CARGO LOADING--A PROPOSED APPROACH FOR MAXIMIZING SPACE UTILIZATION OF CONTAINERS LOADED WITH PALLETTIZED LOADS

An in-depth analysis of space utilization in cargo containers to be loaded with palletized loads and shipped by barge indicates that space utilization can be significantly improved. This report describes a loading procedure which uses analytic techniques to precisely define the amount and arrangement of loads at the three different loading levels involved. Development of a computer program to implement the results of the study is recommended.
(Block 10)
Program Element 62760N
Task Area TF 60 531 005
Work Unit 1800-007

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification

By
Distribution/
Availability Codes

Dist
Avail and/or
Special

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>ADMINISTRATIVE INFORMATION</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>ALTERNATIVE APPROACHES TO AUTOMATED LOADING</td>
<td>5</td>
</tr>
<tr>
<td>FIXED PATTERN APPROACH</td>
<td>5</td>
</tr>
<tr>
<td>ANALYTIC FORMULATIONS</td>
<td>7</td>
</tr>
<tr>
<td>PROPOSED APPROACH</td>
<td>11</td>
</tr>
<tr>
<td>OVERVIEW</td>
<td>11</td>
</tr>
<tr>
<td>GUIDELINES FOR SOLUTION STRATEGY</td>
<td>12</td>
</tr>
<tr>
<td>INPUT REQUIREMENTS AND PRELIMINARY LOAD DATA ANALYSIS</td>
<td>25</td>
</tr>
<tr>
<td>PALLET LOADING</td>
<td>28</td>
</tr>
<tr>
<td>Pattern Formation</td>
<td>28</td>
</tr>
<tr>
<td>Stacking</td>
<td>47</td>
</tr>
<tr>
<td>Completion</td>
<td>57</td>
</tr>
<tr>
<td>CONTAINER LOADING</td>
<td>59</td>
</tr>
<tr>
<td>BARGE LOADING</td>
<td>61</td>
</tr>
<tr>
<td>ORGANIZATION AND FUNCTIONAL DESCRIPTION OF PROPOSED PROGRAM LOADER</td>
<td>63</td>
</tr>
<tr>
<td>SAMPLE PROBLEM</td>
<td>65</td>
</tr>
<tr>
<td>STATEMENT OF PROBLEM</td>
<td>65</td>
</tr>
<tr>
<td>SOLUTION</td>
<td>67</td>
</tr>
<tr>
<td>Assumptions and Restrictions</td>
<td>67</td>
</tr>
<tr>
<td>Preliminary Load Data Analysis</td>
<td>68</td>
</tr>
<tr>
<td>Pallet Loading</td>
<td>70</td>
</tr>
<tr>
<td>Group 1 Processing</td>
<td>71</td>
</tr>
<tr>
<td>Group 2 Processing</td>
<td>73</td>
</tr>
<tr>
<td>Group 3 Processing</td>
<td>78</td>
</tr>
<tr>
<td>Group 4 Processing</td>
<td>81</td>
</tr>
<tr>
<td>Group 5 Processing</td>
<td>84</td>
</tr>
<tr>
<td>Group 6 Processing</td>
<td>85</td>
</tr>
<tr>
<td>Group 7 Processing</td>
<td>86</td>
</tr>
<tr>
<td>Group 8 Processing</td>
<td>87</td>
</tr>
<tr>
<td>Remainder Group Processing</td>
<td>88</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Stacking and Completion</td>
<td>89</td>
</tr>
<tr>
<td>Container #1</td>
<td>90</td>
</tr>
<tr>
<td>Final Stack #1</td>
<td>94</td>
</tr>
<tr>
<td>Final Stack #2</td>
<td>95</td>
</tr>
<tr>
<td>Container #2</td>
<td>96</td>
</tr>
<tr>
<td>Initial Stack #1</td>
<td>97</td>
</tr>
<tr>
<td>Initial Stack #2</td>
<td>99</td>
</tr>
<tr>
<td>Container Loading</td>
<td>102</td>
</tr>
<tr>
<td>DISCUSSION OF RESULTS</td>
<td>102</td>
</tr>
<tr>
<td>RECOMMENDATIONS AND CONCLUSIONS</td>
<td>105</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>109</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>111</td>
</tr>
<tr>
<td>APPENDIX A - INDEX CHART FOR PALLET PATTERN DETERMINATION</td>
<td>113</td>
</tr>
<tr>
<td>APPENDIX B - PALLET PATTERNS ON 40&quot; X 48&quot; PALLETS</td>
<td>117</td>
</tr>
<tr>
<td>APPENDIX C - FLOWCHARTS OF PROPOSED PROGRAM LOADER</td>
<td>121</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Difficulty of Mixing Box Sizes Using Fixed Patterns</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Pictorial Overview of Proposed Approach</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Example of Cluster Types</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Type 1 Configuration</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>Type 2 Configuration</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>Type 3 Configuration</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>Type 4 Configuration</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>Pattern Completion by Nesting of Type 3 and Type 4 Configurations</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>Alternative Pallet Loads Spanning Identical Stack Height</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>Improvement of Pallet Load Stability</td>
<td>60</td>
</tr>
<tr>
<td>11</td>
<td>Example of Pattern Completion by Synthesizing Group Height</td>
<td>107</td>
</tr>
<tr>
<td>12</td>
<td>Generalization of Row, Column Construction Techniques</td>
<td>107</td>
</tr>
<tr>
<td>13</td>
<td>Main Program LOADER, Flowchart</td>
<td>121</td>
</tr>
<tr>
<td>14</td>
<td>Subroutine INPUT, Flowchart</td>
<td>122</td>
</tr>
<tr>
<td>15</td>
<td>Subroutine ANALYZ, Flowchart</td>
<td>123</td>
</tr>
<tr>
<td>16</td>
<td>Subroutine PATRNS, Flowchart</td>
<td>124</td>
</tr>
<tr>
<td>17</td>
<td>Subroutine STRING, Flowchart</td>
<td>130</td>
</tr>
<tr>
<td>18</td>
<td>Subroutine TYPE1, Flowchart</td>
<td>133</td>
</tr>
<tr>
<td>19</td>
<td>Subroutine TYPE2, Flowchart</td>
<td>134</td>
</tr>
<tr>
<td>20</td>
<td>Subroutine TYPE3, Flowchart</td>
<td>135</td>
</tr>
<tr>
<td>21</td>
<td>Subroutine TYPE4, Flowchart</td>
<td>138</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1 - System State Definitions ...................................... 52
2 - State Transition Matrix ........................................ 52
3 - Sample Problem Load Population Data .......................... 69
ABSTRACT

An in-depth analysis of space utilization in cargo containers to be loaded with palletized loads and shipped by barge indicates that space utilization can be significantly improved. This report describes a loading procedure which uses analytic techniques to precisely define the amount and arrangement of loads at the three different loading levels involved. Development of a computer program to implement the results of the study is recommended.

ADMINISTRATIVE INFORMATION

This research was performed for the Logistics Division, Code 187, of the Computation, Mathematics and Logistics Department, Code 18. The author was on the staff of the Computer Aided Design and Manufacturing Division, Code 185. The work was sponsored by the Naval Supply Systems Command, Code 043 as a subtask of Work Request WR-09012, Program Element 62760N, Task Area TF 60 531 005, and Work Unit 1800-008.
INTRODUCTION

One of the foremost problems in the field of containerization is the need for improving the use of cubic capacity of cargo containers. Both Tabak and NAVSEASYSCOM mention the significance of the cube limitation problem and the inefficient manner in which storage space is used.

The importance of the problem and the economic savings to be realized if a successful solution could be obtained led to a study of space utilization in cargo containers to be loaded with palletized loads. Loading takes place at the three different levels, and each level affects the overall efficiency of space utilization. The three levels are:

- loading of boxes onto pallets
- loading of pallets into containers
- loading of containers onto a barge.

The results of the study, reported here, include the development of an automated loading procedure that precisely defines the amount and arrangement of loads at each level of loading.

*A complete listing of references is given on page 111.
FIXED PATTERN APPROACH

MIL-STD-147B describes and illustrates the practices and procedures for palletizing unit loads. Appendixes C and D of the standard, included in this report as Appendixes A and B, provide an important step in the direction of automated loading. The first appendix is an index chart for pallet pattern determination. The chart covers box sizes ranging in length from 6 to 52 inches in increments of half an inch, and in width from 6 to 43 inches in increments of half an inch. A pattern number read from the chart for a box of specified length and width corresponds to a pattern in the second appendix. Each of the 124 pallet patterns provides for efficient use of at least 80 percent of the pallet surface. However, there are numerous box sizes for which no pattern is provided (for example, 15"W X 18"L), and generally a pattern can be relied upon only if all boxes are the same size. For example, uniform loads of boxes 17-1/2"W X 34"L or of boxes 21-1/2"W X 28"L can be loaded on a 40" X 48" pallet using Pattern 1 as shown in Figures 1A and 1B. However, if the sizes are mixed as shown in Figure 1C, the pattern is not suitable, since the maximum allowable load width is 52 inches. If the boxes are arranged as shown in Figure 1D, the pattern is acceptable.

The idea of using previously developed, efficient, fixed patterns for selected box sizes is simple and logical, but has several limitations:
Figure 1a - Acceptable loading of $17\frac{1}{2}'' \times 34''$ boxes using pattern 1

Figure 1b - Acceptable loading of $21\frac{1}{2}'' \times 28''$ boxes using pattern 1

Figure 1c - Unacceptable loading of 2 sizes of boxes using pattern 1

Figure 1d - Acceptable loading of 2 sizes of boxes using pattern 1

Figure 1 - Difficulty of Mixing Box Sizes Using Fixed Patterns
If the number of permissible box sizes is large, the number of pre-developed fixed patterns becomes large; if mixing of sizes is permitted, determining the large number of mixed patterns becomes infeasible.

There is no correspondence between the actual frequency of use of certain sizes and the frequency with which the sizes occur in the fixed patterns.

There is no mechanism for improving space utilization efficiency; if a fixed pattern is selected with a reasonable amount of waste, a more efficient pattern might be easily found but is never sought.

If the allowable load widths or lengths are changed, or if the maximum permissible dimensions are variable, then considerable rework may be needed to expand or alter the fixed patterns. The same difficulties would apply if the permissible box sizes were altered.

ANALYTIC FORMULATIONS

A more general approach than using fixed patterns would be to allow for linear combinations of box sizes and more general types of permissible pattern variations. Gilmore and Gomory,\(^4\) Herz\(^5\) and Christofides and Whitlock\(^6\) reported work of this nature but in a different context.

Gilmore and Gomory\(^4\) consider multi-stage cutting stock problems of two and more dimensions. In the two-dimensional cutting stock problem a supply of stock rectangles of width \(W\) and length \(L\) is used to fill a demand for \(N_i\) rectangles of width \(w_i\) and length \(l_i\), \(i = 1, 2, \ldots, m\).
problem is to cut the stock rectangles into smaller demand rectangles using as few stock rectangles as possible. In the cargo loading problem, the fixed size pallet corresponds to the stock rectangle of the cutting stock problem and the width and length of the box correspond to the demand rectangle dimensions. The number of boxes of each size corresponds to the $N_i$ rectangles and the number of different size boxes corresponds to the $m$ rectangles. In both problems, the objective is to minimize waste.

In the three-dimensional problem, a rectangular parallelepiped is used instead of a rectangle, and its height becomes the third dimension.

Gilmore and Gomory discuss a linear programming formulation and consider the computational difficulties which arise from the immense number of columns that can occur in the matrix. For the one-dimensional cutting stock problem, the computational difficulty can be reduced by transforming the problem and solving it as a knapsack problem. Unfortunately, for two dimensions, the analogous transformation from the linear programming problem into a generalized knapsack problem results in a problem for which no practical means of solution is known. The difficulty increases if the three-dimensional problem is considered.

Herz considers the two-dimensional cutting stock problem. The Gilmore-Gomory solution technique involves an iterative algorithm, but the Herz algorithm is recursive and its author claims that, when implemented, it results in higher computational speeds in the solution of the cutting-stock problem. Herz gives some computational results for a program which was written in PL/1 and executed on an IBM computer.
Christofides and Whitlock present a tree-search algorithm in which the maximum number of each type of piece produced is constrained. A dynamic programming solution technique is used. The results of a program developed for the CDC-7600 computer indicate that the algorithm is effective in solving cutting-stock problems of moderate size. When ten pieces are to be cut from the stock rectangles, the solution time is approximately 15 seconds. When the number of pieces is increased to twenty, the solution time is increased to 1 minute. For a larger number of pieces, the number of nodes in the tree and computational times become so large as to be impractical.
PROPOSED APPROACH

OVERVIEW

The approach proposed here uses analytic procedures to generalize the fixed pattern approach and to take into account the specific loading constraints associated with the cargo loading problem. This approach necessitates the development of a computer program which permits the three-dimensional cargo loading problem to be attacked from a more realistic point of view. The key concepts involve:

- Iteration to improve space utilization to within acceptable tolerances; implicit exclusion of unwarranted iterations.
- Generation of load clusters.
- Generation of configuration types dependent on priorities and on frequency of box size occurrence within the load population.
- Pattern completion by integer linear combination and nesting of clusters.
- Generation of load stacks.
- Stack filling for crude but rapid utilization of space.
- Load trading for refinement of space utilization.
- Automatic insertion of pallets into load stacks for adjustment of pallet load heights.
- Automatic determination of arrangement of pallets in containers and containers in barges.
- Automatic monitoring of load weights.
- Automatic placement of loads for improved load stability.
- Ability to allow for variability in available box sizes, pallet sizes, containers sizes, barge types, and wasted space required for load accessibility.
- Ability to trade off space utilization efficiency versus computer costs.
- Provision for complete computer bookkeeping and output reporting, including space utilization efficiency, at all loading levels.

Although these concepts have been listed separately, many of them are interdependent and receive simultaneous consideration in the development of the problem solution. A pictorial overview of the proposed approach appears in Figure 2.

GUIDELINES FOR SOLUTION STRATEGY

A number of basic criteria and mathematical operations must be stated or developed in order to understand the solution strategy. These fundamental ideas provide the basis for the decision rules that deal with important questions frequently raised in the loading process, such questions as:
1. Group boxes on basis of box heights

Group 1
All boxes of height $H$ in range $8'' < H \leq 10''$

Group 2
All boxes of height $H$ in range $10'' < H \leq 12''$

2. Generate box clusters within each group

Various clusters
3. Form patterns by combining clusters

![Diagram of a pattern]

4. Assign patterns to tiers; and insert shims to level tiers; generate load stacks by stacking tiers into preliminary piles

![Diagram of a stack with labeled patterns and tiers]
5. Refine piles and insert pallets into stack

Details of pattern on tier face not shown

Stack #1 refined and palletized

6. Arrange stacks in container

7. Arrange containers on barge

Figure 2 (Continued)
In what sequence should the boys be selected from the load population?

How should loading patterns be formed and what measure should be used to compare their effectiveness in space utilization?

How can a current loading pattern be developed that gives consideration to enhancement of the load pattern opportunities for the remaining load population?

If wasted space exists, what types of modifications should be allowed in altering the configuration and composition of a given subset of the load population to improve space utilization?

