.92	
						1		16.33	48		783.84	
												15145.44

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Cherry (con't)												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40					
72/144/360/360/144/72	47			1	1				29.02	48	1392.96	
				1	1				29.02	48	1392.96	
				1	1				29.02	48	1392.96	
				1	1				29.02	48	1392.96	
				1	1				29.02	48	1392.96	
			1			1			29.02	48	1392.96	
				1	1				29.02	48	1392.96	
			1			1			29.02	48	1392.96	
		1		1	1		1		55.70	48	2673.60	
			1			1			29.02	48	1392.96	
		1		1	1		1		55.70	24	1336.80	
												16547.04
	Size	30	32	34	36	38	40					
	108			1	1				29.02	108	3134.16	
				1	1				29.02	108	3134.16	
			1	1	1	1			55.46	108	5989.68	
		1					1		29.02	72	2089.44	
			1	1	1	1			55.46	36	1996.56	
												16344.00

Improved Cherry													
Order	Ply		Patterns in							Pattern		Total inches	
	Height		order						length	ply	in pattern	in order	
		Size	30	32	34	36	38	40					
6/9/25/2/5/1	47				2				28.76	12	345.12		
				1		1			28.76	2	57.52		
			1				1		28.76	5	143.80		
				3	1				54.19	1	54.19		
			1	1					28.01	1	28.01		
				2					28.26	1	28.26		
				1				1	29.27	1	29.27		
												686.17	
		Size	30	32	34	36	38	40					
	108				2				28.76	12	345.12		
				1		1			28.76	2	57.52		
			1				1		28.76	5	143.80		
				3	1				54.19	1	54.19		
			1	1					28.01	1	28.01		
				2					28.26	1	28.26		
				1				1	29.27	1	29.27		
												686.17	
		Size	30	32	34	36	38	40					
200/200/200/200/200/200	47		1	1	1	1	1	1	72.52	47	3408.44		
			1	1	1	1	1	1	72.52	47	3408.44		
			1	1	1	1	1	1	72.52	47	3408.44		
			1	1	1	1	1	1	72.52	47	3408.44		
			1	1	1	1	1	1	72.52	12	870.24		
												14504.00	
		Size	30	32	34	36	38	40					
	108		1	1	1	1	1	1	72.52	108	7832.16		
			1	1	1	1	1	1	72.52	92	6671.84		
												14504.00	
		Size	30	32	34	36	38	40					
0/0/0/0/960/240	48						3	3	75.19	48	3609.12		
							4	2	74.97	48	3598.56		
							6		74.52	48	3576.96		
							6		74.52	48	3576.96		
							1		16.33	48	783.84		
												15145.44	
		Size	30	32	34	36	38	40					
	108						3	3	75.19	24	1804.56		
							1		16.33	36	587.88		
							6		74.52	48	3576.96		
							4	2	74.97	84	6297.48		
							2		29.77	60	1786.20		
							3		43.38	36	1561.68		
												15614.76	

