



Title	PRACTICAL APPROACHES TO THE SPACE ALLOCATION PROBLEMS
Author(s)	Furukawa, Masashi
Citation	北海道大学. 博士(工学) 乙第2140号
Issue Date	1981-09-30
Doc URL	http://hdl.handle.net/2115/32634
Type	theses (doctoral)
File Information	2140.pdf



[Instructions for use](#)

PRACTICAL APPROACHES
TO THE SPACE ALLOCATION PROBLEMS

MASASHI FURUKAWA

Contents

	Page
1. Introduction	1
1.1 Objectives and Motivations	2
1.2 A General Model	5
1.2.1 Resources	5
1.2.2 Material	6
1.2.3 Relation	7
1.2.4 Geometry Description	7
1.2.5 Performance Measure	9
1.3 Background	11
1.3.1 One-dimensional Form---Knapsack Problem, Optimum Packing, and Space Allocation Problem	11
1.3.2 Two-dimensional Form---Cutting Stock Problems, Multi-job Shop Scheduling	17
1.3.3 Three dimensional Form---Packaging Problem	22
1.3.4 Graphic Processing in Space Allocation	26
1.3.5 C.A.D. in L.S.I. Design	34
1.4 Conclusion	37
2. A practical New Solution to Flow-Shop Scheduling Problem---1 1/2-Dimensional Space Allocation Problem---	44
2.1 Introduction	44

2.2	A Formulation of the Problem	47
2.3	A Utility Function under a Weak Order	
	Relation in the Flow Shop Scheduling	49
2.3.1	Preference as a Weak Order	49
2.3.2	An Order-Preserving Utility Function	51
2.3.3	Precedence Relation on the Flow Shop	
	Scheduling	52
2.3.4	A Directed Graph and Tournament	53
2.3.5	A Utility Function in the Tournament	55
2.4	Algorithm	58
2.5	Validity of the Algorithm for Flow	
	Shop Scheduling	61
2.6	Numerical Experiments	62
2.7	Conclusion	70
3.	P.B.M. Approach to the Space Allocation	
	problem---The Optimum Trimming of Many	
	Rectangular Plates---	72
3.1	Introduction	72
3.2	A Formula of the Problem	73
3.3	Fundamental Approach to Problem	76
3.4	The Algorithm of P.B.M.	78
	3.4.1 The Basic Procedure	79
	3.4.2 Waste Matrix	80
	3.4.3 Formulation as the Assignment Problem	83
	3.4.4 Solving Method for the Assignment	84

Problem	
3.4.5 Determining the Lines	88
3.5 Given Blank Having Restricted Width	90
3.6 Approach with the Iterative Enumerative Method	91
3.7 The Numerical Experiments	93
3.8 Conclusion	102
4. A Practical Approach to the Cutting Stock Problem	104
4.1 Introduction	104
4.2 Formula of the problem	105
4.3 Nesting and Assignment Algorithm	108
4.3.1 Recursive Procedure for Nesting	109
4.3.2 Nesting Procedure	110
4.3.3 The Data Structure for Nesting	115
4.3.4 Determination of the Number of the Row Blanks	119
4.4 Numerical Experiments	121
4.5 Conclusion	142
5. An Approach to the Package Problem	
---Optimum Packing and Loading---	146
5.1 Introduction	146
5.2 Problem Description	147
5.3 Formula of the Problem	150

5.3.1	Notations and Assumptions	150
5.3.2	Constraints and Object Function	151
5.3.3	Qualifying the Formulas	153
5.4	Enumeration Method	155
5.5	Determination of Standard Carton Box	157
5.6	Numerical Experiments	159
5.7	Additional Criteria	168
5.8	Conclusion	170
6.	Minimum Partition of a Compound Rectangular Cell	172
6.1	Introduction	172
6.2	Problem Description	173
6.3	Analysis of Minimum Partition	176
6.3.1	Base Vertices for Partition and Their Number	177
6.3.2	Theorem for Minimum Partition	179
6.4	Minimum Partition Algorithm	189
6.4.1	A Graph Representation of Poli-cell P	189
6.4.2	Judgement of Base Vertices	190
6.4.3	Determination of Partition Vertices	191
6.4.4	Minimum Partition Algorithm	193
6.5	Extraction Algorithm of Uni-cells	195
6.6	Experiments	197
6.7	Conclusion	203

7.	Graphic Processing in the Space Allocation Problem	205
7.1	Introduction	205
7.2	Geometric Definition and Data Base	207
7.3	Recognition of Space Allocation Feasibility	210
7.3.1	Boundary Evaluator	211
7.3.2	New Evaluator	215
7.3.3	The Recognition Method for Space Allocating Feasibility	218
7.3.4	Collision Prohibition	219
7.3.5	Surface Equations of the Space	220
7.4	Application of the Collision Prohibition Technique to Material Shear Scheduling	222
7.4.1	Recognition of Shearing Feasibility	223
7.4.2	Shear Scheduling	226
7.5	Conclusion	229
8.	The Development of CAM Software System for Punching-press and Shearing	231
8.1	Introduction	231
8.2	System Design	232
8.2.1	System Specification	232
8.2.2	Input Information and Language	236
8.2.3	Input Translating Processor SCANNER and Canonical Data File	237
8.2.4	The Allocation Processor OPTNST	243

8.2.5	Task Sorting Processor TSKCLS	244
8.2.6	Punching-press Tool Path Deter- mination Processor OPTPTH	246
8.2.7	Shear Scheduling Processor SHEARS	248
8.3	Examples	250
8.4	Conclusion	256
9.	Conclusion	257
Acknowledgement		261
Appendix A Mitsubishi Metal Sheet Production System		263
A.1	Introduction	263
A.2	Hardware System	263
A.3	Software System	267
A.4	Effects of System Development	271
Appendix B OPTI-CUT System Mannual		272
Appendix C Comparison of Boundary Evaluators		278

1. Introduction

This chapter is intended to introduce and organize a common contexts and their contents to be dealt with in the following chapters. For this purpose, the common terminology for models to be analyzed later is presented with some materials with which to survey the background. Though the description of the problems varies from chapter to chapter, the problems themselves are recognized from the common point of view and accordingly generalized in this chapter.

In section 1.1, the objectives and motivations are described. Section 1.2 is intended to create generalized space allocation problems discussing:

- 1.2.1 Resource
- 1.2.2 Material
- 1.2.3 Relation
- 1.2.4 Geometry Description
- 1.2.5 Performance Measures

The background of the space allocation problems is surveyed in section 1.3 in the following order.

- 1.3.1 One-Dimensional Forms
- 1.3.2 Two-Dimensional Forms
- 1.3.3 Three-Dimensional Forms
- 1.3.4 Graphic Processing in Space Planning

1.3.5 CAD in LSI Design

The materials given in 1.3 do not completely cover all the space allocation problems, but still they extract the common recognition of the problems.

Section 1.4 concludes the remarks of the space allocation problems.

1.1 Objectives and Motivations

The space allocation problem may roughly be described as follows:

- a) to make up an optimum resource by building material spaces under the given criteria.
- b) to divide the given resource into optimum materials under the given criteria.

The objectives of this paper is to present new methods and optimum solutions, or near optimum solutions at the least, practical enough for engineering.

These types of problems are actually and easily found in such programs as to how to apply a number of tiles on the floor, how to allocate rooms within a restricted area, how to load cargos on a truck, how to patch tasks on a gantt chart and so on. In the field of engineering too, there are same type of problems related to the extension of information

processing for manufacturing system. These problems include a cutting stock problem in metal sheet industry, machinery layout problem in a factory, multi-job scheduling problem in a computer, LSI design in CAD, etc.

Before using a computer for these problems, either an engineer, a planner or a designer has to spend quite a few days in solving or designing them through "trial and error" and "experience", which are the only approaches available. To make matters worse, the problems described above are essentially combinatorial, therefore there are a number of solutions, and the engineer has to select the best possible one. Certainly the recent development of computers of high speed processing with huge memory capacity has done much to eliminate such difficulties. However, efficient methods have not yet been established except for a few cases. Under such circumstances, an approximate solution to the problems would bring a great deal of economical effects to industry. In fact, a computer-aided manufacturing system for metal sheet cutting, developed by Mitsubishi Electric Company, adopted the method to be presented in chapter 3. This saved 9882 hr/yr and 7.2 workers in producing desired products by raw metal sheet shearing as compared with the productivity before the system development.

Also, the method to be presented in chapter 4 is actually applied in Murata Machinery Company to a shear CAM system which is developed in chapter 8. The efficiency of raw metal

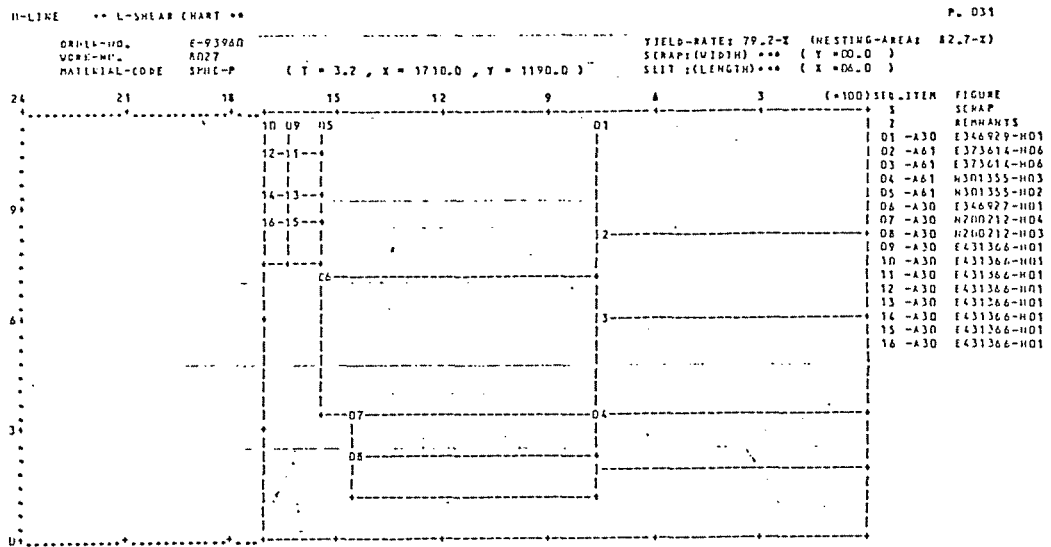


Fig. 1.1 An example of metal sheet product allocation
 (Mitsubishi Electric Company; presented by
 Y. Tomooka.)

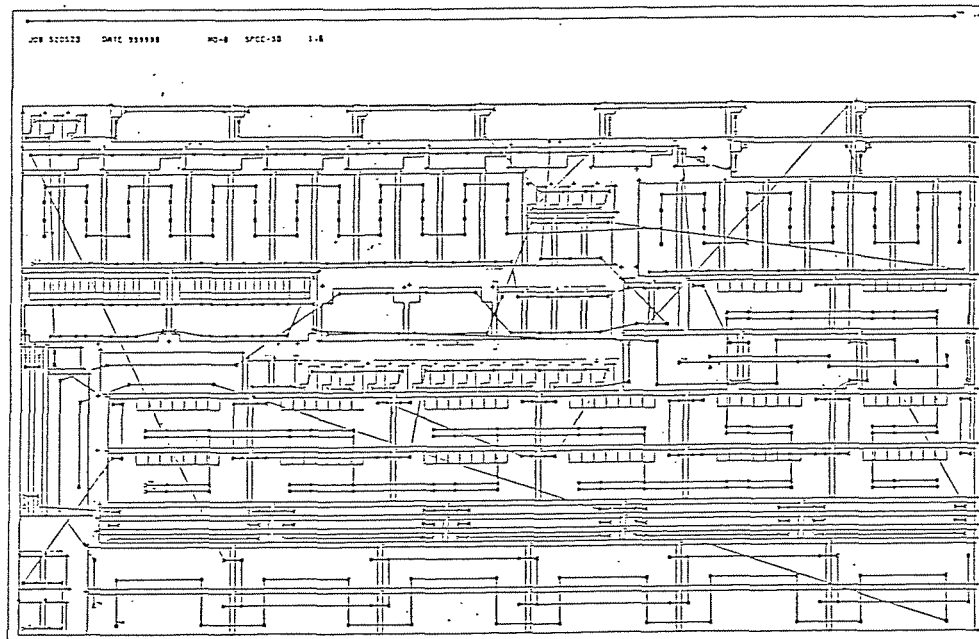


Fig. 1.2 An example of metal sheet product allocation
 (Sapporo Software Engineering; presented by
 M. Yoneyama.)

sheet utility reached around 90% --- a rise from 70%.

The studies on the problem start with the motivations under the situation mentioned above. Most of the problems to be dealt with in this paper are known as "Non-Polynomial Problems", meaning "Polynomial Time Algorithm" on the description size n of the problem has not been found as yet. Therefore, this paper aims at practical approaches to the space allocation problems, which will give optimum solutions practical enough for engineering.

1.2 A General Model

A space allocation model, from which subsequent problems are drawn, is described by considering resource, material, geometry description, relation and performance measure.

1.2.1 Resources

Resources mean spaces either composed of given spaces or which can be divided into separate spaces. Such spaces are blanks nested by material spaces, buildings consisting of given functional rooms, a gantt chart for job shop scheduling dispatched by jobs, computer processing capacities assigned

by tasks, holes filled in with bricks and so on.

Resources are written by a set R ,

$$R = \{R_1, R_2, \dots, R_m\}.$$

A resource R_i is often divided into a subset such as:

$$\mathcal{Z} = \{T_1(R_i), T_2(R_i), \dots, T(R_i)\},$$

where $R \supset \mathcal{Z}$. An example of this is found in the computer multi-job scheduling where T_1 is an input, T_2 is a subroutine library, T_3 is processing and T_4 is an output.

1.2.2 Material

Materials mean either spaces into which resources are divided or spaces with which to compose resources. Such materials include the ones sheared from blanks of raw metal sheets, rooms allocated in a building, machines located in a factory, tasks for multi-job scheduling, CM time for TV time scheduling, boxes loaded on a pallet and so on.

Materials are written by a set M ,

$$M = \{M_1, M_2, \dots, M_n\}.$$

There are sometimes space constraints which restrict materials.

The space constraints are written by a set C_i ,

$$C = \{C_1, C_2, \dots, C_n\} \text{ restricting as}$$

$$M_i \subset C_i, i = 1, 2, \dots, n.$$

1.2.3 Relation

There is a case in which the resources and the materials have a mutual relation in their elements. Such a relation is called a "binary relation". A notation which shows the binary relation can be introduced as:

$R_i \ B \ R_j$ R_i has the relation B with R_j
and $M_i \ B \ M_j$ M_i has the relation B with M_j

Such an example is found in the relation of a room M_i with the adjoining room M_j .

1.2.4 Geometry Description

Spaces dealt with in the problems have their own shapes. Therefore, geometry descriptions for the spaces are essential. In order to describe any shapes of the spaces, a generalized description method need be introduced. The method must also be effective not only for the geometry descriptions but to remedy graphic processing which occurs in space allocation. Such graphic processing to be overcome includes collision problems between spaces and recognition problems on where the spaces are allocated. From the viewpoint mentioned previously,
(1)
"Formulated Pattern Method" developed by Prof. N. Okino is applied as the description method if such situations come out. "Formulated Pattern Method" (FPM) is simply illustrated as

follows.

The given space P is divided into primitive spaces P_i , and P is presented by the use of a union operator in a set theory as below:

$$P = \bigcup_{i=1}^a P_i, \quad (1.1)$$

where a is the number of primitive spaces. Primitive spaces are divided into half spaces P_{ij} , then we gain, by the use of an intersection operator,

$$P = \bigcup_{i=1}^a \bigcap_{j=1}^b P_{ij} \quad (1.2)$$

where b is the number of half spaces of primitive one P_i .

As set $P_{ij} = \{x \mid P_{ij}(x) \geq 0\}$, Eq. (1.2) becomes

$$P = \bigcup_{i=1}^a \bigcap_{j=1}^b \{x \mid P_{ij}(x) \geq 0\}. \quad (1.2')$$

As to the application of this description method to graphic processing in the space allocation, the usage of "Boundary Evaluator" is offered in chapter 7.

1.2.5 Performance Measure

Performance measures for the problems depend on the situation in which a problem occurs. The subsequent chapters mainly treat the followings as the performance measures:

- a) A minimum waste in cutting and trimming situation
- b) A minimum cost in packing situation
- c) A minimum partition situation

Each of the above is briefly discussed in the following section.

- a) A minimum waste in cutting and trimming situation

An extremely common industrial problem is of the following type: cutting material items to satisfy a set of orders for non-material size of a resource. The material-depletion form of the problem occurs frequently where cutting is necessary and has been treated extensively under various names, such as "stock cutting", "scrap reduction" and "trim loss". Exactly parallel one-, two- and three-dimensional forms are displayed in connection with cutting length, sheets, and blocks respectively. Otherwise, if we consider "locating" instead of "cutting", the depletion problem

becomes a packing problem. As the performance measure in the stock cutting case, the trim loss (the waste) needs to be minimized. In general, the problem of this type is known as the "knapsack problem".

b) A minimum cost in packing situation

A complex problem may arise when a number of similar products are made in a wide variety of sizes. Each of the products is to be packed in an individual cardboard box. A typical instance can be found in ball-bearings. If each product is packed in the smallest box that will contain it exactly, the warehouse space for stocking the completed articles, transport costs based on volume and the actual box costs will all be reduced to the minimum.

c) A minimum partition situation

When the resource is divided up into the materials whose shapes are restricted, the number of the materials needs be minimized. A similar performance is found in making the materials in the simplest shape in as large area as possible from complex shape resources. In an automated LSI pattern development, this performance is needed to minimize the developing time due to the number of patterns decoded from the original LSI pattern.

1.3 Background

1.3.1 One-dimensional Form---Knapsack Problem, Optimum Packing, and Space Allocation Problem

Consider a group of materials M_j ($j = 1, 2, \dots, m$) and a resource R . The problem is simply to pack (allocate) as many materials as possible into the resource without any protruding. Let us set u_j to the area of M_j , d_j to the price of M_j and w to the area of R . The problem is described as

$$\text{minimize} \quad \sum_{j=1}^m d_j x_j, \quad (1.3)$$

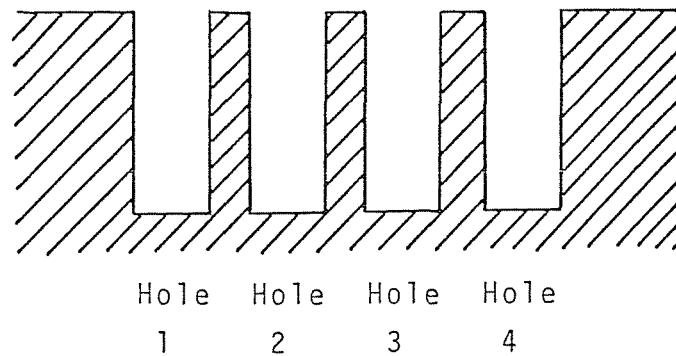
$$\text{subject to} \quad \sum_{j=1}^n u_j x_j \leq w \quad (1.4)$$

$$\text{where } x_{ij} = \begin{cases} 1 & M_j \text{ is packed} \\ 0 & \text{Otherwise} \end{cases}$$

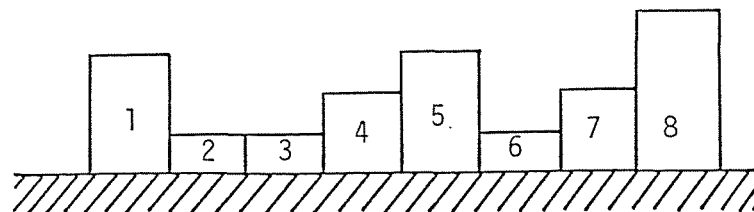
The inequality Eq. 1.4 is derived from the geometry condition under which the sum of the material area packed should be smaller than the area of the resources. If we regard u_j as the weight of the given j -th item and w as the sum of the weight allowed to load items and d_j as the price of the j -th item, this problem is known as a "Knapsack Problem" which occurs in some cargo loading operation. Gomory⁽²⁾⁽³⁾⁽⁴⁾ solved it by formulating it as

"An Integer Linear Programming" for "All Cutting Plane Method". Bellman and Drefus applied "Dynamic Programming Method" to the problem and Kolser⁽⁵⁾ approached to it by the use of "Branch and Bound Method". It seems that Kolser's method is most efficient among them, but the program dimensions solvable are ranged from 3 to 100 items by Kolser using IBM 7094.

In the case where some resources are given, the problem becomes packing of all the materials in the resources. Let us consider the materials as bricks and the resources as holes. Then, the holes are of a similar size, and the bricks are all of the same cross-section as the holes but different in length. (Fig 1.3)



(a) Holes



(b) Bricks

Fig. 1.3 Optimum packing problem

The problem is simply to pack all the bricks into the holes without any protruding. This type of the problem is known as "Optimum Packing in One-Dimensional Form". "TV Spot Reservation Problem"⁽⁶⁾ and "Time Tabling Problem"⁽⁷⁾ are two examples actually found. A.R. Brown described the detail of the treatment of this problem in his "Optimum Packing and Depletion"⁽⁸⁾

Instead of surveying this problem, let us introduce another problem in the same situation as the above. If we treat the bricks as jobs and the holes as the machines, the problem becomes "Job Shop Scheduling Problem". In order to describe the problem, let us formulate the problem as follows:

$$\text{minimize} \quad \sum_{j=1}^m (\max_{j=1}^m P_j - P_j), \quad (1.5)$$

$$\text{subject to} \quad P_j = \sum_{i=1}^n F_i x_{ij}, \quad j = 1, 2, \dots, n \quad (1.6)$$

$$\sum_{j=1}^n x_{ij} = 1, \quad x_{ij} = 1 \text{ or } 0 \quad (1.7)$$

Where P_j is a processing time of the machine R_j , F_i is the job time M_i , $x_{ij} = 1$ implies the job M_i is processed by the machine R_j and $x_{ij} = 0$ implies otherwise. An objective function adopted as the performance measure is the situation of the cutting stock which extracts a minimum waste shown in Fig. 1.4 by cross-hatched lines.

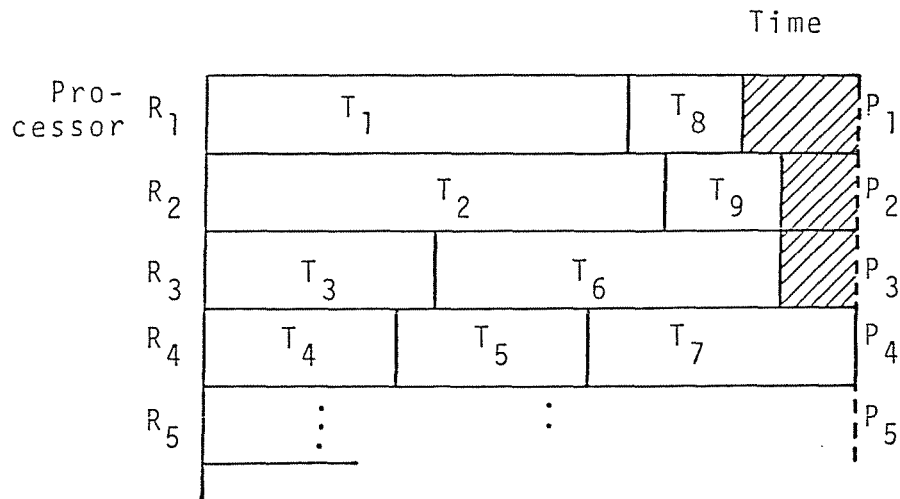


Fig. 1.4 Job shop scheduling problem

As the second term becomes constant without wait times, the objective function is rewritten by

$$\text{minimize } \max_{j=1}^m P_j \quad (1.8)$$

Eq. 1.5 shows "mean weighted finishing time" and Eq. 1.4 shows "maximum finishing time" in job scheduling problem. To be more general, sequence conditions under which job M_i must be preemptive before processing M_j are added to the problem and the problem becomes more sophisticated. The researches into this aspect have been made by E.G. Coffman.⁽⁹⁾

One of the typical problems that belong to job shop scheduling is "Flow-shop Problem"⁽¹⁰⁾⁽¹¹⁾⁽¹²⁾⁽¹³⁾. This is the N-P complete problem, on which S.M. Johnson⁽¹⁰⁾, I. E. Ignal⁽¹¹⁾, I. Nabeshima⁽¹²⁾ and H. Kubo⁽¹³⁾ attempted to solve it. S. M. Johnson presented and analyzed the formulation of the problem and gave to 2-machine n-job problem a simple rule. I. E. Ignal, I. Nabeshima and H. Kubo tried to apply "Branch and Bound Method" to the problem and they made an effort to establish "more efficient lower bound". But solvable number of the machines for the problem is maximum 200 by H. Kubo's method using a computer with large capacity (FACOM 230/75). In order to solve a practical problem in a factory, more efficient method should be desired even if it cannot reach an exact solution within a reasonable calculation time and a reasonable cost.

Another performance measure of the space allocation is offered by D. M. Simmon⁽¹⁴⁾. The problem discussed arises when an architect tries to arrange rooms of fixed area but unspecified shape in a floor plan in such a way as to minimize a given linear combination of the distances between all pairs of rooms. In one dimension, this is the problem of ordering line segments along a simple axis or rooms along one side of a corridor. In this problem, the rooms take the place of the materials M_i and the fixed

area implies the resource.

The performance measure for the problem is the average daily traffic between two rooms---walking distance and traffic densities. Therefore, the expected total distance traveled by people of all distances is minimized. In this case, the space allocation problem becomes an ordering problem rather than one of the space allocations. The formula of the problem by D. M. Simmons is as follows:

$$\text{minimize } \sum_{i=1}^n \sum_{j=1}^i C_{ij} S_{ij}, \quad S_{ij} \neq 0 \quad (1.9)$$

This is being visualized as the problem of ordering a set of rooms of nonuniform known length along one side of a corridor where the cost accessed for each pair (i, j) of rooms is some scalar multiple C_{ij} of their separation S_{ij} . The operation S_{ij} is the sum of the half-length of rooms i and j added to the length of all the rooms between them.

As $\sum_{j=1}^i C_{ij} S_{ij}$ for $i \neq j$ becomes the same formulation as

the mean weighted job shop scheduling regarding the room length and C_{ij} as the processing time and the job weight respectively (where the sum of the half-length of rooms i and j is ignored), it becomes possible to make use of SPT rules that figure out the lower bound for applying the branch and bound method to solve the problem. D. M. Simmons solved the problem by finding this property. The maximum

number of rooms for his numerical experiments is 15 by the IBM 360-40/65 which is roughly 370k bytes memory available.

1.3.2 Two-dimensional Form ---Cutting Stock Problems, Multi-job Shop Scheduling

The approaches to the cutting stock problem are well known owing to Gomory and Gilmore.⁽³⁾⁽⁴⁾ Their first approach is to formulate the problem as a linear programming and to develop the efficient simplex method originated for the problem. In this problem situation, stocks take the place of resources.

The formulation and algorithm developed by them are as follows.

Let us define the notation:

L_1, L_2, \dots, L_k stock length

l_1, l_2, \dots, l_m : ordered material length

N_1, N_2, \dots, N_m the number of pieces of the ordered material

a_{ij} ($j=1, \dots, n$) the number of pieces of length l_j created by j -th activity.

c_j ($j=1, \dots, n$) the costs of the stock length cut by j -th activity

x_j ($j=1, \dots, n$) the variables assigned to j -th activity

The objective function to be minimized is

$$c_1 x_1 + c_2 x_2 + \dots + c_n x_n. \quad (1.10)$$

The variables x_1, \dots, x_n must satisfy m inequalities,

$$a_{i1} x_1 + a_{i2} x_2 + \dots + a_{in} x_n \geq n_i. \quad (1.11)$$

$(i = 1, \dots, m)$

By the use of the vector notation, the problem is

$$\text{minimizing} \quad C^T X \quad (1.12)$$

$$\text{subject to} \quad A X \geq N \quad (1.13)$$

where $C = (c_1, \dots, c_n)^T$, $X = (x_1, \dots, x_n)^T$, $A = \{a_{ij}\}$,
and $N = (n_1, \dots, n_m)^T$. Their method is based on the relation
between the primary problem and the dual problem. The
dual problem against the problem described by Eqs.

(1.12) and (1.13) become

$$\text{maximizing} \quad N^T U \quad (1.14)$$

$$\text{subject to} \quad A^T U \leq C \quad (1.15)$$

where U is the variables $(u_1, u_2, \dots, u_m)^T$ for the dual
problem. If the both problems are optimized, it is known
as $C^T X = N^T U$. The algorithm presented by them is based on
this principle and it is briefly described as follows.

1. Select feasible X so that $AX = N$ where the activity
 $A_i = (a_{i1}, a_{i2}, \dots, a_{mi})^T$ is figured out by solving a
subproblem as a knapsack problem.

2. Determine U so that $A^T U = N$.

3. Test optimality of the solution and its dual by

$$C^T X = N^T U.$$

4. If $C^T X = N^T U$ the solution is optimal for the activities selected. If not, go to step 5.
5. Select a new activity $A_s = (a_{1s}, a_{2s}, \dots, a_{ms})^T$, so that $A_s^T U \leq c_k$. ($1 \leq k \leq m$).
6. If A_s is a really new variable to be entered into the basis, change A_s into A_k and update A and C .
7. Determine U so that $A^T U = C$.
8. Determine X so that $A X = N$. Then go to step 3.

An important point in this algorithm is that all possible activities are not listed in each step. This saves a great deal of memory for calculation. It is considered that the algorithm is the revised simplex method.

In the continued research work, they have formulated the problem as a generalized knapsack problem and solved it by Dynamic Programming. S. H. Hahn applied the D. P. method principally developed by them for the cutting stock problem with defects.

Mathematical programming approaches to the two-dimensional problem have briefly been surveyed as above. On the other hand, two-dimensional optimum packing problem is identified with the very geometric two-dimensional problem by Codd, in scheduling hardware facilities of a large, multi-programming computer.

This problem is called a multi-job scheduling problem, and in its problem the resource and the materials are regarded as a program load and programs, respectively.

Codd's procedure⁽¹⁵⁾ attempts to place program's component rectangles on their load diagrams according to a set of placement rules. The programs are ordered according to the rules dependent on priority or precedence or, in the absence of both, on the longest running time.

The placement rules offered by him consists of the following three.

The program being considered is P.

Rule 1 --- Fitting criteria

- (1) P's rectangle must be within the upper bound of facility.
- (2) P must not intersect any rectangles placed earlier.

Rule 2 --- Left justification

- (a) No program may start unless another is terminating.
- (b) P's rectangle should be placed as far left as possible.

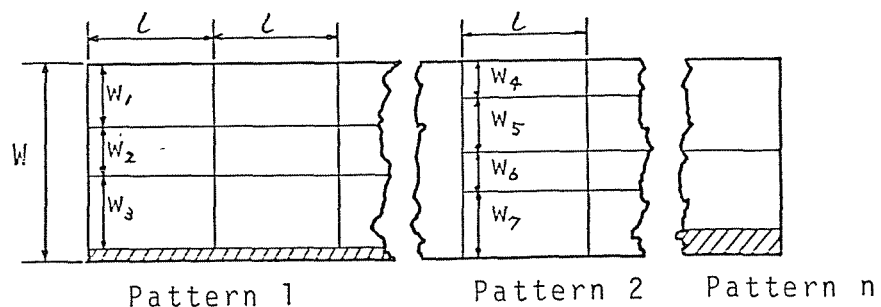


Fig. 1.5 Simplex method model

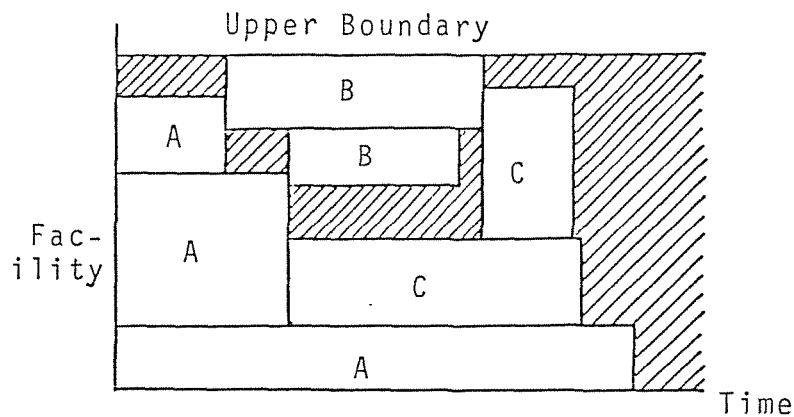


Fig. 1.6 An example of Codd's rule

Each program to be scheduled will give rise to several rectangles on several load diagrams. The object of this rule is to pack the schedule tight so that it is less likely to be affected by changing situations or requirements. It is obviously less densely packed to the right, leaving room for manipulation in the event of unavoidable changes.

Rule 3 --- No fragmentation

The "vertical" space---facility extent---is not split unless absolutely necessary.

These rules are satisfied, considering each program in turn as mentioned earlier, by a stepping procedure

looking for "pyramid" bases or to extend existing "pyramid". The pyramid concept is best illustrated as shown in Fig. 1.6, where rectangles A form one pyramid and B and C others. The heavy lines are their basis which are always either:

- (a) the upper boundary,
- (b) the lower boundary, or
- (c) the uncovered part of a layer of an existing pyramid.

The pyramid can be built up or down from the basis, and the whole process is easy to specify in terms of manipulation of the coordinates of the rectangles. The similar procedure has been developed for metal sheet nesting by M. Yoneyama.

1.3.3 Three-dimensional Form---Packaging Problem

Three-dimensional form of the space allocation problem is known as a generalized knapsack problem being studied by P. S. Gilmore and R. E. Gomory⁽³⁾⁽⁴⁾⁽⁵⁾ and as a packaging problem by R. C. Wilson.⁽¹⁶⁾ It is not sufficient enough to do research in this aspect only.

This section briefly describes the latter problem.

In many consumer goods product lines, a class of similar products are produced in a large number of different physical sizes. Each product must be packed into its own

cardboard box. The costs are minimized if each product is packed in the smallest box which fits the product exactly. The savings from using a different size box for each product are offset by the higher price of boxes purchased in small quantities and the additional ordering, inventorying, and handling costs of many different box size. If one assumes that demand activity for the period is known exactly for each product, the problem is to determine:

1. the number of n of different size boxes for the line of m different size products,
2. the dimensions of these n boxes, and
3. which products to insert in which box in order to minimize the total cost of packing during the period.

The problem is formulated as below.

$$\text{Minimize} \quad \sum_{i=1}^m \sum_{j=1}^n a_j c_j x_{ij} + \sum_{j=1}^n y_j k_j, \quad (1.17)$$

$$\text{subject to} \quad \sum_{i=1}^m b_{ij} x_{ij} = 1 \quad (i = 1, \dots, m) \quad (1.18)$$

$$\sum_{i=1}^m a_i b_{ij} x_{ij} \geq d_i \quad (1.19)$$

or that

$$\sum_{i=1}^m a_i b_{ij} x_{ij} = 0. \quad (j = 1, 2, \dots, n) \quad (1.20)$$

The notations used are defined as:

- a_i the forecasted demand activity for product i during the period, in units ($i = 1, \dots, m$),
- c_j total cost of box size j ($j = 1, \dots, n$),
- c_{j1} : purchase cost of the box size j ($j = 1, \dots, n$),
- c_{j2} cost of the warehouse space occupied by the box size j ,
- k_j system operating cost if the box size j is used, and
- $y_j = \begin{cases} 1 & \text{box } j \text{ is used} \\ 0 & \text{otherwise} \end{cases}$
- d_i minimum acceptable activity of box size j in the period, and
- $x_{ij} = \begin{cases} 1 & \text{if the product } i \text{ is packed in the} \\ & \text{box size } j \\ 0 & \text{otherwise} \end{cases}$

A constraint Eq. 1.18 implies each product must be packed in only one size box. Constraints Eqs. 1.19 and 1.20 imply if the box size j is to appear in the solution, economics of purchasing requires that at least d_j of the box size j be purchased during the period, or else none are to be purchased. In the objective function, the first term is the cost when the product i is packed in the box size j and the second term is the total operating cost of the

number n boxes in the system.

The resources are the boxes and the materials are the products in the program obviously.

In the environment in which this problem arose, the number of different products, m , was about 2000 and the number of meaningful increment, h , in each dimension was 150, giving $n = 3,375,000$. The number of variables therefore exceeds a billion and the number of constraints imposed by Eqs 1.18 and 1.19 exceed 10 million. For this reason, a heuristic method is presented. The method consists of three steps---step 1, box size generation, step 2, box reduction, step 3, selection. The list of candidate boxes is generated in the step 1. The list is augmented by comparing each subsequent product with the boxes already on the list and adding a new box if none on the list are within the acceptable tolerances. In the step 2, each box is eliminated step by step from the initial set (established in the step 1) so that the cost is least incremental. Then, the number of boxes is decided to satisfy Eq. 1.18. In the step 3, the results of box reduction are examined as to whether they satisfy the restrictions Eqs. 1.19 or 1.20. Also the total cost is calculated. If the total cost becomes relatively flat by inspection, step 3 is concluded. If not, alternative box sizes which are the function of the operating cost are sought and tested again.

R. C. Willson succeeds in saving over 25% of the cost of present cardboard boxes and space costs per year.

In chapter 5, the similar situation of the problem is discussed to determine the box size and the carton size in which the boxes are packed.

1.3.4 Graphic Processing in Space Allocation

In the survey made in the previous sections, the geometries of the resource and the materials are simplified as rectangles and cubics in order to turn the problem to mathematical programming. Therefore, the problem is not dealt with from the viewpoint of graphic processing. In spite of the efforts to make the problem simple, information is needed on how the materials are located within the resources or how the resources are cut out into the materials, at least. For instance, Dynamic Programming approach by P. C. Gilmore⁽⁵⁾ and others prepare the two arrays to show the material location. The two arrays store the upper value coordinate of the material located sequentially on the x axis and y axis within the resource.

As shown in the above simple instance, another important phase of the space allocation problem is the graphic processing.

The graphic processing necessary in the space allocation problem must solve the problems of how the material and the resource geometries are described and stored into the computer, how the existence of the materials within the resource is recognized by a computer, and how the relation of the locations between materials or between the resource and the material can be recognized. There are few attacks in the field of mathematical programming but there are some found in the field of architecture space planning. The space allocation situation is called Space Planning in the Computer-Aided Design of architecture. Let us inspect the graphic processing in architecture in the respect of the space allocation problem briefly.

In space planning, the rooms for a house is considered as the materials and the area allowed for building the house as the resource.

The most common problem formulation in space planning deals with the weighted distance between an arranged set of rooms.⁽¹⁷⁾ The objective function is

$$\min \sum_{i=1}^n \sum_{j=1}^n d_{ij} w_{ij}$$

$$\text{where } d_{ij} = ((x_i - x_j)^2 + (y_i - y_j)^2)^{1/2} \quad (1.21)$$

or similar distance function, based on Cartesian coordinates and subject to the limitation that a room occupies only

one location. Of course Eq. 1.21 responds to only one special relation between rooms. The other relations such as direct adjacency, sightliness, specific distance constraints and others are combined with the distance relation.

The basic operation for the generation of alternative solutions for space planning problems involves the relocation of a single domain or a set of domains. The Cartesian points represented by a domain were altered though their attributes, shapes and dimensions remained invariable. A basic test involved in all relocation operations is an evaluation of a proposed empty domain so as to determine if the space required to locate the rooms is completely disjointed from all other filled domains. This can be considered as a test to ascertain whether an empty domain will completely encompass the solid domain it is receiving. Alternatively, it can be a test to check whether the solid domains already located have points in common with the solid being located. Furthermore, any alternative space planning can be generated and evaluated if the following capabilities are available and facilitated.

- (1) Representation of domains of any shape
- (2) Determination of any dimensions or attributes of a represented domain.

- (3) Identification of any desired set of adjacent domains
- (4) Determination of any dimensions or attributes of a set of adjacent domains.

The above capabilities seem to well define those needed for space planning. C. M. Eastman⁽¹⁸⁾ compares four ways of the data structure of space planning representation among themselves: plain arrays, hierarchical array, string representation, and adjacency structures.

Plain arrays use the two-dimensional array. In this representation, each subscripted variable in a predefined array represents a rectangular unit area, that is, domain. The subscripts of the domain provide the definition of its x and y Cartesian coordinates. The value of a variable in the array denotes its state. The domains used by an array to represent the room and the actual array used to represent the room are shown in Figs. 1.7, and 1.8(a) and (b).

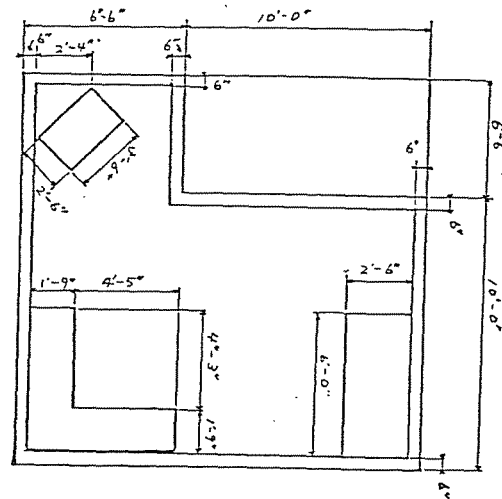


Fig. 1.7 Orthographic
projection of the plan
of a room

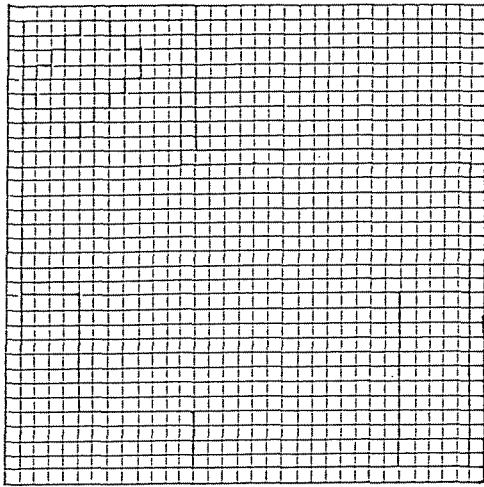


Fig. 1.8 (a) The domain represented by an array

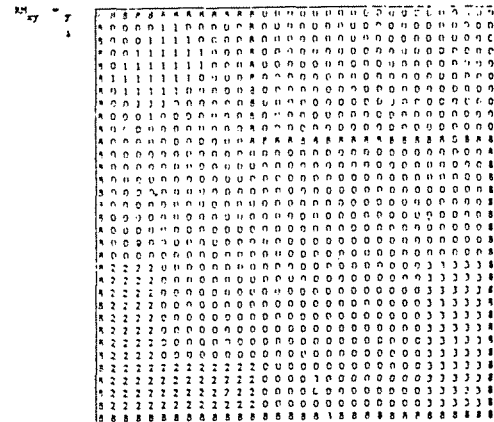


Fig. 1.8 (b) The actual array used to represent a room

Variations of the plain array representation have been developed that lessen memory requirements and processing time. One of them has been developed at Stanford Research Institute for use as a robot's internal representation of the world. Instead of a single predefined grid, they use a method of subdividing any given rectangular domain into 4 x 4 grid cells. Each cell can be further subdivided into 4 x 4 grids recursively. Homogeneous domains are not subdivided. Subdivision is only required at the boundaries of elements. A diagram of domains expressed in such a representation is presented in Fig. 1.9.