The first guideline to be introduced is associated with the utility or value of a loading pattern and the contributing worth of each of its components. Ideally, it would be desirable to have a precise quantitative measure of each of these factors, but unfortunately none is available. For a small load population, trial and error or possibly the use of dynamic programming might provide the best overall utilization of space and, a posteriori, assign the highest value of utility to this result. For a large population, this approach is not practical.

Intuitively, anyone concerned with a loading problem is inclined to place greater value on the satisfactory placement of a large object than a small one. This could be justified on the basis that a larger object, by definition, requires more space than a smaller one. It is important to maximize the opportunities for placement of a larger object; the number of opportunities is greater without the space limitations resulting from
prior placement of smaller objects. Alternatively, if a large object is placed first, the likelihood of finding an acceptable place for a follow-on object would be greater if the follow-on object were smaller. This intuitive idea is used for establishing the precedence guideline that, all other factors being equal, a larger box will always be selected and scheduled before a smaller one. The phrase, "all other factors being equal," is important. For example, if in certain instances it could be determined that selection of a smaller box in lieu of a larger one would ultimately result in less wasted space, then in this case, all other factors are not equal and the precedence relationship could be reversed. If, however, it could not be determined, a priori, which size selection would result in best overall space utilization, then the precedence guideline would be applicable and larger sizes would be assumed to have higher box size values.

Space utilization can be measured in terms of space utilization efficiency, which is defined as:

\[
\text{Space Utilization Efficiency} = \frac{\text{Volume of Space Utilized}}{\text{Volume of Available Space}} \times 100\%
\]

Although the objective is to maximize the space utilization efficiency at each phase of the loading process, sometimes it may be necessary to trade off space utilization efficiency against the value of the box sizes contained in a particular loading pattern. It may be advantageous to use for a large box a pattern having relatively high waste with the expectation of finding for a smaller size box a more efficient pattern.
which will give greater space utilization efficiency for the complete load.

In order to trade off box size value versus space utilization efficiency, a tolerance must be specified to denote the amount of space acceptable as waste, provided that sufficient box size value is achieved.

A mathematical technique, known as the integer linear combination technique, has been developed which combines the box size value and space utilization concepts. This technique can be used to accelerate and make decisions concerning the load solution process. In applying the technique, it is assumed that the precedence guideline always applies to the first box size to be selected. After selection of the largest box, successive selections will still pick the same size box if it is still available. Therefore, it becomes advantageous to determine first the maximum number of the largest size box that will fit in the available space and at the same time produce an acceptable configuration with respect to minimum waste. The remaining load population is then checked to determine whether the quantity is available, and, if so, all boxes of that size are simultaneously selected. If either the configuration or size availability precludes the maximum quantity assignment, then the assignment is reduced to the maximum available quantity that produces an acceptable configuration.

The technique can be used to solve two problem versions which depend on whether two sizes of a box $x_1$ and $x_2$ are known or if only box size $x_1$ is known, and the other size $x_2$ is unknown. The integer combination
applies to a one-dimensional space, since the items to be combined will always have a common dimension, and their combination will always be sought in the remaining one-dimensional space.

**Problem Version A**

A one-dimensional space of magnitude X is to be filled within tolerance TOL by integer linear combination of specified size $x_1$ and unknown size $x_2$. The availability of $x_1$ and $x_2$ is given by constants $n_1\text{AVAIL}$ and $n_2\text{AVAIL}$. Find maximum $n_1$ that satisfies:

\[ X - TOL \leq n_1 x_1 + n_2 x_2 \leq X \]

\[ n_1 \leq n_1\text{AVAIL} \]

\[ n_2 \leq n_2\text{AVAIL} \]

\[ n_1 > 0 \]

\[ n_2 > 0 \]

$n_1, n_2$ integer

**Problem Version B**

A one-dimensional space of magnitude X is to be filled within tolerance TOL by integer linear combination of specified sizes $x_1$ and $x_2$. The availability of $x_1$ and $x_2$ is given by constants $n_1\text{AVAIL}$ and $n_2\text{AVAIL}$.
Find maximum $n_1$ that satisfies:

$$X - TOL \leq n_1 x_1 + n_2 x_2 \leq X$$

$n_1 \leq n_{1\text{AVAIL}}$

$n_2 \leq n_{2\text{AVAIL}}$

$n_1 > 0$

$n_2 \geq 0$

$n_1, n_2$ integer

**Integer Linear Combination Technique**

Problem Version A is solved:

Step 1: Set $n_2 = 0$ and $n_1 = \min \left\{ \left\lfloor \frac{X}{x_1} \right\rfloor, \ n_{1\text{AVAIL}} \right\}$

where notation $\left\lfloor \cdot \right\rfloor$ denotes smallest integer $\leq$ number in brackets.

Step 2: Test if $n_1$ satisfies $X - TOL \leq n_1 x_1 \leq X$. If relation holds, then $n_1, n_2$ are optimum. Otherwise, go to Step 3.
Step 3: Let $x_{2\text{SMALL}}$ denote smallest available size of $x_2$.

If $x_{2\text{SMALL}}$ is void, go to Step 8.

Otherwise, set $n_2 = n_2 + 1$ and go to Step 4.

Step 4: Test if $n_1$, $n_2$ satisfy

$$\frac{X - n_1 x_1}{n_2} < x_{2\text{SMALL}}$$

where $x_{2\text{SMALL}}$ denotes smallest available size.

If this relation holds, go to Step 7.

Otherwise, go to Step 5.

Step 5: Solve for $x_{2\text{LOWER}}$ and $x_{2\text{UPPER}}$

where $x_{2\text{LOWER}} = \frac{X - n_1 x_1}{n_2}$

and $x_{2\text{UPPER}} = \frac{X - n_1 x_1}{n_2}$
Let \( \{x_2\} \) denote set of available sizes which satisfy

\[ x_{2\text{LOWER}} \leq x_2 \leq x_{2\text{UPPER}}. \]

If \( \{x_2\} \) is void, go to Step 3.

Otherwise, go to Step 6.

**Step 6:** Test if \( n_2 \leq n_{2\text{AVAIL}} \), where \( n_{2\text{AVAIL}} \) denotes cumulative quantity of all members of \( \{x_2\} \).

If relation holds, \( n_1 \) and \( n_2 \) are optimum and largest sizes belonging to \( \{x_2\} \) are to be used.

Otherwise, go to Step 3.

**Step 7:** Set \( n_1 = n_1 - 1 \)

and

\[ n_2 = 1. \]

If \( n_1 = 0 \), go to Step 8.

Otherwise, go to Step 4.

**Step 8:** STOP! Problem has no solution.
Problem Version B is solved:

Step 1: Set \( n_2 = 0 \), \( n_1 = \min \left\{ \left\lfloor \frac{X}{x_1} \right\rfloor , n_{1\text{AVAIL}} \right\} \)

\[ n_{2\text{MAX}} = \min \left\{ \left\lfloor \frac{X}{x_2} \right\rfloor , n_{2\text{AVAIL}} \right\} \]

Step 2: Test if \( X - \text{TOL} \leq n_1x_1 \leq X \). If it is, then \( n_1 \) and \( n_2 \) are optimum. Otherwise, go to Step 3.

Step 3: Set \( n_2 = n_2 + 1 \).

Test if \( n_2 > n_{2\text{MAX}} \). If it is, go to Step 5.

Otherwise, go to Step 4.

Step 4: Test if \( X - n_1x_1 - n_2x_2 < \text{TOL} \).

If relation holds, then \( n_1 \) and \( n_2 \) are optimum.

Otherwise, go to Step 3.

Step 5: Set \( n_1 = n_1 - 1 \) and \( n_2 = 0 \).

If \( n_1 = 0 \), go to Step 6.

Otherwise, go to Step 3.

Step 6: Stop! Problem has no solution.
The solution procedure ensures that unnecessary iterations are implicitly excluded when size availability becomes limited. For both problem versions, the iteration proceeds by initially allocating the maximum quantity to $n_1$ and the minimum to $n_2$. The procedure guarantees that $n_1$ and $n_2$ are integers.

In the process of generating and evaluating various loading patterns, criteria are needed to limit the types of loading patterns permitted. With sufficient time and financial expenditure, complicated, unusual loading patterns might be found to reduce wasted space; these solutions are ignored on the basis of practical considerations. The loading patterns sought are to be built up in a logical sequence of steps which favor uniformity of box size and regularity of configuration. The use of one size box is subject to its availability in the population or population subset, and also to the degree of success to be realized in the economical use of space. When it becomes necessary to mix sizes, usually a mix of only two sizes is permitted to keep the solution procedure simple and practical.

Regularity of configuration refers to configurations with parallel rows of boxes, parallel columns of boxes, and clusters of boxes.
INPUT REQUIREMENTS AND PRELIMINARY LOAD DATA ANALYSIS

To start the analysis, the user will input data on container size, pallet size, permissible pallet overhang, maximum allowable pallet loading height, batch size, barge type, various tolerances for different kinds of acceptable waste space, and, if applicable, reduced stack height for load accessibility.

The container size may be standard or non-standard. The solution technique will, in general, require that the user select the container size if more than one container size is available. The program will also be useful as a tool for selecting the container size by analytical repetition of the solution technique for various container sizes. The most efficient container size or sizes for the particular load on hand will be selected.

The pallet may be the normal 40-inch by 48-inch size or any other size. The permissible pallet overhang could extend the load size to as much as 43 inches by 52 inches for the normal pallet size. The program will have a stored table of overhang data extracted from MIL-STD-147B.³

The maximum allowable pallet loading height (including the pallet height) may be optionally specified by the user. Default values of 43 inches for SEAVAN and 41 inches for MILVAN as specified by MIL-STD-147B will be used.
The batch size is the maximum number of cargo boxes to be analyzed and scheduled for loading by the program. Before actual program development, it is difficult to estimate what range of batch sizes should be specified to obtain both economical program execution and favorable space utilization. Tests will be required after program development to determine optimum batch size for specific loading problems.

The barge type is specified to identify the quantity and arrangement of available space for cargo stowage. The program will contain stored data for the LASH and SEABARGE to characterize the available space. All available space will be subdivided or, where necessary, approximated in terms of rectangular parallelepipeds. For barges other than LASH or SEABARGE, the user will be responsible for describing the available space.

The initial data will be followed by input which describes in detail the characteristics of the load. It is assumed that the load will consist of boxes of random sizes. The diversity of available sizes is exhibited in the Federal Supply Catalog. For each box, the following information is needed:

- identification
- dimensions (specification of box cube instead of edge dimensions is unacceptable)
permissible orientation

- weight
- destination
- special value (optional)

The last characteristic will be used by the program to alter the sequence in which the load scheduling steps occur.

Preliminary load data analysis is performed to arrange the data in the most convenient form for carrying out the loading process. Rearrangement of the data can reduce the number of iterations required to reach a solution and can also reduce computer execution costs. Additional advantages will become evident as the loading procedure progresses.

Each batch of data is categorized by box heights. A list of box groups is generated so that all boxes in a group will have a common height, or heights which are judged to be close enough on the basis of an acceptable tolerance. For a group in which all boxes do not have the same height, it is assumed that shims will be used and the group height will be considered the largest box height in the group.

The list of box groups is rearranged so that the groups are ordered by decreasing group heights. When boxes in different groups are compared, the value of any member of one group is considered to be proportional to
the group height, that is, a group of large height has greater value than one of a smaller height. Since all members within each group have, by definition, a common (or common within tolerance) height, area becomes the measure of value of a box and a decreasing sequence of box cross-sectional areas provides the yardstick for determining the next available box of maximum value.

PALLET>Loading

After preliminary load data analysis, the next step in problem solution is to load pallets with minimum waste. Pallet loading progresses in three stages. The first stage, pattern formation, is fundamental and is concerned with automatically forming efficient patterns for loading tiers of boxes. The stacking stage consists of analytically determining the sequence in which tiers are to be stacked. The completion stage involves automatic insertion of pallets into the stacks, additional checks and adjustments resulting from weight and stability considerations, and generation of pallet loading reports.

Pattern Formation

The difficulties of working with fixed patterns have already been noted. The development which follows attempts to provide flexibility in pattern formation and to correlate the remaining box populations in a batch with the types of patterns to be used.
The basic building blocks in pattern formation are referred to as clusters. A cluster is a particular configuration of boxes considered beneficial for pattern building. Each cluster is developed from the box of highest value consistent with the available space constraints. This box is referred to as the box seed, or just the seed. The following definitions differentiate among the types of clusters to be used. For brevity, parentheses denote additional definitions. Examples of various cluster types are given in Figure 3.

A row (column) cluster, or just a row (column), has boxes arranged in the form of a row (column).

A complete row (column) is one which can be placed on a pallet, parallel to a side, so that any resultant wasted space is small, that is, within tolerance. The width (height) of a complete row (column) must be equal to or only slightly less than the pallet width (height).

An incomplete row (column) is one having width (height) not within tolerance requirements for completeness.

A partial row (column) is one which fills within tolerance an available row (column) space of smaller width (length) than the pallet. The width (height) of a partial row (column) is too small for the row (column) to be designated as complete.
Figure 3 - Example of Cluster Types
A uniform row (column) is constructed from boxes of only one size.

A non-uniform row (column) is constructed from boxes of two sizes.

A type 1 row (column) is one in which each box has a dimension which fills the height (width) of the row (column).

A type 2 row (column) is one in which at least one portion of the row (column) is completed by combining boxes to fill the height (width) of the row (column) within tolerance.

A modular cluster, or module, is a cluster obtained by combining partial rows and columns.

A symmetric modular cluster is a modular cluster having a symmetric arrangement of rows and columns, an exterior rectangular boundary, and an interior space, if any, which is rectangular.

A repeated symmetric modular cluster, or repeated cluster, is a cluster obtained by repetition of a symmetric modular cluster.

An asymmetric modular cluster is a modular cluster having a non-symmetric arrangement of rows and columns, no interior space, and an exterior boundary which is rectangular except for the possible presence of a corner rectangular space.

A cluster seed is a cluster which is considered to have the highest value of all clusters to be combined in the formation of a particular configuration.
Pattern formation progresses in an orderly sequence of steps. Although patterns are not fixed with respect to box dimensions, the types of configurations to be included in a pattern are fixed. Each pattern is constructed by beginning with box seeds and developing clusters. Cluster seeds are then combined with other clusters, according to predetermined rules, to form specific types of configurations. Patterns of simpler configuration are sought first, and if they cannot be used, more elaborate types are considered. Once again, specific rules will be enforced to limit the amount of work expended in seeking the desired result and also to eliminate unwarranted iterations. Pattern formation for all boxes in the loading population is not guaranteed. When failure occurs, the task will be returned to the analyst for further consideration.

Before even simple patterns are formed, complete rows and columns must be generated. The procedure for generating complete rows relies extensively on the integer linear combination technique already discussed.

**Row Completion Procedure**

*Step 1:* Select as the seed box the next remaining box of maximum value; let its side dimensions be \( a \) and \( b \) where \( a > b \).

