Development of 3D Packing Simulator

Masaaki MINAGAWA, Ryoji KANAZAWA
and
Yukinori KAKAZU

Abstract

This paper describes an approach to build an expert system of 3D (three dimensional) packing. Given different sizes, weights and other special attributes of boxes to be packed, the system automatically allocates spaces of containers to each box so that the least amount of empty space is left. To realize the allocation, a combinatorial problem of 3D geometry is solved by taking a heuristic approach. Obtained results of the allocation are shown as graphic outputs such as perspective views and top views which are drawn layer by layer from bottom to top layer.

1. Introduction

Packing a number of various sizes and weights of boxes into shipping containers is a continual challenge and the determination of box location in the container has traditionally required human experts with many years of experience. And, as the types and the number of shipping items increase, there will be a limit to human capability in this time-consuming work and error process.

Due to difficulties in implementing such experience, we have not arrived at a good solution until today. The 3D packing simulator (hereafter the simulator) aims at improving this situation by realizing automatic calculation of this 3D space allocation. When different sizes, weights and other special attributes of boxes to be packed are given, the simulator automatically determines box locations and orientations so that the least amount of empty space is left after selective allocations. During the determination, for practical reasons, packing conditions on the orientations and positions of boxes are taken into consideration. Users can check the result of the calculation using graphic outputs such as perspective views and top views which are drawn layer by layer from the bottom to the top layer individually.

If we consider the use of the simulator in highly automated environment, it can be used to construct an automated shipping system such as shown in Fig. 1. To realize realtime simulation, we adopted heuristic rules which are expected to reduce calculation time of the combinatorial problem of 3D geometry.

In this paper, 1) an outline of the packing simulation, 2) description of the problem solved by the simulator, 3) employed approach method to solve the

Faculty of Engineering, Hokkaido University Kita 13-Jo, Nishi 8-Chome Kita-Ku Sapporo, 060 Japan.

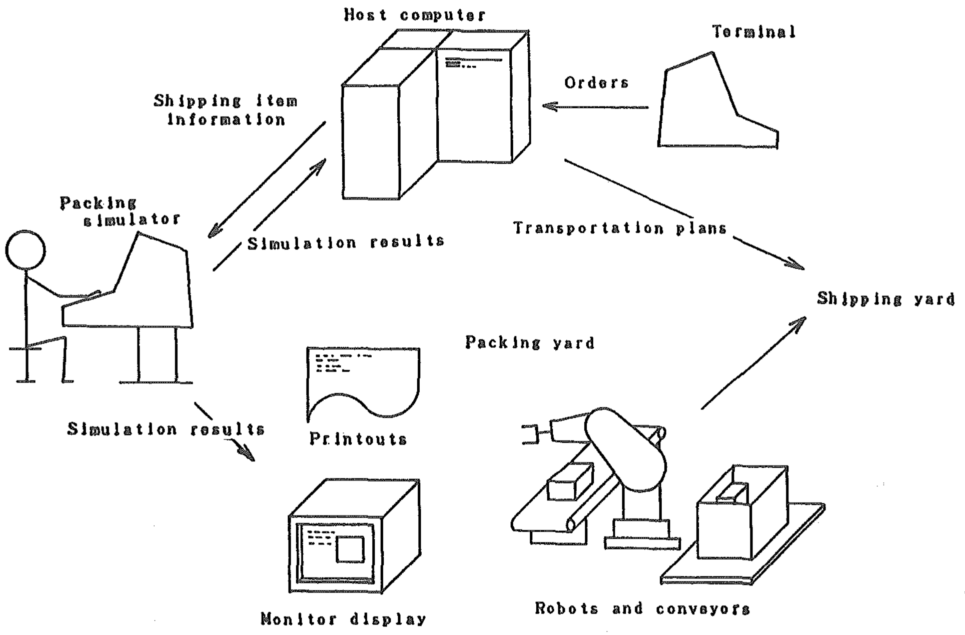

Fig. 1 An example of automated shipping system with the packing simulator

problem, 4) an outline of the developed system 5) some graphic outputs of obtained results and required CPU time are explained.