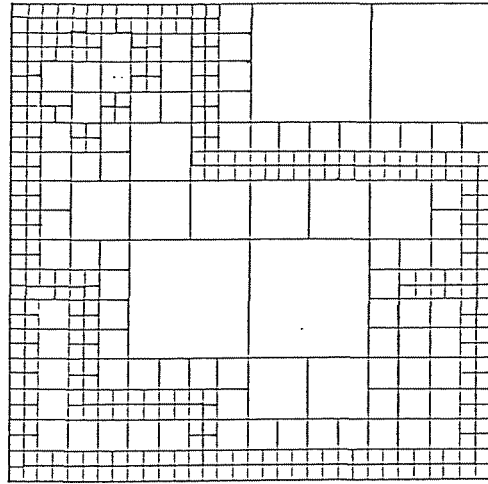


Fig. 1.9 The hierarchical set of domains defined by
the SRI array

In a string representation, domains are defined according to a particular strategy for collecting homogeneous point locations. The state of a particular point location is determined by summing row prefixes in the y coordinate and scanning the appropriate rows in the x coordinate. The application of this domain definition technique to the floor plan shown in Fig. 1.7 produces the actual data structure as shown in Fig. 1.10.

The letter suffix expresses the state of blocks:
W, A, B, C, E represent walls, objects A, B and C, and
empty space respectively. The prefix defines the
vertical extent of a set of domains. Thus a prefix,
along with each symbol string within commas, defines a
domain.

A single rule of adjacency in both coordinates
that allow a single accessing rule give a new
representation. Especially the rule should be applied
so that only single domain may be adjacent to each
other in either coordinate. This is called an adjacency
structure. This example produced in Fig. 1.7 is shown
in Fig. 1.11.

```
.5(6.5W, 10.0E)
.5(.5W, 2.1E, .5A, 2.9E, .5W, 10.0E)
.5(.5W, 1.6E, 1.4A, 2.5E, .5W, 10.0E)
.5(.5W, 1.1E, 2.3A, 2.1E, .5W, 10.0E)
.5(.5W, .6E, 3.1A, 1.5E, .5W, 10.0E)
.5(.5W, .3E, 3.1A, 2.1E, .5W, 10.0E)
.5(.5W, .5E, 2.4A, 2.6E, .5W, 10.0E)
.5(.5W, 1.1E, 1.3A, 3.1E, .5W, 10.0E)
1.0(.5W, 5.5E, .5W, 10.0E)
1.0(.5W, 5.5E, 10.5W)
4.5(.5W, 15.5E, .5W)
4.3(.5W, 1.8B, 11.2E, 2.5C, .5W)
1.2(.5W, 6.0B, 7.0E, 2.5C, .5W)
.5(16.5W)
```

Fig. 1.10 The actual
data structure used
in the string repre-
sentation

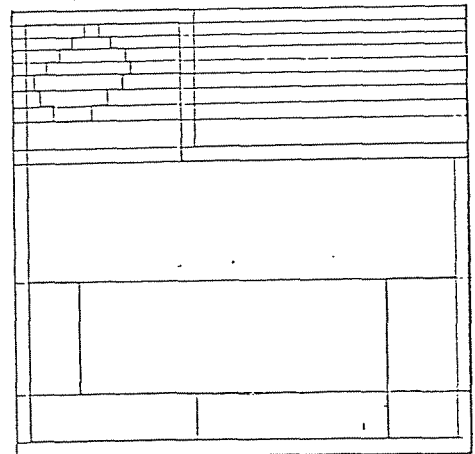


Fig. 1.11 An example of
simple adjacency structure

The adjacency structure with variable sized domain is the algorithm extended from the two-dimensional array. In this algorithm, the domain size corresponding to the element of the array is variable. This is shown in Fig. 1.12.

Table 1.1 shows the comparison of four space planning representations.

In this paper, the space representation is presented by "Formulated Pattern" developed by Prof. N. Okino, and the algorithm needed for graphic processing of space allocation is discussed in chapter 7.

$x_{11} =$

x \ y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	.5	.5	.5	.5	.5	.5	.5	.5	1.5	.5	7.0	2.5	.5	
1	.5	8	8	8	8	8	8	8	8	8	8	8	0	0	0
2	.5	8	0	0	0	0	1	1	0	0	0	8	0	0	0
3	.5	8	0	0	0	1	1	1	0	0	0	8	0	0	0
4	.5	8	0	0	1	1	1	1	1	1	0	8	0	0	0
5	.5	8	0	1	1	1	1	1	1	1	0	8	0	0	0
6	.5	8	0	1	1	1	1	1	1	0	0	8	0	0	0
7	.5	8	0	1	1	1	1	1	0	0	0	8	0	0	0
8	.5	8	0	0	1	1	1	0	0	0	0	8	0	0	0
9	.5	8	0	0	0	1	0	0	0	0	0	8	0	0	0
10	.5	8	0	0	0	0	0	0	0	0	0	8	0	0	0
11	.5	8	0	0	0	0	0	0	0	0	0	8	8	8	8
12	4.5	8	0	0	0	0	0	0	0	0	0	0	0	0	8
13	4.25	8	2	2	2	2	0	0	0	0	0	0	0	3	8
14	1.25	8	2	2	2	2	2	2	2	2	2	2	0	3	8
15	.5	8	8	8	8	8	8	8	8	8	8	8	8	8	8

Fig. 1.12. An array of variable domains

Table 1.1 Comparison of four space planning representation

	REPRESENTATION			
	Plain array	Hierarchical array	String representation	Adjacency structure
Number of domains required to represent example arrangement	1089	411	65	225
Greatest error in representation of elements not parallel to coordinate	8.5	8.5	4.2	8.5
Greatest error in representation for elements parallel to coordinate	6.0	6.0	0.0	0.0
Domains require definition?	no	yes	yes	yes
Structure similar in both coordinates?	yes	yes	no	yes

1.3.5 C.A.D. in L.S.I. Design

Today the production method for Large Scale Integrated circuits adopts the following design process. Each unit cell, which is the basic element structure of the circuit being integrated by transistors, diode inter connections and so on, is allocated within the circuit area allowed for wiring it, and then the unit cells are wired mutually among themselves. As the number of the cells and the needs of the circuit become larger and more sophisticated, the determination of the location and wiring of the cells become more difficult. And the difficulty exceeds human ability. Therefore, the design process must be automated. This problem is the same as the one mentioned in section 1.3.3 in that the performance measure must be suited for minimizing the wiring area.

As the wiring depends on the predetermined placement of the cells, wiring automation procedure is developed first. The recent works⁽¹⁹⁾⁽²⁰⁾⁽²¹⁾ treat the placement problem of the cells by developing building-block methods. Fig. 1.13 and Fig. 1.14 show the examples of the results of the building block methods. The way shown in Fig 1.13 is called a bottom-up method and the way shown in Fig. 1.14 is called a top-down method.

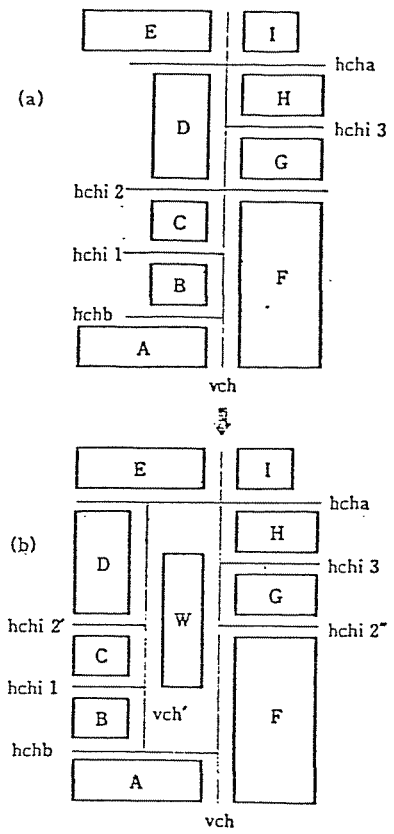
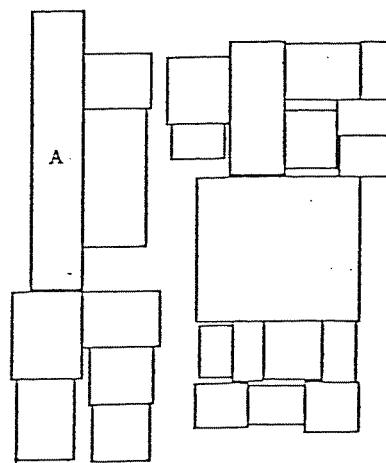
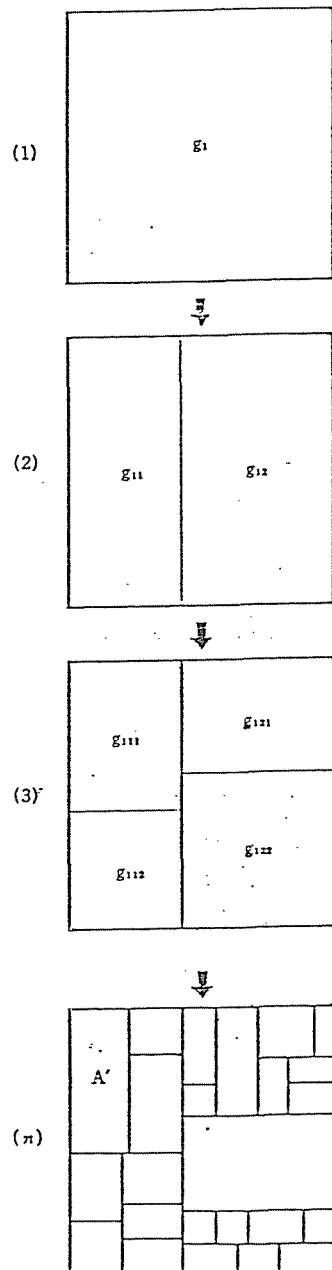


Fig. 1.13 Building block method
(Bottom-up method)

Fig. 1.14 Building block method (Top-down method)

On the other hand, the following problem occurs in the manufacturing process of printed circuit masks. The masks are divided into small rectangles and developed by the machine called a pattern generator which can produce any size of rectangular masks. The number of rectangles divided to produce a mask becomes so large that it is necessary to reduce the possible number of rectangles for developing the mask. Therefore the problem can be described as

$$\text{minimize} \quad K \quad (1.22)$$

$$\text{subject to} \quad R = \bigcup_{i=1}^k S_i \quad (1.23)$$

$$\bigcap_{i=1}^k S_i = \phi. \quad (1.24)$$

Where R is the domain of the given mask and S_i is the divided rectangular cell. In this problem, the resource is considered as the mask shape, the materials as the rectangular cells and the performance measure to be adopted is a minimum partition. In originating this problem, T. Oyamada⁽²²⁾ takes a consideration into a minimum partitioning theorem and develops a heuristic mask division procedure by the use of graph theory. But this procedure does not yield an optimum partition. In chapter 6, the optimum partition procedure will be discussed and presented.

1.4 Conclusion

This chapter attempts to describe a common recognition of the problems. For this purpose, the generalization of the models of the problem and the survey of the problems are attempted.

It is known that the various types of the problems occur depending on the situations and their circumstances and that the suitable methods must be developed for them. For the recognition of the problems, we must approach from the two phases: mathematical programming and graphic processing. Though there are scarcely discussions arise on the graphic processing except in the architecture field, the effort for the graphic processing in the space allocation problem must be made. This is because the problem is always restricted by the geometry of the resource and the materials. This implies that the procedure for the graphic processing must be developed and established for the space allocation problem. Such processing involves the method of geometric space description, the data structure of the space placement, the space recognition method, etc.

As to the mathematical programming, the most of the methods surveyed, except for the Codd's rule, are based on the iterative calculation process, each step gradually

bringing the solution to the optimum. But the improvement of the solution is very slow and a number of memories and huge calculation time are required for a computer. The methods being presented are based on P. C. Gilmore and R. E. Gomory's work in most cases. But they essentially treat the problem in one-dimensional form, so their methods become less efficient than those that treat the problem in two- and three-dimensional forms.

The most of the problem formulations for the space allocation problem such as the Knapsack problem is known as the N-P complete, which means there is not an algorithm being found out by the polinomial order calculation time for the program description size n . Therefore, the heuristic methods are developed, which give optimum or suboptimum solution by straight or small calculation. In so developing, the problem must be thought over with regard to its situation and the necessity for the complete optimum solution.

From the viewpoint mentioned above, the followings should be taken into consideration:

- (a) Compare the situation of the problems already studied with the one occurring.
- (b) Check the necessity of 100 per-cent optimization carefully.

- (c) Analyze the software and hardware facilities available.
- (d) Consider both treatment of mathematical programming and graphic processing.

The subsequent chapters present new approaches to the space allocation problem, considering the above items.

References

1. Okino, N. et. al., "Formulated Pattern for Computer-Aided-Design and Computer-Aided Manufacturing," J.S.P.E., Vol.5, No.4, 1971.
2. Gilmore, P.C., and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem, Part I," Opns. Res., Vol.9, P.849, 1961, "Part II," Opns. Res., Vol.11, No.6, p.863, 1963.
3. Gilmore, P.C., and R.E. Gomory, "Many Stage Cutting Stock Problems of Two and More Dimensions," Opns. Res., Vol.13, No.1, P.94, 1965.
4. Gilmore, P.C., and R.E. Gomory, "The Theory of Computation of Knapsack Functions," Opns. Res., Vol.14, No.6, p.1045, 1966.
5. Kolesar, P.J., "A Branch and Bound Algorithm for the Knapsack Problem," Man.Sci., Vol.13, No.9, p.723, 1967.
6. Lawrie, N.L., "An Integer Programming Model of a School Timetabling Problem," Comp. J, Vol. 12, p.207, 1969.
7. Brown, A.R., "Selling Television Time: An Optimization Problem," Comp. J, Vol.12, P.201, 1969.

8. Brown, A.R., "Optimum Packing and Depletion,"
Jeffres and Hill Limited, 1971.
9. Coffman, E.G. Jr., "Computer & Job/Shop
Scheduling Theory," John Wiley & Sons, 1976.
10. Johnson, S.M., "Optimum Two- and-three-Stage
Production Schedules with Set-up Time Included,"
Nav. Res. Log. Quart., Vol.1, p.61, 1954.
11. Ignal, E.J., and L. Schrage, "Application of
the Branch-and-Bound Technique to Flow-Shop
Scheduling Problems," Opns. Res., Vol.13, p.400,
1965.
12. Nabeshima, I., "Theory of Scheduling," Morikita
Syuppan, 1974.
13. Kubo, H., "A Highly-Efficient Branch-and-Bound
Method for Combinatorial Optimization Problems,"
Ph.D Thesis of Hokkaido University, 1976.
14. Simmon, D.M., "One-Dimensional Space Allocation:
An Ordering Algorithm," Opns. Res., Vol.17, P.812,
1969.
15. Codd, E.F., "Multiprogram Scheduling - Intro-
duction and Theory," Comm ACM, Vol.3, No.6, p.347,
1960.
16. Wilson, R.C., "A Packing Problem," Man. Sic.,
Vol.12, No.4, P.B-135, 1965.

- 17 . Eastman, C.M., "Representations for Space Planning," Comm. ACM, Vol.13, No.4, p.242, 1970.
- 18 . Eastman, C.M., "Preliminary Report on a System for General Space Planning," Comm. ACM, Vol.15, No.2, P.76, 1975.
- 19 . Tanaka, Z., "Layout Automation for Developing LSI," Nikkei Electronics, Vol.9, 1979.
- 20 . Woude, M., "CALDI: Computer-Aided Layout Design of I²L, Digital Process," Vol.2, p.143, 1976.
- 21 . Dunlop, A.E., "SLIP: Symbolic Layout of Integrated Circuits with Compaction," CAD, Vol.10, No.6, p.387, 1978.
- 22 . Oyamada, T., "Minimum Partitioning of a LSI Artwork Pattern," I.P.S.J., Vol.16, No.7, 1975.
- 23 . Weglaz, J., "Multiprocessor Scheduling with Memory Allocation-A Deterministic Approach," IEEE Computers, Vol.C-29, No.8, p.703, 1980.
- 24 . Foulds, L.R., S.M. Perera and D.F. Robinson, "Network Layout Procedure for Printed-Circuit Design," CAD, Vol.12, No.1, p.177, 1980.
- 25 . Hadley, G., "Nonlinear and Dynamic Programming," Addison-Wesley, 1972.

- 26 . Claude, M. Jr., "Mathematical Programming,"
John-Wiley and Sons, 1970.
- 27 . Fishburn, P.C., "Utility Theory for Decision
Making", John-Wiley and Sons, 1970.
- 28 . Roberts, F.S., "Discrete Mathematical Models,"
Prentice Hall, 1976.

2. A practical New Solution to Flow-Shop Scheduling Problem---1 $\frac{1}{2}$ -Dimensional Space Allocation Problem---

2.1 Introduction

In an optimum packing situation, a problem in one and a half dimensional space allocation arises. A typical problem is simply to pack all the bricks into the holes without any protruding. This is called "Optimum Packing Problem". The problem can also arise if we consider "processors" instead of holes and "jobs" instead of bricks. This is clearly the same as the packing problem and it is called "Job Shop Scheduling Problem". Assuming a particular packing situation where a satisfactory feasible solution has been found, there may be a further requirement to achieve the "best" arrangement. In the job shop scheduling, two well-known "bests" are:

- (a) To minimize a maximum processing time among the given processors (minimal-length schedule)
- (b) To minimize a mean flow type of the processors (mean flow-time schedules)

One of the job scheduling problems is encountered in a flow shop type production line and a computer

task system. If we treat the computer task system, a program is processed by a number of distinct machines during its passage through a computer system. Since all programs pass through the input, execution, and output phase, a task system can be viewed as a set of chains of m tasks, where i -th task in the chain must be executed on processor P_i . Determining a minimal-length schedule in such a situation is referred to as the flow shop problem. Jobs are regarded as tasks in this problem.

There are a number of studies on the flow shop scheduling. The first method to the problem is presented by S.M. Johnson⁽⁴⁾ and his method is called "Johnson's rule". This rule is based on an exact analysis on n -chain, 2-processor flow shop scheduling problem. E. Ignal and L. Schrage⁽⁵⁾ apply a branch and bound method to the flow shop scheduling problem after almost ten years since Johnson's study. Then, there are some studies, the purpose of which is to improve an efficiency of the branch and bound method. (6)(7)

However, the method based on the branch and bound requires the computer memory in use to be so large, and a lot of computing time is needed for solving the problem. Therefore, it is not easy to solve a practi-

cally large scale of the flow shop scheduling problem.

This chapter presents a method to find a solution for an n -chain and m -machine flow shop scheduling problem admitting no task passing. By using this method, it is possible to get an approximate solution for a problem of a practical scale. In relation to 2 and 3 processor problems, the method produces the same results as Johnson's method does.

This method proceeds as follows. At first, the binary relation of all the jobs is examined in connection with the precedence relation. From the results of the examination, the preference relation can be derived. The latter relation makes it possible to draw a directed graph. By using this directed graph, it is possible to find out a utility function for each job to be calculated. The values obtained in this way are then compared, and the chains are scheduled in the order of value. In addition, this chapter deals with experiments in which this method is applied to flow shop scheduling to obtain an optimum solution. The results by numerical experiments prove that the method gives a practical solution.

2.2 A Formulation of the Problem

For the m -processor flow shop problem, let the task system (\mathcal{T}, \prec) consist of n chains c_1, c_2, \dots, c_n , where each chain has m tasks A_i, B_i, \dots, Z_i , $A_i \prec B_i \prec \dots \prec Z_i$. \prec implies task A_i must be executed on processor P_1 and once A_i finishes, task B_i must be executed on processors P_2 and so is this relation till task Z_i must be executed on processors P_m .

Fig. 2.1 shows an example. In Fig. 2.1 tasks Z_1, Z_2, \dots, Z_n on processor P_m, \dots , and B_1, B_2, \dots, B_n on processor P_2 are executed in the same order as A_1, A_2, \dots, A_n are executed on P_1 . This is shown in Fig. 2.1 as a gantt chart.

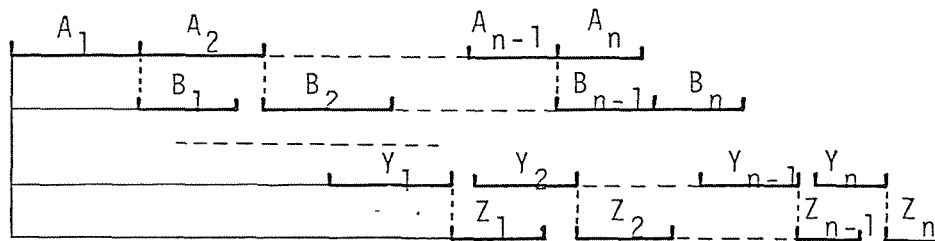


Fig. 2.1 A gantt chart of flow shop scheduling.

Let us define some notations before formulating the flow shop scheduling problem as follows:

σ_k A set of k chains to be scheduled.
 $q(\sigma_k, j)$ A schedule length, which is obtained
 by k chains scheduling, on processor j .
 t_{kj} A processing time on processor j
 in the k -th chain.

Then the minimal length flow shop scheduling problem is formulated to find out a sequence of chains as

$$\min. \quad F = q(\sigma_n, m), \quad (2.1)$$

$$\text{subj. to} \quad q(\sigma_k, 1) = q(\sigma_{k-1}, 1) + t_{k1}, \quad (2.2)$$

$$q(\sigma_k, j) = \max (q(\sigma_k, j-1), q(\sigma_{k-1}, j)) \\ + t_{kj}$$

$$j = 2, 3, \dots, m \quad (2.3)$$

$$k = 1, 2, \dots, n,$$

where $q(\sigma_0, j) = 0, \quad j = 1, 2, \dots, m$.

$q(\sigma_n, j)$ in Eq. 2.3 implies a finished processing time on processor j . This formulation is derived from Fig. 2.1 easily.

2.3 A Utility Function under a Weak Order Relation in the Flow Shop Scheduling

A utility function for chains is established to the flow shop scheduling. In the discussion, a weak order is assumed for a set of chains in the relation of preceedingly processing between any two chains, and the weak order is mapped to a directed graph. The utility function of chains is derived from a measurement of nodes in the graph.

2.3.1 Preference as a Weak Order

A binary relation R on a set X is a set of ordered pairs (x, y) with $x \in X$ and $y \in X$, where $X = \{1, 2, \dots, n\}$. We write xRy to mean $(x, y) \in R$. The binary relation will be assumed to have certain properties.⁽⁶⁾

- p1. reflexive if xRx for every $x \in X$,
- p2. irreflexive if not xRx for every $x \in X$,
- p3. symmetric if $xRy \Rightarrow yRx$, for every $x, y \in X$,
- p4. asymmetric if $xRy \Rightarrow$ not yRx , for every $x, y \in X$,
- p5. antisymmetric if $(xRy, yRx) \Rightarrow x = y$ for every $x, y \in X$,

- p6. transitive if $(xRy, yRz) \Rightarrow xRz$, for every $x, y, z \in X$,
- p7. negative transitive if $(\text{not } xRy, \text{not } yRz) \Rightarrow \text{not } xRz$, for every $x, y, z \in X$,
- p8. connected or complete if xRy or yRx for every $x, y \in X$,
- p9. weakly connected if $x \neq y \Rightarrow (xRy \text{ or } yRx)$ throughout X .

A binary relation that has or is assumed to have certain properties is given a special name. One is a weak order. The weak order is defined as follows:

Definition: A binary relation R on a set X is a weak order if R on X is asymmetric and negatively transitive.

Now, taking preference \prec as basis (read $x \prec y$ as x is less preferred than y , or y is preferred to x), we shall define indifference \sim

$$x \sim y \quad (\text{not } x \prec y, \text{not } y \prec x). \quad (2.4)$$

When preference relation \prec on X is a weak order, the following theorem is well known.

Theorem 2.1 [P.C. Fishburn] Suppose \prec on X is a weak order, being asymmetric and negative transitive. Then

- a. exactly one of $x \prec y$, $y \succ x$, $x \sim y$ holds
for each $x, y \in X$;
- b. \prec is transitive;
- c. \sim is an equivalence (reflexive, symmetric,
transitive);
- d. $(x \prec y, y \sim z) \Leftrightarrow x \prec z$, and $(x \sim y, y \prec z) \Leftrightarrow x \prec z$;
- e. \succ is transitive and connected.

2.3.2 An Order-Preserving Utility Function

It is known that a utility function exists based on the next theorem as well.

Theorem 2.2 [P.C. Fishburn] If \succsim on X is a weak order and x/\sim is countable, then there is a real-valued function u on X such that

$$x \prec y \Leftrightarrow u(x) < u(y), \text{ for all } x, y \in X. \quad (2.5)$$

x/\sim in the theorem means the set of equivalence classes of x under \sim .

The utility function u in 2.5 is said to be order-preserving since the numbers $u(x)$, $u(y)$, ... as ordered by $<$ faithfully reflect the order of x , y , ... under \succsim . Clearly, if (2.5) holds, then

$$x \succ y \Leftrightarrow v(x) < v(y), \text{ for all } x, y \in X,$$

for a real function v on X if and only if $[v(x) < v(y) \Leftrightarrow u(x) < u(y)]$ holds throughout X .

Under the conditions of Theorem 2.2, 2.5 implies that, for all $x, y \in X$, $x \sim y \Leftrightarrow u(x) = u(y)$, and $x \preceq y \Leftrightarrow u(x) \leq u(y)$.

2.3.3 Precedence Relation on the Flow Shop Scheduling

In order to apply the results of preference relation to the flow shop problem, we adopt a precedence relation as a preference relation. The precedence relation is defined as follows:

If a chain x must be processed before a chain y , we call that the relation between x and y is precedence, where $x, y \in C$ and C is a set of chains.

When the chain x and y is precedence relation and x is preceedingly processed before y , we denote it as $y \succ x$, for $x, y \in C$. If the precedence relation is a weak order, threre exists a utility function u which is order-preserving by reflecting the order of chains from 2.5. Therefore, establish-
ing the utility function for the chains in the flow

shop scheduling, we may make a schedule as the order of chains by rearranging the value of utility functions corresponding to chains from the largest one to the smallest one

2.3.4 A Directed Graph and Tournament

Let us define V as a set of vertices and A as a set of ordered pairs of elements of V . A is called the set of arcs. Then, a pair (V, A) is called a directed graph or digraph D . The directed graph is called a tournament when for all $x \neq y$ in V , (x,y) is in A , or (x,y) is in A but not both.

Regarding a set of chains X as the set of vertices V and a precedence relation among all the chains as the set of A , the directed graph D is made up from the preference relation of chains in the flow shop scheduling.

The next theorem is known about the tournament.

Theorem 2.3 [Reclei] Every tournament (V, A) has a complete simple path (a hamiltonian path).

This complete simple path corresponds to a schedule which satisfies the preference relation.

In this way solving the flow shop scheduling problem is transformed into finding the complete simple path on the tournament.

If the precedence relation is a weak order, being asymmetric and negative transitive, a complete path obtained from the tournament has the property shown in Theorem 2.4.

Theorem 2.4 The tournament (V, A) is acyclic when and only when the precedence relation is the weak order.

We prove this theorem according to J.G. Kemmeny⁽²⁾ and J.L. Snell.

Proof. Suppose that the tournament has a cycle, $(a, b), (b, c), \dots, (j, k), (k, a)$. Since the precedence relation is the weak order, a path must be transitive [Theorem 2.1].

By applying this property iteratively to the cyclic path $(a, b), (b, c), \dots, (j, k), (k, a)$, a preceeds a . As the tournament is irreflexive, it contradicts irreflexive condition. Thus, the tournament derived from the precedence relation of the chains has no cycle. On the other hand, suppose that the tournament has no cycle, then a does not preceed a . Because the tournament has cycle if a preceeds a . If $b \rightarrow a$ and

$a \succ b$, a path exists from a to b . It means cycle.
 If $b \succ a$ and $c \succ a$, a path b to a and a path c to a
 do not intersect. Because the tournament has
 cycle if they intersect. Therefore the tourna-
 ment has no cycle, if the precedence relation is
 the weak order. Q.E.D.

The utility function is established based on the
 complete simple path of the tournament and the
 above theorem is used in the following section.

2.3.5 A Utility Function in the Tournament

We can now introduce the four conditions which
 a utility function will be asked to satisfy. Let
 $u(x)$ be the utility function of the chain x . The
 conditions for u are as follows:

Condition 1. $u(x)$ is always an integer number.

Condition 2. If the chain x has no succeeding
 chains, $u(x) = 0$.

Condition 3. If $x \succ y$, then $u(x) \leq u(y)$.

Condition 4. If, without otherwise changing
 chain order, we add a new chains
 to the partial path which succeeds
 to the chain x , then the value
 $u(x)$ increases.

By the above conditions, the value of the utility function $u(x)$ is decreasing in the order of chains on the optimum schedule, and zero at the last chains, and positive number.

Now, we establish the utility function which satisfies the above conditions.

Theorem 2.5 Let $u(x)$ be a number of chains succeeding to chain x . $u(x)$ is the utility function which satisfies Conditions 1 to 4.

Proof. Let us define an indirect precedence as that the chain x is preceedingly processed before the chain i , and an direct precedence as that the chain x is processed just one before the chain i . And we denote them as \prec and \succ respectively.

Suppose that the chain x cannot be processed before the chain i ($i \neq x$). By applying theorem 2.4, we obtain the followings.

- 1° $i \succ x$ because $x \succ i$ is not allowed.
- 2° If $x \succ k$, for all the chains k , then
 $i \succ k$ because $x \succ k \succ i$ is not allowed.

From 1° and 2°, the number of chains succeeding to the chain i is larger than the number of chains succeeding to x . Namely $u(i) \geq u(x)$.

Now, let us examine the number of chains succeeding to each of all the chains $\{1, 2, \dots, x\}$, and set them to $u(1), u(2), \dots, u(n)$ respectively. Then suppose $u(x)$ is the largest number among $u(1), u(2), \dots, u(n)$. Then, it is proved that the chain x is the first one to be processed. The proof is as follows. If there is the chain i which may not succeed to the chain x , $u(i) > u(x)$ because of 1° and 2°. This contradicts $u(x)$ is the maximum number. Therefore, the chain x must be processed first.

Next, we remove the chain x and repeat the same procedure. In this way, we obtain the decreasing series of value $u(x)$. The chain processed at last has no succeeding chains. If we set this chain to the chain z , $u(z) = 0$.

If we add a certain chain to a partial path succeeding to the chain x , the number of chains succeeding to the chain x increases by one. Thus, $u(x)$ increases.

Therefore, the utility function $u(x)$ satisfies conditions 1 - 4. Q.E.D.

The utility function established in this way is a measure of vertices in the directed graph. And it is expressed by

$$u(x) = \sum_k k \cdot n_k, \quad (2.6)$$

where $k = 1$ and n_k is the number of vertices succeeding to x in a level k . Eq. 2.6 is suggested by F. Harary.

2.4 Algorithm

If a precedence relation in the flow shop scheduling is a weak order, a utility function is figured out in the following four steps and an optimum schedule is obtained.

1° Let $q(\sigma_2(i, j), m)$ be a processing time when the chain i is preceedingly processed before the chain j for any of $i, j \in C$. Then, set t_{ij} as

$$t_{ij} = q(\sigma_2(i, j), m), \quad i, j = 1, 2, \dots, n \quad (2.7)$$

$$t_{ij} = 0, \quad i = j, \quad i, j = 1, 2, \dots, n \quad (2.8)$$

where matrix $T = \{t_{ij}\}$ is called a cost matrix.

2° Make up a precedence relation matrix $W = \{w_{ij}\}$ by

$$\begin{aligned} &\text{if } t_{ij} > t_{ji}, \text{ then } w_{ij} = 0 \text{ and } w_{ji} = 1 \\ &\quad i, j = 1, 2, \dots, n \quad i \neq j, \\ &\text{if } t_{ij} = t_{ji}, \text{ then } w_{ij} = w_{ji} = 1, \\ &\quad i, j = 1, 2, \dots, n \quad i \neq j, \end{aligned}$$

if $t_{ij} < t_{ji}$, then $w_{ij} = 1$ and $w_{ji} = 0$.
 $i, j = 1, 2, \dots, n \quad i \neq j$.

$$3^\circ \quad \alpha_i = \sum_{j=1}^n w_{ij}, \quad i = 1, 2, \dots, n.$$

4° Set $u(j) = \alpha_i$, then arrange $u(i)$ from the largest number of $u(i)$ to the smallest one. After arrangement, the order of chains corresponding to the order of $u(i)$ becomes an optimum schedule.

In the algorithm, the precedence relation is determined by comparing a processing time of the chain paired order (i, j) with one of the chain paired order (j, i) . It is discussed in the next section under what conditions the precedence relation in the flow shop scheduling is successful. But even if the precedence relation is broken up, the algorithm becomes still available by improving the steps 3 and 4 to the followings.

3'° Calculate α_i and β_i as

$$\alpha_i = \sum_{j=1}^n w_{ij}, \quad \beta_i = \sum_{j=1}^n w_{ji} \quad \alpha_i,$$

$$i = 1, 2, \dots, n.$$

4'° Set $u'(i) = \alpha_i + \beta_i$. Then, execute the procedure as described in 4°.

In step 3', β_i means the number of chains which indirectly succeed to the chain i . Therefore, the utility function $u'(i)$ in the step 4' presents the measure of vertices as arcs. Furthermore, since $u'(x)$ is the function by a simply increased transformation of $u(x)$, $u'(x)$ is the utility function as well that can determine the schedule.

The memory size in a computer for the proposed algorithm is requested to be $n(n+m)$ words which is summed by an array $n \cdot m$ words for the input data and an array $n \cdot n$ words for the cost matrix. The cost matrix may be used as the precedence relation matrix. So, even if the number of chains n is 100 and the number of processors m is 10, the memory size requested becomes about $100(100+10)$ words. It is almost 11K words. This size is much less than the size requested by a branch and bound method and is small enough to make use of for a mini-computer.

2.5 Validity of Algorithm for Flow Shop Scheduling

Let us discuss a validity of an algorithm proposed in 2.4 for a flow shop scheduling. It is executed by examining whether a weak order assumption for a precedence relation is broken up. As to an irreflexivity, it is obvious that an irreflexive condition is satisfied. Therefore, we only look up transitivity.

When the number of processors is two, S.M. Johnson points out that the transitive condition is satisfied in the flow shop scheduling problem. Furthermore he establishes his rule based on the transitivity. As seen in Johnson's rule, the transitivity exactly stood so that the proposed algorithm gives an optimum scheduling when the number of the processors is two.

When the number of processors is three, the transitive condition is not satisfied. Such an example is shown in Table 2.1. S.M. Johnson applies his rule to the problem only when the transitivity is satisfied. If not satisfied, his rule gives an approximate solution. So does the proposed algorithm.

When the number of processors is m , there are few rules to determine the optimum schedule by a dispatching rule such as Johnson's. The proposed algorithm reaches the exact optimum schedule only when the transitivity is satisfied. In general, it gives near optimum solution. It is considered that the proposed algorithm is the extension of Johnson's rule to m processors.

2.6 Numerical Experiments

In order to show a validity of the proposed method, numerical experiments are executed by a mini-computer OKITAC 4500-C (40K words). A program is coded with FORTRAN. Task processing time to be used in the experiments are generated by uniform random integer numbers with one digit.

Experiment 1) The proposed method is applied to the 10-chain 3-processor flow shop scheduling problem with minimal length, which is represented by E. Ignal and L. Schrage as the most time consuming sample for a branch and bound method. The input is shown in Table 2.2. The optimum schedule length is 66 and the sequence of chains

is (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).

The proposed method gives an approximate solution whose schedule length is 67 and sequence of chains is (1, 2, 3, 5, 7, 6, 8, 9, 10). It takes 1.15 seconds to get the solution. Though Ignal's branch and bound method is also coded and applied to this problem, it is too many branching to solve the problem by the computer equipped.

Table 2.1 An example of cahins which does not satisfy the transitive condition

Chain No. Processor	1	2	3
P_1	3	25	45
P_2	22	42	56
P_3	2	22	40

Table 2.2 E. Ignal and L. Scharge's example

Chain No. Processor	1	2	3	4	5	6	7	8	9	10
P_1	1	5	7	8	3	7	9	8	6	3
P_2	2	9	6	9	2	10	7	9	1	1
P_3	9	7	8	9	3	4	7	4	3	1

Experiment 2) The proposed method is tested for 4- and 5-chain and 3- and 4-processor problems with minimum length and their results are compared with the optimum solution obtained by a complete enumeration method. The comparison is shown in Table 2.3. In Table 2.3, the worst solution means the smallest value of an approximate solution ratio among the test sample ones. The approximate solution ratio (A.S.R.) is defined here as

$$\text{A.S.R.} = \frac{\text{the number of schedules whose length are shorter than the schedule length of the proposed method}}{\text{all the number of possible schedules}}$$

Experiment 3) On 4-chain, 5-chain and 10- to 50-processor flow shop problems, the same experiments as the experiment 2 are made. This test is intended to understand effects of many processors. The results are shown in Table 2.4.

Experiment 4) On 10-chain and 3-, 4- and 5-processor, 20-chain and 3-, 4- and 5-processor, and 30-chain and 3-, 4- and 5-processor of the flow shop scheduling problems, comparisons between the solution scheduled by the proposed method and one not scheduled are printed out on the line printer paper as diagrams. The reason why A.S.R. is not used is that the enumeration method is unavailable for many chains. The results are shown in Figs. 2.4 - 2.12.

Table 2.3 Results in experiment 2.

Problem	The Num. of tested examples	The num. of optimum solutions obtained	Worst solution (%)	A.S.R. (%)
4-chain 3-proc.	50	41	72.9	98.2
5-chain 3-proc.	31	20	61.7	94.2
4-chain 4-proc.	37	28	87.5	98.0
5-chain 4-proc.	34	15	51.6	94.0

Table 2.4 Results in experiment 3.

