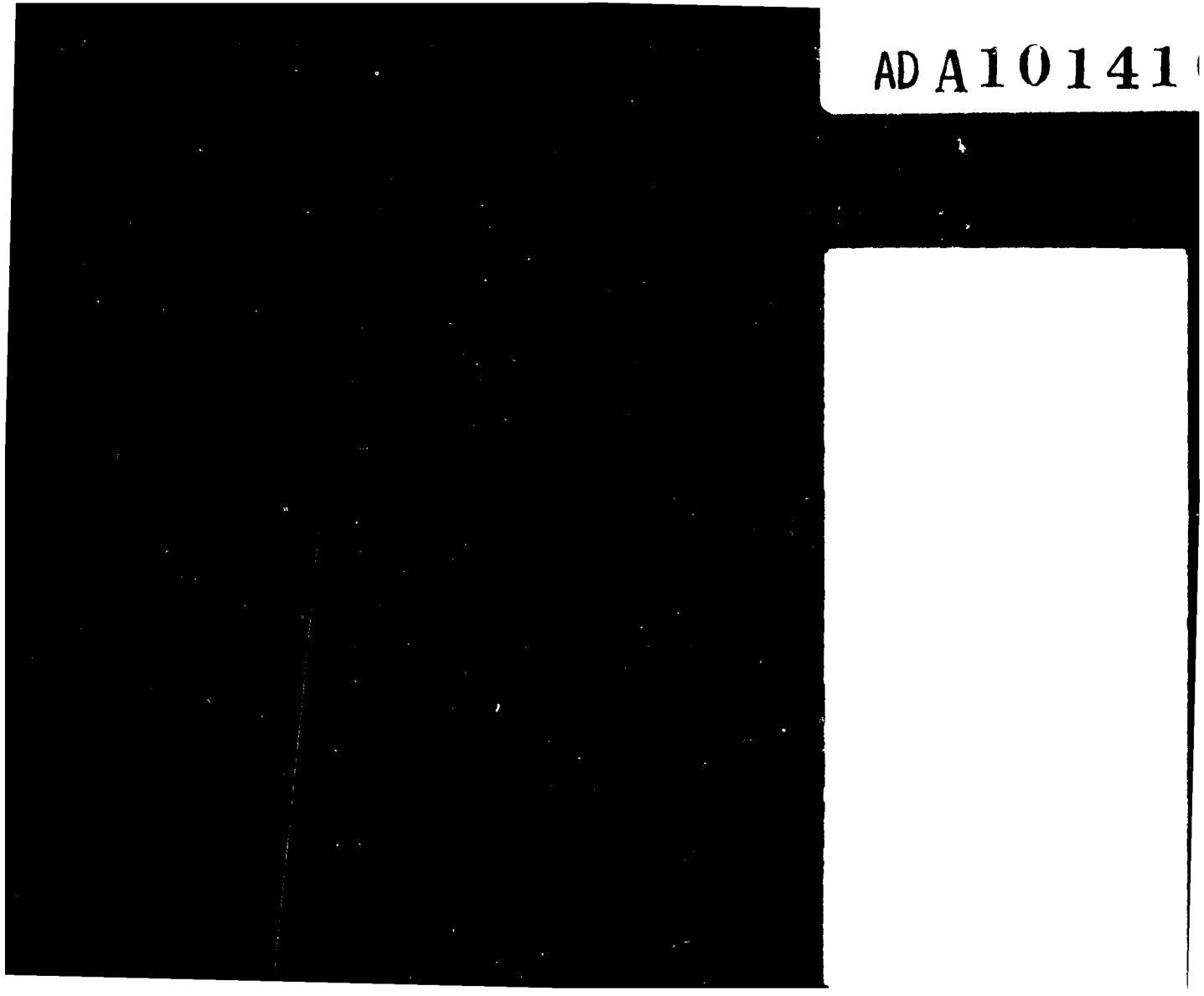
AD-A101 410 FLORIDA UNIV GAINESVILLE DEPT OF INDUSTRIAL AND SYS--ETC F/G 15/5
IPLS: INTERACTIVE PALLET LOADING SYSTEM,(U)
JUN 81 T J HODGSON
UNCLASSIFIED RR-81-9 N00014-76-C-0096 NL

END
DATE FILMED
8-81
DTIC


AD A10141

(12)

IPLS
INTERACTIVE PALLET LOADING SYSTEM

Research Report No. 81-9

by

Thom J. Hodgson

June, 1981

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

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This research was supported in part by the U.S. Air Force, under contract number F73AFL-00360001, and by the Office of Naval Research, under contract number N00014-76-C-0096.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 81-9	2. GOVT ACCESSION NO.: AD-A101410
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): IPLS: INTERACTIVE PALLET LOADING SYSTEM	5. TYPE OF REPORT & PERIOD COVERED: Technical
6. PERFORMING ORG. REPORT NUMBER: AR - 81-9	
7. AUTHOR(s): Thom J. Hodgson	8. CONTRACT OR GRANT NUMBER(s): F73AFL-00360001 (Air Force); N00014-76-C-0096 (Navy)
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: 1.128
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research - A.F. Logistics Mgmt. Center, Arlington, VA, Gunter AFS, Alabama	12. REPORT DATE: June 1981
	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 RELEASE: DISTRIBUTION UNLIMITED.

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

N/A

18. SUPPLEMENTARY NOTES

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

Pallet Loading
Dynamic Programming
Heuristic
Real-Time

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

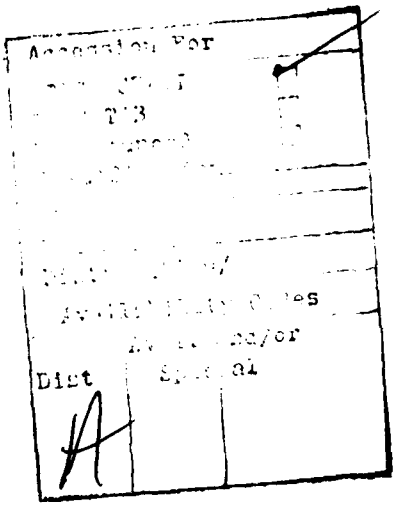
The USAF Interactive Pallet Loading System (IPLS) is a real-time system for the planning of loading pallets. The system used a combination of Dynamic Programming and Heuristics, along with user interaction to accomplish the pallet loading task. The system is designed to be interfaced with the Base Automated Mobility System (BAMS). Examples 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



ABSTRACT

The USAF Interactive Pallet Loading System (IPLS) is a real-time system for the planning of loading pallets. The system used a combination of Dynamic Programming and Heuristics, along with user interaction to accomplish the pallet loading task. The system is designed to be interfaced with the Base Automated Mobility System (BAMS). Examples are given.