*Step 2:* Try to generate uniform complete rows.

a. Attempt to form a type 1 uniform complete row of height \( a \), using an integer linear combination of \( b \). If unsuccessful,
and a and b have not been interchanged, then interchange them and repeat Step 2a; otherwise restore the original values of a and b and go to Step 2b.

b. Attempt to form a type 2 uniform complete row of height a by first obtaining an integer linear combination of b to generate columns within tolerance of height a and, if successful, taking integer linear combinations of a and b to complete the row. If unsuccessful, and a and b have not been interchanged in Step 2b, then interchange them and repeat Step 2b; otherwise restore the original values of a and b and go to Step 3.

Step 3: Try to generate non-uniform complete rows.

a. Attempt to form a type 1 non-uniform complete row of height a using an integer linear combination of dimensions b and d of two different boxes; dimension b is that of the seed box having known lateral dimensions a × b and d is unknown and belongs to a box having lateral dimensions a × d where d < b. If unsuccessful, and a and b have not been interchanged, interchange them and repeat Step 3a; otherwise restore the original values of a and b and go to Step 3b.
b. Attempt to form a type 2 non-uniform complete row of height $a$ by first obtaining an integer linear combination of $c$, where $c$ is unknown and $c < a$, to generate columns within tolerance of height $a$ and, if successful, taking integer linear combinations of two dimensions; dimension $b$ is that of the seed box having dimensions $a \times b$ and dimension $d$ is that of a box having dimensions $c \times d$ for which columns of height $a$ have been successfully generated. If unsuccessful, and $a$ and $b$ have not been interchanged in Step 3b, then interchange them and repeat Step 3b; otherwise, restore the original values of $a$ and $b$ and go to Step 4.

Step 4: Flag failure; a complete row cannot be constructed for box seed of size $a \times b$.

The generation of complete columns is analogous to that for complete rows with row and column dimensions reversed. A list of complete rows and a separate list of complete columns is generated with its members arranged in decreasing sequence of row heights and column widths.

Patterns are characterized as simple or complex, depending on the types of configurations from which they are constructed. (Configuration types differ from the previously defined row types.) Patterns synthesized
from a type 1 or type 2 configuration are simple; patterns derived from type 3, type 4, or combinations of these types are considered to be complex.

A type 1 configuration is developed by integer linear combination of either all complete rows or all complete columns. If the configuration has only complete rows, it is called type 1A, and if it consists of only complete columns, it is called type 1B. (If a configuration is identified as type 1A, it will be so designated and it will be immaterial whether that configuration simultaneously satisfies the criteria for type 1B.) The type 1 configuration is the first to be sought. Examples of a type 1 configuration are given in Figure 4.

The procedure for generating a type 1 configuration is almost identical to that previously described for the row completion procedure and that inferred for the column completion procedure. A type 1A configuration can be looked upon as a complete column derived by integer linear combination of boxes having width equal to the complete row width. The seed is the row having the largest row height. A type 1B configuration can be considered as an integer linear combination of boxes having length equal to the complete column height. The seed is the column having the largest column width.
TYPE 1A Configuration obtained by LINEAR combination of complete rows

TYPE 1B Configuration obtained by LINEAR combination of complete columns

Figure 4 - Type 1 Configuration
Type 1 Configuration Formation Procedure

Step 1: Attempt to exhaust the list of complete rows by formation of type 1A configurations using integer linear combinations.

Step 2: Attempt to exhaust the list of complete columns by formation of type 1B configurations using integer linear combinations.

Step 3: Make any remaining complete rows or columns available for subsequent type 2 configuration analysis.

A type 2 configuration is developed by either complete rows or complete columns followed by refinements for improving space utilization. A type 2A configuration is derived from complete columns. Examples of the type 2 configuration are given in Figure 5. The formation of the type 2 configuration places greater value on complete row and complete column clusters than on boxes which are not clustered. This priority of values helps to further reduce the number of unused complete rows and columns remaining after the type 1 configurations have been formed.

Type 2 Configuration Formation Procedure

Step 1: Use the list of unused complete rows and select in succession as many rows as possible without overfilling the available space.

Step 2: Try to reduce the unused pattern space by trading the next available unused complete row for any previously selected row, other than the seed row, having lesser height.
Acceptable waste

Type 2A configuration obtained by sequential access of complete row list

Type 2B configuration obtained by sequential access of complete column list

Figure 5 - Type 2 Configuration
Step 3: Repeat Step 2 twice if necessary. At this point, if the cluster seed has not generated an acceptable type 2A configuration, this type of configuration need no longer be pursued with the same seed.

Step 4: Repeat Steps 1 through 3 to obtain as many type 2A configurations as possible.

Step 5: Repeat Steps 1 through 4 using complete columns instead of complete rows to obtain as many type 2B configurations as possible.

The type 2 configuration formation procedure is followed by the type 3 configuration formation procedure, which attempts to synthesize an asymmetric modular cluster. The type 3A configuration is generated by exhausting the list of unused rows and alternately filling the remaining space with partial columns and rows. This type of configuration is illustrated in Figure 6.

In the previous type 1 and type 2 configuration formation procedures, a configuration was considered successful when it filled the available pallet space within tolerance. In that case, the configuration produced a complete pattern; for both types 1 and 2, it was advantageous to repeat the procedures to obtain as many complete patterns as possible.
Type 3A configuration obtained by alternate filling of rows followed by columns

Type 3B configuration obtained by alternate filling of columns followed by rows

Figure 6 - Type 3 Configuration
In the type 3 configuration formation procedure success does not necessarily imply full use of the available space. Rather, a promising configuration is further analyzed and possibly combined with other configurations to synthesize a pattern which produces efficient utilization of the available space. If a type 3 configuration is not within tolerance, the major portion of the wasted space will always be in one corner of the pattern. That corner space should be filled with a type 4 configuration before returning to a type 3 configuration procedure for further improvement or for beginning a new pattern. Type 3A and type 3B configurations are generated in an identical manner except that the type 3A formation begins by considering unused column space, and the type 3B first considers unused row space.

Type 3 Configuration Formation Procedure

Step 1: Exhaust any remaining complete rows (columns). If none, go to type 4 procedure; otherwise choose next available box seed that fits into remaining available space and go to Step 2 (Step 3).

Step 2: Try to form as many partial columns as possible to reduce waste width. If successful (that is, at least one partial column is found) and both corner waste and pattern waste are within tolerance, stop. If successful and corner waste is out of tolerance, go to Step 3; otherwise, go to Step 4.
Step 3: Try to form as many partial rows as possible to reduce waste height. If successful (that is, at least one partial row is found) and both corner waste and pattern waste are within tolerance, stop. If successful and corner waste is out of tolerance, go to Step 2; otherwise go to Step 4.

Step 4: If at least one success is achieved for type 3 (in either Step 2 or Step 3), flag the next box seed for type 4 as variable. If no success is achieved for type 3, flag the seed selected in Step 1 for retention; attempt further completion of the pattern by going to the type 4 configuration procedure.

The type 4 configuration, the most intricate type of configuration to be sought, is characterized by the presence of a symmetric modular cluster. There are three subtypes of this configuration. The first is formed by a symmetric integer linear combination of two sizes of boxes. An example of this subtype, type 4A, is given in Figure 7A. This type of combination represents two distinct linear combinations, one for each of the two dimensions of the available space. In this type of combination, both the height and width of the available space must be decreased and any resultant waste space must be confined to the interior of the original available space. If the resulting waste space is not within tolerance, then an attempt is made to further reduce the waste by returning to the type 3 configuration procedure and trying to insert a type 3 configuration...
Type 4A configuration fills outer portion of space in 2 dimensions

Type 4P configuration fills available space in only 1 dimension

Type 4C configuration does not fill available space in either dimension

Figure 7 - Type 4 Configuration
into the wasted space. An example of such nesting of type 3 and type 4 configurations is given in Figure 8.

The second subtype is similar to the first subtype except that, as a result of the symmetric integer linear combination, a symmetric modular cluster is obtained which reduces only one of the dimensions of the original available space. The second dimension is reduced by repetition, if possible, of this modular cluster. An example of this second subtype, type 4B, is shown without repetition in Figure 7B. The resultant waste space consists of an interior hole and a remainder wasted space. Filling of these spaces is attempted, once again, by a type 3 configuration.

The third subtype, a further generalization in configuration formation, is formed as a result of symmetric integer combination of two box sizes which does not fill either the height or width of the original available space. This type of configuration includes a further restriction that the resultant interior waste space must be so small that no additional interior fill is needed. Repetition of the modular cluster formed by this combination is sought in two dimensions to reduce the available space. Two rectangular remainder spaces may result, each parallel to an edge of the original space. An example of this subtype, type 4C without repetition, is given in Figure 7C.

Type 4 Configuration Formation Procedure

Step 1: Choose next suitable box seed.
Figure 8 – Pattern Completion by Nesting of Type 3 and Type 4 Configurations
Step 2: Try to obtain type 4A configuration by a symmetric integer linear combination of two box sizes which is as large as both the height and width of the available space. If successful, and within tolerance, stop. If successful, and out of tolerance, go to type 3 procedure (omitting Step 1) to further reduce waste; otherwise, go to Step 3.

Step 3: Try to obtain type 4B configuration by a symmetric integer linear combination of two box sizes which is as large as either the height or width of the available space. If successful, repeat as often as possible in the other direction. If successful and within tolerance, stop. If successful and out of tolerance, go to type 3 procedure (omitting Step 1) to further reduce waste; otherwise go to Step 4.

Step 4: Try to obtain type 4C configuration by a symmetric integer linear combination of two box sizes with a small interior hole. If successful, repeat as often as possible, to fill both the height and width of the available space. If successful and within tolerance, stop. If successful and out of tolerance, go to type 3 procedure (omitting Step 1) to further reduce waste; otherwise flag failure and stop.

The four types of configurations which have been described may be used to generate countless numbers of patterns. The actual patterns and
the specific box sizes will be governed by the available load population. All of the patterns in Appendix B can be generated. The sequence of steps in generating the various configurations gives priority to the formation of patterns which are uniform, that is, all boxes in a pattern will be of one size if sufficient numbers are available in the load population.

When a promising type 3 or type 4 configuration cannot be completed and the cluster seed cannot be efficiently loaded, then the analyst will be given the opportunity to intercede and possibly to improve the incomplete pattern. The same holds for any pattern which is incomplete and cannot be completed because the load population is exhausted.

Stacking

The stacking stage determines the sequence in which tiers of boxes, loaded in accordance with the previously formed patterns, will be stacked. (Refer to Figure 2, Step 4 for stacking terminology.) The height of the space available for stacking is the interior height of the container.

The stacking sequence is subject to the following additional constraints:

- height limitation of the load on an individual pallet
- allowance for heights of pallets to be inserted into stack
- allowance for necessary waste to permit accessibility
weight limitation of the load on an individual pallet

stability of the stack

The stacking procedure initially ignores the weight and stability constraints and adjusts for them later. Since it is advantageous to keep the number of items involved at one time as small as possible, the procedure used will develop subsequences for preliminary formation of individual pallet piles, preferably to their maximum permissible heights. These subsequences are then combined and modified to obtain a sequence that spans the total effective height. In this way, a full stack is obtained more quickly by stacking piles instead of individual tiers.

Even though the objective is to produce full preliminary pallet piles, the initial procedure is done rapidly with no attempt at refinements. Selected piles are stacked one upon the other and, when necessary, additional tiers not included in any piles are added to the stacks. The procedure emphasizes the formation of full stacks since, once they are obtained, it is immaterial whether two partially loaded pallets or a fully loaded pallet and a partially loaded pallet are used if in
both instances the two pallets spanned the same stack height, as shown in Figure 9.

The following definitions will be needed in the discussion of the stacking procedure:

- $\text{HSTACK} = \text{stack height of load including pallets}$
- $\text{HTFMAX} = \text{maximum permissible height of a full pile including pallet height}$
- $\text{HTPALL} = \text{pallet height}$
- $\text{HTCONT} = \text{effective height of container (interior height of container - waste height required for accessibility)}$
- $\text{TOLPIL} = \text{allowable tolerance on a full pile (minimum height of a full pile = HTFMAX - TOLPIL.} )$
- $\text{TOLSTK} = \text{allowable tolerance on a full stack}$

The minimum number of pallets, $\text{NMIN}$, required to span the effective container height is, therefore:

$$\text{NMIN} = \left\lceil \frac{\text{HTCONT}}{\text{HTFMAX}} \right\rceil$$

where $\lceil \cdot \rceil$ denotes next largest integer.

The easiest way to build a full stack with the minimum number of pallets is to stack $\text{NMIN} - 1$ full piles and a partially loaded pallet with a minimum height, $\text{HTPMIN}$, given by:

$$\text{HTPMIN} = \text{HTCONT} - (\text{NMIN} - 1) \times (\text{HTFMAX} - \text{TOLPIL})$$

provided

$$\text{HTPMIN} < \text{HTFMAX} - \text{TOLPIL}.$$ 

(If $\text{HTPMIN} > \text{HTFMAX} - \text{TOLPIL}$, simply stack $\text{NMIN}$ full piles.)
Figure 9 - Alternative Pallet Loads Spanning Identical Stack Height
As many full piles as possible are formed from tiers contained in a list of available tiers which is arranged by decreasing tier heights. Successive tiers are selected until either the list of available tiers is exhausted or the addition of the next tier will cause the pile to exceed the maximum permissible pallet height, HTFMAX. If the pile height, HTPILE, is sufficiently high to be considered full, i.e., if

\[ HTFMAX - .1TPALL - TOLPIL < HTPILE \leq HTFMAX - HTPALL \]

then the sequence of tiers constituting the full pile is saved by putting it on a list of full piles and updating the total number of full piles, NFULL. If the pile is not full, then a test is made to determine whether

\[ HTPMIN - HTPALL \leq HTPILE \leq HTFMAX - HTPALL - TOLPIL \]

If so, the sequence of tiers is saved on a list of partial piles, and the total number of partial piles, NPART, is updated. If, however, HTPILE < HTPMIN - HTPALL, then the tiers are placed on a special list of unused tiers having NUNUSE entries. Thus, the desired lists of full piles, partial piles, and unused tiers are generated without excessive expenditure of time and money in rearranging tiers.

