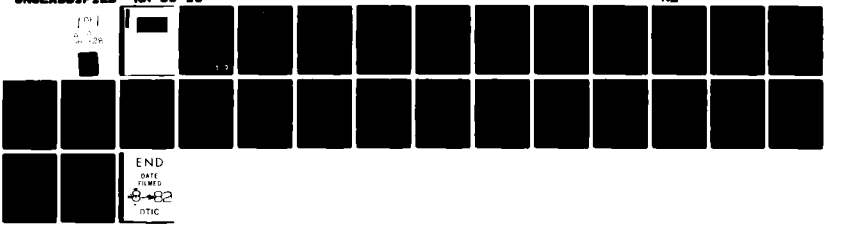
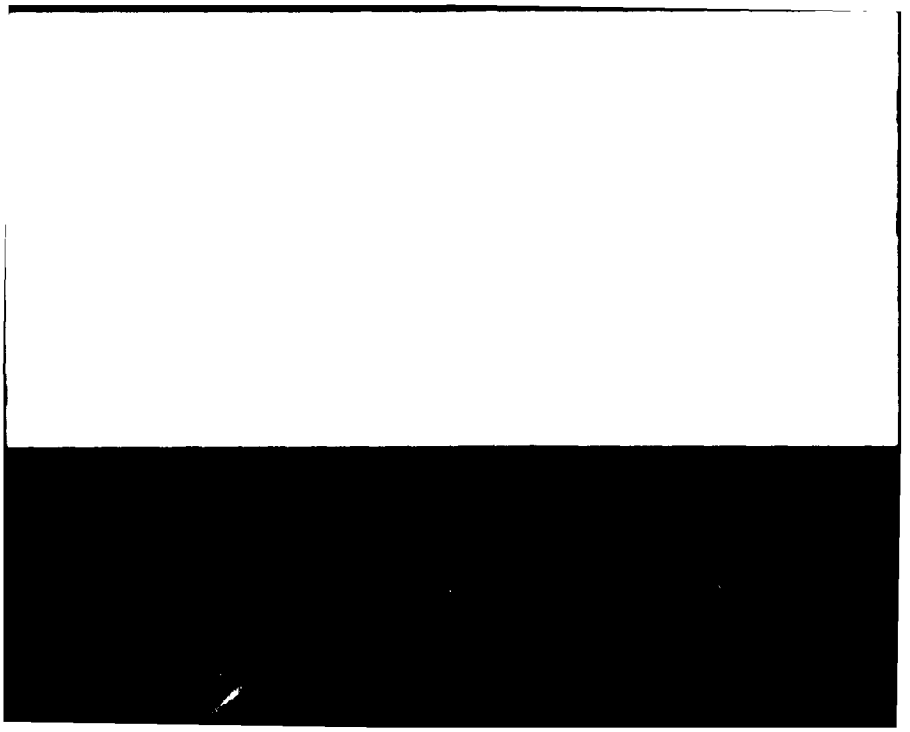
AD-A116 988 FLORIDA UNIV GAINESVILLE DEPT OF INDUSTRIAL AND SYS--ETC F/G 15/5
A COMBINED APPROACH TO THE PALLET LOADING PROBLEM.(U)
MAY 82 T J HODGSON N00014-76-C-0096
UNCLASSIFIED ISR-81-11 NL



END
DATE FILMED
-82
DTIC



A COMBINED APPROACH TO THE PALLET LOADING PROBLEM

Research Report No. 81-11

by

Thom J. Hodgson

August, 1981
revised
May, 1982

Department of Industrial and Systems Engineering
University of Florida
Gainesville, Florida 32611

Department of Applied Economic Sciences
Katholieke Universiteit Leuven
Leuven (Louvain), Belgium

This research was supported in part by the U.S. Air Force, under contract number F73AFL-00360001, by the Office of Naval Research, under contract number N00014-76-C-0096, and by the Onderzoeksfonds K.U. Leuven under Grant OT/IX/7.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 81-11	2. GOVT ACCESSION NO. AD-A116928
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) A COMBINED APPROACH TO THE PALLET LOADING PROBLEM	5. TYPE OF REPORT & PERIOD COVERED Technical
	6. PERFORMING ORG. REPORT NUMBER 81-11
7. AUTHOR(s) Thom J. Hodgson	8. CONTRACT OR GRANT NUMBER(s) F73AFL-00360001 (Air Force) N00014-76-C-0096 (Navy) OT/IX/7 (Onderzoeksfonds)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Industrial and Systems Engineering University of Florida Gainesville, Florida 32611	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - A. F. Logistics Arlington, VA Mgmt. Center Gunter AFS, Alabama	12. REPORT DATE 8/81 - Revised May, 1982
	13. NUMBER OF PAGES 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC ~~DISTRIBUTION~~ 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)

Pallet Loading
Stock Cutting
Dynamic Programming
Packing

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

In this paper the two-dimensional pallet loading problem is considered: that is, the problem of loading a rectangular pallet of size "L" by "W", drawing from a set of "n" rectangular boxes. The objective is to maximize the area covered on the pallet by the boxes loaded. The problem is approached using a combination of Dynamic programming and heuristics. The structured solutions resulting from the application of the "dynamic program" have two serendipitous characteristics: any item may be placed on the periphery of the pallet for easy access, and some control may be retained over the center of gravity of the pallet. Computational results are given.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

A COMBINED APPROACH TO THE PALLET LOADING PROBLEM

ABSTRACT

In this paper the two-dimensional pallet loading problem is considered : that is, the problem of loading a rectangular pallet of size "L" by "W", drawing from a set of "n" rectangular boxes. The objective is to maximize the area covered on the pallet by the boxes loaded. The problem is approached using a combination of Dynamic programming and heuristics. The structured solutions resulting from the application of the "dynamic program" have two serendipitous characteristics : any item may be placed on the periphery of the pallet for easy access, and some control may be retained over the center of gravity of the pallet. Computational results are given.