Improved Cherry (con't)													
Order	Ply		Patterns in							Pattern		Total inches	
	Height		order						length	ply	in pattern	in order	
		Size	30	32	34	36	38	40					
163/239/599/45/124/30	47				4	1	1		72.52	6	435.12		
				1	5				71.63	8	573.04		
			2						27.75	40	1110.00		
			1	1					28.01	45	1260.45		
				2					28.26	47	1328.22		
					5			1	72.52	7	507.64		
				1	3		2		72.52	9	652.68		
					6				71.86	2	143.72		
				3					41.11	2	82.22		
					3		3		73.19	2	146.38		
					4	1	1		72.52	31	2248.12		
				1	1				28.52	2	57.04		
				1	1	1	3		73.19	2	146.38		
				1	4		1		72.08	22	1585.76		
				1	2	1	2		72.75	5	363.75		
				2	2		1	1	72.52	23	1667.96		
					3	1	2		72.97	1	72.97		
					6				71.86	23	1652.78		
			1		1				28.26	38	1073.88		
					2				28.76	2	57.52		
												15165.63	
		Size	30	32	34	36	38	40					
	108				1		3	2	74.52	2	149.04		
			1		1				28.26	8	226.08		
				2					28.26	24	678.24		
			2						27.75	37	1026.75		
					2				28.76	70	2013.20		
					5			1	72.52	8	580.16		
					3	2	1		72.75	2	145.50		
					4	1	1		72.52	6	435.12		
						2	2	2	74.52	2	149.04		
				2	1				41.36	2	82.72		
				1	1				28.52	28	798.56		
				1	3		1	1	72.75	2	145.50		
					3	3			72.52	5	362.60		
				1	2	2	1		72.52	2	145.04		
				1	3	1		1	72.52	2	145.04		
					1		1		29.27	91	2663.57		
					3	1	1	1	73.19	4	292.76		
					4		1	1	72.97	1	72.97		
				2	3			1	72.01	2	144.02		
			1	2	3				70.97	66	4684.02		
				2	4				71.41	1	71.41		
			1	1					28.01	15	420.15		
					1	2	2	1	73.86	3	221.58		
												15653.07	

[illegible]

IMPROVEMENT												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40					
6/9/25/2/5/1	47				1	4	1	74.52	1		74.52	
		4	1	1				69.26	1		69.26	
			1	5				71.63	4		286.52	
				4	1	1		72.52	1		72.52	
		2	4					70.08	1		70.08	
												572.90
	Size	30	32	34	36	38	40					
6/9/25/2/5/1	108				1	4	1	74.52	1		74.52	
		4	1	1				69.26	1		69.26	
			1	5				71.63	4		286.52	
				4	1	1		72.52	1		72.52	
		2	4					70.08	1		70.08	
												572.90
	Size	30	32	34	36	38	40					
200/200/200/200/200/200	47			4	1	1		72.52	47		3408.44	
			2		3	1		72.52	12		870.24	
		2			1	3		72.52	35		2538.20	
		1	2		1		2	72.52	47		3408.44	
		1	1	1			3	73.19	12		878.28	
		1	2		1		2	72.52	35		2538.20	
		3				3		71.86	12		862.32	
												14504.12
	Size	30	32	34	36	38	40					
	108	1	4	1				70.52	4		282.08	
				1	1	1	3	74.52	4		298.08	
				3	3			72.52	16		1160.32	
				1		3	2	74.52	20		1490.40	
					2	2	2	74.52	60		4471.20	
		4	1	1				69.26	48		3324.48	
			1	5				71.63	4		286.52	
			4	1			1	71.63	4		286.52	
			1	2	2	1		72.52	4		290.08	
			2			1	3	73.41	8		587.28	
		1	3	1		1		71.19	4		284.76	
			6					70.52	4		282.08	
			3	2	1			71.41	20		1428.20	
												14472.00