2. An outline of the packing simulation

Fig. 2 shows an outline of the packing simulation using the developed simulator. In the system, two types of inputs are defined···box data and container data. The box data specifies information regarding boxes to be packed. The data involves box dimensions (widths, lengths and heights), weights, required packing conditions and other attributes such as product names.

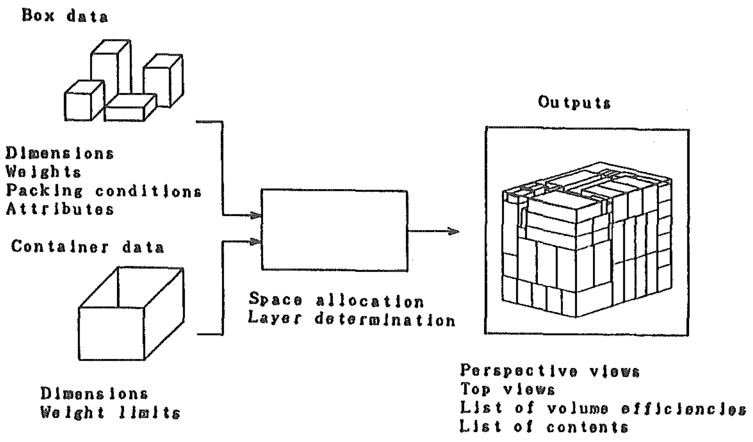

Fig. 2 Packing simulation by the simulator

The container data specifies information about shipping containers. This data has dimensions of containers and weight limits for packing. Using the information above, the simulator executes calculation of 1) space allocation of the containers to the boxes and 2) layer determination of packed boxes.

After this calculation, graphic outputs such as perspective views and top views, list of volume efficiencies and list of contents of packed containers are obtained on a CRT or a printer. Users can check the calculation results using these outputs.

3. The descpription of the solved problem

The developed system solves the space allocation problem which can be considered as a combinatorial problem of 3D geometry. This problem will be forumulated and explained below.

3-1 Terms and symbols

First, we define terms and symbols to be used in the following descriptions. The definitions are given by using predicate logic and we use predicates and functions of the following form

Predicates: PRED $(x_1, x_2, ..., x_n)$ ··(1)

Functions : FUNC $(x_1, x_2, ..., x_n)$ ··(2)

where PRED represents a predicate symbol and FUNC represents a funtion symbol, and $x_1, x_2, ..., x_n$ are variables or conditions. Each predicate has T (true) or F (false) value and each function will return a corresponding value when their variables are fixed. The list of defined predicates and functions are shown in Table-1.

1) B : boxes before allocation

B denotes a set of boxes to be packed into containers. Each box is identified

Predicates	Conditions to be T
DEFSET(A, B, C)	The set A of B is defined under condition C
MEMBER(A, B)	A is a member of the set C
SUBSET(A, B)	A is a sebset of the set B
DET(A)	A is determined
EXIST(A, B)	A exists under condition B
EQ(N, M)	N is equal to M
NE(N, M)	N is not equal to M
GT(N, M)	N is greater than M
LE(N, M)	N is less than or equal to M

Table 1(a) Predicates

Funcions	Return value
SUM(M)	summation of M
MAXIM(A)	maximum value of A
INTSEC(A, B)	the intersection of A and B
SQRT(A)	square root of A

Table 1(b) Functions

Tale 1 The list of predicates and functions by an integer value between 1 and n.

DEFSET (I , (1, 2, ..., n), NULL) ……(3)

DEFSET (B, b_i, NULL) ……(4)

MEMBER (i, I) ……(5)

b_i : = (bx_i, by_i, bz_i, w_i, pc_i, a_i) ……(6)

bx_i, by_i, bz_i : width, length and height of box i

w_i : weight of box i

pc_i : packing condition of box i

a_i : attribute information of box i

(NULL : null set)

2) C : sizes of containers

C denotes a set of containers in which boxes selected from the set B will be packed.

