AD-A078 274 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 13/4

A CONTAINER STUFFING ALGORITHM FOR RECTANGULAR SOLIDS WHEN VOID--ETC(U)

SEP 79 N B NELSON

UNCLASSIFIED NL

1 OF 2

AD A078274

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A CONTAINER STUFFING ALGORITHM FOR RECTANGULAR SOLIDS WHEN VOIDS MAY BE REQUIRED.

by

Napoleon Bonaparte Nelson, III

September 1979

Thesis Advisor: A. W. McMasters

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Container Stuffing Algorithm for Rectangular Solids When Voids May Be Required	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1979
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Napoleon Bonaparte Nelson III	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE September 1979
	13. NUMBER OF PAGES 112
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Containerization, Loading, Algorithm, Palletizing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An algorithm was designed to load different sized rectangular solids into a container. It allows the option of forming pallets of material before loading the container. The algorithm will permit loading of cargo that may or may not be used as load bearing support for other cargo. Cargo is allowed to be rotated if desired to improve efficiency and both the pallets and the shipping container may contain

"voids" or volumes in which cargo is not permitted. A test of the algorithm utilizing an actual cargo list showed two-dimension (area) efficiencies of 95% and three-dimension (volume) efficiencies of 89%.

A Container Stuffing Algorithm For Rectangular Solids When Voids May Be Required

by

Napoleon Bonaparte Nelson III
Lieutenant Commander, Supply Corps, United States Navy
B.S., Georgia Institute of Technology, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1979

Author: Napoleon B. Nelson III

Approved by: Alan W. McMasters
Thesis Advisor

Ellen F. Roland
Second Reader

Michael G. Sovereign
Chairman, Department of Operations Research

D. A. Schrady
Dean of Information and Policy Sciences

ABSTRACT

An algorithm was designed to load different sized rectangular solids into a container. It allows the option of forming pallets of material before loading the container. The algorithm will permit loading of cargo that may or may not be used as load bearing support for other cargo. Cargo is allowed to be rotated if desired to improve efficiency and both the pallets and the shipping container may contain "voids" or volumes in which cargo is not permitted. A test of the algorithm utilizing an actual cargo list showed two-dimension (area) efficiencies of 95% and three-dimension (volume) efficiencies of 89%.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Computerized analyses and computer assisted algorithms have been utilized extensively in most areas of transportation systems. The military, in particular, has relied heavily on loading simulations and computer assisted algorithms to predict the assets required to meet a given transportation demand [2]. However, no reference can be found which indicates that these computer techniques have been accurate, flexible, or descriptive enough to act as an actual blueprint for loading multicommodity cargo into the transportation container, be it a sea van, truck or airplane. The actual loading is apparently still performed, for the most part, by personnel without the help of computers.

In an attempt to partially fill this void, an heuristic algorithm was developed which should be efficient and precise enough to use as an actual blueprint for loading one, two, or three dimension cargo. This algorithm exceeded the unassisted performance of loading crews for sample data; is adaptable to any shaped container; permits container "voids" or volumes where cargo can not be loaded (i.e., refrigeration vests, reserved space, etc.); recognizes that some cargo may be rotated for increased efficiency while other cargo can not be rotated; permits only weight bearing cargo to be used as a base upon which to stack other cargo; and, allows the optional requirement that smaller boxes must be loaded on a standard pallet prior to being placed into the container (the formal term of placing cargo into the container is 'stuffing' as opposed to 'palletizing' the cargo prior to stuffing). In addition, the algorithm is capable of solving a large problem within several computer (CPU) minutes and requires relatively little main core memory.

The algorithm described above is hereafter designated as the container stuffing algorithm. The need for such an algorithm is discussed in the next section.

B. THE NEED FOR AN ALGORITHM

The advent of mechanized warehouses, sharply increasing transportation costs, and increased availability of computers greatly increases the potential return on investment that is expected to be realized from the implementation of an algorithm as described above.

Mechanized warehouses permit extremely rapid, efficient, and flexible issuance of material from the warehouse. Material in a mechanized warehouse is received, stored, and issued with very little manual intervention by the warehouseman. This is accomplished by the use of real time data bases, one hundred percent visibility within the receipt-issuance cycle, and complete knowledge of the item characteristics (weights, dimensions, etc.) of the material being stored. However, some of the efficiency gained by the mechanized warehouse is lost once the material is dispatched from the mechanized warehouse. For example, the material must be staged in the shipping section prior to the actual loading of the material into a shipment container. This is necessary because of the current inability to accurately predict necessary transportation assets and because of the need by the loading personnel to physically view and study the physical characteristics of the cargo prior to commencement of the loading process. By eliminating the need for staging material prior to shipment, savings could be realized in manpower necessary to actually load the material, in staging cost, and in costs associated with positioning the container prior to commencement of the loading. Additionally, there is a cost associated with delaying release of the container pending completion of documentation which must be prepared after the cargo is loaded but prior to releasing the container.

Considerable savings could also be realized if greater efficiency of cargo volume to container volume were possible. As one specific example, if Naval Supply Center, Oakland, California, could increase its efficiency for shipments to Japan and Philippines (705,400 cubic feet or 17,635 measurement tons per year) from the current rate of 80% to, say, 87%, a yearly savings of transportation costs would be approximately $266,000 [12].

The algorithm developed to satisfy these requirements will be discussed in detail after the next section which briefly discusses the generic operations research problem which is becoming known as the loading problem [3].

C. THE LOADING PROBLEM

In order to understand the stuffing problem it is first necessary to review its superset, the loading problem. The generalized loading problem is one in which items, $I_i \varepsilon I$ of magnitude q_i and value v_i, are placed in containers, $C_j \varepsilon C$ capacity c_j and cost of d_j. The sets I and C contain, respectively, all items to be loaded and all containers used in the loading.

The problem may indicate:

a. $\sum_{j \varepsilon C} c_j \geq \sum_{i \varepsilon I} q_i$ and all items are loaded, or

b. $\sum_{j \varepsilon C} c_j \geq \sum_{i \varepsilon I} q_i$ and all items need not be loaded; or

$\sum_{j \varepsilon C} c_j < \sum_{i \varepsilon I} q_i$.

The objective may be to

a. minimize $\sum_{i \varepsilon S} (q_i \cdot v_i)$ where S is the set of all items not loaded $(S \varepsilon I)$;

b. minimize $\sum_{j \varepsilon C} (c_j \cdot d_j)$

A matrix of the objective functions and problem statement is shown below and summarizes the various possible problems. An asterisk indicates that the assumption of additivity has been made. Under this assumption, whenever $\sum_{i \varepsilon I} q_i \leq \sum_{j \varepsilon C} c_j$, all i items may be loaded in the j containers; that is, the assumption of additivity allows the quantities which are being loaded to be added together without geometrical considerations of individual containers.

This assumption is easily made when measurements are in terms of money, weights, liquid volume, or when $\max_{i \varepsilon I}(q_i) << \min_{j \varepsilon C}(c_j)$ and prior palletization is not used.

Problem Statement

		a	b
Objective	a	1	2^*, 3
	b	3^*, 4	5

Figure 1. Loading Problem Subsets

Problem 2^* is the classical multidimensional knapsack loading problem which has been extensively analyzed [8]. Problem 3^* has been solved for the case where $c_k = c_j$ for all k and j by Eilon and Christofides [3]. The Problem 3 solution for the case where items I are rectangular solids was developed by Gilmore and Gomory who used very large scale integer programming techniques [7]. The Problem 3 solution where $q_i = q_k$ for all i, k and items I were rectangular solids with one set of common dimensions was given by Seam and Sivazlian [10]. Problem 4 solution where items I were rectangular solids was presented by DeSha [2]. The Problem 4 solution for the case where items I were parallelepipeds and C was a single container was given by Galata and Stoyan [4]. This paper is concerned with a specific subset of Problem 4 called herein the container stuffing problem which is a representation of the generalized method typically used to ship cargo. Although the problem was formulated in terms of a shipping problem, it may be easily expanded to solve related problems such as those presented by Brown [1].

D. THE CONTAINER STUFFING PROBLEM

This problem is one in which n boxes of size BOX_j are to be loaded onto pallets of capacity p_k, which are, in turn, loaded (stuffed) into containers, C_i, of capacity c_i. In the problem $BOX_j \leq p_k \leq c_i$ for all i, k and $\sum_j BOX_j \leq \sum_k p_k \leq \sum_i c_i$ with an objective of minimizing the number of containers required to load a given series of BOX_j, $j=(1,n)$. Because of the relative closeness (in size) of BOX_j to p_k and p_k to c_i, geometric considerations are extremely important in obtaining a feasible solution to the minimization problem; and, thus an elementary but important constraint must be addressed: none of the boxes (pallets, containers) may overlap into the space occupied by another box (pallet, container).

A successful practicable solution to this problem must also consider these following points. A rectangular solid box may be loaded into a container six different ways depending on the relative positioning of the moving coordinate system (x', y', z') associated with the box and the fixed coordinate system (x, y, z) associated with the container. It is assumed the container has its "origin" located at position (0, 0, 0) with length, width, and height in the x, y, z direction. Figure 2 shows the six possible orientations of a box in a container. These degrees of freedom

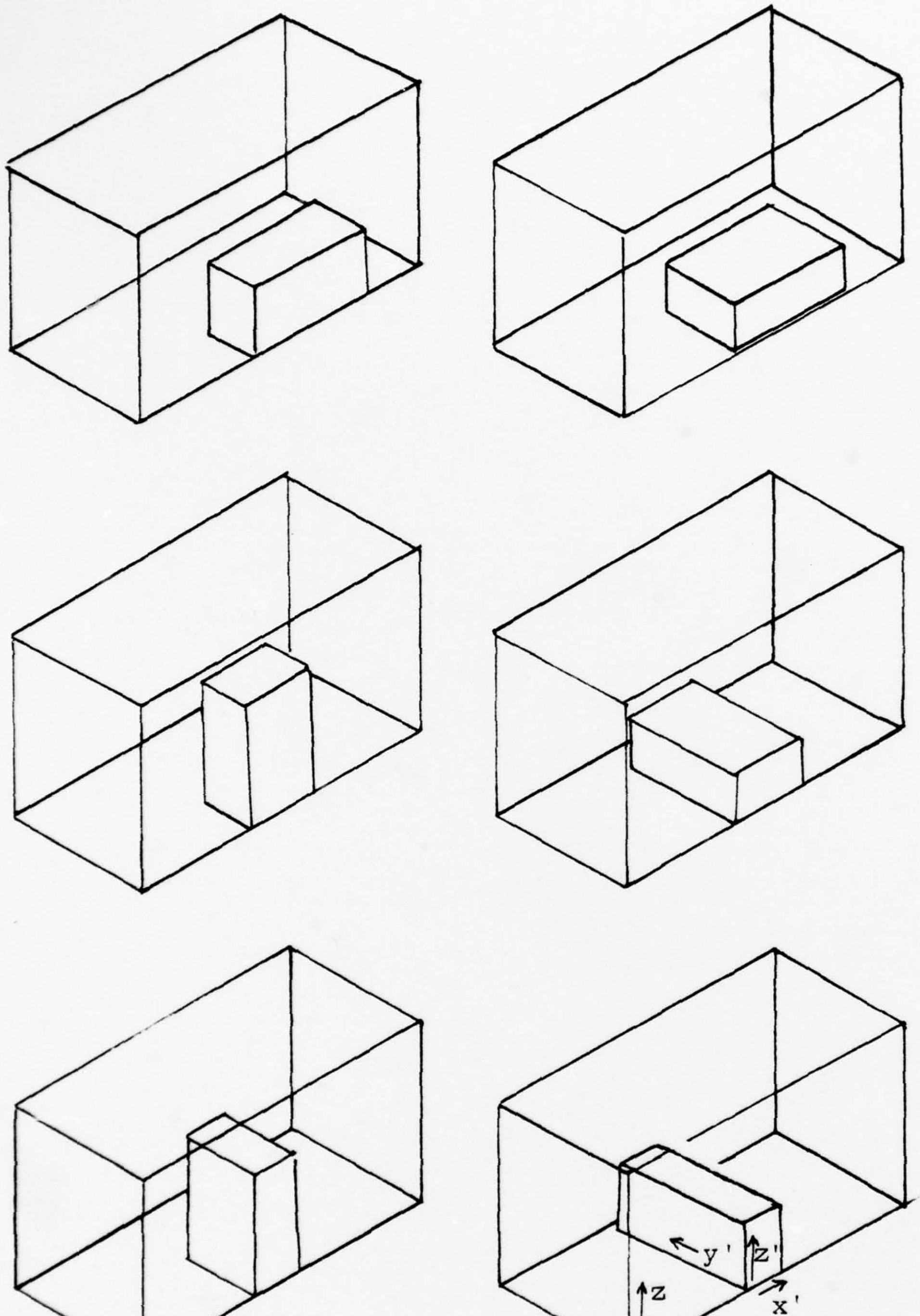


Figure 2. Possible Orientation of a Box within a Pallet/Container.

permit $n!6^n$ possible sequences for loading n boxes. For instance, six boxes may be loaded into a container in more than 33 million different sequences.

Because of this vast number of sequences, a practicable solution must be descriptive as well as prescriptive. It must define not only the sequence of the loading but also the relative position of each box in the pallet and the relative position of each pallet in the container.

Fortunately, certain real world constraints reduce the number of feasible solutions. The nature of some of the boxes requires that the box be loaded "this side up", that is, the box has a predetermined orientation (this reduces the problem to only two possible orientations). Also, the boxes have different load bearing capabilities resulting in the larger, heavier boxes being placed near the bottom of the stack.

The loaded containers must meet specified maximum weights and distributed weight parameters and hence may not end up being completely filled. Also, some containers require that voids be reserved to permit air circulation around vents, or to provide space for future cargo to be loaded elsewhere, etc. Finally, it is often the practice of loading personnel to utilize one of the larger boxes as the "pallet base" for equal size (length to length and width to width) and smaller boxes.

E. RELIANCE ON HEURISTICS

No reference could be found that presents an exact solution for the container stuffing problem. The complexity of the problem and need for rapid solutions exceed the capabilities of even the most sophisticated mathematical solutions currently available. Therefore, the solution techniques presented in this paper rely heavily on heuristics to approximate the exact solution to the stuffing problem.

II. OBJECTIVES AND SCOPE

The objective of this study was to develop a flexible algorithm capable of solving the above defined container stuffing problem. The algorithm which was developed presupposed that a computer would be required for the calculations and that all necessary data on the items to be loaded were available. These necessary items include dimensions of the box and its load bearing capabilities.

For tractability, each input box and the container was assumed to be a rectangular solid. This assumption as it related to the input boxes could be relaxed in actual practice by defining a rectangular solid which superscribes the object to be loaded and by designating this rectangular solid as a non-load bearing box. Likewise, the assumption of rectangularity of the shipping container can be relaxed by defining a rectangular solid which is superscribed by the actual container and by defining voids within this rectangular solid. The accuracy of this approximation is simply a function of the scaling of the dimensions used in the algorithm.

Weight distribution of items within the container was not explicitly addressed by the algorithm. Neither was center of gravity restrictions. However, these restrictions could be easily included by modification of peripheral logic in the algorithm.

III. THE STUFFING ALGORITHM

A. GENERAL DESCRIPTION

The stuffing algorithm (the FORTRAN program is contained in Appendix A) was designed to provide as much flexibility as possible, to be descriptive (i.e., describe the loading procedure in terms of relative positioning of each box) as well as prescriptive, and to run as fast as possible on a computer. As will be shown below, the amount of time required to execute the algorithm is a function of the options used as well as the level of optimization desired. By varying the input parameters the following options may be exercised:

(1) Prior to stuffing the container, the input boxes may be first loaded onto a "standard" pallet whose dimensions are specified by input parameters.

(2) Boxes larger than a certain size can be used as a base upon which to stack other boxes.

(3) Before loading a box the algorithm may or may not require that the box be supported at each of the box's lower four corners. If support is not required, boxes may "overhang" or even be suspended with no support as may be desired when, say, designing an electronic component comprised of many subcomponents which may be held in place by wiring.

(4) Boxes may be individually specified as non-load bearing boxes which denies their use as support for other boxes. Non-load bearing boxes may still be overstacked

provided the overstacked box receives support from load bearing boxes.

(5) The container is initially defined to be a rectangular solid with rectangular solids (i.e., voids) cut from the original rectangular solid. Thus, any reasonable geometric shape may be approximated.

(6) Voids may be described either in pallets or in the container or both, provided the voids can be constructed of a series of rectangular solids. The void need not be contiguous to a boundary of the pallet or the container.

(7) Voids placed in pallets and/or containers may either be load bearing or non-load bearing.

(8) Boxes may be rotated from zero to five times in order to improve local optimization of the loading process.

(9) Boxes may be specified whereby no other box is allowed contiguous to one of the specified box's five remaining sides.

(10) The level of optimization, and, therefore, the amount of computer time is specified by input parameters.

(11) The algorithm may be used to load either three, two, or one dimensional objects (rectangular solids, rectangles, or lines).

B. MEASURE OF EFFECTIVENESS

The measure of effectiveness (MOE) used in the optimization sequence was defined as the total volume of input boxes divided by the volume of the containers into which the boxes were stuffed. The volume of the containers was defined as:

$$(N-1)(VOL) + (MAXW \div CONW)(VOL) ,$$

where: N = Number of containers utilized

VOL = Volume of one container

MAXW = Maximum width utilized in the last container

CONW = Container width

This measure of effectiveness was devised in order to penalize for any wasted volume on pallets as well as wasted volume in the container itself and to allow differentiation between various loading sequences for the case in which all boxes were stuffed into one container.

As an example, the measure of effectiveness for the example problem solved in Chapter IV in Figure 3, is 0.733 and is computed as follows.

Total box volume is 228,096, container dimensions and volume are 60x60x96 and 345,600, respectively, and all boxes were stuffed into one container.

The formula shown above then gives:

$$MOE = (228,096)\div((1-1)(345,600)+(54\div60)(345,600))=0.733.$$

C. PREVIEW OF THE ALGORITHM

Before discussing the algorithm in detail, it is first necessary to describe the general approach to the stuffing algorithm and to set forth a few basic definitions.

The algorithm loads one "pallet" (as defined below) at a time by inspecting each of the n boxes in the ordered input stream of boxes. Box number one is inspected first,

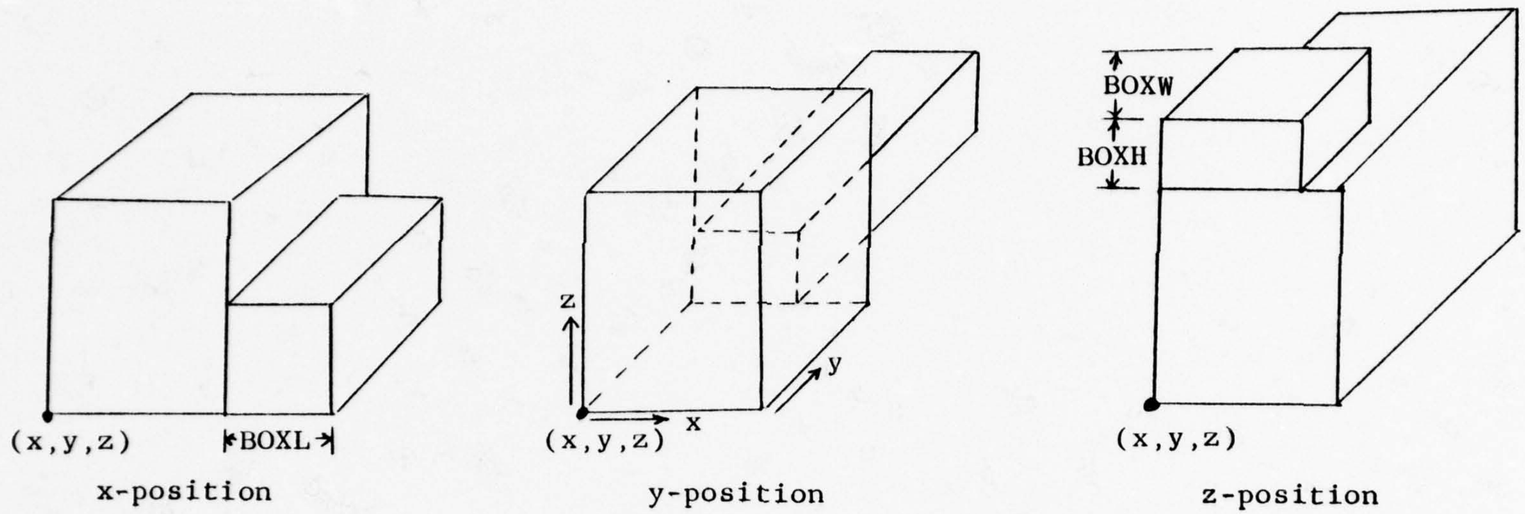


Figure 3. Possible Origins for Placing Additional Boxes on the Pallet.

box number two is inspected second, and so forth until all n boxes have been inspected. If the box under inspection can be loaded without violation of one of the conditions (constraints) discussed in Section IIIA above, the box is loaded. If the current box does violate one of the constraints, it is passed over and the next box in the sequence is inspected. If none of the boxes waiting to be loaded can be loaded, a new pallet is begun. Thus, the algorithm maintains "feasibility" while searching for one particular solution (i.e., a local minimum) to the stuffing problem.

The loading procedure requires a decision to be made as to where an additional box may be placed. These locations (defined as "possible" origins) are limited by the algorithm to be the origin of the pallet or one of three positions relative to each of the boxes and voids previously loaded on the pallet as measured from the origin of the pallet. These positions are defined as the "x-position", "y-position", and "z-position" and are shown in Figure 3. The dimensions of the box being added are denoted by BOXL, BOXW, and BOXH corresponding to its length, width and height, respectively. A pallet which contains j boxes and voids will have (3j+1) possible origins. These positions were selected as possible origins because they limit the possible positions of the next box to a finite, manageable number of locations and a fast verification of feasibility. Finally, of all the "possible" origins, a subset of "permissible" origins is defined. This subset of "permissible" origins is determined by deleting from all possible origins those origins which have already been utilized by loaded boxes; by deleting the z-positions of all boxes defined to be non-load bearing boxes; by deleting positions at which none of the boxes still in the input stream can possibly fit; and by deleting positions at which it is desired to have no boxes contiguous to a loaded box's face (for example, if it be desired to have no box to the right (y-direction) of a given box, the y-position associated with the given box would not be included in the set of permissible origins.

The order of inspection of those permissible origins is (1) the x-position ordered from the first loaded box to the last loaded box; (2) the y-position similarly ordered; (3) lastly the z-position likewise ordered from the first to the last loaded box. This order of inspection tends to fill the pallet in layers, always starting from the pallet's origin and progressing away from the origin in all directions.