IMPROVEMENT (con't)													
Order	Ply		Patterns in							Pattern		Total inches	
	Height		order						Length	Ply	in pattern	in order	
		Size	30	32	34	36	38	40					
163/239/599/45/124/30	47		4	1	1				69.25	32	2216.32		
				1	3		2		72.52	27	1958.04		
					5			1	72.52	26	1885.52		
							6		74.52	2	149.04		
				3	2	1			71.41	22	1571.02		
				3			3		72.52	3	217.56		
				2			1	3	73.41	1	73.41		
			2	2	2				70.52	3	211.56		
			1	4	1				70.52	13	916.76		
				6					70.52	3	211.56		
				1	5				71.63	19	1360.97		
					4	1	1		72.52	23	1667.96		
			1	1	1		3		72.52	3	217.56		
					3		3		73.19	5	365.95		
			1	1	3		1		71.63	1	71.63		
			2						27.75	6	166.50		
					6				71.86	14	1006.04		
				4	1			1	71.63	1	71.63		
												14339.03	
		Size	30	32	34	36	38	40					
	108						6		74.52	3	223.56		
				1	5				71.63	49	3509.87		
					4	1	1		72.52	15	1087.80		
			4	1	1				69.26	37	2562.62		
					5			1	72.52	18	1305.36		
				3			3		72.52	2	145.04		
				4	1			1	71.63	9	644.67		
				2	2		1	1	72.52	1	72.52		
					1		3	2	74.52	1	74.52		
			1		2	1	2		72.52	2	145.04		
				2	1	1	2		72.52	1	72.52		
				3	2	1			71.41	26	1856.66		
				1	3		2		72.52	28	2030.56		
			4						52.93	3	158.79		
			1	1	1		3		72.52	1	72.52		
					1	1	4		73.96	1	73.86		
					3		3		73.19	4	292.76		
												14328.67	
		Size	30	32	34	36	38	40					
/ / / /960/240	48						4	2	74.97	48	3598.56		
							6		74.52	48	3576.96		
							3	3	75.19	48	3609.12		
							6		74.52	48	3576.96		
							1		16.33	48	783.84		
												15145.44	

[illegible]

[illegible]

[illegible]

[illegible]

Improved Package A													
Order	Ply	Patterns in								Pattern		Total inches	
	Height	order								Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40						
6/9/25/2/5/1	47			5			1			72.52	1	72.52	
			1	1		2	2			72.52	1	72.52	
				3			3			72.52	1	72.52	
			1	1	4					71.19	5	355.95	
													573.51
	Size	30	32	34	36	38	40						
	108			5			1			72.52	1	72.52	
			1	1		2	2			72.52	1	72.52	
				3			3			72.52	1	72.52	
			1	1	4					71.19	5	355.95	
													573.51
	Size	30	32	34	36	38	40						
200/200/200/200/200/200	47			1		3	2			74.52	6	447.12	
			4	1	1					69.26	14	969.64	
				3		1	1	1		72.52	7	507.64	
					3	3				72.52	8	580.16	
				1	1	3	1			72.75	6	436.50	
			2							27.75	20	555.00	
				1	3	1		1		72.52	7	507.64	
				3	1			2		72.52	6	435.12	
					2			4		74.52	7	521.64	
					1	1	1	3		74.52	7	521.64	
			4							52.93	7	370.51	
						3		3		74.52	6	447.12	
				4		1	1			71.63	21	1504.23	
				1	2	2		1		72.75	6	436.50	
						1	4	1		74.52	7	521.64	
				3	2	1				71.41	1	71.41	
							3	3		75.19	1	75.19	
			1	4			1			70.97	1	70.97	
			2	2						53.45	1	53.45	
			1		2	2		1		72.52	13	942.76	
					3	1		2		73.41	6	440.46	
			2					2		55.46	6	332.76	
					2	1		3		74.08	1	74.08	
			2		1	1				54.19	6	325.14	
				5			1			71.19	5	355.95	
			1				4	1		73.86	26	1920.36	
					2	3		1		73.19	13	951.47	
			2	2	2					70.52	5	352.60	
													14728.70