INTRODUCTION

Much of the packaged material which is shipped in trucks, railcars, aircraft, and ships is packed on a pallet or in some other bulk container. The packing problem can be stated simply as trying to pack as many packages as possible into a container. Certainly the general packing problem would include irregularly shaped packages and containers. However, in this paper, only rectangularly shaped packages (boxes) and containers (pallets) are dealt with. There are, at least, two major problems that can be identified as "The Pallet Packing Problem". The first problem could be called "The Manufacturer's Pallet Packing Problem". In this problem, the manufacturer produces a product which is packaged in identical boxes; the boxes may be packed in identical cartons; the cartons are packed on identical pallets; and the pallets are loaded in standard sized trucks, railcars, or shipping containers. The problem is to choose the package, carton, pallet, (and possibly the container) dimensions so that the volume of product packed in a container is maximized. This problem requires a one-time analysis to find the solution. With the exception of Steudel [18], little has appeared on this problem in the open literature. However, it is clear that industry is attacking this problem and several consulting firms offer services in this area.

The second problem could be called "The Distributor's Pallet Packing Problem". In this problem, the distributor fills an order from a customer. The order is packaged in boxes of varying dimensions. The problem is to pack the boxes on a standard pallet so as to maximize the volume placed on each pallet (i.e., minimize the number of pallets used to ship the order). The problem requires a new analysis for each pallet packed. As a consequence, from an economic stand point, the cost of a solution for the Distributor's Problem can be, at most, a fraction of the cost of a solution for the Manufacture's Problem. In addition, in most applications, the Distributor's Problem must be solved quickly (i.e., real-time computation) in order

for the solution to be applied.

For the Manufacturer's Problem, present technology supports the packing of pallets using automated material handling systems. However, the Distributor's Problem, by its nonrepetitive nature and solution time requirements, is more difficult. In order to automate the physical packing of distributor's pallet, one first needs a packing algorithm which essentially is real-time.

The problem addressed in this paper is a constrained version of the Distributor's Problem. Some of the boxes to be packed on the pallet may contain volitile liquids or explosives. As a consequence, those items must be packed on the periphery of the pallet so that, if necessary, they can be removed quickly. This problem is faced by the U.S. Air Force when they transport palletized cargo consisting of military equipment and supplies. In this case, since the pallets are loaded by hand it is necessary only to give a plan for loading, which might be modified slightly during the actual packing.

BACKGROUND

The pallet loading problem is related to problems long studied in the Operations Research literature : The Cutting Stock and Bin Packing Problems. A review of the literature is beyond the scope of this paper-however, the interested reader is directed to reviews by Golden [10], Hinxman [14], and Garey and Johnson [8]. Between these papers, over 125 articles, books, and papers are reviewed. Of particular interest is the work typified by Coffman, et al. [4] and Baker, et al [2]. The approach is to develop efficient, relatively simple, approximation algorithms, study their limitations and derive worst-case bounds on the performance of the packings they produced. Other approaches have concentrated

on more complex procedures for which performance bounds are more difficult to provide. This paper falls in the latter category.

Of interest is the work of Gilmore and Gomory [9] in their development of knapsack functions. Madsen [16] used a heuristic adaptation of their approach for a glass cutting problem. Adamowicz and Albano [1] used a combination of heuristics and dynamic programming. In their cutting stock problem they generated "strips" of like sized boxes which were fit into rectangular parts of the sheet using D.P. The partitioning of the sheet was performed heuristically. Haims and Freeman [12] applied a dynamic programming approach for a template layout problem, assuming that there is an unlimited supply of boxes of each type and boxes can have only one orientation. The procedure described in this paper is, in fact, a generalization and extension of their approach.

DeSha [6], in an unpublished master's thesis, developed a heuristic for loading containers. His procedure first sets up stacks of items to fit the container height, then loads the stacks in the container to maximize the container floor area covered. The heuristic appears to obtain quite good results using a data base with boxes whose dimensions are randomly generated. Finally, it should be noted that de Cani [5] has shown that non-orthogonal packing may lead to "better" solutions. However, that observation makes more sense for the cutting stock problem than a pallet loading problem.

The pallet loading problem falls in the category of problems called NP-HARD [7]. Consequently, a truly efficient optimal algorithm is not likely to be forthcoming. In developing an approach to the problem that would be consistent with the special needs of the USAF, it also became clear that it would be highly desirable for the system to be interactive.

This would allow a user to guide the solution of a particular loading problem in order to deal with those unquantifiable elements of a "real world" loading problem. With these observations in mind, a system called IPLS (Interactive Pallet Loading System) was developed. A partial description of that system is in [15]. In this paper, the algorithmic development and conceptual use of such a system is discussed.

In the following, a pallet loading procedure is developed for the two-dimensional loading problem. Then ways of using this procedure to load real-life (three-dimensional) pallets is discussed. Computational experience is presented.

PALLET LOADING PROCEDURE

The pallet loading procedure can be described as a combination of the principles of Dynamic Programming [17] and heuristics. To understand the procedure, it will be useful first to consider a "best" procedure. It will be obvious that the "best" procedure is computationally infeasible. Therefore, structural limitations will be introduced which limit the computational effort. A serendipitous by-product of the resulting procedure structure is that positioning of hazardous material and considerations of center of gravity can be handled without loss within the procedure.

Assume that it is desirable (no matter what the cost) to find solutions to the two-dimensional pallet loading problem which maximize the area covered on the pallet. In order to achieve this end, let us consider Dynamic Programming as a solution methodology. The following definitions will be useful.

P = A partition dividing the pallet into two parts (see figure 1). The left-hand sub-pallet must include the origin (0,0), and the right-hand sub-pallet must include the point (L,W).