DEFSET (J, (1, 2, ..., m), NULL) ……(7)

DEFSET (C, c_j, NULL) ……(8)

MEMBER (j, J) ……(9)

c_j : = (cx_j, cy_j, cz_j) ……(10)

cx_j, cy_j, cz_j : width, length and height of container j

3) S : space of containers

S_j is expressed as set of points in rectangular space in j-th container.

DEFSET (S_j, (x, y, z), COND1) ……(11)

COND1 : = (GE (x, 0).AND. LE (x, cx_j)) ……(12)
.AND. (GE (y, 0).AND. LE (y, cy_j))
.AND. (GE (z, 0).AND. LE (z, cz_j))

4) P : boxes after space allocation

Boxes after the allocation are defined. Since boxes can take various orientations (see Fig. 3), each box cannot necessarily take the same orientation given in B. Thus, we use px_k, py_k and pz_k in P instead of bx_i, by_i and bz_i in B.

DEFSET (P_j, p_k, NULL) ……(13)

MEMBER (k, I_j) ……(14)

SUBSET (I_j, I) ……(15)

p_k : = (px_k, py_k, pz_k, w_k, pc_k, a_k) ……(16)

5) W_{lim} : weight limit of shipping container

6) r : volume efficiency

To evaluate results of packing, we introduce the volume efficiency r. r_j defines the ratio of the summation of volumes of packed boxes to that of container j.

r_j : = SUM (v_k)／V_j ……(17)

3-2 The description of the problem

Using the terms and symbols above, we describe the problem. The problem is to select a group of boxes from B and allocate space in a container to each box so as to leave the least amount of empty space. That is, our objective is to gain high volume efficiency.

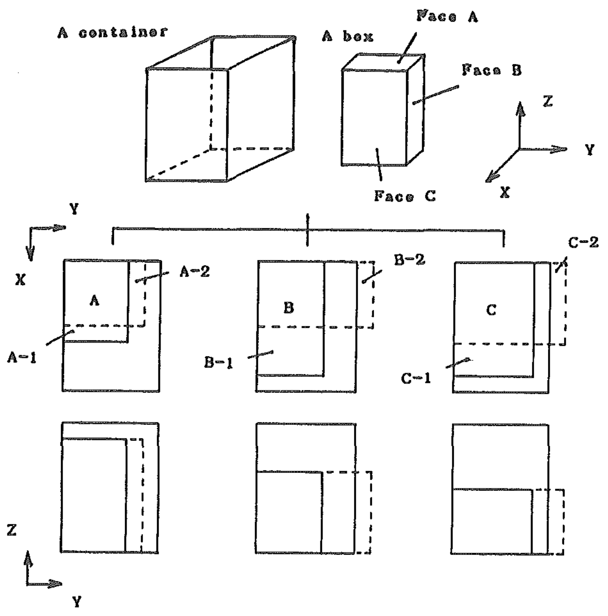

Fig. 3 Placement patterns of a box

DET (I_j) ···(18)

DET $((x_i, y_i, z_i))$ ···(19)

(x_i, y_i, z_i) : the coordinates of origin of placed box i in space Sj so as to

MAXIM (r_j) ···(20)

for all i such that MEMBER (i, I_j) under the following constraints

C1) The limit to allocation space

LE (x_i+px_i, cx_j) ···(21)

LE (y_i+py_i, cy_j) ···(22)

LE (z_i+pz_i, cz_j) ···(23)

D2) The limit to total weight

LE (SUM (w_i), W_{lim}) ···(24)

C3) Prohibition on collision between arbitrary two boxes in C_j for k such that MEMBER (k, I_j) and SUBSET (s_k, S_j), and for i such that MEMBER (i, I_j) and SUBSET (s_i, Sj) NE (i, j),