The process for stacking piles and unused tiers is described in terms of system states which are defined on the basis of the number of entries in each of these lists. The system states are defined by the system conditions given in Table 1. The state transition matrix is given in Table 2. A check mark (✓) denotes that a transfer is possible between
TABLE 1 - SYSTEM STATE DEFINITIONS

<table>
<thead>
<tr>
<th>State</th>
<th>System Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NPART &gt; 0, NFULL &gt; NMIN-1</td>
</tr>
<tr>
<td>2</td>
<td>NPART = 0, NFULL &gt; NMIN</td>
</tr>
<tr>
<td>3</td>
<td>NPART &gt; 0, 0 &lt; NFULL &lt; NMIN</td>
</tr>
<tr>
<td>4</td>
<td>NPART = 0, NFULL &lt; NMIN</td>
</tr>
<tr>
<td>5</td>
<td>NFULL = 0, NPART &gt; NMIN</td>
</tr>
<tr>
<td>6</td>
<td>NFULL = 0, 0 &lt; NPART &lt; NMIN</td>
</tr>
<tr>
<td>7</td>
<td>NFULL = 0, NPART = 0, NUNUSE</td>
</tr>
</tbody>
</table>

TABLE 2 - STATE TRANSITION MATRIX*

<table>
<thead>
<tr>
<th>FROM</th>
<th>TO</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>√</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

*√ denotes state transition can occur
X denotes state transition cannot occur
the "from" and "to" states, and an "X" denotes that a transfer is not possible. The significance of states, state transitions, and their effect on the stacking procedure is explained as follows:

Under the system conditions for state 1, that is, $\text{NPART} > 0$, $\text{NFULL} > \text{NMIN} - 1$, the stack is built by stacking, in succession, the next available $\text{NMIN} - 1$ full piles and placing uppermost a single partially loaded pile. At this point, because of the manner in which full and partial piles were defined and generated, the stack must either be full

$$\text{HTCONT} - \text{TOLSTK} < \text{HSTACK} \leq \text{HTCONT}$$

or overfilled

$$\text{HSTACK} > \text{HTCONT}.$$

The height of the stack, $\text{HSTACK}$, is given by

$$\text{HSTACK} = \text{HTPART}(1) + \sum_{i=1}^{\text{NMIN} - 1} \text{HTFULL}(i) + \text{NMIN} \times \text{HTPALL}$$

where the first term on the right is the height of the partial pile, the second term represents the sum of the heights of the full piles, and the last term represents the space to be occupied by pallets for supporting the piles, provided that the stack contains the minimum number of pallets. At this stage of the loading process, it is convenient to set aside space for $\text{NMIN}$ pallets, but it must be emphasized that the hypothetical stack consists only of tiers which have been tentatively grouped into piles without their supporting pallets. The overfill height, $\text{HOVER}$, is given by

53
HOVER = HSTACK - HTCONT

In state 1, HOVER \( \geq 0 \) and an unloading procedure is needed to remove one or more tiers to make the stack height as close as possible to, but still below, the permissible stack height. Tier removal starts at the uppermost pile and, if an underfill condition within tolerance is not achieved, lower piles are tested until success is attained. The Nth pile is unloaded by removing the largest possible tier which will either create an underfill with the least waste, or, if an underfill is impossible, by removing the largest possible tier to create a stack with the least overfill.

Mathematically, for all I tiers in pile N with tier heights HTTIER \((N,I)\), compute

\[ HTTIER(N,I) - HOV(N) \]

where

\[ HOV(N) = HOVER \]

and select pattern J such that, for all I,

\[ 0 \leq HTTIER(N,J) - HOV(N) \leq HTTIER(N,I) - HOV(N). \]

If J is void, then an underfill cannot be obtained by removing just one tier and, therefore, a Kth tier must be temporarily removed so that:

\[ HTTIER(N,I) - HOV(N) \leq HTTIER(N,K) - HOV(N) < 0 \]

Once again tier J is sought to obtain the desired underfill. When the Jth tier is found, test whether

\[ HTTIER(N,J) - HOV(N) < TOLSTK. \]
If so, permanently remove the patterns selected for the desired underfill. If the patterns to be removed do not result in an acceptable underfill, then proceed to analyze the next lower pile. If all piles in the stack have been analyzed without finding one that fills the requirements, then use the best possible pile for unloading, even if the result is not within specified tolerance.

The type of logic that has been used is typical of the stacking procedure for all states. The specific procedure depends on the number of entries in each of the lists and the sequence in which the lists become exhausted. If, for example, the state 1 stacking procedure is repetitively applied and results in the exhaustion of the list of partial piles, that is, $\text{NPART} = 0$, then a transition of the system state would occur, and either state 2, 4, or 7 could result.

When all lists are available, the procedure gives priority to using, simultaneously, full piles and partial piles (state 1). If no more full piles are available, the procedure uses as many partial piles as possible before modifying the stack with individual tiers selected from the list of unused tiers. If the partial piles are exhausted before the full piles, then preference is given to using as many full piles as possible before accessing the list of unused tiers. In the absence of any full or partial piles (state 7) the complete stack is built with individual tiers.
In state 1, an overfull condition results most frequently, but in other states this is not necessarily the case. In state 7, for instance, individual tiers are selected and a height computation is made to ensure that an overfill condition never results. In the other states, either an overfill or underfill condition may initially result. If an underfill results, then a space filling procedure is required which consists of using the list of unused tiers and adding as many successive tiers as possible until the addition of the next tier causes an overfill. In other words, the space filling procedure reduces wasted space by approaching the acceptable full stack height from below and never exceeding the desired result. For all states, the common goal is to fit the stack into the available space and then reduce waste in a stepwise procedure which always results in improvement.

An additional refinement is available which, if it can be used, will always result in improvement. If a stack is underfilled and not within tolerance, and the space filling procedure has failed to put the stack within tolerance, the next available tier on the list of unused tiers is substituted for an existing tier in the stack. If the height of the next available tier is $H_{\text{NEW}}$, and the height of an existing tier in the stack is denoted as $H(I)$, then any existing tier in the stack which is a candidate for trading must satisfy the criterion

$$0 < H_{\text{NEW}} - H(I) \leq H_{\text{WASTE}}$$
where

\[
HWASTE = H_{\text{OVER}}
\]

The most desirable existing tier to be traded, \(J\), satisfies the criterion that

\[
0 < H_{\text{NEW}} - H(I) \leq H_{\text{NEW}} - H(J) \leq HWASTE.
\]

Additional details on stacking are given in the flowcharts of Appendix C. In certain states the list of unused tiers may be exhausted or nearly exhausted but individual tiers are needed to complete the stack under analysis. Under these circumstances, an available partial pile may be disassembled and its constituent patterns placed on the list of unused tiers. If no partial pile is available, an available full pile can be disassembled and handled in like manner. If a stack is too low and neither full piles, partial piles, or unused tiers are available, the entire load has been stacked.

Completion

The first step in the completion stage is the insertion of pallets into the previously formed stacks. This task is simplified by the procedure used to form a stack. If the stack contains any full piles, they will be at the bottom of the stack, and pallets are first inserted beneath each full pile.

If a stack contains both partial piles and individual tiers not assigned to any pile, the procedure could be to consolidate partial piles
and add tiers to the remaining partial piles, or to add tiers to partial piles and then consolidate. Since both methods have disadvantages, a procedure was developed which represents a mixture of these methods. If the number of full piles plus partial piles exceeds the minimum number of pallets required to load a full stack, partial piles are consolidated before any remaining tiers are loaded. After this initial consolidation, any additional tiers are added to these partial piles and, if all partial piles are exhausted, a new pile is started from the remaining tiers on the list of unused tiers. At no time is a partial pile permitted to become overstacked. When this list is exhausted, all piles except the original full piles are consolidated and a pallet is inserted beneath each remaining pile. In the rare instance in which the number of pallets exceeds the minimum that could be used and, simultaneously, the resulting stack height after pallet insertion is excessive, the most convenient tier having a height which exceeds the overstack is removed and placed on the lowest remaining stack. Each stack is then analyzed by the same procedure until pallets have been appropriately placed into all stacks.

The weight of each pallet and its stability are considered next. The computer is ideally suited for monitoring the total weight of all boxes on a pallet. If a pallet's weight is excessive, the situation is remedied by removing part of its load, beginning at the topmost tier of boxes. Any boxes so removed are rescheduled for loading on another
pallet. To increase the stability of a pallet, tiers placed on a pallet are rearranged in decreasing order of size starting at the bottom of a pallet. This ensures that upper tiers do not excessively overlap lower tiers and also facilitates pallet load strapping and prevents load shifting. In stacking tiers formed repetitively from the same pattern, the rotation of successive tiers further stabilizes a pallet load. Figure 10 shows the improvement of pallet load stability; tier 3, with the same pattern as tier 2, is rotated 180 degrees and tier 1, with a smaller area than the other two tiers, is placed on top.

Another program feature for enhancing load stability will provide information on necessary interior supports to be placed in interior spaces of tiers derived from patterns having interior wasted spaces; this prevents load shifting.

The final task of the completion stage generates an output report which describes in detail all patterns, tier sequencing on each pallet, and the space utilization measures of each pallet and of each stack.

CONTAINER LOADING

After completion of pallet loading, an analytic procedure will provide an efficient spatial arrangement of the stacks for each size container, monitor the total weight of the total load in each container, and provide for load stability. The tasks associated with this procedure are collectively termed "container loading."
Figure 10 - Improvement of Pallet Load Stability
All the essential features of the container loading procedure become available as by-products of the previously developed pallet loading procedure. If all pallets are the same size and if a stack is temporarily considered as one tall box, then determining the arrangement of stacks is really the same problem as finding a pattern for arranging boxes of identical size. For this purpose, the type 1 configuration formation procedure previously described is most suitable. For the present application, however, the procedure is more restrictive, since, if only one size pallet is used, only uniform rows and columns are generated, and the resulting configuration is considerably simpler.

The total weight of the load in each container is obtained by simply summing the weights of the stacks. The stability of a container can be improved by interchanging stacks to offset imbalance of the load resulting, for example, when heavier stacks predominate on one side of a container. A report of the container load describing in detail the arrangement of stacks and the space utilization measure of each container will also be generated.

**BARGE LOADING**

The procedures for arrangement of containers on a barge and for checking weight and stability are virtually identical to the container loading procedure. If containers of more than one size are to be loaded, the type 1 pattern to be used for their arrangement need not necessarily
be derived from uniform rows and columns. Each rectangular space on the barge available for loading will be analyzed independently. A final report will be generated denoting the arrangement of containers, the total weight, and the overall space utilization efficiency measured with respect to the total available space on the barge.
ORGANIZATION AND FUNCTIONAL DESCRIPTION OF PROPOSED PROGRAM LOADER

The organization of main program LOADER, its subroutines, and their interrelationships is provided in Appendix C. A flowchart of the main program is given in Figure 13, and flowcharts for most of the subroutines appear in Figures 14-32. A brief description of the function of each routine follows.

LOADER - main program; performs executive role of controlling, sequencing, and monitoring program operations.

INPUT - reads input data and generates input data report.

ANALYZ - analyzes input data and rearranges data into most convenient form.

PATRNS - forms patterns by generating and combining clusters; assigns patterns to tiers.

STRING - generates complete row and complete column clusters.

TYPE 1 - generates type 1 configurations.

TYPE 2 - generates type 2 configurations.

TYPE 3 - generates type 3 configurations.

TYPE 4 - generates type 4 configurations.

SAVE - creates a file containing data describing complete patterns.

STACKS - controls formation of stacks generated by stacking full piles, partial piles, and unused tiers.

PILES - generates preliminary full piles and partial piles.

STATE - determines the system loading state from the available number of preliminary full piles, partial piles, and unused tiers.
STAKER - forms stacks from full piles, partial piles, and unused tiers; stacking strategy used depends upon system loading state.

SUPPLY - checks and makes available, if possible, a specified minimum number of unused tiers.

FILL - fills available wasted space in a stack by rapid selection of unused tiers.

UNLOD - removes tiers from stacks which are overloaded.

TRADE - attempts refinement of available wasted space in a stack by trading tiers.

PALETS - forms complete pallet loads by consolidating piles, inserting pallets beneath piles, and making final stack adjustments, if necessary.

WEIGHT - monitors weight constraints in stacks, containers, and barges and assists in corrective action.

STABLE - monitors stability in stacks, containers, and barges and assists in corrective action.

CNTNER - controls container loading operations.

BARGE - controls barge loading operations.

REPRT1 - generates report of composition of all stacks including the arrangement of all boxes in each tier.

REPRT2 - generates report on arrangement of stacks in each container.

REPRT3 - generates report on arrangement of containers in barge.
SAMPLE PROBLEM

STATEMENT OF PROBLEM

Containers #1 and #2 are partially loaded with cargo for destinations A and B, respectively. Cargo for container #1 is restricted to a maximum loaded pallet height of 48 inches and for container #2 to 36 inches. Additional cargo for destination C is to be loaded completely on either container #1 or #2. The following additional load data apply:

Pallet dimensions: 48" W X 40" L

Permissible pallet dimensions with overhang: 52" W X 43" L

Pallet height: 6"

Available space in container #1:

- Effective height - 90"
- Width - 90"
- Length - 54"

Available space in container #2:

- Effective height - 84"
- Width - 90"
- Length - 54"
Characteristics of cargo for destination C:

<table>
<thead>
<tr>
<th>Box Dimensions (Inches)</th>
<th>Quantity</th>
<th>Box Dimensions (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>L X W X H</td>
<td></td>
<td>L X W X H</td>
<td></td>
</tr>
<tr>
<td>28 X 16 X 16</td>
<td>2</td>
<td>24 X 21 X 15</td>
<td>2</td>
</tr>
<tr>
<td>42 X 12 X 14</td>
<td>1</td>
<td>13-1/2 X 9 X 12</td>
<td>8</td>
</tr>
<tr>
<td>37 X 12 X 14</td>
<td>2</td>
<td>18 X 6 X 8</td>
<td>4</td>
</tr>
<tr>
<td>36 X 14 X 8</td>
<td>2</td>
<td>15 X 15 X 10</td>
<td>4</td>
</tr>
<tr>
<td>45 X 38 X 10</td>
<td>1</td>
<td>11 X 11 X 11</td>
<td>8</td>
</tr>
<tr>
<td>28 X 6 X 12</td>
<td>4</td>
<td>11 X 7-1/2 X 15-1/2</td>
<td>8</td>
</tr>
<tr>
<td>28 X 22 X 14</td>
<td>2</td>
<td>8 X 6 X 16</td>
<td>6</td>
</tr>
<tr>
<td>26 X 14 X 16</td>
<td>1</td>
<td>8 X 6 X 10</td>
<td>6</td>
</tr>
<tr>
<td>26 X 12 X 16</td>
<td>3</td>
<td>15 X 15 X 14</td>
<td>4</td>
</tr>
<tr>
<td>22 X 11 X 10</td>
<td>4</td>
<td>14 X 12 X 18</td>
<td>12</td>
</tr>
<tr>
<td>22 X 15 X 12</td>
<td>4</td>
<td>18 X 14 X 14</td>
<td>2</td>
</tr>
<tr>
<td>20 X 14 X 8</td>
<td>2</td>
<td>6 X 6 X 6</td>
<td>8</td>
</tr>
</tbody>
</table>
The cost of shipment by container #2 is less than for #1 and it is therefore preferable, if possible, to use container #2. The problem is to determine analytically whether the cargo can be loaded in each of the available spaces, to compare their space utilization, and to examine the effect of the more restrictive maximum loaded pallet height in container #2.