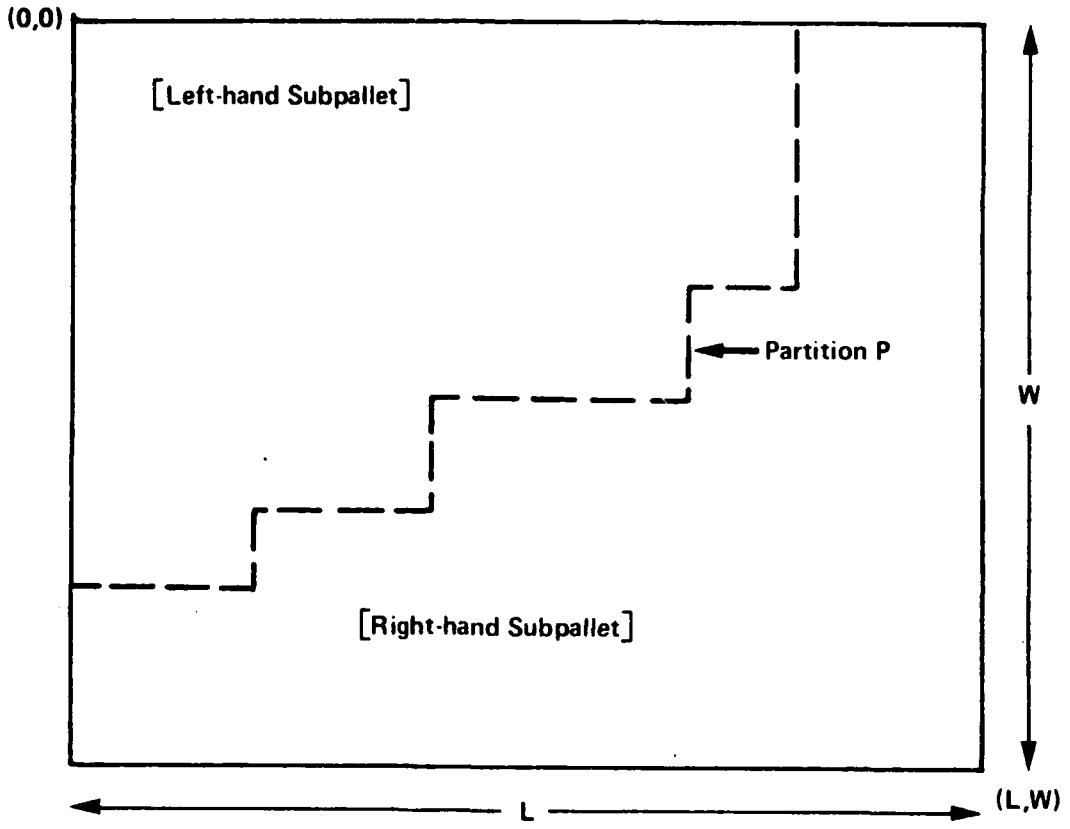


Figure 1: Plan View of Pallet with Sample Partition P

i = The index of boxes to be loaded, i=1,...,n.

l(i) = The length of box i.

w(i) = The width of box i.

S(i) = The profile (shadow) of box i. The profile is a rectangle with length l(i) and width w(i).

N = Set of all boxes to be considered for loading (of size n).

I = Subset of the boxes, 1,2,..., n.

f(P,I) = The maximum area which can be covered of the left-hand sub-pallet of P using the subset of boxes I.

The Dynamic Programming equation for the pallet loading problem can be given as follows :

$$(1) \qquad f(P,I) = \max_{i \in I} \left[l(i) \times w(i) + f(P-S(i), I-i) \right].$$

It should be noted that the notation 'P-S(i)', represents a partition which, in a graphical sense, is the partition P with a profile of box i removed from the right-hand edge of the left-hand sub-pallet. Typically, there could be many different realizations of 'P-S(i)' that should be considered within a dynamic optimization. There are other obvious difficulties with implementing equation (1). The most obvious is that the number of possible partitions P of the pallet is extremely large. This means that the state space for the Dynamic Program will require large amounts of computer storage. It also means that the computer time required to solve the Dynamic Program likely would be well beyond any sensible limit for real world applications.

One approach to developing a more tractable procedure is to limit in some way the form of the possible partitions of the pallet. In the present case, partitions of the pallet have been limited to rectangles (figure 2a). In order to specify the Dynamic Program resulting from the rectangular partitions, the following additional definitions are needed.

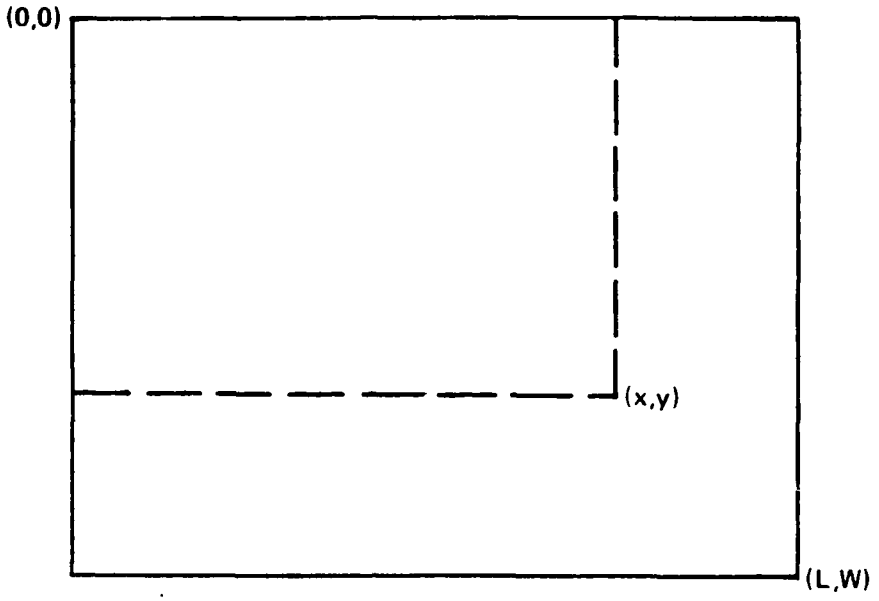


Figure 2a: Pallet with Rectangular Partition (x,y)

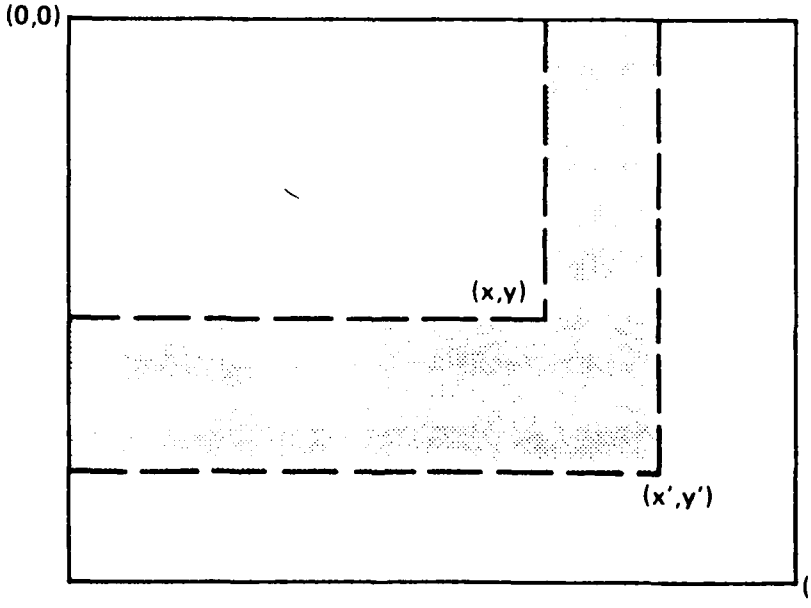


Figure 2b: Pallet with Two Rectangular Partitions (x,y) and (x',y')

x,y = Two-dimensional index specifying a rectangular partition (figure 2a).