In order to determine if a box may be loaded at a given permissible origin, it is necessary to maintain a record of all previously loaded boxes and their relative positions in the pallet. This is accomplished by maintaining a record of the previously loaded boxes' origins (defined as the "current" origins) and the boxes' dimensions in the x, y, and z directions. Thus, the first part of the feasibility question is answered by considering the box at a particular permissible origin and determining if the box is wholly contained within the space of the pallet and if the box does not intersect any previously loaded box or void.

The second part of the feasibility question must be addressed whenever the problem input parameters require that each box must be supported. A box is considered to be supported whenever all four of its lower corners rest upon a load-bearing box or void. This requirement, when exercised, does not allow any "overhang" of the box.

A local minimum is obtained by attempting to move each box, as it is loaded, toward the origin of the pallet. This is accomplished by determining if the box may be moved along one of the three directions, x, y or z, toward the pallet's origin. Movement is permitted only if the box does not intersect any previously loaded box or defined void. The box is moved in one direction at a time and movement is continued in an iterative fashion until no further movement toward the origin is possible.

Finally, the term "pallet" is formally defined as a volume in which boxes are loaded. A "pallet" has dimensions of length, width, and maximum stacking height. A "pallet" does not, itself, occupy space. The dimensions of the pallet are determined by the input parameters. In the algorithm, two types of pallets are used; a "standard" pallet and a "minimum" pallet. The "minimum" pallet actually defines the smallest sized box which will be allowed to serve as a pallet base. The algorithm operates by selecting the next box in the input stream of boxes which is outsized to the minimum pallet. The dimensions of this box are then used to define the pallet base upon which to stack subsequent boxes. If no box is outsized to the minimum pallet, the standard pallet is then used as the pallet upon which to stack boxes. A box is outsized if either the length of the box is larger than the length of the minimum or the width of the box is larger than the width of the minimum pallet. Thus, if prior palletization is not desired, the minimum pallet and the standard pallet are defined to be the same size as the container. Conversely, if it be desired to use the boxes themselves entirely to palletize the remaining boxes, the minimum pallet dimensions are defined to be zero.

In order to clearly present the algorithm, the next section will briefly describe the notation used in the stuffing algorithm. Following this section, the algorithm will be stated. After the statement of the algorithm, a brief, sample problem will be solved for illustration purposes.

D. NOTATION AND DESCRIPTION OF MATRICES

Before setting forth the exact stuffing algorithm, notation will be briefly covered and the matrices used in the algorithm will be defined.

There are n boxes to be loaded and their characteristics are contained in a matrix C. The original identification of an individual box is denoted as N. The subscript which refers to the order in the input sequence of box N is θ. Matrix C is formally defined as the (px5) matrix of input boxes as follows:

$$C = \begin{pmatrix} N_1 & r_1 & BOXL_1 & BOXW_1 & BOXH_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ N_p & r_p & BOXL_p & BOXW_p & BOXH_p \end{pmatrix}$$

where: N_θ = box identification of BOX_θ, $1 \leq \theta \leq p$

r_θ = number of boxes with dimensions of ($BOXL_\theta$ x $BOXW_\theta$ x $BOXH_\theta$).

Since there are n input boxes, $\sum_\theta r_\theta = n$. Matrix C is used in the algorithm as a device to maintain a record of boxes yet to be loaded.

The subscript which refers to the order in which box N is loaded onto the pallet is j. Thus N_{θ_j} represents the completed notation for box ordering. By $N_{\theta_j} = 2_{5_3}$ is meant that a box with identification number two was the fifth box in the ordered input stream and was the third box to be loaded. The voids, V, which may be defined on the pallets and the container are subscripted with an i. Thus $V_i = V_1$ could represent void one on the pallet currently being loaded whose dimensions are length, width, and height of $VL_i = VL_1$, $VW_i = VW_1$ and $VH_i = VH_1$, respectively.

In order to permit a very rapid determination of whether a given box will fit at a given origin, a digital model of the pallet is established and updated with each box which is loaded onto the pallet. This digital model allows the determination of fit to be made through a series of very fast logic checks as described in the next section. This model is defined as Array A which is an ((m+n)x7) matrix as follows:

$$A = \begin{pmatrix} 1 & x_1 & y_1 & z_1 & (x_1+VL_1) & (y_1+VW_1) & (z_1+VH_1) \\ 2 & x_2 & y_2 & z_2 & (x_2+VL_2) & (y_2+VW_2) & (z_2+VH_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m & x_m & y_m & z_m & (x_m+VL_m) & (y_m+VW_m) & (z_m+VH_m) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N_{\theta_{m+j}} & x_{\theta_{m+j}} & y_{\theta_{m+j}} & z_{\theta_{m+j}} & (x_{\theta_{m+j}}+BOXL_{\theta_{m+j}}) & (y_{\theta_{m+j}}+BOXW_{\theta_{m+j}}) & (z_{\theta_{m+j}}+BOXH_{\theta_{m+j}}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

where: x_i, y_i, z_i are the coordinates of the i^{th} void whose length, width and height are VL_i, VW_i, and VH_i, respectively, $0 \leq i \leq m$.

N_{θ_j} is the identification number of BOX_θ whose length, width and height are $BOXL_\theta$, $BOXW_\theta$, $BOXH_\theta$, $1 \leq \theta \leq n$, $1 \leq j \leq n$, and whose origin is located at coordinates $(x_{\theta_j}, y_{\theta_j}, z_{\theta_j})$.

Thus, matrix A column one identifies the box (or void), columns two through four identify the box's (or void's) location nearest the pallet origin, and columns five through seven in conjunction with columns two through four describe the volume occupied by the box (or void).

To facilitate the selection of the next origin at which the algorithm will attempt to load the current box, a logical array of possible and permissible origins is established. By an extremely rapid scan of this model, defined as matrix B, the next origin is quickly determined. Matrix B is an ((m+n)x3) matrix as follows:

$$B = \begin{pmatrix} XORG_1 & YORG_1 & ZORG_1 \\ \vdots & \vdots & \vdots \\ XORG_m & YORG_m & ZORG_m \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ XORG_{\theta_{m+j}} & YORG_{\theta_{m+j}} & ZORG_{\theta_{m+j}} \\ \vdots & \vdots & \vdots \end{pmatrix}$$

where: $XORG_i$, $YORG_i$, $ZORG_i$, $0 \leq i \leq m$, are logical variables which when true indicate that $((x_i + VL_i, y_i); (x_i, (y_i + VW_i), z_i);$ and $(x_i, y_i, (z_i + VH_i)))$, respectively are permissible origins next to voids and $XORG_{\theta_j}$, $YORG_{\theta_j}$, $ZORG_{\theta_j}$, $1 \leq j \leq n$, are logical variables which when true indicate that $((x_{\theta_j} + BOXL_{\theta_j}, y_{\theta_j}, z_{\theta_j}); (x_{\theta_j}, (y_{\theta_j} + BOXW_{\theta_j}), z_{\theta_j});$ and $(x_{\theta_j}, y_{\theta_j}, (z_{\theta_j} + BOXH_{\theta_j})))$, respectively are permissible origins next to loaded boxes. For ease of notation, once a permissible origin has been selected for inspection, it will be designated merely as (ORX, ORY, ORZ).

Each row of matrix B corresponds to a row in matrix A. The three elements in each row of matrix B correspond to the x-direction, y-direction and z-direction possible origins associated with each void and box on the current pallet (and described by matrix A). Of all the possible origins, the permissible origins are then defined by setting to a true value each element in matrix B which corresponds to the possible origin which is also a permissible origin. It is precisely these permissible origins where attempts will be made to load additional boxes.

E. EXACT STATEMENT OF THE STUFFING ALGORITHM

To stuff n boxes (which are described by matrix C) into containers the following steps are used.

1. Select the first ordered box which is outsized to the "minimum" pallet dimensions. Define the pallet base as this outsized box. If no outsized box is found, define the pallet as the standard pallet.

2. Establish matrix A as the digital model of the pallet.

3. Establish matrix B as the logical model of possible and permissible origins.

4. Load the outsized box if one were found. Otherwise, load the first box from matrix C. Augment matrices A and B with an additional row to represent this box. Adjust matrix B as necessary to remove, if necessary, an origin from the set of permissible origins.

5. Select the next box in matrix C. If no more boxes are left, go to step 12.

6. Select the next permissible origin. The next permissible origin corresponds to the next true element in column 1 of matrix B (x-position), followed by the next true element in the second column of matrix B (y-position), and finally followed by the next true element in the third column of matrix B (z-position). Call the selected origin (ORX, ORY, ORZ). If all origins have been tried and the box may

be rotated, go to step 11. If all positions have been tried and the box may not be rotated go to step 5.

7A. Determine if the box will fit at this origin. If it will not fit go to step 6 (The method of this determination will be described below).

7B. Improve the density of packing, if possible (This procedure will also be described below).

8. If boxes must be supported (as discussed above), determine if the box is supported (as discussed below). If the box must be supported but is not supported, go to step 6.

9. Load the box and augment matrices A and B with an additional row. Adjust matrix B to preclude any origin that may not be used.

10. Go to step 5.

11. Turn the box and go to step 6.

12. If all boxes in matrix C are not yet loaded, go to step 5. If all boxes are loaded and the container has been stuffed, terminate the algorithm. If all boxes are loaded onto pallets but the pallets have not been stuffed into containers, move the pallets into array C, define the minimum and standard pallets as the container dimensions, and repeat the algorithm by returning to step 1.

Step 7A is determined as follows. Inspect each previously loaded BOX_{θ_j}, $m+1 \leq j < m+n-1$ and reject the loading of the current box BOX_{θ_k} because it would intersect BOX_{θ_j} if

$A(j,2) < (ORX + BOXL_{\theta_k})$ and $A(j,5) > ORX$ and

$A(j,3) < (ORY + BOXW_{\theta_k})$ and $A(j,6) > ORY$ and

$A(j,4) < (ORZ + BOXH_{\theta_k})$ and $A(j,7) > ORZ$

where: A(j,k) is an element in matrix A and (ORX, ORY, ORZ) is the origin being inspected.

If voids (V) were introduced, a similar inspection would be necessary of each V_i, $0 \leq i \leq m$ whereby VL_i, VW_i, and VH_i would be substituted for $BOXL_{\theta_j}$, $BOXW_{\theta_j}$, and $BOXH_{\theta_j}$, respectively.

Step 7B is determined as follows. Inspect, one at a time, each possible direction of improvement (x, y, z). Each direction of improvement is found by inspecting the origin (ORX, ORY, ORZ) under question and each row of matrix A. To determine if improvement be possible in the x direction, the following logic check is made on each row, k, of matrix A:

$$A(j,2) > (ORX+BOXL_{\theta_k}) \text{ or } A(j,3) > (ORY+BOXW_{\theta_k}) \text{ or}$$
$$A(j,4) > (ORZ+BOXH_{\theta_k}) \text{ or } A(j,6) < ORY \text{ or}$$
$$A(j,7) < ORZ.$$

A true condition indicates that improvement is not possible at this row in matrix A. A false condition indicates that improvement is possible. The magnitude of improvement is: ORX - A(j,5) if improvement is possible or 99,999 if improvement is not possible. Now denote as $slack_k$ the magnitude of improvement found by inspecting row k of matrix A. The improvement found over all rows of matrix A is then:

$$\min(slack_1, \ldots, slack_{m+n}, ORX).$$

To determine improvement in the y direction the logic check is:

$A(j,3) > (ORY+BOXW_{\theta_k})$ or $A(j,2) > (ORX+BOXL_{\theta_k})$ or

$A(j,4) > (ORZ+BOXH_{\theta_k})$ or $A(j,5) < ORX$ or

$A(j,7) < ORZ$.

The magnitude of improvement when the logic check is false is ORY - A(j,6).

To determine improvement in the z direction the logic check is:

$A(j,4) > (ORZ+BOXH_{\theta_k})$ or $A(j,2) > (ORX+BOXL_{\theta_k})$ or

$A(j,3) > (ORY+BOXW_{\theta_k})$ or $A(j,5) < ORX$ or

$A(j,6) < ORY$.

The magnitude of improvement when the logic check is false is ORZ - A(j,7).

With each improvement, the origin (ORX, ORY, ORZ) of the box being loaded is adjusted. The search for improvement is continued until no improvement is found in any of the three dimensions.

Step 8 is determined as follows. Accept BOX_{θ_k} as supported if some BOX_{θ_j}, $m+1 \leq j \leq m+n-1$, for which $z_{\theta_j} + BOXH_{\theta_j} = z_{\theta_k}$, $m+1 < j < k \leq m+n$, and BOX_{θ_j} is not declared a non-load bearing box, satisfies the following condition:

$A(j,2) \leq xx_u$ and $A(j,5) \geq xx_u$ and

$A(j,3) \leq yy_u$ and $A(j,6) \geq yy_u$

for $u = 1,2,3,4$ where xx_u and yy_u are defined as follows:

When: $u = 1$ or 3 $\quad xx_u = x_{\theta_j}$

$u = 2$ or 4 $\quad xx_u = (x_{\theta_j} + BOXL_{\theta_k})$

$u = 1$ or 2 $\quad yy_u = y_{\theta_j}$

$u = 3$ or 4 $\quad yy_u = (y_{\theta_j} + BOXW_{\theta_k})$

Note the four (u) conditions are independently considered and one or more BOX_{θ_j} must be required to satisfy all these conditions. If load bearing voids were present, a similar inspection of V_i, $0 \leq i \leq m$, would be necessary.

F. SAMPLE TWO-DIMENSIONAL PROBLEM

In order to illustrate the algorithm, the following example is presented. It is a two dimensional problem since all features of the algorithm can be covered in two dimensions and two dimensions are more easily demonstrated. The three-dimensional stuffing algorithm is converted to a two-dimensional one by defining the height of each input box to be the height of the container, so that no stacking in the z direction occurs.

In this example problem, 4 boxes (of base sizes 30x30, 34x24, 20x20, and 26x10 and identification numbers 1 through 4 respectively) are to be loaded without standard pallets and without prior palletization into a container of 60(L) x 60(W) x 96(H) with one void (10x10) which is located in the lower right corner of the container. For illustration, this void may not have a contiguous box to its upper side. Each box may be rotated only once (i.e., all the boxes are marked "this side up"). The input stream of boxes has been ordered by area (length times width). Since only two-dimensional loading is considered, the boxes' heights are defined to be the height of the container (i.e., 96). The minimum pallet and the standard pallet widths, lengths, and heights are 60, 60, and 96, respectively. Matrix C has elements as follows:

$$C = \begin{pmatrix} 1 & 1 & 30 & 30 & 96 \\ 2 & 1 & 24 & 34 & 96 \\ 3 & 1 & 20 & 20 & 96 \\ 4 & 1 & 26 & 10 & 96 \end{pmatrix}$$

Step 1. Select the first box outsized to the minimum pallet. There were none, so define the current pallet dimensions to be 60x60x96.

Step 2. Establish the A matrix which now contains all the voids (in this example there is only one). The A matrix, at this point is:

$$A = (1 \quad 50 \quad 0 \quad 0 \quad 60 \quad 10 \quad 96)$$

Step 3. Establish the B matrix of origins. Since the void has been specified so that no box may touch its upper face, its y-direction origin is not a permissible origin. The void also does not have a permissible x-direction origin nor z-direction permissible origin because no boxes yet to be loaded can fit at either of these origins inasmuch as the distance from these origins to a boundary of the pallet is zero. In fact, since all boxes have the same height as the pallet, B matrix will never show a permissible origin in the z-direction. Thus, B matrix is now:

$$B = (F \quad F \quad F)$$

Step 4. Load the next box in matrix C. This is box one which has dimensions of 30x30x96. Augment the A matrix to include this box as follows:

$$A = \begin{pmatrix} 1 & 50 & 0 & 0 & 60 & 10 & 96 \\ 1 & 0 & 0 & 0 & 30 & 30 & 96 \end{pmatrix}$$

Augment B matrix and show permissible origins by setting the applicable element to true. Thus,

$$B = \begin{pmatrix} F & F & F \\ T & T & F \end{pmatrix}$$

Step 5. Select the second box in the C matrix (24x34x96).

Step 6. Select the first permissible origin. That is, scan matrix B column by column always starting at the top of each column and working down. In this case, element B(2,1) is the next permissible origin. This element translates into an origin of (A(2,5), A(2,3), A(2,4)) or (30, 0, 0). Denote this as the current (ORX, ORY, ORZ).

Step 7. Determine if the box will fit at this origin. This is accomplished by the following logic checks of matrix A. A true condition for any row of matrix A indicates that the box will not fit. Thus for, say, row one the check proceeds as follows:

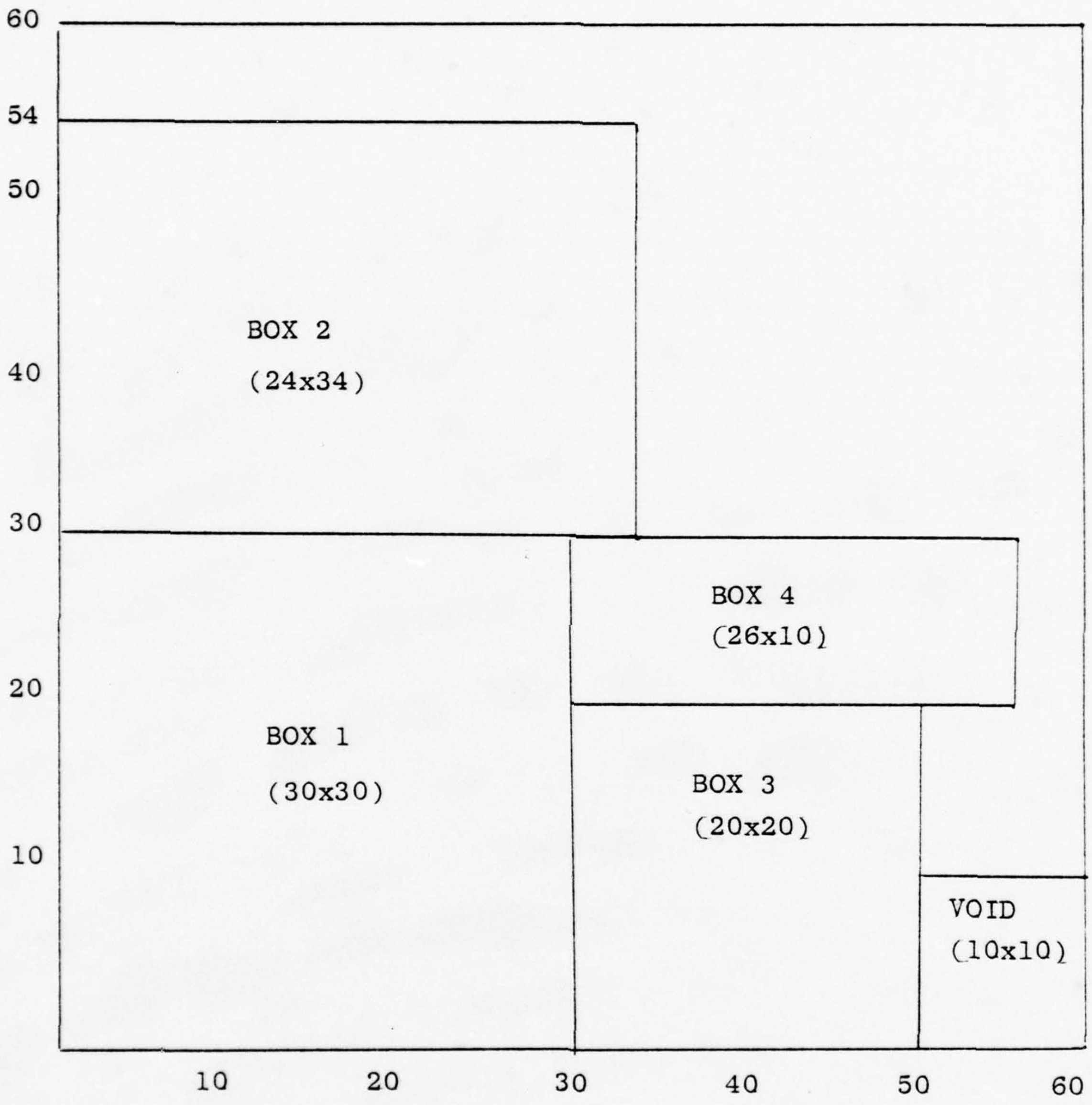


Figure 4. Example Problem.

$$A(1,2) < (ORX+BOXL_{2_2}) \quad \text{and} \quad A(1,5) > ORX \quad \text{and}$$

$$A(1,3) < (ORY+BOXW_{2_2}) \quad \text{and} \quad A(1,6) > ORY \quad \text{and}$$

$$A(1,4) < (ORZ+BOXH_{2_2}) \quad \text{and} \quad A(1,7) > ORZ$$

This equates to:

$$50 < (30+34) \quad \text{and} \quad 60 > 30 \quad \text{and}$$

$$0 < (0+24) \quad \text{and} \quad 60 > 0 \quad \text{and}$$

$$0 < (0+96) \quad \text{and} \quad 96 > 0$$

which is obviously true. Therefore, the box will not fit at this origin because it intersects the void.

Step 6. Select the next permissible origin which is related to B(2,2) i.e.(A(2,2), A(2,6), A(2,4)) or (0, 30, 0).

Step 7A. Determine if box will fit at this origin. The following logic checks are made for rows one and two, respectively:

$$50 < 34 \text{ and } 60 > 0 \text{ and } 0 < 54 \text{ and } 10 > 30 \text{ and}$$
$$0 < 96 \text{ and } 96 > 0 \text{ and}$$
$$0 < 34 \text{ and } 30 > 0 \text{ and } 0 < 54 \text{ and } 30 < 30 \text{ and}$$
$$0 < 96 \text{ and } 96 > 0$$

Since for rows one and two, respectively, $50 \nless 34$ and $30 \nless 30$, all logic checks are false and the box will, therefore, fit at this origin.

Step 7B. Improve the density of packing if possible. Since by simple inspection the box may not be moved toward the origin of the pallet, a detailed inspection will not be presented here. A detailed inspection will be given at a subsequent box.

Step 8. The second box is supported since the example deals with only two dimensions.

Step 9. Update matrices A and B which are now:

$$A = \begin{pmatrix} 1 & 50 & 0 & 0 & 60 & 10 & 96 \\ 1 & 0 & 0 & 0 & 30 & 30 & 96 \\ 2 & 0 & 0 & 0 & 34 & 54 & 96 \end{pmatrix}$$

$$B = \begin{pmatrix} F & F & F \\ T & F & F \\ T & F & F \end{pmatrix}$$

Step 5. Select box number 3 (20x20x96).

Step 6. Select origin (30,0,0), B=(2,1).

Step 7. Box will fit and improvement is not possible.