Improved Package A (con't)												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40					
	108				1	4	1	74.52	40	2980.80		
			1		1		4	74.52	15	1117.80		
		1	3		1		1	71.63	15	1074.45		
		4						52.93	15	793.95		
					3	3		73.86	5	369.30		
			3	2	1			71.41	20	1428.20		
			4	1			1	71.63	15	1074.45		
		1		2	2		1	72.52	15	1087.80		
		2	1	1	1		1	71.63	5	358.15		
			2	1	1	2		72.52	5	362.60		
		2		2	1		1	71.86	50	3593.00		
			1	1	1	3		73.19	5	365.95		
											14606.45	
		Size	30	32	34	36	38	40				
163/239/599/45/124/30	47			1		3	2	74.52	6	447.12		
		1	1	1	1	1	1	72.52	2	145.04		
			1	2	2	1		72.52	1	72.52		
			1	5				71.63	20	1432.60		
		1	5					70.30	1	70.30		
			6					70.52	8	564.16		
		6						69.19	1	69.19		
						6		74.52	4	298.08		
		1	1	2	2			71.63	12	859.56		
		1		2	2		1	72.52	4	290.08		
				4	1	1		72.52	1	72.52		
		1	4	1				70.52	2	141.04		
			1	3		2		72.52	2	145.04		
		4	1	1				69.26	2	138.52		
		1	1	3		1		71.63	12	859.56		
		1						15.31	2	30.62		
			3	1	2			71.63	1	71.63		
			1	1		4		73.41	2	146.82		
			1	1	4			71.19	30	2135.70		
				3	2			71.86	3	215.58		
		1	1	4			71.19	27	1922.13			
				1		4	1	74.30	12	891.60		
			2	3				71.86	6	431.16		
		1	1	4				71.19	23	1637.37		
			2	4				71.41	10	714.10		
		3	1					53.19	10	531.90		
		1		5				71.41	1	71.41		
											14405.35	

Improved Package A (con't)													
Order	Ply	Patterns in								Pattern		Total inches	
	Height	order								Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40						
	108	2	1	1		2			71.63	1		71.63	
				1		3	2		74.52	2		149.04	
			1	2	2	1			72.52	3		217.56	
			1	5					71.63	5		358.15	
			4	1			1		71.63	1		71.63	
			6						70.52	8		564.16	
		5	1						69.41	1		69.41	
					1	4	1		74.52	6		447.12	
				4	1	1	1		72.52	1		72.52	
		1	4	1					70.52	3		211.56	
			1	3		2			72.52	2		145.04	
		1		2	2		1		72.52	2		145.04	
		2		3			1		71.63	6		429.78	
						6			74.52	6		447.12	
			1	2		2	1		73.19	1		73.19	
		2		1		2	1		72.52	1		72.52	
			5						61.76	1		61.76	
		1		4	1				71.63	4		286.52	
		1	1	3		1			71.63	1		71.63	
						5	1		74.75	8		598.00	
		1		3	2				71.86	12		862.32	
		1	1	4					71.19	103		7332.57	
			2	4					71.41	13		928.33	
			2			3	1		73.86	1		73.86	
		1	3						53.69	7		375.83	
				5					63.03	1		63.03	
		2							27.75	2		55.50	
		2	2						53.45	3		160.35	
													14415.17
	Size	30	32	34	36	38	40						
/ / / /960/240	48					4	2		74.97	48		3598.56	
						4			56.97	12		683.64	
						6			74.52	36		2682.72	
						4	2		74.97	36		2698.92	
						6			74.52	36		2682.72	
						4	2		74.97	36		2698.92	
													15045.48
	Size	30	32	34	36	38	40						
	108					3	3		75.19	48		3609.12	
						4			56.97	12		683.64	
						6			74.52	60		4471.20	
						4	2		74.97	12		899.64	
						5	1		74.75	72		5382.00	
													15045.60

Improved Package A (con't)													
Order	Ply	Patterns in							Pattern		Total inches		
	Height	order							Length	Ply	in pattern	in order	
	Size	30	32	34	36	38	40						
/ / / /1200/	48					2			29.77	36	1071.72		
						6			74.52	48	3576.96		
						4			56.97	48	2734.56		
						6			74.52	48	3576.96		
						6			74.52	24	1788.48		
						6			74.52	36	2682.72		
												15431.40	
	Size	30	32	34	36	38	40						
	108					2			29.77	24	714.48		
						6			74.52	108	8048.16		
						6			74.52	84	6259.68		
												15022.32	
	Size	30	32	34	36	38	40						
72/144/360/360/144/72	48					2	2	2	74.52	24	1788.48		
			1	2	2	1			72.52	24	1740.48		
			1	5					71.63	6	429.78		
				3	3				72.52	30	2175.60		
			2	2		1	1		72.52	24	1740.48		
		4	2						69.63	6	417.78		
		1	1		2	2			72.52	24	1740.48		
			2	3	1				71.63	12	859.56		
			3	1	2				71.63	6	429.78		
		1		4	1				71.63	12	859.56		
		1	4	1					70.52	12	846.24		
			1	4	1				71.86	6	431.16		
			1	3	2				72.08	6	432.48		
												13891.86	
	Size	30	32	34	36	38	40						
	108			1	2	2	1		73.86	72	5317.92		
				3	3				72.52	72	5221.44		
		1	2	1					53.94	72	3883.68		
												14423.04	