J = Subset of the boxes, 1,2,...,n.

s(x,y,I) = The maximum area which can be covered of the left-hand sub-pallet of x,y using the subset of boxes I.

h(x,y,x',y',I) = The maximum area which can be covered of the left-hand sub-pallet of x',y' less the left-hand sub-pallet of x,y using boxes from the set I (not all elements of I necessarily are used, see figure 2b).

The Dynamic Programming equation for the pallet loading problem (limited to rectangular partitions) can be given as follows :

$$(2) \quad s(x',y',I) = \max_{\substack{x<=x' \\ y<=y' \\ J \subset I}} [\, s(x,y,J) + h(x,y,x',y',I-J) \,]$$

The implementation of equation (2) also has its difficulties. The function h(x,y,x',y',I-J) itself requires an optimization in order to pack the L-shaped area common to the partition x',y', but not common to the partition x,y (i.e., I-J is the cross-hatched area in figure 2b). The state-space is still too large to deal with on a practical basis.

In order to limit the size of the state space, it is necessary to carry only one partial solution (s(x,y,I)) for each partition x,y. The obvious choice is to carry

$$\max_{I} [\, s(x,y,I) \,].$$

With this limitation on the state space, the implementation of equation (2) is relatively straightforward. It is necessary, however, to specify several important details first :

1. an optimization structure for h(x,y,x',y',I-J);
2. bounding rules for the elimination of partial solutions;
3. bounding rules for the minimization of computational effort.

The optimization for h(x,y,x',y',I) is done simply by breaking the L-shaped area into two rectangular areas (figure 3) and filling each area using a linear Dynamic Programming procedure. The following definition is useful.

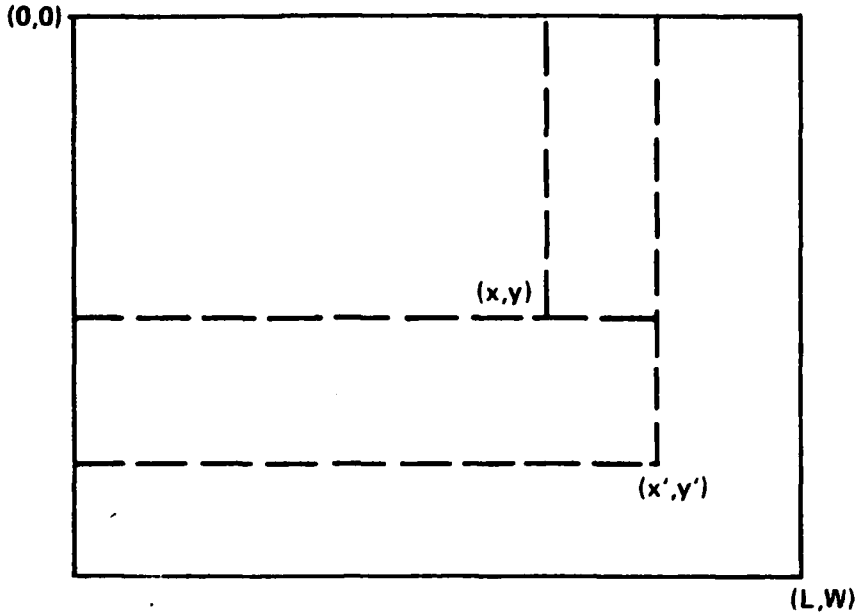
b(j,x) = The maximum possible space covered in a rectangular area of length x by stacking boxes from the set 1,...,j (Note the limitation of "linear" stacking imposed).

The Dynamic Programming equation for the rectangular loading problem can be given as follows :

$$(3) \quad b(j,x) = \max \left[b(j-1,x),\ b(j-1,x-l(j)) + l(j)*w(j) \right]$$

The implementation of equation (3) is achieved by first eliminating boxes too large to fit in the rectangle, then turning each box in the candidate set so that its longest dimension is perpendicular to the long dimension of the rectangular area (if the longest dimension of the box is less than or equal to the short dimension of the rectangular area, that is). This insures an optimal packing of the rectangular area. The L-shaped area is broken into two rectangular areas (corridors) two different ways (figure 3) for the application of equation (3). The best solution obtained, in terms of area covered, is retained.

A simple, and almost obvious, bounding rule that eliminates a great deal of the storage requirements for the state space in computing equation (2) is that a partial solution does not need to be retained for the partition x,y if there exists a partial solution for a partition x',y',



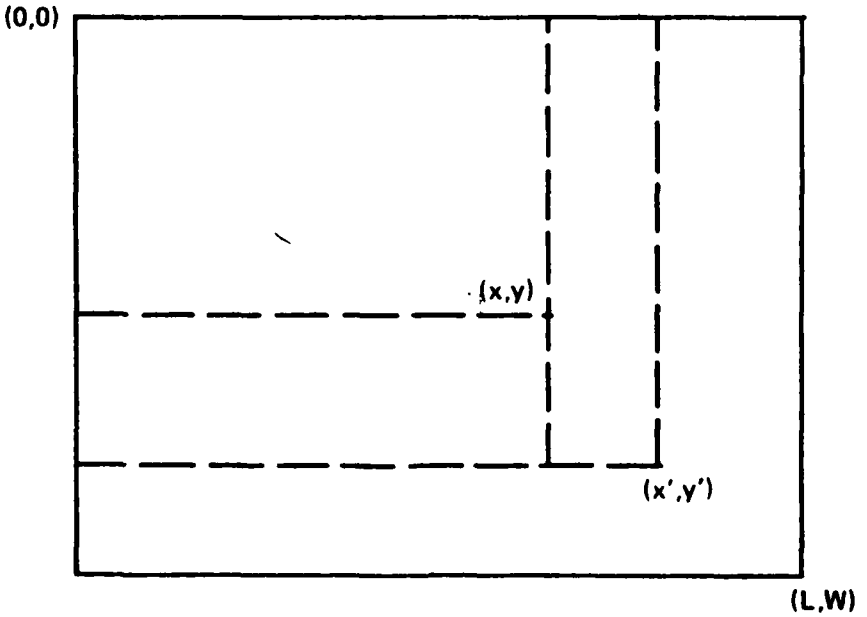


Figure 3: Two Ways to Break Up L-Shaped Area

$(x' <= x,\ y' <= y)$ such that the solution value for x', y' ($\max [\, g(x', y', I) \,]$) is greater than or equal to the solution value for x,y ($\max [\, g(x,y,I) \,]$).