Step 8. This box is supported.

Step 9. Adjust A and B matrices as follows:

$$A = \begin{pmatrix} 1 & 50 & 0 & 0 & 60 & 10 & 96 \\ 1 & 0 & 0 & 0 & 30 & 30 & 96 \\ 2 & 0 & 30 & 0 & 34 & 54 & 96 \\ 3 & 30 & 0 & 0 & 50 & 20 & 96 \end{pmatrix}$$

$$B = \begin{pmatrix} F & F & F \\ F & F & F \\ T & F & F \\ F & T & F \end{pmatrix}$$

Step 5. Select box 4 (26x10).

Step 6. Select origin (34,30,0), B=(3,1).

Step 7A. Box will fit.

Step 7B. Improve density. Inspect each row of matrix A for an improving direction in the x, y, and z directions. For illustration, the improvement in the y direction will be shown. It is first necessary to find the row of matrix A in which the following relationship does not hold (as discussed in Section IIIE above). Note the origin is (ORX, ORY, ORZ) = (34,30,0) and that the box dimensions are ($BOXL_{4_4}$, $BOXW_{4_4}$, $BOXH_{4_4}$) = (26x10x96).

For illustration, the inspection of row four of matrix A is shown. This corresponds to box number 3.

A(4,2) > (34+26) or A(4,3) > 30 or A(4,4) > 96 or
A(4,5) < 34 or A(4,7) < 0 or A(4,6) < 30.

This results in: 30 > 60 or 0 > 30 or 0 > 96 or
50 < 34 or 96 < 0 or 20 < 30

Since the last term (20 < 30) is true, improvement in the y direction is possible. This improvement is 30-20 = 10. Therefore, the origin of the box is now defined to be (34,20,0). A similar inspection in the x direction would show an improvement of 4 units, making the final origin (30,20,0).

Step 8. Box is supported.

Step 9. Update matrices A and B as follows:

$$A = \begin{pmatrix} 1 & 50 & 0 & 0 & 60 & 10 & 96 \\ 1 & 0 & 0 & 0 & 30 & 30 & 96 \\ 2 & 0 & 30 & 0 & 34 & 54 & 96 \\ 3 & 30 & 0 & 0 & 50 & 20 & 96 \\ 4 & 30 & 20 & 0 & 56 & 30 & 96 \end{pmatrix}$$

Matrix A now shows the pallet as it was loaded. Column one gives the identification number of the box, columns two, three and four give the origin of the box, and columns five, six and seven give the orientation of the box.

Figure 4 shows the result of the above example.

G. OPTIMIZATION

In order to move from a local optimum toward the global optimum the following branching search can be made as suggested by Stoyan [4]:

Step 1. An initial sequence A_o of the boxes BOX_i, i = (1,n), is chosen, the boxes are palletized, and the pallets are loaded in the container.

Step 2. Using a uniform (1,n) pseudorandum number generator, s random numbers are generated ($s \ll n$) and the boxes with these s indexes are shuffled.

Step 3. The new sequence is loaded and the efficiency of the new local optimum is measured. If the efficiency improved go to step number 2.

Step 4. Replace the sequence as it existed prior to the shuffle and go to step 2.

The above algorithm is repeated until either a predetermined amount of time is consumed or a predetermined efficiency is reached.

H. OBTAINING A BETTER INITIAL SOLUTION

Several approaches were found useful in obtaining a better initial solution. These included defining the input sequence of the boxes to be loaded according to a preconceived routine, selecting an optimal maximum number of turns of the input boxes, and sorting the pallets prior to stuffing the pallets into the container. These approaches are discussed at the end of the next chapter.

IV. VERIFICATION

A. RELATED ALGORITHMS

Because the problems solved by Galata and Stoyan [4] and by DeSha [2] are subsets of the stuffing problem, verification of the current stuffing algorithm was possible by (a) solving the same example as presented by Galata and Stoyan and (b) by using DeSha's FORTRAN program to solve a problem which had been also solved by the stuffing algorithm.

A solution to the minimization example of Galata and Stoyan was obtained by the stuffing algorithm in 0.9 seconds with a minimum value of z = 23.2 as opposed to the original solution of z = 24.5 obtained by 540 searches which took approximately 31 seconds each. According to reference 4, in their problem Galata and Stoyan loaded 95 rectangular solid items onto a base with the objective of minimizing the height, z, which was needed in order to fit all 95 solids onto the base. In their problem, boxes need not be supported.

The FORTRAN program given by DeSha was used to solve the sample data discussed below: DeSha's solution showed a 86.9% volume efficiency and an area efficiency of 92%. The stuffing algorithm showed an 89.1% volume efficiency and a 95% area efficiency.

B. SAMPLE DATA

In order to fully test the stuffing algorithm, actual data were collected at Naval Supply Center, Oakland, California, Navy Exchange Retail Distribution Center in December 1978. These data, as shown in table I, were gathered by individually measuring boxes which were actually loaded (stuffed) into an eight-foot by eight-foot by forty-foot shipping container. The container was loaded in six hours by two men who utilized one forklift truck. The men were asked to load the container as efficiently as possible in order to measure their abilities against simulated stuffing by a computer program. The cargo was not palletized externally to the container but pallets were used implicitly in the container. The pallets consisted of larger boxes placed on the floor of the container upon which smaller boxes were stacked. All boxes which were loaded were load-bearing boxes, and weight and center of gravity considerations were not addressed by the loading crew. Actual effectiveness achieved by the loading crew was 87% as compared to reported Naval Supply Center, Oakland averages of 90% for Navy Exchange cargo and 80% for general cargo (which consists of repair parts, equipment, and general consumable supplies).

C. SAMPLE DATA RESULTS

The above sample data were utilized to test the stuffing algorithm utilizing various ordering of the input stream of boxes and various number of turns allowed. Table II gives the results of a nine by seven factorial experiment layout. During this experiment, pallets were defined to be the next box in the ordered input stream of boxes to be loaded. Each cell of table II gives the numbers of pallets necessary to stack all boxes, the IBM 360 computer run time in seconds necessary to load the boxes onto pallets and to stuff the pallets into containers, and the percent volumetric efficiency realized. Table III performs the same experiment except pallets were defined to be a standard pallet (40 inch by 44 inch) if the next box in the input stream of boxes could be contained in this standard pallet. If the box could not be contained, the pallet was defined to be the box itself.

Sample output of the FORTRAN program is presented in tables IV through VII for the test case in which standard pallets were not utilized and the input stream of boxes was ordered (highest to lowest) according to the boxes' base perimeter (sort control 4 as defined in the FORTRAN program) shown in the Computer Program section below. Table IV shows the boxes after they were ordered, table V shows the first pallet which was stacked, table VI presents a summary of all pallets which were stacked, and table VII shows the results of stuffing the pallets into the container.

Similar results are shown in tables VIII through XI for the test case in which standard pallets were utilized and the input stream of boxes was ordered according to the boxes' height.

D. ANALYSIS OF VARIANCE

An analysis of variance of tables II and III was performed in order to determine if a difference in efficiency of loading existed due to different methods of sorts and due to different number of turns allowed of the box. A level of significance of α =.10 was chosen. These analyses showed that for the case in which standard pallets were not utilized, there was a significance between the type of sort utilized and there was a difference between the number of turns allowed. (The F statistic for these tests was, respectively, $F_{(6,48)} = 3.78$ and $F_{(8,48)} = 57.9$.) These analyses for the case in which standard pallets were utilized showed there was a difference in the number of turns allowed ($F_{(6,48)} = 1.97$) and that there was a difference in the type of sort utilized ($F_{(8,48)} = 3.91$).

Although the data in tables II and III indicate that strong interaction may exist between the number of turns allowed and the type of sort utilized, the lack of replications of the experiment due to the existence of only one set of data make the existence of possible interaction impossible to verify.

E. RANGE TEST AND CONFIDENCE LIMITS

In order to determine which type of sorts in tables II and III and which number of turns in table II and table III produced the highest efficiences, Newman-Keuls range tests [13] were performed. The results of these tests are shown in tables XII and XIII in which an 'x' indicates a difference in the means and an '0' indicates no difference in the means. Of course, the range test was performed at the level of significance of $\alpha = .10$.

Tables XII and XIII also give 90-percent confidence limits on the means listed therein.

F. DISCUSSION OF TECHNIQUES TO IMPROVE SOLUTIONS

Table XII shows that the greatest efficiencies when loading boxes without standard pallets occurs when boxes are sorted by area prior to their stacking. Stoyan's procedure [4], discussed in Chapter V, was then applied in an attempt to improve on that solution. No improvements were found in 97 separate trials with four pairs of boxes interchanged on each trial. These trials tend to confirm that for the specific data available and for the specific method selected to load the boxes, the area ordering represents the best initial local optimum for the stuffing algorithm (when loading boxes without standard pallets). A similar test was conducted in the case where standard pallets were utilized. In this test 10 separate trials were conducted during which no improvement was found.

V. CONCLUSIONS AND RECOMMENDATIONS

The stuffing algorithm presented in this paper has demonstrated the possibility of achieving slightly better "loading" performance than is usually obtainable by experienced loading personnel. However, its major advantages may be in its ability to allow reasonably accurate predictions of container requirements and in the increased speed with which a container may be loaded because the loading personnel have a plan to follow.

This type of algorithm may be capable of allowing the full potential of mechanized warehouses to be realized by allowing the shipping department to "call forth" issues from the storage department when transportation assets are available. This would allow issue documents to accumulate in the mechanized warehouse's computerized data system as opposed to accumulating the issued material on the shipping dock.

Additional data are required to verify or disprove the results presented in this study and to determine the existence of the suspected interaction in the analysis of variance for the sorting of the boxes and the effects of the number of turns allowed.

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
1.000	4.000	14.000	10.000	8.000
2.000	2.000	20.000	15.000	8.000
3.000	1.000	27.000	27.000	7.000
4.000	20.000	10.000	8.000	9.000
5.000	14.000	12.000	8.000	6.000
6.000	6.000	12.000	9.000	7.000
7.000	6.000	9.000	9.000	6.000
8.000	4.000	25.000	25.000	8.000
9.000	3.000	34.000	26.000	5.000
10.000	1.000	21.000	14.000	12.000
11.000	1.000	46.000	26.000	30.000
12.000	1.000	12.000	9.000	9.000
13.000	2.000	26.000	16.000	18.000
14.000	4.000	25.000	20.000	27.000
15.000	11.000	12.000	10.000	10.000
16.000	4.000	13.000	12.000	10.000
17.000	1.000	16.000	15.000	12.000
18.000	1.000	26.000	16.000	15.000
19.000	7.000	23.000	23.000	30.000
20.000	1.000	24.000	24.000	35.000
21.000	1.000	12.000	12.000	15.000
22.000	50.000	13.000	13.000	12.000
23.000	8.000	18.000	18.000	8.000
24.000	12.000	19.000	8.000	9.000
25.000	6.000	9.000	8.000	6.000
26.000	8.000	7.000	6.000	4.000
27.000	42.000	9.000	8.000	9.000
28.000	2.000	16.000	11.000	18.000
29.000	4.000	18.000	12.000	8.000
30.000	4.000	16.000	10.000	8.000
31.000	1.000	26.000	20.000	28.000
32.000	2.000	13.000	10.000	7.000
33.000	2.000	18.000	10.000	8.000
34.000	4.000	15.000	10.000	8.000
35.000	5.000	21.000	14.000	7.000
36.000	7.000	16.000	12.000	8.000
37.000	6.000	12.000	6.000	8.000
38.000	12.000	12.000	9.000	8.000
39.000	24.000	12.000	10.000	8.000
40.000	2.000	11.000	10.000	7.000
41.000	3.000	12.000	12.000	7.000
42.000	4.000	9.000	7.000	6.000
43.000	5.000	11.000	8.000	8.000
44.000	18.000	16.000	11.000	6.000
45.000	1.000	13.000	12.000	13.000
46.000	2.000	20.000	16.000	21.000
47.000	1.000	20.000	15.000	31.000
48.000	2.000	29.000	20.000	31.000
49.000	14.000	22.000	9.000	10.000
50.000	1.000	47.000	26.000	15.000
51.000	2.000	65.000	21.000	19.000
52.000	1.000	12.000	12.000	13.000
53.000	4.000	18.000	13.000	14.000
54.000	3.000	57.000	11.000	10.000
55.000	5.000	60.000	10.000	6.000
56.000	3.000	13.000	7.000	6.000
57.000	2.000	15.000	8.000	8.000
58.000	8.000	12.000	8.000	9.000
59.000	1.000	35.000	25.000	5.000
60.000	12.000	27.000	20.000	18.000
61.000	18.000	8.000	8.000	6.000
62.000	26.000	10.000	6.000	6.000
63.000	10.000	12.000	8.000	8.000

SAMPLE DATA

TABLE I

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
64.000	12.000	9.000	6.000	8.000
65.000	1.000	32.000	18.000	22.000
66.000	1.000	32.000	18.000	15.000
67.000	1.000	32.000	18.000	20.000
68.000	2.000	25.000	23.000	15.000
69.000	1.000	25.000	25.000	16.000
70.000	1.000	25.000	23.000	23.000
71.000	1.000	35.000	23.000	24.000
72.000	1.000	46.000	40.000	20.000
73.000	1.000	46.000	44.000	34.000
74.000	3.000	10.000	8.000	6.000
75.000	14.000	12.000	9.000	6.000
76.000	24.000	14.000	6.000	16.000
77.000	3.000	20.000	14.000	12.000
78.000	1.000	47.000	40.000	43.000
79.000	2.000	24.000	19.000	15.000
80.000	2.000	24.000	19.000	13.000
81.000	1.000	43.000	36.000	51.000
82.000	1.000	24.000	18.000	29.000
83.000	1.000	43.000	43.000	46.000
84.000	2.000	26.000	15.000	17.000
85.000	1.000	24.000	16.000	10.000
86.000	1.000	48.000	40.000	62.000
87.000	1.000	27.000	24.000	18.000
88.000	1.000	36.000	24.000	28.000
89.000	1.000	12.000	12.000	16.000
90.000	2.000	11.000	8.000	10.000
91.000	4.000	8.000	7.000	8.000
92.000	10.000	8.000	7.000	6.000
93.000	1.000	24.000	19.000	32.000
94.000	4.000	13.000	8.000	8.000
95.000	12.000	12.000	11.000	9.000
96.000	7.000	12.000	10.000	8.000
97.000	3.000	20.000	6.000	8.000
98.000	158.000	18.000	12.000	7.000
99.000	1.000	52.000	44.000	48.000
100.000	1.000	50.000	42.000	39.000
101.000	1.000	49.000	42.000	38.000
102.000	1.000	48.000	40.000	72.000
103.000	3.000	13.000	10.000	12.000
104.000	1.000	48.000	42.000	40.000
105.000	1.000	48.000	42.000	26.000
106.000	2.000	20.000	11.000	10.000
107.000	16.000	22.000	6.000	9.000
108.000	8.000	12.000	8.000	18.000
109.000	8.000	12.000	8.000	13.000
110.000	8.000	12.000	9.000	14.000
111.000	12.000	12.000	9.000	5.000
112.000	6.000	18.000	14.000	6.000
113.000	1.000	14.000	11.000	10.000
114.000	25.000	23.000	9.000	10.000
115.000	2.000	20.000	12.000	14.000
116.000	1.000	21.000	16.000	19.000
117.000	17.000	12.000	8.000	5.000
118.000	3.000	14.000	14.000	8.000
119.000	60.000	8.000	6.000	8.000
120.000	1.000	50.000	22.000	8.000
121.000	2.000	16.000	14.000	9.000
122.000	16.000	8.000	6.000	5.000
123.000	20.000	9.000	6.000	5.000
124.000	3.000	16.000	14.000	22.000
125.000	20.000	12.000	8.000	7.000
126.000	1.000	23.000	14.000	15.000
127.000	4.000	26.000	22.000	14.000

SAMPLE DATA

TABLE I (CONTINUED)

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
128.000	2.000	13.000	8.000	10.000
129.000	5.000	10.000	8.000	8.000
130.000	3.000	24.000	16.000	20.000
131.000	60.000	10.000	10.000	8.000
132.000	2.000	8.000	8.000	8.000
133.000	10.000	13.000	9.000	4.000
134.000	3.000	10.000	9.000	11.000
135.000	3.000	18.000	12.000	9.000
136.000	1.000	11.000	11.000	10.000
137.000	1.000	20.000	16.000	17.000
138.000	2.000	22.000	19.000	15.000
139.000	3.000	20.000	19.000	10.000
140.000	2.000	15.000	10.000	12.000
141.000	1.000	17.000	12.000	14.000
142.000	2.000	20.000	13.000	16.000
143.000	3.000	16.000	16.000	5.000
144.000	1.000	26.000	18.000	16.000
145.000	6.000	9.000	8.000	7.000
146.000	14.000	10.000	7.000	7.000
147.000	4.000	12.000	10.000	10.000
148.000	9.000	18.000	12.000	6.000
149.000	2.000	14.000	8.000	10.000
150.000	27.000	16.000	16.000	16.000
151.000	69.000	18.000	16.000	14.000
152.000	80.000	19.000	12.000	7.000
153.000	1.000	48.000	42.000	4.000
154.000	12.000	15.000	14.000	14.000
155.000	27.000	9.000	7.000	7.000

SAMPLE DATA

TABLE I (CONTINUED)

TYPE SORT (input)	NUMBER OF TURNS OF BOX ALLOWED							
	0	1	2	3	4	5	6	Mean
No Sort (0)	81 40.7 72.87	88 79.0 73.00	88 68.3 73.14	93 102.2 75.12	93 105.2 74.4	88 139.9 78.15	88 137.5 78.15	88 96.1 74.98
Height (1)	91 30.5 82.65	97 50.6 73.70	97 49.5 73.70	95 89.7 79.11	95 84.7 81.10	96 119.1 81.27	96 113.1 85.01	95 76.7 79.51
Length (2)	117 13.3 85.95	118 21.6 86.14	118 22.1 86.53	126 29.6 84.82	126 28.3 84.82	124 44.2 86.33	124 41.3 85.01	122 28.6 85.65
Width (3)	111 13.2 85.20	111 24.5 85.01	111 23.8 84.45	115 36.4 85.20	115 34.2 85.20	120 56.3 86.33	120 51.9 85.76	115 34.3 85.31
Area (4)	125 7.6 88.50	124 11.6 89.11	124 11.3 89.11	128 19.3 85.57	128 19.2 85.76	129 23.9 86.14	129 25.5 86.33	127 16.9 87.22
Volume (5)	103 16.4 8.144	97 45.1 81.95	97 43.3 83.01	126 24.4 84.45	126 24.3 84.45	128 28.0 86.72	128 27.0 86.72	116 29.8 84.11
Total Volume (6)	74 69.0 72.87	72 121.7 72.87	72 119.8 73.98	79 191.5 78.15	79 188.0 78.63	80 320.0 75.42	80 344.3 75.42	77 193.5 75.33
Random (7)	73 89.7 67.18	66 129.5 70.23	66 127.3 70.23	73 24.79 73.56	73 14.0 73.00	69 341.9 72.87	69 342.0 72.87	70 184.6 71.42
Number Boxes (8)	123 8.4 70.48	123 11.6 70.22	123 11.3 70.48	123 26.5 73.48	123 25.7 73.14	121 36.7 77.22	121 36.9 77.22	123 22.4 73.14
Mean	100 32.1 78.57	100 55.0 78.03	100 53.0 78.29	106 85.3 79.94	106 58.2 80.06	106 123.3 81.16	106 124.4 81.39	103 75.8 79.63

CASE 1: Standard Pallets Not Used
Number of Pallets/Time to Stack Pallets (secs)/Total % Volume Efficiency

TABLE II

TYPE SORT (input)	NUMBER OF TURNS OF BOX ALLOWED							
	0	1	2	3	4	5	6	Mean
No Sort (0)	38 135.9 49.26	36 227.6 52.17	36 227.8 52.17	35 334.2 55.46	35 334.2 55.46	35 498.8 55.46	35 499.0 55.46	36 322.5 53.68
Height (1)	38 93.8 52.10	37 153.7 55.38	37 153.9 52.17	34 275.8 59.09	34 276.0 59.18	32 380.0 62.93	32 380.0 63.34	35 244.7 57.74
Length (2)	30 105.7 55.46	32 267.5 52.17	32 267.8 52.17	33 520.1 52.10	33 520.1 52.10	32 685.1 52.17	32 685.1 52.17	32 435.9 52.62
Width (3)	35 92.4 55.46	34 125.4 59.09	34 125.5 59.09	37 406.0 55.46	37 406.1 55.46	37 609.5 55.46	37 609.6 55.46	36 339.2 56.50
Area (4)	31 158.0 55.38	30 295.4 55.46	30 295.5 55.46	30 505.9 55.38	30 505.9 55.38	30 784.3 55.38	30 784.4 55.38	30 475.6 55.40
Volume (5)	34 165.4 52.10	32 243.2 55.38	32 243.3 55.38	33 536.2 52.10	33 536.3 52.10	32 684.9 52.10	32 684.9 52.10	33 442.3 53.04
Total Volume (6)	36 119.7 52.10	33 191.8 59.09	33 192.0 59.09	34 387.9 59.09	34 388.0 55.46	34 589.5 55.46	34 589.6 55.46	34 351.1 56.54
Random (7)	38 94.4 49.26	35 165.6 55.46	35 165.8 55.46	36 356.9 55.54	36 357.0 59.18	37 477.1 55.46	37 477.1 55.46	36 299.1 55.12
Number Boxes (8)	39 86.4 52.17	37 132.5 55.46	37 133.0 55.46	37 414.6 52.17	37 414.7 55.46	37 615.6 55.38	37 615.7 55.46	37 344.6 54.51
Mean	35 116.8 52.59	34 200.3 55.51	34 200.5 55.16	34 415.3 55.15	34 415.4 55.53	34 592.6 55.53	34 591.7 55.58	34 361.8 55.01

CASE 2: Standard Pallets Used
Number of Pallets/Time to Stack Pallets (secs)/Total % Volume Efficiency