Package B												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40					
6/9/25/2/5/1	47	1	1	4					71.19	6	427.14	
			1		1	2			55.96	2	111.92	
			1	1		1	1		55.96	1	55.96	
												595.02
	Size	30	32	34	36	38	40					
	108	1	1	4					71.19	6	427.14	
			1		1	2			55.96	2	111.92	
			1	1		1	1		55.96	1	55.96	
												595.02
	Size	30	32	34	36	38	40					
200/200/200/200/200/200	47	1	1	1	1	1	1		72.52	47	3408.44	
		1	1	1	1	1	1		72.52	47	3408.44	
		1	1	1	1	1	1		72.52	47	3408.44	
		1	1	1	1	1	1		72.52	47	3408.44	
		1	1	1	1	1	1		72.52	12	870.24	
												14504
	Size	30	32	34	36	38	40					
	108	1	1	1	1	1	1		72.52	108	7832.16	
		1	1	1	1	1	1		72.52	92	6671.84	
												14504
	Size	30	32	34	36	38	40					
163/239/599/45/124/30	47	1	1	4					71.19	47	3345.93	
		1	1	4					71.19	47	3345.93	
		1	1	4					71.19	47	3345.93	
		1	1	4					71.19	8	569.52	
			2		1	3			72.97	41	2991.77	
		1					2		42.88	14	600.32	
			2		1				41.62	4	166.48	
				1			1		29.52	2	59.04	
				1		1			29.27	1	29.27	
												14454.19
	Size	30	32	34	36	38	40					
	108	1	1	4					71.19	108	7688.52	
		1	1	4					71.19	41	2918.79	
			2		1	3			72.97	41	2991.77	
		2					4		73.64	7	515.48	
			2		1				41.62	4	166.48	
				3		1	2		73.64	1	73.64	
												14354.68

Package B Con't												
Order	Ply	Patterns in							Pattern		Total inches	
	Height	order							Length	Ply	in pattern	in order
	Size	30	32	34	36	38	40					
/ / / /960/240	48					5	1	74.75	48		3588	
						5	1	74.75	48		3588	
						5	1	74.75	48		3588	
						5	1	74.75	48		3588	
						6		75.86	8		606.88	
												14958.88
	Size	30	32	34	36	38	40					
	108					5	1	74.75	108		8073	
						5	1	74.75	84		6279	
						6		75.86	8		606.88	
												14958.88
	Size	30	32	34	36	38	40					
/ / / /1200/	48					6		74.52	48		3576.96	
						6		74.52	48		3576.96	
						6		74.52	48		3576.96	
						6		74.52	48		3576.96	
						6		74.52	8		596.16	
												14904
	Size	30	32	34	36	38	40					
	108					6		74.52	108		8048.16	
						6		74.52	92		6855.84	
												14904
	Size	30	32	34	36	38	40					
72/144/360/360/144/72	48		1	2	2	1		72.52	48		3480.96	
			1	2	2	1		72.52	48		3480.96	
			1	2	2	1		72.52	48		3480.96	
		2		2	2			71.41	36		2570.76	
						6		75.86	12		910.32	
												13923.96
	Size	30	32	34	36	38	40					
	108		1	2	2	1		72.52	108		7832.16	
			1	2	2	1		72.52	36		2610.72	
		2		2	2			71.41	36		2570.76	
						6		75.86	12		910.32	
												13923.96