Another effective bounding scheme is used to eliminate computation time. It involves computing an upper bound on the amount to be loaded in a rectangular area of dimension LxW by using the result of a linear Dynamic Program in multiplicative fashion. The following definition is useful.

$c(j,x)$ = The maximum possible linear space covered in a length x by stacking boxes from the set $1,\ldots,j$.

The Dynamic Programming equation for the linear loading problem can be given as follows :

$$(4) \qquad c(j,x) = \max [\, c(j-1,x), c(j-1,x-l(j))+l(j), c(j-1,x-w(j))+w(j) \,]$$

The function c(n,x) specifies the maximum linear coverage that is possible on the line segment $[\,0,x\,]$ choosing from the set of boxes $1,\ldots,n$ (positioning them by either length or width). For a pallet (rectangular sub-pallet) of size L by W, an upper bound on the maximum load (coverage) possible is $c(n,L) \times c(n,W)$. The function c(n,x) can be computed prior to the pallet loading and is easily implemented within the structure of equation 2. The upper bound can be used to eliminate the need to consider a given partial solution (x',y') in equation (2) altogether. It can also be used to eliminate the computation of Dynamic Program equation (3) within the optimation of equation (2) when considering a specific partial solution (x,y).

A more powerful bounding procedure can be used for rectangular sub-pallets of certain dimensions. Clearly, if the width of the sub-pallet is less than the minimum dimension (length and/or width) in the candidate box

set, it is impossible to pack any of the candidate boxes on the sub-pallet. Consequently, the upper bound on the amount which can be loaded is zero. Now, let MINDIM equal the minimum dimension in the candidate box set. If the width of the sub-pallet satisfies

$$MINDIM <= W < 2*MINDIM,$$

then boxes not fitting within the width of the sub-pallet can be eliminated from the computation of equation (4), and the upper bound function is just c(n,x,W) (where the "W" indicates the elimination of non-fitting box lengths and/or widths from the candidate set, i.e., l(j) > W and/or w(j) > W). The matrix layout of the bound is shown graphically in figure 4.

Another bound to suppliment the above bounds can be calculated. The following definitions are useful.

a(i) = The area of box i (i.e., a(i) = l(i)*w(i))
d(j,z) = The maximum possible area covered on a pallet of area z by loading boxes from the set 1,...,j, and ignoring considerations of box shape.

The dynamic Programming equation for the area loading problem can be given as follows :

(5) $$d(j,z) = \max [\, d(j-1,z),\ a(j)+d(j-1,z-a(j)) \,]$$

The function d(j,z) specifies the maximum area coverage that is possible on a pallet of size z, choosing from the set of boxes 1,...,n, and assuming that the boxes can be "mashed" into any shape without loss of area. The upper bound for a rectangular pallet (sub-pallet) of size L by W, is just d(n,L*W). The upper bound used in the implimentation of equation (2), then, is just the minimum of all the bounds described above.

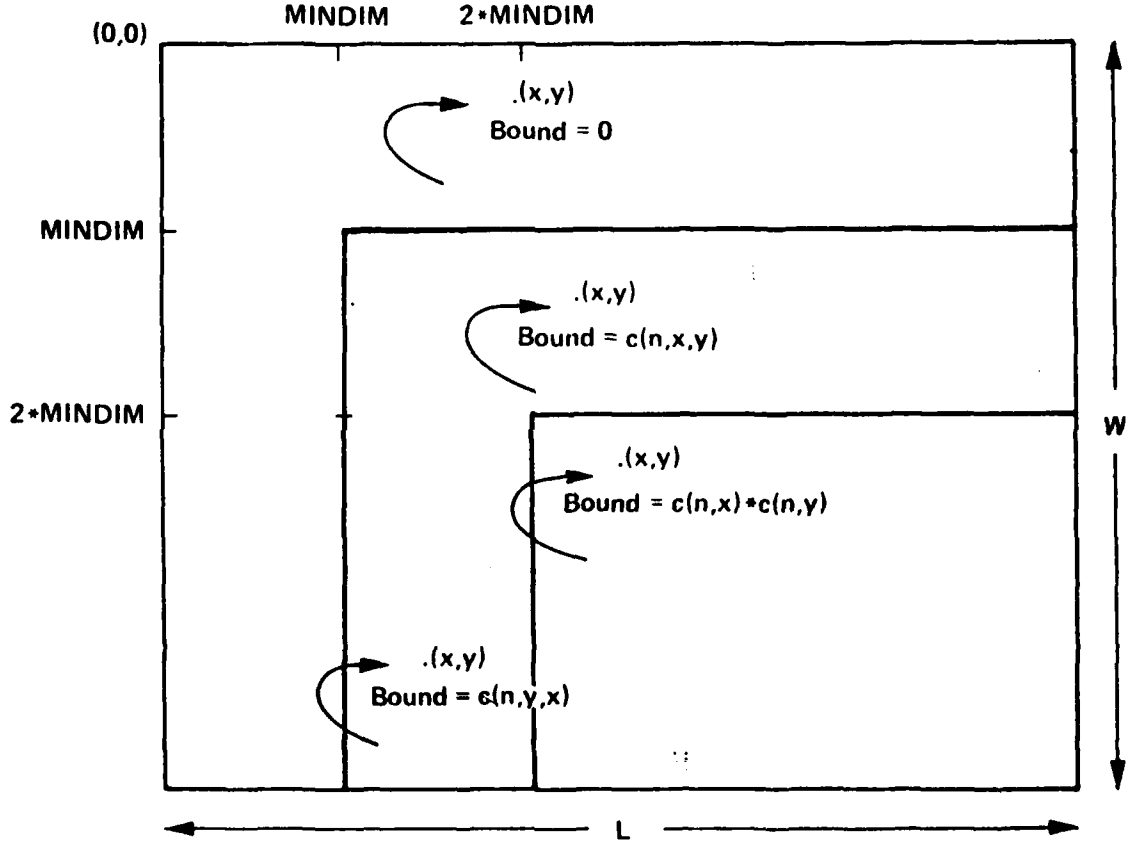


Figure 4: Matrix Layout of Bounds for Pallet (sub-pallet)