Introduction

The USAF Interactive Pallet Loading System (IPLS) is an interactive computerized system to assist in the planning of loading pallets. IPLS is designed to interface with the USAF Base Automated Mobility System (BAMS) data base. The objective of IPLS is to automate the palletization of the unit material described in the BAMS data base, and to maximize the utilization of the available cube on the pallet. IPLS is programmed in FORTRAN IV on a PDP-11/34. In what follows, previous efforts and current research on the pallet loading problem are discussed along with a general statement of the difficulty of the problem and the philosophical approach taken here. In the following section the general structure and use of IPLS is presented. Then the mathematical structure of the pallet loading algorithm is laid out. Finally, comments on the requirements to implement IPLS in the "real world" are discussed along with potential future enhancements. IPLS, at this point, should be considered a developmental system.

Background

The pallet loading problem is related to a problem long studied in the Operations Research literature: The Cutting Stock Problem. The cutting stock literature is not discussed here, but the interested reader is directed to a recent review paper by Golden [5]. The two-dimensional cutting problem (of which the pallet loading problem is a special case) has been studied by Christofides and Whitlock [1], Gilmore and Gomory [4], Hahn [6], and Herz [7]. The two-dimensional cutting problem becomes the pallet loading problem when the requirement for guillotine type cuts is dropped (i.e.,straight cuts made in stages from one edge to the opposite edge of the object being cut, such as with a common paper cutter). Steudel [9] studied the pallet loading problem. However, his work was limited to the case where all boxes have the same length and width. He developed a procedure combining heuristics and dynamic programming.

DeSha [2], 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. The procedure does not, however, include considerations of center of gravity, positioning of hazardous material in the container, or box manipulation (no "this end up" assumption).

The pallet loading problem falls in the category of problems called NP-HARD [3]. 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,

IPLS has been developed.

Structure and Use of IPLS

IPLS is structured in two parts. The first part is the "BAMS Data Manipulator." The second is the "Pallet Loading Procedure." The Data Manipulator is a special purpose front end "sorter" that allows the user to obtain a set of candidate boxes for the Pallet Loading Procedure. The Data Manipulator is structured on the "menu" approach and uses a series of data bases in order to build the candidate set. The data bases are named BAMS.DAT, BAM1.DAT,... ,BAM4.DAT, and PACK.DAT. BAMS.DAT contains a card image file of the Base Automated Mobility System Data (obtained in the present case from Moody, AFB). BAM1.DAT is an unformatted copy of BAMS.DAT. BAM2.DAT,...,BAM4.DAT are temporary files used in the selection of candidate boxes. PACK.DAT is used to store the BAMS data for boxes allocated to the pallet.

The menu (option list) of the Data Manipulator contains seven options:

1. IDENTIFY THE LIST OF ITEMS
allows the user to choose which squadron increment(s) to include in the candidate data base. The load data for selected increment(s) are read from BAM1.DAT and written into BAM2.DAT;
2. SORT FILE OF BOXES
allows the user to sort the candidate data base by length-width or height (see figure 1a, and note that user responses are circled). This facilitates the choice process (particularly when coupled with option 4 in the menu). The sort is two phase (i.e., if the primary sort is by height, the secondary sort is by length-width within height, and visa-versa). A frequency analysis of the box dimensions is also provided on screen (figure 1a). The sort can be performed on BAM2.DAT,...,BAM4.DAT;
3. SELECT SUBSET OF BOXES FROM FILE
allows the user to select from BAM2.DAT (BAM3.DAT) and place the desired boxes in BAM3.DAT (BAM4.DAT). Selection is made by inputting a lower bound and upper bound on the desired dimension (length-width or height) for the selection process. An abbreviated frequency analysis of the box dimensions is provided on screen (figure 1b) to facilitate the choice process;
4. VIEW A "BAMS" FILE
allows the user to view the BAM1.DAT,...,BAM4.DAT, and PACK.DAT files (figures 2a and 2b);
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
allows the user to eliminate box(es), to move box(es) from BAMx.DAT to BAMx+1.DAT, or change orientation of box(es) (figures 3a, 3b, and 3c);
6. STACK BOXES PRIOR TO PALLET LOADING
allows the user to create a file of "stacked" boxes (figure 4). The boxes are stacked, using a Dynamic Programming "knapsack" code [8], with the objective of maximizing the cube filled over a "base" box. The "stacks" can then be loaded using the Pallet Loading Procedure.
9. STOP
terminates the Data Manipulator.

CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A "BAMS" FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP

(1/2/3/4/5/6/9)? 2

WANT "BAM2","BAM3",OR "BAM4" (2,3,4) ? 3

PRIMARY SORT BY "LTH-WTH", OR "HGT" (L,W,H) ? L

SECONDARY SORT BY "LTH-WTH","HGT",OR NONE (L,W,H,N) ? H

LENGTH-WIDTH-HEIGHT ANALYSIS

L,W,H	NO#	WEIGHT	AREA-FT	CUBE-FT	L,W,H	NO#	WEIGHT	AREA-FT	CUBE-FT
3	1	90.	11.	3.	32	2	170.	3.	[illegible]
4	3	125.	22.	7.	33	23	4659.	16[illegible].	[illegible]
8	1	50.	1.	1.	34	3	215.	[illegible].	[illegible]
10	4	152.	34.	28.	35	19	1100.	21.	[illegible]
11	5	382.	23.	21.	36	5	307.	17.	92.
12	1	89.	12.	12.	37	7	1125.	19.	59.
13	4	220.	6.	7.	38	4	704.	34.	10[illegible].
14	1	130.	8.	10.	40	1	50.	0.	1.
15	8	1019.	29.	36.	42	1	52.	2.	5.
16	14	1932.	55.	73.	43	33	3165.	75.	270.
17	27	2730.	114.	161.	46	4	1000.	11.	40.
18	45	4933.	241.	361.	47	6	1200.	15.	59.
19	42	5178.	223.	353.	48	2	415.	9.	36.
20	23	3067.	112.	187.	51	3	300.	12.	53.
21	11	1680.	50.	88.	52	1	130.	2.	10.
22	6	600.	80.	146.	56	2	760.	11.	52.
23	1	130.	5.	10.	57	1	267.	9.	41.
24	6	535.	38.	76.	58	6	600.	30.	146.
25	4	625.	42.	86.	61	1	215.	4.	22.
26	1	75.	7.	15.	67	2	2120.	31.	174.
27	1	75.	3.	7.	68	2	2120.	31.	174.
28	1	400.	4.	10.	73	1	50.	1.	5.
29	1	50.	2.	5.	74	1	55.	2.	14.
30	2	190.	9.	24.	92	1	89.	2.	12.
31	12	1484.	27.	70.	0	0	0.	0.	0.
TOTAL	126	17431.		1388.					