TABLE III

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
99.000	1.000	52.000	44.000	48.000
100.000	1.000	50.000	42.000	39.000
101.000	1.000	49.000	42.000	38.000
73.000	1.000	46.000	44.000	34.000
105.000	1.000	48.000	42.000	26.000
153.000	1.000	48.000	42.000	4.000
104.000	1.000	48.000	42.000	40.000
102.000	1.000	48.000	40.000	72.000
86.000	1.000	48.000	40.000	62.000
78.000	1.000	47.000	40.000	43.000
83.000	1.000	43.000	43.000	46.000
72.000	1.000	46.000	40.000	20.000
81.000	1.000	43.000	36.000	51.000
51.000	2.000	65.000	21.000	19.000
50.000	1.000	47.000	26.000	15.000
11.000	1.000	46.000	26.000	30.000
120.000	1.000	50.000	22.000	8.000
9.000	3.000	34.000	26.000	5.000
59.000	1.000	35.000	25.000	5.000
88.000	1.000	36.000	24.000	28.000
71.000	1.000	35.000	23.000	24.000
3.000	1.000	27.000	27.000	7.000
87.000	1.000	27.000	24.000	18.000
54.000	3.000	57.000	11.000	10.000
69.000	1.000	25.000	25.000	16.000
8.000	4.000	25.000	25.000	8.000
55.000	5.000	60.000	10.000	6.000
48.000	2.000	29.000	20.000	31.000
65.000	1.000	32.000	18.000	22.000
66.000	1.000	32.000	18.000	15.000
67.000	1.000	32.000	18.000	20.000
20.000	1.000	24.000	24.000	35.000
68.000	2.000	25.000	23.000	15.000
70.000	1.000	25.000	23.000	23.000
127.000	4.000	26.000	22.000	14.000
60.000	12.000	27.000	20.000	18.000
19.000	7.000	23.000	23.000	30.000
31.000	1.000	26.000	20.000	28.000
14.000	4.000	25.000	20.000	27.000
144.000	1.000	26.000	18.000	16.000
79.000	2.000	24.000	19.000	15.000
80.000	2.000	24.000	19.000	13.000
93.000	1.000	24.000	19.000	32.000
82.000	1.000	24.000	18.000	29.000
138.000	2.000	22.000	19.000	15.000
13.000	2.000	26.000	16.000	18.000
18.000	1.000	26.000	16.000	15.000
84.000	2.000	26.000	15.000	17.000
130.000	3.000	24.000	16.000	20.000
85.000	1.000	24.000	16.000	10.000
139.000	3.000	20.000	19.000	10.000
116.000	1.000	21.000	16.000	19.000
23.000	8.000	18.000	18.000	8.000
126.000	1.000	23.000	14.000	15.000
46.000	2.000	20.000	16.000	21.000
137.000	1.000	20.000	16.000	17.000
2.000	2.000	20.000	15.000	8.000
47.000	1.000	20.000	15.000	31.000
10.000	1.000	21.000	14.000	12.000
35.000	5.000	21.000	14.000	7.000
151.000	69.000	18.000	16.000	14.000
77.000	3.000	20.000	14.000	12.000
142.000	2.000	20.000	13.000	16.000

SAMPLE DATA SORTED BY AREA

TABLE IV

143.000	3.000	16.000	16.000	5.000
150.C00	27.000	16.000	16.000	16.000
112.000	6.000	18.000	14.000	6.000
115.000	2.000	20.000	12.000	14.000
17.COC	1.000	16.000	15.000	12.000
53.000	4.000	18.000	13.000	14.000
152.000	80.000	19.000	12.000	7.000
121.C00	2.000	16.000	14.000	9.000
124.000	3.000	16.000	14.000	22.000
106.000	2.000	20.000	11.000	10.000
135.COC	3.000	18.000	12.000	9.000
148.000	9.000	18.000	12.000	6.000
29.CC0	4.000	18.000	12.000	8.000
98.000	158.000	18.000	12.000	7.000
154.000	12.000	15.000	14.000	14.000
114.C00	25.000	23.000	9.000	10.000
141.000	1.000	17.000	12.000	14.000
49.000	14.000	22.000	9.000	10.000
118.000	3.000	14.000	14.000	8.000
36.000	7.000	16.000	12.000	8.000
33.C00	2.000	18.000	10.000	8.000
28.000	2.000	16.000	11.000	18.000
44.000	18.000	16.000	11.000	6.000
22.COC	50.000	13.000	13.000	12.000
30.000	4.000	16.000	10.000	8.000
45.000	1.000	13.000	12.000	13.000
16.000	4.000	13.000	12.000	10.000
113.000	1.000	14.000	11.000	10.000
24.COC	12.000	19.000	8.000	9.000
140.000	2.000	15.00C	10.000	12.000
34.000	4.000	15.000	10.000	8.000
89.C00	1.000	12.000	12.000	16.000
41.000	3.000	12.000	12.000	7.000
52.000	1.000	12.000	12.000	13.000
21.000	1.000	12.000	12.000	15.000
1.000	4.000	14.000	10.000	8.000
95.COC	12.000	12.000	11.000	9.000
107.00C	16.000	22.000	6.000	9.000
32.000	2.000	13.000	10.000	7.000
103.COC	3.000	13.000	10.000	12.000
136.000	1.000	11.000	11.000	10.000
96.000	7.000	12.000	10.000	8.000
147.COC	4.000	12.000	10.000	10.000
57.000	2.000	15.000	8.000	8.000
15.COC	11.000	12.000	10.000	10.000
97.000	3.000	20.000	6.000	8.000
39.000	24.000	12.000	10.000	8.000
133.COC	10.000	13.000	9.000	4.000
149.000	2.000	14.000	8.00C	10.000
40.000	2.000	11.000	10.000	7.000
111.000	12.000	12.000	9.000	5.000
110.000	8.000	12.000	9.000	14.000
12.C00	1.000	12.000	9.000	9.000
75.COC	14.000	12.000	9.000	6.000
6.000	6.000	12.000	9.000	7.000
38.COC	12.000	12.00C	9.000	8.000
128.000	2.000	13.000	8.000	10.000
94.000	4.000	13.000	8.000	8.000
131.000	60.000	10.000	10.000	8.000
5.000	14.000	12.000	8.000	6.000
58.CCC	8.000	12.000	8.000	9.000
125.COC	20.000	12.000	8.000	7.000
117.000	17.000	12.000	8.000	5.000
108.COC	8.000	12.000	8.000	18.000
63.000	10.000	12.000	8.000	8.000

SAMPLE DATA SORTED BY AREA

TABLE IV (CONTINUED)

109.000	8.000	12.000	8.000	13.000
56.000	3.000	13.000	7.000	6.000
134.000	3.000	10.000	9.000	11.000
90.000	2.000	11.000	8.000	10.000
43.000	5.000	11.000	8.000	8.000
76.000	24.000	14.000	6.000	16.000
7.000	6.000	9.000	9.000	6.000
129.000	5.000	10.000	8.000	8.000
74.000	3.000	10.000	8.000	6.000
4.000	20.000	10.000	8.000	9.000
145.000	6.000	9.000	8.000	7.000
27.000	42.000	9.000	8.000	9.000
37.000	6.000	12.000	6.000	8.000
25.000	6.000	9.000	8.000	6.000
146.000	14.000	10.000	7.000	7.000
132.000	2.000	8.000	8.000	8.000
61.000	18.000	8.000	8.000	6.000
42.000	4.000	9.000	7.000	6.000
155.000	27.000	9.000	7.000	7.000
62.000	26.000	10.000	6.000	6.000
92.000	10.000	8.000	7.000	6.000
91.000	4.000	8.000	7.000	8.000
123.000	20.000	9.000	6.000	5.000
64.000	12.000	9.000	6.000	8.000
122.000	16.000	8.000	6.000	5.000
119.000	60.000	8.000	6.000	8.000
26.000	8.000	7.000	6.000	4.000

SAMPLE DATA SORTED BY AREA

TABLE IV (CONTINUED)

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.0	52.0	44.0	96.0	209464.0	C.1	95.4

FOLLOWING BOXES WERE STACKED

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
99.0	52.0	44.0	48.0	0.0	0.0	C.0	52.0	44.0	48.0
100.0	50.0	42.0	39.0	0.C	0.0	48.0	50.0	42.0	87.0
153.0	48.0	42.0	4.0	0.0	0.0	87.0	48.0	42.0	91.0
9.0	34.0	26.0	5.0	0.0	0.0	91.0	34.0	26.0	96.0
143.0	16.0	16.0	5.0	0.0	26.0	91.0	16.C	42.0	96.0
143.0	16.C	16.0	5.0	16.C	26.0	91.0	32.0	42.0	96.0
143.0	16.0	16.0	5.0	32.0	26.0	91.0	48.0	42.0	96.0
133.0	13.0	9.0	4.0	34.0	0.0	91.0	47.0	9.0	95.0
133.0	13.0	9.0	4.0	34.C	9.0	91.0	47.0	18.0	95.0
117.C	12.0	8.0	5.0	34.0	18.0	91.0	46.0	26.0	96.0

PALLET ONE CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE V

SUMMARY LINE FOR EACH PALET THAT WAS STACKED

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.00	52.00	44.00	96.00	209464.00	0.13	95.36
2.00	49.00	42.00	96.00	185392.0C	0.30	93.84
3.00	46.00	44.00	95.00	177963.0C	C.56	91.59
4.00	48.00	42.00	96.00	183865.00	0.32	95.00
5.00	48.00	40.00	96.CC	178934.0C	0.27	97.08
6.00	48.00	40.00	96.0C	170730.00	0.67	92.63
7.00	43.00	36.00	96.00	140113.00	0.77	94.28
8.00	65.00	21.00	96.CC	122932.0C	C.67	93.81
9.00	50.00	22.00	96.00	84564.00	0.65	80.08
10.00	35.00	23.00	95.00	57000.00	0.34	73.76
11.00	27.00	24.00	96.0C	51906.0C	0.15	83.44
12.00	25.00	25.00	95.00	52530.00	0.13	87.55
13.00	26.00	22.00	93.CC	50552.0C	0.08	92.06
14.00	27.00	20.00	96.00	51684.0C	C.08	99.70
15.00	27.00	20.00	96.00	51684.00	0.08	99.70
16.00	27.00	20.00	95.0C	46728.0C	0.10	90.14
17.00	23.00	23.00	96.00	50340.00	0.07	99.13
18.00	23.00	23.00	96.00	50340.00	0.07	99.13
19.00	25.00	20.00	92.00	41700.0C	C.C6	86.87
20.00	26.00	16.00	96.00	36888.00	0.07	92.37
21.00	24.00	16.00	94.CC	32496.0C	0.06	88.15
22.00	18.00	18.00	95.0C	27360.0C	0.07	87.96
23.00	20.00	16.00	92.00	27216.00	0.06	88.59
24.00	18.00	16.00	96.CC	27072.0C	0.06	97.92
25.00	18.00	16.00	93.00	26208.0C	0.06	94.79
26.00	18.00	16.00	93.00	26136.00	0.06	94.53
27.00	18.00	16.00	93.00	26136.CC	C.06	94.53
28.00	18.00	16.00	93.00	26136.00	0.06	94.53
29.00	18.00	16.00	96.CC	26784.0C	0.07	96.88
30.00	18.00	16.00	96.00	26784.0C	C.07	96.88
31.00	18.00	16.00	96.00	26544.00	0.07	96.01
32.00	18.00	16.00	92.0C	25920.0C	0.06	93.75
33.00	18.00	16.00	92.00	25920.00	0.06	93.75
34.00	18.00	16.00	96.00	25440.00	0.06	92.01
35.00	20.00	13.00	95.00	22232.00	C.C6	89.07
36.00	16.00	16.00	96.00	24576.00	0.06	100.00
37.00	16.00	16.00	96.CC	24576.CC	0.05	100.00

SUMMARY OF ALL PALLETS LOADED (LOADING WITHOUT STANDARD PALLETS)

TABLE VI

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
38.00	16.00	16.00	96.00	24576.00	0.06	10C.00
39.C0	16.00	16.00	96.00	24576.00	0.05	100.00
40.00	19.00	12.00	96.00	21558.CC	C.C9	98.49
41.00	19.00	12.00	96.00	21558.00	0.09	98.49
42.00	19.00	12.00	96.0C	21558.0C	0.09	98.49
43.00	19.00	12.00	96.00	21558.CC	0.09	98.49
44.00	19.00	12.00	96.00	21558.00	0.09	98.49
45.00	19.00	12.00	96.CC	20886.0C	0.08	95.42
46.00	16.00	14.00	94.00	20664.00	0.04	96.09
47.00	20.00	11.00	95.00	15300.00	0.07	72.44
48.00	18.00	12.00	96.00	20466.0C	0.C8	98.70
49.00	18.00	12.00	96.00	20406.00	0.08	98.41
50.0C	18.00	12.00	96.CC	20406.0C	0.08	98.41
51.00	18.00	12.00	96.00	20406.0C	C.C8	98.41
52.00	18.00	12.00	96.00	20406.00	0.08	98.41
53.00	18.00	12.00	96.CC	20406.0C	0.08	98.41
54.00	18.00	12.00	96.00	20406.00	0.08	98.41
55.0C	18.00	12.00	96.00	20376.00	0.08	98.26
56.00	18.00	12.00	96.00	20376.0C	C.C8	98.26
57.00	18.00	12.00	96.00	20376.00	0.08	98.26
58.00	18.00	12.00	96.0C	20376.0C	0.08	98.26
59.00	18.03	12.00	96.00	19344.0C	0.07	93.29
60.00	15.00	14.00	92.00	19208.00	0.05	95.28
61.00	15.00	14.00	96.CC	18952.00	0.06	94.01
62.00	23.00	9.00	96.00	19764.00	0.05	99.46
63.00	23.00	9.00	96.00	19764.00	0.05	99.46
64.00	23.00	9.00	96.0C	19494.0C	0.05	98.10
65.00	22.00	9.00	96.00	18954.00	0.05	99.72
66.C0	22.00	9.00	92.CC	14904.00	0.05	78.41
67.00	16.00	12.00	92.00	16320.0C	C.C6	88.54
68.00	16.00	11.00	92.00	13624.00	0.06	80.63
69.00	13.00	13.00	96.0C	16224.0C	0.05	100.00
70.00	13.00	13.00	96.00	16224.00	0.05	100.0C
71.00	13.00	13.00	96.00	16224.00	0.05	100.00
72.00	13.00	13.00	96.00	16224.0C	C.C5	100.00
73.00	13.00	13.00	96.00	16224.00	0.05	100.00
74.00	13.00	13.00	96.0C	16224.0C	0.05	100.00
75.00	13.00	12.00	96.0C	14244.0C	0.05	95.11
76.00	19.00	8.00	96.00	10568.00	0.07	72.42

SUMMARY OF ALL PALLETS LOADED (LOADING WITHOUT STANDARD PALLETS)

TABLE VI (CONTINUED)

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
77.00	15.00	10.00	96.00	12240.00	0.04	85.00
78.00	12.00	12.00	96.00	12828.00	0.05	92.80
79.00	12.00	11.00	93.00	11090.0C	C.04	87.52
80.00	22.00	6.00	96.00	12600.00	0.03	99.43
81.00	22.00	6.00	95.CC	9036.0C	0.04	71.31
82.00	12.00	10.00	96.00	11448.0C	C.C4	99.37
83.C0	12.00	10.00	92.00	11040.00	0.04	95.83
84.00	12.00	10.00	96.0C	11520.0C	0.04	100.00
85.00	12.00	10.00	92.00	10704.00	0.04	92.92
86.00	12.00	9.00	93.00	10044.00	0.03	96.88
87.00	12.00	9.00	96.00	10296.0C	C.C4	99.31
88.00	12.00	9.00	96.00	10368.00	0.04	100.00
89.00	10.00	10.00	96.0C	9600.0C	0.04	100.00
90.00	10.00	10.00	96.00	9600.0C	C.04	100.00
91.00	10.00	10.00	96.00	9600.00	0.04	100.00
92.00	10.00	10.00	96.CC	9600.00	0.04	100.00
93.00	10.00	10.00	94.00	9180.00	0.03	95.62
94.00	12.00	8.00	93.00	8928.00	0.04	96.88
95.00	12.00	8.00	94.00	9024.0C	0.03	97.92
96.00	12.00	8.00	96.00	9216.00	0.04	100.00
97.00	12.00	8.00	95.CC	9120.00	0.02	98.96
98.00	12.00	8.00	96.00	9216.0C	C.02	100.00
99.00	12.00	8.00	95.00	9056.00	0.02	98.26
100.00	12.00	8.00	95.CC	8640.0C	0.02	93.75
101.00	10.00	9.00	92.00	6930.00	0.03	80.21
102.C0	14.00	6.00	96.00	8064.00	0.01	100.00
103.00	14.00	6.00	96.00	8064.0C	C.C1	100.00
104.00	14.00	6.00	96.00	8064.0C	0.01	100.00
105.00	10.00	8.00	94.CC	7520.0C	0.02	97.92
106.00	10.00	8.00	96.00	7632.0C	0.02	99.37
107.00	10.00	8.00	96.00	7056.00	0.02	91.87
108.00	9.00	8.00	96.CC	6912.0C	0.02	100.00
109.00	9.00	8.00	96.00	6912.0C	0.02	100.00
110.00	9.00	8.00	96.00	6864.00	0.02	99.31
111.00	9.00	8.00	96.00	6136.CC	0.03	88.77
112.00	12.00	6.00	96.00	6048.00	0.03	87.50
113.00	10.00	7.00	96.CC	6610.0C	0.02	98.36
114.00	10.00	7.00	96.00	6097.0C	C.C2	90.73
115.00	8.00	8.00	96.00	5664.00	0.03	92.19

SUMMARY OF ALL PALLETS LOADED (LOADING WITHOLT STANCARC PALLETS)

TABLE VI (CCNTINUED)

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
116.00	9.00	7.00	96.00	5973.00	0.02	98.76
117.00	9.00	7.00	90.00	4891.00	0.02	80.87
118.00	10.00	6.00	96.00	5472.00	0.03	95.00
119.00	9.00	6.00	96.00	4896.00	0.02	94.44
120.00	8.00	6.00	96.00	4608.00	0.02	100.00
121.00	8.00	6.00	96.00	4608.00	0.02	100.00
122.00	8.00	6.00	96.00	4608.00	0.02	100.00
123.00	8.00	6.00	96.00	4608.00	0.02	100.00
124.00	8.00	6.00	16.00	768.00	0.00	16.67

SUMMARY OF ALL PALLETS LOADED (LOADING WITHOUT STANDARD PALLETS)

TABLE VI (CONTINUED)

CONTAINER NR	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.0	96.0	480.0	96.0	3819456.0	3.0	86.3