EQ (PROSET (s_k, s_i), NULL) ···(25)

C4) Prohibition on existence of a box in the air for z_i such that GT $(z_i, 0)$

EXIST (k, COND2) ···(26)

COND2 : = (GT (x_i, x_k).AND. LT (x_i, x_k+px_k))

.AND. (GT (y_i, y_k).AND. LT (y_i, y_k+py_k))

.AND. (EQ (z_i, z_k+pz_k)) ···(27)

C5) Prohibition on using side faces as bottom faces (packing condition 1).

if EQ $(p_i, 1)$ then

NE (px_k, pz_k) ···(28)

NE (py_k, pz_k) ……(29)

C6) Prohibition on placing a box at bottom layer (packing condition 2).
if EQ (p_i, 2) then
NE (z_i, 0) ……(30)

4. The approach method

The approach method basically consists of selective assignment of boxes to given allocation spaces. Given a space, a box which is considered to be the most suitable one is selected according to calcualted values of the evaluation function which is examined by checking actual packing examples given by a human expert.

4-1 Selection from feasible solutions

The number (n) of remaining box data to be processed decreases and the distribution of the data changes as the calculation proceeds. The strategy for packing is subject to change depending on this change. From this reason, in the system, we prepare different types of packing programs with different strategies to follow the change. And, to obtain the best feasible solution, we compare the results given by the programs and select the best one.

We define the group of programs as
DEFSET (PR, pr_u, NULL) ……(31)
MEMBER (u, IP) ……(32)
IP : = (1, 2, ..., n_p) ……(33)
n_p : the number of programs
Each program returns r_{ju} as one of outputs and we select u such that
MAXIM (r_j) ……(34)

4-2 An example of the packing program

We explain strategy employed in a packing program. The strategy is based on
1) selection of box using evaluation function and
2) recursive allocation space dividing.

The search space will be a tree which is expanded as the space dividing proceeds. Employed search method is depth-first search. The procedure based on this strategy is explained below.

step 1) Set an allocation space G (G : = S_j at initial condition)
step 2) Calculate a value of evalution function F_i
The evaluation function consists of four functions shown below
F_i : = SUM ($e_i * f_i$) (i = 1, 2, 3, 4) ……(35)
f1 : = $(px_i/cx_j)**2+(py_i/cy_j)**2+(pz_i/cz_j)**2$ ……(36)
f2 : = a ……(37)
f3 : = $(pz_i/D)**2$ ……(38)
D : = SQRT ($px_i**2+py_i**2$)
f4 : = (S_2/S_1) ……(39)
$S_1 = cx_j * cy_j$
$S_2 = px_i * py_i$

e_i : coefficient determined experimentally
($*$ $*$ 2 : squared)
f_1 : evaluate the differences of dimensions between b_i and G
f_2 : a constant value to give particular boxes high priority for allocation
f_3 : evaluate tall boxes
f_4 : evalute differences of bottom area between b_i and G

step 3) Select box b_i which gives MAXIM (F_i) and place it in the given space G. At this step, possible placement patterns are checked as seen in Fig. 3. When the box is to be placed at the same corner, there will appear 6 patterns (A-1 to C-2). In this example, patterns B-2, C-2 are not acceptable from space limit consideration.

step 4) Generate allocation spaces for further packing. The new spaces are obtained based on the result above. That is, as seen in Fig. 4, the space left after the box selection is divided according to a space dividing pattern. In this program, three types of allocation spaces are obtained. Type A, B and C spaces are generated by cutting the given initial space (G0) using planes perpendicular to x, y and z axes respectively.