Figure 1a

```
CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A 'BAMS' FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                   [1/2/3/4/5/6/9]? 3

READ FROM 'BAM2', WRITE TO 'BAM3', OR
READ FROM 'BAM3', WRITE TO 'BAM4' [2/3]? 3

CHOOSE OPTION

1. SELECT ALL DATA
2. SELECT BY HEIGHT
3. SELECT BY LENGTH-WIDTH
9. RETURN
                                   [1,2,3,9]? 3

 L,W,H   NO#      L,W,H   NO#      L,W,H   NO#      L,W,H   NO#
   10     4        18     18        21     2        43     33
   17    11        19     29        36     2         0      0

CHOOSE LOWER BOUND 17

CHOOSE UPPER BOUND 19

CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A 'BAMS' FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                   [1/2/3/4/5/6/9]? 9

TT1  --  STOP
```

Figure 1b

```
CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A 'BAMS' FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                        [1/2/3/4/5/6/9]? 4

WANT 'BAM1','BAM2','BAM3','BAM4', OR 'PACK'
                                        [1/2/3/4/7]? 4
```

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
B001	006	8	17	19	43	200		0
B001	013	8	17	19	43	200	C	0
B001	008	8	17	19	43	125		0
B001	009	8	17	19	43	125		0
B001	010	8	17	19	43	125		0
B001	011	8	17	19	43	125		0
B001	012	8	17	19	43	100		0
B001	007	8	17	19	43	85		0
I041	002	9	18	19	43	95		0
I041	003	9	18	19	43	95		0
I041	004	9	18	19	43	95		0
I041	005	9	18	19	43	95		0
I041	006	9	18	19	43	95		0
I041	007	9	18	19	43	95		0
I041	008	9	18	19	43	95		0
I041	009	9	18	19	43	95		0
I041	010	9	18	19	43	95		0
I041	011	9	18	19	43	95		0
I041	012	9	18	19	43	95		0
I041	013	9	18	19	43	95		0

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
I041	014	9	18	19	43	95		0
I041	015	9	18	19	43	95		0
I041	016	9	18	19	43	95		0
I041	017	9	18	19	43	95		0
I041	018	9	18	19	43	95		0
I041	019	9	18	19	43	67		0
I041	020	8	17	19	43	100		0
I041	021	8	17	19	43	100		0
I041	022	8	17	19	43	46		0

Figure 2a

CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A "BAMS" FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP

[1/2/3/4/5/6/9]? 4

WANT "BAM1","BAM2",BAM3","BAM4", OR "PACK"
[1/2/3/4/7]? 7

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE	LOC L W	LEVEL IN.
B001	006	8	17	19	43	200		0	3600	0
B001	013	8	17	19	43	200	C	0	5300	0
B001	008	8	17	19	43	125		0	7000	0
B001	009	8	17	19	43	125		0	8700	0
B001	010	8	17	19	43	125		0	19	0
B001	011	8	17	19	43	125		0	1719	0
B001	012	8	17	19	43	100		0	3419	0
B001	007	8	17	19	43	85		0	5119	0
I041	002	9	18	19	43	95		0	0	0
I041	003	9	18	19	43	95		0	1800	0
I041	004	9	18	19	43	95		0	6819	0
I041	005	9	18	19	43	95		0	8619	0
I041	006	9	18	19	43	95		0	38	0
I041	007	9	18	19	43	95		0	1938	0
I041	008	9	18	19	43	95		0	3838	0
I041	009	9	18	19	43	95		0	5738	0
I041	010	9	18	19	43	95		0	7638	0
I041	011	9	18	19	43	95		0	56	0
I041	012	9	18	19	43	95		0	1856	0
I041	013	9	18	19	43	95		0	3656	0

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE	LOC L W	LEVEL IN.
I041	014	9	18	19	43	95		0	5456	0
I041	015	9	18	19	43	95		0	7256	0

Figure 2b

```
CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A "BAMS" FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                        [1/2/3/4/5/6/9]? 5

CHOOSE OPTION

1. ELIMINATE A BOX
2. COPY A BOX FROM BAM*.DAT TO BAM*1.DAT
3. ROTATE SINGLE BOX, LTH TO HGT OR WTH TO HGT
4. ROTATE BOXES FOR COMMON HEIGHT
9. RETURN
                                        [1/2/3/4/9]? 1

MANIPULATE BAM2, BAM3, OR BAM4 [2,3,4] ?4

    INCR   ITEM   CUBE    LTH    WTH    HGT   WEIGHT   H   STACK
     NR     NR    FT.     IN.    IN.    IN.    LBS.    Z   CODE
 1. B001   006      8      17     19     43     200         0
 2. B001   013      8      17     19     43     200     C   0
 3. B001   008      8      17     19     43     125         0
 4. B001   009      8      17     19     43     125         0
 5. B001   010      8      17     19     43     125         0
 6. B001   011      8      17     19     43     125         0
 7. B001   012      8      17     19     43     100         0
 8. B001   007      8      17     19     43      85         0
 9. I041   002      9      18     19     43      95         0
10. I041   003      9      18     19     43      95         0
11. I041   004      9      18     19     43      95         0
12. I041   005      9      18     19     43      95         0
13. I041   006      9      18     19     43      95         0
14. I041   007      9      18     19     43      95         0
15. I041   008      9      18     19     43      95         0

ENTER NUMBER OF BOX TO BE ELIMINATED OR 0 TO CONTINUE 2
  1 BOXES ELIMINATED
 28 BOXES REMAIN IN BAM4.DAT


MORE ELIMINATIONS [Y/N]?
N
```