[illegible]

Improved Package B (con't)													
Order	Ply		Patterns in							Pattern		Total inches	
	Height		order						Length	Ply	in pattern	in order	
		Size	30	32	34	36	38	40					
163/239/599/45/124/30	47		1		2	2		1	72.52	2	145.04		
					4	1	1		72.52	2	145.04		
					5			1	72.52	6	435.12		
					1		3	2	74.52	1	74.52		
			3						40.36	2	80.72		
				6					70.52	2	141.04		
				3			3		72.52	2	145.04		
			6						69.19	2	138.38		
			1	3			2		71.63	1	71.63		
				1	5				71.63	47	3366.61		
				2	3	1			71.63	1	71.63		
			2		3			1	71.63	1	71.63		
				1	4	1			71.86	1	71.86		
			1	5					70.30	1	70.30		
			1	2	2			1	71.63	1	71.63		
			2	1	3				70.75	43	3042.25		
				2		1	3		72.97	37	2699.89		
					4			2	73.19	9	658.71		
			1	1	4				71.19	36	2562.84		
			2	1					40.61	8	324.88		
												14388.76	
		Size	30	32	34	36	38	40					
	108		1		4	1			71.63	3	214.89		
				1	5				71.63	5	358.15		
			2	2		1	1	1	71.63	1	71.63		
				2			1	3	73.41	2	146.82		
			4	1	1				69.26	1	69.26		
			1	1	2			2	72.52	1	72.52		
			1		3		1	1	72.52	1	72.52		
				2	3	1			71.63	2	143.26		
				2	1	1	2		72.52	1	72.52		
			2				1	3	73.40	1	73.40		
			6						69.19	1	69.19		
				1	3	1		1	72.52	1	72.52		
				3			3		72.52	1	72.52		
			1	1	3	1			71.41	1	71.41		
			1				1	4	74.52	1	74.52		
				3	3				71.19	1	71.19		
			1	3	2				71.63	1	71.63		
				2		1	3		72.97	37	2699.89		
			2					4	73.64	3	220.92		
			1	1	4				71.19	101	7190.19		
			1	1	4				71.19	34	2420.46		
												14329.41	

Improved Package B (con't)													
Order	Ply	Patterns in							Pattern		Total inches		
	Height	order							Length	Ply	in pattern	in order	
	Size	30	32	34	36	38	40						
/ / / /960/240	48					4	2	74.97	48		3598.56		
						6		74.52	48		3576.96		
						4	2	74.97	48		3598.56		
						6		74.52	48		3576.96		
							6	75.86	8		606.88		
													14957.92
	Size	30	32	34	36	38	40						
	108					4	2	74.97	96		7197.12		
						6		74.52	96		7153.92		
							6	75.86	8		606.88		
													14957.92
	Size	30	32	34	36	38	40						
/ / / /1200/	48					6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						6		74.52	48		3576.96		
						6		74.52	8		596.16		
													14904.00
	Size	30	32	34	36	38	40						
	108					6		74.52	108		8048.16		
						6		74.52	92		6855.84		
													14904.00
	Size	30	32	34	36	38	40						
72/144/360/360/144/72	48			1	2	2	1	72.52	48		3480.96		
				1	2	2	1	72.52	48		3480.96		
			1	1	2	2		71.63	12		859.56		
			1		2	2	1	72.52	36		2610.72		
			2				1	73.40	12		880.80		
				1	2	2	1	72.52	36		2610.72		
													13923.72
	Size	30	32	34	36	38	40						
	108			1	2	2	1	72.52	108		7832.16		
			1	1	2	2		71.63	12		859.56		
			1		2	2	1	72.52	36		2610.72		
			2				1	73.40	12		880.80		
				1	2	2	1	72.52	24		1740.48		
													13923.72