FOLLOWING BOXES WERE STACKED

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
1.0	44.0	52.0	96.0	21.0	0.0	0.0	65.0	52.0	96.0
9.0	22.0	50.0	96.0	65.0	0.0	0.0	87.0	50.0	96.0
2.0	42.0	49.0	96.0	0.0	65.0	0.0	42.0	114.0	96.0
4.0	42.0	48.0	96.0	42.0	52.0	0.0	84.0	100.0	96.0
5.0	40.0	48.0	96.0	0.0	114.0	0.0	40.0	162.0	96.0
6.0	40.0	48.0	96.0	40.0	114.0	0.0	80.0	162.0	96.0
3.0	44.0	46.0	96.0	0.0	162.0	0.0	44.0	208.0	96.0
7.0	36.0	43.0	96.0	44.0	162.0	0.0	80.0	205.0	96.0
10.0	23.0	35.0	96.0	0.0	208.0	0.0	23.0	243.0	96.0
11.0	24.0	27.0	96.0	23.0	208.0	0.0	47.0	235.0	96.0
14.0	20.0	27.0	96.0	47.0	205.0	0.0	67.0	232.0	96.0
15.0	20.0	27.0	96.0	67.0	205.0	0.0	87.0	232.0	96.0
16.0	20.0	27.0	96.0	0.0	243.0	0.0	20.0	270.0	96.0
20.0	16.0	26.0	96.0	80.0	100.0	0.0	96.0	126.0	96.0
13.0	22.0	26.0	96.0	20.0	243.0	0.0	42.0	269.0	96.0
19.0	20.0	25.0	96.0	42.0	235.0	0.0	62.0	260.0	96.0
12.0	25.0	25.0	96.0	62.0	232.0	0.0	87.0	257.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
21.0	16.0	24.0	96.0	80.0	126.0	0.0	96.0	150.0	96.0
18.0	23.0	23.0	96.0	0.0	270.0	0.0	23.0	293.0	96.0
17.0	23.0	23.0	96.0	23.0	269.0	0.0	46.0	292.0	96.0
63.0	9.0	23.0	96.0	87.0	0.0	0.0	96.0	23.0	96.0
64.0	9.0	23.0	96.0	84.0	50.0	0.0	93.0	73.0	96.0
62.0	9.0	23.0	96.0	80.0	150.0	0.0	89.0	173.0	96.0
81.0	6.0	22.0	96.0	80.0	173.0	0.0	86.0	195.0	96.0
65.0	9.0	22.0	96.0	46.0	260.0	0.0	55.0	282.0	96.0
66.0	9.0	22.0	96.0	86.0	173.0	0.0	95.0	195.0	96.0
80.0	6.0	22.0	96.0	89.0	150.0	0.0	95.0	172.0	96.0
35.0	13.0	20.0	96.0	55.0	260.0	0.0	68.0	280.0	96.0
23.0	16.0	20.0	96.0	68.0	257.0	0.0	84.0	277.0	96.0
47.0	11.0	20.0	96.0	84.0	257.0	0.0	95.0	277.0	96.0
45.0	12.0	19.0	96.0	0.0	293.0	0.0	12.0	312.0	96.0
44.0	12.0	19.0	96.0	12.0	293.0	0.0	24.0	312.0	96.0
43.0	12.0	19.0	96.0	24.0	292.0	0.0	36.0	311.0	96.0
76.0	8.0	19.0	96.0	36.0	292.0	0.0	44.0	311.0	96.0
40.0	12.0	19.0	96.0	44.0	292.0	0.0	56.0	311.0	96.0
41.0	12.0	19.0	96.0	56.0	280.0	0.0	68.0	299.0	96.0
42.0	12.0	19.0	96.0	68.0	277.0	0.0	80.0	296.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BCXW	Z+BCXH
8.0	21.0	65.0	96.0	0.0	0.0	0.0	21.0	65.0	96.0
34.0	16.0	18.0	96.0	80.0	277.0	0.0	96.0	295.0	96.0
33.0	16.0	18.0	96.0	0.0	312.0	0.0	16.0	330.0	96.0
22.0	18.0	18.0	96.0	16.0	312.0	0.0	34.0	330.0	96.0
31.0	16.0	18.0	96.0	34.0	311.0	0.0	50.0	329.0	96.0
48.0	12.0	18.0	96.0	50.0	311.0	0.0	62.0	329.0	96.0
30.0	16.0	18.0	96.0	62.0	299.0	0.0	78.0	317.0	96.0
29.0	16.0	18.0	96.0	78.0	296.0	0.0	94.0	314.0	96.0
28.0	16.0	18.0	96.0	0.0	330.0	0.0	16.0	348.0	96.0
50.0	12.0	18.0	96.0	16.0	330.0	0.0	28.0	348.0	96.0
51.0	12.0	18.0	96.0	28.0	330.0	0.0	40.0	348.0	96.0
49.0	12.0	18.0	96.0	40.0	329.0	0.0	52.0	347.0	96.0
52.0	12.0	18.0	96.0	52.0	329.0	0.0	64.0	347.0	96.0
53.0	12.0	18.0	96.0	64.0	317.0	0.0	76.0	335.0	96.0
54.0	12.0	18.0	96.0	76.0	317.0	0.0	88.0	335.0	96.0
55.0	12.0	18.0	96.0	84.0	73.0	0.0	96.0	91.0	96.0
56.0	12.0	18.0	96.0	0.0	348.0	0.0	12.0	366.0	96.0
57.0	12.0	18.0	96.0	12.0	348.0	0.0	24.0	366.0	96.0
58.0	12.0	18.0	96.0	24.0	348.0	0.0	36.0	366.0	96.0
59.0	12.0	18.0	96.0	36.0	348.0	0.0	48.0	366.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
27.0	16.0	18.0	96.0	48.0	347.0	0.0	64.0	365.0	96.0
26.0	16.0	18.0	96.0	64.0	335.0	0.0	80.0	353.0	96.0
25.0	16.0	18.0	96.0	80.0	335.0	0.0	96.0	353.0	96.0
24.0	16.0	18.0	96.0	0.0	366.0	0.0	16.0	384.0	96.0
32.0	16.0	18.0	96.0	16.0	366.0	0.0	32.0	384.0	96.0
67.0	12.0	16.0	96.0	32.0	366.0	0.0	44.0	382.0	96.0
36.0	16.0	16.0	96.0	44.0	366.0	0.0	60.0	382.0	96.0
37.0	16.0	16.0	96.0	60.0	365.0	0.0	76.0	381.0	96.0
68.0	11.0	16.0	96.0	76.0	353.0	0.0	87.0	369.0	96.0
39.0	16.0	16.0	96.0	0.0	384.0	0.0	16.0	400.0	96.0
46.0	14.0	16.0	96.0	16.0	384.0	0.0	30.0	400.0	96.0
38.0	16.0	16.0	96.0	30.0	384.0	0.0	46.0	400.0	96.0
60.0	14.0	15.0	96.0	46.0	382.0	0.0	60.0	397.0	96.0
77.0	10.0	15.0	96.0	60.0	381.0	0.0	70.0	396.0	96.0
61.0	14.0	15.0	96.0	70.0	381.0	0.0	84.0	396.0	96.0
102.0	6.0	14.0	96.0	88.0	314.0	0.0	94.0	328.0	96.0
104.0	6.0	14.0	96.0	87.0	353.0	0.0	93.0	367.0	96.0
103.0	6.0	14.0	96.0	84.0	369.0	0.0	90.0	383.0	96.0
72.0	13.0	13.0	96.0	21.0	52.0	0.0	34.0	65.0	96.0
75.0	12.0	13.0	96.0	42.0	100.0	0.0	54.0	113.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
71.0	13.0	13.0	96.0	54.0	100.0	0.0	67.0	113.0	96.0
70.0	13.0	13.0	96.0	67.0	100.0	0.0	80.0	113.0	96.0
74.0	13.0	13.0	96.0	0.0	400.0	0.0	13.0	413.0	96.0
73.0	13.0	13.0	96.0	13.0	400.0	0.0	26.0	413.0	96.0
69.0	13.0	13.0	96.0	26.0	400.0	0.0	39.0	413.0	96.0
87.0	9.0	12.0	96.0	39.0	400.0	0.0	48.0	412.0	96.0
88.0	9.0	12.0	96.0	48.0	397.0	0.0	57.0	409.0	96.0
95.0	8.0	12.0	96.0	34.0	52.0	0.0	42.0	64.0	96.0
86.0	9.0	12.0	96.0	57.0	397.0	0.0	66.0	409.0	96.0
85.0	10.0	12.0	96.0	66.0	396.0	0.0	76.0	408.0	96.0
84.0	10.0	12.0	96.0	76.0	396.0	0.0	86.0	408.0	96.0
78.0	12.0	12.0	96.0	64.0	353.0	0.0	76.0	365.0	96.0
94.0	8.0	12.0	96.0	84.0	383.0	0.0	92.0	395.0	96.0
82.0	10.0	12.0	96.0	0.0	413.0	0.0	10.0	425.0	96.0
83.0	10.0	12.0	96.0	10.0	413.0	0.0	20.0	425.0	96.0
112.0	6.0	12.0	96.0	90.0	367.0	0.0	96.0	379.0	96.0
99.0	8.0	12.0	96.0	20.0	413.0	0.0	28.0	425.0	96.0
98.0	8.0	12.0	96.0	28.0	413.0	0.0	36.0	425.0	96.0
96.0	8.0	12.0	96.0	36.0	413.0	0.0	44.0	425.0	96.0
97.0	8.0	12.0	96.0	44.0	412.0	0.0	52.0	424.0	96.0
100.0	8.0	12.0	96.0	52.0	409.0	0.0	60.0	421.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
79.0	11.0	12.0	96.0	60.0	409.0	0.0	71.0	421.0	96.0
89.0	10.0	10.0	96.0	71.0	408.0	0.0	81.0	418.0	96.0
90.0	10.0	10.0	96.0	81.0	408.0	0.0	91.0	418.0	96.0
101.0	9.0	10.0	96.0	87.0	23.0	0.0	96.0	33.0	96.0
92.0	10.0	10.0	96.0	80.0	195.0	0.0	90.0	205.0	96.0
91.0	10.0	10.0	96.0	46.0	282.0	0.0	56.0	292.0	96.0
105.0	8.0	10.0	96.0	76.0	369.0	0.0	84.0	379.0	96.0
106.0	8.0	10.0	96.0	0.0	425.0	0.0	8.0	435.0	96.0
107.0	8.0	10.0	96.0	8.0	425.0	0.0	16.0	435.0	96.0
118.0	6.0	10.0	96.0	90.0	195.0	0.0	96.0	205.0	96.0
93.0	10.0	10.0	96.0	16.0	425.0	0.0	26.0	435.0	96.0
113.0	7.0	10.0	96.0	26.0	425.0	0.0	33.0	435.0	96.0
114.0	7.0	10.0	96.0	33.0	425.0	0.0	40.0	435.0	96.0
108.0	8.0	9.0	96.0	40.0	425.0	0.0	48.0	434.0	96.0
110.0	8.0	9.0	96.0	48.0	424.0	0.0	56.0	433.0	96.0
111.0	8.0	9.0	96.0	56.0	421.0	0.0	64.0	430.0	96.0
116.0	7.0	9.0	96.0	64.0	421.0	0.0	71.0	430.0	96.0
117.0	7.0	9.0	96.0	71.0	418.0	0.0	78.0	427.0	96.0
109.0	8.0	9.0	96.0	78.0	418.0	0.0	86.0	427.0	96.0
119.0	6.0	9.0	96.0	86.0	418.0	0.0	92.0	427.0	96.0

CONTAINER CONFIGURATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BCXW	Z+BOXH
115.0	8.0	8.0	96.0	23.0	235.0	0.0	31.0	243.0	96.0
120.0	6.0	8.0	96.0	31.0	235.0	0.0	37.0	243.0	96.0
121.0	6.0	8.0	96.0	56.0	299.0	0.0	62.0	307.0	96.0
122.0	6.0	8.0	96.0	84.0	91.0	0.0	90.0	99.0	96.0
123.0	6.0	8.0	96.0	90.0	91.0	0.0	96.0	99.0	96.0
124.0	6.0	8.0	96.0	87.0	33.0	0.0	93.0	41.0	96.0

CONTAINER CONFIGLRATION (LOADING WITHOUT STANDARD PALLETS)

TABLE VII (CONTINUED)

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
102.000	1.000	48.000	40.000	72.000
86.000	1.000	48.000	40.000	62.000
81.000	1.000	43.000	36.000	51.000
99.000	1.000	52.000	44.000	48.000
83.000	1.000	43.000	43.000	46.000
78.000	1.000	47.000	40.000	43.000
104.000	1.000	48.000	42.000	40.000
100.000	1.000	50.000	42.000	39.000
101.000	1.000	49.000	42.000	38.000
20.000	1.000	24.000	24.000	35.000
73.000	1.000	46.000	44.000	34.000
93.000	1.000	24.000	19.000	32.000
47.000	1.000	20.000	15.000	31.000
48.000	2.000	29.000	20.000	31.000
11.000	1.000	46.000	26.000	30.000
19.000	7.000	23.000	23.000	30.000
82.000	1.000	24.000	18.000	29.000
31.000	1.000	26.000	20.000	28.000
88.000	1.000	36.000	24.000	28.000
14.000	4.000	25.000	20.000	27.000
105.000	1.000	48.000	42.000	26.000
71.000	1.000	35.000	23.000	24.000
70.000	1.000	25.000	23.000	23.000
65.000	1.000	32.000	18.000	22.000
124.000	3.000	16.000	14.000	22.000
46.000	2.000	20.000	16.000	21.000
67.000	1.000	32.000	18.000	20.000
130.000	3.000	24.000	16.000	20.000
72.000	1.000	46.000	40.000	20.000
51.000	2.000	65.000	21.000	19.000
116.000	1.000	21.000	16.000	19.000
60.000	12.000	27.000	20.000	18.000
28.000	2.000	16.000	11.000	18.000
108.000	8.000	12.000	8.000	18.000
13.000	2.000	26.000	16.000	18.000
87.000	1.000	27.000	24.000	18.000
84.000	2.000	26.000	15.000	17.000
137.000	1.000	20.000	16.000	17.000
150.000	27.000	16.000	16.000	16.000
89.000	1.000	12.000	12.000	16.000
69.000	1.000	25.000	25.000	16.000
142.000	2.000	20.000	13.000	16.000
76.000	24.000	14.000	6.000	16.000
144.000	1.000	26.000	18.000	16.000
126.000	1.000	23.000	14.000	15.000
18.000	1.000	26.000	16.000	15.000
68.000	2.000	25.000	23.000	15.000
66.000	1.000	32.000	18.000	15.000
21.000	1.000	12.000	12.000	15.000
50.000	1.000	47.000	26.000	15.000
79.000	2.000	24.000	19.000	15.000
138.000	2.000	22.000	19.000	15.000
115.000	2.000	20.000	12.000	14.000
110.000	8.000	12.000	9.000	14.000
53.000	4.000	18.000	13.000	14.000
154.000	12.000	15.000	14.000	14.000
151.000	69.000	18.000	16.000	14.000
127.000	4.000	26.000	22.000	14.000
141.000	1.000	17.000	12.000	14.000
80.000	2.000	24.000	19.000	13.000
45.000	1.000	13.000	12.000	13.000
109.000	8.000	12.000	8.000	13.000

SAMPLE DATA SORTED BY HEIGHT

TABLE VIII

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
52.000	1.000	12.000	12.000	13.000
77.000	3.000	20.000	14.000	12.000
22.000	50.000	13.000	13.000	12.000
140.000	2.000	15.000	10.000	12.000
103.000	3.000	13.000	10.000	12.000
17.000	1.000	16.000	15.000	12.000
10.000	1.000	21.000	14.000	12.000
134.000	3.000	10.000	9.000	11.000
15.000	11.000	12.000	10.000	10.000
114.000	25.000	23.000	9.000	10.000
90.000	2.000	11.000	8.000	10.000
54.000	3.000	57.000	11.000	10.000
147.000	4.000	12.000	10.000	10.000
149.000	2.000	14.000	8.000	10.000
85.000	1.000	24.000	16.000	10.000
128.000	2.000	13.000	8.000	10.000
113.000	1.000	14.000	11.000	10.000
106.000	2.000	20.000	11.000	10.000
49.000	14.000	22.000	9.000	10.000
136.000	1.000	11.000	11.000	10.000
16.000	4.000	13.000	12.000	10.000
139.000	3.000	20.000	19.000	10.000
58.000	8.000	12.000	8.000	9.000
107.000	16.000	22.000	6.000	9.000
135.000	3.000	18.000	12.000	9.000
24.000	12.000	19.000	8.000	9.000
4.000	20.000	10.000	8.000	9.000
95.000	12.000	12.000	11.000	9.000
27.000	42.000	9.000	8.000	9.000
12.000	1.000	12.000	9.000	9.000
121.000	2.000	16.000	14.000	9.000
131.000	60.000	10.000	10.000	8.000
38.000	12.000	12.000	9.000	8.000
96.000	7.000	12.000	10.000	8.000
43.000	5.000	11.000	8.000	8.000
57.000	2.000	15.000	8.000	8.000
63.000	10.000	12.000	8.000	8.000
64.000	12.000	9.000	6.000	8.000
29.000	4.000	18.000	12.000	8.000
30.000	4.000	16.000	10.000	8.000
91.000	4.000	8.000	7.000	8.000
33.000	2.000	18.000	10.000	8.000
120.000	1.000	50.000	22.000	8.000
34.000	4.000	15.000	10.000	8.000
129.000	5.000	10.000	8.000	8.000
97.000	3.000	20.000	6.000	8.000
23.000	8.000	18.000	18.000	8.000
94.000	4.000	13.000	8.000	8.000
36.000	7.000	16.000	12.000	8.000
37.000	6.000	12.000	6.000	8.000
1.000	4.000	14.000	10.000	8.000
8.000	4.000	25.000	25.000	8.000
39.000	24.000	12.000	10.000	8.000
2.000	2.000	20.000	15.000	8.000
118.000	3.000	14.000	14.000	8.000
119.000	60.000	8.000	6.000	8.000
132.000	2.000	8.000	8.000	8.000
152.000	80.000	19.000	12.000	7.000
146.000	14.000	10.000	7.000	7.000
125.000	20.000	12.000	8.000	7.000
32.000	2.000	13.000	10.000	7.000
155.000	27.000	9.000	7.000	7.000

SAMPLE DATA SORTED BY HEIGHT

TABLE VIII (CONTINUED)

NR LINE	NR BOXES	LENGTH	WIDTH	HEIGHT
3.000	1.000	27.000	27.000	7.000
145.000	6.000	9.000	8.000	7.000
35.000	5.000	21.000	14.000	7.000
6.C00	6.000	12.000	9.000	7.000
98.C00	158.000	18.000	12.000	7.000
40.000	2.000	11.000	10.000	7.000
41.CCC	3.000	12.000	12.000	7.000
92.000	10.000	8.000	7.000	6.000
75.000	14.000	12.000	9.000	6.000
62.000	26.000	10.000	6.000	6.000
61.000	18.000	8.000	8.000	6.000
55.CCC	5.000	60.000	10.000	6.000
56.000	3.000	13.000	7.000	6.000
44.000	18.000	16.000	11.000	6.000
42.C00	4.00C	9.000	7.000	6.000
7.000	6.000	9.000	9.000	6.000
74.000	3.000	10.000	8.000	6.000
148.000	9.000	18.000	12.000	6.000
112.000	6.000	18.000	14.000	6.000
5.C0C	14.000	12.000	8.000	6.000
25.C0C	6.000	9.000	8.000	6.000
59.000	1.000	35.000	25.000	5.000
117.C0C	17.000	12.000	8.000	5.000
123.000	20.000	9.000	6.000	5.000
9.000	3.000	34.000	26.000	5.000
111.000	12.000	12.000	9.000	5.000
122.000	16.000	8.000	6.000	5.000
143.C00	3.000	16.000	16.000	5.000
153.C00	1.000	48.000	42.000	4.000
133.000	10.000	13.000	9.000	4.000
26.C00	8.000	7.000	6.000	4.000

SAMPLE DATA SORTED BY HEIGHT

TABLE VIII (CONTINUED)

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.0	48.0	40.0	96.0	184272.0	0.5	100.0

FOLLOWING BOXES WERE STACKED

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
102.0	48.0	40.0	72.0	0.0	0.0	0.0	48.0	40.0	72.0
20.0	24.0	35.0	24.0	0.0	0.0	72.0	24.0	35.0	96.0
93.0	24.0	32.0	19.0	24.0	0.0	72.0	48.0	32.0	91.0
108.0	12.0	8.0	18.0	24.0	32.0	72.0	36.0	40.0	90.0
108.0	12.0	8.0	18.0	36.0	32.0	72.0	48.0	40.0	90.0
64.0	9.0	8.0	6.0	24.0	32.0	90.0	33.0	40.0	96.0
64.0	9.0	8.0	6.0	33.0	32.0	90.0	42.0	40.0	96.0
117.0	12.0	8.0	5.0	24.0	0.0	91.0	36.0	8.0	96.0
117.0	12.0	8.0	5.0	36.0	0.0	91.0	48.0	8.0	96.0
117.0	12.0	8.0	5.0	24.0	8.0	91.0	36.0	16.0	96.0
117.0	12.0	8.0	5.0	36.0	8.0	91.0	48.0	16.0	96.0
117.0	12.0	8.0	5.0	24.0	16.0	91.0	36.0	24.0	96.0
117.0	12.0	8.0	5.0	36.0	16.0	91.0	48.0	24.0	96.0
117.0	12.0	8.0	5.0	24.0	24.0	91.0	36.0	32.0	96.0
117.0	12.0	8.0	5.0	36.0	24.0	91.0	48.0	32.0	96.0
117.0	12.0	5.0	8.0	0.0	35.0	72.0	12.0	40.0	80.0
117.0	12.0	5.0	8.0	12.0	35.0	72.0	24.0	40.0	80.0
117.0	12.0	5.0	8.0	0.0	35.0	80.0	12.0	40.0	88.0
117.0	12.0	5.0	8.0	12.0	35.0	80.0	24.0	40.0	88.0
117.0	12.0	5.0	8.0	0.0	35.0	88.0	12.0	40.0	96.0
117.0	12.0	5.0	8.0	12.0	35.0	88.0	24.0	40.0	96.0
122.0	6.0	8.0	5.0	42.0	32.0	90.0	48.0	40.0	95.0

PALLET ONE CONFIGURATION (LOADING WITH STANDARD PALLETS)
TABLE IX

SUMMARY LINE FOR EACH FALET THAT WAS STACKED

PALLET NUMBER	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.00	48.00	40.00	96.00	184272.00	0.49	99.97
2.00	48.00	40.00	96.CC	173462.0C	1.33	94.11
3.00	43.00	36.00	96.00	137730.00	3.18	92.68
4.00	52.00	44.00	96.CC	206974.00	0.81	94.23
5.00	47.00	40.00	95.00	169816.0C	1.C7	94.09
6.00	48.00	42.00	96.00	184524.00	1.26	95.34
7.00	49.00	42.00	96.CC	172856.0C	5.32	87.49
8.00	48.00	42.00	96.00	179498.00	7.40	92.75
9.00	46.00	40.00	96.00	131466.00	15.16	74.43
10.00	65.00	21.00	96.00	117259.0C	1.58	89.48
11.00	65.00	21.00	96.00	120927.00	2.37	92.28
12.00	47.00	26.00	96.0C	91541.0C	6.64	78.03
13.00	57.00	11.00	96.00	43950.0C	4.55	73.02
14.00	57.00	11.00	96.00	46860.00	3.50	77.85
15.00	50.00	22.00	96.CC	79032.0C	6.74	74.84
16.00	60.00	10.00	96.00	24648.00	2.68	42.79
17.00	60.00	10.00	96.00	22500.00	3.83	39.06
18.00	60.00	10.00	96.00	34160.CC	2.81	59.31
19.00	60.00	10.00	96.00	40704.00	12.08	70.67
20.00	60.00	10.00	96.CC	30186.0C	7.99	52.41
21.00	48.00	42.00	96.00	144940.0C	147.87	74.89
22.00	40.00	44.00	96.00	85494.00	12.83	50.60
23.00	40.00	44.00	95.0C	91194.0C	34.02	53.97
24.00	40.00	44.00	96.00	121072.00	46.57	71.66
25.00	40.00	44.00	96.00	151516.00	9.72	89.68
26.00	40.00	44.00	96.00	131420.0C	14.63	77.78
27.00	40.00	44.00	96.00	118176.00	9.28	69.94
28.00	40.00	44.00	96.0C	76295.0C	10.60	45.16
29.00	40.00	44.00	96.00	150885.0C	14.71	89.30
30.00	40.00	44.00	91.00	80221.00	15.81	47.48
31.00	40.00	44.00	96.CC	146548.00	9.62	86.74
32.00	40.00	44.00	89.00	82400.00	2.20	48.77

SUMMARY OF ALL PALLETS LOADED (LCADING WITH STANCARD PALLETS)

TABLE X

CONTAINER NR	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
1.0	96.0	480.0	96.0	3461760.0	C.5	78.3

FOLLOWING BOXES WERE STACKED

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BCXW	Z+BOXH
1C.C	21.0	65.0	96.0	0.0	0.0	0.0	21.0	65.0	96.0
11.0	21.0	65.0	96.0	21.0	0.0	0.0	42.0	65.0	96.0
17.0	10.0	60.0	96.0	42.C	0.0	0.0	52.0	60.0	96.0
18.C	10.0	60.0	96.0	52.0	0.0	0.0	62.0	60.0	96.0
19.0	10.0	60.0	96.0	62.0	0.0	0.0	72.0	6C.0	96.0
20.0	10.0	60.0	96.0	72.C	0.0	0.0	82.0	60.0	96.0
16.C	10.0	60.0	96.0	82.0	0.0	0.0	92.0	60.0	96.0
14.0	11.0	57.0	96.0	0.0	65.0	0.0	11.0	122.0	96.0
13.0	11.0	57.0	96.0	11.0	65.0	0.0	22.0	122.0	96.0
4.0	44.0	52.0	96.0	22.0	65.0	0.0	66.0	117.0	96.0
15.C	22.0	50.0	96.0	66.0	60.0	0.0	88.0	11C.0	96.0
7.0	42.0	49.0	96.0	0.0	122.0	C.C	42.C	171.0	96.0
8.0	42.0	48.0	96.0	42.C	117.0	0.0	84.0	165.0	96.0
6.C	42.0	48.0	96.0	0.0	171.0	0.0	42.0	219.0	96.0
2.0	40.0	48.0	96.0	42.0	165.0	C.0	82.0	213.0	96.0
1.0	40.0	48.0	96.0	0.C	219.0	0.0	40.0	267.0	96.0

CONTAINER CONFIGURATION (LOADING WITH STANDARD PALLETS)

TABLE XI

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
21.0	42.0	48.0	96.0	40.0	219.0	0.0	82.0	267.0	96.0
5.0	40.0	47.0	96.0	0.0	267.0	0.0	40.0	314.0	96.0
12.0	26.0	47.0	96.0	40.0	267.0	0.0	66.0	314.0	96.0
9.0	40.0	46.0	96.0	0.0	314.0	0.0	40.0	360.0	96.0
22.0	40.0	44.0	96.0	40.0	314.0	0.0	80.0	358.0	96.0
23.0	40.0	44.0	96.0	0.0	360.0	0.0	40.0	404.0	96.0
25.0	40.0	44.0	96.0	40.0	358.0	0.0	80.0	402.0	96.0
26.0	40.0	44.0	96.0	0.0	404.0	0.0	40.0	448.0	96.0
27.0	40.0	44.0	96.0	40.0	402.0	0.0	80.0	446.0	96.0