SOLUTION

Assumptions and Restrictions

1. Boxes must be loaded with height dimensions oriented vertically.
2. Group height tolerance for box height, h:

<table>
<thead>
<tr>
<th>Height Range</th>
<th>Tolerance (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>h &gt; 40</td>
<td>5</td>
</tr>
<tr>
<td>30 &lt; h ≤ 40</td>
<td>4</td>
</tr>
<tr>
<td>20 &lt; h ≤ 30</td>
<td>3</td>
</tr>
<tr>
<td>10 &lt; h ≤ 20</td>
<td>2</td>
</tr>
<tr>
<td>h &lt; 10</td>
<td>1</td>
</tr>
</tbody>
</table>

3. Cluster waste tolerances:
   10% for all row and column clusters
   10% for holes in symmetric modular clusters
   10% for corner waste in asymmetric modular clusters
   15% for outerfill waste of symmetric modular clusters

4. Pattern waste tolerances:
   10% for simple patterns (types 1 and 2)
15% for complex patterns (types 3, 4, or combinations)

Stack tolerances:

3" for stack height waste
6" for preliminary full piles

Preliminary Load Data Analysis

The maximum box height is 18" and the minimum is 6". Define groups by range of box heights, h:

<table>
<thead>
<tr>
<th>Group</th>
<th>Height Range (Inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16 &lt; h ≤ 18</td>
</tr>
<tr>
<td>2</td>
<td>14 &lt; h ≤ 16</td>
</tr>
<tr>
<td>3</td>
<td>12 &lt; h ≤ 14</td>
</tr>
<tr>
<td>4</td>
<td>10 &lt; h ≤ 12</td>
</tr>
<tr>
<td>5</td>
<td>9 &lt; h ≤ 10</td>
</tr>
<tr>
<td>6</td>
<td>8 &lt; h ≤ 9</td>
</tr>
<tr>
<td>7</td>
<td>7 &lt; h ≤ 8</td>
</tr>
<tr>
<td>8</td>
<td>6 &lt; h ≤ 7</td>
</tr>
</tbody>
</table>

The load population consists of 100 boxes which will be analyzed collectively, that is, batch size = 100. The following table is obtained by grouping the batch data, computing the box areas and volumes, sequencing the groups by decreasing group heights, and sequencing the boxes within each group by decreasing areas.
### TABLE 3 - SAMPLE PROBLEM LOAD POPULATION DATA

<table>
<thead>
<tr>
<th>Group Height (inches)</th>
<th>Box Dimensions L X W X H</th>
<th>Box Area (square inches) L X W</th>
<th>Box Volume (cubic inches) L X W X H</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 18</td>
<td>14 X 12 X 18</td>
<td>168</td>
<td>3024</td>
<td>12</td>
</tr>
<tr>
<td>2 16</td>
<td>24 X 21 X 15</td>
<td>504</td>
<td>7560</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>28 X 16 X 16</td>
<td>448</td>
<td>7168</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>26 X 14 X 16</td>
<td>364</td>
<td>5824</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>26 X 12 X 16</td>
<td>312</td>
<td>4992</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>11 X 7-1/2 X 15-1/2</td>
<td>82.5</td>
<td>1279</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>8 X 6 X 16</td>
<td>48</td>
<td>768</td>
<td>6</td>
</tr>
<tr>
<td>3 14</td>
<td>28 X 22 X 14</td>
<td>616</td>
<td>8624</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>42 X 12 X 14</td>
<td>504</td>
<td>7056</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>37 X 12 X 14</td>
<td>444</td>
<td>6216</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>18 X 14 X 14</td>
<td>252</td>
<td>3528</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15 X 15 X 14</td>
<td>225</td>
<td>3150</td>
<td>4</td>
</tr>
<tr>
<td>4 12</td>
<td>22 X 15 X 12</td>
<td>330</td>
<td>3960</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>28 X 6 X 12</td>
<td>168</td>
<td>2016</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>13-1/2 X 9 X 12</td>
<td>121.5</td>
<td>1458</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>11 X 11 X 11</td>
<td>121</td>
<td>1331</td>
<td>8</td>
</tr>
</tbody>
</table>

* Group Height is maximum box height of all boxes in a group.
<table>
<thead>
<tr>
<th>Group Height (inches)</th>
<th>Box Dimensions (L X W X H)</th>
<th>Box Area (square inches)</th>
<th>Box Volume (cubic inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>45 X 38 X 10</td>
<td>1710</td>
<td>17100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 X 11 X 10</td>
<td>242</td>
<td>2420</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 X 15 X 10</td>
<td>225</td>
<td>2250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 X 6 X 10</td>
<td>48</td>
<td>480</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>36 X 14 X 8</td>
<td>504</td>
<td>4032</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 X 14 X 8</td>
<td>280</td>
<td>2240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 X 6 X 8</td>
<td>108</td>
<td>864</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>6 X 6 X 6</td>
<td>36</td>
<td>216</td>
</tr>
</tbody>
</table>

* Group Height is maximum box height of all boxes in a group.

Pallet Loading

Each group is processed independently to form patterns; intermediate diagrams are provided for instructional purposes but are unnecessary for carrying out any of the procedures.
Group 1 Processing

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L X W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>14 X 12</td>
<td>12</td>
</tr>
</tbody>
</table>

Group 1 contains only one box size, and the box seed is, therefore, 14 X 12. Apply row completion procedure.

Step 1: a = 14, b = 12

Step 2: Generate uniform complete rows.

Try to form a type 1 uniform complete row of height 14 using an integer linear combination of 12. First iteration of integer linear combination technique (problem version A in section entitled "Proposed Approach") applied by considering X₁ = b = 12 and X₂ unknown, n₁AVAIL = 12, gives n₁ = min \( \frac{52}{12} \), 12 = 4, n₂ = 0. Since complete row tolerance is 10% and maximum permitted pallet load width = 52, TOL = .10(52) = 5.2.

Substituting in

\[ X - TOL \leq n_1 x_1 \leq X \]

\[ 52 - 5.2 \leq 4(12) \leq 52 \]

This statement is valid, n₁ = 4 and n₂ = 0 are optimum, and a successful type 1 uniform complete row is obtained.
Row waste = \frac{\text{row waste area}}{\text{available row area}} = \frac{4 \times 14}{52 \times 14} \times 100 = 7.6\%.

Since there are eight more boxes of the same size, two additional complete rows of the same configuration may be immediately generated. There are no remaining boxes in group 1, so proceed to pattern formation and begin with type 1 configuration.

**Step 1:**

All complete rows have the same row height, so the cluster seed has height = 14. Try to form a type 1A configuration by integer linear combination (problem version A) of \(x_1 = 14\), \(x_2\) unknown, \(n_{\text{AVAIL}} = 3\), which immediately gives \(n_1 = 3\), \(n_2 = 0\) and the following acceptable complete pattern is obtained:

\[
\begin{array}{cccc}
12 & 12 & 12 & 12 \\
14 & 14 & 14 & 14 \\
1 &  &  &  \\
\end{array}
\]

**Pattern No. 1**

Pattern efficiency = \frac{\text{actual load area}}{\text{maximum pallet load area}} = \frac{12 \times 168}{43 \times 52} \times 100 = 90.2\%.

Pattern waste = 100 - 90.2 = 9.8\%.

Group 1 is exhausted; go to next group.
Group 2 Processing

Group 2 Box List

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L X W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>24 X 21</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>28 X 16</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>26 X 14</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>26 X 12</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>11 X 7-1/2</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8 X 6</td>
<td>6</td>
</tr>
</tbody>
</table>

The box seed is 24 X 21. Apply the row completion procedure.

Step 1:

\[ a = 24; \quad b = 21. \]

Step 2:

Try to generate a type 1 uniform complete row of height 24. First iteration of integer linear combination technique gives \( n_1 = 2, n_2 = 0 \) but the row has excessive waste.
Row waste = \( \frac{11}{53} \times 100 = 21.2\% \).

Interchanging a and b gives \( n = 2 \), \( n_0 = 0 \). The following acceptable \( 1 \) \( 2 \) complete row is generated and saved on list of complete rows.

\[
\begin{array}{ccc}
21 & 24 & 24 & 4 \\
\end{array}
\]

Row waste = \( \frac{4}{52} \times 100 = 7.69\% \).

The next available seed is 28 X 16. Attempts to obtain either a type 1 or type 2 uniform complete row by applying steps 1 and 2 of row completion procedure fail; likewise, an attempt to form a type 1, non-uniform row in step 3a fails. Step 3b gives a non-uniform row of height 28 by selecting the 14 X 24 size box and generating a column by a combination of 14-inch dimensions and the complete row by combination of the 16-inch dimension of the seed and 24-inch dimension of the selected box. The row is shown here, but its waste is excessive.

\[
\begin{array}{ccc}
16 & 24 & 12 \\
28 & 14 & 14 \\
\end{array}
\]

Row waste = \( \frac{12}{52} \times 100 = 23.1\% \).
Interchanging \(a\) and \(b\), so that \(a = 16, b = 24\), gives a successful type 2 non-uniform complete row which is put on list of complete rows.

\[
\begin{array}{cccc}
28 & 11 & 11 & 2 \\
16 & 7-1/2 & 7-1/2 & 7-1/2 & 7-1/2 \\
\end{array}
\]

Column waste = \(\frac{1}{16} \times 100 = 6.25\%\).

Row waste = \(\frac{2}{52} \times 100 = 3.85\%\).

Since only one of the two available \(28 \times 16\) boxes has been used, the next available seed is once again \(28 \times 16\). Sufficient \(11 \times 7-1/2\) boxes are available for immediate repetition of the last row cluster, and this row is placed on the list of complete rows.

Next available seed is \(26 \times 14\). Formation of a uniform complete row fails, but step 3a of the row-completion procedure provides an acceptable type 1 non-uniform complete row of height 26 inches by combining the 14-inch dimension of the seed and the 12-inch dimension of the \(26\)-inch \(\times\) \(12\)-inch size box selected by the integer linear combination technique.

\[
\begin{array}{cccc}
14 & 12 & 12 & 12 & 2 \\
26 & & & & \\
\end{array}
\]

Row waste = \(\frac{1}{52} \times 100 = 3.85\%\).
The box list now consists only of 8 X 6 boxes. Using a seed of 8 X 6, a uniform type 1 complete row is quickly obtained to exhaust the box list for group 2.

8 8 8 8 8 8 8 4

Row waste = \( \frac{4}{52} \times 100 = 7.69\% \).

The row list now consists of five complete rows, rearranged by decreasing row height as follows:

<table>
<thead>
<tr>
<th>Row #</th>
<th>Cluster Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>364 + 3(312) = 1300</td>
</tr>
<tr>
<td>2</td>
<td>1008</td>
</tr>
<tr>
<td>3</td>
<td>778</td>
</tr>
</tbody>
</table>

76
The row cluster seed has height 26 inches. Using the type 1 configuration formation procedure gives the following combination of rows 1 and 3:

Pattern efficiency $= \frac{1300 \times 778}{43 \times 52} \times 100 = 92.9\%$.

The next row cluster seed has row height = 21. The type 1 procedure can not produce a pattern with acceptable waste, but step 1 of the type 2 procedure immediately provides a satisfactory pattern.
Row 

\[ 24 \times 24 \times 4 \]

\[ 2 \times 21 \]

\[ 28 \times 11 \times 11 \]

\[ 4 \times 16.5 \]

Pattern efficiency = \[ \frac{1008 + 778 + 288}{43 \times 52} \times 100 \] = 93.2%.

Group 2 is finished.

Group 3 Processing

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L x W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>28 x 22</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>42 x 12</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>37 x 12</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>18 x 14</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>15 x 15</td>
<td>4</td>
</tr>
</tbody>
</table>

Seed is 28 inches x 22 inches. All efforts to obtain any type of complete row or complete column fail. Retain the 28 x 22 boxes and
use the next available seed, 42 X 12. Again, no acceptable complete row can be obtained, but the following acceptable type 1 uniform complete column is obtained and is saved on the list of complete columns.

\[ \text{Column waste} = \frac{1}{43} \times 100 = 2.3\% . \]

Next seed is 37 inches X 12 inches. No satisfactory complete rows or columns can be constructed and the 37 X 12 box is retained on the box list. The same situation occurs with the two remaining seeds on the list.

The column cluster has greater value than any individual box, but efforts to complete a pattern with the column cluster seed by the type 1 and type 2 procedures fail because there are not enough column clusters. Applying steps 1 and 3 of the type 3 procedure gives excessive corner waste.

\[ \text{Corner waste} = \frac{18 \times 40}{42 \times 40} \times 100 \]
\[ = 42\% . \]
Applying step 2 satisfactorily completes this pattern by the addition of partial columns.

Corner waste =
\[
\frac{4 \times 14}{40 \times 14} \times 100 = 10\%.
\]

Pattern efficiency =
\[
\frac{504 + 2(444) + 2(252)}{43 \times 52} \times 100 = 85\%.
\]

PATTERN NO. 4

Since the complete row and complete column lists are empty, the next seed is obtained from the box list and is 28 x 22. In the absence of complete rows and/or columns, the type 1, 2, and 3 procedures all fail. Applying the type 4 procedure gives a type 4A configuration which is successful (success measured by outerfill efficiency), requires no interior filling (fill measured by innerfill efficiency), and is a complete pattern (completeness measured by pattern efficiency).

PATTERN NO. 5
Outerfill efficiency = \( \frac{\text{actual load area including inner waste area}}{\text{maximum available load area}} \times 100 \)

= \( \frac{(22 + 15 + 15) \times (28 + 15)}{43 \times 52} \times 100 = 100\% \).

Outerfill waste = 100\% - outerfill efficiency = 0.

Hole waste = \( \frac{\text{inner waste area}}{\text{actual load area including inner waste area}} \times 100 \)

= \( \frac{8 \times 13}{(22 + 15 + 15)(28 + 15)} \times 100 = 4.65\% \).

Pattern efficiency = \( \frac{\text{actual load area}}{\text{maximum pallet load area}} \)

= \( \frac{2(616) + 4(225)}{43 \times 53} \times 100 = 95.35\% \).

Group 3 is exhausted.

Group 4 Processing

<table>
<thead>
<tr>
<th>Group 4 Box List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group Height (Inches)</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

No complete rows or columns can be formed using any of the boxes as a seed, and, consequently, type 1, 2, and 3 configurations cannot be
constructed. Applying the type 4 procedure with a seed of 22 X 15 gives a successful type 4B configuration.

\[
\text{Outerfill efficiency } = \frac{(28 + 22)(15 + 6)}{52(15 + 6)} \times 100 = 96.2. 
\]

Outerfill waste = 100 - 96.2 = 3.8%.

\[
\text{Hole waste } = \frac{9 \times 6}{(28 + 22)(15 + 6)} \times 100 = 5.14\%.
\]

Repetition of the configuration leads immediately to a complete pattern.
Pattern efficiency = \frac{2[2(15)(22) + 2(28)(6)]}{43 \times 52} \times 100 = 89.1\%.

Next seed is 13-1/2 X 9; type 4 procedure leads to a type 4C configuration which becomes, after repetition in two directions, a complete pattern.

Hole waste = \frac{2(2-1/2)}{20(24-1/2)} \times 100 = 1.02\%.

Outerfill efficiency = \frac{2(13-1/2 + 11) (2(9 + 11))}{43 \times 52} \times 100 = 87.7\%.

Outerfill waste = 100 - 87.7 = 12.3\%.

Pattern efficiency = \frac{8(121.5) + 8(121)}{43 \times 52} \times 100 = 86.8\%.

Group 4 is exhausted.
Group 5 Processing

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L X W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>45 X 38</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>22 X 11</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>15 X 15</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>8 X 6</td>
<td>6</td>
</tr>
</tbody>
</table>

Only one row may be completed and this occurs when the seed is 8 X 6; no complete columns are obtainable.

\[
\begin{array}{cccccccc}
8 & 8 & 8 & 8 & 8 & 8 & 4 \\
\end{array}
\]

Row waste = \( \frac{4}{52} \times 100 = 7.69\% \).