Problem	The num. of tested examples	The num. of optimum solutions obtained	Worst solution (%)	A.S.R. (%)
4-chain				
10-proc.	10	7	66.7	90.9
20-proc.	10	3	33.3	78.4
30-proc.	10	2	45.8	76.8
40-proc.	10	2	54.2	89.3
50-proc.	10	3	47.7	83.1
5-chain				
10-proc.	10	8	56.6	96.5
20-proc.	10	3	48.5	85.6
30-proc.	10	1	76.7	93.7
40-proc.	10	1	35.4	89.6
50-proc.	10	2	68.3	89.2

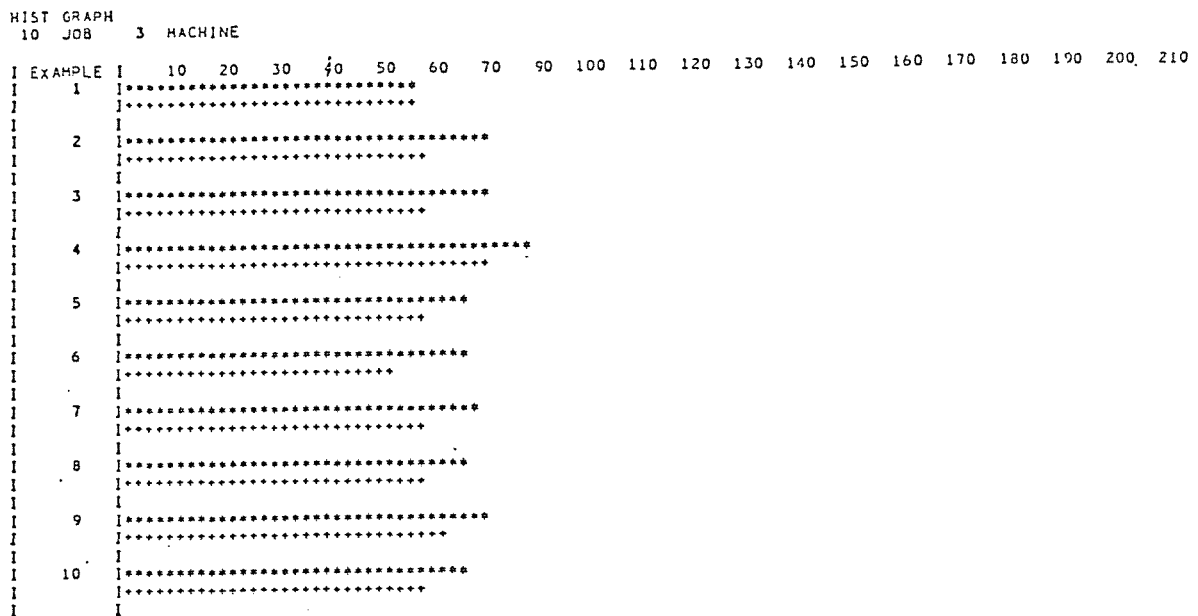


Fig. 2.4 10-chain 3-processor experiments

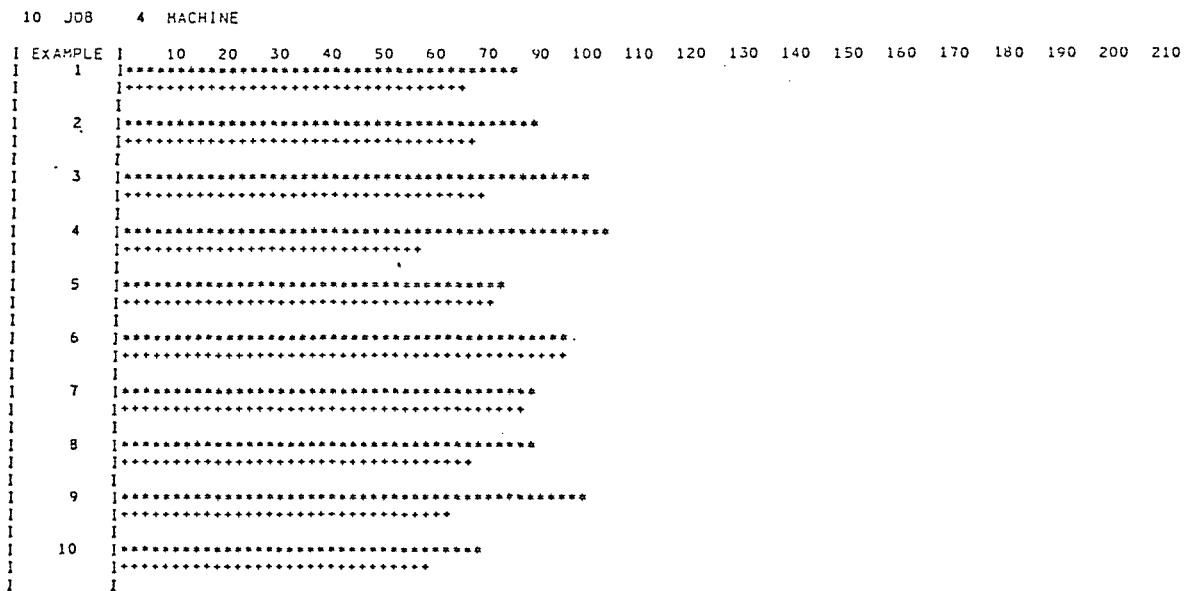


Fig. 2.5 10-chain 4-processor experiments

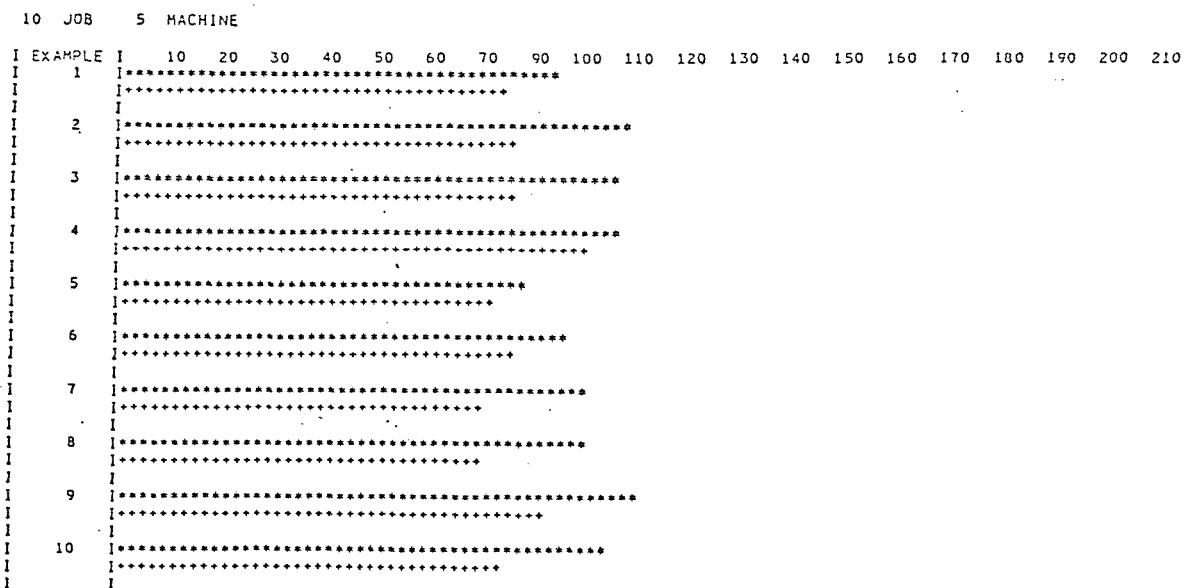


Fig. 2.6 10-chain 5-processor experiments

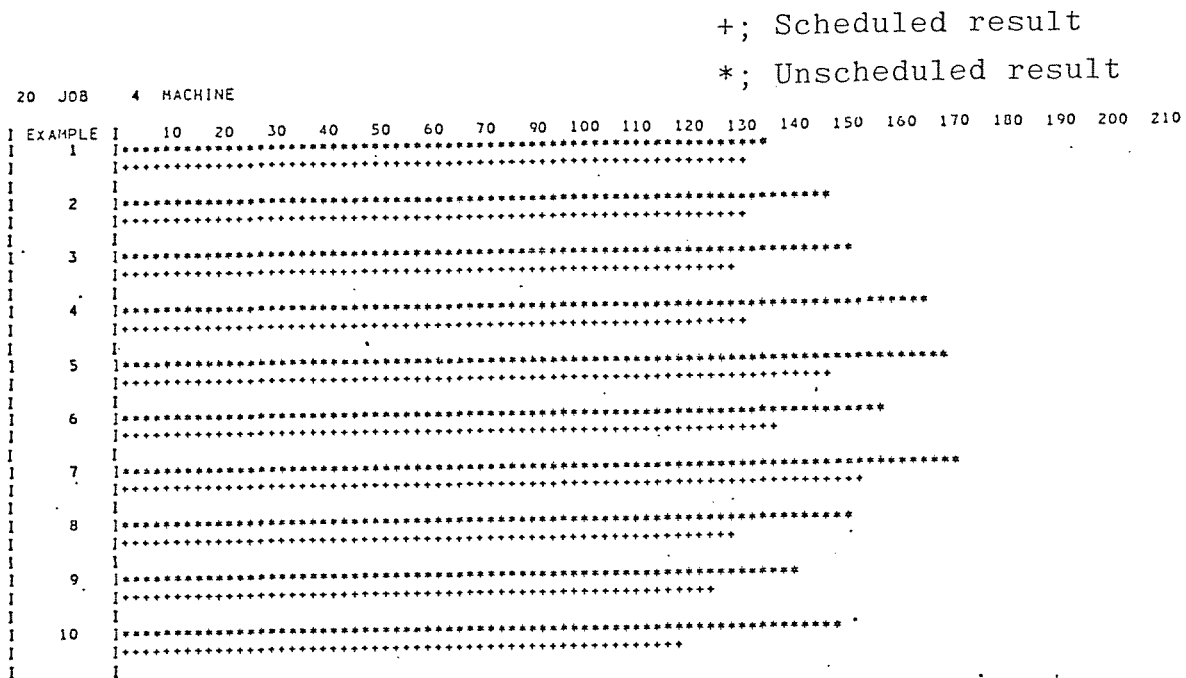


Fig. 2.7 20-chain 4-processor experiments

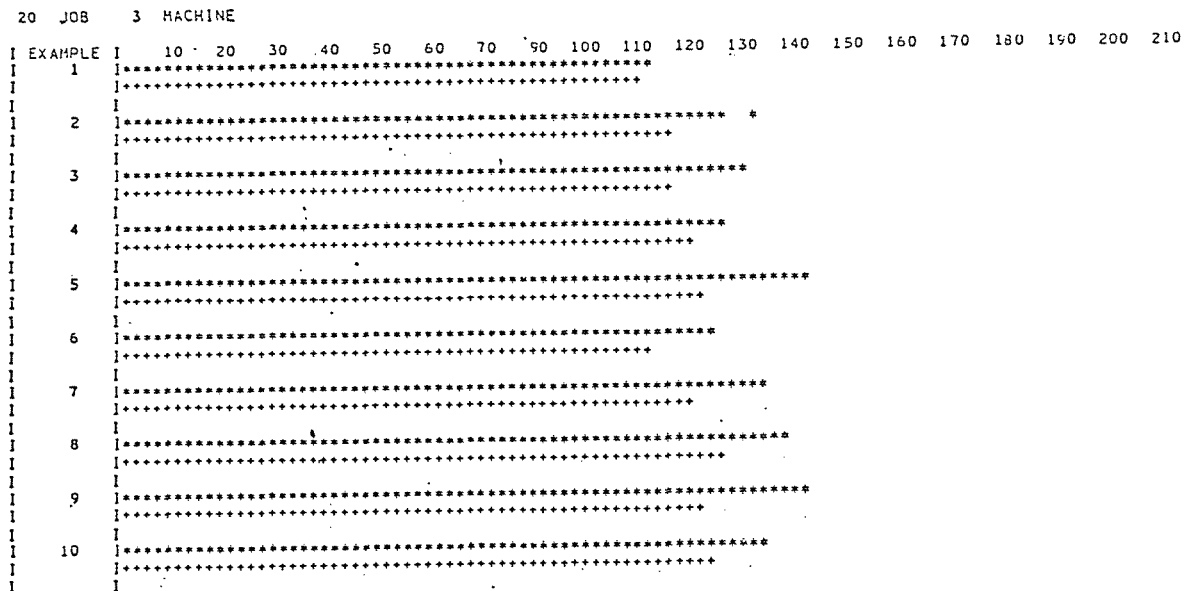


Fig. 2.8 20-chain 3-processor experiments

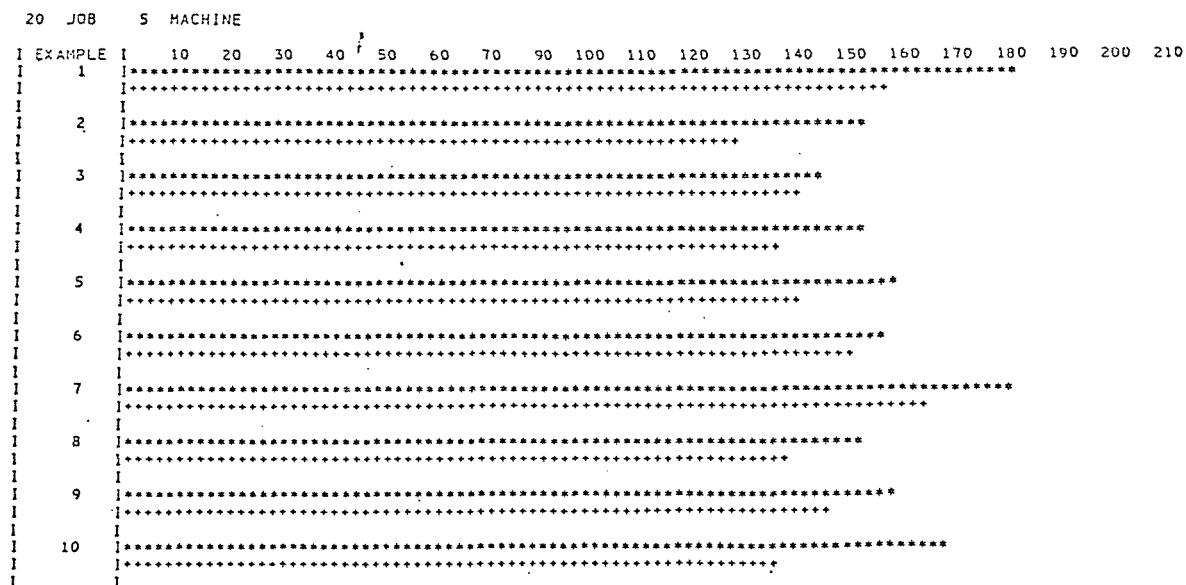


Fig. 2.9 20-chain 5-processor experiments

30 JOB 3 MACHINE

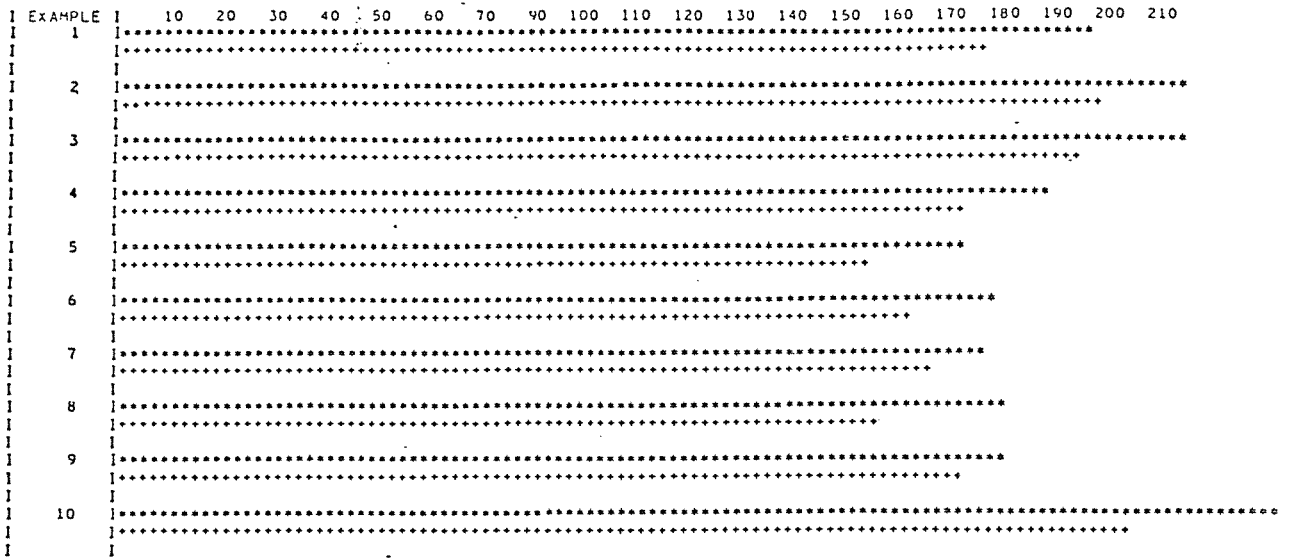


Fig. 2.10 30-chain 3-processor experiments

30 JOB 4 MACHINE

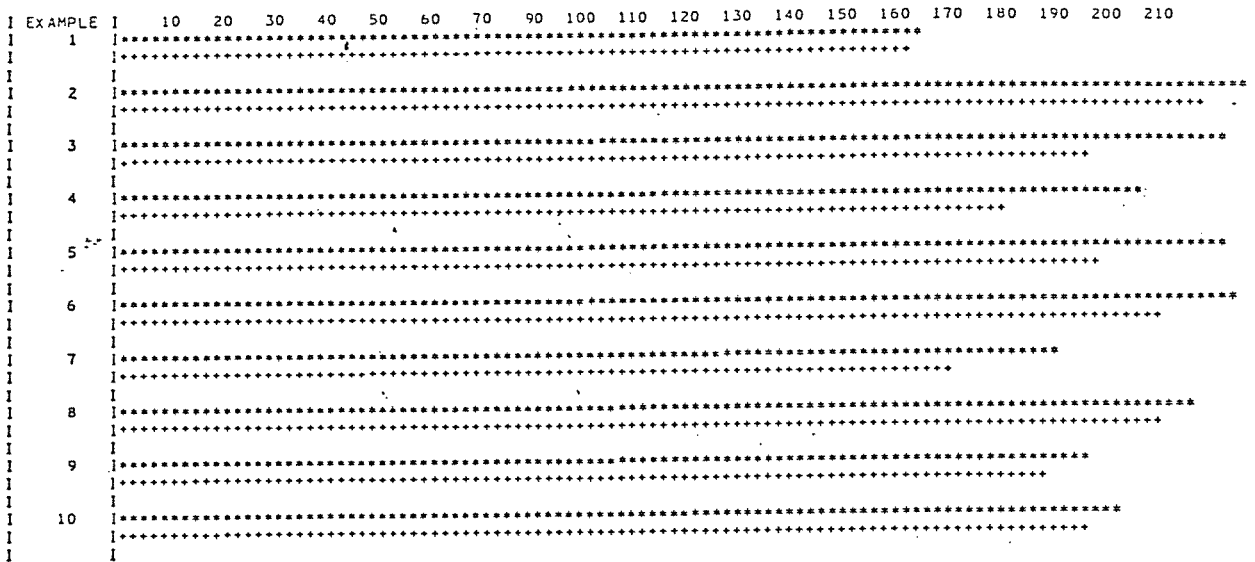


Fig. 2.11 30-chain 4-processor experiments

30 JOB 5 MACHINE

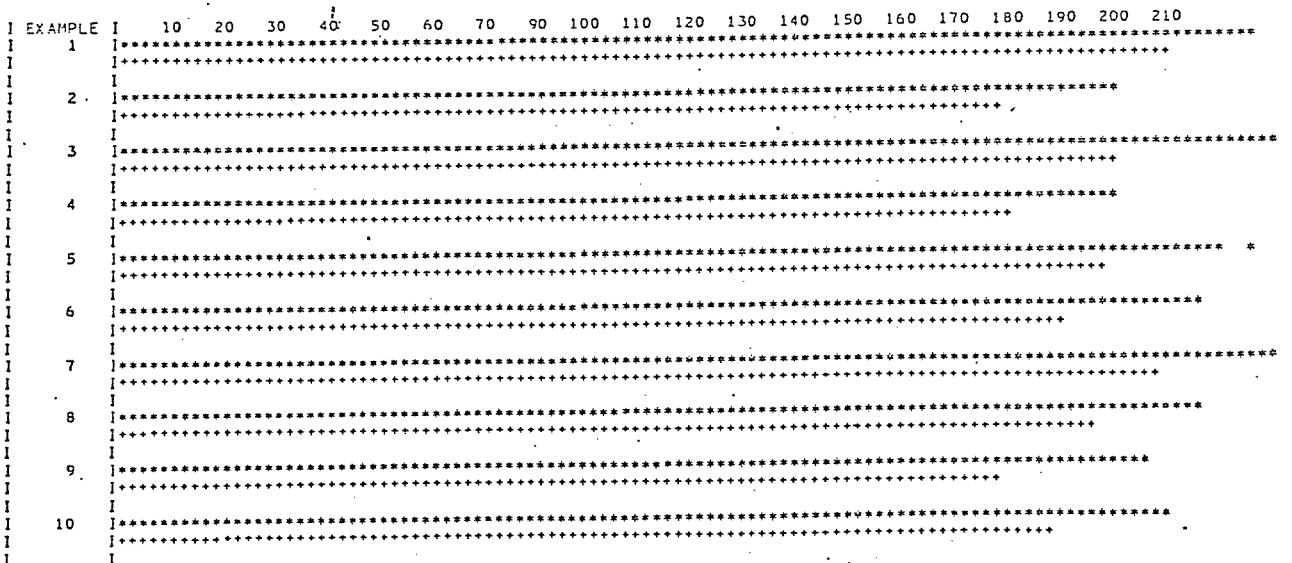


Fig. 2.12 30-chain 5-processor experiments

Through the numerical experiments, the followings are to be concluded.

In the experiment 1, the proposed method provides a good enough approximate solution. In the experiments 2 and 3, the solution is compared with all the possible schedule made up by the enumeration method. The results shows A.S.R. is around 90% and this is practical enough for the approximate solution. In the experiment 4, the exact solution is impossible to be reached by a mini-computer. Therefore, the schedule not to be scheduled is compared with the schedule to be scheduled by the proposed method. By these comparisons, the scheduled solution makes an effect for flow shop scheduling.

2.7 Conclusion

Through the discussion and the numerical experiments, we can reach the following conclusion.

- (1) A new method of a flow shop scheduling problem as a one and a half dimensional space allocation problem is presented.
- (2) The proposed method is based on the weak order of the precedence relation in the flow shop scheduling. When 2- and 3-processor are used, the results obtained by the proposed method becomes the same as the one obtained by Johnson's rule. The proposed method shows that Johnson's rule is extended to the case of m-processors.
- (3) On the n-chain and m-processor problems, the numerical experiments are executed. The experiments prove that the proposed method gives a practical enough solution.
- (4) The memory size required for the proposed method is around $n(n+m)$ words and this size is small enough for a mini-computer.

References

1. Fishburn, P., "Utility Theory for Decision Making", John Wiley & Sons, N.Y., 1970.
2. Kemeny, J., and J.L. Snell, "Applications of Graph Theory to Group Structure" (Japanese version), Prentice-Hall, INC., N.J., 1963.
3. Harary, H., "Graph Theory" (Japanese version), Kyoritsu Shuppan, p.280, 1971.
4. Johnson, S.M., "Optimal Two- and Three-Stage Production Schedules with Setup Times Included", Nav. Res. Log. Quart., 1,1, p.61, 1954.
5. Ignall, E. and Schrage, "Application of the Branch-and-Bound Technique to Some Flow Shop Scheduling Problems", Opns. Res., 13, 3, p.4000, 1965.
6. Kubo, H., "A Highly-Efficient Branch-and-Bound Method for Combinatorial Optimization Problems", Ph.D Thesis of Hokkaido University, 1976.
7. Nabeshima, I., "Theory of Scheduling", Morikita Shuppan, 1974.

3. P.B.M Approach to the Space Allocation Problem

---The Optimum Trimming of Many Rectangular Plates----

3.1 Introduction

This chapter deals with a problem that a number of rectangular plates---materials---are allocated to make a compact rectangular sheet---a resource---as small as it can be. As briefly seen in chapter 1, most of the developed methods apply only to specific cases. For instance, the two-dimensional problem is constrained in such a way to reduce it to a one-dimensional problem and therefore it can hardly be applied to the general two-dimensional problem.

In order to solve the general two-dimensional problem, a new method named P.B.M---Pair to Block Method---is proposed. This method is especially designed for solving a large number of materials with different sizes.

The basic strategy of P.B.M may be summed up as follows.

In the first place, all the rectangles are paired to produce a half of new rectangles so that the sum of wastes area included in the new rectangles will become minimum. Each of the new rectangles

are named "Block". If the number of blocks is one, the allocation is finished. Otherwise, the blocks are regarded as rectangles, and they are paired again. This process is repeated until a large block is formed from the whole of the given initial data.

Each time a new pair is formed, the well known "assignment problem" is introduced, that is, an assignment matrix must be bound to determine a pair.

This P.B.M is applied to a number of numerical experiments and the validity of the method is proved.

3.2 A Formula of the Problem

First, let us define some symbols and terminology, then describe the problem. The symbols are a little different from those in chapter 1 because the first letter of a key word representing the problem is used. The symbols are defined as follows.

R_i Given i -th rectangle having width a_i and length b_i . This corresponds to the material in the general model.

B_k The k -th rectangular blank in all the possible allocating ways of the given rectangles,

in some cases predetermined and restricted, with a width W_k and a length L_k .

$A(B_k)$ The area of blank B_k .

W_k The waste area summed up by all of waste areas included in blank B_k .

S : The total area of the given n rectangles, namely,

$$S = \sum_{i=1}^n a_i b_i \quad (3.1)$$

S_i The area of the given i -th rectangles, namely,

$$S_i = a_i b_i \quad (3.2)$$

$S_{i,j}$: The sum area of the given i -th and j -th rectangles.

$W_{i,j}$ The waste area included in a rectangle to be produced by joining the i -th and j -th rectangles.

The terminology is as follows.

Waste Scrap or trimloss

Pair Combination of two rectangles

Block Combination of two pairs

And the problem is described as follows.

The problem Minimize the area of blank in order to include all given rectangles without their

overlapping.

Then, the problem is formulated as follows.

The formula:

$$\min_{B_k} W_k = \min_k A(B_k) - S, \quad (3.3)$$

$$\text{subject to } \bigcap_{i=1}^n R_i(x_i) = \emptyset \quad (3.4)$$

$$\text{and } \bigcup_{i=1}^n R_i(x_i) \subset B_k(x), \quad (3.5)$$

Where $R_i(x_i)$ is the region of rectangle with its left under coordinate x_i and $B_k(x)$ is the region of the blank.

Namely, the problem is to determine an allocation of all R_i under the Eqs. 3.3, 3.4 and 3.5.

Fundamental to P.B.M is the determination of the algorithm to be used. Several factors influence this decision. First, the area of blank B_k is unconstrained in some cases and constrained in other cases. Second, the physical condition that occurs in trimming process is taken into consideration. For this purpose, the giullotine cutting technique is used.

3.3 Fundamental Approach to Problem

With respect to the two-dimensional problem mentioned above, it seems possible to obtain a certain guaranteed solution by adopting an exhaust enumeration search method. However, if we use this simple technique for the problems on a large scale, the solution time will be unrealistically long. In making a block B_{ij} with any two given rectangles R_i and R_j , and joining another one to B_{ij} , and so on, the combinatorial number becomes $(n - 1) ! \cdot 4^{n-1}$. Then, it is very difficult to seek a solution for large n .

There are two other approaches to such problems. Before describing two approaches, let us define the given problem $P_0(n)$, where n is the number of rectangles. The one is to divide the problem $P_0(n) = P_1(n_1) \cup P_2(n_2) \cup \dots \cup P_k(n_k)$, (3.6)

where $n_i < n$ and $\sum_{i=1}^k n_i = n$. The solution will be

searched consequently through the subproblem or be combined by the solution of the subproblem. This type of algorithm is the Branch and Bound Method or the Dynamic Programming. But in this two-dimensional problem, the number of combination of the subproblem solution grows larger, too.

The one is to transform the problem to which the solution is known. By transforming the given problem to the known problem such as

$$P \leftarrow T (P) \quad (3.7)$$

Such transformation is seen in S.U.M T technique and simplex technique. If the transformation is iteratively executed and the dimensions of the variables are gradually decreased, it may be described as

$$P_{i+1} (n_{i+1}) = T_i (P (n_i))$$

where T_i implies transformation and n_{i+1} , n_i are the number of variables to be solved. In this transformation, if w_i becomes small enough to solve the problem transformed, the solution is searched. As the problem is consquently transformed as

$$P_0 (n_0) \rightarrow P_1 (n_1) \rightarrow P_2 (n_2) \rightarrow \dots \rightarrow P_k (n_k), \quad (3.8)$$

$$\text{where } n_0 > n_1 > \dots > n_k,$$

the upper limit of the computing time may be K times longer than the computing time required to solve the problem $P_0 (n_0)$.

The basic idea of P.B.M. is based on this concept.

In order to transform the primitive problem into the problem reducing the number of variables solved, a formulation of the assignment problem is employed in P.B.M. By solving the assignment problem, the number

of the variables of the reduced problem will become half. The formulation of the assignment problem is iteratively repeated till the problem satisfy the terminating conditions. The detail of the algorithm of P.B.M. is illustrated in the next section.

3.4 The Algorithm of P.B.M.

The procedure of reducing the size of the problem is accomplished by formulating the problem as an assignment problem. Solving the assignment problem means the given number of rectangles, which will produce a blank, and generate a half of n rectangles. When pairing, each two rectangles become a new rectangle with a waste shown in Fig. 3.1. The pairing is done in such a way that the sum of the waste areas included in a half of n new rectangles are minimized. In so doing, the problem is formulated as the assignment problem. Each new rectangle is named a block. Then, the blocks are regarded as the rectangles. In this way, the size of the problem is reduced from n rectangles allocation problem to a half of n rectangles. If the number of the block is one, reducing procedure is.

finished.

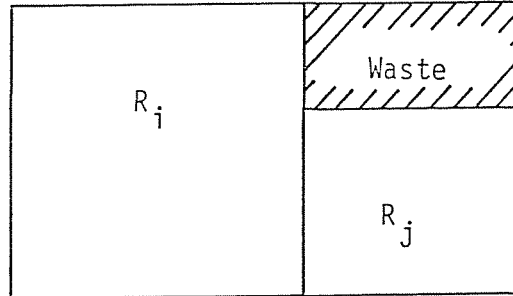


Fig. 3.1 A waste

3.4.1 The Basic Procedure

The basic procedure of the algorithm comprises the following four steps. (Fig. 3.2)

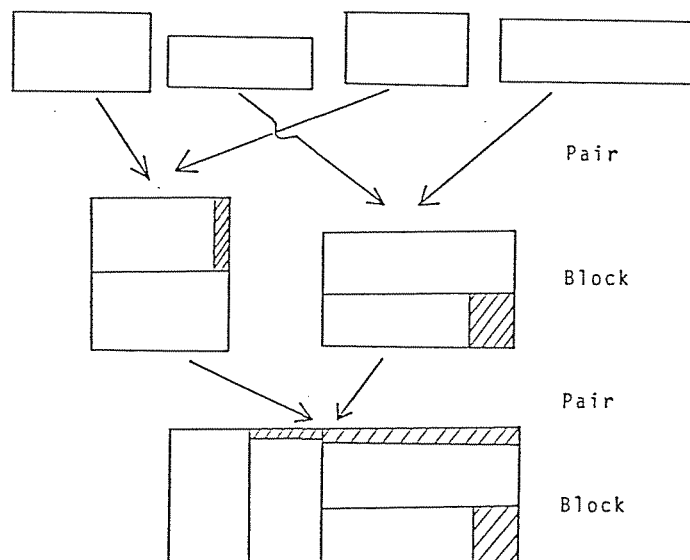


Fig. 3.2 Pair to block step

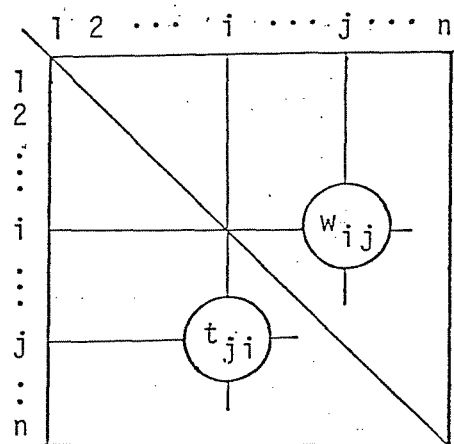
1. BY applying the theory of the assignment problem, determine a set of pairs ($n/2$ blocks) from the given n rectangles so as to minimize the waste.
2. Replace n with $n/2$ $n/2 \rightarrow n$.
3. Consider a block as a new rectangle.
4. End the process if only one block is obtained, otherwise go back to step 1.

3.4.2 Waste Matrix

In order to establish the assignment problem, a cost matrix is needed. Here as the cost matrix's element, the area of the waste generated when pairs are made into blocks is used, therefore this matrix is called a waste matrix.

The waste matrix is constituted in the following manner as shown in Fig. 3.3.

Fig. 3.3 Waste matrix



In considering the method of joining any two rectangles R_i and R_j , there are always four possible ways. In each of them, the area of block B_{ij} will be

$$\begin{aligned}
 A_1 &= [\max(a_i, a_j)] \cdot [b_i + b_j] \\
 A_2 &= [\max(a_i, b_j)] \cdot [b_i + a_j] \\
 A_3 &= [\max(b_i, b_j)] \cdot [a_i + a_j] \\
 A_4 &= [\max(b_i, a_j)] \cdot [a_i + b_j]
 \end{aligned} \tag{3.9}$$

and total area S_{ij} of both R_i and R_j is

$$S_{ij} = a_i b_i + a_j b_j \tag{3.10}$$

Where a_i and b_j are two sides of a rectangle R_i .

As an element of the waste matrix, let us set it to

$$W_{ij} = \min(A_1, A_2, A_3, A_4) - S_{ij} \tag{3.11}$$

for all $i, j = 1, 2, 3, \dots, n, i \neq j$.

Furthermore, in order to determine the pair R_i and R_j , the aspects shown in Eq. 3.11 must also be considered. That is, which sides of R_i and R_j adjoin in making out a block. The matrix t_{ij} denotes the aspects for the pair R_i and R_j in the form of code. In table 1, these aspects are given in the code t_{ij} and each of the codes 1 through 4 corresponds to the S_1 through S_4 in the Eq. 3.9.

Because of the symmetric properties $w_{ij} = w_{ji}$ and $t_{ij} = t_{ji}$, the waste matrix can be divided into two parts by a diagonal line, leaving the obtained

waste in the upper triangular matrix and substituting the obtained code in the lower one. This operation saves the memory size to a half.

Table 5.1 Joining code table

t_{ij}	R_i	R_j
1	l_i	l_j
2	l_i	l_j
3	w_i	w_j
4	w_i	w_j

l_i Length of rectangle i
 w_i Width of rectangle i
 l_j Length of rectangle j
 w_j Width of rectangle j

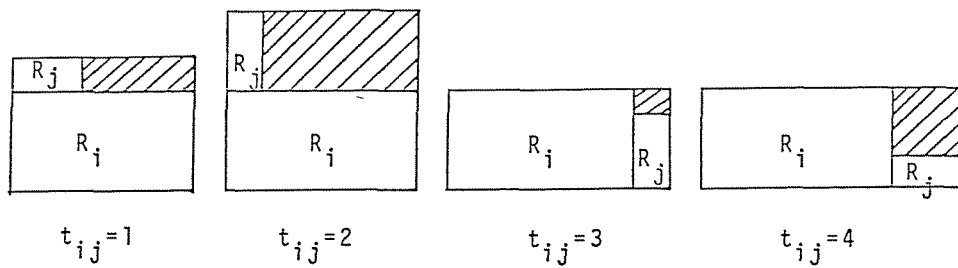


Fig. 3.4 Four way joining corresponding to the code

3.4.3 Formulation as the Assignment Problem

In order to reduce the problem size and solve the two-dimensional problem, the problem is transformed to an assignment problem. The assignment problem can be formed as follows.

Determine $X = \{x_{ij}\}$ which satisfies the problem 3.12 and the Eqs. 3.13-15.

$$\text{Minimize} \quad F = \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_{ij} \quad (3.12)$$

$$\text{subject to } x_{ij} = 0 \text{ or } 1 \quad i, j = 1, 2, \dots, n, \quad (3.13)$$

$$\sum_{i=1}^n x_{ij} = 1, \quad j = 1, 2, \dots, n, \quad (3.14)$$

$$\text{and} \quad \sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, \dots, n, \quad (3.15)$$

where X is called the assignment matrix and w_{ij} is an element of the waste matrix. The matrix X has the following properties.

$$(a) \quad x_{ij} = \begin{cases} 1 & R_i \text{ is assigned to } R_j \\ 0 & \text{Otherwise} \end{cases}$$

- (b) From the Eqs. 3.13-14, each column and row of the matrix X has the elements all of which except one are zero, the exception is one.

The number of matrices which satisfy the Eqs. 3.12-15 is $n!$. If the number of n becomes large,

the trial number to find out X which satisfies the problem 3.11 becomes very large. For instance, when $n = 10$ is given, $n !$ becomes almost 3.6×10^6 . Therefore, it becomes hard to determine the matrix X even if the problem is transformed to the assignment problem. Then, the basic idea of Flood's Hungarian Method is adopted to remedy this obstacle.

3.4.4 Solving Method for the Assignment Problem

The Hungarian Method is based on the following two theorems.

Theorem 1 If it is possible to divide a set of elements of a given matrix into two kinds by some property Q , the minimum number of lines which involve all the elements having property Q , where the line is a column or a row containing (an) element(s) with property Q is equal to the maximum number of lines which have property Q if each line has only one element with property Q .

Theorem 2 For a given cost matrix $W_0 = [w_{ij}^{(0)}]$, if it is possible to create any other matrix $D = [d_{ij}]$, both solutions for W and D are the same, where

$$d_{ij} = w_{ij}^{(0)} - u_i - v_j \quad (3.16)$$

and u_i, v_j are arbitrary constants, respectively.

Theorem 1 is known as the König's one and theorem 2 is applied to its dual problem. In solving the assignment problem, the following equations are derived from theorems 1 and 2,

$$w_{ij}^{(o)} \geq u_i + v_j \quad x_{ij} = 0 \quad (3.17)$$

$$w_{ij}^{(o)} = u_i + v_j \quad x_{ij} > 0, \quad (3.18)$$

where $w_{ij} = w_{ij}$.

And the following Eq. 3.19 is obtained by substituting the Eq. 3.16 into the Eq. 3.12,

$$\begin{aligned} F &\geq \sum_i^n \sum_j^n (u_i + v_j) x_{ij} \\ &= \sum_{i=1}^n (u_i \sum_j x_{ij}) + \sum_{j=1}^n (v_j \sum_i x_{ij}) \end{aligned} \quad (3.19)$$

Then the Eqs. 3.14 and 15 are substituted into the Eq. 3.19,

$$F \geq \sum_{i=1}^n u_i + \sum_{j=1}^1 v_j \quad (3.20)$$

is obtained. Here E denotes the right-hand side of the Eq. 3.20, the primary assignment problem is rewritten as the dual problem such as

$$\text{maximize} \quad E = \sum_{i=1}^n u_i + \sum_{j=1}^n v_i \quad (3.21)$$

$$\text{subject to} \quad u_i + v_j \leq w_{ij}^{(o)} \quad (3.22)$$

The Hungarian Method uses these results. In other words, the Hungarian Method is to determine the reduced matrix, which involves the minimum number of lines, for the aim of theorem 1. The reduced matrix must satisfy the Eq. 3.22. How to determine the reduced matrix is explained in the following eight steps.

- Step 1 Determine the minimum value v_j in each column of cost matrix, and subtract v_j from each element of column j .
- Step 2 Decide the line involving the property such as $w_{ij} = 0$.
- Step 3 Stop if the number of lines is n , otherwise go to step 4.
- Step 4 Determine the minimum value u_i in each row of the cost matrix and subtract u_i from each element of row i .
- Step 5 Decide the line again.
- Step 6 Stop if the number of lines is n , otherwise go to step 7.
- Step 7 Let us define a set of lines as SL and the current reduced matrix as $w_{ij}^{(k+2)}$.

Then set

$$w_{ij}^{(k+3)} = \begin{cases} w_{ij}^{(k+2)} + h_{(k+2)}, & i, j \in SL, \\ w_{ij}^{(k+2)}, & i \in SL \text{ or } j \in SL, \\ w_{ij}^{(k+2)} - h_{(k+2)}, & i, j \notin SL, \end{cases} \quad (3.23)$$

where $k = 1, 2, \dots, K$ (K is the trial number to make the reduced matrix until the optimum solution is obtained and note that $h_{(k+2)}$ shows the minimum number of the elements which do not belong to the line SL . And $w_{ij}^{(1)}$, $w_{ij}^{(2)}$, $h_{(1)}$ and $h_{(2)}$ are given by step 1 and step 4, namely,

$$w_{ij}^{(1)} = w_{ij}^{(0)} - h_{(1)}, \quad h_{(1)} = v_j, \quad (3.24)$$

$$w_{ij}^{(2)} = w_{ij}^{(0)} - h_{(2)}, \quad h_{(2)} = u_i. \quad (3.25)$$

Step 8 Determine the line and go back to step 6.

3.4.5 Determining Lines

As to computing time, the procedure used to determine the lines becomes important. A clue to the fast computing is to determine the exact minimum number of lines, which are drawn on the rows and columns including all the zero elements of the reduced assignment matrix. But this procedure is not known. Therefore the efficient procedure is presented to determine the lines. The procedure is described as follows:

Let us consider that there are n_t zero elements in the t -th reduced assignment matrix. n_t is no less than n . On the other hands, the maximum number of lines containing n_t zero element drawn on the matrix is no more than n . In order to find out the minimum number of lines, two conditions are desired:

1. The line contains as many zero elements as possible.
2. The zero elements of the line drawn on some column (row) is not contained in the lines drawn on the row (column) lines.

In other words, the line is the column (row), when it includes as many zero elements as possible

which are not to be included in the lines drawn on the row (column). This means that the column (row) is the line when it saves the maximum number of the lines drawn on the rows (columns). The number of the lines saved in the rows (columns) by adopting a certain column (row) as the line becomes

$$\begin{aligned} & (\text{the number of zero elements of the column}) - \\ & - (\text{the number of rows which have zero elements} \\ & \quad \text{other than the zero element crossing in the} \\ & \quad \text{column}). \end{aligned} \quad (3.26)$$

From the point of view mentioned above, a new procedure is established.

For the column (row) of the matrix, let us set symbols p_i (q_j) and p_{d_i} (q_{d_j}).

p_i (q_j) Number of zero elements in the i -th column (j -th row).

p_{d_i} (q_{d_j}) Number of lines which are needed if the column i (row j) is adopted as the line, where the line is the column (row) containing zero elements.

And let us define α and β as the efficiencies for the columns i and j ,

$$\alpha = \max_i (p_i - p_{d_i}) / \sum_i p_{d_i} \quad (3.27)$$

$$\beta = \max_j (q_j - q_{d_j}) / \sum_j q_{d_j} \quad (3.28)$$

The true line is determined in turn of the larger number of α and β until all the zero elements are included in the lines.

If the number of lines is not equal to n , a reducing procedure is executed and one more line is increased at least.

3.5 Given Blank Having Restricted Width

In the former section, the discussion on how the given rectangular plates are allocated on the half infinite space, and the algorithm of P B.M. is developed.

When the given blank has any restricted width, the same algorithm can be applied to this case, too.

The problem in this case is described as follows.

Determine the allocation that satisfies the Eqs.

3.29-3.32.

$$\text{Minimize} \quad W_k = \min_k A(B_k) - S \quad (3.29)$$

$$\text{subject to} \quad \bigcap_{i=1}^n R_i(x_i) = \phi \quad (3.30)$$

$$\bigcup_{i=1}^n R_i(x_i) \subset B_k(x), \quad (3.31)$$

$$a(B_k(x)) \leq a_0, \quad (3.32)$$

where $a(B_k(x))$ is the width of the blank B_k and a_0 is the given restricted width.

To satisfy the Eq. 3.32, the width of the block paired by two rectangles is always no more than a_0 . This is easily realized by adding the following conditions to the Eq. 3.9; namely:

if $b_i + b_j > a_0$, then $b_i + b_j = \infty$,
 if $b_i + a_j > a_0$, then $b_i + a_j = \infty$,
 if $a_i + a_j > a_0$, then $a_i + a_j = \infty$,
 if $a_i + b_j > a_0$, then $a_i + b_j = \infty$.

3.6 Approach with the Iterative Enumerative Method

The P.B.M. seems to be effective in dealing with a large number of rectangles, but sometimes it seems to be ineffective in dealing with a small number. In view of this situation, a kind of enumeration method is prepared, and it is named "Iterative Enumeration Method (abbreviated as I.E.M.).

To concrete this procedure, the following notation is used.

R_i	Given i -th rectangular plate.
B_t	The t -th block which is made of the ($t - 1$)-th block and a rectangular plate selected in the t -th step.
W_t	The area of the waste included in block B_t .
$S(B_t)$	The areas of block B_t .
$S(R_i)$	The area of rectangular plate R_i .
E_t	A set of suffix with which rectangular plates have not yet been adopted for allocation till t -th step.
$C(B_t, R_i)$	The area of the block produced by pairing block B_t and rectangular plate R_i .

The allocation procedure is presented to find out the solution that

$$\text{determine} \quad \min_{k=1} W_{kn}, \quad (3.33)$$

$$\begin{aligned} \text{subject to} \quad W_t &= W_{t-1} + \\ &\min_{i \in E_t} [C(B_{t-1}, R_i) - \{S(B_{t-1}) \\ &\quad + S(R_i)\}], \end{aligned} \quad (3.34)$$

where $W_{kn} = W_n$ when $B_o = R_k$.

Eqs. 3.33 and 3.34 imply a sequential jointing method of Dynamic Programming type.

This I.E.M. is added to P.B.M. when the number of blocks is less than eight. It cannot be said the solution gained by adding I.E.M. to P.B.M. is better than the one only worked by P.B.M., because this choice is relative to the given data.

3.7 The Numerical Experiments

In order to prove the validity of the presented method, the numerical experiments were made on the computer, FACOM 230/60 system of the computing center of Hokkaido University, and the programs were coded with FORTRAN.

One of the criteria to judge the allocation is the waste ratio. Let us be said waste ratio,

$$\gamma_k = 100 \quad (A(B_k) - S) / A(B_k) \quad (\%), \quad (3.35)$$

where the suffix k implies k -th element of all the possible allocations.