Figure 3a

```
CHOOSE OPTION

1. ELIMINATE A BOX
2. COPY A BOX FROM BAM*.DAT TO BAM*+1.DAT
3. ROTATE SINGLE BOX, LTH TO HGT OR WTH TO HGT
4. ROTATE BOXES FOR COMMON HEIGHT
9. RETURN
                                   [1/2/3/4/9]? 2

COPY FROM "BAM2" TO "BAM3" OR
COPY FROM "BAM3" TO "BAM4" [2/3]? 3

    INCR   ITEM   CUBE    LTH    WTH    HGT   WEIGHT  H  STACK
     NR     NR     FT.    IN.    IN.    IN.    LBS.  Z  CODE
 1. B001   015       9     10     36     43      46        0
 2. B001   017       9     10     36     43      46        0
 3. B001   006       8     17     19     43     200        0
 4. B001   013       8     17     19     43     200     C  0
 5. B001   008       8     17     19     43     125        0
 6. B001   009       8     17     19     43     125        0
 7. B001   010       8     17     19     43     125        0
 8. B001   011       8     17     19     43     125        0
 9. B001   012       8     17     19     43     100        0
10. B001   007       8     17     19     43      85        0
11. EA02   012       5     10     21     43      30        0
12. EB02   012       5     10     21     43      30        0
13. I041   002       9     18     19     43      95        0
14. I041   003       9     18     19     43      95        0
15. I041   004       9     18     19     43      95        0

ENTER NUMBER OF BOX TO BE COPIED FROM "BAM3" TO "BAM4"
OR 0 TO CONTINUE 4

    INCR   ITEM   CUBE    LTH    WTH    HGT   WEIGHT  H  STACK
     NR     NR     FT.    IN.    IN.    IN.    LBS.  Z  CODE
 4. B001   013       8     17     19     43     200     C  0

  1 BOXES COPIED INTO "BAM4"

COPY ANOTHER BOX [Y/N]? N
```

Figure 3b

```
CHOOSE OPTION

1. ELIMINATE A BOX
2. COPY A BOX FROM BAM*.DAT TO BAM*+1.DAT
3. ROTATE SINGLE BOX, LTH TO HGT OR WTH TO HGT
4. ROTATE BOXES FOR COMMON HEIGHT
9. RETURN
                                   [1/2/3/4/9]? 4

MANIPULATE BAM2, BAM3, OR BAM4 [2,3,4] ?4

ENTER UPPER BOUND ON COMMON HEIGHT 19

ENTER LOWER BOUND ON COMMON HEIGHT 12

 29 BOXES IN "BAM4" HAVE BEEN ROTATED


DO YOU WISH TO VIEW "BAM4" [Y/N]? N

CHOOSE OPTION

1. ELIMINATE A BOX
2. COPY A BOX FROM BAM*.DAT TO BAM*+1.DAT
3. ROTATE SINGLE BOX, LTH TO HGT OR WTH TO HGT
4. ROTATE BOXES FOR COMMON HEIGHT
9. RETURN
                                   [1/2/3/4/9]? 9

CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A "BAMS" FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                   [1/2/3/4/5/6/9]? 9

TT1  --  STOP
>
```

Figure 3c

```
CHOOSE OPTION

1. IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A *BAMS* FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STACK BOXES PRIOR TO PALLET LOADING
9. STOP
                                   [1/2/3/4/5/6/9]? 6

WHICH FILE IS TO BE STACKED [2/3]? 3
```

	INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
1	B001	015	9	10	36	43	46		0
2	B001	017	9	10	36	43	46		0
3	B001	006	8	17	19	43	200		0
4	B001	013	8	17	19	43	200	C	0
5	B001	008	8	17	19	43	125		0
6	B001	009	8	17	19	43	125		0
7	B001	010	8	17	19	43	125		0
8	B001	011	8	17	19	43	125		0
9	B001	012	8	17	19	43	100		0
10	B001	007	8	17	19	43	85		0
11	EA02	012	5	10	21	43	30		0
12	EB02	012	5	10	21	43	30		0
13	I041	002	9	18	19	43	95		0
14	I041	003	9	18	19	43	95		0
15	I041	004	9	18	19	43	95		0

```
WANT ONE OF THESE FOR BASE?
IF SO, INPUT NUMBER, OTHERWISE HIT *RETURN* 13

INPUT MAXIMUM PALLET HEIGHT (INCHES) 96
```

	INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
14	I041	003	9	18	19	43	95		0
13	I041	002	9	18	19	43	95		0

```
WANT TO PUT STACK ON BAM4.DAT [Y/N]? Y
```

Figure 4

The main option list of the Pallet Loading Procedure contains five options:

1. LOAD PALLET
Allows the user to load the pallet using the set of candidate boxes. This represents a forward pass on the Pallet Loader. The Pallet Loader is structured as a Dynamic Program [8] and solves a two-dimensional loading problem (i.e., loads one layer of boxes or stacks of boxes, on the pallet) (figures 5,6);
2. GET PALLET LOAD SOLUTION
Allows the user to fetch the solution generated by the DYNALOAD routine. This represents a backward pass on the Pallet Loading DP. A list of loaded boxes and their locations (x,y coordinates) is provided. A plan view picture of the boxes loaded on the pallet is provided as an option. (figure 6);
3. PUT PACKED BOXES ON PACK.DAT
Allows the user to remove the "loaded" boxes from the BAM..DAT currently being used and places them in the PACK.DAT file. The user can specify the level (height) that the boxes are to be placed on the pallet. This facilitates multiple passes of the packing routine while loading a pallet in "layers";
4. CRUNCH LOOSE BOXES IN SOLUTION TOWARD ORIGIN
Allows the user to "tighten" a pallet load solution by shifting boxes toward the upper left-hand corner (plan view) of the pallet as much as possible. A plan view picture of the boxes loaded on the pallet is provided as an option; and
9. STOP
terminates the Pallet Loading Procedure.

The following describes a way of using the IFLS system. Assume that the BAMS.DAT file contains a list of equipment appropriate to the planned mission. If only a partial list is to be used, the system "editor" can be used to delete specific items from the file. Then the "READIT" program (see appendix) can be used to load the BAM1.DAT file from BAMS.DAT.

One approach to loading (using the system) would be, first, to identify a set of candidate boxes for loading and then to load the pallet(s) with boxes from the candidate list.

The first step in identifying a set of candidate boxes is to use the IDENTIFY THE LIST OF ITEMS option in the Data Manipulator to select the squadron increment(s) to be loaded on the pallet(s). This results in a subset of the BAMS.DAT boxes being stored in the BAM2.DAT file. Next, the boxes should be sorted using the SORT THE FILE OF BOXES option. By examining the frequency analysis of box dimensions (provided as part of the sort, figure 1a), the user starts to get an overall feeling as to the characteristics of the set of candidate boxes. Then using the VIEW A 'BAMS' FILE option (figure 2a) the user can further enhance his knowledge of the (sorted) candidate set. At this point the user can take several different approaches to pallet loading. The following outlines three possible approaches.