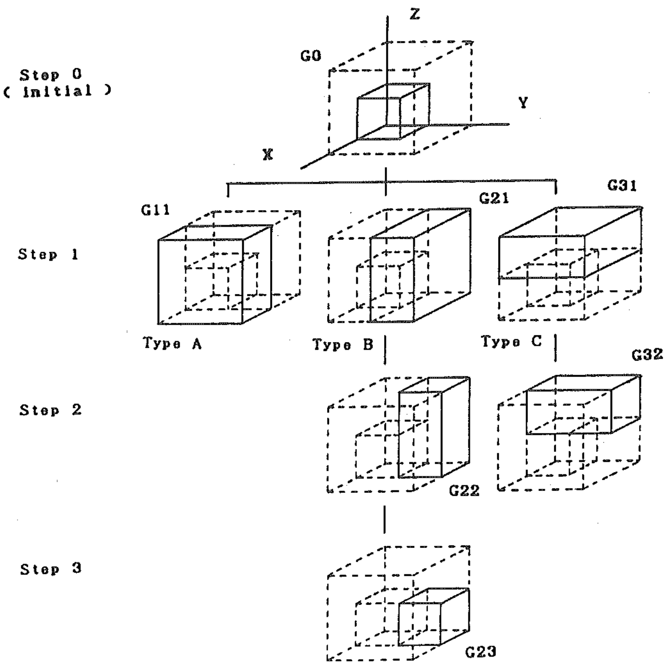

Fig. 4 An example of allocation space dividng

1) Type A

DEFSET (G11, (x, y, z), COND1) ··(40)

COND1 : = (GE (x, x_i+px_i).AND. LE (x, x_i+cx_j)
.AND. (GE (y, y_i).AND. LE (y, y_i+cy_j))
.AND. (GE (z, z_i).AND. LE (z_i, cz_j)) ·························(41)

2) Type B

DEFSET (G21, (x, y, z), COND2) ······(42)

COND2 : = (GE (x, x_i).AND. LE (x, x_i+cx_j))
.AND. (GE (y, y_i+py_i).AND. LE (y, y_i+cy_j))
.AND. (GE (z, z_i).AND. LE (z, z_i+cz_j)) ······(43)

3) Type C

DEFSET (G31, (x, y, z), COND3) ······(44)

COND3 : = (GE (x, x_i).AND. LE (x, x_i+cx_j))
.AND. (GE (y, y_i).AND. LE (y, y_i+cy_j))
.AND. (GE (z, z_i+pz_i).AND. LE (z, z_i+cz_j)) ······(45)

Calculation of the space dividing is executed in a recursive manner.

step5) Compare the volumes of these three allocation spaces. In this program, adopted strategy for the space selection is to use the largest allocation space first. The common space between genarated spaces is exclusively used by the space of higher priority. In the example of Fig. 4, the subsequent volume will be generated by the following procedure.

s1) Compare the volumes of G11, G21, G31
result : GT (G11, G31), GT (G31, G21)
Pack boxes into G11 until stop rule holds

s2) Subtract the common space
G22 = G21 − PROSET (G11, G21) ······(46)
G32 = G31 − PROSET (G11, G31) ······(47)
Compare the volumes of G22 and G32
result : GT (G32, G22)
Pack boxes into G32 until stop rule holds

s3) Subtract the common space
G23 = G22 − PROSET (G22, G32) ······(48)
Pack boxes into G23 until stop rule holds

step6) Stop rules of the procedure are the following. If not satisfied, return to step1 and repeat the same procedure.

sr1) GT (SUM (w), W_{lim}) ······(49)

sr2) EXIST (b_i) is F for all allocation spaces such that
SUBSET (G, S_j) ······(50)

Therefore, the calculation will continue until total weight exceeds the given limit or allocation spaces are divided into small rectangular areas in which no box can be placed.

5. The developed system

We first explain inputs and outputs of the system and then explain calculation flow of the simulation.

5-1 Inputs and Outputs

The developed system has the following inputs and outputs.

Inputs :

I1) The box data B

Dimensions, weights, packing conditions and attribute information.