Even if this criteria can be used here and in other case with the same data, it is invalid if the data is different. The data used in the experiments except for the first experiment is taken from the center routine's RANDOM NUMBER. As shown in the above figures which are the results of computation,

N is the number of rectangular plates to be input, and MP is the number of remaining blocks in the following of I.E.M. to P.B.M. However, if MP = 1 is found in some figures, it means that only P.B.M. is used. JP implies the kind of random number used. For instance, JP = 1 is the normal distribution and JP = 2 is the uniform random numbers.

Experiment 1 The given data is taken from Fig. 3.5 and the result is shown in Fig. 3.6. Note that, if guillotine cutting is used, it is impossible to allocate in the same manner as Fig. 3.5. So, when the data taken from the allocation shown in in Fig. 3.5 is given, the P.B.M. finds a new result shown in Fig. 3.6(a). The numerical numbers for the data are listed in Table 3.2. I.E.M. works and improves this solution when the number of the blocks becomes five. The result is shown in Fig. 3.6(b).

Experiment 2 Twenty kinds of rectangles are produced by the uniform random numbers. These are listed in Table 3.3. The result by the use of P.B.M. is shown in Fig. 3.7. The waste ratio of this is 17.7% and the computing time is 8.0 seconds. I.E.M. follows up this solution when the number of blocks is five. The result is shown in Fig. 3.8. The solution is improved more than by one of P.B.M.

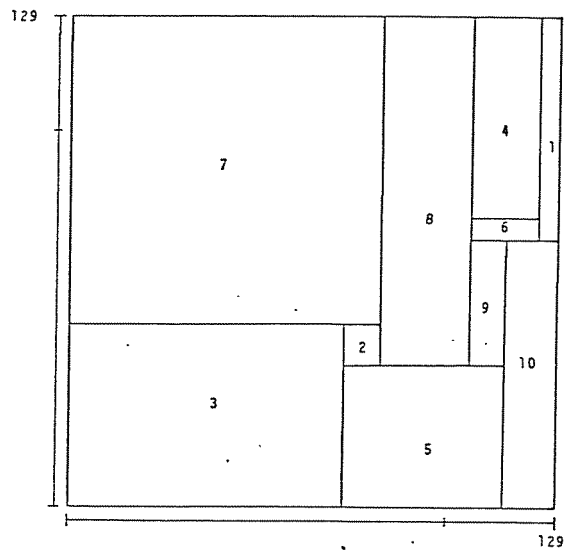


Table 3.2 Input data made
by Fig. 3.5

Material No.	Length	Width
1	58.	5.
2	10.	10.
3	48.	73.
4	18.	53.
5	42.	38.
6	5.	18.
7	81.	83.
8	91.	23.
9	9.	33.
10	71.	14.

Fig. 3.5 An example of
hand-made data

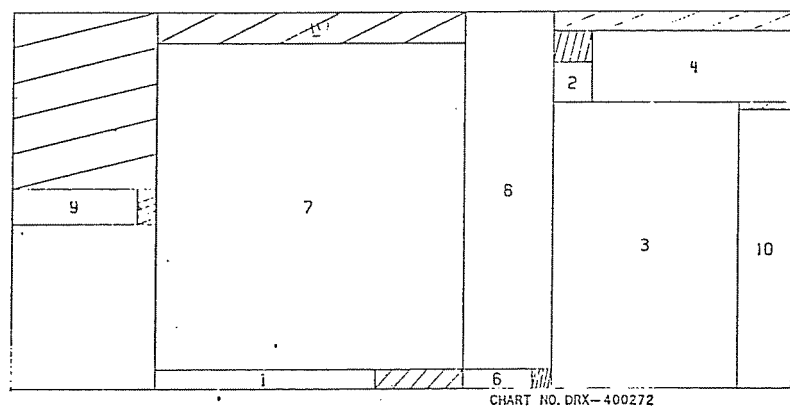


Fig. 3.6 (a) P.B.M. result

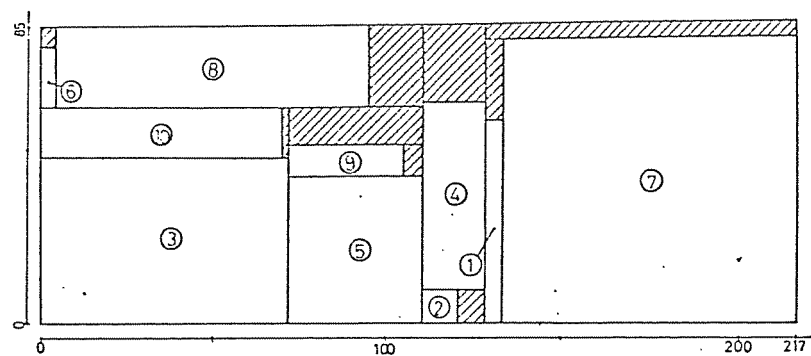


Fig. 3.6 (b) Followed by I.E.M.

Experiment 3 Twenty kinds of rectangles are produced by the normal random numbers listed in Table 3.4. P.B.M. gives the result as shown in Fig. 3.8., with the waste ratio 10.9% and the computing time 5.3 seconds. When I.E.M. works, the solution is not improved. This result with the waste 13.3% and computing time 5.1 seconds, is shown in Fig. 3.9.

Experiment 4 Thirty kinds of rectangles are produced by the uniform random number listed in Table 3.5. The result is the waste ratio 39.7% and the computing time 16.7 seconds. This allocation is shown in Fig. 3.10. I.E.M. improves this solution when the number of block becomes five. The improved solution shows the waste ratio 14.5% and the computing time 110.4 seconds.(Fig. 3.11)

Experiment 5 In order to estimate the computing time of the number of given rectangles, forty and sixty kinds of rectangles are produced by the uniform random numbers. The result is shown in Table 3.6 and Fig. 3.12.

Experiment 6 When the width of the blank (the last block) is not free, the experiment is executed. Fig. 3.13 shows the allocation of sixty rectangles, where the last width is restricted. The result is the waste ratio 32.1% and the computing time 573 seconds.

Table 3.3 Data input
produced by the uni-
form random numbers

Material No.	Length	Width
1	88.	46.
2	2.	14.
3	69.	41.
4	60.	67.
5	85.	73.
6	96.	82.
7	87.	58.
8	94.	8.
9	92.	77.
10	99.	56.
11	75.	73.
12	20.	1.
13	77.	51.
14	28.	79.
15	99.	79.
16	57.	89.
17	8.	87.
18	1.	93.
19	29.	68.
20	21.	15.

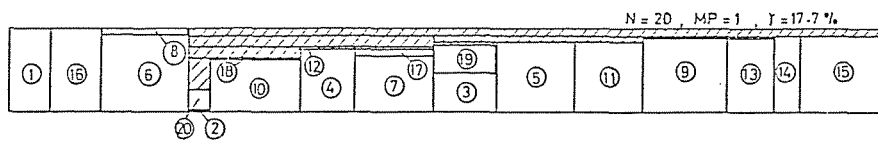


Fig. 3.7 P.B.M. result

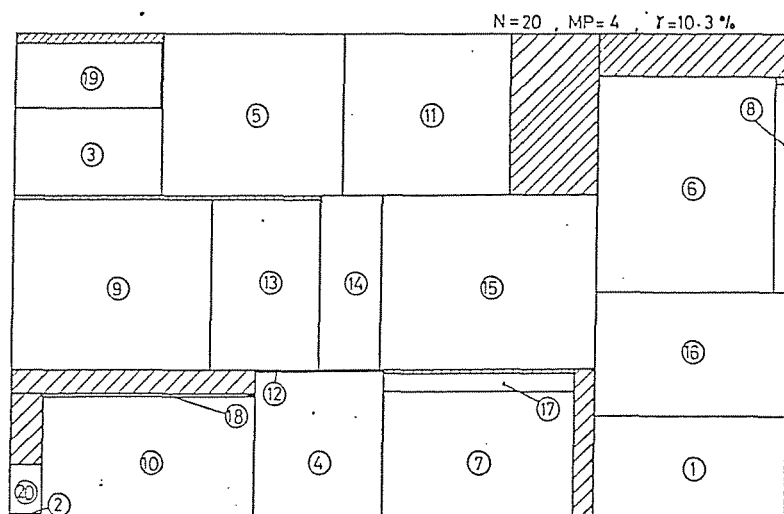


Fig. 3.8 Result followed up by I.E.M.

Table 3.4 Input data produced
by the normal random numbers

Material No.	Length	Width
1	20.	20.
2	24.	37.
3	46.	34.
4	60.	23.
5	26.	27.
6	40.	43.
7	73.	58.
8	24.	35.
9	67.	64.
10	33.	46.
11	42.	68.
12	16.	38.
13	48.	47.
14	69.	46.
15	59.	36.
16	56.	46.
17	30.	53.
18	40.	31.
19	30.	11.
20	48.	44.

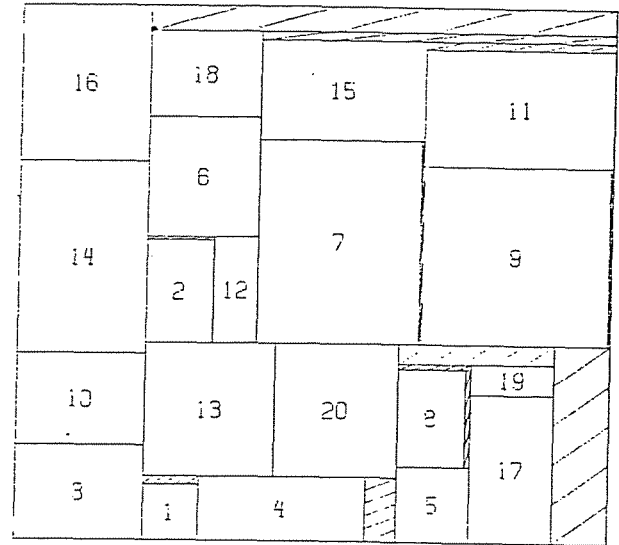


Fig. 3.8 P.B.M. result

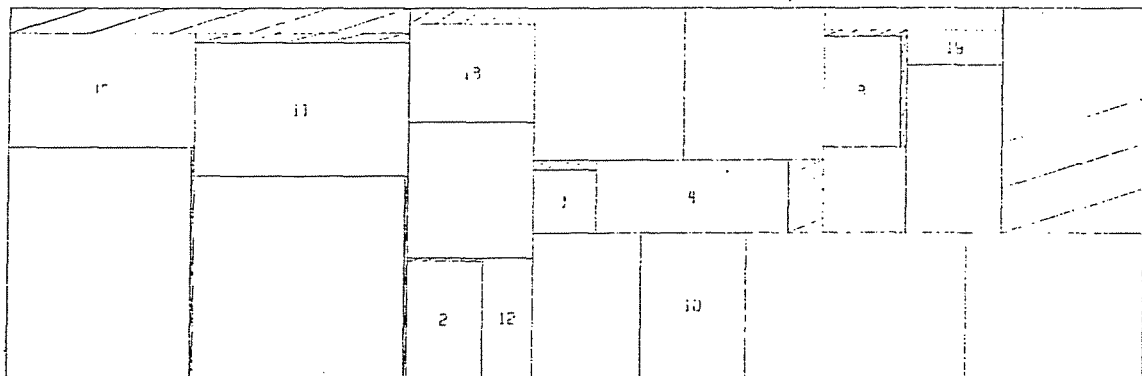


Fig. 3.9 Result followed by I.E.M.

Table 3.5 Input data produced
by the uniform random numbers

Material No.	Length	Width
1	40.	17.
2	19.	45.
3	86.	82.
4	91.	49.
5	30.	13.
6	38.	78.
7	92.	93.
8	10.	47.
9	51.	70.
10	17.	34.
11	50.	51.
12	35.	77.
13	95.	7.
14	98.	80.
15	53.	72.
16	8.	6.
17	54.	42.
18	24.	84.
19	92.	75.
20	72.	2.
21	22.	92.
22	38.	13.
23	61.	77.
24	71.	33.
25	90.	75.
26	82.	37.
27	52.	95.
28	46.	66.
29	52.	8.
30	99.	21.

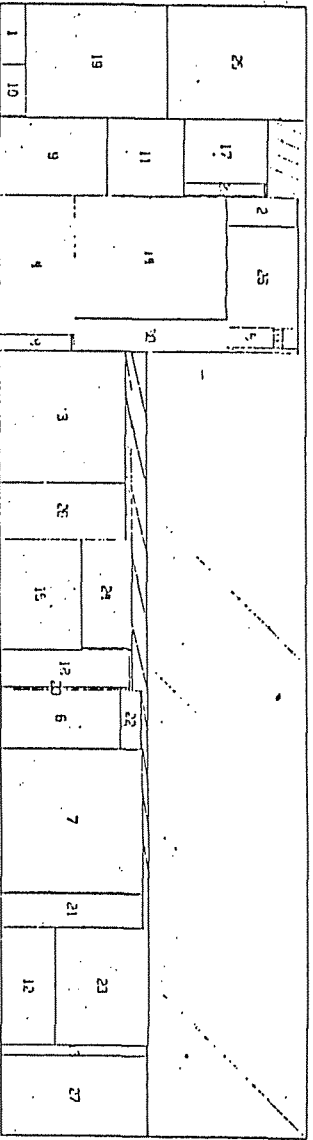


Fig. 3.10 P.B.M. result

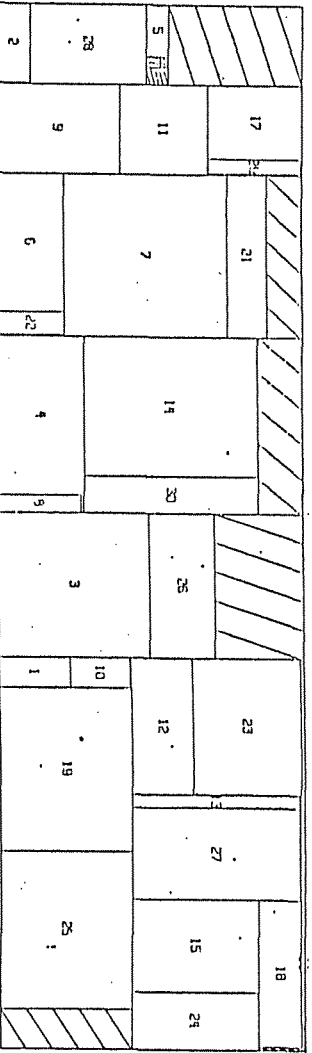
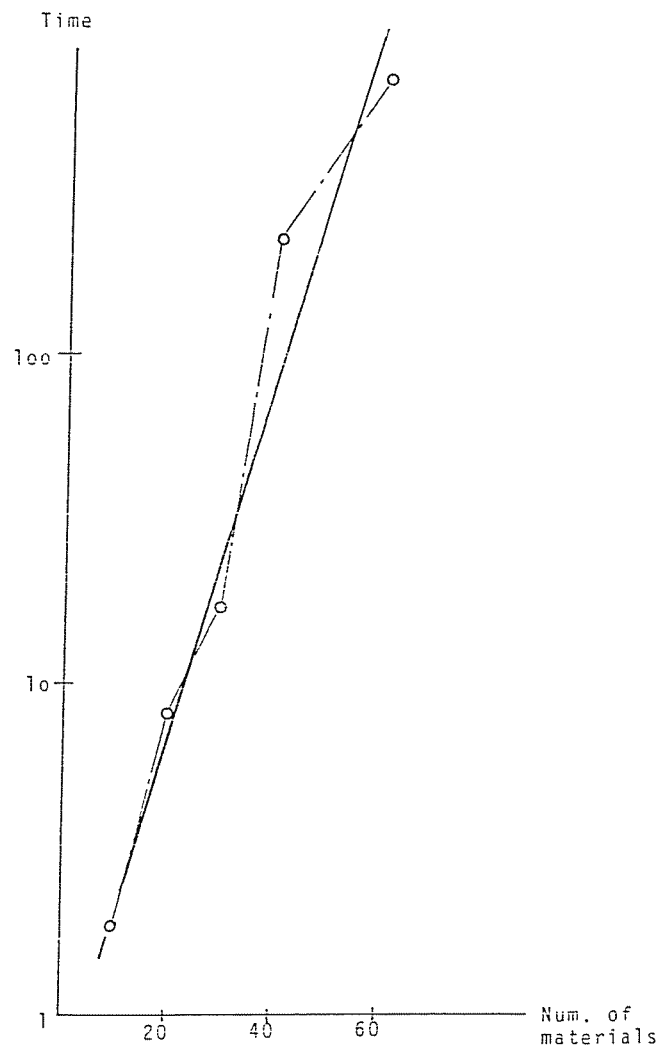


Fig. 3.11 Result followed by I.F.M.



(Produced by the
uniform random
numbers)

Fig. 3.12 Computation time of P.B.M.

Table 3.6 Computation time

n	P.B.M.		P.B.M. followed by I.E.M		Random number
	Waste ratio(%)	Time(sec)	Waste ratio(%)	Time(sec)	
10	16.3	1.8	9.7	1.6	Mannual
20	17.7	8.0	10.3	8.3	Uniform
20	10.0	5.3	13.3	5.1	Normal
30	39.7	16.7	14.5	110.4	Uniform
30			5.3	16.4	Uniform
40	15.3	218.7	10.2	202.4	Uniform
60	21.2	594.5			Uniform

Experiment 7 In order to estimate the goodness of the solution, numerical experiments are executed. The data of rectangles are handmade of a rectangular sheet in length 150 and width 100. The tests are tried when the number of rectangles is ten, twenty, and thirty and the number of tested times are eight, nine and nine, respectively. The data are reproduced in each experiment. The result is shown in Table. 3.7. The waste ratio reached by these experiments are around 20 % and this value is good enough for the practical solution.

Table 3.7 The result of experiment 7

Number of rectangles	Number of experiments	Average of waste area	Average of waste ratio (%)
10	8	4010.8	20.3
20	9	4584.5	21.8
30	9	3950.4	18.9

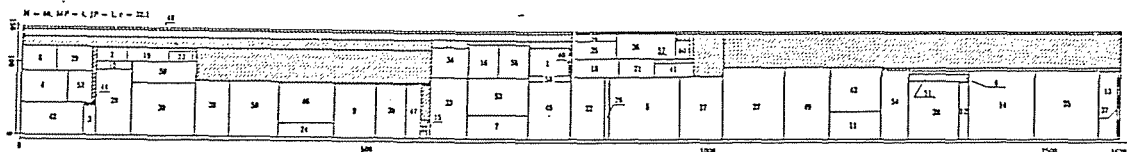


Fig. 3.13 Result in experiment 6; the width is restricted.

3.8 Conclusion

Through the consideration and discussion of the two-dimensional space allocation problem, the effective algorithm is developed, and by using this algorithm, we can reach the following conclusion.

1. The effective solution of the two-dimensional space allocation problem is obtained by defining it as an assignment problem.
2. This method is valuable when many different sizes of materials (rectangles) are given.
3. Even if a restricted width is placed on the blank, the proposed method is available.
4. A new criterion is given for the determination of the line on the Hungarian method.

However, P.B.M. still seems to have room for improvement from the theoretical point of view.

From the practical point of view, P.B.M. is modified for the use of metal sheet cutting planning by Mitsubishi Electric Company, and it is on execution. This is described in Appendix A.

References

1. Ohlin, C., "TRIM67-A Heuristic Trim Program for the Paper Industry," IBM Nordic Laboratory, Sweden, 1963.
2. Eisemann, K., "The Trim Problem," Management Science, Vol.12, p.B-279, 1965/66.
3. Gilmore, P.C., and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem," Opns. Res., Vol.9, p.849, 1961.
4. Gilmore, P.C., and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem - Part II," Opns. Res., Vol.11, p.863, 1963.
5. Gilmore, P.C., and R.E. Gomory, "Multi-Stage Cutting Stock Problems of Two and More Dimensions," Opns. Res., Vol.13, p.94, 1965.
6. Gummersall, F.R., "IBM 1130 Combinatorial Corrugator Scheduling Problem," IBM Program Information Department, 1967.
7. Churchman, C.W., Ackoff, R.L., and E.L. Arnoff, Japanese version of "Introduction to Operations Research," Kinokuniya Shoten, P.417, 1961.

4. A Practical Approach to the Cutting Stock Problem

4.1 Introduction

In this chapter, a practical solution is presented in dealing with the problem as one of the cutting stock problem situation. The object of the problem is to determine the number of row blanks stocked in order to produce materials required by the user with minimum economy. The resources are regarded as the blanks, but the materials are used in their own terms in this problem. As various sizes of the materials and the blanks with various costs are given to the problem, not only the number of the blanks is determined but also the placements of the materials on the specified blank must be solved. In treating the problem, therefore, the problem is parted into two subprograms such as a nesting problem to allocate the materials onto the specified blank and an assignment problem to determine the number of the blanks relating to various nesting ways. For the first subproblem, a heuristic method based on a recursive approximation is presented and a tree structure model

of the data structure is prepared to store the information on the location of the materials being nested to the specified blank. For the second, the problem formulation is simply turned to the algorithm to determine the number of the row blanks.

The problem occurs in the small and midium sized metal sheet cutting companies, so that the method is desired for the use of a mini-computer. The presented method is available for the mini-computer and it gives the numerical experiment result of a waste ratio around 7%. This result is practical enough to compare with manual planning results and with the result in chapter 3.

4.2 Formula of the problem

The problem dealt with in this chapter is as follows:

Problem: Given b_j plates of the blanks B_j ($j = 1, 2, \dots, m$) with its price $C_{b,j}$, assign r_i plates of materials R_i ($i = 1, 2, \dots, n$) to the blanks with the minimum price.

To formulate the problem, constraints expressed below are considered. A shape of the materials and the blanks is restricted to a rectangle. Considering that the materials are assigned to the blank B_j , there are a number of material assignment (nesting) patterns. In the k -th pattern of the assignment to the blank B_j as the number of the material R_i , let us set $R_i(x_s)$ and $B_j(x_v)$ as the regions of $R_i(x_s)$ and $B_j(x_v)$ with left under corner coordinate x_s and x_v , respectively.

The condition under which the materials are located within the blank B_j is given by

$$\bigcup_{i=1}^n \left\{ \bigcup_{s=1}^{a_{ijk}} R_i(x_s) \right\} \subset B_j(x_v), \quad (4.1)$$

where a_{ijk} is the number of the material R_i in the k -th nesting pattern. And the condition under which the materials are located without their overlapping each other is given by

$$\bigcap_{s=1}^{a_{ijk}} R_i(x_s) = \phi \quad (i = 1, 2, \dots, n) \quad (4.2)$$

and

$$\bigcap_{i=1}^n \left\{ \bigcap_{s=1}^{a_{ijk}} R_i(x_s) \right\} = \phi \quad (4.3)$$

Then, let us set all assignment patterns as u_j . As the number of blank B_j of the k -th pattern, x_{jk} to be used for producing the materials is less than the given number of the blanks, the inequality constraint

$$\sum_{k=1}^{u_j} x_{jk} \leq b_j \quad (4.4)$$

should be satisfied, where b_j is the number of the blank B_j stocked. Also, the number of materials produced from the blanks must be requested to no less than r_i ($i = 1, 2, \dots, n$), the inequality constraint,

$$\sum_{j=1}^m \sum_{k=1}^{u_j} a_{ijk} x_{jk} \geq r_i \quad (i = 1, 2, \dots, n) \quad (4.5)$$

should be satisfied, where r_i is the number of the materials required by the user.

The goal of the problem is to determine x_{jk} and a_{ijk} which satisfy the constraints 4.1-4.5 with minimum costs, that is,

$$\text{minimize} \quad \sum_j^m \sum_k^{u_j} c_{jk} x_{jk} \quad (4.6)$$

subject to Eqs. 4.1-4.5,

where $c_{jk} = w_1 c_{bj} + w_2 c_{wj}$ In the problem

description 4.6, the cost c_{jk} is set as a sum of the blank price c_{bj} and the waste cost c_{wj} in the k -th nesting way of the blank B_j with suitable weights w_1 and w_2 , respectively.

4.3 Nesting and Assignment Algorithm

It is difficult to find out a solution which satisfies the constraints 4.1 through 4.5, for the problem belongs to the combinatorial one. Even if "An Integer Linear Programming" is employed, variables solved become so many that the solution is hardly found out, and it costs so expensive for the use of a computer. Therefore, an heuristic method is effective if it gives a good approximation. In solving the problem, it is considered that the problem consists of two subproblems: the one is to determine the location of the materials to the specified blank and the other is to determine the number of the row blanks selected for cutting out the materials. A heuristic method based on a recursive approximation is developed for the first problem and a simple method is adopted for the latter.

4.3.1 Recursive Procedure for Nesting

A nesting problem for the allocation of the materials to the specified blank is described as follows:

Subproblem 1: Allocate the materials R_i ($i = 1, 2, \dots, n$) to the specified blank B_j with a minimum waste, that is,

$$\begin{aligned} \text{minimize} \quad C_{wj} &= A(B_j) - \sum_{i=1}^n a_{ijk} \cdot A(R_i) \\ a_{ijk} \end{aligned} \quad (4.7)$$

subject to the inequations 4.1-4.5,
where $A(B_j)$ and $A(R_i)$ are the areas of the blank B_j and the material R_i , respectively.

The algorithm presented here is composed based on the recursive approximation.

Let us set W_t to t -th waste area as the remainder of the blank area by removing the material areas to be selected until $(t - 1)$ -th procedure. In order to select the material that results in the minimum waste in the t -th step, the following equation is satisfied:

$$\text{minimize} \quad W_t = \min_{R_i} (W_{t-1} - S_t) \quad (4.8)$$

subject to $W_{t-1} - S_t \geq 0,$

where S_t is the material area, selected to satisfy the above equation, and W_{t-1} is the waste produced in the preceeding step. By repeating this procedure until it becomes impossible to allocate the material within the remaining waste area W_t , the selection of the materials for the specified blank is accomplished. To begin the procedure, let us set W_0 equal to $A(B_j)$.

From the equation 4.8, the algorithm becomes

$$\begin{aligned} \text{minimize} \quad W_t &= \min_{R_i} (W_{t-1} - S_t) \\ &= W_{t-1} - \max_{R_i} S_t \end{aligned} \quad (4.9)$$

$$\begin{aligned} \text{subject to} \quad W_{t-1} - S_t &\geq 0, \quad W_0 = A(B_j) \quad (4.10) \\ \text{and } t &= 1, 2, \dots, e, \end{aligned}$$

where suffix e implies the last selection of the material. This shows a simple rule that selects the maximum area's material to be allocated in the current waste.

4.3.2 Nesting Procedure

In order to execute the rule obtained in the previous section, the calculating operation is to select the maximum area of the material to be

allocated in the waste. This is simply done by following the steps below:

- 1° Renumber the suffices of the materials in the order of their areas from large to small.
- 2° Set suffix t equal to 1.
- 3° Allocate the material R_t within the areas of the waste W_{t-1} if possible and continue allocating. If impossible, then go to step 4°.
- 4° Set $t+1$ to t . If t is equal to the last suffix number plus one, stop the procedure. If not, return to step 3°.

In the previous procedure, how to place the material on the blank under the inequations 4.1-4.3 is still unknown. To concrete this procedure, the following conditions are considered.

Condition 1) The algorithm should be iterative so that the new waste shape must be kept rectangle after the placement of the material.

Condition 2) Make out the waste area as large as it can be, leaving greater possibility of locating unused materials.

There are some cases in which the material is placed on the waste as shown in Fig. 4.1.

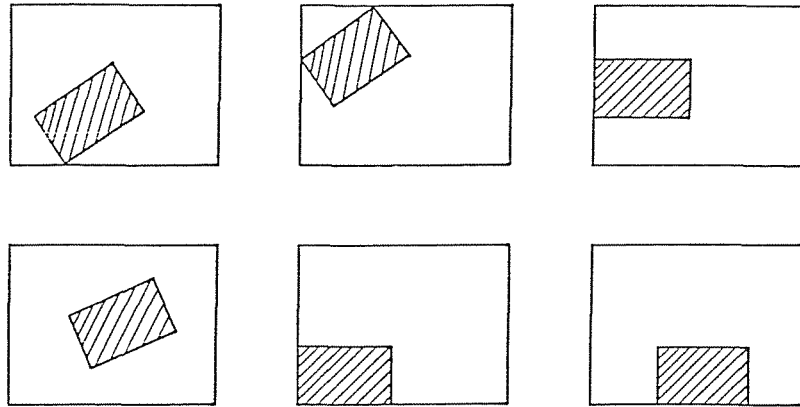


Fig. 4.1 Possible allocating ways

As shown in this figure, the shape of the waste does not become a rectangle. In order to apply the iterative procedure, the shape of the waste must keep a rectangular shape. This turns the division of the waste into some rectangular wastes. Furthermore, the condition 2 must be satisfied. Consequently, the material is allocated at the corner of the preceeding waste with

which the material edges meet as shown in Fig. 4.2.

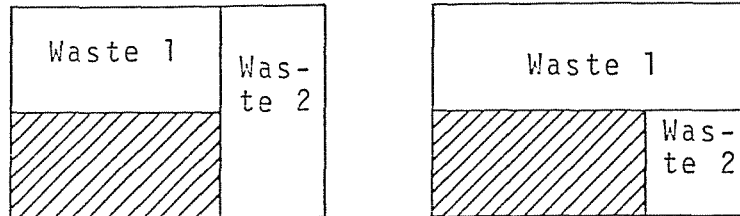


Fig. 4.2 Two ways of dividing the waste

There are two ways of dividing the waste because the new waste becomes L-shaped. The choice depends on the problem situation, that is, if there are a lot of long strip rectangular materials, a long rectangular waste had better be produced, and if not, the rectangular waste with a large area had better be produced. In either way, the old waste is divided into three parts: the material area selected and two new wastes. Therefore, this procedure is named "Divide-into-three" procedure.

To begin the procedure, we suppose that R_i ($i = 1, 2, \dots, n$) is ordered from large area

to small one, or from long strip to small strip by the criteria of area or length.

In this way, the nesting procedure is re-written as follows, where W_{tk} is the waste to be produced in the t -th step, and the k -th waste through all of the wastes, and W_{te} is the waste to be produced as the last one.

- 1° Set o to t , k and e , where $W_{00} = A(B_j)$.
- 2° Reset $t+1$ to t and $k+1$ to K .
- 3° If all of the wastes that have not been allocated by the materials become smaller than the rest of the material areas, stop the procedure. If not, go to step 4°.
- 4° Allocate the unused material R_i whose suffix is the smallest to waste W_{tk} such as

$$\text{maximize} \quad \sum_{h=1}^q A(R_i(x_h)) \quad (4.11)$$

$$\text{subject to} \quad \bigcup_{h=1}^q R_i(x_h) \subset W_{tk} \quad (4.12)$$

$$\text{and} \quad q \leq r_i \quad (4.13)$$

Where $R_i(x_h)$ is the region of the material with its left under corner coordinates x_h , $A(R_i(x_h))$ is the area of $R_i(x_h)$, and W_{tk} means the waste area and its region. Then

set q to d .

5° Divide the new waste into two rectangular wastes and set them to $W_{t+1,e+1}$ and $W_{t+1,e+2}$, respectively. In this step, the relation,

$$W_{t+1,e+1} + W_{t+1,e+2} = W_{tk} - d \cdot A(R_i) \quad (4.14)$$

is held.

6° Set $e + 2$ to e .

7° Set $k+1$ to k and go back to 3° if the materials with suffix t remain. Otherwise, go back to 3°

4.3.3 The Data Structure for Nesting

The information on how the materials are allocated must be given to the computer, therefore the data and the data structure on the information are prepared. Each datum keeps the waste information or the material information with its geometry and position.

In the previous procedure, the specified waste is always divided into three parts: the material area and two wastes, but the waste is smaller than any material area unused for nesting. Therefore, it becomes necessary for the datum to

be indicated which waste corresponds to this datum. and which wastes and materials are produced from this datum when it represents the waste. In this way, the datum d_k is the function as shown in the following equation,

$$d_k = (g(x_k), t, k, w_p, p_1, p_2), \quad (4.15)$$

where $g(x_k)$ shows the geometry information of this datum, x_k determines the relative position of the geometry, t is the step number, k is the number given for the datum generation, w is the parameter which shows whether this datum is the waste or the material, p_1 is the pointer directing the preceeding datum, and p_2 is the indicator of the waste not nested. As the geometry of the material and the waste are rectangles, the datum d_k is

$$d_k = (w, l, x, y, t, k, w_p, p_1, p_2), \quad (4.16)$$

$$w_p = \begin{cases} 1 & \text{material} \\ -1 & \text{material being transposed} \\ 0 & \text{waste} \end{cases}$$

where w is the width of the rectangle, l is the length of the rectangle, x and y are the coordinates of the left-under corner point.

The data structure becomes a tree directed by the pointers as shown in Fig. 4.3.

Now, the capacity of the data to store the information on the location becomes important because it restricts the availability of the procedure. Assuming that all the wastes having been produced become impossible to be allocated by the materials at the same time in the t -th step. The first waste (the specified blank) produces three data corresponding to two wastes and the material selected. Then, two wastes produce six data corresponding to four wastes and two materials selected. By repeating this iterative procedure, the t -th step produces $(2^t + t)$ data. Therefore the sum of the data S produced becomes

$$\begin{aligned}
 S &= \sum_{i=0}^t (2^i + i) \\
 &= (2^{t+1} - 1) + \frac{t(t-1)}{2} \quad (4.17)
 \end{aligned}$$

When t is equal to ten, the sum of the data S becomes around two thousands. This means, when ten words are used for the datum, forty kilo words in a computer are used for nesting data. Also Eq. 17 implies $(2^t - 1) + \frac{1}{2} t(t-1)$ materials are used for nesting. When t is equal to ten, it means almost ten thousands materials are used for nesting. The number is small enough for the object of this problem.

4.3.4 Determination of the Number of Row Blanks

The second subproblem is on how the number of row blanks producing the materials are determined. This is described as follows:

Subproblem 2 Determine the number of the row blanks, in which some materials has already been nested in order to produce the number of the material required by the user. Although the problem formulations 4.4 - 4.5 are applied by an integer linear programming directly, it is hardly possible for all of the blanks to enumerate all of the nesting patterns. A better way is to select a nesting pattern with a minimum waste for each blank and to solve the problem described by Eqs. 4.4 - 4.6. If the problem is solved in this way, the number of nesting pattern for each blank u_k , becomes one and Eqs. 4.4 - 4.5 are rewritten as follows: the restrictions are

$$x_j \leq b_j, \quad (4.18)$$

$$\sum_{j=1}^m a_{ij} \cdot x_j \geq r_i \quad (i = 1, 2, \dots, n), \quad (4.19)$$

and the objective function is

$$\text{minimize} \quad \sum_{j=1}^m c_j x_j \quad (4.20)$$

This formulation may allow the use of an integer linear programming method in less small size than before. But a unique nesting pattern for each blank may have a possibility of not using some material, namely, for some j^* , a_{ij}^* for any i becomes zero.

Also, it is possible to use R.C. Gomory and R.E. Gilmore's approach. But their approach includes the inverse matrix calculation in their simplex method. And as the nesting pattern for each blank is unique, the solution does not allow to vary activity nor to introduce to the same blank more than two new activities.

Considering above mentioned items, a heuristic method for the assignment procedure is developed. The feature of the procedure is that the problems 4.18 - 4.20 are generated iteratively by figuring out the activities a_{ij} and that the control variables are b_j and r_i .

The procedure is as follows.

- 1° Determine the allocation of materials to each B_j ($j = 1, 2, \dots, m$) minimizing wastes, and set cost c_j as the sum of the waste ratio and its blank price such as

$$c_j = w_1 \quad c_{wj} + w_2 \quad c_{cj}$$

where c_{wj} is the waste ratio and c_{cj} is the j -th blank price.

2° Find j^* as $j^* = \min_j c_j$,

3° Set x_{j^*} as

$$x_j = \min_j \left(\min_i^n [r_i / a_{ij^*}], b_{j^*} \right),$$

where a_{ij^*} is the number of material M_i included in the j -th blank nesting activity, r_i is the currently required number of material M_i and b_{j^*} is the number of current blank B_{j^*} stocked and $[]$ is gaussian notation.

4° Reset $r_i - a_{ij^*} x_{j^*}$ to r_i . If all of r_i become zero, stop the procedure.

5° Reset $b_{j^*} - x_{j^*}$ to b_{j^*} . If all of b_j become zero, stop the procedure. If not, go back to 1°.

4.4 Numerical Experiments

To prove that the method proposed for the cutting stock problem is efficient and practical, numerical experiments are executed by mini-computer PDP11/20 and OKITAC 4500-C. The program is coded with FORTRAN. As one of the criteria, a waste

ratio is used to estimate the goodness of the solution. A term "waste ratio" used below means

$$\frac{(\text{a total area of waste})}{(\text{an area of a blank})} \times 100 (\%) \quad (4.21)$$

The results of numerical experiments are as follows.

Experiment 1) When the number of blanks is one, the efficiency of the proposed method is computed with comparison to Gilmore's Dynamic Programming Method. Because of the memory size, the blank size tested is 50 in length and 50 in width, and the material sizes tested are one-digit uniform random numbers to be generated by the computer function, and the number of material kinds is 5.

The results obtained by OKITAC 4500-C are shown in Table 4.1. Fig. 4.4 is one of the graphic output of the solution. This table shows that the proposed method reaches a good approximate solution and takes less than half of the computing time shown in Gilmore's Dynamic Programming results.

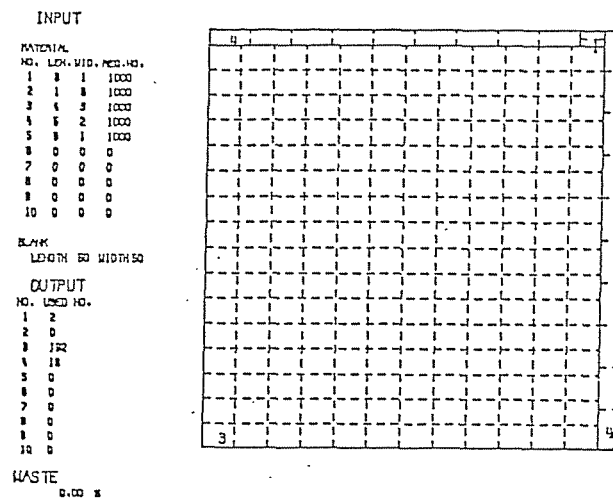


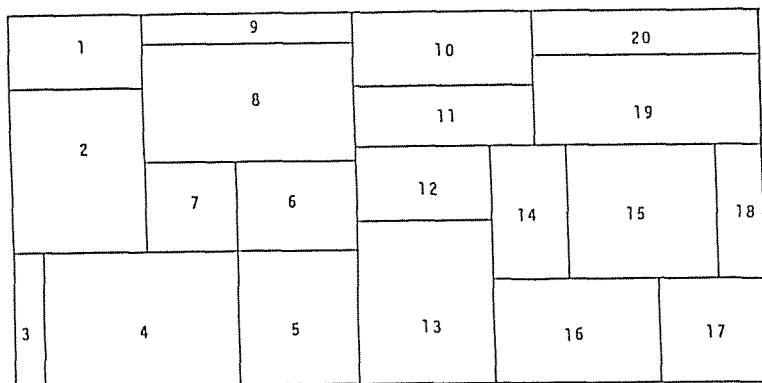
Fig. 4.4 One of graphic outputs in experiment 1 solved by the proposed method

Table 4.1 Comparison with Gilmore's D.P. result

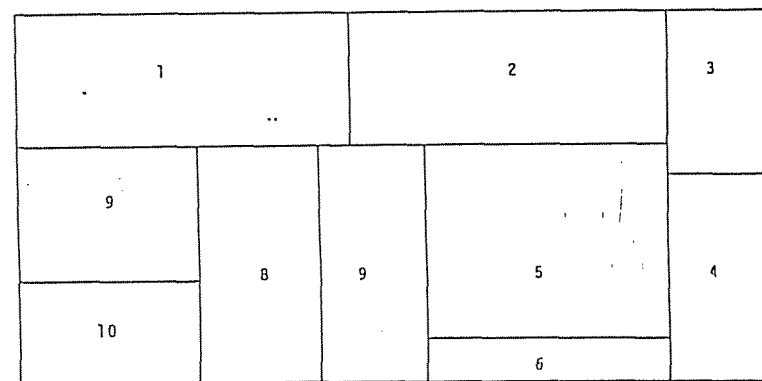
Experiment No.	Presented method		Gilmore's D.P.	
	Excution time(sec)	Waste ratio(%)	Excution time(sec)	Waste ratio(%)
1	1.39	0.00	3.10	0.00
2	1.17	0.40	3.04	0.00
3	1.28	0.16	3.10	0.00
4	1.45	0.00	3.12	0.00
5	1.34	0.16	3.08	0.16
6	1.42	0.20	3.07	0.00
7	1.23	0.00	3.11	0.00
8	1.22	0.36	3.08	0.00
9	1.22	0.04	3.10	0.00
10	1.20	0.00	3.12	0.00
Average	1.48	0.22	3.15	0.016

Experiment 2) In most of the cases, the exact optimum solution is unknown of the space allocation problem, so that it is difficult to estimate the goodness of the solution. In this experiment, the sizes of materials are produced by the given blank, and each size of ten materials are requested as user's product. Numerical experiments are executed to test the validity of the method. If ten blanks are required for the solution, the allocation obtained by the use of the proposed method is optimum. These experiments are made in PDP 11/20, and the program used is the same as the one used in the experiment 1. Tables. 4.2 - 4.4 list up the data of the material sizes and Figs. 4.5 - 4.7 are their original allocations corresponding to Tables 4.2 - 4.7 and the blank size is 1000 in length and 500 in width. The results are shown in Tables 4.5 - 4.7.

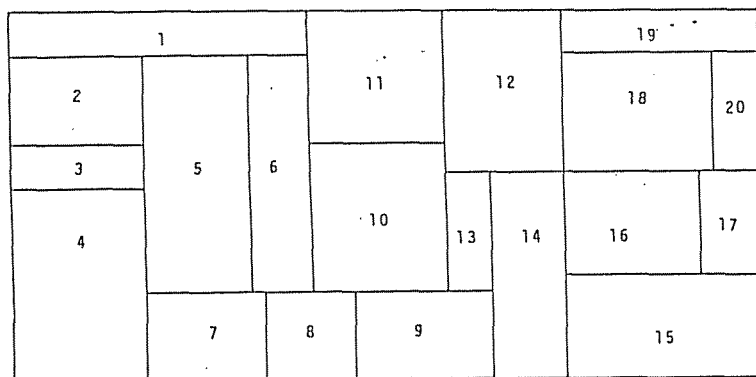
The solutions are practical enough because the waste ratio, when man allocates materials onto the blank, is more than 20 %.