1. Attempt to select a set of boxes with a common height. This will allow the user to load a layer of boxes on the pallet (using the pallet loading procedure). Then he/she can select another set of boxes and load another layer until the maximum pallet height is reached (figure 8a).

```
CHOOSE OPTION

   IDENTIFY LIST OF ITEMS
2. SORT FILE OF BOXES
3. SELECT SUBSET OF BOXES FROM FILE
4. VIEW A "BAMS" FILE
5. MANIPULATE SPECIFIC BOX(ES) IN FILE
6. STORE BOXES PRIOR TO PALLET LOADING
7. EXIT
                                   [1/2/3/4/5/6/7]: 4

NAME "BAM1","BAM2",BAM3","BAM4", OR "PACK"
                                   [1/2/3/4/7]: 3
```

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
B001	017	9	10	36	43	46		0
B001	006	8	17	19	43	200		0
B001	013	8	17	19	43	200	C	0
B001	008	8	17	19	43	125		0
B001	009	8	17	19	43	125		0
B001	010	8	17	19	43	125		0
B001	011	8	17	19	43	125		0
B001	012	8	17	19	43	100		0
B001	007	8	17	19	43	85		0
EB02	012	5	10	21	43	30		0
I041	009	9	18	19	43	95		0
I041	010	9	18	19	43	95		0
I041	011	9	18	19	43	95		0
I041	012	9	18	19	43	95		0
I041	013	9	18	19	43	95		0
I041	014	9	18	19	43	95		0
I041	015	9	18	19	43	95		0
I041	016	9	18	19	43	95		0
I041	017	9	18	19	43	95		0
I041	018	9	18	19	43	95		0

INCR NR	ITEM NR	CUBE FT.	LTH IN.	WTH IN.	HGT IN.	WEIGHT LBS.	H Z	STACK CODE
I041	019	9	18	19	43	67		0
I041	020	8	17	19	43	100		0
I041	021	8	17	19	43	100		0
I041	022	8	17	19	43	46		0
1041	014	9	18	19	43	95		0

Figure 5

```
 RUN LOADIT

WHICH FILE: BAM1? BAM2? BAM3? BAM4? [1/2/3/4] 3

ZERO PACK.DAT? [Y/N] Y

INPUT LENGTH 60

INPUT WIDTH 50

CHOOSE OPTION

1. LOAD PALLET
2. GET PALLET LOAD SOLUTION
3. PUT PACKED BOXES ON PACK.DAT
4. CRUNCH LOOSE BOXES IN SOLUTION TOWARD ORIGIN
9. STOP
                [1/2/3/4/9] 1

PURE RECTANGLE [Y/N] Y

INPUT LENGTH 60

INPUT WIDTH 50

CHOOSE OPTION

1. LOAD PALLET
2. GET PALLET LOAD SOLUTION
3. PUT PACKED BOXES ON PACK.DAT
4. CRUNCH LOOSE BOXES IN SOLUTION TOWARD ORIGIN
9. STOP
                [1/2/3/4/9] 2

        LOCATION
  BOX#     L W     LTH     WTH
     1      36      36      10
    10    3636      21      10
    11       0      19      18
    12    1900      19      18
    13    3800      19      18
    14      18      19      18
    15    1918      19      18
    16    3818      19      18

WANT PICTURE [Y/N] Y
```