CONTAINER NR	LENGTH	WIDTH	HEIGHT	VOLUME	TIME	% EFFICIENCY
2.0	96.0	480.0	96.0	1162368.0	0.0	26.3

FOLLOWING BOXES WERE STACKED

ID NR	LENGTH	WIDTH	HEIGTH	X	Y	Z	X+BOXL	Y+BOXW	Z+BOXH
28.0	40.0	44.0	96.0	0.0	0.0	0.0	40.0	44.0	96.0
29.0	40.0	44.0	96.0	40.0	0.0	0.0	80.0	44.0	96.0
24.0	40.0	44.0	96.0	0.0	44.0	0.0	40.0	88.0	96.0
30.0	40.0	44.0	96.0	40.0	44.0	0.0	80.0	88.0	96.0
32.0	40.0	44.0	96.0	0.0	88.0	0.0	40.0	132.0	96.0
31.0	40.0	44.0	96.0	40.0	88.0	0.0	80.0	132.0	96.0
3.0	36.0	43.0	96.0	0.0	132.0	0.0	36.0	175.0	96.0

CONTAINER CONFIGURATION (LOADING WITH STANDARD PALLETS)

TABLE XI (CONTINUED)

Method of Sort

	No Sort (0)	Height (1)	Length (2)	Width (3)	Area (4)	Volume (5)	Total Volume (6)	Random (7)	Nr Boxes (8)
No Sort (0)	74.21 75.75	x	x	x	x	x	0	x	0
Height (1)	x	78.74-80.28	x	x	x	x	x	x	x
Length (2)	x	x	84.88-86.42	0	0	0	x	x	x
Width (3)	x	x	0	84.54-86.08	0	0	x	x	x
Area (4)	x	x	0	0	86.45-87.99	x	x	x	x
Volume (5)	x	x	0	0	x	83.34-84.88	x	x	x
Total Vol. (6)	0	x	x	x	x	x	74.56 76.10	x	x
Random (7)	x	x	x	x	x	x	x	70.65-72.19	0
Nr Boxes (8)	0	x	x	x	x	x	x	0	72.37-73.91

Number of Turns

	0	1	2	3	4	5	6
0	77.69-79.44	0	0	0	0	x	x
1	0	77.16 78.90	0	0	0	x	x
2	0	0	77.42-79.16	0	0	x	x
3	0	0	0	79.07-80.81	0	0	0
4	0	0	0	0	79.19-80.93	0	0
5	x	x	x	0	0	80.29-82.03	0
6	x	x	x	0	0	0	80.51-82.26

RANGE TEST RESULTS AND MEANS CONFIDENCE INTERVALS LOADING WITHOUT STANDARD PALLETS

TABLE XII

Method of Sort

	No Sort (0)	Height (1)	Length (2)	Width (3)	Area (4)	Volume (5)	Total Volume (6)	Random (7)	Nr Boxes (8)
No Sort (0)	52.44-54.92	x	0	0	0	0	0	0	0
Height (1)	x	56.50-59.98	x	0	0	x	0	0	0
Length (2)	0	x	51.38-53.86	x	0	0	x	0	0
Width (3)	0	0	x	55.26-57.74	0	x	0	0	0
Area (4)	0	0	0	0	54.16-56.64	0	0	0	0
Volume (5)	0	x	0	x	0	51.80-54.28	x	0	0
Total Vol. (6)	0	0	x	0	0	x	55.30-57.78	0	0
Random (7)	0	0	0	0	0	0	0	53.87-56.36	0
Nr Boxes (8)	0	0	0	0	0	0	0	0	53.27-55.75

Number of Turns

	0	1	2	3	4	5	6
0	51.53-53.65	x	x	x	x	x	x
1	x	54.45-56.57	0	0	0	0	0
2	x	0	54.10-56.22	0	0	0	0
3	x	0	0	54.09-56.21	0	0	0
4	x	0	0	0	54.47-56.59	0	0
5	x	0	0	0	0	54.47-56.59	0
6	x	0	0	0	0	0	54.52-56.64