Pattern completion with the type 1 and type 2 procedures fails because of insufficient complete rows or columns. Applying the type 3 procedure provides no acceptable partial column, so a complete pattern cannot be made. The incomplete pattern consists of only a single row cluster.

The next available seed that fits in the reduced available space, 37 inches by 52 inches, is a box 22 inches by 11 inches. Application of the type 4 procedure immediately leads to a successful type 4A
configuration which, combined with the type 3 row contribution, results in a complete pattern.

PATTERN NO. 8
Pattern efficiency = \frac{6(48) + 4(242) \times 4(225)}{43 \times 52} \times 100 = 96.4\%.

The only item remaining in group 5 is the one 45- by 38-inch box. No acceptable cluster can be formed, so the 45- by 38-inch box is placed on the list of remaining boxes.

Group 6 Processing
This group is empty.
Group 7 Processing

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L X W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>36 X 14</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>20 X 14</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>18 X 6</td>
<td>4</td>
</tr>
</tbody>
</table>

No complete rows or columns are obtainable. Use of the 36- by 14-inch seed leads to a successful type 4A configuration with an incomplete pattern.

\[
\text{Hole waste} = \frac{22 \times 30}{(14 + 2(18))(36 + 6)} \times 100 = 31.43\%.
\]
An innerfill is required and is obtained by returning to the type 3 procedure and inserting a partial column composed of two 20- by 14-inch boxes.

Pattern efficiency = \( \frac{2(504) + 4(1080 + 2(280))}{43 \times 52} \times 100 = 89.4\% \)

Group 8 Processing

<table>
<thead>
<tr>
<th>Group Height (Inches)</th>
<th>Box Lateral Dimensions L X W (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6 X 6</td>
<td>6</td>
</tr>
</tbody>
</table>
There are too few boxes to form any acceptable cluster; these boxes are put on list of remaining boxes.

**Remainder Group Processing**

<table>
<thead>
<tr>
<th>Remainder Group Height (Inches)</th>
<th>Box Dimensions L X W X H (Inches)</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>45 X 38 X 10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6 X 6 X 6</td>
<td>6</td>
</tr>
</tbody>
</table>

The seed box is 45 inches by 38 inches and a non-uniform type 2 row may be quickly constructed by the row completion procedure.

Since no clusters or boxes remain, the row is used as an acceptable incomplete pattern with a flag denoting that the load population is exhausted.
Stacking and Completion

Ten patterns have been completed. For the relatively small number of boxes which have been arranged, no pattern repetition has occurred so that each unique pattern can be assigned to a tier. The height of a tier is equal to the height of the group from which the pattern was derived. Arranging the tiers in decreasing order of tier heights gives the following initial sequence: 18, 16, 16, 14, 14, 12, 12, 10, 10, 8.

The following correlation for patterns, tiers, and tier heights is provided to monitor the composition of the stacks which result from tier rearrangement.

<table>
<thead>
<tr>
<th>PATTERN NO.</th>
<th>TIER NO.</th>
<th>TIER HEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>
Container #1

Using the notation and formulas previously developed gives:

\[ \text{HTCONT} = 90'' \]
\[ \text{HTFMAX} = 48'' \]
\[ \text{HTPALL} = 6'' \]
\[ \text{TOLPIL} = 6'' \text{ (by assumption)} \]
\[ \text{TOLSTK} = 3'' \text{ (by assumption)} \]

\[ N_{\text{MIN}} = \left| \frac{\text{HTCONT}}{\text{HTFMAX}} \right| = \left| \frac{90}{48} \right| = 2 \]

The acceptable height range of full piles is given by:

\[ \text{HTFMAX} - \text{HTPALL} - \text{TOLPIL} < \text{HTPILE} \leq \text{HTFMAX} - \text{HTPALL} \]
\[ 48 - 6 - 6 < \text{HTPILE} \leq 48 - 6 \]

The full pile height range is (36, 42].

\[ \text{HTPMIN} = \text{HTCONT} - (N_{\text{MIN}} - 1) \times (\text{HTFMAX} - \text{TOLPIL}) \]
\[ = 90 - (2 - 1) \times (48 - 6) \]
\[ = 48 \]

Since 48 > \text{HTFMAX} - \text{TOLPIL}, no parti. piles can be formed.

When successive tier heights from the initial sequence are added until the addition of the next height would exceed the maximum permitted full pile height (42 inches), the following partitioning of the sequence into full piles and unused tiers is obtained:
Unused Tiers since HTPILE = 18 + 16 = 34 ≤ 36
18

18 + 16 + 16 = 50 > 42
16

Unused Tiers since 16 + 14 = 30 < 36
16

16 + 14 + 14 = 44 > 42
14

Full Pile since 36 < 14 + 12 + 12 = 38 ≤ 42
12

Unused Tiers since 10 * 10 * 3 = 28 ≤ 36
10

Unused Tiers since 10 * 10 * 3 = 28 ≤ 36
10

12

10

8

NMIN = 2
NFULL = 1
NPART = 0
NUSED = 9

Since NPART = 0, NFULL ≤ NMIN, Table 1 shows that the system is in state 4.

The state 4 stacking procedure forms a stack from all remaining full piles, in this instance only 1.

<table>
<thead>
<tr>
<th>Initial Stack #1</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>Full Pile</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
CARGO LOADING--A PROPOSED APPROACH FOR MAXIMIZING SPACE UTILIZATION--ETC(U)

SEP 82 G. GERSTEL

DTNSROC-82/099
The height of the stack \( HSTACK \) is given by

\[
HSTACK = \sum_{\text{all}} \text{HTFULL}(I) + \text{NMIN} \times \text{HTPALL} \\
= 14 + 12 + 12 + 2 \times 6 = 50
\]

An initial underfill condition exists since \( \text{HOVER} = - \text{HWASTE} = \text{HSTACK} - \text{HTCONT} = 50 - 90 - 40 \). The wasted space in the stack is filled by adding unused tiers to the top of the stack until the addition of the next tier would overfill the stack. The stack then becomes

<table>
<thead>
<tr>
<th>Modified Stack #1</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Tiers</td>
<td>16</td>
</tr>
<tr>
<td>18 Tiers</td>
<td>14</td>
</tr>
<tr>
<td>14 Full Pile</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

\( HSTACK = 50 + 16 + 18 = 84 \)

\( HWASTE = HTCONT - HSTACK = 90 - 84 = 6 \)

Since stack tolerance \( TOLSTK = 3 \) and \( HWASTE > TOLSTK \), additional stack modification is necessary.

The next stack refinement consists of trading the next entry on the list of unused tiers, if possible, for another tier in the stack to reduce the stack waste. The next entry is 16. Since the heights of all
individual tiers $\geq 16$, the first pile beneath these tiers is considered. Since $H_{\text{WASTE}} = 6$, the most desirable tier in the pile to trade for is a tier of height $= 12$. (If more than one tier has the same height, the uppermost tier will always be removed.) Before the trade is permitted, a check is made to ensure that the upper limit of the full pile height, 42, would not be exceeded. This check is satisfactory, and, therefore,

<table>
<thead>
<tr>
<th>Modified Stack #1</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

$H_{\text{WASTE}} = 6 - (16 - 12) = 2 \leq \text{TOLSTK.}$

The stack is within tolerance and the final refinement in the completion stage may proceed. Insert a pallet under each full pile; begin at the topmost individual tier, add successive tiers without exceeding the maximum permitted full pile height 42, and insert a pallet under the newly formed pile.
The tiers on each pallet could be further analyzed to improve stability and weight but this is not required for this problem. The stack is complete and the next stack may be begun. The system is now in state 7, since NFULL = 0, NPART = 0, and NUNUSED = 5. The stacking strategy for this state is to immediately stack NMIN tiers, and to stack additional successive tiers from the list of unused tiers, one at a time, to fill the stack to its maximum permitted height without overfilling. The maximum permitted height (obtained by subtracting the height of NMIN pallets from the container height, HTCONT) is 90 - 2(6) = 78.
Modified Stack #2

<table>
<thead>
<tr>
<th>Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>HSTACK</td>
</tr>
</tbody>
</table>

List of Unused Tiers

| Empty |

HSTACK = ZHTTIER(I) + NMIN × HTPALL

\[
\text{HSTACK} = \sum_{i=1}^{\text{all}} \left( \text{HTTIER}(i) + \text{NMIN} \times \text{HTPALL} \right)
\]

\[
= 12 + 8 + 10 + 10 + 14 + 2 (6)
\]

HSTACK = 64

HWASTE = HCNT - HSTACK = 90 - 64 > TOLSK

No further improvement is possible since the load population is exhausted. Pallet insertion gives:

Final Stack #2

<table>
<thead>
<tr>
<th>Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

HSTACK = HCNT - HWASTE

\[
= 90 - 26 = 64
\]
Container #2

\[
\begin{align*}
HTCONT &= 84'' \\
HTFMAX &= 36'' \\
HTPALL &= 6'' \\
TOLPIL &= 6'' \text{ (by assumption)} \\
TOLSTK &= 3'' \text{ (by assumption)},
\end{align*}
\]

\[
\text{NMIN} = \left| \frac{HTCONT}{HTFMAX} \right| = \frac{84}{36} = 3
\]

For full piles,

\[
HTFMAX - HTPALL - TOLPIL < HTPILE \leq HTFMAX - HTPALL
\]
\[
36 - 6 - 6 \leq HTPILE \leq 36 - 6
\]

The full pile range is (24, 30].

For partial piles,

\[
HTMIN = HTCONT - (NMIN - 1) \times (HTFMAX - TOLPIL)
\]
\[
= 84 - (3 - 1) \times (36 - 6)
\]
\[
= 24 < HTFMAX - TOLPIL = 36 - 6 = 30
\]

\[
HTMIN \leq HTPALL \leq HTPILE \leq HTFMAX - HTPALL - TOLPIL.
\]
\[
24 - 6 \leq HTPILE \leq 36 - 6 - 6
\]

The partial pile range is [18, 24].

Partitioning the initial sequence of tier heights into full piles, partial piles, and unused tiers gives:
List of Full Piles | List of Partial Piles | List of Unused Tiers
---|---|---
16 | 18 | 16
14 | 12 |
14 | 10 |
12 | 10 |
10 | 8 |

\[ \text{NFULL} = 2 \quad \text{NPART} = 3 \quad \text{NUNUSE} = 1 \]

NMIN = 3; Table 1 shows that the system is in state 3.

**Initial Stack #1**

<table>
<thead>
<tr>
<th>18</th>
<th>Partial Pile #1</th>
<th>HTPART(1) = 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Full Pile #1</td>
<td>HTFULL(1) = 14 + 12 = 26</td>
</tr>
<tr>
<td>12</td>
<td>Full Pile #1</td>
<td>HTFULL(2) = 16 + 14 = 30</td>
</tr>
<tr>
<td>16</td>
<td>Full Pile #2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ NMIN - 1 \]
\[
HSTACK = HTPART(1) + \sum_{i=1}^{I} HTFULL(i) + NMIN \times HTPALL
\]
\[
= 18 + 26 + 30 + 3 \times 6
\]
\[ HSTACK = 92 \]
\[ HOVER = HSTACK - HTCONT = 92 - 84 = 8 \]

At this point the stack is overfilled and must be unloaded. The first pile that can be unloaded to produce a stack within tolerance is full pile #1, and the tier having height 12 inches is removed and added to the bottom of the list of unused tiers.

<table>
<thead>
<tr>
<th>Modified Stack #1</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Partial Pile #1</td>
<td>16</td>
</tr>
<tr>
<td>14 Full Pile #1</td>
<td>12</td>
</tr>
<tr>
<td>15 Full Pile #2</td>
<td></td>
</tr>
<tr>
<td>14 Full Pile #2</td>
<td></td>
</tr>
</tbody>
</table>

\[ HWASTE = 8 - 12 = 4 > TOLSTK = 3 \]

The next entry on the list of unused tiers = 16. Trading it for the 14-inch tier in full pile #1 gives

<table>
<thead>
<tr>
<th>Modified Stack #1</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Partial Pile #1</td>
<td>12</td>
</tr>
<tr>
<td>16 Full Pile #1</td>
<td>14</td>
</tr>
<tr>
<td>16 Full Pile #2</td>
<td></td>
</tr>
<tr>
<td>14 Full Pile #2</td>
<td></td>
</tr>
</tbody>
</table>

\[ HWASTE = 4 - (16 - 14) = 2 \leq TOLSTK = 3 \]
The stack is within tolerance and, with a pallet under each pile, the final stack consists of:

**Final Stack #1**

- 18 pallet
- 16 pallet
- 14 pallet

HSTACK = HCONT - HWASTE

\[ = 84 - 2 = 82 \]

The remaining lists are:

<table>
<thead>
<tr>
<th>List of Full Piles</th>
<th>List of Partial Piles</th>
<th>List of Unused Tiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

NFULL = 0
NPART = 2
NUNUSE = 2

Table 1 now shows the system in state 6.

**Initial Stack #2**

- 10 Partial Pile #1
- 8 Partial Pile #1
- 12 Partial Pile #2
- 10

HTPART(1) = 10 + 8 = 18
HTPART(2) = 12 + 10 = 22
\[ HSTACK = \sum_{i=1}^{NMIN} HTPART(i) + NMIN \times HTPALL \]

\[ = 18 + 22 + 3(6) = 58 \]

\[ HWASTE = HTCONT - HSTACK = 84 - 58 = 26 \]

Filling the stack with unused tiers gives

- Individual Tiers
- Partial Pile #1
- Partial Pile #2

\[ HWASTE = 26 - 26 = 0 \]

Inserting pallets gives

- 14 pallets
- 12 pallets
- 10 pallets
- 8 pallets
- 12 pallets
- 10 pallets

Since there are three pallets in the stack and \( NMIN = 3 \), no pallet consolidation can be obtained.
## SUMMARY OF STACKING RESULTS

<table>
<thead>
<tr>
<th>Container #1</th>
<th>Stack #1</th>
<th>Stack #2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requires 2 pallets.</td>
<td>Requires 2 pallets.</td>
</tr>
<tr>
<td>Stack efficiency</td>
<td>( \frac{\text{HSTACK}}{\text{HTCONT}} \times 100 )</td>
<td>( \frac{64}{90} \times 100 )</td>
</tr>
<tr>
<td></td>
<td>( \frac{88}{90} \times 100 = 97.8% )</td>
<td>( = 71.1% )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Container #2</th>
<th>Stack #1</th>
<th>Stack #2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Requires 3 pallets.</td>
<td>Requires 3 pallets.</td>
</tr>
<tr>
<td>Stack efficiency</td>
<td>( \frac{82}{84} \times 100 )</td>
<td>( \frac{84}{84} \times 100 )</td>
</tr>
<tr>
<td></td>
<td>( = 97.6% )</td>
<td>( = 100% )</td>
</tr>
</tbody>
</table>
Container Loading

The available spaces in both containers have identical width and length dimensions.