Figure 6

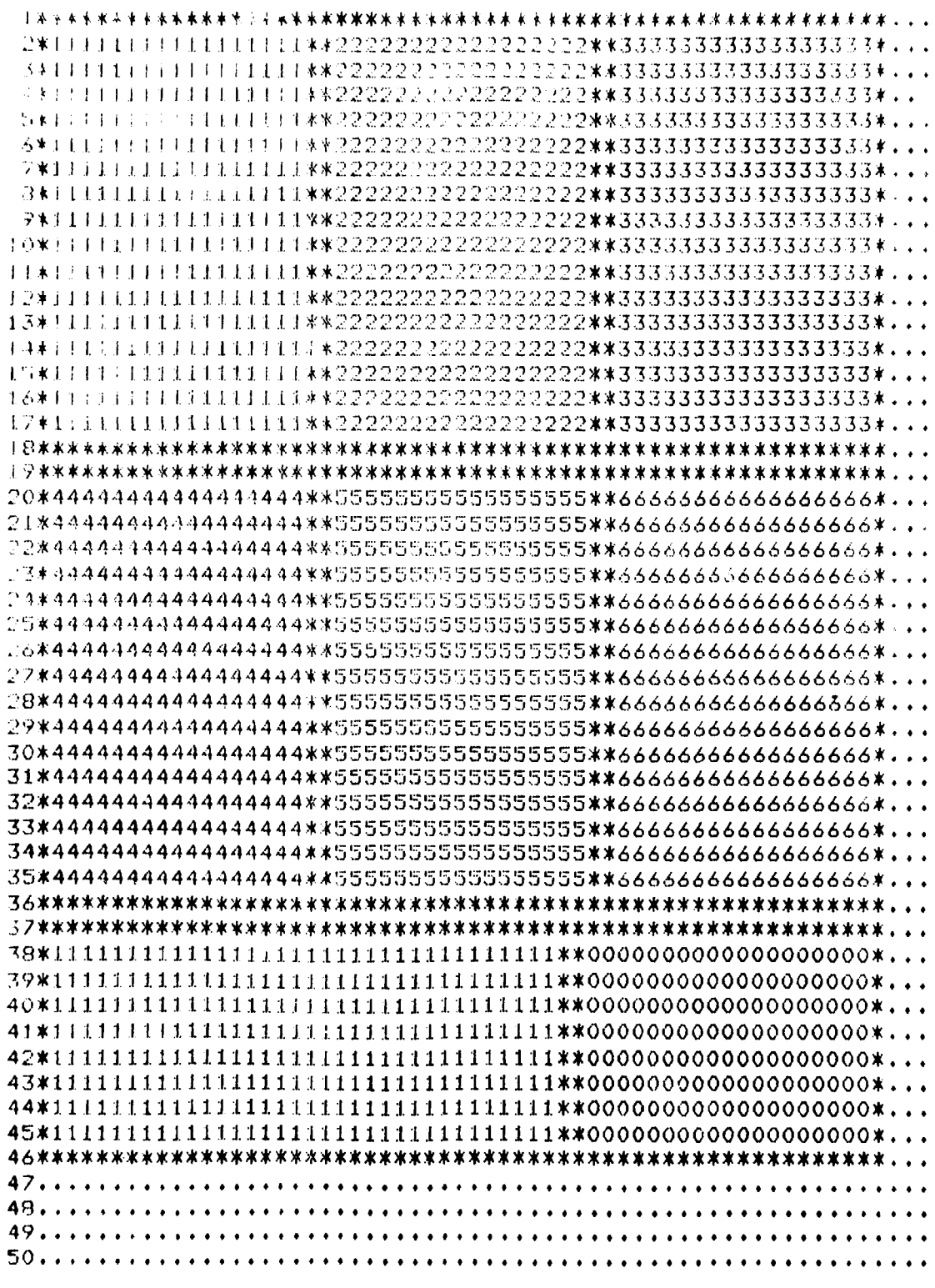


Figure 7

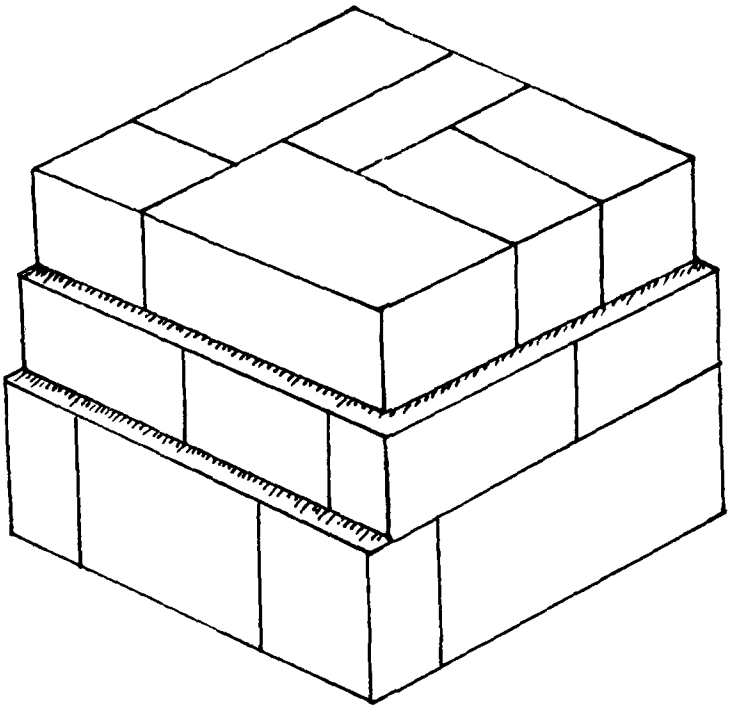


Figure 8a

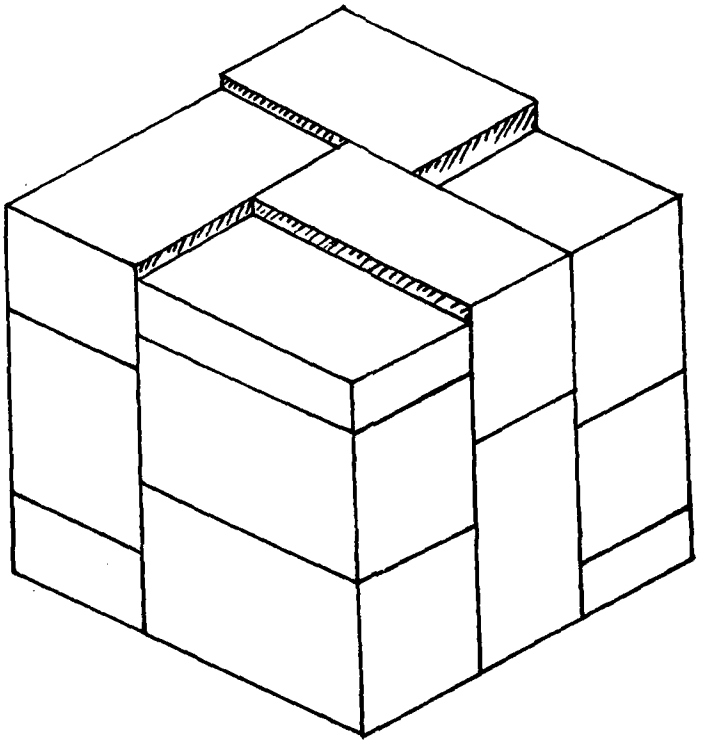


Figure 8b

2. Attempt to select a set of the larger boxes without regard for common dimensions. Then load the pallet. Since the tops of the boxes probably will not be the same height, it will be necessary to define the tops of each of the boxes (or several boxes of common height) as "pallets" and proceed to load boxes on the tops of the "pallets". This is easily done within the structure of the DYNALOAD routine since both length and width are user input and "L" shaped pallets can be specified. Combinations of this and the previous approach can be used to complete loading of the pallet.
3. Select sets of boxes using the STACK BOXES PRIOR TO PALLET LOADING option. Then use the Pallet Loading Procedure to position the stacks on the pallet (figure 8b).

Pallet Loading Algorithm

The pallet loading procedure used in IPLS can be described as a combination of the principles of Dynamic Programming (D.P.) and heuristics. A serendipitous by-product of the structure of this combination is that positioning of hazardous material and considerations of center of gravity can be handled without loss within the procedure. To understand the procedure, it is useful first to consider a "best" 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, Dynamic Programming will be used. The following definitions will be useful.

- P = A partition dividing the pallet into two parts (see figure 9). The left-hand sub-pallet must include the origin (0,0), and the right-hand sub-pallet must include the point (L,W).
- 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:

$$(1) \quad f(P,I) = \max_{i \in I} [l(i)*w(i) + f(P-S(i),I-i)].$$

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

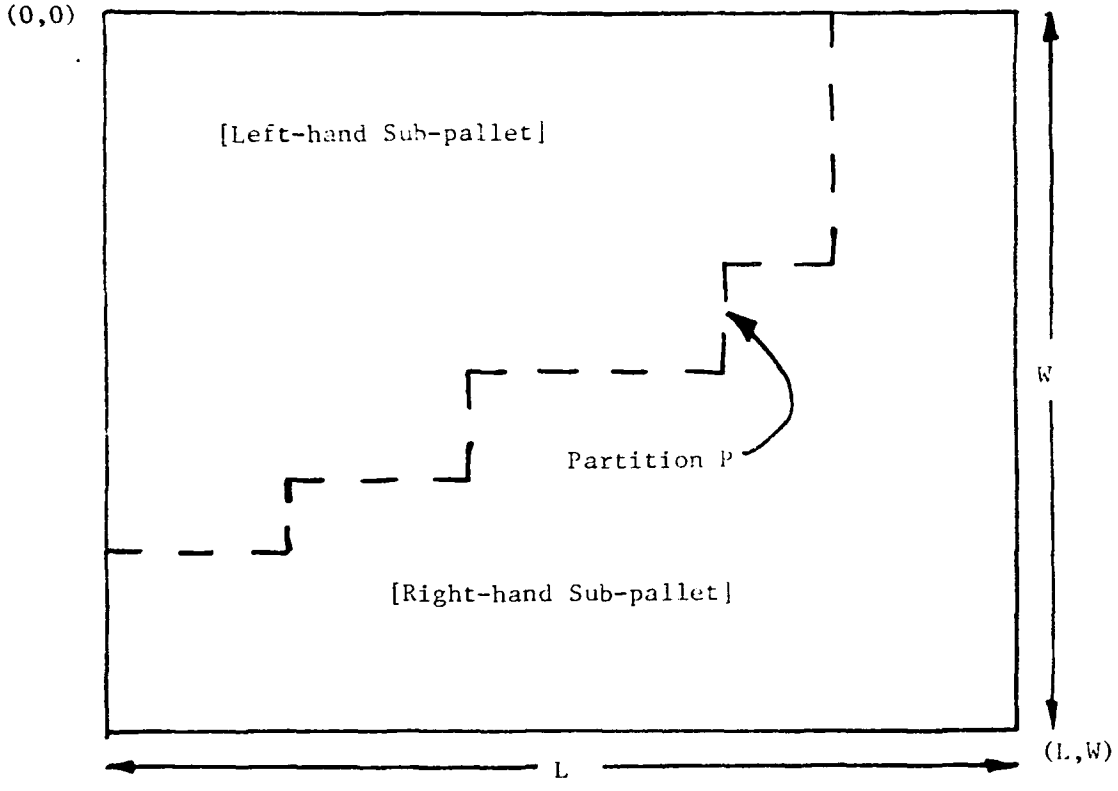


Figure 9 .

Plan View of Pallet With
Sample Partition P

implementing equation (1). The main obvious is that the number of possible partitions P of the pallet is extremely large. This means that the state space for the D.P. will require large amounts of computer storage. It also means that the computer time required to solve the D.P. likely would be well beyond any sensible limit for real world applications.

One approach to developing a more efficient 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 10a). In order to specify the Dynamic Program resulting from the rectangular partitions, the following additional definitions are needed.

- x,y = Two-dimensional index specifying a rectangular partition (figure 6a).
- J = Subset of the boxes, 1,2,...,n.
- g(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 10b).

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

$$(2) \quad g(x',y',I) = \max_{\substack{x\le x' \\ y\le y' \\ J\in I}} [g(x,y,J) + h(x,y,x',y',I-J)]$$

The implementation of equation (2) also has its difficulties. The function h(x,y,x',y',I) 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., the cross-hatched area in figure 10b). 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 was decided to carry only one partial solution (g(x,y,I)) for each partition x,y. The obvious choice is to carry

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

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); and
2. bounding rules for the elimination of partial solutions.

The optimization for h(x,y,x',y',I) is done simply by breaking the L-shaped area into two rectangular areas (figure 11) and filling each area useing a simple D.P. procedure. The following definition is useful.

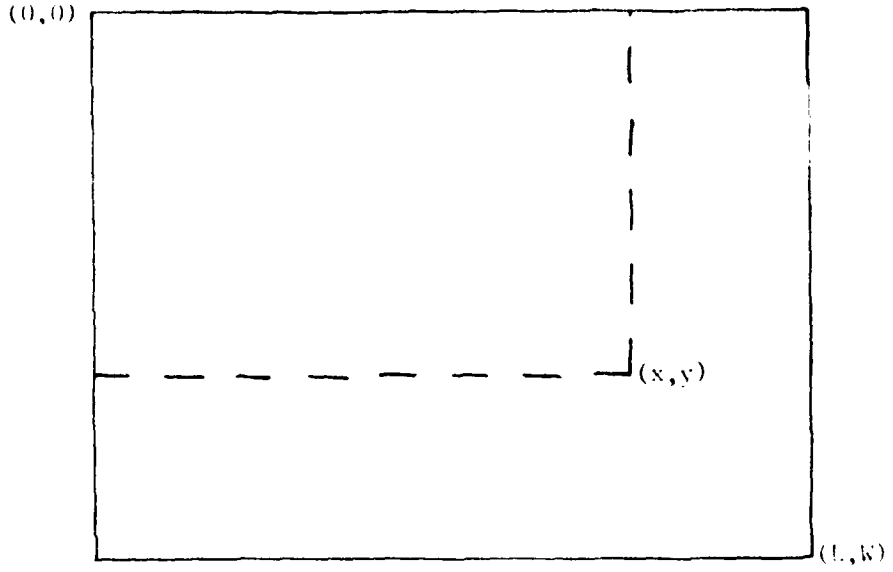


Figure 10a
Pallet With Rectangular Partition (x,y)

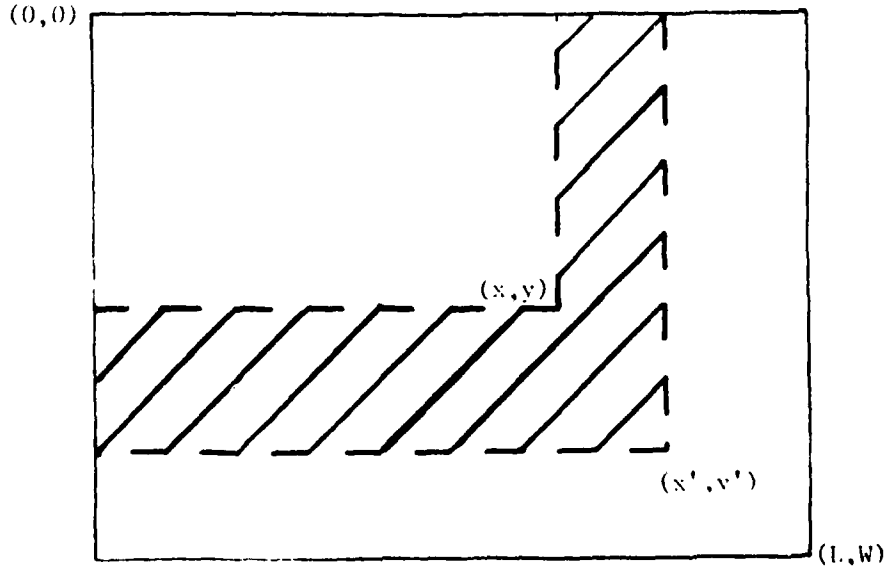
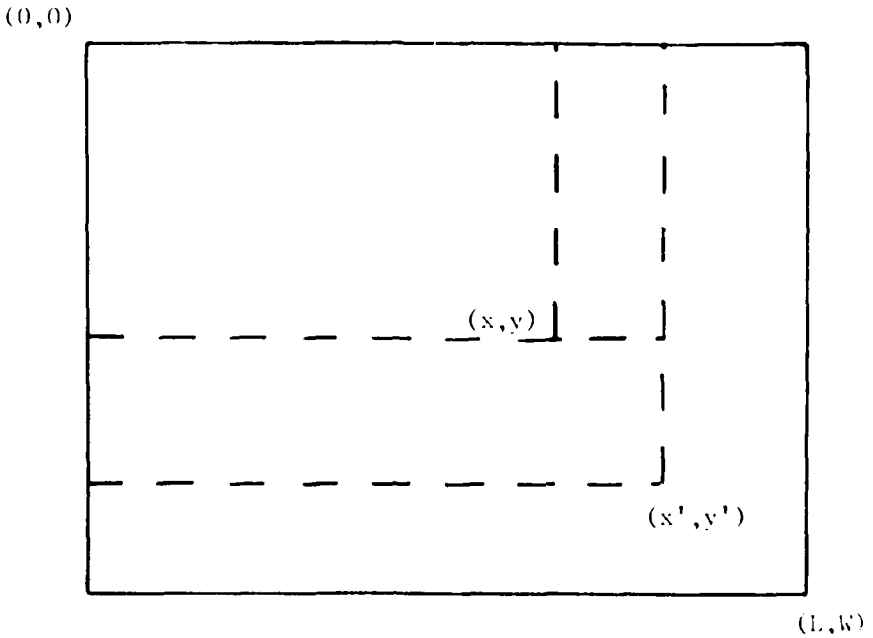