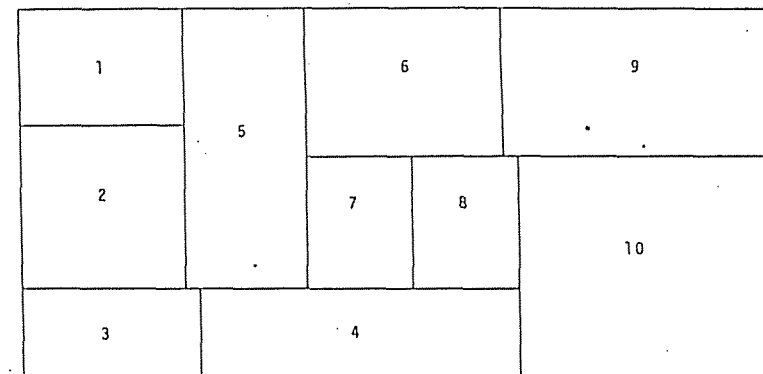
(a) No. 1



(a) No.1



(b) No. 2



(b) No.2

Fig. 4.6 Original allocation of tested materials (twenty kinds of materials)

Fig. 4.5 Original allocation of tested materials (ten kinds of materials)

1	4		8	9	10	11		14
2	5				12	13		
3	6	7						
22		21	20	19	18	17	16	15
23	25		26	27	28		29	30
24								

(a) No. 1

1		8		9	10	22		25	
2	3					23		26	
		4	5	11		12	13	24	
18	16			15	14			30	
	17								
6	7	19							

(b) No. 2

Fig. 4.7 Original allocation of tested materials (thirty kinds of materials)

Table 4.2 Ten kinds of materials input
data in experiment 2

No.	No. 1		No. 2	
	Length	Width	Length	Width
1	440	180	220	160
2	420	180	220	220
3	220	140	240	120
4	280	140	420	120
5	320	200	380	160
6	320	60	200	260
7	320	140	180	140
8	320	160	180	140
9	240	180	360	200
10	240	140	300	340

Table 4.3 Twenty kinds of materials input
data in experiment 2

No.	No. 1		No. 2	
	Length	Width	Length	Width
1	180	100	400	60
2	220	180	180	120
3	180	40	180	60
4	260	180	200	180
5	160	180	320	140
6	160	120	320	80
7	120	120	120	160
8	280	160	120	120
9	280	40	180	120
10	240	100	200	180
11	240	100	180	180
12	180	100	220	160
13	220	180	160	60
14	180	100	280	100
15	200	180	260	140
16	220	140	140	180
17	140	140	140	80
18	180	60	200	160
19	300	120	260	60
20	300	60	160	60

Table 4.4 Thirty kinds of materials
input data in experiment 2.

No.	No. 1		No. 2	
	Length	Width	Length	Width
1	5	3	13	4
2	5	6	9	9
3	5	3	9	4
4	11	3	9	5
5	11	6	8	3
6	4	3	9	8
7	11	3	5	3
8	4	9	10	6
9	11	3	6	4
10	6	2	6	4
11	8	2	10	10
12	6	9	10	4
13	8	9	10	4
14	13	11	9	4
15	13	7	6	4
16	5	7	7	3
17	3	8	7	3
18	6	8	6	3
19	3	8	3	3
20	3	7	7	3
21	8	7	4	3
22	9	7	13	6
23	4	9	13	5
24	9	2	13	5
25	6	8	6	6
26	3	6	6	5
27	3	6	6	5
28	6	9	6	9
29	2	7	9	2
30	16	7	11	9

Table 4.5 Result of experiment 2 (ten kinds of materials)

Activity	NO.1		NO.2	
	Waste ratio(%)	Required number	Waste ratio(%)	Required number
1	6.88	3	7.44	3
2	23.36	1	7.60	1
3	11.92	1	7.84	1
4	24.40	1	18.72	1
5	8.32	1	20.72	1
6	14.24	1	4.00	1
7	5.12	1	12.96	1
8	20.80	1	18.88	1
9	14.32	1	12.16	1
10	56.88	1	74.8	1
	Avelage	Total	Avelage	Total
	16.67	12	16.67	12

Table 4.6 Result of experiment of 2 (twenty kinds of materials)

Activity	NO.1		NO.2	
	Waste ratio(%)	Required number	Waste ratio(%)	Required number
1	0.48	1	0.78	1
2	1.36	1	0.47	1
3	2.56	1	5.06	1
4	3.04	1	1.23	1
5	5.84	1	0.72	1
6	0.48	1	0.62	1
7	4.32	1	2.85	1
8	6.72	1	2.08	1
9	2.08	1	2.15	1
10	4.8	1	2.03	1
11	68.32	1	83.2	1
	Avelage	Total	Avelage	Total
	9.01	11	9.01	11

Table 4.7 Result of experiment 2 (twenty kinds of materials)

Activity	NO.1		NO.2	
	Waste ratio(%)	Required number	Waste ratio(%)	Required number
1	7.84	1	5.36	1
2	1.28	1	2.00	1
3	5.12	1	2.24	1
4	1.60	1	3.04	1
5	6.40	1	1.36	1
6	1.12	1	7.92	1
7	4.00	1	7.36	1
8	6.88	1	9.76	1
9	3.36	1	8.32	1
10	6.40	1	14.08	1
11	56.00	1	38.56	1
	Avelage	Total	Avelage	Total
	9.01	11	9.01	11

Experiment 3) By generating uniform random numbers as ten kinds of blank sizes and material sizes as well, numerical experiments are executed in OKITAC-4500C. The range of length, width and the stock number of blanks generated by random numbers are 1000 to 1400, 500 to 1200, and 0 to 100 respectively. The range of length, width and the required number of materials tested are 10 to 90, 10 to 70, and 0 to 100 respectively. The result is shown in Table 4.8. The result shows that an average of waste ratio to be computed for the experiments is around 7.0%.

Experiment 4) By generating normal random number, numerical experiments are executed here as in the experiment 3. The random numbers generated for lengths, widths and stocked numbers of blanks are values of an average of 455, 355 and 50 respectively and of a standard deviation 151, 118 and 16 respectively. The result is shown in Table 4.9. The result shows that an average of the waste ratio is around 7.2%, a little bit more than one of the experiment 3.

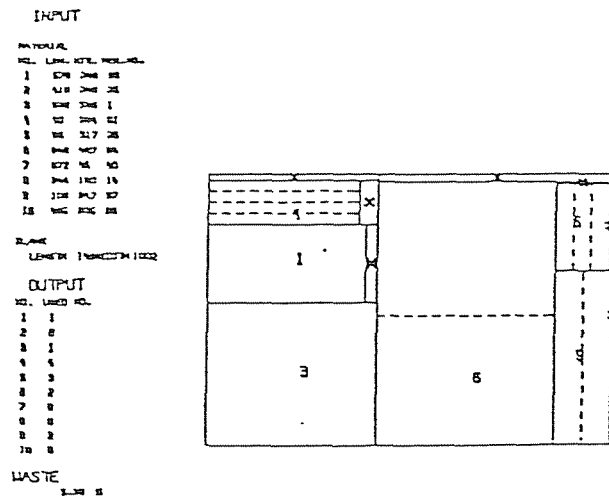
Examples of graphic outputs in experiments 3 and 4 are shown in Fig. 4.8.

Table 4.8 Result by uniform random numbers

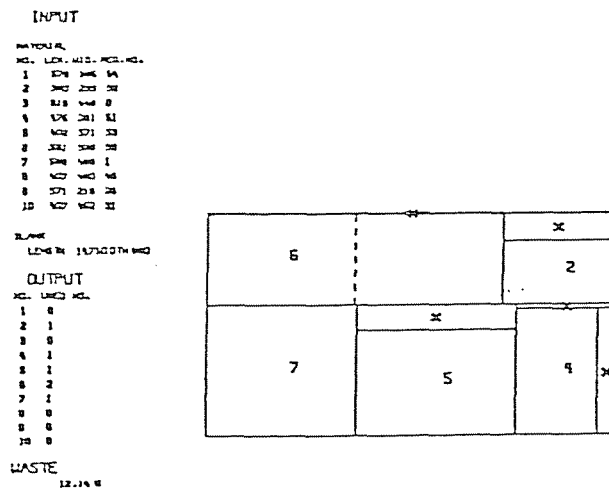
Experiment No.	The number of blanks to be determined	Execution time (min:sec)	Waste ratio (%)	Execution time per an assignment pattern(sec)
1	55	19:43	9.52	21.18
2	40	20:55	6.84	30.84
3	37	17:38	2.87	28.20
4	43	22:47	6.18	31.38
5	75	20:48	11.01	16.38
6	61	21:08	6.79	20.76
7	40	20:02	6.23	30.06
8	40	21:02	3.74	31.56
9	53	22:56	7.49	25.56
10	42	29:01	8.11	41.46
Average	48.8	21:04	6.99	26.40

Table 4.9 Result by normal random numbers

Experiment No.	The number of blanks to be determined	Execution time (min:sec)	Waste Ratio (%)	Execution time per an assignment pattern(sec)
1	54	22:12	6.62	24.67
2	55	21:25	5.10	23.36
3	60	21:16	8.95	21.27
4	87	23:25	7.57	16.15
5	60	24:29	9.16	24.48
6	54	23:55	6.36	26.57
7	49	24:10	5.01	29.54
8	60	21:15	9.09	21.25
9	71	16:07	8.46	13.60
10	60	19:59	5.98	19.98
Average	61.0	21:49	7.23	22.09



An example of outputs (uniform random numbers data).



An example of outputs (normal random numbers data).

Fig. 4.8 Examples of outputs in experiments

3 and 4

Experiment 5) In order to examine the character of the proposed method, three kinds of distribution area of the materials are tested by PDP 11/20 when three kinds of the blanks are given. We set ten to the material kinds and to the number of materials as well. The number of the given blanks are assumed to be unlimited. The area of the materials is distributed in large size in the experiment A, in small size in the experiment B and in all sizes in the experiment C. These data are produced by computer random function. The data and their results are shown in Tables 4.10-4.11. The distribution of the material area is shown by the ratio of the number of materials to all the number of materials generated.

In this experiment, when the material areas to the blank ones are relatively large, the waste becomes large, when small, the waste becomes small, and when uniformed, the waste becomes small. The waste ratio is less than twenty percent in any case except for only one in the experiment A.

Table 4.10 Result of experiment 6

TEST NO. D.M.A.	A	B	C
$M < 5 \times 10^3$	20	0	10
$5 \times 10^3 \leq M < 1 \times 10^4$	40	20	0
$1 \times 10^4 \leq M < 1.5 \times 10^4$	20	30	10
$1.5 \times 10^4 \leq M < 2.0 \times 10^4$	10	20	10
$2.0 \times 10^4 \leq M$	10	30	70
BLANK			
3' x 6'	40	20	10
4' x 8'	30	40	70
5' x 10'	30	40	20
TOTAL	100	100	100
AVERAGE WASTE(%)	4.00	6.42	16.66

Table 4.11 Result of experiment 6

TEST NO. D.M.A.	A	B	C
$M < 5 \times 10^3$	70	20	5
$5 \times 10^3 \leq M < 1 \times 10^4$	20	5	0
$1 \times 10^4 \leq M < 1.5 \times 10^4$	5	10	0
$1.5 \times 10^4 \leq M < 2.0 \times 10^4$	5	10	10
$2.0 \times 10^4 \leq M$	0	55	85
BLANK			
3' x 6'	30	20	20
4' x 8'	10	50	40
5' x 10'	60	30	40
TOTAL	100	100	100
AVERAGE WASTE(%)	2.73	7.20	13.18

D.M.A.: Distribution of material areas

Experiment 6) As to many numbers of the materials, the experiments are executed in PDP 11/20. A thousand materials are produced out of ten kinds of materials by the computer random number. The material kinds are three and the number of materials required by users are unlimited. The distribution of the material area and the results are shown in Table 4.12. By this experiment it may be said that the larger the number of materials is, the smaller the waste ratio becomes.

Table 4.12 (a) Result of experiment 6

TEST NO. D.M.A.	1	2	3	4	5	
$M < 2.5 \times 10^5$	70	30	30	40	50	
$2.5 \times 10^5 \leq M < 5 \times 10^5$	10	30	40	30	20	
$5 \times 10^5 \leq M < 7.5 \times 10^5$	0	20	40	20	30	
$7.5 \times 10^5 \leq M < 10^6$	10	10	0	10	0	
$10^6 \leq M$	10	10	0	0	0	
BLANK						TOTAL
3' x 6'	25	20	4	8	3	60
4' x 8'	0	4	8	0	6	18
5' x 10'	0	2	4	5	3	14
TOTAL	25	26	16	13	12	92
AVERAGE WASTE(%)	13.24	12.57	19.68	13.27	15.58	14.87

M Area of material

(b)

TEST NO. D.M.A.	1	2	3	4	5	
$M < 2.5 \times 10^5$	30	50	40	60	20	
$2.5 \times 10^5 \leq M < 5 \times 10^5$	30	20	30	20	40	
$5 \times 10^5 \leq M < 7.5 \times 10^5$	30	20	10	0	10	
$7.5 \times 10^5 \leq M < 10^6$	0	0	10	10	20	
$10^6 \leq M$	10	10	10	10	10	
BLANK						TOTAL
3' x 6'	0	9	8	12	0	29
4' x 8'	6	7	3	2	15	33
5' x 10'	8	2	8	3	4	25
TOTAL	14	18	19	17	19	87
AVERAGE WASTE(%)	15.41	10.94	18.02	6.40	16.10	13.37

(c)

TEST NO. D.M.A.	1	2	3	4	5	
$M < 2.5 \times 10^5$	0	20	0	0	0	
$2.5 \times 10^5 \leq M < 5 \times 10^5$	10	30	40	30	50	
$5 \times 10^5 \leq M < 7.5 \times 10^5$	50	10	30	50	40	
$7.5 \times 10^5 \leq M < 10^6$	20	40	20	0	0	
$10^6 \leq M$	20	0	10	20	10	
BLANK						TOTAL
3' x 6'	5	1	3	5	7	21
4' x 8'	6	16	8	11	8	49
5' x 10'	15	5	12	10	11	53
TOTAL	26	22	23	26	26	123
AVERAGE WASTE(%)	19.24	15.87	21.05	19.59	21.19	19.39

Experiment 7) The recursive approximation procedure gives us the rule that the materials are allocated according to the area sizes---the larger, the faster. When there are some long strip materials, it may be said that the materials are allocated according to their length---the longer, the faster. To assert this fact, the experiment is executed in PDP 11/2. Figs 4.9 and 4.10 are the results of this experiment. In this experiment, the results are satisfactory, but in general, much more experiments are expected to be made.

```

R OPTV2X1
***** SHEAR OPTIMIZATION PROGRAM *****

*** INITIAL DATA SET

* BLIN

** INPUT BLANK LODE 2

2 0 BLANKS YOU HAVE NOW ARE

      BLANK NO      SIZE      NUMBER
      1      1829 00 * 914 00      224

* BLIN

*** INPUT MATERIAL SIZE & NUMBER TO BE SHEAR

      X,Y,N ? 1800 ,25 ,1
      X,Y,N ? 1700 ,20 ,2
      X,Y,N ? 500 ,450 ,2
      X,Y,N ? 250 ,200 ,3
      X,Y,N ? 400 ,600 ,2
      X,Y,N ? 300 ,170 ,6
      X,Y,N ?

* M001

*** MATERIAL REGISTERED

      NO. 1      1800 00 * 25 00      1
      NO. 2      1700 00 * 20 00      2
      NO. 3      500 00 * 450 00      2
      NO. 4      250 00 * 200 00      3
      NO. 5      400 00 * 600 00      2
      NO. 6      300 00 * 170 00      6

..... Given material size

* SINT

*** OPTIMIZED RESULT NO. 1

      BLANK USED      NO. 1
      NUMBER OF BLANK      1 S
      WASTE PERCENTAGE      15.611 %
      NUMBERS OF MATERIAL TO BE SHEARED PER ONE NO. 1 BLANK

      SIZE      NUMBER
      NO. 1      1800 00 * 25 00      1
      NO. 2      1700 00 * 20 00      2
      NO. 3      500 00 * 450 00      2
      NO. 5      400 00 * 600 00      2
      NO. 6      300 00 * 170 00      6

* NEXT

*** OPTIMIZED RESULT NO. 2

      BLANK USED      NO. 1
      NUMBER OF BLANK      1 S
      WASTE PERCENTAGE      40.616 %
      NUMBERS OF MATERIAL TO BE SHEARED PER ONE NO. 1 BLANK

      SIZE      NUMBER
      NO. 4      250 00 * 200 00      3

* NEXT

*** TOTAL USAGE OF BLANK ARE

      NO. 1 1829 00 * 914 00      2

*** TOTAL PRODUCT OF MATERIALS ARE

      NO. 1 1800 00 * 25 00      1
      NO. 2 1700 00 * 20 00      2
      NO. 3 500 00 * 450 00      2
      NO. 4 250 00 * 200 00      3
      NO. 5 400 00 * 600 00      2
      NO. 6 300 00 * 170 00      6

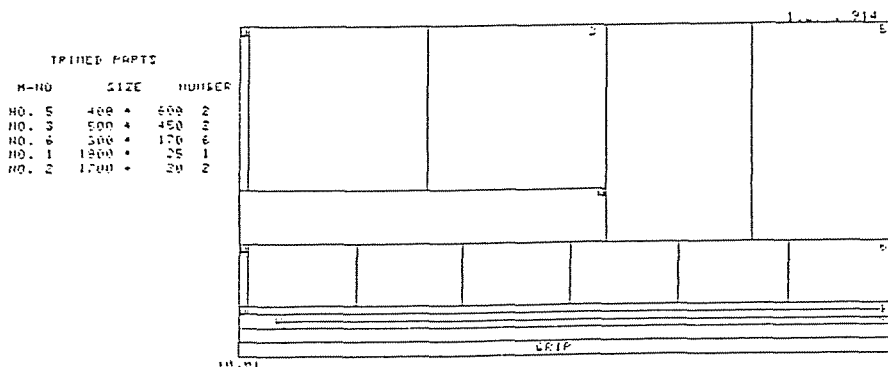
..... Result of nesting

***** HTI *****

```

*** OPTIMIZED RESULT NO. 1

TEIR LAYOUT



..... Material allocation

Fig. 4.9 Result of experiment 7 (Area criterion)


```

R 0011
***** SHEAR OPTIMIZATION PROGRAM *****

*** INITIAL DATA SET

* BEGIN

** INPUT BLANK CODE 2.

2 0 000000 YOU HAVE NOW ARE

      BLANK NO      SIZE      NUMBER
      1      1029 00 * 514 00      1 ..... Given blank size

* BEGIN

*** INPUT MATERIAL SIZE & NUMBER TO BE SHEAR

      X,Y,N ? 1000 .25 .1
      X,Y,N ? 1700 .20 .2
      X,Y,N ? 500 .450 .2
      X,Y,N ? 250 .200 .3
      X,Y,N ? 400 .600 .2
      X,Y,N ? 300 .170 .6 ..... Given material size

* PLUT

*** MATERIAL REGISTERED

      NO. 1      1000 00 * 25 00      NUMBER 1
      NO. 2      1700 00 * 20 00      2
      NO. 3      500 00 * 450 00      2
      NO. 4      250 00 * 200 00      3
      NO. 5      400 00 * 600 00      2
      NO. 6      300 00 * 170 00      6

* LENGTH

* SIKT

-- *** OPTIMIZED RESULT NO. 1
      BLANK USED      NO. 1
      NUMBER OF BLANK      1 5
      WASTE PERCENTAGE      6.22%
      NUMBERS OF MATERIAL TO BE SHEARED PER. (ONE NO. 1 BLANK)
      SIZE      NUMBER
      NO. 1      1000 00 * 25 00      1
      NO. 2      1700 00 * 20 00      2
      NO. 3      500 00 * 450 00      2
      NO. 4      250 00 * 200 00      3
      NO. 5      400 00 * 600 00      2
      NO. 6      300 00 * 170 00
      .... Result of nesting

* NEXT

*** TOTAL USAGE OF BLANK ARE
      NO. 1 1029 00 * 514 00      1

*** TOTAL PRODUCT OF MATERIALS ARE
      NO. 1 1000 00 * 25 00      1
      NO. 2 1700 00 * 20 00      2
      NO. 3 500 00 * 450 00      2
      NO. 4 250 00 * 200 00      3
      NO. 5 400 00 * 600 00      2
      NO. 6 300 00 * 170 00      6

*** DYE ***

STOP --

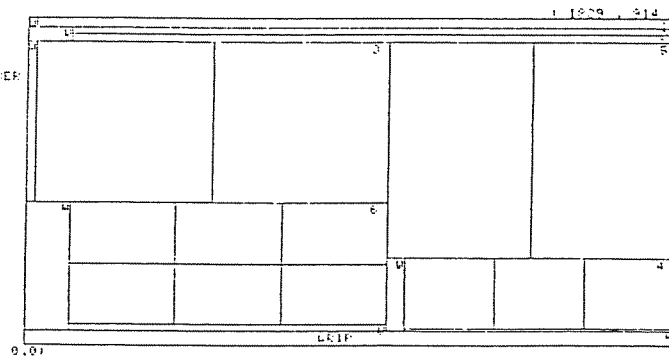
```

** OPTIMIZED RESULT NO.

TECH (40000)

TRAINED PARTS

N-NO	SIZE	NUMBER
NO. 1	1000 *	25 1
NO. 2	1700 *	20 2
NO. 3	500 *	450 2
NO. 4	250 *	200 3
NO. 5	400 *	600 2
NO. 6	300 *	170 6



..... Material allocation

Fig. 10 Result of experiment 7 (Length criterion)

4.5 Conclusion

The practical method for the cutting stock problem is proposed, and the efficiency and validity are discussed by the execution of the numerical experiments. Through the discussion, the followings are concluded.

1. The allocation procedure of the materials to the blank is proposed by formulating the problem as a recursive approximation.
2. A simple and practical procedure to determine the number of the blanks allocated by the materials are proposed.
3. The numerical experiments show the waste ratio is less than twenty percent. This means the proposed method is good enough to be used practically.
4. The memory size required for this proposed method is about 22 KW. A mini-computer is available for this method.

This method is now adopted for sheet metal shear process planning at Murata Machinery Company, and a great deal of planning time has been saved in sheet metal manufacturing. The system using the proposed method is

named "OPTI-CUT" and applied to produce an NC shearing tape. An example of NC tape producing is shown in Fig. 4.11. In this system, edge trimming and repositioning problems are also considered. A system manual is given in Appendix 2 and we can see how to treat edge trimming and repositioning in Appendix 1. And that, because of the small-sized memory, a micro-processor is available for this program currently.

Furthermore, this method is adopted for the integrated sheet metal manufacturing system. This is described in chapter 8.

***** SHEAR OPTIMIZATION PROGRAM *****

*** INITIAL DATA SET

* MAIN

*** INPUT MATERIAL SIZE & NUMBER TO BE SHEAR

X,Y,N ? 400.,340.,20
X,Y,N ? 300.,120.,20
X,Y,N ? 300.,200.,20
X,Y,N ? 600.,400.,1
X,Y,N ? 250.,150.,20
X,Y,N ? 180.,100.,20
X,Y,N ? 70.,50.,50
X,Y,N ?

* START

*** OPTIMIZED RESULT

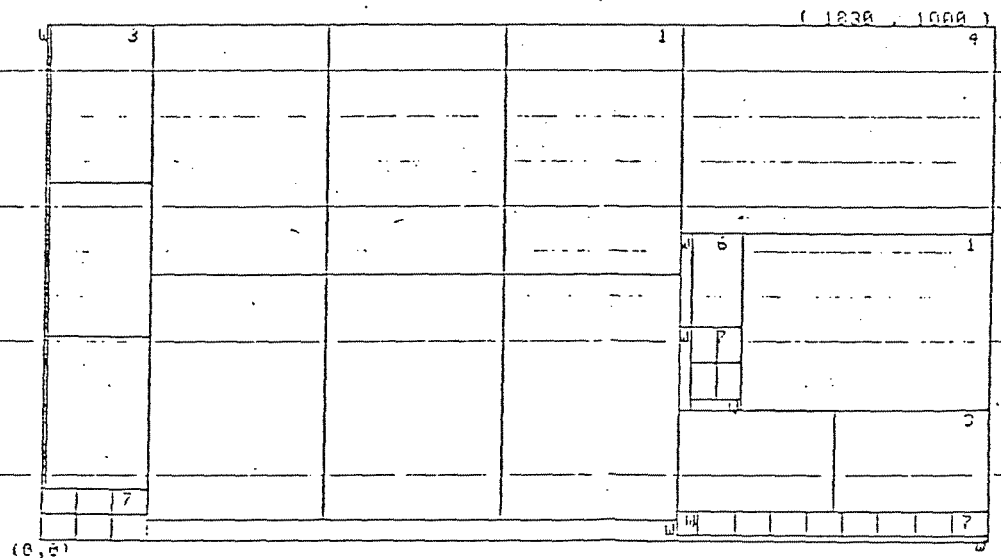
BLANK USED NO. 1
NUMBER OF BLANK 15
WASTE PERCENTAGE 3.639 %
NUMBERS OF MATERIAL TO BE SHEARED PER. ONE NO. 1 BLANK

	SIZE	NUMBER
NO. 1	400.00 * 340.00	7
NO. 3	300.00 * 200.00	5
NO. 4	600.00 * 400.00	1
NO. 6	180.00 * 100.00	1
NO. 7	70.00 * 50.00	18

* GRPH

* HCPY

TRIM LAYOUT



(NC tape information)

* LIST

N001	G00	X	123000Y	60000	N024	X	21000Y	52000
N002	X	135000Y	26000	N025	X	85000Y	4000	
N003	X	125000Y	42000	N026	X	55000Y	4000	
N004	X	123000Y	42000	N027	X	21000Y	4000	
N005	X	130000Y	35000	N028	X	123000Y	000	
N006	X	125000Y	35000	N029	X	1000Y	70000	
N007	X	130000Y	28000	N030	X	1000Y	40000	
N008	X	125000Y	28000	N031	X	1000Y	10000	
N009	X	125000Y	26000	N032	X	51000Y	000	
N010	X	123000Y	26000	N033	X	21000Y	000	
N011	X	153000Y	6000	N034	X	000Y	44000	
N012	X	123000Y	6000	N035	X	000Y	10000	
N013	X	176000Y	1000	N036	X	14000Y	5000	
N014	X	169000Y	1000	N037	X	7000Y	5000	
N015	X	162000Y	1000	N038	X	000Y	5000	
N016	X	155000Y	1000	N039	X	14000Y	000	
N017	X	148000Y	1000	N040	X	7000Y	000	
N018	X	141000Y	1000	N041	X	000Y	000	
N019	X	134000Y	1000	N042	X	185000Y	120000	
N020	X	127000Y	1000					
N021	X	123000Y	1000					
N022	X	85000Y	52000					
N023	X	55000Y	52000					

Fig. 4.11 A generation of an NC tape information of shearing machine.

Reference

1. S.G. Hahn, "In the Optimul Cutting of Defective Sheets", Opns. Res., 11, (1976) p1100.
2. P.C. Gilmore, et. al., "A Linear Programming Approach to the Cutting Stock Problem", Opns. Res., 9, (1961) p94.
3. P.C. Gilmore et. al., "Many Stage Cutting Stock Problems of Two and More Dimensions", Opns. Res., 13, (1965) p863.
4. P.C. Gilmore, et. al., "The Theory of Computation of Knapsack Functions", Opns. Res., 14, (1966) p1045.
5. Y. Kakazu, et. al., "The Optimum Trimming of Rectangular Plates", Bull. of Prec. Ergg., 9, No.5, (1976).
6. M. Furukawa, et. al., "A Practical Method of Solving Flow Shop Scheduling Problems", J. of J.S.P.E., 43, No.2, (1976) p22.

5 An Approach to the Package Problem

---Optimum Packing and Loading---

5.1 Introduction

A complex problem may arise when a number of products are made in a wide variety of sizes. Each of the products is packed in an individual cardboard box, and then they are packed in an individual carton box. The number of carton boxes is usually twelve or twenty-four. Some carton boxes are loaded on a pallet. The pallet size is usually decided according to the size of a truck cargo space, or a container box. The packaging engineer must design a size of the cardboard and the carton box and a method of loading the given number of carton boxes of certain sizes. This decision making is clearly needed as one of three-dimensional space allocation problems in a situation of optimum packing, for example, where carton boxes are loaded on the pallet so as to minimize empty spaces.

This chapter deals with the above mentioned problem and develops a simplified formulation with a new method to solve the problem.

To begin with, the problem in an actual situation is analyzed and formulated. Then a theoretical method is applied to the solution of the problem. The applied method is a kind of enumerative one, and by using it, an optimum carton size can be determined, then the ways of packing and loading are presented.

5.2 Problem Description

The problem can be summarized as follows. (Fig. 5.1)

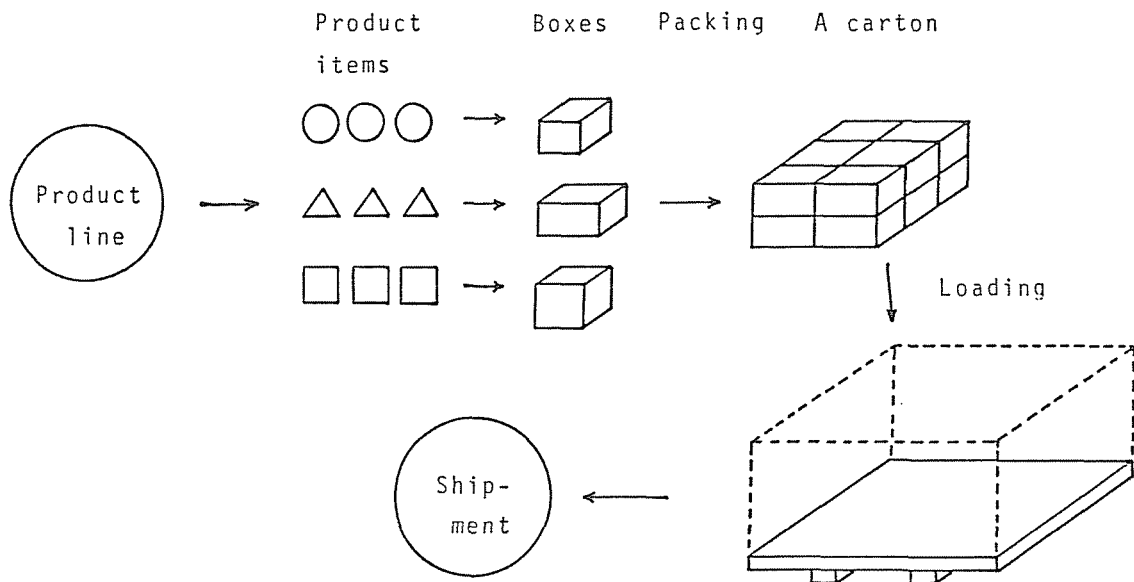


Fig. 5.1 A flow of product items on the shop

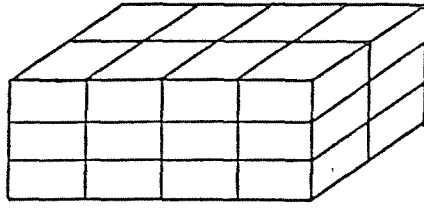


Fig. 5.2 Regular mesh-like packing into a carton

L, W and H are the length, width and height of a product box and the operation // is used as $L//x$, which means the length of the product box is located in parallel with x axis

	ope.	1	2	3	4	5	6
L	//	X	X	Y	Y	Z	Z
W	//	Y	Z	X	Z	X	Y
H	//	Z	Y	Z	X	Y	X

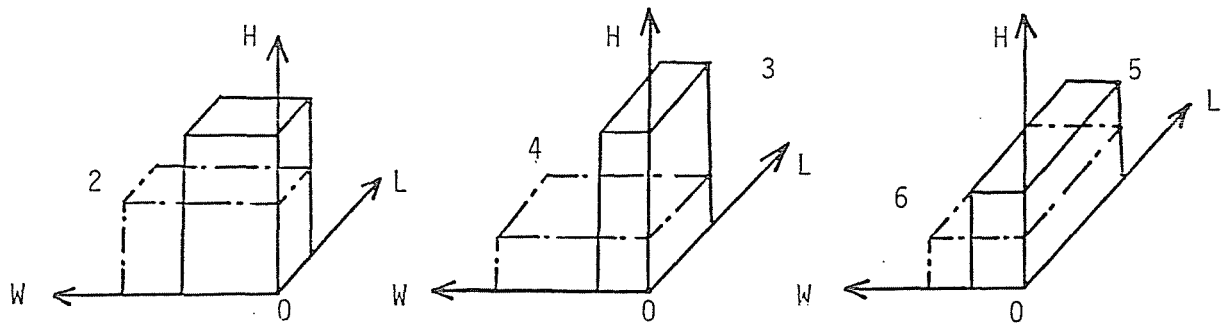


Fig. 5.2 Six packing ways

- a. A company has about 200 product lines which collectively represent 600 to 700 hundred product items.
- b. All product items are usually packed with every twenty-four products per a carton.
- c. The company has a standard size pallet for loading carton boxes. The dimensions of the pallet are 48 in length, 40 in width and 42 in height.
- d. All distributors must order standard pallet loads of a product item; they are not allowed to order more than a full pallet loads and they cannot have more than one product item on the same pallet.
- e. The company wants to prepare several standard-sized carton boxes which will result in a higher utilization of the pallet.

The object of the problem is to find out the optimum sizes of the carton boxes for a highest utilization of the pallet when the sizes of the product boxes are given, and to find out the standard sizes of the carton boxes for the highest utilization of the pallet when widely distributed sizes of the boxes are given.

5.3 Formula of the Problem

5.3.1 Notations and Assumptions

Let us define some notations and assumptions:

a_1, a_2, a_3	:	The dimensions of a product item box.
b_1, b_2, b_3		The limitations of the dimensions of a carton box.
c_1, c_2, c_3		The dimensions of the pallet volume.
x_1, x_2, x_3		The number of product item boxes making a carton according to the length, width and the height direction of the carton box.
y_1, y_2, y_3		The number of carton boxes, which are loaded on the pallet, according to the length, width and the height direction of the pallet.
S		Waste or trim loss.

Assumptions:

- a - 1 Each product item should be packed into a cubic box of x , y and z dimensions.
- a - 2 Only regular mesh-like allocation is acceptable for boxes to make a carton and for cartons being loaded on the pallet.

All the variables used here are positive numbers, and x_i and y_i ($i = 1, 2, 3$) are integer numbers.

If other shapes of boxes, for example a cylindrical can, are used, there is still the unsolved packing problem. Hence, there is no solution method with respect to this case, but we can achieve an approximate solution by substituting the cylindrical can for the box type container.

With respect to a - 2, there is no solution method for the packing and loading problem no matter whether auto-packing and auto-loading, or hand-packing and hand-loading are adopted. The most acceptable approach will be to make it easy for a worker to understand what he should do. Since the regular mesh-like allocation does this, this assumption is acceptable.

5.3.2 Constraints and Object Function

The problem is formulated as follows, where a_i , z_i and c_i ($i = 1, 2, 3$) are given and x_i , y_i and b_i is unknown.

Let us define n as the number of boxes per a carton

$$\prod_{i=1}^3 x_i = n \quad (5.1)$$

and

$$a_i x_i \leq z_i \quad (i = 1, 2, 3) \quad (5.2)$$

The size of a carton is limited within b_i ($i = 1, 2, 3$),

Then,

$$z_i \leq b_i \quad (i = 1, 2, 3) \quad (5.3)$$

Or

$$a_i x_i \leq b_i \quad (i = 1, 2, 3). \quad (5.3')$$

If a_i ($i = 1, 2, 3$) is unknown, the packing problem is to determine a_i and x_i ($i = 1, 2, 3$) such as

$$\min_{a_i, x_i} \quad \prod_{i=1}^3 z_i - \prod_{i=1}^3 a_i x_i, \quad (5.4)$$

subj. to Eqs. 5.1, 5.2 and 5.3.

Eq. 5.4 implies that a total waste volume of a carton box is minimized by the product boxes packing.

If a_i ($i = 1, 2, 3$) is given, Eq. 5.4 becomes constant because of Eq. 5.1. Hence the packing problem is transformed into the problem in which x_i ($i = 1, 2, 3$) should be found.

In the next step, as many carton boxes must be loaded as possible. As the carton must be loaded within the size of a pallet c_i ($i = 1, 2, 3$), the following constraint must be satisfied,

$$c_i \geq z_i \quad y_i \quad (i = 1, 2, 3). \quad (5.5)$$

Then the loading problem becomes

$$\min. \quad \prod_{i=1}^3 c_i - \prod_{i=1}^3 z_i y_i \quad (5.6)$$

subj. to Eq. 5.5.

Eq. 5.6 implies that the total waste volume of a pallet is minimized by the carton box loading.

Therefore, both of the problems are reduced to such a problem as

$$\min. \quad \prod_{i=1}^3 c_i - \prod_{i=1}^3 z_i y_i$$

x_i, y_i, z_i

subj. to Eqs. 5.1, 5.2, 5.3 and 5.5.

5.3.3 Qualifying the Formulas

The problem is qualified by changing the problem 5.6. Supposing that the packing method has already been determined, the problem 5.6 is rewritten as

$$\max_{y_i} \quad \prod_{i=1}^3 z_i \cdot y_i, \quad (5.7)$$

subj. to Eq. 5.5.

Eq. 5.7 implies that minimizing the total waste volume is equal to maximizing the total carton volumes on the pallet.

The solutions of y_i in the problem 5.7
subject to Eq. 5.5 become

$$y_i = [c_i/x_i], \quad (5.8)$$

where, $[]$ is a gaussian notation. Therefore, the
problem is reduced to

$$\begin{aligned} \max. \quad & \prod_{i=1}^3 z_i [c_i/z_i], \\ \text{subj. to} \quad & \text{Eq. 5.2.} \end{aligned} \quad (5.9)$$

In considering that the standard size of a
carton box is just fitted to the size of packed
product item boxes per a carton,

$$a_i x_i = z_i \quad (i = 1, 2, 3) \quad (5.10)$$

is obtained from Eq. 5.2. By substituting Eq. 5.10
to Eq. 5.9, we may simplify the problem 5.9 as

$$\begin{aligned} \max. \quad & \prod_{i=1}^3 a_i x_i [c_i/a_i x_i], \\ \text{subj. to} \quad & \text{Eq. 5.2.} \end{aligned} \quad (5.11)$$

From Eq. 5.1, we can reduce the problem to

$$\begin{aligned} \max. \quad & \prod_{i=1}^3 [c_i/a_i x_i], \\ \text{subj. to} \quad & \text{Eq. 5.2.} \end{aligned} \quad (5.12)$$

because $\prod_{i=1}^3 a_i x_i = n \prod_{i=1}^3 a_i$ is constant.

5.4 Enumeration Method

If feasible solutions can be enumerated, we can find the optimum solution in the problem 5.12 by testing all the solutions. In order to enumerate the feasible solutions, let us define FS as a set of feasible solutions:

$$\text{FS} = \{(x_1, x_2, x_3) \mid \prod_{i=1}^3 x_i = n, \\ x_i \text{ is integer number}\} \quad (5.13)$$

FS can be constructed from an order set OS by combining each element of OS.

$$\text{OS} = \{e \mid e = \lfloor n/j \rfloor\}, j = 1, 2, \dots, n \quad (5.14)$$

where $\lfloor \cdot \rfloor$ is a gaussian notation. We rewrite OS by giving elements e suffices such as e_1, e_2, \dots, e_q , from smaller number to larger one where q is the number of elements e_i

$$\text{OS} = \{e_i \mid i = 1, 2, \dots, q\}. \quad (5.15)$$

Let us define $p_i = \lfloor b_i/a_i \rfloor$, ($i = 1, 2, 3$) and $r_i = \min(p_i, e_q)$. Then, let us define a set of x_i ($i = 1, 2, 3$) as X_i :

$$X_i = \{x_i \mid 1 \leq x_i \leq r_i, x_i \in \text{OS}\}, \quad i = 1, 2, 3 \quad (5.16)$$

We rewrite X_i as

$$X_i = \{x_{i1}, x_{i2}, \dots, x_{it_i}\}$$

where x_{ij} ($j = 1, 2, \dots, t_i$) is all the elements x_i of X_i , and the suffix t_i shows the number of all the elements x_i .

Then, we can obtain FS by enumerating

$$\text{FS} = \{(x_{1u}, x_{2v}, x_{3w}) \mid x_{3w} = n/x_{1u} x_{2v}\}, \\ u = 1, 2, \dots, t_1, v = 1, 2, \dots, t_2.$$

We rewrite FS as

$$\text{FS} = \{x_{1j}, x_{2j}, x_{3j}\}, \quad j = 1, 2, \dots, t,$$

where t is the number of all the feasible solutions.

Thus, the problem 5.12 is transformed to

$$\max_{j=1}^t \quad \prod_{i=1}^3 [c_i/a_i \cdot x_{ij}]. \quad (5.17)$$

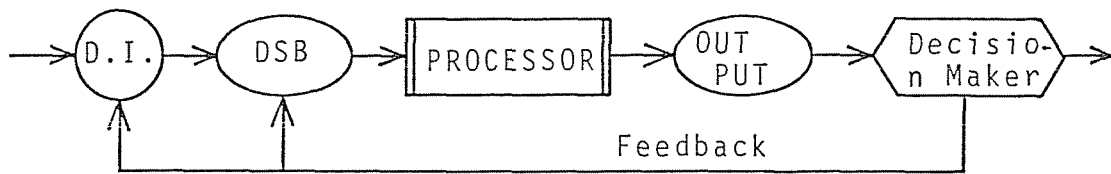
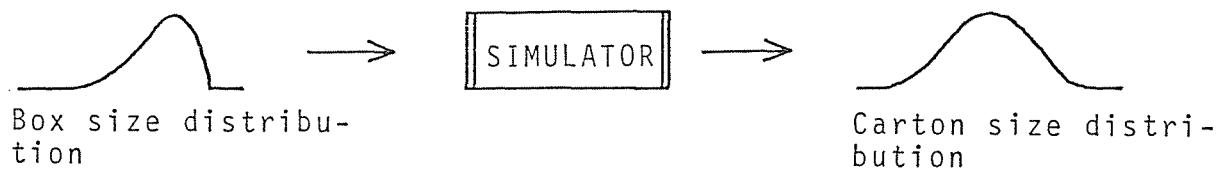
When product item boxes make up a carton, there are six packing ways of box allocation. Therefore, the problem 5.17 must be calculated six times corresponding to the box allocations. Related to six packing ways, we add the suffix k to x_{ij} as x_{ij}^k . Then the problem 5.17 becomes

$$\max_{k=1}^6 \quad \max_{j=1}^t \prod_{i=1}^3 [c_i/a_i x_{ij}^k]. \quad (5.18)$$

Once the solution of how product item boxes are packed into a carton is obtained, the loading method is easily figured out by the use of Eq. 5.8.