The solution procedure results in the boxes being placed in corridors on the pallet. Two serendipitous by-products occur. First, since each corridor has at least one end on the pallet perimeter, any box which contains hazardous material can be placed at the end of the corridor (as per USAF Regulations) within the structure of the solution. It is possible to set multiple "hazardous" boxes on a corridor, so there is no guarantee that any box can be placed on the periphery of the pallet. However, it has been our experience that the probability of not being able to do so is virtually zero. Second, since boxes can be moved within their assigned corridors, some control can be maintained over the center of gravity of the pallet within the structure of the solution. Also, it should be noted that any solution can be decomposed by a series of guillotine cuts.

AN INTERACTIVE APPROACH

Since real-life pallet loading problems typically have considerably more complexity than the present formulation, the approach described above is best used as part of a highly interactive computerized system. The interactive pallet loading system (IPLS) is such a system [15]. The system is, at this point, an experimental system used for develomental purposes. While it is not intended to present the details of IPLS, some description of its functions is appropriate. IPLS is a fully interactive menu driven system. It is intended to use as its data base the basic equipment load of an Air Force squadron. For each box the following characteristics are included : length, width, height, weight, organizational element in the squadron, "stackability" code, "this end up" code, and hazardous material code. IPLS provides the ability to identify those boxes belonging to the squadron and associated with some subset of the squadron which must be packed up for movement. The user also is provided with the ability to select subsets of boxes with desireable attributes (common height, limits

on length and width, etc.) for packing. In addition there is an ability to rotate boxes (for common height, etc.) in the data base.

The following illustrates two ways that the two-dimensional pallet loading algorithm can be used to load three-dimensional pallets. One way is to pack boxes a level at a time; each level made up of boxes of common height. This way there is a box selection process for each level and the loading algorithm is applied once for each level. This process can be continued until the desired pallet height has been achieved. IPLS facilitates this approach to the load planning process in a straightforward way. The result is a pallet with a "layered" load as shown in figure 5a.

A second approach is to load the pallet with columns (or stacks) of boxes. That is, make up stacks of boxes such that the stacks are no higher than the maximum allowable pallet height. Then the stacks can be used as input to the loading algorithm, which is applied only once to load the pallet, IPLS also facilitates this approach. It is possible to create the stacks interactively through the use of a dynamic program which maximizes the volume loaded over a user defined base box. The result is a pallet with a "stacked" load as shown in figure 5b.

It should be noted that, since it may not be possible to load the pallet to 100 % density, gaps will normally occur in a solution which could result in unstable loads. However, that is not the case in the Air Force application for the following reasons :

1. The load plans are to be implimented manually, allowing "tightening" of the load. (In addition, IPLS has a heuristic procedure for "tightening" a load).
2. Voids which cannot be eliminated and would contribute to load instability can be filled with packing material.