Width = 90"
Length = 54"

Since the maximum pallet dimensions are 43" X 52", the following pattern is quickly obtained by first generating a type 1 uniform row which satisfies the type 1 configuration tolerance requirements.

```plaintext
43  43  4
52  Stack #1  Stack #2
2
```

DISCUSSION OF RESULTS

The cargo can be loaded in its entirety in either container #1 or container #2. Container #2 is, therefore, the proper choice, since the cost of shipment is lower. The more restrictive pallet loading height, 36", in pallet #2, requires the use of 6 pallets for loading the complete load instead of 4 pallets which would otherwise suffice. In certain instances, the extra space occupied by the additional pallets could result in a load which would not fit in the available space. For this problem, this extra space is available in container #2. The following comparison of space utilization efficiencies shows container #2 to be the more efficient.

102
Container #1

Space Utilization = \frac{\text{Volume of Space Utilized}}{\text{Volume of Available Space}} \times 100

\text{Efficiency} = \frac{\sum \text{box volume} + \sum \text{pallet volume}}{54 \times 90 \times 90} = \frac{260580 + 46080}{453600} \times 100

= 70.1\%

Container #2

Space Utilization = \frac{260580 + 69120}{54 \times 90 \times 84} \times 100

\text{Efficiency} = 80.8\%
RECOMMENDATIONS AND CONCLUSIONS

The author recommends that a computer program be written to implement the proposed automated procedure for maximizing the space utilization of containers loaded with palletized loads. A candidate loading facility should be selected promptly to provide program inputs and to test and evaluate the proposed software.

Considerable testing will be required to determine how to trade off computer costs against improved space utilization efficiency. The sample problem was developed and solved to demonstrate how the various steps are interrelated and not to indicate any expected space utilization efficiencies. However, it is anticipated that, by adhering to a loading procedure which is systematic, logical, and without unwarranted iterations, and by using computers to do simple but laborious types of bookkeeping operations, a favorable balance between computer costs and space utilization efficiencies can be attained. The computer program could be a valuable analytic tool in making decisions, as the sample problem demonstrated. Additional benefits can also be expected from the comprehensive computer-generated reports to be produced at significant stages in the loading operation.

After initial program development and testing, the following enhancements merit consideration:

- Provide for varied loading orientation of boxes. The length or width of a box could be substituted for the height if the box were placed on its side.
• Extend program capability so that a promising pattern in a group can be completed, if necessary, by combining boxes in other groups to synthesize the height of the first group. In Figure 11, the pattern is completed by stacking boxes A and B or boxes C and D to obtain a "composite" box of the required group height.

• Provide for the construction of rows and columns in which the row height and column width exceed, within tolerance, the dimensions of the seed box. In Figure 12a, the result of the row construction technique in the initial program version is contrasted with that in Figure 12b where the row height can be enlarged within tolerance.

• Explore further the possibility of obtaining the program developed by Christofides and Whitlock for dealing with loading situations in which only a limited number of boxes in a large variety of sizes is available. This program might also be beneficial for dealing with boxes for which successful patterns have not been found. Ideally, the proposed program could be interfaced with a program of the type developed by Christofides and Whitlock. The proposed program would be used when a large number of boxes is to be loaded, and the Christofides and Whitlock program would be used when a relatively small number of boxes of diverse and troublesome sizes remains.

   Appendix C presents the flowcharts, Figures 13 through 32, for the proposed program loader.
Figure 11 Example of Pattern Completion by Synthesizing Group Heights

Figure 12a - Initial Program Development

Figure 12b - Advanced Program Development

Figure 12 - Generalization of Row, Column Construction Techniques
ACKNOWLEDGMENTS

The author wishes to acknowledge Mr. Maurice Zubkoff for providing and explaining the background information related to some difficult, hard-to-find details of containerization, and also for his support and overall supervision of the project.
REFERENCES


**APPENDIX A**

**INDEX CHART FOR PALLET PATTERN DETERMINATION**

**INCHES IN LENGTH**

<table>
<thead>
<tr>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>122</td>
<td>118</td>
<td>116</td>
<td>113</td>
<td>112</td>
<td>111</td>
<td>110</td>
<td>109</td>
<td>108</td>
<td>106</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>98</td>
<td>96</td>
</tr>
<tr>
<td>9</td>
<td>122</td>
<td>118</td>
<td>116</td>
<td>113</td>
<td>112</td>
<td>111</td>
<td>110</td>
<td>109</td>
<td>108</td>
<td>106</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>98</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>117</td>
<td>114</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>112</td>
<td>109</td>
<td>106</td>
<td>103</td>
<td>100</td>
<td>97</td>
<td>94</td>
<td>91</td>
<td>88</td>
<td>85</td>
<td>82</td>
<td>79</td>
<td>76</td>
<td>73</td>
<td>70</td>
</tr>
<tr>
<td>6</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
</tbody>
</table>

**INCHES IN WIDTH**

<table>
<thead>
<tr>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>122</td>
<td>118</td>
<td>116</td>
<td>113</td>
<td>112</td>
<td>111</td>
<td>110</td>
<td>109</td>
<td>108</td>
<td>106</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>98</td>
<td>96</td>
</tr>
<tr>
<td>9</td>
<td>122</td>
<td>118</td>
<td>116</td>
<td>113</td>
<td>112</td>
<td>111</td>
<td>110</td>
<td>109</td>
<td>108</td>
<td>106</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>98</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>117</td>
<td>114</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>112</td>
<td>109</td>
<td>106</td>
<td>103</td>
<td>100</td>
<td>97</td>
<td>94</td>
<td>91</td>
<td>88</td>
<td>85</td>
<td>82</td>
<td>79</td>
<td>76</td>
<td>73</td>
<td>70</td>
</tr>
<tr>
<td>6</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
<td>108</td>
<td>105</td>
<td>102</td>
<td>99</td>
<td>96</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>84</td>
<td>81</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>69</td>
</tr>
</tbody>
</table>

**USE OF TABLES**

1. Determine length and width of container to nearest 1/2 inch.

2. Locate the length of container at the top, and width at left side of index chart, the container pattern number will be found at the intersection of two columns. (Height may be substituted for width when the containers are of sufficient strength to withstand superimposed loads, and where such placement will cause no adverse effect in the shipment or storage of the materiel.)
### APPENDIX A (Continued)

#### INCHES IN WIDTH

<table>
<thead>
<tr>
<th>Width (inches)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### INCHES IN LENGTH

<table>
<thead>
<tr>
<th>Length (inches)</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B
PALLETS PATTERNS ON 40" x 48" PALLETS
APPENDIX B (Continued)

Diagram of grids and patterns numbered from 36 to 70.
APPENDIX C - FLOWCHARTS OF PROPOSED PROGRAM LOADER

MAIN PROGRAM LOADER

START

INPUT

PILE

STATE

STAKER

SUPPLY

FILL

UNLOAD

TRADE

PALETS

WEIGHT

STABLE

REprt1

STRING

TYPE1

TYPE2

TYPE3

TYPE4

SAVE

PATRNS

STACKS

STRING

ARRANG

(Entry in PATRNS)

TYPE1

SAVE

WEIGHT

STABLE

REprt2

CNTNER

STRING

ARRANG

(Entry in PATRNS)

TYPE1

TYPE2

TYPE3

TYPE4

SAVE

BARGE

WEIGHT

STABLE

REprt3

END

Figure 13 - Main Program Loader, Flowchart

121
SUBROUTINE INPUT

START

Read large type identifying data

Read identifying data for container types and container dimensions if non-standard

Read pallet dimensions and pallet load constraints

Read box data including identifiers, dimensions, weights, permissable orientations, and batch size

Read tolerance data

Generate and print report of input data

END

Figure 14 - Subroutine INPUT, Flowchart
SUBROUTINE ANALYZ

START

Form groups of boxes, each group consisting of boxes having a common height, with tolerance

Generate list of groups with groups arranged in decreasing sequence of groups heights; let number of groups be NGROUP

For each group, IGROUP, generate list of boxes arranged in decreasing sequence of box areas; let number of boxes in group be NBOX (IGROUP)

RETURN

Figure 15 - Subroutine ANALYZ, Flowchart
Figure 16 - Subroutine PATRNS, Flowchart

SUBROUTINE PATRNS

START

Initialization:
Set group identifier, IGROUP=1;
Set Pattern Type ITYPE=1;
Set flag for box patterns, IBOX on;
Set flags for pallet patterns, IPALL and container patterns, ICONT off.
Set flags for complete rows and complete columns on;
Set PALLL=maximum permissible pallet load length;
Set PALLW = maximum permissible pallet load width
Set seed type as fixed, ITYS=1

Call STRING to get complete rows and columns

ENTRY ARRANG
Order rows in decreasing sequence of row heights

10 Set subtype, ISUBTY = 1
and seed, ISELD = 0

20 ISEED = ISEED = 1
Figure 16 (Continued)

1. **ISUBTY = ?**

2. Call **TYPE3A**

   - **Configuration successful?**
     - **YES**
       - Flag success for **TYPE3**
     - **NO**
       - **315** Call **TYPE3B**

3. **ISUBTY = 1**

   - Set seed type as variable, **ITYS = 2**
   - Set **WASTEL** and **WASTEW** for next waste area
   - **NWASTE = NWASTE - 1**

4. Flag failure for type 3

   - **Pattern within tolerance PTOI?**
     - **YES**
       - **NWASTE = NWASTE - 1**
     - **NO**
       - **316** Call **SAVE**

5. **318** Set **WASTEL = PALLL**, **WASTEW = PALLW**

6. Update seed sequence

7. **400**

   - Retain fixed seed **ITYS = 1**
   - Previous **TYPE3 success?**
     - **YES**
       - Set seed type as variable **ITYS = 2**
     - **NO**
       - Previous **TYPE3 success?**
         - **YES**
           - **ITYPE = 4**
         - **NO**
           - Retain fixed seed **ITYS = 1**

8. **310**

   - Set seed type as variable **ITYS = 2**
   - Previous **TYPE3 success?**
     - **YES**
       - **ITYPE = 1**
     - **NO**
       - Retain fixed seed **ITYS = 1**
Figure 16 (Continued)

- Put container back on list of unused containers
- Set flag for manual assistance in loading of rectangular space
- Flag failure for type 4

- Configuration successful?
  - YES: Flag success for type 4
  - NO: Continue

- I CONT on?
  - YES: Return
  - NO: Put seed on list of unused boxes

- IGROUP > IGROUP?
  - NO: IGROUP = IGROUP + 1
    - YES: Pattern within tolerance, PTOL?
      - YES: Set WASTEL and WASTEW for next waste area
      - NO: ITYPE = 3

- ITYPE = 3

- WASTE = 1?
  - YES: Set seed type as variable Irys = 2
  - NO: ITYPE = 3

- ITYPE = 3

126
Set flag for manual assistance in loading of unused rows, unused columns, and unused pallets

RETURN

125 IPALL on?

IYPE = 2

10

110 Call TYPE1A

Pattern successful?

YES Call SAVE

NO Update seed sequence

115 Call TYPE1B

I SEED = I SEED + 1

I SEED = 0

125 IPALL on?

I SUBTY = 1?

YES

100

150 Any columns?

YES Order columns in decreasing sequence of column widths

NO

125

170 I SUBTY = 2 I SEED = 0

20

Figure 16 (Continued)
Figure 16 (Continued)

1. ISUBTY = ?

210 Call TYPE2A

Row supply full?

YES

Pattern successful?

YES

Call SAVE

NO

Iseed = Iseed +1

170

Save unused rows

NO

ISEED = 0

Any untested seeds left?

YES

ISUBTY = 1?

NO

200

Save unused columns

215 Call TYPE2B

Column supply full?

YES

Update seed sequence

ITYPE = 3

Set number of waste areas, NWASTE=1

Set waste space length, 'ASTEL=PALLL

Set waste space width, WASTEW = PALLW

128
Figure 16 (Continued)

500

Flag manual intervention for difficult boxes

Assign patterns to tiers

Assign shims to tiers

Return
Figure 17 - Subroutine STRING, Flowchart

SUBROUTINE STRING

START

10 any box seeds left

Select next available seed

20 Put on list of row clusters

Attempt type 1 uniform complete row

YES Success?

NO Attempt type 2 uniform complete row

YES Success?

Attempt type 1 non-uniform complete row
Figure 17 (Continued)

A

20

YES

Success?

NO

Attempt type 2 non-uniform complete row

20

YES

Success?

NO

10

Add box seed to list of column seeds

30

Any column seeds left?

NO

Return

YES

Select next available seed

B

131
Figure 17 (Continued)

- Attempt type 1 uniform complete column
  - YES Success?
  - NO Attempt type 2 uniform complete column
    - YES Success?
    - NO Attempt type 1 non-uniform complete column
      - YES Success?
      - NO Attempt type 2 non-uniform complete column
        - YES Success? NO Put seed on list of unused boxes

Put on list of column clusters

132
SUBROUTINE TYPE 1

ENTRY TYPE1A

Select next entry on list of rows as seed

Seed found? NO Flag failure Return

YES

Try linear combination of seed row height to complete pattern

Pattern with tolerance? YES Flag success Return

NO

Try linear combination of heights of seed row and a second row to complete pattern

Pattern within tolerance? YES Flag success Return

NO Flag failure

ENTRY 1B

Same flowchart as TYPE 1A except that column replaces row and column width replaces row height.

Figure 18 - Subroutine TYPE1, Flowchart
SUBROUTINE TYPE 2

Entry TYPE2A

NTIME = 0

Use as many successive complete rows as possible without overfilling pattern

Pattern within tolerance?

YES Flag success Return

NO

Row supply complete?

NO Flag short supply Return

YES NTIME = NTIME + 1

NTIME > 4?

YES Flag failure Return

NO Try to trade next complete row for an existing row in pattern

NO Successful trade?

YES Pattern within tolerance?

Flag success Return

NO

ENTRY TYPE2B

Same flowchart as TYPE 2A except that columns are to be substituted for rows

Figure 19 - Subroutine TYPE2, Flowchart
Figure 20 - Subroutine TYPE3, Flowchart

SUBROUTINE TYPE3

Entry

TYPE3A

Complete

row flag

on?

YES

NO

Any

complete

rows

YES

NO

Set complete row flag off

Flag row success

Exhaust remaining complete rows

Set complete row flag off

Configuration within tolerance ATOL?

YES

Flag configuration success

Return

NO

ATOL = PTOL

NWASTE

Reduce waste length WASTEL by total height of complete rows in configuration

NO

5 Select next entry on list of boxes which fits in unused column space

Above entry found?

NO

Flag failure

Return

YES

Try to obtain partial column; repeat as often as possible

A

135
Figure 20 (Continued)

A

Yes

At least 1 partial column found?

NO

Seed fixed?

Yes

10

NO

S

Flag partial column success; reduce waste width by total width of partial columns

ATOL = PTOL

NWASTE

Configuration within tolerance, ATOL?

Flag configuration success

Return

10 Select next entry on list of boxes which fits in unused row space

Above entry found?

No

flag failure

Return

Yes

Try to obtain partial row; repeat as often as possible

At least 1 partial row found?

No

Return

Seed fixed?

Yes

Flag partial row failure

NO

136
Flag partial row success; reduce waste length by total height of partial columns

\[ \text{ATOL} = \frac{\text{PTOL}}{\text{NWASTE}} \]

Configuration within tolerance ATOL?