5.5 Determination of Standard Carton Box

In the previous method proposed, there is assumption that the size of packed product item boxes per a carton becomes the size of a standard carton box. However, one of our objects is to design the size of the standard carton boxes. To develop such a design method, we use the enumeration method proposed as the simulator. If the size of the product item boxes belongs to a certain distributing function, the size of carton boxes will be expected to have distribution. By using the proposed method as the simulator, we can obtain this distributing character.. After recognizing this distributing character, we may determine the standard carton boxes which covers the calculated carton size by modifying the size to a larger one with a little fraction. If a little modification for the product item boxes is allowed, we can feed back the results of the standard box size to modify the product item boxes. Thus, we can also design the product item boxes. This process is shown in Fig. 5.4.



D.I. ; Data input.
 DSB ; Box size
 distribution.

Fig. 5.4 A feedback system for determination of standard carton sizes and box sizes

5.6 Numerical Experiments

In order to show the validity of the developed theories and algorithms, and an example of how to calculate each available equation, several results will be presented with detailed processing calculations. Experiments are executed by OKITAC 4500-C and the program is coded with FORTRAN. Table 1 presents the precalculated data for determining the feasible solution. It shows that the pallet dimensions as input data are (48, 40, 42), (length x width x height), and each product item should be packed 24 boxes to a carton. An ordering set OS is shown here, too. The values of j in Table 5.1 shows the method of packing product item boxes to a carton.

Table 5.1 Feasible solutions FS

J	x_{1j}	x_{2j}	x_{3j}	J	x_{1j}	x_{2j}	x_{3j}	J	x_{1j}	x_{2j}	x_{3j}
1	1	1	24	11	2	3	4	21	4	3	2
2	1	2	12	12	2	4	3	22	4	6	1
3	1	3	8	13	2	6	2	23	6	1	4
4	1	4	6	14	2	12	1	24	6	2	2
5	1	6	4	15	3	1	8	25	6	4	1
6	1	8	3	16	3	2	4	26	8	1	3
7	1	12	2	17	3	4	2	27	8	3	1
8	1	24	1	18	3	8	1	28	12	1	2
9	2	1	12	19	4	1	6	29	12	2	1
10	2	2	6	20	4	2	3	30	24	1	1

$$n = 24$$

$$OS = \{1, 2, 3, 4, 6, 8, 12, 24\}$$

Efficiency is defined to estimate the solution as

$$\text{Efficiency} = \frac{\text{Total product item box volume}}{\text{Pallet volume}} \cdot 100(\%).$$

Experiment 1 The input as the product item box dimensions are (1, 1, 1). In this case we can get ten cases which give an optimum solution (Table 5.2). Optimum solutions are marked by * in Table 5.2.

Experiment 2 The input as the product item box dimensions are (2, 2, 2). In this case, we can have seven optimum solutions. The results are shown in Table 5.3 (a)-(c). Optimum solutions are marked by * in Table 5.3.

Experiment 3 The input as the product item box dimensions are (4.5, 6.5, 9.5). In this case, we can get nine optimum solutions. However, the efficiency is 82.7% in optimum solution. The results are shown in Table 5.4 (a)-(f). Optimum solutions are marked by * as well. With respect to this experiment, the optimum solution efficiency is 82.7%. If the dimensions of the product item boxes are unchangable, there is no room to improve the efficiency. However, if they are changed as (4.5 → 4.0, 6.5 → 6.9, 9.5 → 10.0), the total volume of the boxes is almost the

Table 5.3 (a) Result of experiment 2

k		j	y ₁	y ₂	y ₃	n y _i	Total Volume	Efficiency (%)
1	—	1	24	20	—	—	—	—
1		2	24	10	1	240	69120	85.71
1		3	24	6	1	144	41472	51.14
1		4	24	5	2	240	69120	85.71
1		5	24	3	3	216	62208	77.14
1		6	24	2	4	192	55296	68.57
1	—	7	24	—	—	—	—	—
1	—	8	24	—	—	—	—	—
1		9	12	20	7	240	69120	85.71
1		10	12	10	2	240	69120	85.71
1		11	12	6	3	216	62208	77.14
1		12	12	5	4	240	69120	85.71
1		13	12	3	7	252	72576	90.0
1		14	12	1	14	168	48384	60.0
1		15	8	20	2	160	46080	57.14
1		16	8	10	4	240	69120	85.71
1	*	17	8	5	7	280	80640	100.0
1		18	8	2	14	224	64512	80.0
1		19	6	20	2	240	69120	85.71
1		20	6	10	4	240	69120	85.71
1		21	6	3	7	126	36288	45.0
1		22	6	2	14	168	48384	60.0
1		23	4	20	3	240	69120	85.71
1		24	4	10	7	210	60480	75.0
1		25	4	5	14	210	60480	75.0
1		26	3	20	4	240	69120	85.71
1		27	3	6	14	252	72576	90.0
1	*	28	2	20	7	280	80640	100.0
1	*	29	2	10	14	280	80640	100.0
1	*	30	1	20	14	280	80640	100.0

Table 5.2 Result of experiment 1

k		j	y ₁	y ₂	y ₃	n y _i	Total Volume	Efficiency (%)
1		1	48	40	1	1920	46080	57.14
		2	48	20	3	2880	69120	85.71
		3	48	13	5	3120	74880	93.03
	*	4	48	10	7	3360	80640	100.0
		5	48	6	10	3024	72576	90.0
	*	6	48	5	14	3360	80640	100.0
		7	48	3	21	3024	72576	90.0
		8	48	1	42	2016	48384	60.0
		9	24	40	3	2880	69120	85.71
		10	24	20	7	3360	80640	100.0
		11	24	13	10	3120	74880	93.03
		12	24	10	12	2880	69120	85.71
		13	24	6	21	3024	72576	90.0
		14	24	3	42	3024	72576	90.0
		15	16	40	5	3200	76800	95.23
		16	16	20	10	3200	76800	95.23
	*	17	16	10	21	3360	80640	100.0
	*	18	16	5	42	3360	80640	100.0
	*	19	12	40	7	3360	80640	100.0
		20	12	20	13	3120	74880	93.03
		21	12	13	21	3276	78624	97.5
		22	12	6	42	3024	72576	90.0
		23	8	40	10	3200	76800	95.23
	*	24	8	20	21	3360	80640	100.0
	*	25	8	10	42	3360	80640	100.0
		26	6	40	13	3120	74880	93.03
		27	6	13	42	3276	78624	97.5
	*	28	4	40	21	3360	80640	100.0
	*	29	4	20	42	3360	80640	100.0
	*	30	2	40	42	3360	80640	100.0

* : Optimum Packing and Loading Methods, — : Violencing the Constraint

Table 5.3 (c)

k	j	y ₁	y ₂	y ₃	Πy_i	Total Volume	Efficiency (%)
3	—	16	20	—	—	—	—
3	2	16	10	1	160	46080	57.14
3	3	16	6	2	192	55296	68.57
3	4	16	5	3	240	69120	85.71
3	5	16	3	5	240	69120	85.71
3	6	16	2	7	224	64511	80.0
3	7	16	1	10	160	46080	57.14
3	—	16	—	—	—	—	—
3	9	8	20	1	160	46080	57.14
3	10	8	10	3	240	69120	85.71
3	11	8	6	5	240	69120	85.71
3	*	12	8	5	280	86040	100.0
3	13	8	3	10	240	69120	85.71
3	14	8	1	21	168	48384	60.0
3	15	5	20	2	200	57600	71.42
3	16	5	10	5	250	72000	89.28
3	17	5	5	10	250	72000	89.28
3	18	5	2	21	210	60480	75.0
3	19	4	20	3	250	69120	85.71
3	*	20	4	10	280	86040	100.0
3	21	4	6	10	240	69120	85.71
3	22	4	3	21	251	72288	89.64
3	23	2	20	5	200	57600	71.42
3	24	2	10	10	200	57600	71.42
3	25	2	5	21	210	60480	75.0
3	*	26	2	20	280	86040	100.0
3	27	2	6	21	251	72288	89.64
3	28	1	20	10	200	57600	71.42
3	29	1	10	21	210	60480	75.0
3	—	—	—	—	—	—	—

Table 5.3 (b)

k	j	y ₁	y ₂	y ₃	Πy_i	Total Volume	Efficiency (%)
2	—	24	13	—	—	—	—
2	2	24	6	1	144	41472	51.14
2	3	24	4	2	192	55296	68.57
2	4	24	3	3	216	62208	77.14
2	5	24	2	3	240	69120	80.64
2	6	24	1	7	168	48384	60.0
2	7	24	1	10	240	69120	85.71
2	—	24	—	—	—	—	—
2	9	12	13	1	156	44928	55.71
2	10	12	6	3	216	62208	71.14
2	11	12	4	5	240	69120	85.71
2	12	12	3	7	252	72576	90.0
2	13	12	2	10	240	69120	85.71
2	14	12	1	21	252	72576	90.0
2	15	8	13	2	208	59904	74.28
2	16	8	6	5	250	69120	85.71
2	17	8	3	10	240	69120	85.71
2	18	8	1	21	168	48384	60.0
2	19	6	13	3	234	67392	83.57
2	20	6	6	7	252	72576	90.0
2	21	6	4	10	240	69120	85.71
2	22	6	2	21	252	72576	90.0
2	23	4	13	5	260	74880	92.85
2	24	4	6	10	240	69120	85.71
2	25	4	3	21	252	72576	90.0
2	*	26	3	13	273	78624	97.5
2	27	3	4	21	252	72576	90.0
2	28	2	13	10	260	74880	92.85
2	29	2	6	21	252	72576	90.0
2	*	30	1	13	273	78624	97.5

Table 5.4 (b)

k	j	y ₁	y ₂	y ₃	Π y _i	Total Volume	Efficiency (%)
2	1	10	4	—	—	—	—
2	2	10	2	—	—	—	—
2	3	10	1	—	—	—	—
2	4	10	—	—	—	—	—
2	5	10	—	—	—	—	—
2	6	10	—	—	—	—	—
2	7	10	—	—	—	—	—
2	8	10	—	—	—	—	—
2	9	5	4	—	—	—	—
2	* 10	5	2	1	10	66690.0	82.70
2	11	5	1	1	5	33345.0	41.35
2	* 12	5	1	2	10	66690.0	82.70
2	13	5	—	—	—	—	—
2	14	5	—	—	—	—	—
2	15	3	4	—	—	—	—
2	16	3	2	1	6	40014.0	49.62
2	17	3	1	3	9	60021.0	74.43
2	18	3	—	—	—	—	—
2	19	2	4	1	8	53352.0	66.16
2	20	2	2	2	8	53352.0	66.16
2	21	2	1	4	8	53352.0	66.16
2	22	2	—	—	—	—	—
2	23	1	4	1	4	26676.0	33.08
2	24	1	2	4	8	53352.0	66.16
2	25	1	1	6	6	53352.0	66.16
2	26	1	4	2	8	53352.0	66.16
2	27	1	1	6	6	40014.0	49.62
2	28	—	—	—	—	—	—
2	29	—	—	—	—	—	—
2	30	—	—	—	—	—	—

Table 5.4 (a) Result of experiment 3

k	j	y ₁	y ₂	y ₃	Π y _i	Total Volume	Efficiency (%)
1	1	10	6	—	—	—	—
1	2	10	3	—	—	—	—
1	3	10	2	—	—	—	—
1	4	10	1	—	—	—	—
1	* 5	10	1	1	10	66690.0	82.70
1	6	10	—	—	—	—	—
1	7	10	—	—	—	—	—
1	8	10	—	—	—	—	—
1	9	5	6	—	—	—	—
1	10	5	3	—	—	—	—
1	* 11	5	2	1	5	66690.0	82.70
1	12	5	1	1	5	33345.0	41.35
1	* 13	5	1	2	10	66690.0	82.70
1	14	5	—	—	—	—	—
1	15	3	6	—	—	—	—
1	16	3	3	1	9	60021.0	74.43
1	17	3	1	2	6	40014.0	49.62
1	18	3	—	—	—	—	—
1	19	2	6	—	—	—	—
1	20	2	3	1	6	40014.0	49.62
1	21	2	2	2	8	53352.0	66.16
1	22	2	1	4	8	53352.0	66.16
1	23	1	6	1	6	40014.0	49.62
1	24	1	3	2	6	40014.0	49.62
1	25	1	1	4	4	26676.0	33.08
1	26	1	6	1	6	40014.0	49.62
1	27	1	2	4	8	53352.0	66.16
1	28	—	—	—	—	—	—
1	29	—	—	—	—	—	—
1	30	—	—	—	—	—	—

Table 5.4 (d)

k	j	y ₁	y ₂	y ₃	Π y _i	Total Volume	Efficiency (%)
4	1	5	6	—	—	—	—
4	2	5	2	—	—	—	—
4	3	5	2	1	10	66690.0	82.70
4	4	5	1	1	5	33385.0	41.35
4	5	5	1	2	10	66690.0	82.70
4	6	5	—	—	—	—	—
4	7	5	—	—	—	—	—
4	8	5	—	—	—	—	—
4	9	2	6	—	—	—	—
4	10	2	2	1	4	26676.0	33.08
4	11	2	2	2	8	53352.0	66.16
4	12	2	1	2	4	26676.0	33.08
4	13	2	1	4	8	53352.0	66.16
4	14	2	—	—	—	—	—
4	15	1	6	1	6	40014.0	49.62
4	16	1	2	2	4	26676.0	33.08
4	17	1	1	4	4	26676.0	33.08
4	18	1	—	—	—	—	—
4	19	1	6	1	6	40014.0	49.62
4	20	1	2	2	4	26676.0	33.08
4	21	1	2	4	8	53352.0	66.16
4	22	1	1	9	9	60021.0	74.43
4	23	—	—	—	—	—	—
4	24	—	—	—	—	—	—
4	25	—	—	—	—	—	—
4	26	—	—	—	—	—	—
4	27	—	—	—	—	—	—
4	28	—	—	—	—	—	—
4	29	—	—	—	—	—	—
4	30	—	—	—	—	—	—

Table 5.4 (c)

k	j	y ₁	y ₂	y ₃	Π y _i	Total Volume	Efficiency (%)
3	1	5	9	—	—	—	—
3	2	5	4	—	—	—	—
3	3	5	2	—	—	—	—
3	4	5	2	1	70	66690.0	82.70
3	5	5	1	1	5	33345.0	41.35
3	6	5	1	2	10	66690.0	82.70
3	7	5	—	—	—	—	—
3	8	5	—	—	—	—	—
3	9	2	9	—	—	—	—
3	10	2	4	1	8	53352.0	66.16
3	11	2	2	1	4	26676.0	33.08
3	12	2	2	2	8	53352.0	66.16
3	13	2	1	3	6	40014.0	49.62
3	14	2	—	—	—	—	—
3	15	1	9	—	—	—	—
3	16	1	4	1	4	26676.0	33.08
3	17	1	2	3	6	40014.0	49.62
3	18	1	1	6	6	40014.0	49.62
3	19	1	9	1	9	60021.0	74.43
3	20	1	4	2	8	53352.0	66.16
3	21	1	2	3	6	40014.0	49.62
3	22	1	1	6	6	40014.0	49.62
3	23	—	—	—	—	—	—
3	24	—	—	—	—	—	—
3	25	—	—	—	—	—	—
3	26	—	—	—	—	—	—
3	27	—	—	—	—	—	—
3	28	—	—	—	—	—	—
3	29	—	—	—	—	—	—
3	30	—	—	—	—	—	—

Table 5.4 (f)

k	j	y ₁	y ₂	y ₃	Πy_i	Total Volume	Efficiency (%)
6	1	7	4	—	—	—	—
6	2	7	2	—	—	—	—
6	3	7	1	1	7	46683.0	59.89
6	4	7	1	1	7	46683.0	59.89
6	5	7	—	—	—	—	—
6	6	7	—	—	—	—	—
6	7	7	—	—	—	—	—
6	8	7	—	—	—	—	—
6	9	3	4	—	—	—	—
6	10	3	2	1	6	40014.0	49.62
6	11	3	1	2	6	40014.0	49.62
6	12	3	1	3	9	60021.0	82.70
6	13	3	—	—	—	—	—
6	14	3	—	—	—	—	—
6	15	2	4	1	8	53352.0	66.16
6	16	2	2	2	8	53352.0	66.16
6	17	2	1	4	8	53352.0	66.16
6	18	2	—	—	—	—	—
6	19	1	4	1	4	26676.0	33.08
6	20	1	2	3	6	40014.0	49.62
6	21	1	1	4	4	26676.0	33.08
6	22	1	—	—	—	—	—
6	23	1	4	2	8	53352.0	66.16
6	24	1	2	4	8	53352.0	66.16
6	25	1	1	9	9	60021.0	82.70
6	26	—	—	—	—	—	—
6	27	—	—	—	—	—	—
6	28	—	—	—	—	—	—
6	29	—	—	—	—	—	—
6	30	—	—	—	—	—	—

Table 5.4 (e)

k	j	y ₁	y ₂	y ₃	Πy_i	Total Volume	Efficiency (%)
5	1	7	9	—	—	—	—
5	2	7	4	—	—	—	—
5	3	7	2	—	—	—	—
5	4	7	2	—	—	—	—
5	5	7	1	1	7	46683.0	57.89
5	6	7	1	1	7	46683.0	57.89
5	7	7	—	—	—	—	—
5	8	7	—	—	—	—	—
5	9	3	9	—	—	—	—
5	10	3	4	—	—	—	—
5	11	3	2	1	6	40014.0	49.62
5	12	3	2	1	6	40014.0	49.62
5	13	3	1	2	6	40014.0	49.62
5	14	3	—	—	—	—	—
5	15	2	9	—	—	—	—
5	16	2	4	1	8	53352.0	66.16
5	17	2	2	2	8	53352.0	66.16
5	18	2	1	4	8	53352.0	66.16
5	19	1	9	—	—	—	—
5	20	1	4	1	4	26676.0	33.08
5	21	1	2	2	4	26676.0	33.08
5	22	1	1	4	4	26676.0	33.08
5	23	1	9	1	9	60021.0	82.70
5	24	1	4	2	8	53352.0	66.16
5	25	1	2	4	8	53352.0	66.16
5	26	—	—	—	—	—	—
5	27	—	—	—	—	—	—
5	28	—	—	—	—	—	—
5	29	—	—	—	—	—	—
5	30	—	—	—	—	—	—

same, but the efficiency is improved from 82.7% to 99.3%.

Experiment 4 We set the pallet dimensions as (96, 80, 84) and the product item box dimensions as random numbers whose average and standard deviations are 5.5 and 1.43, respectively. The number of boxes to be generated by random numbers are three hundreds. Thus, the carton size distribution is examined by using the proposed method as a simulator. The fraction of the carton size to be distributed is five. The results are shown in Fig. 5.5 (a)-(c). To design standard carton sizes, for instance, we can combine the four numbers largest in length, width and height distributed in the graph.

Let us define Z_1 , Z_2 , and Z_3 as sets of the largest four numbers in length, width and height, respectively. From the experimental result, Z_1 , Z_2 and Z_3 become

$$Z_1 = \{20, 25, 15, 10\};$$

$$Z_2 = \{15, 5, 10, 25\},$$

and

$$Z_3 = \{5, 10, 15, 25\}.$$

The standard sizes of a carton box (z_1 , z_2 , z_3) are determined as such combination as

$$(z_1, z_2, z_3) \in Z_1 \times Z_2 \times Z_3.$$

Fig. 5.5 (a) Carton
size distribution
of length

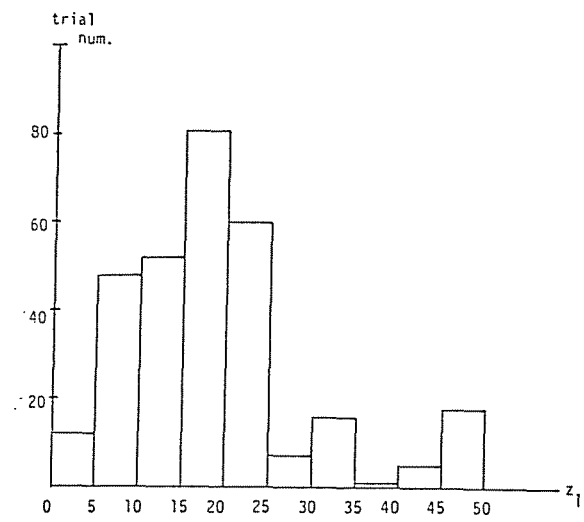


Fig. 5.6 (b) Carton
size distribution
of width

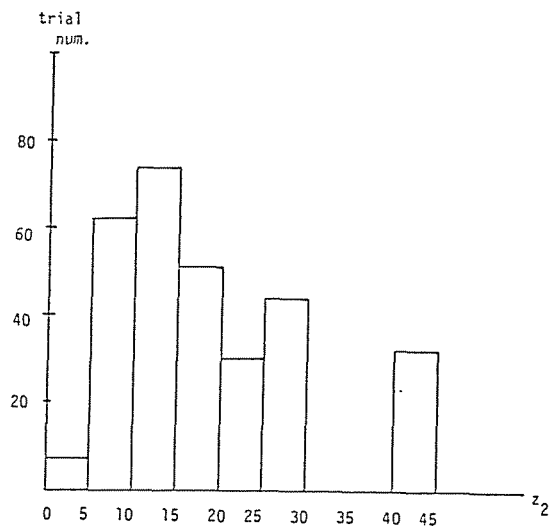
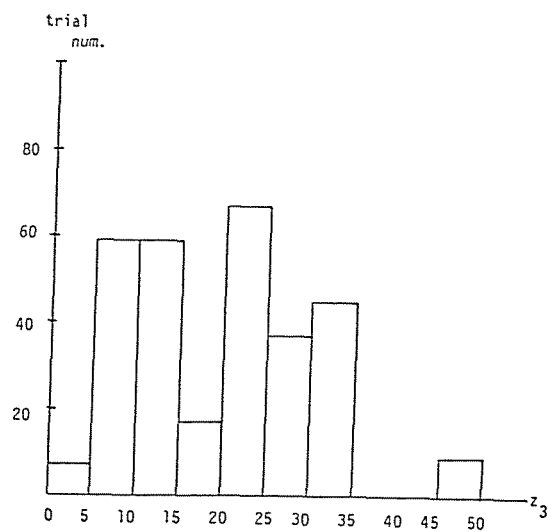


Fig. 5.7 (c) Carton
size distribution
of height



In this way, the standard sizes of a carton box are determined, for instance, their values are
 $(20, 15, 5), (20, 5, 10), (25, 15, 15)$
and so on.

5.7 Additional Criteria

In observing the results, it often happens that several feasible solutions of FS are found out as the optimum solutions. Although there is no analysis in the discussions, we can still solve the problem by using additional objective functions. For instance, we can set a subjective function from the economical viewpoint, to minimize the total surface area of a carton box. The total surface area of a carton box SA is $2(z_1 \cdot z_2 + z_2 \cdot z_3 + z_3 \cdot z_1)$. Hence, the additional criteria is written by

$$\min z_1 \cdot z_2 + z_2 \cdot z_3 + z_3 \cdot z_1 \quad (5.19)$$

With respect to the previous experiments,
Eq. 5. 18 produce the following results:

Experiment 1) $t = 17, (y_1, y_2, y_3) = (48, 10, 7)$
 $(x_1, x_2, x_3) = (3, 4, 2), (z_1, z_2, z_3) = (1, 8, 3)$
Efficiency = 100%

Experiment 2) $k = 1, t = 17, (y_1, y_2, y_3)$
 $= (8, 5, 7), (x_1, x_2, x_3) = (3, 4, 2),$
 $(z_1, z_2, z_3) = (6, 8, 6)$
Efficiency = 100%

$k = 3, t = 12, (y_1, y_2, y_3) = (8, 5, 7),$
 $(x_1, x_2, x_3) = (2, 4, 3), (z_1, z_2, z_3)$
 $= (6, 8, 6)$
Efficiency = 100%

Experiment 3) $k = 6, t = 6, (y_1, y_2, y_3) = (1,$
 $1, 9), (x_1, x_2, x_3) = (6, 4, 1), (z_1, z_2,$
 $z_3) = (39, 38, 4.5)$
Efficiency = 82.7%

5.8 Conclusion

In summary, the following steps are executed:

1. System equations were developed.
2. Qualifications of the problem were analyzed.
3. Problem-oriented algorithms were developed.
4. In order to prove the validity of the developed theory and algorithms, numerical experiments are carried out with acceptable results.
5. A design method for the standard carton box is suggested by using the proposed algorithm as a simulator.

Reference

1. Wilson, R.C.. "A Packing Problem," Man. Sic., Vol.12, No.4, P.B-135, 1965.

6. Minimum Partition of a Compound Rectangular Cell

6.1 Introduction

In a minimum partition criteria, the space allocation problem often occurs. Such a problem is described as "divide the given resource into some materials so as to minimize the number of materials to a possible extent". A typical problem is seen in a computer-aided development in an LSI artwork design. The pattern generator is equipped for the development in an LSI mask manufacturing process and it develops a number of rectangular shapes on a mask film till rectangular shapes are completely buried in the LSI mask shape. It takes a lot of time for the development in proportion to the number of rectangular shapes composing the mask. Therefore, minimum partition is desired to reduce the development time.

Some theorem and a new algorithm are reached for solving the minimum partition problem in this chapter. And to develop the theory, a graph theory is employed here. A shape for the resource and material is described as a graph when the problem is

analyzed and the new method is developed.

The presented method is experimented with a mini-computer and a validity of the presented method is assured.

6.2 Problem Description

As shown in 1.3.4, any shape P in a plane may be described as

$$p = \bigcup_{i=1}^n \left[\bigcap_{j=1}^{m(i)} P_{ij} \right]. \quad (6.1)$$

Setting $P_i = \bigcap_{j=1}^{m(i)} P_{ij}$, Eq. 6.1 is rewritten as

$$P = \bigcup_{i=1}^n P_i. \quad (6.2)$$

Eq. 6.2 implies that there are a number of description ways of the shape P even if we only note an union operator for P 's composition. This aspects is shown in Fig. 6.1.

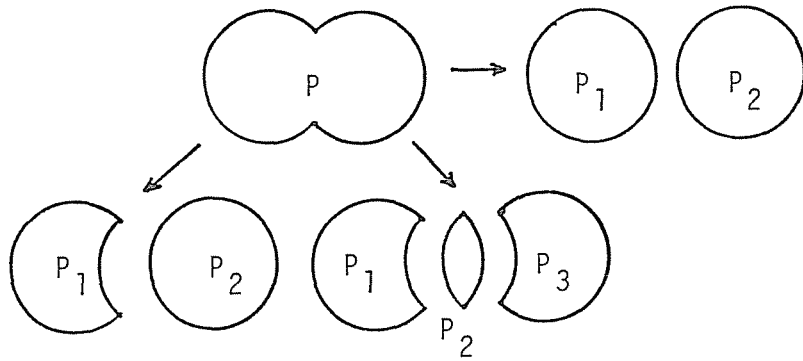


Fig. 6.1 A shape composition by the union operator.

Let us define a term "partition" as the way in which P is also described by

$$P = \bigcup_{i=1}^m R_i, \quad (6.3)$$

with a constraint

$$\bigcap_{i=1}^m R_i = \phi \quad (6.4)$$

A minimum partition problem dealt with in this chapter is

$$\min. \quad m \quad (6.5)$$

subj. to Eq. 6.3 and 6.4.

Then, we restrict the shape of P and the shape of R_i that

1. Edges of P are parallel to a horizontal line (X-axis) or a vertical line (Y-axis).
2. The shape of R_i is rectangular and its edges are parallel to a horizontal line or a vertical line as well.

We name such a P and a R_i a compound rectangular cell, and a uni-rectangular cell, respectively and simply call them a poli-cell and a uni-cell respectively. Then we rewrite the problem to

$$\min. \quad \bigcup_{i=1}^m R_i \quad (6.6)$$

$$\text{subj. to } P = \bigcup_{i=1}^n P_i, \quad (6.7)$$

$$\bigcup_{i=1}^m R_i = P, \quad (6.8)$$

$$\bigcap_{i=1}^m R_i = \phi. \quad (6.9)$$

Thus, the problem is expressed as follows.

For the given poli-cell, divide it into the minimum number of uni-cells which are completely buried on the poli-cell without uni-cell's protruding and overlapping.

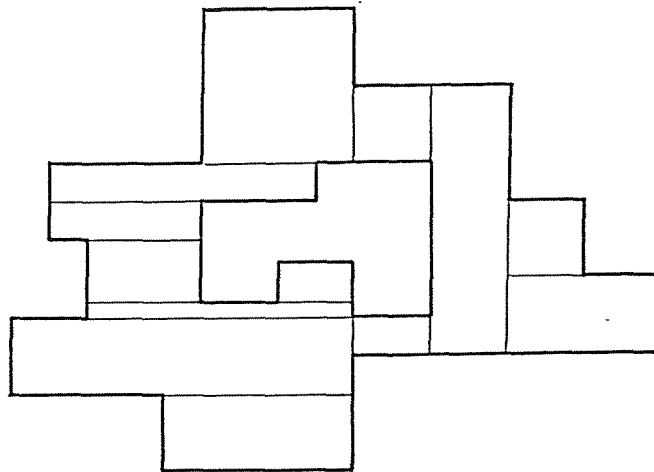


Fig. 6.2 A poli-cell and uni-cells.

6.3 Analysis of Minimum Partition

Though a problem is described by the use of a set theory, it is convenient to a minimum partition analysis to apply a result of a graph theory. When the graph theory is made a use of, a poli-cell P is presented by a set of vertices V and a set of arcs A . In the succeeding discussion, the poli-cell, P is represented by the relation among the vertices in two of which the arc exists or not. As the vertices are figured out by a result of section 7.6, the description method of P by the set theory can be transformed to the one by the graph theory.

6.3.1 Base Vertices for Partition and Their Number

There are vertices from which a poli-cell is parted, if a partition is possible. Let us define these vertices.

Definition: Let a base vertex be defined as a vertex at which corner of P an interior angle is three right angles. A line drawn from the base vertex for a partition of P is named a partition line and a point intersected by the partition line and edges of P are named partition vertices.

Assume that the number of vertices in P is N_0 , the number of the base vertices are obtained by the following theorem.

Theorem 6.1: Set J as the number of the base vertices. Then, the number of the base vertices is given by

$$J = \frac{N_0 - 4}{2} \quad (6.10)$$

Proof: Let us set L as the number of vertices except for the base vertices, then.

$$J + L = N_0 \quad (6.11)$$

The sum of all the interior angles of N_0 -angle

shape is $2(N_o - 2) \cdot \angle R$, and the interior angle at the base vertex is $3 \cdot \angle R$, and the interior angle at the partition vertex is $\angle R$, where $\angle R$ is 90° . Then we obtain

$$3J \cdot \angle R + L \cdot \angle R = 2(N_o - 2) \cdot \angle R. \quad (6.12)$$

From Eqs. 6.11 and 6.12,

$$J = \frac{N_o - 4}{2} \quad L = \frac{N_o + 4}{2} \quad (6.13)$$

Q.E.D.

When P has empty spaces as shown in Fig. 6.2, we define the base vertex as a vertex whose external angle is three right angles. In this case, the number of the base vertices of N_i -angle shape becomes

$$J = \frac{N_i + 4}{2} \quad (6.14)$$

because this case becomes just an inverse of the above one.

In this way, when P has f -empty spaces within itself, the number of the base vertices is obtained in the following theorem.

Theorem 6.2 When P is presented by N_o circumscribe vertices and f empty spaces whose number of vertices are N_i , the number of base vertices J becomes

$$J = \frac{N_o - 4}{2} + \sum_{i=1}^f \frac{N_i + 4}{2} \quad (6.15)$$

If we denote N as all the number of vertices for the presentation of P ,

$$J = \frac{N - 4}{2} + 2f. \quad (6.16)$$

Proof: By the results of the theorem 1, the number of base vertices in the circumscribe vertices is $(N_o - 4)/2$, and the number of base vertices in each empty space is $(N_i + 4)/2$. Therefore, we obtain

$$J = \frac{N_o - 4}{2} + \sum_{i=1}^f \frac{N_i + 4}{2}.$$

Then, since $N = N_o + \sum_{i=1}^f N_i$, this is

substituted into Eq. 6.15 and we obtain

$$J = \frac{N - 4}{2} + 2f. \quad \text{Q.E.D.}$$

6.3.2 Theorem for Minimum Partition

The number of uni-cells which are buried into the given poli-cell P is equal to a cycle rank of the graph. The cycle rank is given by Euler Polyhedron Formula. Therefore, by substituting the number of partition lines, the number of vertices of P , and the relation between edges and the vertices of P , into Euler Polyhedron Formula, we prove basic theorems in order to design a minimum partition algorithm.

Theorem 6.3 Let us define m as the number of uni-cells. Then, there exists a partition such as

$$m \leq \frac{N_0}{2} - 1, \quad (6.17)$$

when P has no empty space, and

$$m \leq \frac{N_0 + \sum_{i=1}^f N_i - 2}{2} + 2f = \frac{N - 2}{2} + 2f \quad (6.18)$$

when P has f empty spaces.

Proof: The number of base vertices J is

$$J = \frac{N_0 - 4}{2} + \sum_{i=1}^f \frac{N_i + 4}{2}$$

As shown in Fig. 6.3, one partition line parallel to y axis may be drawn from each base vertices.

We set H as the number of partition vertices.

Since the partition vertices sometimes overlap,

$$H \leq J. \quad (6.19)$$

If H partition vertices are generated by H partition lines, $2H$ edges are increased to the poli-cell P . After partition, the number of uni-cell, m is equal to the cycle rank in Euler Polyhedron Equation. As we denote k and e as the number of edges and the number of vertices in

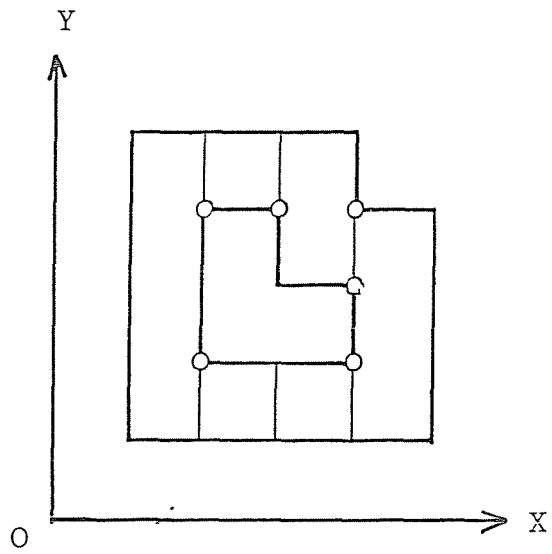


Fig. 6.3 Patitioning along y-axis.

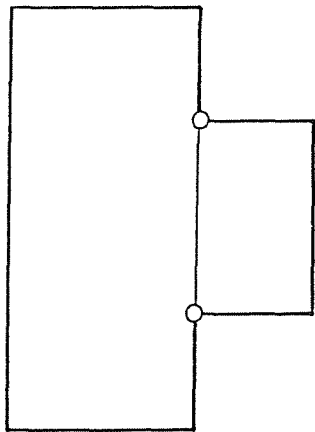


Fig. 6.4 Two base vertices on
the straight partition line.
(Case 1)

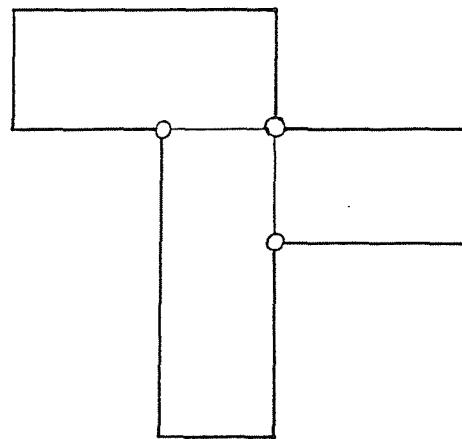


Fig. 6.5 Two base vertices on
the straight patition line.
(Case 2)

poli-cell P after partition respectively

$$m = k - e + 1. \quad (6.20)$$

By partition, k and e becomes

$$k = N_0 + \sum_{i=1}^f N_i + 2H \quad (6.21)$$

$$e = N_0 + \sum_{i=1}^f N_i + H. \quad (6.22)$$

By substituting Eqs. 6.21 and 6.22 into Eq. 6.20, we obtain

$$m = H + 1. \quad (6.23)$$

From Eqs 6.19 and 6.23, we obtain Eq. 6.18.

If empty spaces are not included in P, we obtain Eq. 6.17 by setting $f = 0$. Q.E.D.

Theorem 6.4 Suppose that any two base vertices of the poli-cell P do not exist on the same straight partition line. Then, for any partition, the number of uni-cell m satisfies

$$m \geq \frac{N_0 + \sum_{i=1}^f N_i - 2}{2} + 2f = \frac{N - 2}{2} + 2f \quad (6.24)$$

when empty spaces are included in P, and

$$m \geq \frac{N_0 - 2}{2} \quad (2.25)$$

when empty spaces are not included in P.

Proof: The partition lines must be drawn from the base vertices, and their maximum number from a specified vertex is two and the minimum number is one. Therefore the number of partition vertices H is

$$H \geq J \quad (6.26)$$

As $2H$ edges are increased by the partition, we obtain the same result as Eq. 6. 21 and 6.22 for the partition. From Euler Polyhedron Equation and Eqs. 6.21 and 6.22, we obtain

$$m = H + 1. \quad (6.27)$$

Then, by substituting Eq. 6.26 to 6.27, we obtain Eqs. 6. 24. If empty spaces are not included in P , we obtain Eq. 6.25 by setting $f = 0$.

Q.E.D.

Theorem 6.3 implies that the number of uni-cells which are buried into the poly-cell P becomes $(N - 2)/2 + 2f$ only when any of two base vertices do not exist on the same straight line. And theorem 6.4 implies that the minimum number of the uni-cells is $(N - 2)/2 + 2f$ only when any of two base vertices do not exist on the same straight line.

By the use of theorem 6.3 and 6.4, we obtain the result that only a unique partition line must be drawn from all the vertices in the minimum partition if any of two base vertices do not exist on the same straight

line. This fact leads to the following theorem for the minimum partition.

Theorem 6.5 When a given poly-cell P is partitioned into the minimum number of uni-cells, in which any of two base vertices do not exist on the same straight partition line, a degree of the base vertices is three.

Proof: Before a partition, a base vertex has two edges. After the partition, a partition line must be drawn from the base vertex. Hence, three edges are met with on the base vertex.

Theorem 6.5 gives the minimum partition algorithm a basic hint. Namely, when any of two base vertices do not exist on the same straight lines, the minimum partition is executed by drawing only one partition line from all the base vertices.

Now, we consider the case when two base vertices exist on the same straight line. There are two cases as shown in Fig. 6.4 and 6.5.

At first we treat the case as shown in Fig.

6.4. Suppose that there are $Q/2$ pairs of Q base vertices such as shown in Fig. 6.4, and that partition lines are drawn between pairs of base vertices. After any partition except for the above case, the number of partition vertices are satisfied by

$$H \geq J - Q. \quad (6.28)$$

Since $2H$ edges is increased by the partition in using H partition vertices, the number of all the vertices e , and the number of all the edges after the partition are

$$e = N_0 + \sum_{i=1}^f N_i + H, \quad (6.29)$$

$$k = N_0 + \sum_{i=1}^f N_i + \frac{Q}{2} + 2H. \quad (6.30)$$

Hence, the number of uni-cells m obtained by substituting Eqs. 6. 29 and 6. 30 into Euler Polyhedron Equation is

$$\begin{aligned} m &= k - e + 1 \\ &= H + \frac{Q}{2} + 1 \\ &\geq J - \frac{Q}{2} + 1, \end{aligned} \quad (6.31)$$

By the use of $0 \leq Q \leq J$,

$$m \geq J - \frac{1}{2} J + 1$$

$$\begin{aligned}
&= \frac{J}{2} + 1 \\
&= \frac{N_0 - 4}{4} + \frac{\sum_{i=1}^f (N_i + 4)}{4} + 1 \\
&= \frac{N_0 + \sum_{i=1}^f N_i}{4} + f \\
&= \frac{N}{4} + f.
\end{aligned} \tag{6.32}$$

If there is no interior space empty, Eq. 4. 32 becomes

$$m \geq \frac{N_0}{4}. \tag{6.33}$$

From Eq. 6.31, when $Q/2$ partition lines are drawn between $Q/2$ pairs of base vertices, the minimum number of uni-cells are buried into the poli-cell P.

Therefore, theorem 6.5 is applied to this case.

In the second, we deal with the case as shown in Fig. 6.5. Suppose there are Q base vertices as shown in Fig. 6.5, where Q is a multiple of three. Since the number of partition lines at such base vertices are $2Q/3$, the number of vertices e and the number k of edges after such a partition are

$$e = N_0 + \sum_{i=1}^f N_i + H, \tag{6.34}$$

$$k = N_0 + \sum_{i=1}^f N_i + \frac{2}{3} Q + H. \tag{6.35}$$

From Euler Polyhedron Equation, the number of uni-cells is

$$\begin{aligned}
 m &= k - e + 1 \\
 &= H + \frac{2}{3} Q + 1 \\
 &\geq (J - \frac{1}{3} Q) + 1
 \end{aligned} \tag{6.36}$$

By the use of $0 \leq Q \leq J$,

$$\begin{aligned}
 m &\geq \frac{2}{3} J + 1 \\
 &= \frac{2}{3} \left(\frac{N_0 - 4}{2} + \sum_{i=1}^f \frac{N_i + 4}{2} \right) + 1 \\
 &= \frac{N - 1}{3} + \frac{4}{3} f
 \end{aligned} \tag{6.37}$$

If there is no interior empty space, Eq. 6. 37

becomes

$$m \geq \frac{N_0 + 1}{3}. \tag{6. 38}$$

From Eq. 6 35, when $\frac{2}{3} Q$ partition lines are drawn at the corner of base vertices, the minimum number of uni-cells are buried into the poli-cell P. In this way, theorem 6.5 is also applied to this case.

It is possible to change the partition way shown in Fig. 6.5 into the one in Fig. 6.4. In such a case, the number of uni-cells altered is the same as before. This is proved by the following theorem.

Theorem 6.6 Suppose that there are Q base vertices shown in Fig. 6.5. Then, the number of uni-cells partitioned by the way shown in Fig. 6.5 is the same as the number of uni-cells partitioned by the way in which a partition line is drawn between a pair of base vertices and another partition line is drawn from the rest of the base vertex to a corresponding partition vertex.

Proof: When the partition lines are drawn as shown in Fig. 6.5, the number of uni-cells m after partition is obtained by applying theorem 6.4.

$$m_1 = H + \frac{2}{3} Q + 1.$$

On the other hand, when the latter partition lines are drawn, the number of edges k and the number e of vertices after the partition are respectively

$$k = N_0 + \sum_{i=1}^f N_i + 2 \cdot \frac{1}{3} \cdot Q + \frac{1}{2} \cdot \frac{2}{3} \cdot Q + 2H, \quad (6.39)$$

$$e = N_0 + \sum_{i=1}^f N_i + \frac{1}{3} \cdot Q + H \quad (6.40)$$

Hence, the number of uni-cells m_2 after the partition becomes

$$m_2 = H + \frac{2}{3} Q + 1. \quad (6.41)$$

Thus we gain $m_1 = m_2$. Q.E.D.