RANGE TEST RESULTS AND MEANS CONFIDENCE INTERVALS LOADING WITH STANDARD PALLETS

TABLE XIII

STUFFING ALGORITHM FCRTRAN COMPLTER PROGRAM

```
C          PROGRAM- STUFFING ALGORITHM
C          PROGRAMMER- N. B. NELSON
C          DATE WRITTEN-SUMMER 1979
C          MACHINE UPON WHICH RUN- IBM 360/60
C          METHOD OF RUNNING-COMPLIED AND WRITTEN TC DISK (LINK ECITEC)
C          DESCRIPTION OF INPUT VARIABLES
C            CARD NUMBER 1 PARAMETERS
C                NR OF OPTIMIZATION LOOPS
C            CARD NUMBER 2 PARAMETERS
C                MAX TURNS OF BOX ALLOWED
C            CARD NUMBER 3 PARAMETERS
C                SORT CONTROL DIRECTION (SEE INPUT SUBROUTINE)
C            CARD NUMBER 4 PARAMETERS
C                STACKING CONTROL PARAMETER
C                                 CONTINUE
C             CARD NUMBER 5 PARAMETERS
C                PLMIN-IF BOX EXCEEDS THIS LENGTH, BOX
C                ITSELF IS USED FOR PALLET BASE
C                PWMIN-SAME AS ABOVE FOR WIDTH
C                PL-STANDARD PALLET LENGTH
C                PW-STANDARD PALLET WIDTH
C                PH-MAXIMUM PALLET HEIGTH
C                CONL-CONTAINER LENGTH
C                CONW-CONTAINER WIDTH
C                CONH-CONTAINER HEIGTH
C                                 CONTINUE
C             CARD NUMBER 6 (ONWARDS) PARAMETERS
C                BOX ID NUMBER
C                NUMBER OF BOXES ON THIS LINE (SAME DIMENSIONS)
C                BOX LENGTH
C                BOX WIDTH
C                BOX HEIGTH
                                  CONTINUE
C          SUBROUTINES USED IN PROGRAM
C                INISH-INITIALIZE VARIABLES
C                INPUT-READ INPUT VARIABLES
C                PRT...-PRINTS ALL RESULTS (HAS ENTRY POINTS)
C                SORTIN-SORTS INPUTS OF BOX SIZES
C                NEWPAL-STARTS A NEW PALLET
C                LOAD-DOES THE FOLLOWING
C                      -SELECTS NEXT BOX
C                      -SELECTS NEXT ORIGIN
C                      -PALLITIZE THE BOX
```

```
C             -TURN THE BOX
C          STUFF-LOAD (STUFF) THE PALLETS IN THE CONTAINER
C          GETEFF-CALCULATES EFFICIENCY OF THE LOADING
C          RESETC- RESET ARRARY C TO SEQUENCE FRICR TC LAST SORT
C          SHUFL-SHUFFLES THE ORDER OF ARRAY C
C          VSORTP-SORTS INPUT DATA
C          GETIME-GETS ELAPSED TIME
C          ITIME-TIME OF DAY IN HUNDREDTHS OF SECOND
                            CONTINUE
C      ARRAYS USED IN THE PROGRAM
C          A-LIST THE ID OF BOX, THE ORIGIN OF BOX, AND
C              THE POSSIBLE STACKING POSITIONS CFF THE BOX
C          B-LOGICAL ARRAY WHICH POINTS TO POSSIBLE ORIGINS
C              WHICH ARE LISTED IN A
C          C-INPUT ARRAY-LISTS EACH LINE OF INFUT(BOXES)
C          E-PALLET ARRAY-STORES INFO ON PALLETS
                            CONTINUE
C      LOGICAL VRIABLES USED IN PROGRAM
C          FIRST-INDICATES IF THIS IS FIRST PALLET TO LOAD
C          ALLGON-ALL BOXES HAVE BEEN LOADED
C          OUTSIZ-AT LEAST ONE BOX IS OUTSIZE TO MIN PALLET
C              PARAMENTERS
C          TRIED-ALL BOXES HAVE BEEN TRIED (TO LCAD) AT  THIS
C              ORIGIN
C          TURNED-BOX WAS TURNED
C          STUFED-THE CONTAINER STUFFING OPERATION HAS BEGUN
                            CONTINUE
C      OTHER VARIABLES USED IN THE PRCGRAM
C          BOXL-BOX LENGTH
C          BOXW-BOX WIDTH
C          BOXH-BOX HEIGTH
C          CONL-CONTAINER LENGTH
C          CCNW-CONTAINER WIDTH
C          CONH-CONTAINER HEIGTH
C          IAP-POINTS TO CURRENT ROW OF A
C          IBPR-POINTS TO ROW OF B
C          IBPC-POINTS TO COLUMN OF B
C          IEP-POINTS TO ROW OF E
C          ICP-POINTS TO ROW OF C
C          MXTURN-MAXIMUM TURNS OF THE BOX WHICH IS
C              PERMITTED (5 ARE POSSIBLE)
                            CONTINUE
C          NL-NUMBER OF INPUT LINES (TO ARRAY C)
C          NRTURN-COUNTS THE TIMES A BOX HAS BEEN TURNED
C          NXORG-NEXT X ORIGIN TO TRY
C          NYORG-NEXT Y ORIGIN TO TRY
C          NZORG-NEXT Z ORIGIN TO TRY
C          PALL-PALLET DIMENSIONS (CURRENT)
```

```
C               PALW-PALLET WIDTH
C               PALH-PALLET HEIGHT
C               PL-'STANDARD' PALLET LENGTH
C               PW-'STANDARD' PALLET WIDTH
C               PH-'STANDARD' PALLET HEIGTH
C               PLMIN-INPUT PARAMENTER-MIN PALLET LENGTH
C               PWMIN-MIN PALLET WIDTH
C               RPH-ACCUMULATES TIME
C               TIME-ACCUMULATES TIME TO LOAD ALL PALLETS
C               VOLUME-ACCUMULATES VOLUME OF ALL BOXES LOADED
C                    ON EACH PALLET

C           MAIN PROGRAM

      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,
     1PRINT,PRELON,STACK,STUFED,GRAVTY
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXT
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),
     1HCP(300),HCS(300),STACK(500),SORVEC(500)
      OSEFF = 0.
      RPH = 0.
      XTPXT = 0.
C               SET NUMBER OF OPTIMIZATION LOOPS
C               FOR LOADING PALLETS INTO CONTAINER
C               (MINIMUM NUMBER MUST BE 1)
      NSLOOP = 1
C
C
C               INITIALIZE COMMON BLOCK
      CALL INISH
C               READ IN GENERAL PARAMETERS AND
C               SPECIFIC BOX SIZES
      CALL INPUT
C               PRINT INPUT DATA
      CALL PRTIN
C               SORT ARRAY C (BOXES TO LOAD)
C               AND PRINT AGAIN INPUT DATA (SORTED)
      CALL SORTIN
      CALL PRTINS
C               START AT NEW PALLET BUT FIRST PRINT
C               RESULTS OF LAST PALLET (IF THIS NOT BE FIRST)
  105 CONTINUE
      IF (.NOT.FIRST) CALL PRTPAL
```

```
      CALL NPALL                                                        T   0C340
C             SELECT NEXT BOX TO LOAD. IFF BOXES ARE                    T   00350
C             ALL GONE GONE GO PRINT STATS. IFF ALL                     T   00360
C             BOXES HAVE BEEN TRIED START A NEW PALLET.                 T   00370
  110 CONTINUE                                                          T   00380
      CALL LOAD (&105)                                                  T   00390
  125 CONTINUE                                                          T   00400
C             NO MORE BOXES ARE LEFT TO LOAD.                           T   00410
      IF (PRINT) CALL PRTPST                                            T   0C420
C             NOW STUFF CONTAINER BY CALLING STUFF                      T   00430
C             CALL PRTSTF ON FIRST TIME THROUGH                         T   00440
      IF (PRINT .AND. .NOT.STUFED) CALL PRTSTF                          T   00450
C             NOW TURN OFF PRINT CONTROL                                T   00460
C             UNTIL STUFFING OPTIMAZATION OF                            T   00470
C             STUFFING PALLETS INTO CONTAINER IS FINISH                 T   00480
      IF (.NOT. STUFED) CALL STUFF                                      T   00490
C             NOW OPTIMIZE LOADING OF PALLETS BY                        T   00500
C             MEANS OF IMPROVING SEARCH OF NSLOOP CIFFERENT SEQUENCES   T   00510
C             OF PALLET STUFFING                                        T   00520
      PRINT = .FALSE.                                                   T   00530
      IF (.NOT. STUFED) GO TO 149                                       T   00535
      NRSTF = NRSTF + 1                                                 T   00536
      OPSTUF = .TRUE.                                                   T   00537
      CALL GETEFF                                                       T   00538
      IF (SEFF .GE. OSEFF) GO TO 130                                    T   00539
      CALL RESTO                                                        T   00561
      GO TO 135                                                         T   00562
  130 CONTINUE                                                          T   00563
      OSEFF = SEFF                                                      T   00564
  135 CONTINUE                                                          T   00565
      IF (NRSTF .EQ. NSLOOP .AND. PRELON) GO TO 147                     T   00590
      IF (NRSTF .GT. NSLOOP) GO TO 150                                  T   00600
      GO TO 148                                                         T   00610
  147 CONTINUE                                                          T   00620
      PRINT = .TRUE.                                                    T   00630
      GO TO 149                                                         T   00640
  148 CONTINUE                                                          T   00650
      CALL SHUFL                                                        T   0C660
  149 CONTINUE                                                          T   00670
      CALL INISH2                                                       T   00680
      STUFED = .TRUE.                                                   T   00690
      OLTSIZ = .FALSE.                                                  T   00700
      MXTURN = 1                                                        T   00710
C             REPLACE SECONC COLUMN OF C WHICH WAS DEPLETED             T   0C720
C             IN LOADING SUBROUTINE.                                    T   00730
      DO 155  I=1,NL                                                    T   00740
      C(I,2) = 1                                                        T   00750
  155 CONTINUE                                                          T   00760
```

```
      GC TO 105                                                   T   00770
  150 CONTINUE                                                    T   00780
C               RESET COUNTER FOR OPTIMIZATION OF                 T   00790
C               STUFFING (LOADING PALLETS INTO CONTAINER          T   00800
      NRSTF = 0                                                   T   00810
C               NOW OPTIMIZE ENTIRE LOADING BY MEANS OF           T   00820
C               IMPROVING SEARCHES OF 'NRLOAD' DIFFERENT          T   00830
C               SFQLENCES OF INPUT BOX LINES.                     T   00840
      CALL GETIME(IET)                                            T   0C850
      CALL RESETC                                                 T   00860
      NRPLT = NRPLT + 1                                           T   00870
      OFSTUF = .FALSE.                                            T   00880
      STUFED = .FALSE.                                            T   00891
      IF (OSEFF .GE. TEFF) GO TO 152                              T   00892
      CALL RESTO                                                  T   00893
      GO TO 154                                                   T   00894
  152 CONTINUE                                                    T   00895
      TEFF = CSEFF                                                T   00896
  154 CONTINUE                                                    T   00897
      IF (NRPLT .GT. NRLOOP) GO TO 16C                            T   00900
      CALL SHUFL                                                  T   C0920
      XT = IET * .000026                                          T   00930
      CUMTIM = CUMTIM + XT                                        T   00940
      WRITE (30,180) NRPLT,TEFF, XT, CUMTIM                       T   00950
      CALL INISH2                                                 T   00960
      CALL SETIME                                                 T   00970
      GO TO 105                                                   T   00980
  160 CONTINUE                                                    T   00990
C               OPTIMIZATION IS COMPLETED SO LOOP                 T   01000
C               BACK THRU AND PRINT RESULTS AS THEY               T   01010
C               ARE REALIZED  (PROVIDED OPTIMIZATION WAS DONE.)   T   01020
      IF (PRELCN) GO TO 165                                       T   01030
      PRINT = .TRUE.                                              T   01050
      PRELON = .TRUE.                                             T   01060
      CALL INISH2                                                 T   01070
      RPH = CUMTIM                                                T   01080
      GO TO 105                                                   T   01090
  165 CONTINUE                                                    T   01100
      SEFF = TEFF                                                 T   01110
      CALL PRTFIN                                                 T   01120
      STOP                                                        T   01130
  180 FORMAT (I8,4F8.3)                                           T   01140
      ENC                                                         T   C1180
```

```
      SUBROUTINE INISH                                                 T   01190
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                            T   01200
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                 T   01210
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,          T   01220
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,                 T   01230
     2     NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P          T   01240
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME             T   01250
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VCLIN,NLHOLD,NRSTF,IMXTT   01260
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF                      T   01270
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),       T   00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                         T   00100
      CALL SETIME                                                      T   01300
      TEFF = 0.                                                        T   01310
      NRLOOP = 1                                                       T   01320
      PRINT = .FALSE.                                                  T   01330
      PRELON = .FALSE.                                                 T   01340
      NRPLT = 0                                                        T   01350
      NRSTF = 0                                                        T   01360
      CUMTIM = 0.                                                      T   01370
      VOLIN = 0.                                                       T   01380
      SEFF = 0.                                                        T   01390
      ISEED = 12345                                                    T   01400
      ENTRY INISH2                                                     T   01410
      TIME = 0.                                                        T   01420
      NXORG = 1                                                        T   01430
      NYORG = 1                                                        T   01440
      NZORG = 1                                                        T   01450
      STUFED = .FALSE.                                                 T   01460
      FIRST = .TRUE.                                                   T   01470
      OUTSIZ = .TRUE.                                                  T   01480
      NRTURN = 1                                                       T   01490
      NBOX = 1                                                         T   01500
      VOLUME = 0.                                                      T   01510
      TIME = 0.                                                        T   01520
      IAP = 1                                                          T   01530
      ICP = 1                                                          T   01540
      IEP = 0                                                          T   01550
      IOUTIN = 1                                                       T   01560
      RETURN                                                           T   01570
C&    DEBUG SUBCHK,SUBTRACE,INIT,UNIT(30),TRACE                        T   01580
      END                                                              T   01590
```

```
      SUBROUTINE INPUT                                                  T 01600
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                             T 01610
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                  T 01620
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,           T 01630
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBOX,NRPERM,NRTURN,                  T 01640
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P            T 01650
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME              T 01660
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXTT 01670
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELCN,IAPEFF                       T 01680
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),        T 00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                          T 00100
C           READ GENERAL PROBLEM PARAMENTERS                            T 01710
C                   READ CARD NR 1.                                     T 01720
C                           NR OPTIMIZE LOOPS                           T 01730
C                             (F10.0)                                   T 01740
      READ (5,315) XX                                                   T 01750
      NRLOOP = IFIX(XX)                                                 T 01760
C           IF NO OPTIMIZATION IS DESIRED,                              T 01770
C           PRINT RESULTS OF LOADING AS IT OCCURS                       T 01780
      IF (NRLOOP .LE. 0) PRINT = .TRUE.                                 T 01790
      IF (NRLOOP .LE. 0) PRELON = .TRUE.                                T 01800
C                   READ CARD NR 2                                      T 01810
C                           MAX TURNS OF BOX                            T 01820
C                             (I1)                                      T 01830
      READ (5,320) MXTURN                                               T 01840
      IMXT = MXTURN                                                     T 01850
C                   READ CARD NR 3                                      T 01860
C                           SORT CONTROL                                T 01870
C                             (I2)                                      T 01880
      READ (5,321) ISORT                                                T 01890
C                   READ CARD NR 4                                      T 01900
C                           STACKING CONTROL (T=NO OVERHANG)            T 01910
C                                            (F=OVERHANG)               T 01920
C                             (L1)                                      T 01930
      READ (5,322) GRAVTY                                               T 01940
C                   READ CARD NR 5                                      T 01950
C                           LENGTH,WIDTH WHICH IF EXCEED                T 01960
C                           CAUSES BOX TO BE PALLET;                    T 01970
C                           LENGTH,WIDTH,HEIGHT OF STANDARD PALLET .    T 01980
C                           LENGTH,WIDTH,HEIGHT OF CONTAINER.           T 01990
C                             (F10.0)                                   T 02000
      READ (5,315) PLMIN,PWMIN,PL,PW,PH,CONL,CONW,CONH                  T 02010
C                             .                                         T 02020
C           READ EACH BOX (ID,NR,L,W,H) (DSRN=1)                        T 02030
C                             .                                         T 02040
      NL = 0                                                            T 02050
  305 CONTINUE                                                          T 02060
C                             .                                         T 02070
```

```
      NL = NL + 1
      READ (1,315,END=370) ((C(NL,I),I=2,5),XX)
C              IF FLAG IS SET ON THIS LINE, DO NOT LET
C              THIS BOX BE USED AS A BASE FOR OTHER BOXES
C              (IE, THIS BOX IS A NON-LOAD BEARING BOX).
      STACK(NL) = .TRUE.
      IF (XX .EQ. 1.) STACK(NL) = .FALSE.
C              NOW SET LENGTH TO MAX (LENGTH,WIDTH)
      IF (C(NL,3) .GE. C(NL,4)) GO TO 307
      XX = C(NL,3)
      C(NL,3) = C(NL,4)
      C(NL,4) = XX
  307 CONTINUE
      C(NL,1) = NL
      DO 306  I=1,5
      CHOLD(NL,I) = C(NL,I)
  306 CONTINUE
      VOLIN = C(NL,2) * C(NL,3) * C(NL,4) * C(NL,5) + VOLIN
C
C              NOW ESTABLISH THE SORTING VECTOR (SORVEC)
C              AS DIRECTED BY INPUT VARIABLE, SORT, AS FOLLOWS:
C
C                  ISORT    SORT BY (LARGEST TO SMALLEST)
C                      0    (NO SORT)
C                      1    HEIGHT
C                      2    LENGTH
C                      3    WIDTH
C                      4    AREA (LXW)
C                      5    VOLUME(LXWXH)
C                      6    TOT VOL  (LXWXHXNR)
C                      7    RANDOM SORT
C                      8    NR (BOXES PER LINE)
C                      9    (INVERSE ORDER OF READ-IN)
C                     10    RANDOM SORT (WITH RANDOM SEED)
C
      IF (ISORT .EQ. 0) GO TO 305
      GO TO (325,330,335,340,345,350,355,360,365,355),ISORT
      GO TO 340
  325 CONTINUE
      SORVEC(NL) = C(NL,5)
      GO TO 305
  330 CONTINUE
      SORVEC(NL) = C(NL,3)
      GO TO 305
  335 CONTINUE
      SORVEC(NL) = C(NL,4)
      GO TO 305
  340 CONTINUE
```

```
      SCRVEC(NL) = C(NL,4) * C(NL,3)                                    T   02560
      GO TO 305                                                         T   02570
  345 CONTINUE                                                          T   02580
      SORVEC(NL) = C(NL,4)*C(NL,3)*C(NL,5)                              T   02590
      GO TO 305                                                         T   02600
  350 CONTINUE                                                          T   02610
      SCRVEC(NL) = C(NL,4)*C(NL,3)*C(NL,5)*C(NL,2)                      T   02620
      GO TO 305                                                         T   02630
  355 CONTINUE                                                          T   02640
C             CAUTION.  FOLLOWING RANDCM GENERATOR IS RATHER POOR       T   02650
C                       AND DESIGNED FOR IBM 360 HARDWARE CNLY.         T   02660
C                       HOWEVER, IT IS VERY FAST.                       T   02670
C             IF SORT BE SET TO RANDOM SEED                             T   02785
C             SET ISEED RANDOMLY BY USING CURRENT TIME                  T   02790
C             OF DAY IN HUNDREDTHS OF SECS.(ISEED SHOULD BE ODD INTEGER)T   02800
      IF (NL .LE. 1 .AND. ISORT .EQ. 10) ISEED=ITIME(XX)/2*2+1          T   02810
      ISEED = ISEED * 65539                                             T   02680
      IF (ISEED .LT. 0) ISEED = ISEED + 2147483647 + 1                  T   02690
      SORVEC(NL) = ISEED                                                T   02700
      GO TO 305                                                         T   02710
  360 CONTINUE                                                          T   02720
      SORVEC(NL) = C(NL,2)                                              T   02730
      GC TO 305                                                         T   02740
  365 CONTINUE                                                          T   02750
      SORVEC(NL) = VL                                                   T   02760
      GC TO 305                                                         T   02770
  370 CONTINUE                                                          T   02830
      NL = NL - 1                                                       T   02840
C             RESET ISEED TO RANDOM NUMBER FOR USE IN REST OF PROGRAM   T   02850
      ISEED = ITIME(XX) / 2 * 2 + 1                                     T   02855
      RETURN                                                            T   02880
C&       DEBUG SUBCHK,SUBTRACE,INIT,UNIT(30),TRACE                      T   02890
C&    AT 305                                                            T   02900
C&    TRACE ON                                                          T   02910
C                                                                       T   02920
C                                                                       T   02930
C                                                                       T   02940
  315 FORMAT (8F10.0)                                                   T   02950
  320 FORMAT (I1)                                                       T   02960
  321 FORMAT (I2)                                                       T   02970
  322 FORMAT (L1)                                                       T   02980
      END                                                               T   02990
```

```
      SUBROUTINE SORTIN
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,
     1PRINT,PRELON,STACK,STUFED,GRAVTY
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBOX,NRPERM,NRTURN,
     2     NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXT
     5,TEFF,NRPLT,SEFF,CLMTIM,ISEED,PRELCN,IAPEFF
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),
     1HCP(300),HCS(300),STACK(500),SORVEC(500)
      DIMENSION CS(500,5)
      DIMENSION IPER(500)
C           IF SORT NOT DESIRED RETURN TO MAIN
      IF (ISORT .EQ. 0) RETURN
C           SET UP WORKING SORT ARRAY AND
C           INDEX ARRAY
      KEND = NL+1
      IENDSO = IFIX(FLOAT(NL) / 2. + .6)
C
C
C
      DO 410 I=1,NL
      K = KEND-I
C
C
C
      DO 405 J=1,5
      CS(I,J) = C(K,J)
  405 CONTINUE
C
C
C
      IPER(I) = I
      IF (I .GT. IENDSO) GO TO 410
      XX = SORVEC(I)
      SORVEC(I) = SORVEC(K)
      SORVEC(K) = XX
  410 CONTINUE
C
C
C
C           NOW SORT BY ORDER OF SORVEC
      CALL VSRTR (SORVEC,NL,IPER)
C           NOW REWRITE C WITH LARGEST VALUE FIRST
C
C
C
```

```
      DO 420 K=1,NL
      I = KEND-K
C
C
C
      DO 415 J=1,5
      XX = CS(IPER(I),J)
      C(K,J) = XX
      CHOLD(K,J) = XX
  415 CONTINUE
C
C
C
  420 CONTINUE
C
C
C
      RETURN
C&             DEBUG SUBCHK,SUBTRACE,INIT(C,K),UNIT(30),TRACE
C&    AT 405
C&    TRACE ON
      END
```

```
      SUBROUTINE NPALL
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,
     1PRINT,PRELON,STACK,STUFED,GRAVTY
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P
     3RINT,PLMIN,FWMIN,FPH,STUFED,TIME,TRIED,TURNED,VOLUME
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLFCLD,NRSTF,IMXT
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),
     1HCP(300),HCS(300),STACK(500),SORVEC(500)
  705 CONTINUE
      FIRST = .FALSE.
      VOLUME = 0.
      PALH = 0.
C             RESET POINTER AND LOGICAL USED IN GETBOX
      ICP = 1
C             SET A ARRAY POINTER
C             BUT FIRST RECORD ITS VALUE FOR USE IN GETEFF S.R.
      IAPEFF = IAP
      IAP = 0
      B(1,1) = .TRUE.
C             IF NO BOXES ARE OUTSIZE USE PREVIOUS DEFAULT VALUES
      IF (.NOT.OUTSIZ) GO TO 720
C             CHECK TO SEE IF BOX LARGER THAN
C             MAX ALLOWABLE SIZE EXISTS
C             IF SO, USE FOR PALLET BASE
C
C
C
      DO 710 I=IOUTIN,NL
      IF (C(I,2).GT.0..AND.(C(I,3).GT.PLMIN.OR.C(I,4).GT.PWMIN)) GO TO 7
     115
  710 CONTINUE
C
C
C
      PALL = PL
      PALW = PW
      OUTSIZ = .FALSE.
      GO TO 720
  715 CONTINUE
      IOUTIN = I
C             FOUND AN OUTSIZED BOX. LOAD THIS BOX
C             AND ADJUST PALLET DIMENSIONS.
C             LOAD ID INTO ARRAY A
      A(1,1) = C(1,1)
C             LOAD X,Y,Z INTO A
```

```
      A(1,2) = 0.
      A(1,3) = 0.
      A(1,4) = 0.
C              LOAD L,W,H INTO A
      A(1,5) = C(I,3)
      A(1,6) = C(I,4)
      A(1,7) = C(I,5)
C              UPDATE B ARRAY
      B(1,3) = .TRUE.
C              SET PALET DIMENSIONS
      PALL = C(I,3)
      PALW = C(I,4)
C              POSITION A POINTER
      IAP = 1
C              NOW SUBTRACT THIS BOX FROM C
      C(I,2) = C(I,2)-1
      VOLUME = PALL*PALW*C(I,5)
  720 CONTINUE
C              SELECT SMALLEST PERMISSIBLE MARGINS
C              IN X,Y, AND Z DIRECTIONS
      SMLX = 2.E6
      SMLY = 2.E6
      SMLZ = 2.E6
C
C
C
      DO 725 I=1,NL
      IF (C(I,2).LE.0) GO TO 725
      IF (C(I,3).LT.SMLX) SMLX=C(I,3)
      IF (C(I,4).LT.SMLY) SMLY=C(I,4)
      IF (C(I,5).LT.SMLZ) SMLZ=C(I,5)
  725 CONTINUE
C
C
C
      IF (MXTURN.LE.0.) GO TO 730
      SMLX = AMIN1(SMLX,SMLY)
      SMLY = SMLX
      IF (MXTURN.EQ.1) GO TO 730
      SMLX = AMIN1(SMLX,SMLZ)
      SMLY = SMLX
      SMLZ = SMLX
  730 CONTINUE
      RETURN
C&    DEBUG SUBTRACE,TRACE,SUBCHK,INIT,UNIT(30)
C&    AT 710
C&    TRACE ON
      END
```

```
      SUBROUTINE LOAD(*)
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,
     1PRINT,PRELON,STACK,STUFED,GRAVTY
      LOGICAL CHANGE, SOMCHG, CHECKD
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,
     1 IAP, ICP, IEP,MXTURN,OPSTUF,NL,NBOX,NRPERM,NRTURN,
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P
     3R INT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXT
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF
      COMMON A(1000,7),B(1000,3),C(50C,5),CHOLD(5C0,5),E(300,6),
     1HCP(300),HCS(300),STACK(500),SORVEC(500)
C             .
C             AREA FOR SELECTION OF BOX
C             .
  500 CONTINUE
C             RESET NRTURN USED IN TURN
C             RESET POINTER TO NEW ORIGIN
      IF (OUTSIZ) GO TO 505
      NXCRG = 1
      NYORG = 1
      NZORG = 1
      GC TO 510
  505 CONTINUE
      NXORG = 2
      NYORG = 2
      NZORG = 1
  510 CONTINUE
      NRTURN = 1
C             SELECT THE NEXT BOX WHICH IS NOT
C             OUTSIZED TO PALLET IN BOTH DIMENSION
      IF (ICP.EQ.1) ALLGON=.TRUE.
C
C
C
      DO 515 I=ICP,NL
      IF (C(I,2).LE.0.) GO TO 515
      IF (C(I,3).LE.PALL.OR.C(I,4).LE.PALW) GO TO 525
      IF (ICP.EQ.1) ALLGON=.FALSE.
  515 CONTINUE
C
C
C
  520 CONTINUE
C             IF ALL BOXES ARE GONE (LOADED) DROP
C             THRU TO PRINT SUBROUTINE
      IF (ALLGON .AND. .NOT. OUTSIZ) RETURN
C             ALL BOXES HAVE BEEN TRIED AT THIS PALLET
```

```
C               THEREFORE, GO START ANOTHER PALLET
      RETURN 1
  525 CONTINUE
C               BOX HAS BEEN FOUND.  RECORD LINE
C               NUMBER,L,W,H CF BOX AS TAKEN
C               FROM ARRAY C.
      IF (ICP.EQ.1) ALLGON=.FALSE.
      ICP = I
      BOXL = C(I,3)
      BOXW = C(I,4)
      BCXH = C(I,5)
C               END OF AREA TO GET BOX
C                                 .
C                                 .
C               AREA TO DETERMINE ORIGIN
C                                 .
C               IF A STANDARD PALLET IS BEING USED
C               SELECT THE CORNER (0,0,0) AS
C               THE ORIGIN AND GO DIRECT TO PALLETIZE
  600 CONTINUE
      IF (IAP.GE.1) GO TO 605
      ORX = 0.
      ORY = 0.
      ORZ = 0.
      IBPR = 1
      IBPC = 1
      GO TO 800
  605 CONTINUE
      IF (NXORG.GT.IAP) GO TO 615
C               TRY FOR X POSITION FIRST
C
C
C
      DO 610 IBPR=NXORG,IAP
      IF (.NOT.B(IBPR,1)) GO TO 610
      IF ((A(IBPR,5)+BOXL).GT.PALL) GO TO 610
      IF ((A(IBPR,3)+BOXW).GT.PALW) GO TO 610
      IF ((A(IBPR,4)+BOXH).GT.PH) GO TO 610
      GO TO 640
  610 CONTINUE
C
C
C
      NXORG = IAP+1
  615 CONTINUE
C               SINCE NO MORE X POSTIONS TRY FOR Y POSITION
      IF (NYORG.GT.IAP) GO TO 625
C
```

```
C
C
      DO 620  IBPR=NYORG,IAP
      IF (.NOT.B(IBPR,2)) GO TO 620
      IF ((A(IBPR,6)+BOXW).GT.PALW) GO TO 620
      IF ((A(IBPR,2)+BOXL).GT.PALL) GO TO 620
      IF ((A(IBPR,4)+BOXH).GT.PH) GO TO 620
      GO TO 645
  620 CONTINUE
C
C
C
      NYORG = IAP+1
  625 CONTINUE
C             SINCE NO Y POSITIONS, TRY FOR Z
C             PROVIDED  STUFFING OPERATION IS NOT
C             GOING ON  (SINCE PROGRAM ARBATARILY DECIDED
C             NOT  TO STACK PALLETS).
      IF (.NOT. STUFED .AND. NZORG.LE.IAP) GO TO 630
C             THERE ARE NO ORIGINS AVAIL
C             SO GO TURN THE BOX
      GO TO 900
  630 CONTINUE
C
C
C
      DO 635 IBPR=NZORG,IAP
      IF (.NOT.B( IBPR,3)) GO TO 635
      IF ((A(IBPR,7)+BOXH).GT.PH) GO TO 635
      IF ((A( IBPR,2)+BOXL).GT.PALL) GO TO 635
      IF ((A(IBPR,3)+BOXW).GT.PALW) GO TO 635
      GO TO 650
  635 CONTINUE
C
C
C
C             THERE ARE NO ORIGINS AVAIL
C             SO GO TURN THE BOX
      GO TO 900
  645 CONTINUE
      NYORG = IBPR+1
      ORX = A(IBPR,2)
      ORY = A(IBPR,6)
      ORZ = A(IBPR,4)
      IBPC = 2
C             FOUND AN ORIGIN NOW GO LOAD BOX
      GO TO 800
  650 CONTINUE
```

```
      NZCRG = IBPR+1
      ORX = A(IBPR,2)
      ORY = A(IBPR,3)
      ORZ = A(IBPR,7)
      IBPC = 3
C              FOUND AN ORIGIN NOW LOAD BOX
      GC TO 800
  640 CONTINUE
      NXORG = IBPR+1
      ORX = A(IBPR,5)
      ORY = A(IBPR,3)
      ORZ = A(IBPR,4)
      IBPC = 1
C              FOUND AN ORIGIN NOW LOAD BOX
C              END OF AREA TO GET ORIGIN
C                                  .
C                                  .
C              AREA TO TRY TO PALLETIZE BOX
C                                  .
  800 CONTINUE
C              SEE IF BOX WILL FIT
      ORXPL = ORX+BOXL
      ORYPW = ORY+BOXW
      ORZPH = CRZ+BOXH
C              IF THIS IS FIRST BOX AND SINCE IT MUST
C              FIT THE PALLET, LOAD IT NOW
      IF (IAP.LE.0) GO TO 835
C              IF BOX WILL NOT FIT GO GET ANOTHER CRIGIN
C              IMPROVE THE DENSITY IF POSSIBLE
C              DO THIS BY FINDING A BOX THAT WILL
C              INITIALLY FIT AND THEN PROGRESSIVELY MOVE
C              THE BOX TO THE LEFT,DOWN, AND TOWARD THE
C              THE FRONT, IF POSSIBLE.  THAT IS,
C              MOVE THE BOX TOWARD THE FIXED ORIGIN.  REPEAT
C              UNTIL NO FUTHER IMPROVEMENT IS POSSIBLE IN
C              ANY OF THE 3 DIRECTIONS.
      SCMCHG = .FALSE.
      CHECKD = .FALSE.
  700 CONTINUE
       CHANGE = .FALSE.
C              FIND MOST RESTRICTING BOX IN LOWER Y DIRECTION
C              IN ORDER TO SEE IF BOX WILL FIT AND TC
C              SEEK AN IMPROVEMENT. (HOWEVER, NO IMPROVEMENT
C              IS POSSIBLE IF ORY IS ALREADY AT
C              ITS MIN (0) OR ORIGIN IS AN 'Y' ORIGIN (IBPC=2).
      IF (.NOT. SCMCHG .AND. IBPC .EQ. 2) GO TO 711
      IF (ORY .EQ. 0) GO TO 711
      SLACK = ORY
```

```
      CHECKD = .TRUE.
      DO 710 I=1,IAP
      IF (  A(I,2) .GE. ORXPL .OR. A(I,5) .LE. ORX .OR.
     .      A(I,4) .GE. ORZPH .OR. A(I,7) .LE. ORZ .OR.
     .      A(I,3) .GE. ORYPW) GO TO 710
      IF (  A(I,6) .LE. ORY ) GO TO 705
C           GO GET ANOTHER ORIGIN
      GO TO 605
  705 CONTINUE
      XYZ = ORY - A(I,6)
      IF (SLACK .GT. XYZ) SLACK = XYZ
  710 CONTINUE
      IF (SLACK .LT. 1.) GO TO 711
      CHANGE = .TRUE.
      SOMCHG = .TRUE.
      ORY = ORY - SLACK
      ORYPW = ORYPW - SLACK
  711 CONTINUE
C           FIND MOST RESTRICTING BOX IN DOWN Z DIRECTION
C           SIMILAR TO ABOVE SEARCH FOR Y DIRECTION
      IF (ORZ .EQ. 0) GO TO 721
      IF (.NOT. SOMCHG .AND. IBPC .EQ. 3) GO TO 721
      SLACK = ORZ
      CHECKD = .TRUE.
      DO 720 I=1,IAP
      IF (A(I,2) .GE. ORXPL .OR. A(I,5) .LE. ORX .OR.
     .    A(I,3) .GE. ORYPW .OR. A(I,6) .LE. ORY .OR.
     .    A(I,4) .GE. ORZPH) GO TO 720
      IF (A(I,7) .LE. ORZ) GO TO 715
      GO TO 605
  715 CONTINUE
      XYZ = ORZ - A(I,7)
      IF (SLACK .GT. XYZ) SLACK = XYZ
  720 CONTINUE
      IF (SLACK .LT. 1.) GO TO 721
      CHANGE = .TRUE.
      SOMCHG = .TRUE.
      ORZ = ORZ - SLACK
      ORZPH = ORZPH - SLACK
  721 CONTINUE
C           C FIND MOST RESTRICTING BOX IS LEFT X
C           DIRECTION SIMILAR TO ABOVE TWO SEARCHES
C           HOWEVER, MAKE SURE THE LOCATION IS CHECKED
C           AT LEAST ONCE.
      IF (.NOT. CHECKD) GO TO 722
      IF (ORX .EQ. 0.) GO TO 736
      IF (.NOT. SOMCHG .AND. IBPC .EQ. 1) GO TO 736
  722 CONTINUE
```

```
      SLACK = ORX
      CHECKD = .TRUE.
      DO 730 I=1,IAP
      IF (A(I,4) .GE. ORZPH .OR. A(I,7) .LE. ORZ  .OR.
     .     A(I,3) .GE. ORYPW .OR. A(I,6) .LE. ORY  .OR.
     .     A(I,2) .GE. ORXPL) GO TO 730
      IF (A(I,5) .LE. ORX) GO TO 725
      GO TO 605
  725 CONTINUE
      XYZ = ORX - A(I,5)
      IF (SLACK .GT. XYZ) SLACK = XYZ
  730 CONTINUE
      IF (SLACK .LT. 1) GO TO 736
      CHANGE = .TRUE.
      SCMCHG = .TRUE.
      ORX = ORX -SLACK
      ORXPL = ORXPL - SLACK
  736 CONTINUE
C             NOW IF ANY CHANGE HAS OCCURED DURING THE
C             ABOVE SEARCHES, LOOP BACK AND TRY TO FIND
C             MORE IMPROVEMENT IN OTHER DIRECTIONS.
      IF (CHANGE) GO TO 700
C
C
C
C             IF GRAVITY IS A PROBLEM, SEE IF BOX
C             WILL HAVE A BASE UPON WHICH TO REST
C             CHECK EACH BOX AT HEIGTH 'ORZ' AT
C             EACH OF THE 4 CORNERS OF BOX:
C             (BUT DO NOT USE A NON-LOAD BEARING BOX)
C
C                LOOP    POINT
C                  1     (ORX,ORY)
C                  2     (ORX + BOXL,ORY)
C                  3     (ORX,ORY + BOXW)
C                  4     (ORX + BOXL,ORY + BOXW)
C
      IF (.NOT.GRAVTY.OR.ORZ.EQ.0..OR.OUTSIZ.AND.ORZ.EQ.A(1,7)) GO TO 83
     15
      XX = ORX
      YY = ORY
C
C
C
      DO 830 J=1,4
C
C
C
```

AD-A078 274 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 13/4

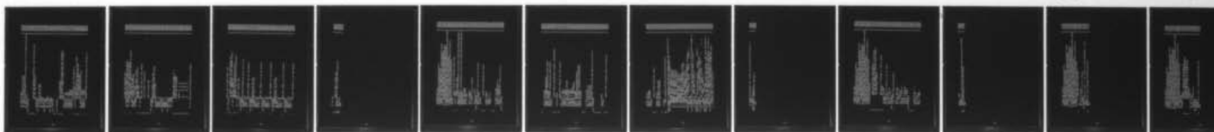
A CONTAINER STUFFING ALGORITHM FOR RECTANGULAR SOLIDS WHEN VOID--ETC(U)