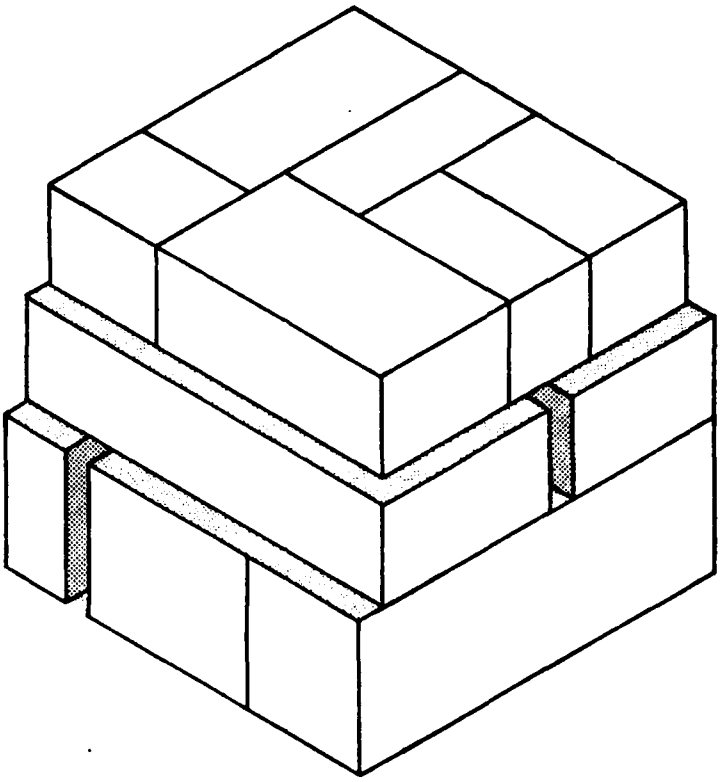


Figure 5a: "Layered" Pallet Load

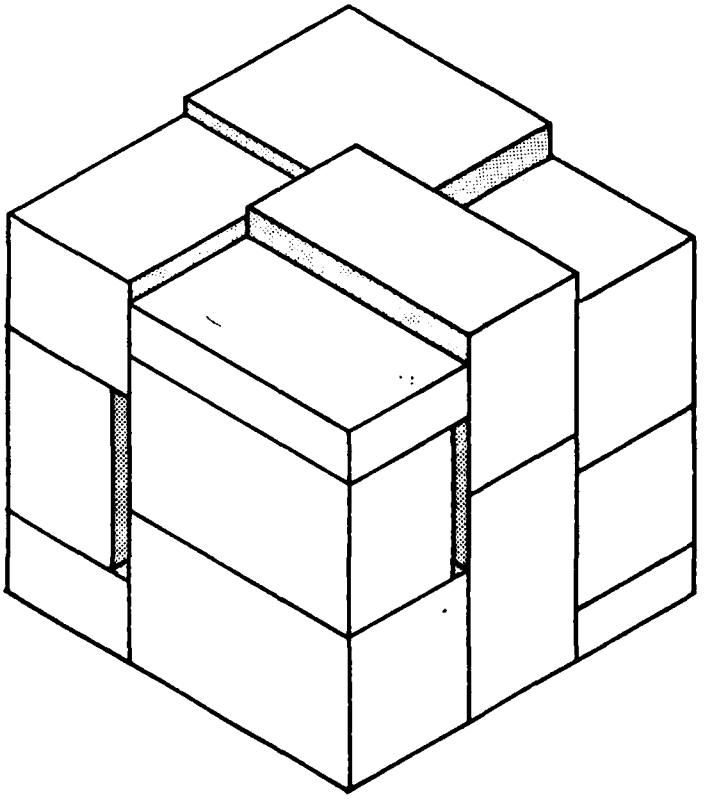


Figure 5b: "Stacked" Pallet Load

3. Loaded pallets are covered with a nylon net which is then tighted. This results in extremely stable loads.

Finally, it has been observed that USAF personnel often start loading a pallet by selecting a large box and placing it in one corner of the pallet (in fact, many times, it is, from a practical standpoint, necessary to do so). Then the pallet load is built by stacking boxes around the large box. IPLS is provided with the capability of performing the same selection process interactively. The act of selecting the first box of the load greatly reduces the combinatorics of the problem and reduces computational effort dramatically.

EXPERIMENTAL COMPUTATIONS

The pallet loading procedure was programmed in fortran IV and implimented on both a PDP-11/34 and the Katholieke Universiteit Leuven IBM 3033. Problems with a data set of 30 boxes were generated with box dimensions uniformly distributed (integer values only) between the upper and lower bounds as indicated in Tables 1, 2, and 3. Table 1 contains computation times for the IBM 3033 which are given in virtual seconds. Computation times (in CPU seconds) for the PDP 11/34 are approximately 20 times those observed for the IBM 3033. Table 2 contains the number of undominated partial solutions generated by the dynamic program. Table 3 contains the percent area of the pallet covered by the boxes loaded. Each entry of Tables 1, 2, and 3 is the average of 5 problems. The maximum deviation of individual computer runs from the average reported is smaller than one might initially expect. For instance, the maximum percent deviation above the mean reported computation time is 47 %. The deviations from the mean for undominated partial solutions and (particularly) percent coverage are considerably less.

Computation times can be seen to be dependent on both pallet size and the range of box dimensions (Table 1). The times can be considered to be upper bounds in that computational experience with data sets derived from a USAF data base tends to be better than randomly generated problems. For instance, a data set consisting of boxes of two sizes (17'x19' and 18'x19') was used to load a standard USAF pallet (104'x84') in approximately 1.52 seconds. A set of boxes randomly generated between 19' and 17' (6 possible box sizes) was used to load the same size pallet in approximately 4.46 seconds.

The number of undominated partial solutions generated by the Dynamic Program also can be seen to be dependent on both pallet size and the range of box dimensions (Table 2). This indicates the growth in the requirement for computer storage. The storage requirement does not grow at the same rate as computational requirement since it is bounded by the number of integral partitions of the pallet. However, on a computer of limited size this can be a problem (i.e. a PDP 11/34).

The quality of the solutions to the randomly generated problems as measured by the percent of the pallet area covered, is extremely good (Table 3). 4 out of the 5 pallet sizes used in the experimentation (60x40, 80x60, 80x80, and 100x80) were chosen so that there would be a high likelyhood of the existance of an extremely good solution. 74 out of the 85 problems generated for these pallet sizes were solved with 100 percent coverage. In the worst individual case, the coverage was 99.08 percent. One pallet size used in the experimentation (70x50) was chosen so that there would not be a high likelyhood of good solutions. With all boxes having dimensions of 20x20, the best possible coverage would be only 68.6 percent. As the randomness of the box dimensions increases, the quality of solutions increases. With box dimensions randomly distributed between10 and 30, the coverage was 99.70 percent. The worst individual case of the 5 was 99.14 percent. While it is impossible to compare fairly the quality of the solutions obtained here with those of other procedures (for instance [1, 12, 16]), it would appear there is little room for improvement.

Table 1 : Computation times in virtual seconds

Box Size	AREA PALLET DIMENSIONS				
Upper bounds / lower	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
18 / 22	.039	.155	.674	1.60	6.07
17 / 23	.081	.350	1.60	3.48	13.537
16 / 24	.140	.864	2.73	6.33	25.44
15 / 25	.193	1.437	5.13	9.07	*
10 / 30	3.112	20.136	*	*	*

* not solved in 30 virtual seconds.

Table 2 : Undominated partial solutions generated

Box Size	AREA PALLET DIMENSIONS				
Upper bounds / lower	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
18 / 22	57.0	65.2	207.4	263.4	554.8
17 / 23	97.0	126.2	384.6	484.0	1059
16 / 24	135.4	238.8	591.4	769.4	1551.5
15 / 25	174.8	379.6	766.6	988.6	*
10 / 30	423.0	842.2	*	*	*

* not solved in 30 virtual seconds

Table 3 : Percent coverage of the pallet area.

Box Size	AREA PALLET DIMENSIONS				
Upper bounds / lower	2400 60 x 40	3500 70 x 50	4800 80 x 60	6400 80 x 80	8000 100 x 80
20 / 20	100.	68.6	100.	100.	100.
18 / 22	100.	75.5	100.	100.	99.95
17 / 23	100.	83.5	100.	99.94	100.
16 / 24	100.	93.2	100.	99.95	99.93
15 / 25	100.	99.0	100.	99.84	*
10 / 30	99.5	99.7	*	*	*

* Not solved in 30 virutal seconds.

Finally, it would appear that potential improvements in the procedure would most likely be in two areas. First, the linear space packing (knapsack) dynamic program is performed countless times in an individual problem solution. Significant reductions in computation for that procedure would certainly reduce computation times. The present D.P. code used has been programmed to take advantage of the characteristics of the particular data set used. This is particularly effective because of the fact that most packings that are performed in the course of solving a problem are relatively small. It would appear initially that a good fast heuristic might speed things up considerably. However, this clearly would come at a cost in the quality of the solutions obtained. Preliminary analysis indicates that the computational effort for a heuristic which would insure high quality solutions might be on the same order as the present procedure. What is necessary to resolve that conflict is to test a series of increasingly complex heuristics in order to find the trade-off between computation time and solution quality. That is beyond the scope of this paper.