Theorem 6.6 implies that the case shown in Fig. 6.5 can be processed in the same way as the case shown in Fig. 6.4. This causes the minimum partition algorithm to make the degree of the base vertex three.

6.4 Minimum Partition Algorithm

A minimum partition is realized by such a way in which partition lines are drawn in order to make the degree of base vertices three. Here, the algorithm for the minimum partition is presented.

6.4.1 A Graph Representation of Poli-cell P

A matrix representation based on a graph theory works to represent a given poli-cell and the partitioned poli-cell. Let us define a incidence matrix A to represent the poli-cell P .

Definition Let us define A as an incidence matrix as

$$A = \{a_{ij}\} \quad a_{ij} = \begin{cases} 1 & \text{when there is an edge} \\ & \text{between vertices } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases}$$

The incidence matrix represents a relation among the vertices and it shows the graph of the poli-cell P.

Poli-cell representation must be established in a computer. We call a poli-cell P before the partition of a primitive poli-cell. The input data for the primitive poli-cell are stored into the computer by the aid of a digitizer. Vertices as the data are input in the counter-clockwise turn of assignments of their number. We denote $x(i)$, $y(i)$ as the coordinates of the vertex i . This first representation has only an information about the circumference vertices relation. We call it a primitive incidence matrix of the poli-cell P, and denote $A = \{a_{ij}^o\}$.

6.4.2 Judgement of Base Vertices

A partition line is always drawn from base vertices. For the partition, the base vertices must be looked for, first. On the base vertex, it occurs a special displacement of the coordinates in a series of vertices. Therefore, the base vertex is searched for as soon as the data of the vertices are input. Judgement conditions of the coordinate displacement

as the base vertices I are as follows:

$$X(I) - X(I - 1) < 0 \text{ and } Y(I + 1) - Y(I) > 0, \quad (6.42)$$

$$X(I) - X(I - 1) > 0 \text{ and } Y(I + 1) - Y(I) < 0, \quad (6.43)$$

$$Y(I) - Y(I - 1) > 0 \text{ and } X(I + 1) - X(I) > 0, \quad (6.44)$$

$$Y(I) - Y(I - 1) < 0 \text{ and } X(I + 1) - X(I) < 0. \quad (6.45)$$

Judgement conditions of 6.42 - 6.45 correspond to the case of Fig. 6.7 (a) - (b), respectively.

6.4.3 Determination of Partition Vertices

After the base vertices are searched for, two partition lines parallel to X and Y axis are drawn from the base vertices. Partition vertices are determined as the intersection points of the drawn partition lines and the edges in the primitive polycell P. The edges are easily found out in the following way.

Let us treat the case when the partition line parallel to x axis is drawn from the base vertex J. The edge on which the partition vertex exist is obtained by finding the edge such as

$$\min. \quad |X(J) - X(I)| \quad I = 1, 2, \dots, n \quad (6.46)$$

$$\text{subj. to } (X(J) - Y(I)) \quad (Y(I + 1) - X(J)) \geq 0 \quad (6.47)$$

$$X(I) - X(I + 1) = 0 \quad (6.48)$$

$$I = 1, 2, \dots, n$$

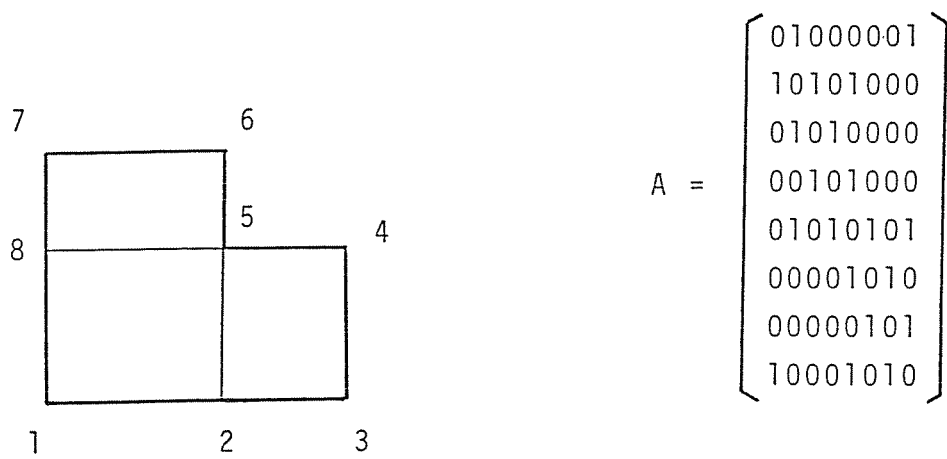


Fig. 6.6 Graph representation and the incidence matrix

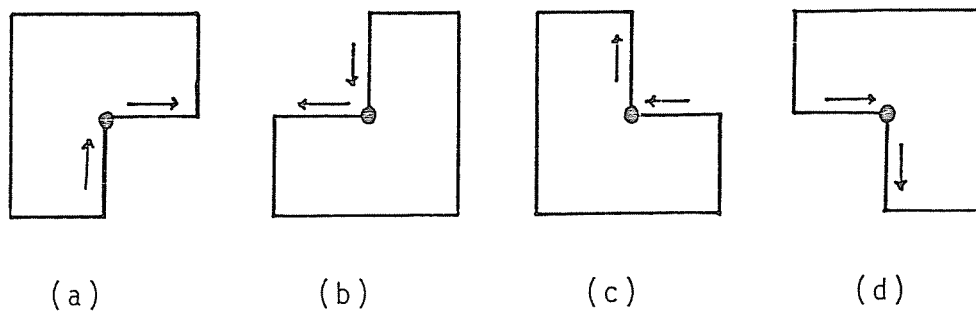


Fig. 6.7 Four types of base vertices

Where n is the number of all the vertices.

The same procedure can be applied when a partition line is drawn parallel to Y axis by changing X to Y, and Y to X. Thus, the partition vertices are registered to the computer.

If a partition point coincides with other base point, only a partition line is registered.

Since all of the partition lines are drawn, we renumber the vertices including partition vertices counter-clockwise and define a new incidence matrix of the poli-cell P. This new matrix has the edges and vertices relation added to the primitive incidence matrix by partition lines and vertices. We call this matrix an basic incidence matrix B, which is

$$B = \{b_{ij}\}, \quad \begin{cases} b_{ij} = 1 & \text{when there is an edge} \\ & \text{between vertices I and J,} \\ b_{ij} = 0 & \text{otherwise.} \end{cases}$$

The basic incidence matrix B has more uni-cells than the ones with minimum partition.

6.4.4 Minimum Partition Algorithm

In order to partition the given poli-cell to the minimum number of uni-cells, a degree of all the base vertices becomes three as shown in theorem 6.5.

If the case shown in Fig. 6.5 occurs, the base vertex is inhibited to have the partition line connected to the partition vertex. From these points of views, a minimum partition algorithm is established as follows:

$$1^{\circ} \quad \text{Set } b_{ii} = \sum_{j=1}^n b_{ij} \quad (i = 1, 2, \dots, n) \text{ in } B.$$

(A diagonal elements of the matrix B has the degree of vertex I.)

$$2^{\circ} \quad \text{Find out the vertex I such as } b_{ii} = 4.$$

$$3^{\circ} \quad \text{Find out the vertex J such as } b_{ij} = b_{ji} = 1 \text{ and } b_{ji} = 3 \text{ subject to } I + 3 < J \text{ or } I - 3 > J.$$

(In this step, one of two partition vertices corresponding to the base vertex I is found.)

$$4^{\circ} \quad \text{Set } b_{ii} = 3, \quad b_{jj} = 2, \text{ and } b_{ij} = b_{ji} = 0$$

(In this step, one partition line is removed and the degree of the base vertex I becomes three.)

$$5^{\circ} \quad \text{Repeat the step } 2^{\circ} - 4^{\circ}. \quad \text{If the vertex I and J which satisfy the condition in step } 3^{\circ} \text{ and } 4^{\circ} \text{ is not found any longer, go to step } 6^{\circ}$$

$$6^{\circ} \quad \text{If there is no vertex such as } b_{ii} = 4, \text{ the minimum partition is reached in the matrix B. If there are vertices such as } b_{ii} = 4, \text{ go to step } 7^{\circ}.$$

7° Find out the vertex I such as $B_{ii} = 4$,
 $b_{jj} = 4$ and $b_{ij} = b_{ji} = 1$ subject to $I + 3$
 $< J$ or $I - 3 > J$.

If such a vertex is not found, stop the
 procedure.

8° Set $b_{ii} = 3$, $b_{jj} = 3$, and $b_{ij} = b_{ji} = 0$,
 then go back to 7

We call this final matrix B a partition
 matrix.

A simple example of the algorithm is shown in
 Fig. 6.8.

6.5 Extraction Algorithm of Uni-Cells

A partition matrix B transformed from a base
 partition matrix gives a graph an incidence relation.
 By using this matrix B, the procedure is requested
 to extract and output each uni-cell being partitioned.
 For the sake of this algorithm, the following algorithm
 is composed.

- 1° Set b_{ij} in B to $b_{ii} = 0$, $b_{i,i-1} = 0$
 $(i = 1, 2, \dots, n)$, $b_{1n} = 0$ and $b_{ij} = -b_{ji}$
 $(i > j, i, j = 1, 2, \dots, n)$.
- 2° Find that $b_{ij} = 1$ ($i, j = 1, 2, \dots, n$). If
 $b_{ij} = 1$ is not found in B, stop the procedure.

$$\begin{pmatrix} 21000001 \\ 13101000 \\ 01210000 \\ 01014101 \\ 00001210 \\ 00000121 \\ 10001013 \end{pmatrix}$$

The basic incidence
matrix.

$$\begin{pmatrix} 21000001 \\ 12100000 \\ 01210000 \\ 00121000 \\ 00013101 \\ 00001210 \\ 00000121 \\ 10001013 \end{pmatrix}$$

$$\begin{pmatrix} 01000000 \\ 00100000 \\ 00010000 \\ 00001000 \\ 00000101 \\ 00000010 \\ 00000001 \\ -10001000 \end{pmatrix}$$

The partition matrix.

Fig. 6.8 An example of the proposed algorithm applied to
the shape shown in Fig. 6.6

- 3° Find that $b_{jk} = 1$ ($k = 1, 2, \dots, n$).
- 4° Find that $b_{jk} \neq 0$ ($j = 1, 2, \dots, n$). If not found, replace $j = k$ and go back to step 3°. If $b_{jk} \neq 0$ is found, go on to step 5°.
- 5° A uni-cell can be extracted as a series of the traced vertex number from the step 2° to step 3°. Then, set that $b_{ij} = 0$, where suffices i and j are the vertex number used in the extracted vertex number, and go back to step 3°.

In this way a series of the vertices corresponding to the uni-cells partitioned from the poli-cell P is obtained. When this algorithm is applied to the matrix shown in Fig. 6.8, the serieses of the vertices become

(1, 2, 3, 4, 5, 8, 1) and (5, 6, 7, 8, 5).

6.6 Experiments

Some of poli-cells are experimented to show a validity of the proposed method for a minimum partition. The computer used is OKITAC-4500C and the program is coded with FORTRAN. Instead of CRT display, an X-Y plotter is equipped for output.

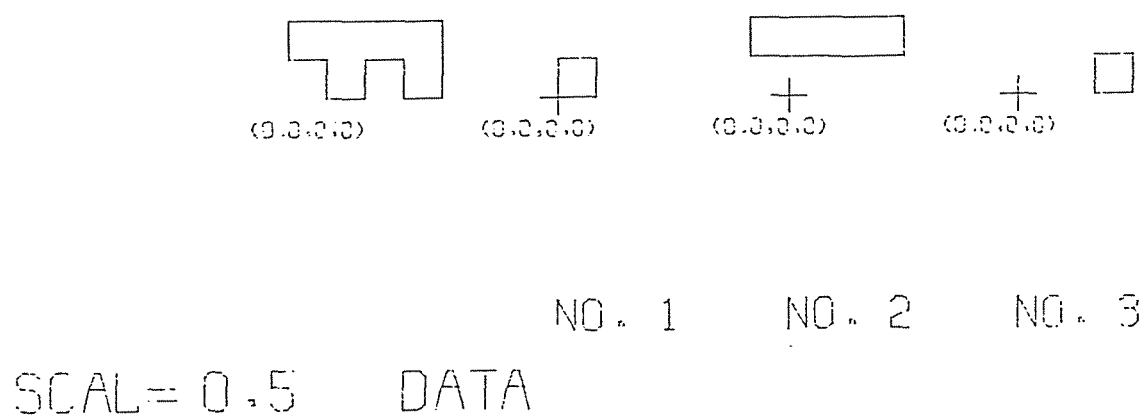


Fig..6.9 Experiment result 1

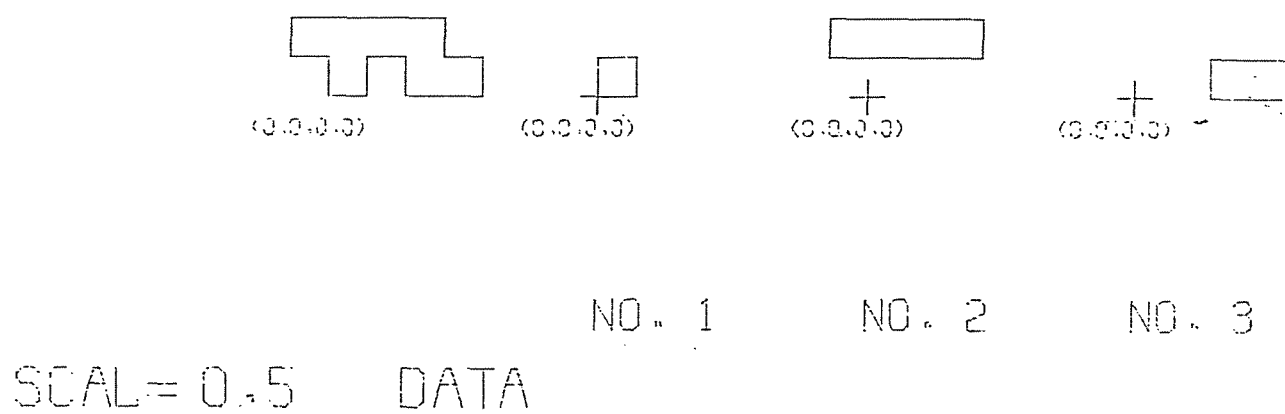


Fig. 1.10 Experiment result 2

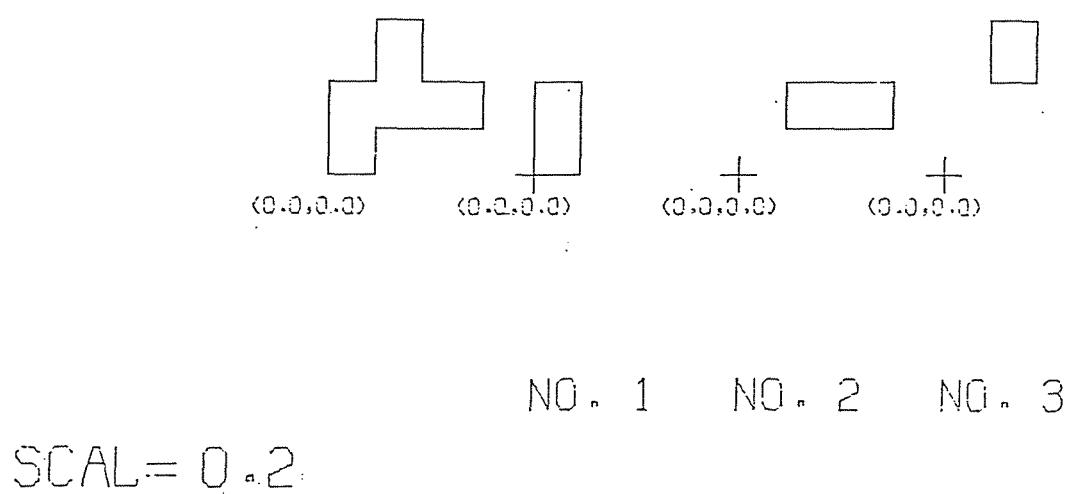
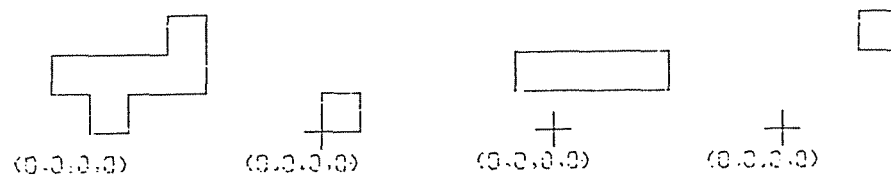


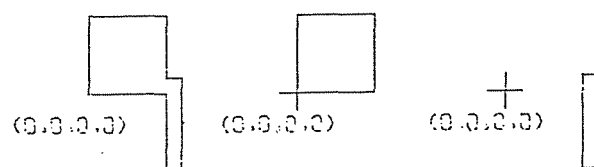
Fig. 6.11 Experiment result 3



NO. 1 NO. 2 NO. 3

SCAL= 0.5 DATA

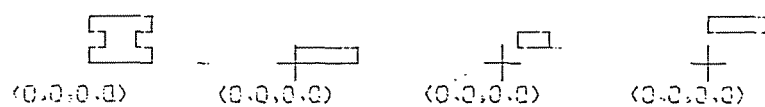
Fig. 6.12 Experiment result 4



NO. 1 NO. 2

SCAL= 0.5 DATA

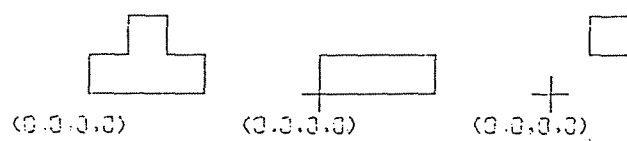
Fig. 6.13 Experiment result 5



NO. 1 NO. 2 NO. 3

SCAL= 0.2

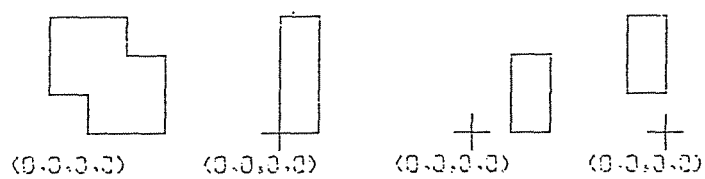
Fig. 6.14 Experiment result 6



NO. 1 NO. 2

SCAL= 0.5 DATA

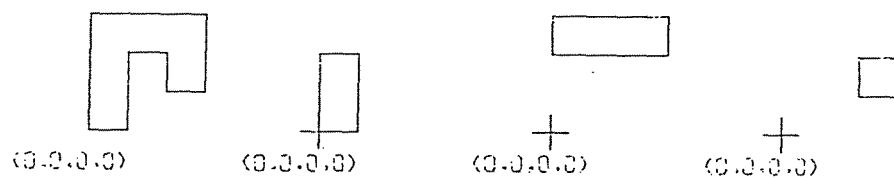
Fig. 15 Experiment result 7



NO. 1 NO. 2 NO. 3

SCAL= 0.5 DATA

Fig. 6.16 Experiment result 8



NO. 1 NO. 2 NO. 3

SCAL= 0.5 DATA

Fig. 6.17 Experiment result 9

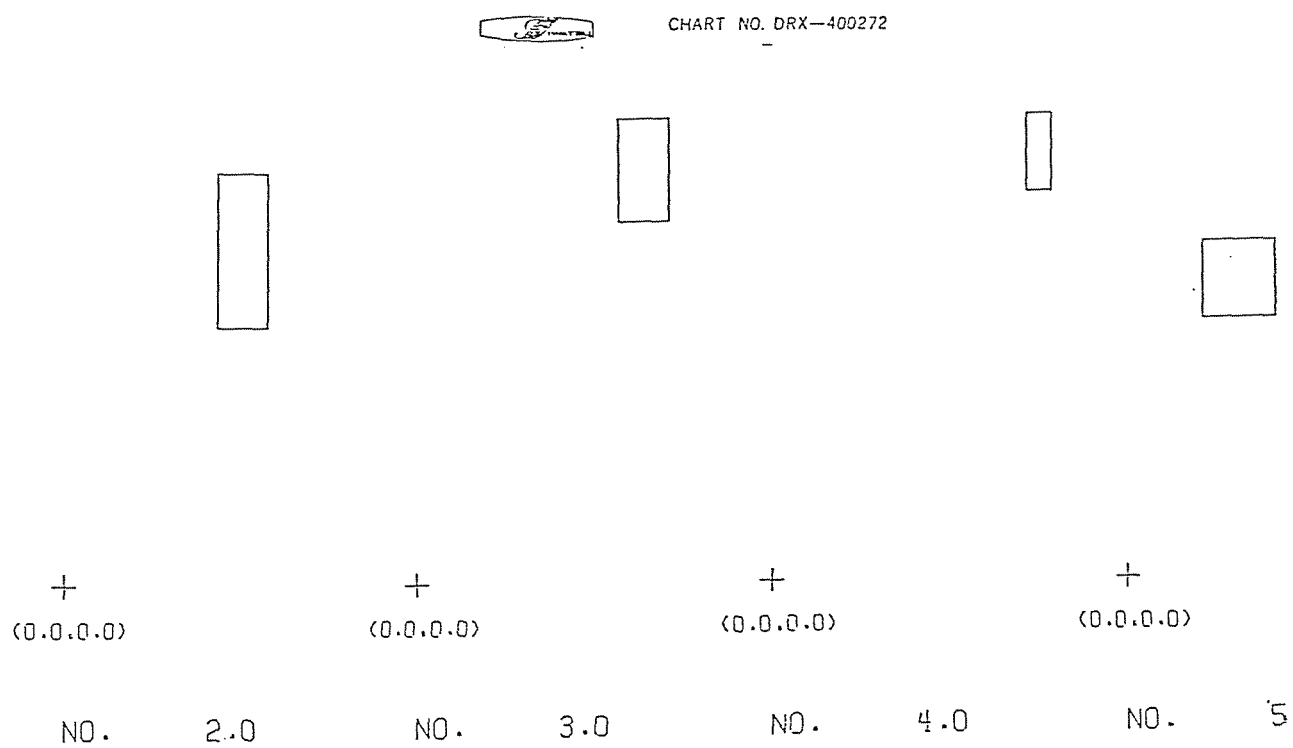
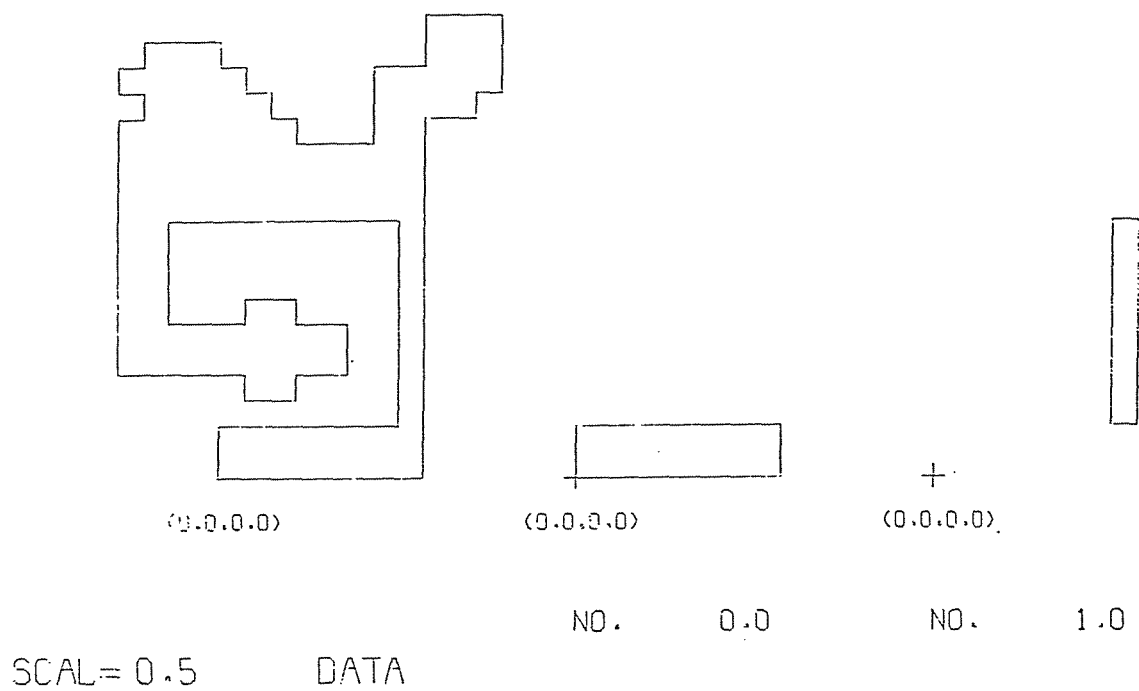


Fig. 18 (a) Experiment result 10

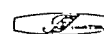
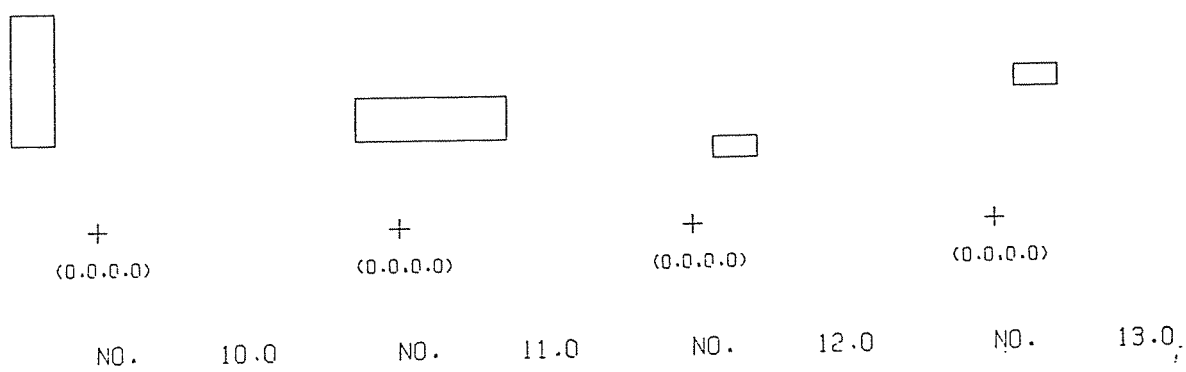
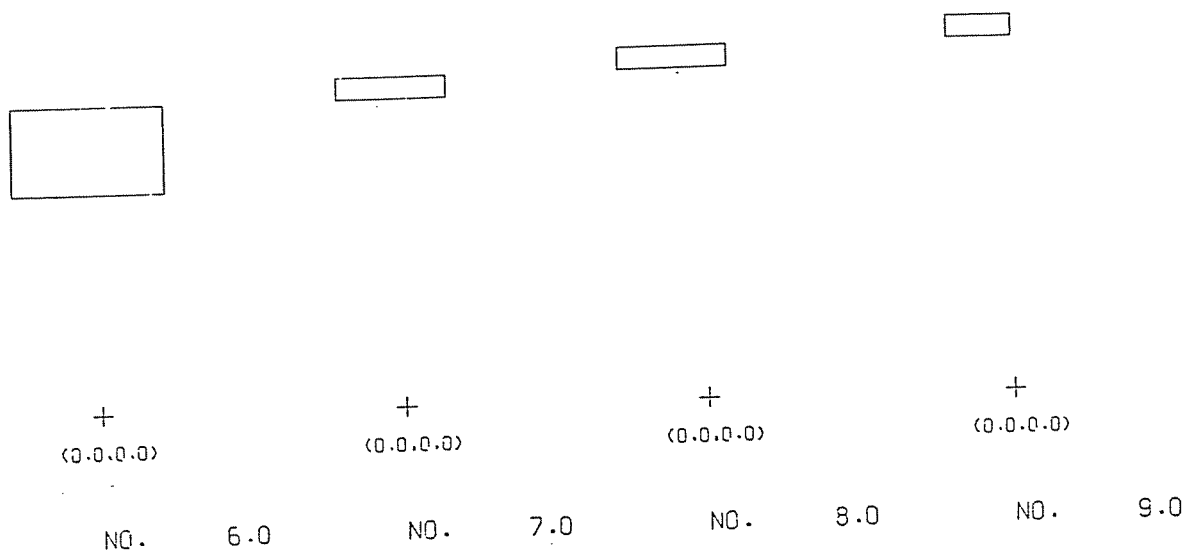


CHART NO. DRX-400272

Fig. 18 (b) Experiment result 10

6.7 Conclusion

An space allocation problem is treated in the situation of a minimum partition. Through a discussion and experiments, we reach the following conclusion.

1. Some theorem for a minimum partition is proved.
2. Conditions for the minimum partition are proposed by the results of the above-mentioned theorems.
3. An algorithm based on the above mentioned 1 and 2 is proposed and an extraction algorithm for uni-cells partitioned from a poli-cell is proposed as well.
4. Experiments to verify the algorithms proposed are done and the validity of the algorithms is proved.

A minimum partition algorithm may be applied not only to LSI art work design but also to other areas. Because this kind of problems are frequently seen in many fields where a graphic processing is required. Such problems are a poligon package problem, an automated process planning, an automated description for 2-D geometries and so on.

Therefore, there are quite a few applications of the minimum partition method.

References

1. Bascker, R.G., "A Graph Theory and Networks"
(in Japanese), Baifukan (1970).
2. Harary, F., "A Graph Theory"(in Japanese), Kyoritsu
Shuppan (1970).
3. Oyamada, T., "A minimum Partition in LSI Art
Work Patterns", IPSJ,(1975),p7.
4. Barton, E.E. and I. Buchanan, "The Poligon Package",
CAD, 12,1 (1980),p3.

7. Graphic Processing in the Space Allocation Problem

7.1 Introduction

The space allocation problem has two phases; the one is to treat the problem as mathematical programming, and the other is to process the problem as graphic processing. Whenever we treat the space, the space geometry can not be separated from the problem, because the geometry of any spaces is essential for the space allocation.

In dealing with the space geometry by a computer, the first problem is to remedy the difficulty of how the geometry is taught to the computer and how the data structure of the geometry is constructed in the computer. The second problem is to process the problem that occurs by the space allocation, for example, the collision problem in the allocation of the space without overlapping and the graphic output to verify the allocation of the space.

As to the first problem, "Formulated Pattern Method (F.P.M)" is developed for the geometric modeling by Prof. N. Okino. Here, we discuss

the space description method in general, and show the relation between the description method and the data structure of the space geometry in computer. As to the second problem, one of the general methods for processing the geometry is established. Namely, the geometry processing method is developed by introducing a boundary evaluator.

In general, the geometric space treatment belongs to the geometric modeling problem, and the results obtained here are one of the applications to the geometric modeling. So the discussion is based on the theory of the geometric modeling.

As the geometries having been treated so far are rectangles or blocks, the results obtained in this chapter is not applicable to the previous problem. However, the theory developed here will be important in processing a free form geometry in future. One of the applications is proposed to solve the collision problem between the shearing blade and the material in metal sheet cutting. This is applied to determine the sequence of shearing out the materials from the nested balnk.

7.2 Geometric Definition and Data Base

The first problem in processing the space geometry is on how the geometric definition is modeled. Here, we introduce the following way of construction to modeling the given space geometry.

A given geometric space model P may be presented by the construction of some basic shapes. Each basic shape is named a primitive and expressed by P_i . P_i is shown in three-dimensional Cartesian coordinate system,

$$P_i = \{x \mid f_i(x) \geq 0\}. \quad (7.1)$$

Eq. 7.1 shows a half space.

Supposing that P is subsequently built up using some constructive operators $\cdot op_i$ and primitives P_i ($i = 1, 2, \dots, n$),

$$P = P_n \cdot op_n \cdot (P_{n-1} \cdot op_{n-1} \cdot (\dots (P_2 \cdot op_1 \cdot P_1) \dots)) \quad (7.2)$$

Let us introduce the set operator as the constructive one. This operation is as follows. When two primitives are operated by a union and a product respectively,

$$\begin{aligned} P_c &= P_a \cup P_b \\ &= \{x \mid f_a(x) \geq 0\} \cup \{x \mid f_b(x) \geq 0\}, \end{aligned} \quad (7.3)$$

and

$$\begin{aligned}
 P_c &= P_a \cap P_b \\
 &= \{x \mid f_a(x) \geq 0\} \cap \{x \mid f_b(x) \geq 0\}, \quad (7.4)
 \end{aligned}$$

Also, the difference operation is defined by using the set operator as follows:

$$\begin{aligned}
 P_c &= P_a - P_b \\
 &= P_a \cap \widetilde{P}_b \\
 &= \{x \mid f_a(x) \geq 0\} \cap \{x \mid -f_b(x) \geq 0\} \quad (7.5)
 \end{aligned}$$

The operations explained in Eqs. 7.3 - 7.5 are shown in Fig. 7.1.

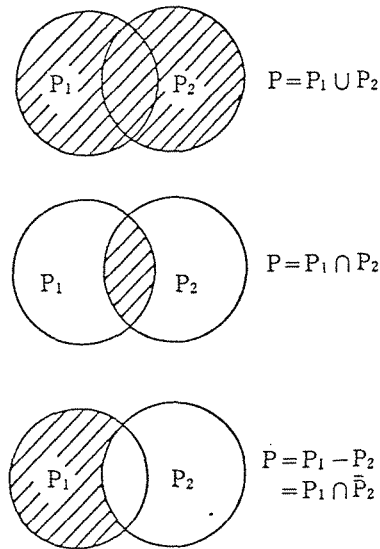


Fig. 7.1 Set operation

Now, let us substitute the set operator in Eq. 7.2, then arrange it as follows:

$$P = (P_k \cap P_l \cap \dots \cap P_m) \cup (P_r \cap P_q \cap \dots \cap P_s) \dots \cup (P_t \cap P_u \cap \dots \cap P_v). \quad (7.6)$$

Suffices in Eq. 7.6 can be exchanged as follows:

$$P = \bigcup_{i=1}^n \bigcap_{j=1}^{m(i)} P_{ij}. \quad (7.7)$$

Eq. 7.7 is offered and named the "Formulated Pattern" by Prof. N. Okino of Hokkaido University. In the process of arranging from Eq. 7.2 to Eq. 7.7, there are various formulations for the given model P. That is, there are a number of possibilities to describe or construct the given geometric space. Therefore, a geometric modeling usually adopts some formulation between Eq. 7.2 and 7.7.

The modeling formulation has an effect on the data base to be structured in the computer corresponding to the geometry. For instance, the modeling method by the use of Eq. 7.2 needs the binary tree as the data base structure, and the modeling method by the use of Eq. 7.7 needs only an indicator, which specifies the primitives having the same suffix i, in the data base.

In general, as the data base has the tree structure corresponding to the geometric modeling method to keep the constructive sequence, it becomes complex to treat it. If we wish a simple data structure for modeling, "Formulated Pattern Method (F.P.M.)" is suitable because its data structure needs no tree. Therefore, F.P.M. is used for the space geometric modeling throughout the discussion without the lack of the generality of modeling.

7.3 Recognition of Space Allocating Feasibility

Once the space geometry is modeled in the computer, the second problem is on how to recognize and extract relations between a given point and the space or two spaces from the geometric model. For this purpose, the boundary evaluation technique is established in the field of geometric modeling. There is room for improvement of the boundary evaluation technique proposed so far, however.

This section first discusses on the boundary evaluation, and a new boundary evaluation technique

is proposed. Then, the recognition method for a space allocating feasibility---the method for solving the collision problem---is taken into consideration.

7.3.1 Boundary Evaluator

Most of outputs of graphic processing can be considered as the mapping of the boundary of space models presented mathematically in the two- or three-dimensional space to some spaces. This mapping procedure is also important for the recognition of the relations between a point and the modeled space, because the recognition of the location of the space is attempted by knowing where the boundary of the space exists. Therefore, the boundary evaluation technique is required and the boundary evaluator serves as the mapping procedure.

The two boundary evaluators are presented so far in geometric modeling: PADL b-function and TIPS-1 penalty function.

They are described as follows.

a) PADL b-function

The b-function adopted by PADL is expressed

as follows:

$$B (P_i \cdot_{op_i} P_{i-1}) = B(\cdot_{op_i}; P_i, P_{i-1}, bP_i, bP_{i-1}), \quad (7.8)$$

where bP_i and bP_{i-1} is the boundary of primitive P_i and P_{i-1} respectively, B is the boolean function whose value is one on the boundary and zero in other areas. When the constructive operator is union one, B operates as

$$B (P_i \cup P_{i-1}) = (bP_i \cap \widetilde{P}_{i-1}) \cup (bP_{i-1} \cap \widetilde{P}_i),$$

where \widetilde{P}_i is negative set of P_i . When the operator is intersection one, B operates as

$$B (P_i \cap P_{i-1}) = (bP_i \cap P_{i-1}) \cup (bP_{i-1} \cap P_i).$$

This operation is performed in the order of the suffix in Eq. 7.2 when modeling the space, so that the data to store the space geometry needs binary tree in order to hold the operation sequence. By the use of PADL boundary evaluator, the graphic processings such as three view drawings, sectional drawing and perspective view drawing become possible because these processings are only mapping of the boundary of the space into two dimensional drawing space. But this evaluator does not give the information on how far there exists the point from the space except that only the point exists on the boundary.

b) TIPS-1 penalty function

TIPS-1 penalty function presented by Dr. Y. Kakazu, Dr. N. Okino and Dr. K. Hoshi is given by

$$S_o(x) = \prod_{i=1}^n \sum_{j=1}^{m(i)} c_{ij} |\min(0, f_{ij}(x))| \quad (7.9)$$

and

$$S_i(x) = \sum_{i=1}^n \prod_{j=1}^{m(i)} c_{ij} |\max(0, f_{ij}(x))|, \quad (7.10)$$

where c_{ij} is a positive number. These equations are based on Eq. 7.7. Eq. 7.9 is used to evaluate the outside of the model and Eq. 7.10 is used to evaluate the inside of the model. For the evaluation, the values of Eqs. 7.9 and 7.10 are figured out. Namely, the value of Eq. 7.9 is zero within the space model and on the boundary, and a positive number which increases toward the outside from the boundary. The value of Eq. 7.10 is the opposite to the above.

The penalty function not only gives the boundary information but it has the outside information of the given model as the potential function. The aspect of this potentiality looks like a declining wall surrounding the given model, and this wall is called a penalty surface. By setting Eqs. 7.9

and 7.10 to

$$S = S_o - S_i, \quad (7.11)$$

and

$$B(S) = \begin{cases} 0 & \text{when } S \neq 0 \\ 1 & \text{when } S = 0, \end{cases} \quad (7.12)$$

the same result as b-function is derived for graphic processing. In addition to this result, the penalty function gives the relation between the specified point and the space model. This information is inducted by measuring the potential value of the penalty function at the specified point. If the value is the function of the distance from the boundary of the space model, it is easy to recognize where the point is. This feature is useful to imply feasible space allocating area. Namely, it becomes possible to know how far the point is from the space model and how distance the point may be moved toward the space model. These are important clues for solving the space location feasibility and collision problem between two spaces.

However, the potential feature is often unclear because Π operation sometimes makes the value of penalty function so large that it becomes impossible to measure the potentiality. To remove this defects, the new evaluator is proposed.

7.3.2 New Evaluator

Defects of Eq. 7.9 against the boundary evaluator are as follows:

- 1° The multiple operation Π corresponding to the union operator makes the value of Eq. 7.9, $So(x)$, so large.
- 2° When the boundary evaluation is needed within the given space, Eq. 7.9 becomes useless. In this case, Eq. 7.10 must be prepared.
- 3° Even if the nearest boundary to the given point is known, all of the function $f_{ij}(x)$ must be operated to figure out Eq. 7.9.

As to 1°, the Π operation produces a steep wall around the model. Especially when some functions have high order terms corresponding to their primitives, the wall becomes extremely steep. In the case of 3°, this sometimes happens and becomes an obstacle to reduce computing time, for the composite basic primitive nearest to the given point is often known or listed up.

Based on Eq. 7.7, a new boundary evaluator is designed by

$$F(x) = - \max_i^n \min_j^{m(i)} c_{ij} f_{ij}(x), \quad (7.13)$$

whose value is negative within the space model, positive outside and zero on the boundary, where c_{ij} is the positive number. The new evaluator overcomes the defects of 1, 2 and 3. As the operations of Eq 7.13 are max and min, the unique function is selected and determines the value of Eq. 7.13. It shows that the displacement of Eq. 7.13 is not so steep as Eqs. 7.9 and 7.10. Furthermore, the composite basic function selected by Eq. 7.13 is usually the nearest primitive to the given point. Therefore, if the nearest primitive is known, the computation of Eq. 7.13 deals only with the nearest basic function. In this way, defects 1 and 3 are overcome by using Eq. 7.13.

The aspects of the boundary evaluators discussed here are shown in Figs. 7.1 - 7.4.