Packing conditions are :

$pc_i = 1$: upright condition is requested

$pc_i = 2$: prohibit packing in the lowest layer

Attribute information is :

(1) product name

(2) product code

(3) factory code

(4) destination code

I2) The container data C

I3) The weight limit W_{lim}

Outputs :

O1) Perspective views and top views of packed boxes in containers

O2) List of volume efficiency of each container

O3) List of contents of each container

5-2 The packing simulation by the simulator

Fig. 5 shows a flowchart of the packing simulation by the simulator. In this

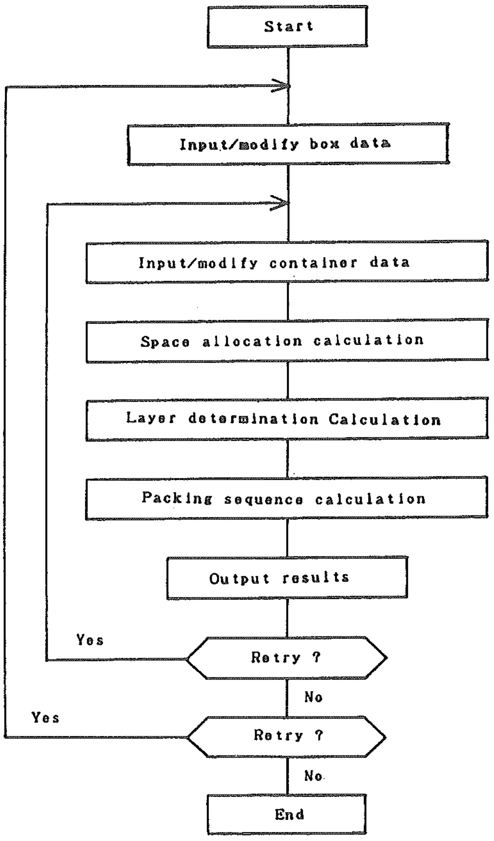
Start
Input/modify box data
Input/modify container data
Space allocation calculation
Layer determination Calculation
Packing sequence calculation
Output results
Yes
Retry ?
No
Yes
Retry ?
No
End

Fig. 5 A flowchart of the packing simulation

simulator, the box data is defined for each destination. The calculation is divided into 3 steps. The last 2 steps (layer determination and packing sequence calculation) are required for graphic outputs.

By modifying container data repeatedly and comparing different results, a container of high volume efficiency can be found.

6. Outputs by the simulator

Fig. 6 shows obtained graphic outputs of the following container.

width = 506 mm
length = 316 mm
height = 363 mm
the number of packed boxes = 90
total weight = 27 kg
volume efficiency = 0.875

The pictures show perspective views drawn from 1st layer to 4th layer. Using these pictures, the result of calculation can be checked easily.

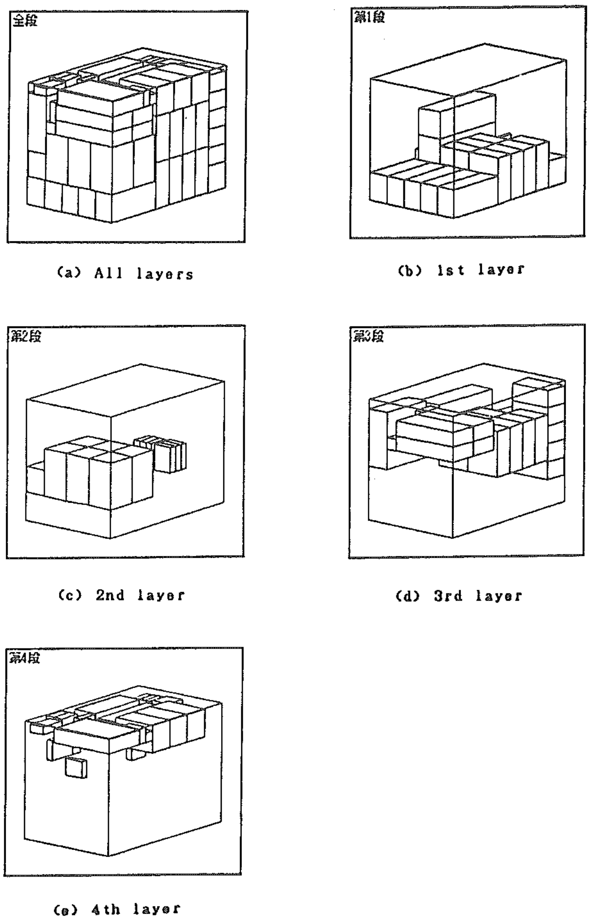

Fig. 6 Output-1 (perspective views)