Figure 10b
Pallet With Two Rectangular Partitions
(x,y) and (x',y')



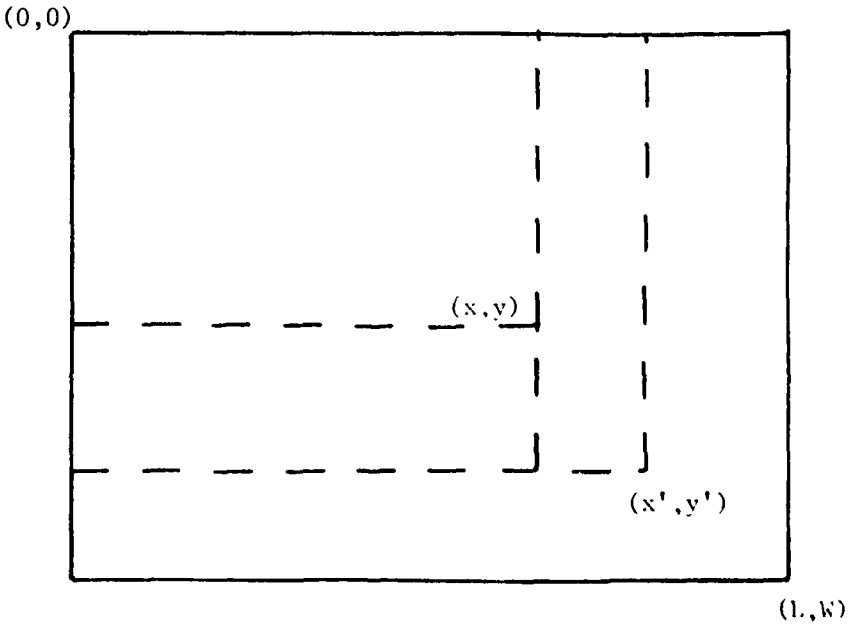


Figure 11
Two Ways To Break Up L-Shaped Area

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

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

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

The implementation of equation (3) is achieved by first 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 11) for the application of equation (3). The algorithm retains the best solution obtained.

A simple, and almost obvious, bounding rule that eliminates a great deal of the storage requirements for the state space 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', (x'<=x, y'<=y) such that the solution value for x',y' (max [s(x',y',I)]) is greater than or equal to the solution value for x,y (max[s(x,y,I)]).

Another effective bounding scheme is used to eliminate computation time. 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,...,j.

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

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

Each box is considered by both length and width as a candidate for insertion. 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,...,n. For a pallet (rectangular sub-pallet) of size L by W, an upper bound on the maximum load (coverage) possible is c(n,L)*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.

Since 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. Second, again 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.

Comments on Implementation

In order to implement the IPLS system in the USAF operating environment, there are several requirements that must be met. First, there are two elements of data that must be added to the BAMS data set; this-end-up information, and stacking capability. Observations of USAF personnel while packing pallets indicate that many items can be packed any-end-up while other items must be packed this-end-up. Adding an indicator for this to the BAMS data base is imperative to any automated packing scheme. There are many items that cannot have anything placed on top of them due to either strength or shape considerations. Further study may indicate that it is necessary for actual strength estimates to be made. However, it is our observation that the vast majority of items are either "stackable" or not. With all this in mind, an additional parmeter of the following form would be appropriate for the BAMS data:

0 - Any-End-Up and Stackable;
1 - This-End-Up and Stackable;
2 - Any-End-Up and Non-Stackable; and
3 - This-End-Up and Non-Stackable.

Second, there is certain equipment that would be most useful. The present system was developed using a PDP-11/34 with the interface instrument being either a "dumb" CRT terminal or a hard copy terminal. This equipment reasonably duplicates the computing power (B-3500) and peripheral equipment presently available at USAF base level. However, Computer equipment has revolutionized in the last ten years. Due to its limited computing power, and limited directly addressable memory, the PDP 11/34 is marginally adequate for the job. Test problems run on the system are solved in anywhere from a few seconds to several minutes. Solution times are dependent on pallet size and the input data (set of boxes to be loaded). Faster CPU speeds would be desireable.

The man-machine interface for the system would be greatly facilitated with the use of a "smart" graphics CRT terminal. Three dimensional views of the pallet (with currently loaded boxes) would give the user a great deal of insight into the loading process. It is difficult to obtain that kind of insight from the present primitive two dimensional plan layout. In addition, it would be helpful to have the three-dimensional views of the pallet in hard copy output in order to facilate the actual loading personnel in implimenting the planned load.

Finally, IPLS should be considered, at this stage, as a developmental system. It has not been totally "idiot proofed", nor is it completely "user friendly". Most of the shortcomings exist because hardware has not been specified for the real-life application. Once the appropriate hardware has been targeted for the real-life application, these shortcomings can be readily corrected.

References

[1] Christofides, N., and Whitlock, C., "An Algorithm for Two Dimensional Cutting Problems," OPERATIONS RESEARCH, Vol. 25, No. 1, Jan. 1977, pp. 30-44.

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

[3] Garey, Michael, and Johnson, David, COMPUTERS AND INTRACTABILITY, W.H. Freeman, San Francisco, 1979.

[4] Gilmore, P.C., and Gomory, R.E., "Multistage Cutting Stock Problems of Two and More Dimensions," OPERATIONS RESEARCH, Vol. 13, No. 1, Jan. 1965, pp.94-120.

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

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

[7] Herz, J.C., "A Recursive Computing Procedure For Two-Dimensional Stock Cutting," IBM JOURNAL OF RESEARCH AND DEVELOPMENT, Vol. 16, 1972, pp. 462-469.

[8] Nemhauser, George L., INTRODUCTION TO DYNAMIC PROGRAMMING, John Wiley and Sons, Inc., New York, NY, 1966.

[9] 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.