SEP 79 N B NELSON

UNCLASSIFIED NL

2 OF 2

AD
A078274



END

DATE FILMED

1 -80

DDC

```
      DO 810 I=1,IAP
      IF (.NOT. STACK(I)) GO TO 810
      IF (A(I,7).NE.ORZ) GO TO 810
      IF (A(I,2).LE.XX.AND.A(I,5).GE.XX.AND.A(I,3).LE.YY.AND.A(I,6).GE.Y
     1Y) GO TO C 815
  810 CONTINUE
C
C
C
C                 BOX HAS NO SUPPORT, THEREFORE, CAN NOT PALLETIZE,
C                 GC GET ANOTHER ORIGIN
      GO TO 605
  815 CONTINUE
      IF (J.LE.1) GO TO 825
      IF (J.LE.2) GO TO 820
      XX = ORXPL
      GO TO 830
  820 CONTINUE
      XX = ORX
      YY = ORYPW
      GO TO 830
  825 CONTINUE
      XX = ORXPL
  830 CONTINUE
C
C
  835 CONTINUE
C
C                 BOX IS SUPPORTED
C                 THIS ORIGIN (ORX,ORY,ORZ) WILL NOT RESTRICT
C                 THE BOX'S LOADING. RECORD THE BOX IN ARRAY A
      IAP = IAP+1
      A(IAP,2) = ORX
      A(IAP,3) = CRY
      A(IAP,4) = ORZ
      A(IAP,5) = CRXPL
      A(IAP,6) = CRYPW
      A(IAP,7) = ORZPH
C                 NOW SUBTRACT BOX FROM ARRAY C
      C(ICP,2) = C(ICP,2)-1
C                 NOW UPDATE LOGICAL ARRAY B (THIS POINT
C                 IS NOW FILLED) AND ESTABLISH THE
C                 3 NEW POSSIBLE ORIGINS
      IF (IAP.GT.1) B(IBPR,IBPC)=.FALSE.
      B(IAP,1) = .TRUE.
      B(IAP,2) = .TRUE.
      B(IAP,3) = .TRUE.
C                 OVERIDE THESE TRUE SETTTINGS OF ORIGINS IF
```

```
C             NO BOX CAN POSSIBLY FIT
      IF ((PALL-ORXPL).LT.SMLX) B(IAP,1)=.FALSE.
      IF ((PALW-ORYPW).LT.SMLY) B(IAP,2)=.FALSE.
      IF ((PH-ORZPH).LT.SMLZ) B(IAP,3)=.FALSE.
C             RECORD HEIGHT, CUMULATIVE VOLUME OF PALLET
      IF (.NOT. PRINT) GO TO 500
      VOLUME = VOLUME+BCXL*BOXW*BOXH
      A(IAP,1) = C(ICP,1)
      IF (ORZPH .GT. PALH) PALH = ORZPH
C             BOX WAS PALLETIZED SO GO GET
C             ANOTHER BOX
      GC TO 500
C             BOX WILL NOT FIT AT THIS CRIGIN
C                                  .
C             END OF AREA TO PALLETIZE
C                                  .
C                                  .
C             AREA TO TURN BOX
C                                  .
  900 CONTINUE
      IF (NRTURN.GT.MXTURN) GO TO 915
C             RESET POINTERS USED IN GETORG
      IF (OUTSIZ) GO TO 905
      NXORG = 1
      NYORG = 1
      NZORG = 1
      GO TO 910
  905 CONTINUE
      NXORG = 2
      NYORG = 2
      NZORG = 1
  910 CONTINUE
      TL = BOXL
      TW = BOXW
      TH = BOXH
C             INCREASE NR OF TURNS BY 1.
C             DO NOT EXCEED MXTURN
C             TURNS ARE AS FOLLOWS
C                                  .
C             NRTURN  X   Y   Z
C               -     L   W   H
C               1     W   L   H
C               2     W   H   L
C               3     L   H   W
C               4     H   L   W
C               5     H   W   L
C                                  .
      GO TO (920,925,930,935,940,915), NRTURN
```

```
C              THERE ARE NC MORE TURNS LEFT
  915 CONTINUE
C              IF ICP=NL THERE ARE NO MORE BOXES LET
C              TO TRY.   THEREFORE, GO GET ANOTHER PALLET.
      IF (ICP .EQ. NL) RETURN 1
C              INCFEASE ICP IN ORDER TO SELECT
C              THE NEXT LINE WHEN GETBOX IS REENTEFED.
      ICP = ICP+1
C              GO GET ANCTHER BOX
      GO TO 500
  920 CONTINUE
      IF (BOXL.GT.PALW.CR.BOXW.GT.PALL) GO TO 925
      NRTURN = 2
      BOXL = TW
      BOXW = TL
C              BOX WAS TURNED. GO GET AN ORIGIN
      GC TO 605
  925 CONTINUE
      IF (MXTURN.LE.2) GO TO 915
      IF (BOXH.GT.PALW) GO TO 935
      NRTURN = 3
      BOXL = TW
      BCXW = TH
      BOXH = TL
C              BOX WAS TURNED. GO GET AN ORIGIN
      GO TO 605
  930 CONTINUE
      NRTURN = 4
      BOXL = TL
      BOXW = TH
      BCXH = TW
C              BOX WAS TURNED. GO GET AN ORIGIN
      GO TO 605
  935 CCNTINUE
      IF (MXTURN.LE.4) GO TO 915
      IF (BOXH.GT.PALL) GO TO 915
      NRTURN = 5
      BOXL = TH
      BCXW = TL
      BCXH = TW
C              BOX WAS TURNED. GO GET AN CRIGIN
      GO TO 605
  940 CCNTINUE
      NRTURN = 6
      BCXL = TH
      BCXW = TW
      BOXH = TL
C              BOX WAS TURNED. GO GET AN ORIGIN
```

```
      GO TO 605                                                         T  09050
C                                                                       T  09060
C                 END CF AREA TO TURN BOX                               T  09070
C&    DEBUG SUBTRACE,TRACE,SUBCHK,INIT,UNIT(30)                         T  09090
C&    AT 500                                                            T  09100
C&    TRACE ON                                                          T  09110
      END                                                               T  09120
```

```
      SUBROUTINE PRT                                                    T   09130
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                             T   09140
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                  T   09150
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,           T   09160
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,                  T   09170
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P            T   09180
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME              T   09190
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLFOLD,NRSTF,IMXTT   09200
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF                       T   09210
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),        T   00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                          T   00100
      RETURN                                                            T   09240
C           ENTRY POINT FOR PRINTING INPUT DATA                         T   09250
      ENTRY PRTIN                                                       T   09260
C           PRINT INPUT DATA                                            T   09270
      WRITE (6,1025) PLMIN,PWMIN,PL,PW,PH,CONL,CONW,CONH,MXTURN,NRLOOP,IT   09280
     1SORT,GRAVTY                                                       T   09290
      WRITE(20,1025) PLMIN,PWMIN,PL,PW,PH,CONL,CONW,CONH,MXTURN,NRLOOP,IT   09300
     1SORT,GRAVTY                                                       T   09310
      WRITE (20,1020)                                                   T   09320
      WRITE (20,1045)                                                   T   09330
      GO TO 1005                                                        T   09340
C                                                                       T   09350
      ENTRY PRTINS                                                      T   09360
      CALL SETIME                                                       T   09370
      IF (ISORT .EQ. 0) RETURN                                          T   09380
C                                                                       T   09390
      WRITE (6,1075)                                                    T   09400
 1005 CONTINUE                                                          T   09410
      WRITE (6,1030) ((C(I,J),J=1,5),I=1,NL)                            T   09420
      CALL SETIME                                                       T   09430
      RETURN                                                            T   09440
C                                                                       T   09450
C           ENTRY POINT FOR PALLET STATS                                T   09460
C                                                                       T   09470
      ENTRY PRTPST                                                      T   09480
      PALVOL = 0.                                                       T   09490
      TIME = 0.                                                         T   9510
      VOLUME = 0.                                                       T   09500
C                                                                       T   09510
C                                                                       T   09520
C                                                                       T   09530
      DO 1010 I=1,IEP                                                   T   09540
      VOLUME = VOLUME+E(I,5)                                            T   09550
      TIME = TIME+E(I,6)                                                T   09560
      PALVOL = PALVOL+E(I,2)*E(I,3)*E(I,4)                              T   09570
 1010 CONTINUE                                                          T   09580
C                                                                       T   09590
```

```
C
C
      RFH = RPH+TIME
      TOTEFF = VOLUME/PALVOL*100.
      WRITE (20,1035) VOLUME,PALVOL,TIME,TOTEFF
      WRITE (6,1040)
      WRITE (6,1035) VOLUME,PALVOL,TIME,TOTEFF
      RETURN
C
C             ENTRY POINT FOR DETAILEC PALLET INFCRMATION
C
      ENTRY PRTPAL
C
C             NOW STORE RESULTS OF LAST PALLET
      IEP = IEP + 1
      E(IEP,2) = PALL
      E(IEP,3) = FALW
      FIRST = .FALSE.
      IF (.NOT. PRINT) RETURN
      CALL GETIME(IET)
      E(IEP,1) = IEP
      E(IEP,4) = PALH
      E(IEP,5) = VOLUME
      E(IEP,6) = IET*.000026
      WRITE (6,1040)
      EFF = VCLUME/(PALL*PALW*PH)*100.
      WRITE (6,1045) ((E(IEP,J),J=1,6),EFF)
      WRITE (20,1070) ((E(IEP,J),J=1,6),EFF)
      WRITE (6,1050)
C
C
C
      DO 1015 I=1,IAP
      BLL = A(I,5)-A(I,2)
      BWW = A(I,6)-A(I,3)
      BHH = A(I,7)-A(I,4)
      WRITE (6,1055) (A(I,1),BLL,BWW,BHH,(A(I,J),J=2,7))
 1015 CONTINUE
C
C
C
C
C
      CALL SETIME
      RETURN
C
C             ENTRY POINT TO PRINT START OF STUFFING
C
```

```
      ENTRY PRTSTF
      WRITE (6,1060)
      WRITE (20,1060)
      RETURN
C
C            ENTRY POINT FOR FINAL STATS
C
      ENTRY PRTFIN
      WRITE (6,1065) RPH,SEFF
      WRITE (20,1065) RPH,SEFF
C
C
      RETURN
C&        DEBUG SUBCHK,SUBTRACE,INIT,UNIT(30),TRACE
C
 1020 FORMAT (' SUMMARY LINE FOR EACH PALET THAT WAS STACKED'//////)
 1025 FORMAT ('1FOLLOWING ARE INPUT PARAMENTERS:'///
     1 ' MINIMUM PALLET LENGTH=',F10.3//
     2 ' MINIMUM PALLET WIDTH=',F10.3//
     3 ' NORMAL PALLET LENGTH=',F10.3//
     4 ' NORMAL PALLET WIDTH=',F10.3//
     5 ' NORMAL PALLET HEIGTH=',F10.3//
     6 ' CONTAINER LENGTH =',F10.3//
     7 ' CONTAINER WIDTH =',F10.3//
     8 ' CONTAINER HEIGHT =',F10.3//
     9 ' MAX NR OFTURNS (6 POSSIBLE)=',I10//
     $ ' NUMBER OF OPTIMIZATION LOOPS= ',I10//
     $ ' SORT CONTROL VARIABLE=',I10//
     $ ' LOGICAL GRAVITY VARIABLE=',L10/'1')
 1030 FORMAT (' FOLLOWING IS AN ECHO PRINT OF BOXES'//
     1' NR LINE     NR BOXES  LENGTH     WIDTH      HEIGHT'//
     2 (5F10.3))
 1035 FORMAT (////' TOTAL BOX VOLUME=',F12.2//' TOTAL PALLET VOLUME =',
     1 F12.1//' TOTAL ELAPSED TIME (IN SECS)=',F12.2//
     2' TOTAL %  EFFICIENCY =',F12.2)
 1040 FORMAT ('1')
 1045 FORMAT (' PALLET NUMBER LENGTH      WIDTH      HEIGHT',
     1' VOLUME        TIME    % EFFICIENCY'// 8F10.1)
 1050 FORMAT (////' FOLLOWING BOXES WERE STACKED'//
     1'   ID  NR  LENGTH    WIDTH  HEIGTH      X         Y         Z  ',
     2' X+BOXL  Y+BOXW  Z+BOXH'//)
 1055 FORMAT (/,    10F8.1)
 1060 FORMAT(///'1 ALL BOXES HAVE BEEN LOADED.  THE BELOW PRINTOUT'/
     1' REFERS TO THE STUFFING OF THE VAN.'/' THUS, ',
     2'THE VAN MAY BE TAKEN AS THE PALLET AND THE PALLET THE BOX.')
 1065 FORMAT (//////'1 NORMAL TERMINATION OF PROGRAM.'/
     1 ' TOTAL ELAPSED TIME FOR BOTH PALLET LOADING AND VAN',
     2' STUFFING WAS (SECS):',F8.2///
```

```
     3' GRAND EFFICIENCY (%) (TOTAL BCX VOL/TOTAL CCNT VCL) =',
     4  F8.2)
 1070 FCRMAT (8F10.2/)
 1075 FORMAT ('1NOW AFTER SORTING:')
      ENC
```

```
      SUBROUTINE STUFF
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,
     1PRINT,PRELON,STACK,STUFED,GRAVTY
      COMMON ALLGON, BOXL, BOXW, BOX H,CONL,CONW,CONH,FIRST,HIGH,
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,
     2    NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXT
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),
     1HCP(300),HCS(300),STACK(500),SORVEC(500)
      DIMENSION IPER(500)
C
C              MOVE THE PALLETS INTO ARRAY C .
C              BUT DO NOT ALLOW STUFFING IN Z DIRECTION
C              (IE NOT NOT STACK) BY MAKING PALLET HEIGHT
C              THE CONTAINER HEIGHT.
C
C
C              NOW PRIOR TO LOADING C, SORT E ARRAY
C              BY WIDTH
      DO 1100 I=1,IEP
      IPER(I) = I
      IF (E(I,3) .GE. E(I,2) ) GO TO 1100
      XX =E(I,2)
      E(I,2) = E(I,3)
      E(I,3) = XX
 1100 CONTINUE
      CALL VSRTR (E(1,3),IEP,IPER)
C
      IEPP1 = IEP + 1
      DC 1105 K=1,IEP
      I = IEPP1 - K
      IX = IPER(I)
      C(K,1) = E(IX,1)
      C(K,2) = 1
      C(K,3) = E(IX,2)
      C(K,4) = E(I,3)
      C(K,5) = CONH
 1105 CONTINUE
C
C
C
      C((IEP+1),2) = 0
      NLHOLD = NL
      NL = IEP
      PALH = CONH
      PALL = CONL
```

```
      PALW = CONW
      RETURN
C&    DEBUG SUBCHK,SUBTRACE,INIT(NLHOLD,NL,PALH,PALL,PALW,C,STUFED,
C&   .OUTSIZ,MXTUFN,IX), UNIT (30), TRACE
      END
```

```
      SUBROUTINE GETEFF                                                 T  11110
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                             T  11120
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                  T  11130
      COMMON ALLGON, BOXL, BOXW, BOXH,CONL,CONW,CONH,FIRST,HIGH,        T  11140
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,                  T  11150
     2      NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P          T  11160
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME              T  11170
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXTT  11180
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF                       T  11190
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),        T  00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                          T  00100
C           FIND MAX WIDTH UTILIZED                                     T  11220
      XW = 0.                                                           T  11230
      DO 10 I=1,IAPEFF                                                  T  11240
      IF (A(I,6) .GT. XW) XW = A(I,6)                                   T  11250
   10 CONTINUE                                                          T  11260
      SEFF = VOLIN /((IEP-1 + XW /CONW) * CONL*CONW*CONH) * 100.        T  11270
      RETURN                                                            T  11350
C&    DEBUG INIT,TRACE,SUBCHK,SUBTRACE,UNIT(30)                         T  11360
      END                                                               T  1137C
```

```
      SUBROUTINE RESETC                                                 T    11380
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                             T    11390
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                  T    11400
      CCMMON ALLGCN,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,           T    11410
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBCX,NRPERM,NRTURN,                  T    11420
     2     NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P           T    11430
     3RINT,PLMIN,FWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME              T    11440
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXTT    11450
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF                       T    11460
      CCMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),        T    00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                          T    00100
      NL = NLHOLD                                                       T    11490
      MXTURN = IMXT                                                     T    11500
C             RESTORE C WHICH WAS DESTROYED BY                          T    11510
C             STUFFING SUBROUTINE OR BY UNSUCCESSFUL                    T    11520
C             OPTIMIZATION OVERALL (AS OPPOSED TO                       T    11530
C             OPTIMIZATION OF PALLET LOADING INTO                       T    11540
C             CONTAINER)                                                T    11550
      DO 20 I= 1,NL                                                     T    11560
      DC 10 J=1,5                                                       T    11570
      C(I,J) = CHCLD(I,J)                                               T    11580
   10 CCNTINUE                                                          T    11590
   20 CCNTINUE                                                          T    11600
      RETURN                                                            T    11610
C&    DEBUG INIT, TRACE, SUBCHK,SUBTRACE,UNIT(30)                       T    11620
      END                                                               T    1163C
```

```
      SUBROUTINE SHUFL                                                   T   11640
      LOGICAL ALLGON,B,FIRST,OUTSIZ,OPSTUF,                              T   11650
     1PRINT,PRELON,STACK,STUFED,GRAVTY                                   T   11660
      COMMON ALLGON,BOXL,BOXW,BOXH,CONL,CONW,CONH,FIRST,HIGH,            T   11670
     1IAP,ICP,IEP,MXTURN,OPSTUF,NL,NBOX,NRPERM,NRTURN,                   T   11680
     2     NXORG,NYORG,NZORG,OUTSIZ,PALL,PALW,PALH,PL,PW,PH,P            T   11690
     3RINT,PLMIN,PWMIN,RPH,STUFED,TIME,TRIED,TURNED,VOLUME               T   11700
     4,ISORT,GRAVTY,SMLX,SMLY,SMLZ,IOUTIN,NRLOOP,VOLIN,NLHOLD,NRSTF,IMXTT   11710
     5,TEFF,NRPLT,SEFF,CUMTIM,ISEED,PRELON,IAPEFF                        T   11720
      COMMON A(1000,7),B(1000,3),C(500,5),CHOLD(500,5),E(300,6),         T   00090
     1HCP(300),HCS(300),STACK(500),SORVEC(500)                           T   00100
      DIMENSION HCPHOL(300),HCSHOL(300)                                  T   00101
      N =.025*NL + 1                                                     T   11750
      ANL=NL                                                             T   11760
      DO 40 K=1,N                                                        T   11770
C               CAUTION:  SEE  SUBROUTIN INPUT FOR WARNING ON            T   11780
C                         USE OF FOLLOWING RN GENERATOR                  T   11790
      ISEED = ISEED * 65539                                              T   11800
      IF (ISEED .LT. 0) ISEED = ISEED + 2147483647 + 1                   T   11810
      I = IFIX (FLOAT(ISEED) * .4656613E-9 * ANL) + 1                    T   11820
      ISEED = ISEED * 65539                                              T   11830
      IF (ISEED .LT. 0) ISEED = ISEED + 2147483647 + 1                   T   11840
      L = IFIX (FLOAT(ISEED) * .4656613E-9 * ANL) + 1                    T   11850
      IF (.NOT. OPSTUF) GO TO 25                                         T   11860
      HCP(K) = I                                                         T   11870
      HCS(K) = L                                                         T   11880
      GO TO 26                                                           T   11885
   25 CONTINUE                                                           T   11890
      HCPHOL(K) = I                                                      T   11891
      HCSHOL(K) = L                                                      T   11893
   26 CONTINUE                                                           T   11894
      DO 30 J=1,5                                                        T   11900
      XX = C(I,J)                                                        T   11910
      YY = C(L,J)                                                        T   11920
      C(I,J) = YY                                                        T   11930
      C(L,J) = XX                                                        T   11940
      IF (OPSTUF) GO TO 30                                               T   11950
C               RECORD C IN CHOLD BECAUSE SUBROUTINE                     T   11960
C               STUFF DESTROYED C.  ALSO, IF NEW                         T   11970
C               OPTIMIZATION TRY WHILE GENERAL                           T   11980
C               LOADING, MUST RESTORE C IF TRIAL IS UNSUCCESSFUL         T   11990
      CHOLD(L,J) = XX                                                    T   12000
      CHOLD(I,J) = YY                                                    T   12010
   30 CONTINUE                                                           T   12020
   40 CONTINUE                                                           T   12030
      RETURN                                                             T   12040
      ENTRY RESTO                                                        T   12045
C               NOW REVERSE THE SHUFFLE IN ORDER                         T   12180
```

```
C               TO REPLACE THE ORIGINAL CRDER                                T  12190
C                 OF C                                                        T  12200
      N =.025* NL + 1                                                         T  12210
      NP1 = N + 1                                                             T  12220
      DC140 K=1,N                                                             T  12230
      M= NP1 - K                                                              T  12240
      IF (.NOT. OPSTUF) GO TO120                                              T  12245
      I = HCP(M)                                                              T  12250
      L = HCS(M)                                                              T  12260
      GC TO125                                                                T  12261
  12C CONTINUE                                                                T  12263
      I = HCPHOL(M)                                                           T  12264
      L= HCSHOL(M)                                                            T  12265
  125 CONTINUE                                                                T  12266
      DO130 J =1,5                                                            T  12270
      XX = C(I,J)                                                             T  12280
      YY = C(L,J)                                                             T  12290
      C(L,J) = XX                                                             T  12300
      C(I,J) = YY                                                             T  12310
      IF (OPSTUF) GO TO130                                                    T  12320
      CHOLD(L,J) = XX                                                         T  12330
      CHOLD(I,J) = YY                                                         T  1234C
  13C CONTINUE                                                                T  12350
  140 CONTINUE                                                                T  12360
      RETURN                                                                  T  12370
      END                                                                     T  12390
```

LIST OF REFERENCES

1. Brown, A. R., Optimum Packing and Depletion, Jeffreys and Hill Limited, 1971.

2. DeSha, Ernest Larry, Area-Efficient and Volume-Efficient Algorithms for Loading Cargo, M.S. Thesis, Naval Postgraduate School, Monterey, California, 1970.

3. Eilon, Samuel and Christofides, Nicos, "The Loading Problem," Management Science, Volume 17, Number 5, p 259-266, 1971.

4. Galata, A. Ya and Stoyan, Yu.G, "The Dense Packing of Parallelepipeds of Arbitrary Dimensions in a Parallelepiped of Least Volume," Cybernetics, Number 2, p 268-274, March 1972.

5. Gilmore, P. C. and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem-Part II," Operations Research, Volume 11, p 863-889, November 1963.

6. Gilmore, P. C. and Gomory, R. E., "The Theory and Computation of Knapsack Functions," Operations Research, Volume 14, p 1045-1074, November 1966.

7. Gilmore, P. C. and Gomory, R. E., "Multistage Cutting Stock Problems of Two and More Dimensions," Operations Research, Volume 13, p 94-119.

8. Ingargiola, Giorgio and Korsh, James F., "An Algorithm for the Solution of 0-1 Loading Problems," Operations Research, Volume 23, p 1110-1119, November 1975.

9. Rvachev, V. L. and Stoyan, Yu.G., "Algorithms for Constructing Inequalities Satisfied by the Location Parameters of Nonintersecting Bodies," Kibernetika, Volume 2, Number 6, p 82-92, 1966.

10. Department of Industrial and Systems Engineering, University of Florida, Gainsville, Report 40, Unitization and Deunitization in Physical Distribution Systems: A Qualitative and Quantitative Analysis of Containerized Cargo by Ravi M. Seam and B. D. Sivazlian, March 1970.

11. Stoyan, Yu.G. and Ponomarenko, L. D., "Algorithm for Approximate Solution of the Problem of Closest Packing of a Group of Parallelepipeds in a Parallelepiped with Forbidden Regions," Automatic Control and Computer Sciences, Volume 9, Number 1, p 41-48, 1975.

12. Camp, Gary L., Adapt Cargo Loading Algorithm to Navy Integrated Storage Tracking and Retrieval System (NISTARS), Staff Study at Naval School Transportation Management, Naval Supply Center, Oakland, California, March 1979.

13. Hicks, Charles R., Fundamental Concepts in the Design of Experiments, Holt, Rinehart, Winston, 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4.	Associate Professor Alan W. McMasters Code 55Mg Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
5.	LT. Ellen Roland, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
6.	LCDR N. B. Nelson III 2958 Hillsdale Drive Pleasant Hill, California 94523	3
7.	CDR Robert D. Grant Special Project Officer (Code 08) Naval Supply Center Oakland, California 94625	5
8.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, Virginia 23801	1
9.	Mr. H. J. Lieberman, Code 0431B Naval Supply Systems Command Washington, D. C. 20376	1