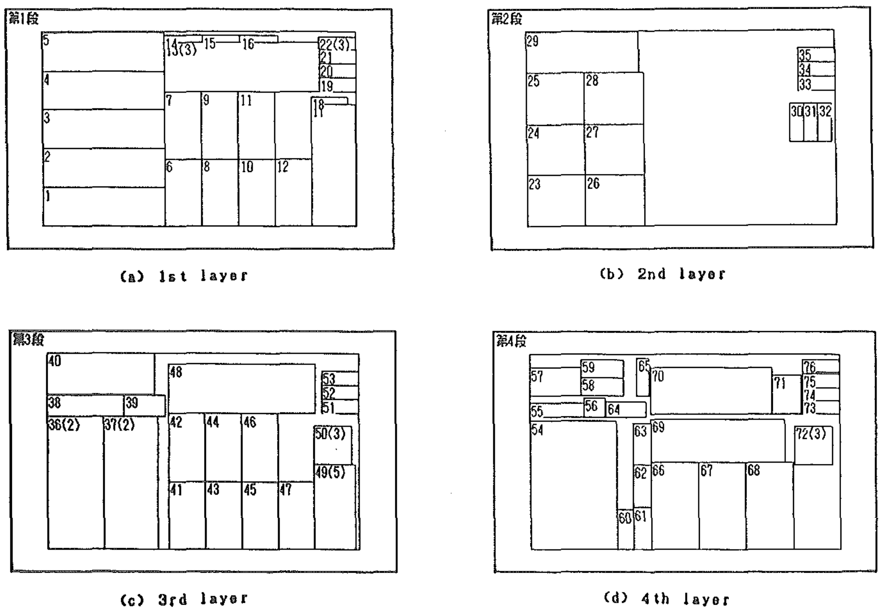

Fig. 7 Eutput-2 (top views)

Destination	n	m	r	CPU time (sec)
A	1,082	23	0. 81	280
B	860	18	0. 80	270
C	650	16	0. 75	250
D	584	13	0. 73	240
E	475	11	0. 70	185
F	408	9	0. 76	150
G	358	10	0. 69	115
H	301	7	0. 70	100

Tale 2 The list of CPU time

Fig. 7 shows top views of the same result as Fig. 6. Each box is identified by number at each layer. The pictures can be used for actual packing work according to the numbers which are related with product names.

6-2 CPU time

Table-2 shows an example of required CPU time for packing calculation. The symbols n, m and r in the table show the number of boxes, the number of containers required and volume efficiency respectively. In the data it can be observed that increase of CPU time by increase of the number of data is small. (CPU : i8086 and i8087)

7. Conclusions

1) An approach to build a 3D packing simulator was made.
2) A heuristic approach method to the combinatorial problem of 3D geometry was shown.

3) Based on the method, an expert system for 3D packing was developed.
4) For verification of calculation results, graphic outputs of packed boxes such as perspective views and top views were shown.
5) For the aid of actual packing operations, the graphic outputs were shown layer by layer from the bottom to the top layer.
6) Required CPU time for the calculation was shown. The proposed approach method was proved to be effective for realitime simulation of 3D packing up to about 1000 boxes.

REFERENCES

1) R. C. Wilson "A Packing problem" Management Science, Vol. 12, No. 4, P. B-135, 1965.
2) M. Furukawa "Practical Approaches to The Space Allocation Problem" (unpublished) Ph. D. Dissertation, Hokkaido University.