Flag configuration success

Return

ENTRY TYPE3B

Same Flowchart as TYPE3A Except Rows and Columns are Interchanged.
SUBROUTINE TYPE 4

START

10 Select next suitable box seed

seed found ?

NO Flag failure

Return

YES

Try symmetric linear combination of 2 box sizes such that
Σ all box widths = waste width - tolerance
Σ all box lengths = waste length - tolerance

Try symmetric linear combination of 2 box sizes such that
Σ all box lengths = waste length - tolerance
Σ all box widths < waste width

Configuration successful ?

NO

YES

Set WASTEL and WASTEW equal to waste hole dimensions

Flag success

Return

Configuration successful?

YES

Reset WASTEW; set remainder waste dimensions; set NWASTE = NWASTE + numbers of new holes

Flag success

Return

A
Try symmetric linear combination of 2 box sizes such that
I all box widths = waste width - tolerance
I all box lengths < waste length

Configuration successful?

NO

Try symmetric linear combination of 2 boxes such that
I all box widths < waste width
I all box lengths < waste length

Yes

Flag failure

Return

Use largest multiple of symmetric cluster in both width and length

Flag remainder waste in two dimensions; reset waste dimensions

NWASTE = NWASTE + 1

Flag success

Return

YES

Reset WASTEL; set remainder waste dimensions; set NWASTE = NWASTE + number of new holes

Flag success

Return
SUBROUTINE STACKS

Initialization
ISPEC=0, NTRADE=0, ISTATP=1, IBOTOM=0
HVOID(1) = HVOID(2) = HVOID(3) = 0
Set ISTACK on, IUNLOD off, IFILL off, ITRADE off
Set constant for minimum supply of tiers, MINC=10.

Determine minimum number of pallets, NMIN to reach container height, HTCONT

\[ \text{NMIN} = \left\lceil \frac{\text{HTCONT}}{\text{HTPAX}} \right\rceil \]
where \( \lceil \cdot \rceil \) denotes next largest integer

Call PILES

20 Call STATE

30 Call STAKER

30 Call STAKER

98 Call REPRT1

Return

A
Figure 22 (Continued)

A

IUNLOD on?

YES

50

IFILL on?

YES

60

NO

IFILL on?

NO

ITRADE on?

YES

70

NO

Return

50

Determine overfill height
HOVER where
HOVER=HSTACK-HTCONT

=0

HOVER =?  

HWASTE= - HOVER

HWASTE=0

>0

110

Call UNLOAD

YES

HWASTE =0?

Set IUNLOD off IFILL on

60

110
Figure 22 (Continued)

B

no supply

message = ?

continue with full supply

Accept; flag reduced number of trades

NTRADE = NTRADE + 1

Call TRADE

Message = ?

go to 85

go check waste

NO

110

98

YES

BOTTOM = 0?

NO

85

NTRADE = 3?

NO

HWASTE < TOLSTK?

YES

BOTTOM = 0

110

YES

Accept; flag for manual intervention

BOTTOM = 0

110

ISTATP = ISTATE

20

85
Figure 22 (Continued)

110 Call PALETS
     Call WEIGHT
     Call STABLE
     Call REPT1

NTRADE = 0
HVOID(1) = HVOID(2) = HVOID(3) = 0
Set ISTACK on, IFILL off,
ITRADE off

ISTATP=ISTATE

20

70

ISTATE = ISTATP? NO

Set NTRADE=0, ISTACK on,
IUNLOAD off, IFILL off, ITRADE off
HVOID(1)=HVOID(2)=HVOID(3)=0
BOTTOM=0

YES

MINTIR=1

75 Call Supply

B
SUBROUTINE STATE

START

1. IF NMIN > NPART
   - ISTATE = 6
   - Return

2. IF NFULL > NMIN
   - ISTATE = 2
   - Return

3. IF NFULL > NMIN - 1
   - ISTATE = 3
   - Return

4. IF NFULL > 0
   - ISTATE = 1
   - Return

5. IF NPART > 0
   - ISTATE = 5
   - Return

6. ISTATE = 7
    - Return

Figure 23 - Subroutine STATE, Flowchart
Figure 24 - Subroutine PILES, Flowchart

SUBROUTINE PILES

START

NFULL=0
NPART=0
NUNUSE=0

Determine minimum height of partially loaded pallet,
HTPMIN given by
HTPMIN = HTCONT-(NMIN-1)x(HTFMAX-TOLPJL)

10

Is
list of tiers
empty?

YES

Return

NO

Get next tier
on list having
height HTTIER(1)

HTPILE = HTTIER(1)

Is
list of tiers
empty

YES

A

I=1

HTPILE = HTPILE + HTTIER(I+1)

NO

HTTEST > HTFMAX-HTPALL

HTTEST = HTPILE + HTTIER(I+1)

YES

A
Figure 24 (Continued)

A

HTPILE > HTFMAX-HTPALL - TOLPIL?

YES

NO

HTPILE > HTFMIN - HTRALL

NPART=NPART+1

NFULL=NFULL +1

Place all tiers on list of unused tiers; update NUNUSE

10

10
SUBROUTINE PALLETS

START

NPYLE > NMIN

NPYLE = NPYLE - 1

15

Try to consolidate any 2 partial piles

Consolidation successful?

Flag unsuccessful consolidation

10

Any individual tiers not on pallet?

Place as many tiers as possible on next available partial pallet with space

20

Any partial pallets with available space

A

B

Previous unsuccessful consolidation?

YES

NO

YES

NO

YES

NO
Correction of HSTACK based on actual number of pallets used:

\[ \text{HSTACK} = \text{HSTACK} + (\text{NPILE} - \text{NMIN}) \times \text{HTPALL} \]

Insert pallet under each pile

\[ \text{HOVER} = \text{HSTACK} - \text{HTCONT} \]

\[ \text{NPILE} = \text{NPILE} + 1 \]

Return

Remove tier having height slightly exceeding HOVER and place on stack having most available space

Add additional tiers if available

HOVER = 0

HOVER > 0

HOVER < 0

Start new pile and put on as many tiers as possible without overloading

\[ \text{10} \]
SUBROUTINE STAKER

START

ISTATE = ?

100

Stock first entry, on list of partial piles, on top of first NMIN-1 entrees on list of full piles

NPILE = NMIN

NMIN-1

HSTACK = PART (1) + \sum_{i=1}^{NMIN} HFULL(i) + NMIN \times HTPALL

Return
Figure 26 (Continued)

200

Stack first NMIN entrees on list of full piles

NMIN
HSTACK = \sum_{I=1}^{NMIN} HFULL(I) + NMIN \times HTPALL

NPILE = NMIN

RETURN

300

NTEST = NFULL + NPART - NMIN

Stack all full piles and all partial piles

NTEST ≥ 0

YES

Stack all full piles and sufficient partial piles, J so that NFULL + NPART = NMIN

A

NFULL
HSTACK = \sum_{I=1}^{NFULL} HFULL(I) + \sum_{I=1}^{NPART} HPART(I) + NMIN \times HTPALL

NPART

NPILE = NFULL + NPART

RETURN

310

NUNUSE = 0?

YES

ISPEC = 1

Set flag to turn UNL0D off, IFILL on

NO

Return

151
Figure 26 (Continued)

400

Stack all full piles

\[ HSTACK = \sum_{i=1}^{NFULL} HFULL(i) + NMIN \times HTPALL \]

NPILE = NFULL

IF \( ISPEC = 2 \) THEN

IF NUNUSE = 0 THEN

Return

ENDIF

ENDIF

500

Stack first NMIN partial piles

\[ HSTACK = \sum_{i=1}^{NMIN} HPART(i) + NMIN \times HTPALL \]

NPILE = NMIN

NPILE = NPILE + 1

HSTACK = HSTACK + HPART(J)

Add next partial pile, J to stack

IF HSTACK - HTCONT \( \geq 0 \) THEN

SET flag to put UNLOD on

ENDIF

IF \( ISPEC = 3 \) THEN

IF NUNUSE = 0 THEN

Return

ENDIF

ENDIF

IF Any more partial piles THEN

RETURN

ENDIF

152
NPART
HSTACK = \sum_{i=1}^{NPART} (HPART(i) + NMIN \times HTPALL)

NPILE = NPART

ISPEC = 4
NUNUSE = 0?

700
Return

Stack all unused tiers
NUNUSE \leq NMIN

Stack NMIN Tiers

HSTACK = \sum_{i=1}^{NMIN} (HTTIER(i) + NMIN \times HTPALL)

ISPEC = 5

NPILE = 0

Return
Figure 26 (Continued)

\[
\text{HSTACK} = \sum_{I=1}^{N\text{FULL}} \text{HFULL}(I) + \sum_{I=1}^{J} \text{HPART}(I) + \text{NMIN} \times \text{HTPALL}
\]

\[
\text{NPILE} = N\text{FULL} + J
\]

\[
\text{If } \text{HSTACK} - \text{HTCONT} > 0, \text{Set flag to turn IUNLOAD on and Return.}
\]

\[
\text{If NO, check if there is an unused entry on list of partial piles?}
\]

\[
\text{If NO, go to 310. If YES, HSTACK = HSTACK + HPART(K). Add next partial pile, K to stack.}
\]
SUBROUTINE SUPPLY

START

Any unused partial piles? NO YES

NUNUSE - Bottom > MINTIR YES NO

Any unused full piles? NO YES

Flag message to continue with full supply

Return

Flag message no supply

Return

Unstack next unused full pile

Unstack next unused partial pile

Add unstacked tiers to list of unused tiers

Flag message to check for change of state

Return

Figure 27 - Subroutine SUPPLY, Flowchart
SUBROUTINE UNLOAD

START

N=1

HOV(N)=HOVER

For all I belonging to pile N, find HTIER(N,I) - HOV(N). Select tier J such that

0 ≤ HTIER(N,J) - HOV(N) ≤ HTIER(N,I) - HOV(N)

Select tier K such that

HTIER(N,I) - HOV(N) ≤ HTIER(N,K) - HOV(N) < 0

Temporarily remove and flag tier K from pile N

HOV(N)=HOV(N) - HTIER(N,K)

Select pile M such that

for all N

0 ≤ HTIER(M,J) - HOV(M)
≤ HTIER(N,J) - HOV(N)

Permanent remove all flagged tiers from pile M and also tier J

M=N

YES

HTIER(N,J) - HOV < TOLSTK?

N > NPILE?

N=N+1

NO

Figure 28 - Subroutine UNLOAD, Flowchart
SUBROUTINE FILL

START

I = 1
HTOT = 0

I > NUNUSE ?

YES

HTOT = HTOT + H(I)

NO

I = I + 1

H(I) > HWASTE ?

YES

HTOT = HTOT - H(I)

NO

I = I - 1

IMAX = NUNUSE

Flag message that supply is used up

Add patterns I = 1, IMAX to previous stack

HWASTE = HWASTE - \sum_{I = 1}^{IMAX} H(I)

NUNUSE = NUNUSE - IMAX

IMAX = 0?

YES

Return

Figure 29 - Subroutine FILL, Flowchart
SUBROUTINE TRADE

Return

Flag message to go to 75

NTRADE=NTRADE-1

Skip this tier

START

Set HNEW = height of next unused tier on list of unused tiers

HNEW equal to any previous voids?

YES

Find all tiers I in stack of tiers such that for all I, 0 < HNEW-H(I) ≤ HWASTE

Select the J belonging to I such that for all I HNEW-H(J) ≥ HNEW-H(I)

J void?

YES

NVoid (NTRADE)=0.0

Trade next unused tier on list of unused tiers for tier J. Put tier J on bottom of list.

Return

Flag message to test waste

HVOID (NTRADE) = HNEW

Return

Flag message to go to 85

Figure 30 - Subroutine TRADE, Flowchart
SUBROUTINE CNTNER

START

Initialization:
IGROUP = 1, NGROUP = 1, ITYPE = 1 ITYS = 1
Container type, ICNTNR set by input
Set IBOX off, IPALL on, ICONT off
Set complete row and complete column flags on

10 Get dimensions of next available container, SPACEL and SPACEW

PALLL=SPACEL
PALLW=SPACEW

Call STRING to get complete rows and columns

Put on list of unused pallets

Flag manual intervention

YES any pallets left

NO Call PATRNS at entry ARRANG

Call REPT2

Call STABLE

Call WEIGHT

Return

Figure 31 - Subroutine CNTNER, Flowchart
Figure 32 - Subroutine BARGE, Flowchart

SUBROUTINE BARGE

START

Initialization:
IGROUP=1, NGROUP=1, ITYPE=1, ITYS=1
Barge type, IBARGE, set by input;
Number of rectangular spaces, NRECT, set by input;
Set IBOX off, IPALL off, ICONT on
Set complete row and complete column flags on

10 Get dimensions of next available rectangular space, SPACEL and SPACEW

PALLL=SPACEL
PALLW=SPACEW

Call STRING to get complete rows and columns

A

160
Figure 32 (Continued)

Call PATRNS at entry ARRANC

NRECT = NRECT - 1

10 NO

NRECT = 0?

YES

Call WEIGHT

Call STABLE

Call REPT3

Return
## INITIAL DISTRIBUTION

<table>
<thead>
<tr>
<th>Copies</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>DLSIE</td>
</tr>
<tr>
<td>1</td>
<td>09</td>
<td>MSC, Code 09</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>MSDEC</td>
</tr>
<tr>
<td>10</td>
<td>521.1</td>
<td>Reports Distribution</td>
</tr>
<tr>
<td>1</td>
<td>522.1</td>
<td>Unclassified Lib (C) (m)</td>
</tr>
<tr>
<td>1</td>
<td>522.2</td>
<td>Unclassified Lib (A)</td>
</tr>
</tbody>
</table>

**Mr. Gerald Gerstel**  
Hq. U.S. Army Avionics  
R&D Activity  
Attn: DAVAA-SS  
Ft. Monmouth, NJ 07703

<table>
<thead>
<tr>
<th>Copies</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>NAVFAC, M. Essoglou</td>
</tr>
<tr>
<td>1</td>
<td>93</td>
<td>FMSO, Code 93</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>NAVSUP</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>SUP 03</td>
</tr>
<tr>
<td>4</td>
<td>04A</td>
<td>SUP 04A</td>
</tr>
<tr>
<td>1</td>
<td>043</td>
<td>SUP 043</td>
</tr>
<tr>
<td>1</td>
<td>05</td>
<td>SUP 05</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>NAVPSCOL</td>
</tr>
<tr>
<td>1</td>
<td>400</td>
<td>NSC CHASN, Code 400</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>NSC OAKLAND, Cdr Nelson</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>NSC PEARL HARBOR</td>
</tr>
<tr>
<td>1</td>
<td>43</td>
<td>NSC PUGET, Code 43</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>NSC SAN DIEGO</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>DTIC</td>
</tr>
</tbody>
</table>

## CENTER DISTRIBUTION

<table>
<thead>
<tr>
<th>Copies</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1800</td>
<td>G.H. Gleissner</td>
</tr>
<tr>
<td>2</td>
<td>1809.3</td>
<td>D. Harris</td>
</tr>
<tr>
<td>1</td>
<td>185</td>
<td>T. Corin</td>
</tr>
<tr>
<td>20</td>
<td>187</td>
<td>M. Zuhkoff</td>
</tr>
</tbody>
</table>

**163**