Second, the bounding function used presently has achieve considerable reductions in computational effort (experimental runs with and without the bounding function resulted in reductions on the order of up to 30-1). Further improvements in the bounds would have strong potential for improving running times.

ACKNOWLEDGEMENTS

This research was supported, in part, by the U.S. Air Force, under contract number F73AFL-00360001, by the office of Naval Research, under contract number N00014-76-C-0096, and by the Onderzoeksfonds, Katholieke Universiteit Leuven, under grant OT/IX/1. I am indebted to an anonymous referee who made many suggestions and observations on the first version of this paper. This paper owes much to him/her.

REFERENCES

[1] ADAMOWICZ, M., and ALBANO, A., "A Solution of the Rectangular Cutting-Stock Problem", IEEE Transaction on Systems, Man, and Cybernetics, Vol. SMC-6, No. 4, April 1976, pp. 302-310.

[2] BAKER, B.S., COFFMAN, E.G., and RIVEST, R.L., "Orthogonal Packings in Two Dimensions", SIAM Journal on Computing, Vol. 9, No. 4, Nov. 1980, pp. 846-855.

[3] CHRISTOFIDES, N., and WHITLOCK, C., "An Algorithm for Two-Dimensional Cutting Problems", Operations Research, Vol. 25, No. 1, Jan. 1977, pp. 30-44.

[4] COFFMAN, E.G., GAREY, M.R., JOHNSON, D.S., and TARJAN, R.E., "Performance Bounds for Level-Orientated Two-Dimensional Packing Algorithms", SIAM Journal on Computing, Vol. 9, No. 4, Nov., 1980, pp. 808-826.

[5] DE CANI, P., "A Note on the Two-Dimensional Rectangular Cutting-Stock Problem", Journal of Operational Research Society, Vol. 29, No. 7, 1978, pp. 703-706.

[6] DeSHA, E.L., "Area Efficient and Volume Efficient Algorithms for Loading Cargo", Masters Thesis, United States Navy Post-Graduate School, Sept. 1970.

[7] GAREY, Michael, and JOHNSON, David, Computers and Intractability, W.H. Freeman, San Francisco, 1979.

[8] GAREY, M.R., and JOHNSON, D.S., "Approximation Algorithms for Bin Packing Problems : A Survey", Analysis and Design of Algorithms for Bin Packing in Combinatorial Optimization, G. Ausiello and M. Lucertini, Eds., Springer, Vienna 1981, pp. 147-172.

[9] GILMORE, P.C., and GOMORY, R.E., "Theory and Computation of Knapsack Functions", Operations Research, Vol. 14, 1966, pp. 1045-1074

[10] GOLDEN, B.L., "Approaches to the Cutting Stock Problem", AIIE Transactions, Vol. 8, No. 2, June 1976, pp. 265-272.

[11] HAHN, S., "On the Optimal Cutting of Defective Glass Sheets", IBM N.Y. Scientific Center Report No. 320-2916, 1967.

[12] HAIMS, M.J. and FREEMAN, H., "A Multistage Solution of the Template-Layout Problem", IEEE Transactions on Systems Science, and Cybernetics, Vol. SSC-6, No. 2, April 1970, pp. 145-151.

[13] HERZ, J.C., "A Recursive Computing Procedure for Two-Dimensional Stock Cutting", IBM Journal of Research and Development, Vol. 16, 1972, pp. 462-469.

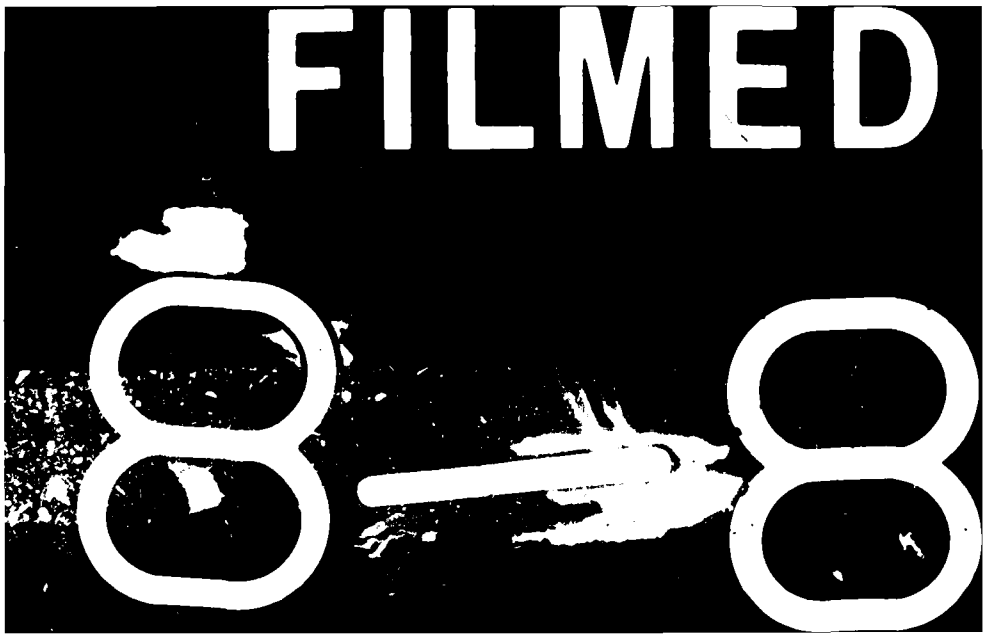
[14] HINXMAN, A.I., "The Trim-Loss and Assortment Problems : A Survey", European Journal of Operational Research, Vol. 5, 1980, pp. 8-18.

[15] HODGSON, Thom J., "IPLS : Interactive Pallet Loading System", Research Report No. 81-9, Industrial and Systems Engineering Department, University of Florida, Gainesville, Florida 32611, June 1981.

[16] MADSEN, Oli B.G.,"Glass Cutting in a Small Firm", Mathematical Programming, Vol. 17, 1979, pp. 85-90.

[17] NEMHAUSER, George L., Introduction to Dynamic Programming, John Wiley and Sons, Inc., New York, NY, 1966.

[18] STEUDEL, H.J., "Generating Pallet Loading Patterns : A Special Case of the Two-Dimensional Cutting Stock Problem", Management Science, Vol. 25, No. 10, Oct., 1979, pp. 997-1004.


FILMED