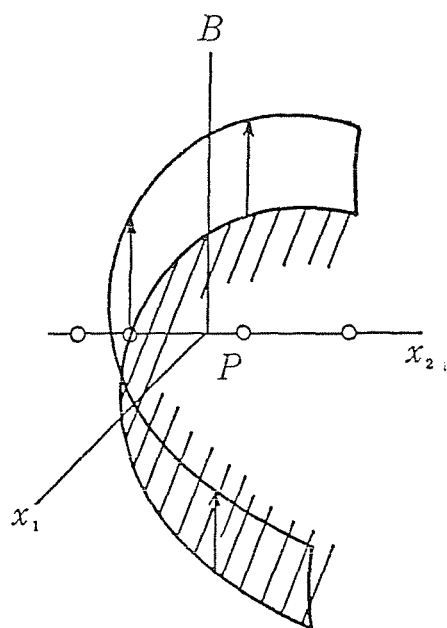


Fig. 7.2 PADL b-function

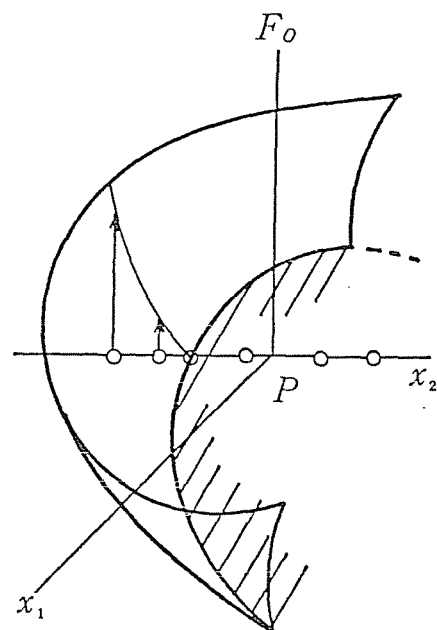


Fig. 7.3 Penalty function
for outside

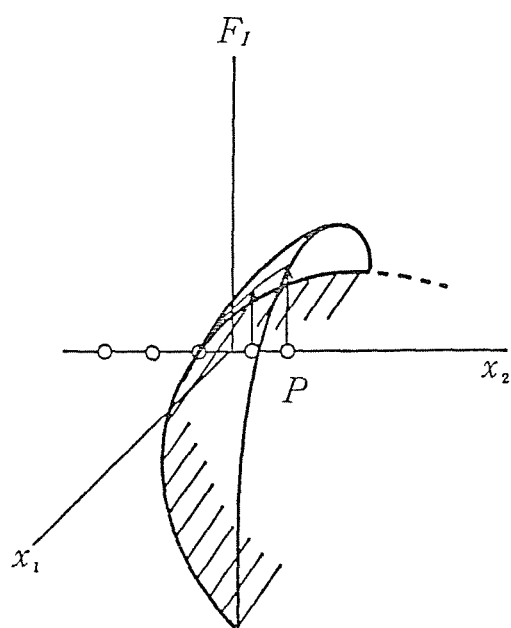


Fig. 7.4 Penalty function
for inside

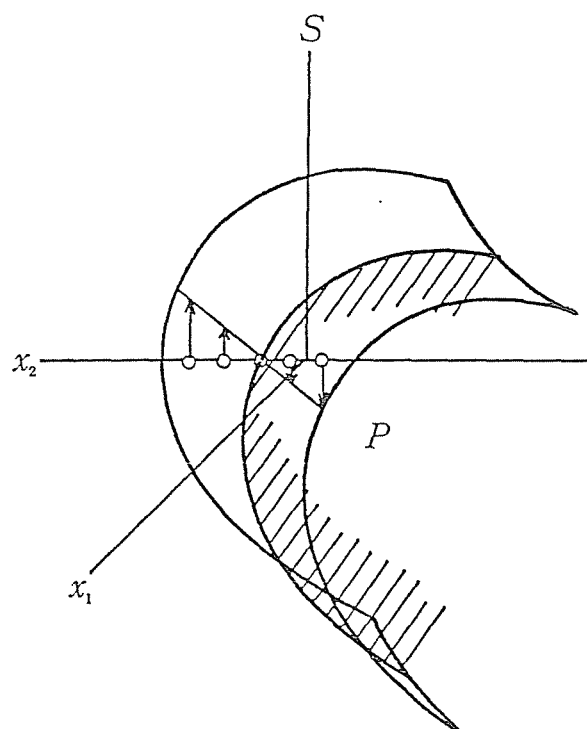


Fig. 7.5 A new evaluator

7.3.3 The Recognition Method for Space Allocating Feasibility

By the use of the new boundary evaluator, the recognition as to whether the given point is outside the given space is easily performed. Let us set x^* to the given point and P_1 to the given space model. Also, the new evaluator

corresponding to P_1 is $F_1(x) = -\max_{i=1}^n \min_{j=1}^{m(i)} c_{ij} f_{ij}(x)$.

Then, we obtain that

$F_1(x^*) > 0$: x^* is outside the model,

$F_1(x^*) = 0$: x^* is on the boundary,

$F_1(x^*) < 0$: x^* is inside the boundary.

This relation shows that the feasible area for allocating another space is restricted by the region which satisfies

$$FA = \{x \mid F_1(x) > 0\}, \quad (7.14)$$

where FA implies the region of the feasible area for allocating another space. If we allocate another space P_2 , P_2 is prohibited from overlapping P_1 .

A point which is inside P_2 such as $y \in P_2$, therefore, satisfies $y \notin FA$. Thus, P_2 is allocated without overlapping P_1 by testing the value of

$F_1(x)$, so that

$F_1(x) \leq 0$: impossible to allocate P_2 ,

$F_1(x) > 0$: possible to allocate P_2 .

And the value of $F_1(x)$ becomes the function of the distance from the boundary of P_1 .

7.3.4 Collision Prohibition

In the previous section, the relation between the two given space locations is considered. Now, we treat n given spaces.

The location constraints on the given n spaces are written by

$$\bigcup_{i=1}^n P_i \subset B, \quad (7.15)$$

$$\bigcap_{i=1}^n P_i = \phi, \quad (7.16)$$

where P_i ($i = 1, 2, \dots, n$) are the given n spaces and B is the resource space in which P_i ($i = 1, 2, \dots, n$) are allocated.

Let us set P_i as

$$P_i = \bigcup_{j=1}^{\ell} \bigcap_{k=1}^{m(j)} \{x \mid f_{jk}^i(x) \geq 0\}, \quad (7.17)$$

$$i = 1, 2, \dots, n,$$

and B as

$$B = \bigcup_{j=1}^h \bigcap_{k=1}^{m(j)} \{x \mid b_{jk}(x) \geq 0\}. \quad (7.18)$$

By applying the boundary evaluator technique to Eqs. 7.15 and 7.16, we gain the following procedure instead,

$$x \in \{x \mid (\max_{i=1}^n (F_i(x)) \leq 0) \wedge (-\max_{j=1}^h \min_{k=1}^{m(j)} (b_{jk}(x)) \leq 0)\} \quad (7.19)$$

and

$$x \notin \{x \mid (\min_{i=1}^n F_i(x)) > 0\} \quad (7.20)$$

where $F_i(x) = -\max_{j=1}^h [\min_{k=1}^{m(j)} c_{jk} f_{jk}^i(x)]$.

In these procedures of testing x , the collision among n spaces is prohibited.

7.3.5 Surface Equations of the Space

It frequently becomes necessary to process the surface of the given space for graphic processing output. By applying the boundary evaluator, the surface equations are simply described. The surface equations include such as plane segments,

line elements and intersection points which are important factors of graphic outputs.

By setting the given space P as

$$P = \bigcup_{i=1}^n \bigcap_{j=1}^{m(i)} \{x \mid f_{ij}(x) \geq 0\},$$

the plane segment equations of the space boundary are expressed by

$$f_{ij}(x) = 0, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, m(i)$$

and

$$- \max_{i=1}^n [\min_{j=1}^{m(i)} f_{ij}(x)] = 0. \quad (7.21)$$

The line segments of the space boundary becomes the intersection of two planes of the space boundary, so that they are expressed by

$$\begin{aligned} f_{ij}(x) &= 0 \\ f_{k\ell}(x) &= 0 \\ i &\neq k, \quad j \neq \ell \quad i, k = 1, 2, \dots, n \\ &\text{and } j, \ell = 1, 2, \dots, m(i) \end{aligned}$$

and

$$- \max_i [\min_j f_{ij}(x)] = 0 \quad (7.22)$$

In the same manner as the above the intersection points of the space boundary are expressed by

$$f_{ij}(x) = 0$$

$$\begin{aligned}
f_{kl}(x) &= 0 \\
f_{uv}(x) &= 0 \\
i &\neq k \neq u, \quad j \neq l \neq v \\
i, k, u &= 1, 2, \dots, n \\
\text{and } j, l, v &= 1, 2, \dots, m(i)
\end{aligned} \tag{7.23}$$

and

$$- \max_i^n \{ \min_j^{m(i)} f_{ij}(x) \} = 0.$$

These equations are the basis of general principle for processing graphic putputs.

7.4 Application of the Collision Prohibition Technique to Material Shear Scheduling

After the materials R_i ($i = 1, 2, \dots, n$) are allocated on the blank (resource) B , they are sheared out with a shear blade. But the determination of the shearing orders for the materials is requested because of the shear blade geometry. In other words, the shear blade must shear out and collide only with the desired material meeting its edges.

It must be avoided to collide with other undesired materials. Thus, the shear order

scheduling is necessary. In the following sections, the shear scheduling method is proposed when the material geometry is restricted to a rectangle and the shear blade geometry an L-shape.

7.4.1 Recognition of Shearing Feasibility

In order to shear out the material, it is necessary for the shear blade to satisfy the conditions that the shear blade shears out and produces the material only desired by the blank and it does not shear the rest of the materials in the blank.

Let us define a region sheared by the blade as $CR(x)$ and the material region desired to be sheared out as $R_{i*}(x)$. Then, the above conditions are expressed by

$$CR(x) \supset R_{i*}(x), \quad (7.24)$$

and

$$CR(x) \cap R_i(x), \quad i \neq i*, \quad i = 1, 2, \dots, n \quad (7.25)$$

Since the blade shape is an L, the blade shearing region with corner coordinates (x_c, y_c) is written by

$$CR(x) = \{(x, y) \mid (x_c - x \geq 0) \cap (y_c - y \geq 0)\}. \quad (7.26)$$

And the region of the material whose shape is a rectangle with its left under corner coordinate (x_i, y_i) is written by

$$R_i(x) = \{(x, y) \mid (x - x_i \geq 0) \cap (y - y_i \geq 0) \cap (x_i + l_i - x \geq 0) \cap (y_i + w_i - y \geq 0)\}, \quad (7.27)$$

where l_i, w_i are the length and width of the material respectively.

By introducing the boundary evaluator to check a given point if it exists inside the shearing region, the following function is established,

$$F_b(x) = - \min (x_c - x, y_c - y). \quad (7.28)$$

If $F_b(x^*) \leq 0$, then x^* is within the region of $CR(x)$.

In the same manner, the following function is established for $R_i(x)$,

$$S_i(x) = - \min (x - x_i, y - y_i, x_i + l - x, y_i + w_i - y). \quad (7.29)$$

If $S_i(x^*) \leq 0$, then x^* is inside the rectangular region of $R_i(x)$.

Supposing that the shear blade shears the material R_i , there exists a point x^* which satisfies

$$F_b(x^*) \leq 0 \quad \text{and} \quad S_i(x^*) \leq 0.$$

From the conditions 7.24, this is rewritten by

$$\begin{aligned} & \max (F_b(x^*), S_i(x^*)) \\ &= - \min (x_c - x^*, y_c - y^*, x^* - x_i, y^* - y_i, \\ & \quad x_i + l_i - x^*, y_i + w_i - y^*) \leq 0. \end{aligned} \quad (7.30)$$

Eq. 7.30 is reduced to

$$\begin{aligned} & - \min (x_c - x^*, y_c - y^*, x^* - x_i, y^* - y_i) \\ & \leq 0. \end{aligned} \quad (7.31)$$

The limit that (x^*, y^*) exists within the region of $CR(x)$ is given by $(x^*, y^*) \leq (x_i, y_i)$.

By substituting (x_i, y_i) to (x^*, y^*) of Eq. 31, we obtain the following relation Eq. 7.32.

$$- \min (x_c - x_i, y_c - y_i) \leq 0 \quad (7.32)$$

When the material R_j is sheared out, the corner point of the blade is met with the upper right corner of the material R_j , Eq. 7. 33 is maintained,

$$- \min (x_j + l_j - x_i, y_j + w_j - y_i) \leq 0, \quad (7.33)$$

where l_j and w_j is the length and the width of the material R_j , and x_j and y_j are left under corner coordinates of material R_j .

Let us set $J(R_j, R_i)$ as

$$\begin{aligned} J(R_j, R_i) = & - \min (x_j + l_j - x_i, \\ & y_j + w_j - y_i). \end{aligned} \quad (3.34)$$

By the above discussion, we can easily test the blade collision by calculating the value of

$J(R_j, R_i)$ to see whether the blade collides with the material R_i when the material R_j is sheared out. If so, $J(R_j, R_i) \leq 0$, and if not, $J(R_j, R_i) > 0$.

7.4.2 Shear Scheduling

When the material R_j is sheared out, the relation that the shear blade for shearing out the material R_j collides with the material R_i or not is generated easily by calculating the value of Eq. 7.34. By testing this relation between all the two materials, we can obtain the binary relation on shearing feasibility between two materials. The binary relation is described by introducing a matrix $P = [P_{ij}]$ which means:

$P_{ij} = 1$: Possible to shear out material R_i without the blade collision with the material R_j .

$P_{ij} = 0$: Impossible to shear out the material R_i because of the blade collision with the material R_j .

By applying Eq. 7.34 so as to determine the value of P_{ij} , the matrix P becomes

$$\begin{aligned}
P_{ij} &= 1 && \text{if } J(R_j, R_i) > 0, \\
P_{ij} &= 0 && \text{if } J(R_j, R_i) \leq 0.
\end{aligned}$$

The relation presented by P gives an information on the relation between two materials as to whether or not shearing out is possible. But it does not give the information on which of the two, R_i and R_k , should be sheared out first when R_i is not sheared out due to the blade collision with R_j , whereas R_i is not sheared out due to the blade collision with R_k , but R_i is not sheared out without the blade collision with R_k . This aspect is shown in Fig. 7.5. In this case, a shear sequence becomes the order of R_k , R_j and R_i . To make such a relation, the binary relation matrix P satisfies a transitive relation of shear order as mentioned above. The following calculation produces the transitive relation,

$$T = P + P^2 + \dots + P^n, \quad (7.36)$$

where $T = [t_{i,j}]$ and the operation is boolean one. The relation derived from T belongs to a weak order so that it becomes possible to determine the shear sequence by the use of the result obtained in chapter 2.

The procedure for determining shear scheduling is as follows:

$$1^{\circ} \quad \text{Calculate } s_j = \sum_j^n t_{ij}. \quad (i = 1, 2, \dots, n).$$

$$2^{\circ} \quad \text{Calculate } v_j = \sum_j^n t_{ij} s_i + s_j. \\ (i = 1, 2, \dots, n).$$

3° Make an order of v_j from the small value of v_j to the larger value of v_j in turn.

The sequence of suffices v_j arranged above becomes a shear scheduling.

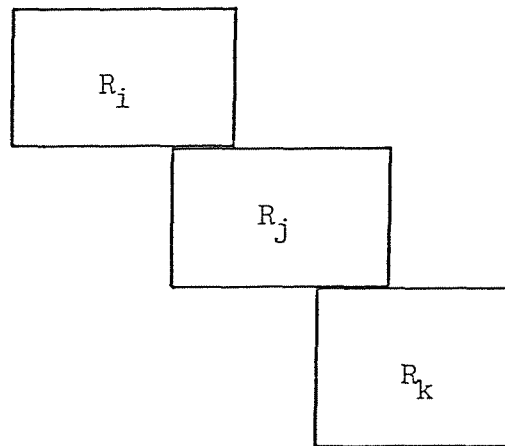


Fig. 7.5 A necessity of a transitive relation

7.5 Conclusion

The discussion has reached the following conclusion.

1. A general constructive method for three-dimensional space is presented, and its relationship to the data base of the space geometry is described.
2. A new boundary evaluator is proposed, which remedies the defects of two other evaluators.
3. The surface of the given space is simply expressed by the use of the new evaluator.
4. The collision prohibition technique among n spaces is proposed.
5. The shear scheduling method is proposed by applying the collision prohibition technique to the shear scheduling problem.

The appendix C shows the experiment results to compare the aspects of boundary evaluators with others.

References

1. Voelcher, H.B., "Discrete Part Manufacturing: Theory and Practice," PADL TR-1-1 and TM22.
2. Okino, N., "Formulated Pattern for Computer-Aided Design and Computer-Aided Manufacturing," Bulletin of Prec. Engg., Vol.5, No.4, 1971.
3. Laning, J.H., "SHAPES User's Manual", Charles Stark Draper Lab., 1973.
4. Braid, I.C. and C.A. Lang, "Computer-Aided Design of Mechanical Component with Volume Building Blocks," Computer Laboratory, University of Cambridge, September, 1972.
5. Oyake, I., "Part Representation CAD/CAM," Proc. of AFIPS NCC, Vol.44, May, 1975.
6. Hosaka, M., et. al., "GIL A Language for Interactive Graphics and Its Applications," Proc. of Second U.S.A.- JAPAN Computer Conference, 1975.
7. Okino, N., et. al., "TIPS-1 Technical Information Processing System for Computer-Aided Design, Drawings and Manufacturing," Proc. of the second IEIP/IFAC PROCAMAT Conf., 1973.

8. The Development of CAM Software System for Punching-press and Shearing

8.1 Introduction

The realization of shearing process as computer-aided manufacturing has been difficult since an NC shearing machine appeared. It comes from the difficulty of making an automated plan for nesting, where nesting means the allocation of materials to the given blank. If this problem is broken up, it becomes possible to develop an integrated software system for punching-press and shearing.

In this chapter, the software system is presented for computer-aided punching-press and shearing by applying the method proposed in chapter 4 to shear process planning and by developing the determination of punching-press tool path. The developed system is named CAMPS (Computer-Aided Manufacturing for Punching-press and Shearing).

8.2 System Design

The highest mountain against the automations of punching-press and shearing is to automate shear process planning. If mountain is climbed, it becomes possible to develop an integrated punching-press and shearing of the software system. It means that the allocation of the materials to the stocked blanks, the determination of absolute coordinate points on the blanks for punching-press, the determination of the punching tool path, and shear scheduling for the materials are automatically executed. CAMPS system is designed and developed to process all of these. The followings are the specifications of CAMPS for the system design and the functions of the processors constructing CAMPS.

8.2.1 System Specification

The software system developed here is based on the use of the following hardware equipments:

NC turret punching-press machine for punching-press
NC shearing machine with a L-shape blade for shearing

The use of an NC shearing machine releases the

restriction of the allocation method, which usually occurs by shearing in such a way as guillotine cut.

We give the system two conditions before the design, assuming to utilize above hardware equipments.

Condition 1 the system available for mini-computer

Condition 2 realization of high automation

Condition 1 is founded to utilize mini-computers which have been implemented at a lot of manufacturing factories. Condition 2 is founded to reduce the processing time. If the system adopts the interactive type, a lot of time is taken for human judgement and response. This relays the processing time. Therefore, the system does not adopt the interactive type in order to satisfy condition 2.

The system specifications under two conditions are set up. They are as follows:

1. Information on the materials, blanks, and punching-press tool is input in a simple language.
2. The input language is translated to a canonical data format and is stored into files.
3. An optimum allocation of the materials onto the blanks is automatically calculated.

4. After the optimum allocation, tasks punch-pressed by the same tools for the allocated blanks are sorted. Then, a tool path is figured out.
5. The shearing sequence of the materials allocated onto the blank is scheduled and positioning of a shear blade is determined in accordance with the shearing sequence.
6. The output of the allocation and the tool path are verified by the use of a CRT display.

In answering to the specifications, the input language is designed and five processors are developed for CAMPS system. Five processors are the input translating processor SCANNER, the allocating processor OPTNST, the task sorting processor TSKCLS, the punch-press tool path generating processor OPTPTH, and the shear scheduling processor SHEARS. The CAMPS system structure is shown in Fig. 8.1. For the sake of specification of 6, graphic output is displayed to the CRT graphic display equipment. The output is the drawings of the allocation results and the trace of punching-press tool path. The repositioning problem is taken into consideration in the system, but it is not mentioned here. Such a problem may be done with post-processing.

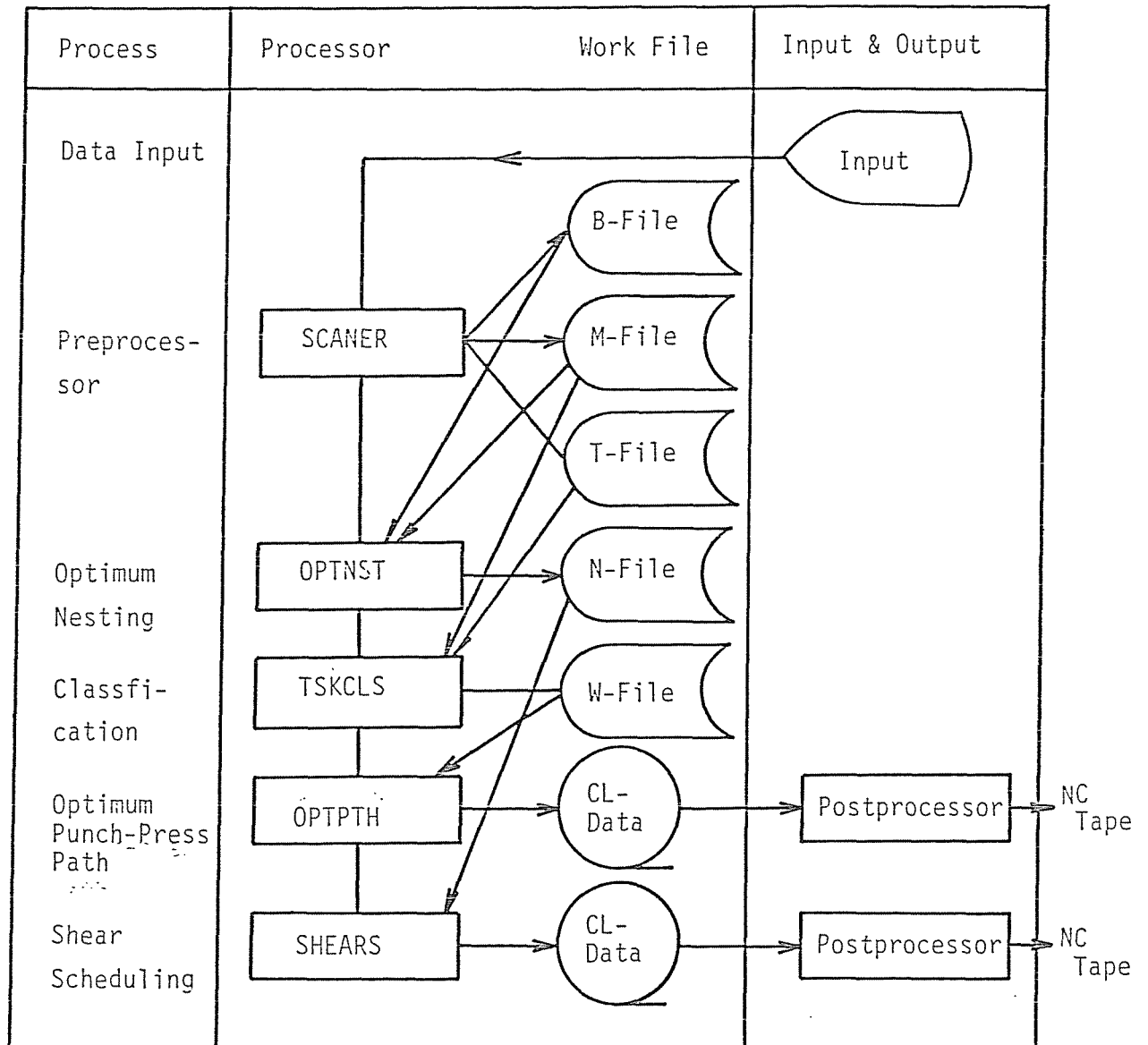


Fig. 8.1 CAMPS system structure

8.2.2 Input Information and Language

Input information to the system consists of the following three: the materials, the blanks and the punching-press tasks. The information is fed into the system in the order as shown in Fig. 8.2. Each information block is discriminated by setting the the discrimination statements to the end of each one. Each information has the following contents.

Blank information : the size and the number
of stocked blank

Material information: the size, the punching-
press geometry (APT-like
language) and the number
of the material requested
by the user

Task information the punching-press tool
assignment to the geometry
and the tool priority

The designed language is shown in Fig. 8.3.

8.2.3 Input Translating Processor SCANNER and Canonical Data File

The input translating processor SCANNER reads the input language and translates it to the canonical data which is manufactured by the following processors. The canonical data is classified to three kinds relating to the materials, the blanks and the tasks. The classified data is stored into three files: B-file, M-file and T-file. Each data file is shown in Fig. 8.4. M-file consists of two arrays, the one having the size of materials and the other having the punching-press-geometry. The pointers are used to connect the geometries to the materials. This is shown in Fig. 8.4 (b). The array for the geometry is one-dimensional and each geometry data is stored in the form shown in Fig. 8.5 into this array.

F-file has two arrays, the one for tool information, and the other for punching-press geometry assigned to the tool. The pointers are used to connect the tool and the geometry punch-pressed by the specified tools. This is shown in Fig. 8.4 (c).

<pre> *BLANK B1=B/100,250, ... B2=B/200,120, </pre>	Blank Information
<pre> *BFINI </pre>	
<pre> *MATERIAL M1=M/50,75, ... MOVE/10,10 L1=PTN/LIR,INCR,5, Punching-press Geometry Statements </pre>	Material Information
<pre> *MEND M2=M/40,60, Punching-press Geometry Statements </pre>	
<pre> *MEND *MFINI </pre>	
<pre> *TASK T1=PUNC/1,5/L1, ... T2=CNC/2,AUTO, *TFINI </pre>	Task Information

Fig. 8.2 Input information sequence

```

Blank data start statement
    *BLANK
Blank data statement
    symbol=B/l,w,n,t,c
    l;Length of a given blank
    w;Width of a given blank
    n;A number of blank stocked
    t;Thickness of a given blank
    c;A cost of a given blank
Blank data end statement
    *BFINI

Material data start statement
    *MATERIAL
Geometri data start statement
    symbol=M/l,w,n
    l;Length of a given material
    w;Width of a given material
    n;A number of a given material required
        by user
Geometry data end statement
    *MEND
Material data end statement
    *MFINI
Task data start statement
    *TASK
Task assignment data statement
    symbol=CNC/priority,AUTO,r
    symbol=CNR/priority,AUTO,t1,tw
    symbol=PUNC/priority,r/symb1,symb2,...,
        symn,material symbol/..., .../...
Task data end statement
    *TFINI

```

Fig. 8.3 (a) Input language

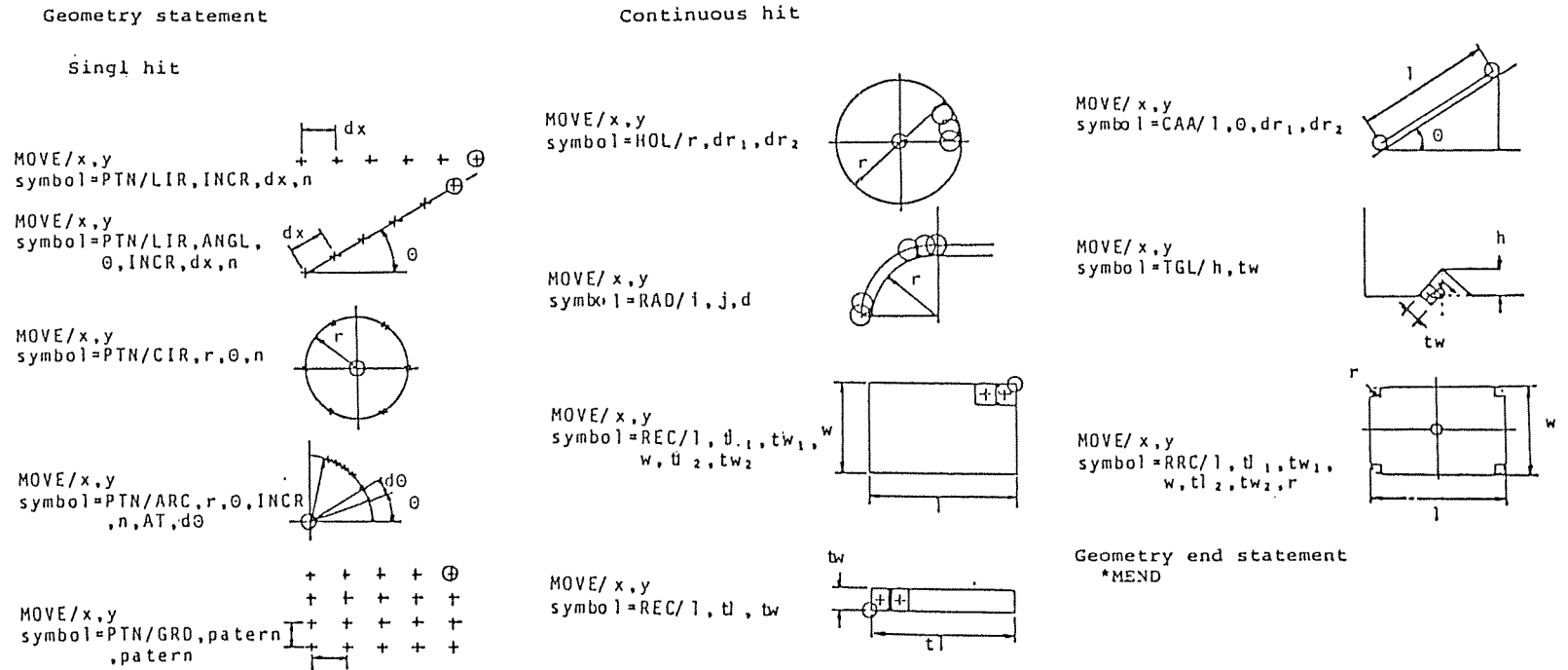


Fig. 8.3 (b) Input language for punching-press geometry

Symbol	Width of blank	Length of blank	A number of blank	Thickness of blank	Cost

Fig. 8.4 (a) B-file standard data format

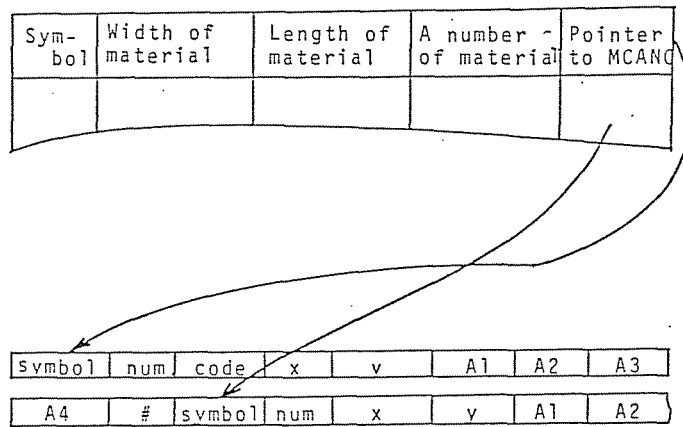


Fig. 8.4 (b) M-file standard data format

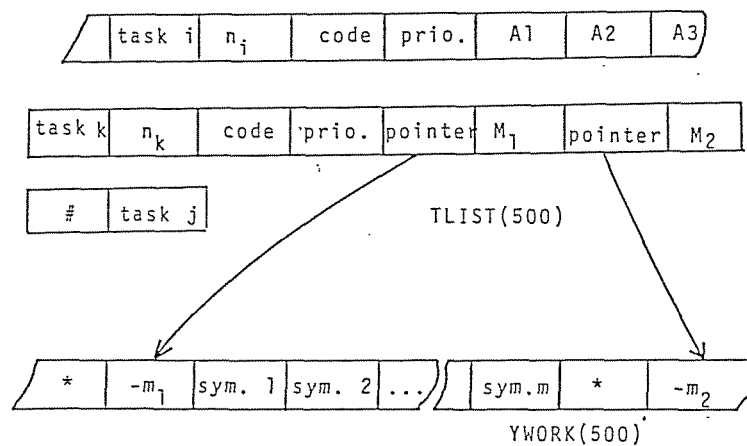


Fig. 8.4 (c) T-file standard data format

POINT	1	x	y	*					
PTN/LIR	21	θ	dx	n	*				
PTN/CIR	22	r	θ	dx	n	*			
PTN/ARC	23	r	θ	n	d θ	n	*		
PTN/GRD	24	θ	dx	n					
		θ	dx	n	*				
HOL	6	r	t	p	*				
RAD	7	r	$\pm t$	θ	d θ	p	*		
REC	8	$\pm z$	$\pm w$	t_1	p_1	p_2	*		
CAA	9	z	θ	t	p	*			
TGL	10	\vec{c}_i	\hat{h}	t	*				
RRC	11	$\pm z$	$\pm w$	t_1	p_1	t_2	p_2	r	*

Fig. 8.5 Canonical data format for punching-press geometry

8.2.4 The Allocation Processor OPTNST

The processor OPTNST is the routine that solves the following problem.

Problem Given the number and the size of the materials as the product, and the number, the size and the cost of the stocked blanks, assign and allocate the materials to the blanks so that the minimum costs and wastes are resulted, and determine the number of blanks to be consumed.

The problem is mathematically modeled. Now, let us set b_j ($j = 1, 2, \dots, m$) and r_i ($i = 1, 2, \dots, n$) to the number of the blanks B_j ($j = 1, 2, \dots, m$) and the number of the materials M_i ($i = 1, 2, \dots, n$), respectively. Also, let us set a_{ijk} to the number of material M_i which is allocated onto the blank B_j in the k -th allocation manner among all of l possible allocating manners. Then, a_{ijk} is determined to satisfy

$$\max. \quad \sum_i^n \sum_j^m a_{ijk} \quad x_{ijk} \quad (8.1)$$

$$\text{subj. to} \quad \sum_i a_{ijk} \quad S_i \leq A(B_j) \quad (8.2) \\ (j = 1, 2, \dots, m)$$

where S_i is the area of the material M_i , $A(B_k)$ is the area of the blank B_j and x_{ijk} is the number of

materials M_i allocated onto the blank B_j in the k -th manner.

Under Eqs. 8.1 and 8.2, find a_{ijk} and x_{ijk} that satisfy

$$\min. \quad \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^l a_{ijk} x_{ijk} \quad (8.3)$$

$$\text{subj. to } \sum_{j=1}^m \sum_{k=1}^l a_{ijk} x_{ijk} \geq r_i \quad (8.4)$$

$$(i = 1, 2, \dots, n)$$

$$\sum_{i=1}^n \sum_{k=1}^l x_{ijk} \leq b_j \quad (8.5)$$

$$(j = 1, 2, \dots, m).$$

The new method for solving this problem is proposed in chapter 3. CAMPS adopts this new method. The processor input is b_j , r_j , and the size of materials and the blanks are extracted from B-file and M-file. The results are stored into Nest file (N-file).

8.2.5 Task Sorting Processor TSKCLS

Punching-press processing is executed against the blank on which the material allocation is already determined by the preceeding processor. As the positions of the punch-pressed geometry are defined on the material, they are translated into the position on the blank onto which the materials are allocated. The tasks

of punching-press by the use of the same tool and the same priority are sorted as a group in order to reduce the manufacturing time.

The algorithm is developed for this sorting.

Let us define symbols under below:

M_i the i -th material ($i = 1, 2, \dots, m$)

t_{ik} the k -th task worked on M_i ($k = 1, 2, \dots, n$)

$T(t_{i1}, t_{i2}, \dots, t_{in})$ a set of tasks worked on M_i .

g_{ikh} the h -th geometry assigned to the task t_{ik} .

$N_{jl}(M_p, M_q, \dots, M_r)$ a set of materials allocated onto the blank in the l -th manner.

$T(N_{jl})$ a set of tasks worked on N_{jl} .

From the allocation result, we obtain a set of materials as

$$N_{jl} = N_{jl}(M_p, M_q, \dots, M_r). \quad (8.6)$$

Let us set $T(M_i)$ to a set of tasks worked on M_i ,

$$T(M_i) = T(t_{i1}, t_{i2}, \dots, t_{in}) \quad (8.7)$$

T -file gives the task information in the form of Eq.

8.7.

The material set is given by Eq. 8.6, the task set working on N_{jl} is

$$\begin{aligned} T(N_{jl}) &= T(M_p) \cup T(M_q) \cup \dots \cup T(M_r). \\ &= \{(t_{p1}, t_{p2}, \dots, t_{pa}) \cup (t_{q1}, t_{q2}, \dots, t_{qb}) \\ &\quad \dots \cup (t_{r1}, t_{r2}, \dots, t_{ra})\} \quad (8.8) \end{aligned}$$

By giving the same suffix to the same task,
Eq. 8.8 is changed into

$$T(N_{j1}) = \{t_{j1}, t_{j2}, \dots, t_{js}\}. \quad (8.9)$$

Eq. 8.9 gives N_{j1} all tasks that are to work.

Then we search for the material set in turn of

t_{ju} ($u = 1, 2, \dots, s$) such as

$$\begin{aligned} W(t_{ju}/N_{j1}) &= \{M_x, M_y, \dots, M_z\} \\ \text{subject to } t_{ji} &= T(M_x) \cup T(M_y) \cup \dots \\ &\quad \cup T(M_z). \end{aligned}$$

Then, we list up the geometries corresponding to task t_{ju} and determines the absolute position of geometries in N_{j1} . In this way, geometry positions are figured out.

8.2.6 Punching-press Tool Path Determination

Processor OPTPTH

Tasks which have a common priority and are assigned to use common tools are sorted to the same group by the previous processor TSKCLS. Now, a punching-press tool path, which is positioned on all the geometries defined by the tasks sorted to the same group, must be calculated.

Let us set (a_i, b_i) ($i = 1, 2, \dots, n$) to a certain positioning point. The positioning point stands for

the geometry punch-pressed in the sorted group. We regard a continuous punch-pressed geometry as a point. Then, the determination problem in which the tool travels and presses out all the points within the least time is to solve the following mathematical programming model:

$$\begin{aligned} \min. \quad & \sum_{i=1}^n \sum_{j=1}^n C_{ij} x_{ij}, \\ \text{subj. to} \quad & \sum_{i=1}^n x_{ij} = 1 \quad (j = 1, 2, \dots, n) \\ & \sum_{j=1}^n x_{ij} = 1 \quad (i = 1, 2, \dots, n), \\ & x_{ij}^2 = x_{ij} \quad (i, j = 1, 2, \dots, n), \end{aligned}$$

where C_{ij} is the distance between two points (a_i, b_i) and (a_j, b_j) . The tool path is presented by the solution in $x_{ij} = 1$. If a velocity of the tool moving with x axis is the same as the one with y axis, the distance becomes,

$$C_{ij} = \max (|a_i - a_j|, |b_i - b_j|).$$

CAMPS system adopts this distance.

The above model is so called "Traveling Salesman Problem". Though the branch and bound method is usually employed in solving the problem efficiently, a huge memory and consuming time is needed by this adoption. In accordance with system design condition 1,

CAMPS system adopts a method that brings an approximate solution practical enough within small memory and time. The method employed as this processor OPTPTH is the nearest path method.

The procedure of the nearest path method is as follows.

Let us define $t(n)$ as a set of all the n point suffices and define $t(k)$ as a set of k point suffix whose point is already punch-pressed. Then procedure of solving the problem is

$$f_{k+1} = f_k + \min_{j \in (t(n) \cap \widetilde{t}(k))} c_{kj}$$

$$f_0 = 0, (k = 0, 1, 2, \dots, n-1),$$

where $\widetilde{t}(k)$ is a negative set of $t(k)$. The tool path is seeked in the order of suffix j determined in k step.

8.2.7 Shear Scheduling Processor SHEARS

We assume an NC shearing has a L-shape blade. When the L-shape blade is applied to shear out the materials, a shearing order must be scheduled. Unless it is scheduled, the blade often shear out the undesired material. A case of undesired shearing is shown in Fig. 8.6.

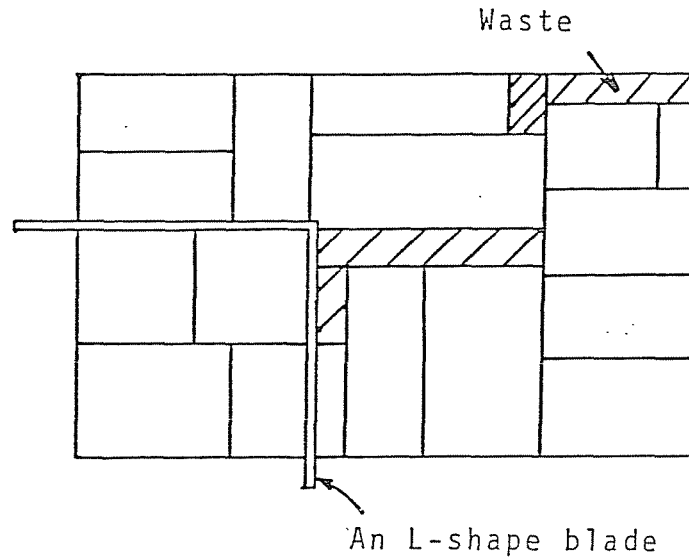


Fig. 8.6 Undesirable shearing with an L-shape blade

In other words, the L-shape blade shears out the material only desired according to the shearing order of the materials and it should not shear out others. The determination of the shearing order becomes such a problem as below:

Make such a schedule that the shear order of the materials satisfies the transitive relation. The transitive relation means the relation that the material R_i is first sheared out among the materials R_i , R_j and R_k when the material R_i is sheared out before the material R_j , and the material R_j is sheared out before R_k . The method is proposed for solving the problem in chapter 7.4.2. The processor SHEARS is the routine loaded by this proposed method.

8.3 Examples

Examples resulted from running CAMPS system are illustrated here. The mini-computer instituted for the system is OKITAC 4500-C. A load module memory size is around 26K words.

Fig. 8.7 shows the material geometries input into the system.

Fig. 8.8 shows the example of input language describing above material geometries, the blank information and the task information. Input is done in turn of the blank data block, the material data block and the task data block.

Fig. 8.9 is line-printed output of the canonical data format that is translated from the input language by SCANNER.

Fig. 8.10 is the material allocation drawings as the results of auto-allocation by OPTNST. In Fig. 8.11, * implies a waste area.

Fig. 8.11 is the table listed up by TSKCLS:

Fig. 8.12 is the tool path simulation results which is output by OPTPTH.

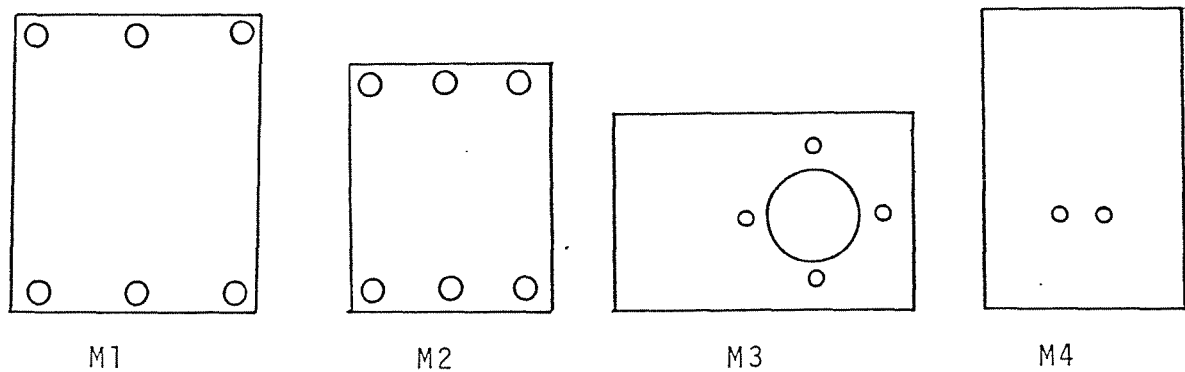


Fig. 8.7 Simple example of material input geometry

Fig. 8.8 An example
of input language

```

#BLANK
B1=B/900,900,3,5,10
#BFINI
#MATERIAL
M1=M/250,300,8
MOVE/25,20
L1=PTH/LIR, INCR,2,100
MOVE/25,280
L2=PTH/LIR, INCR,2,100
#MEND
M2=M/200,250,8
MOVE/25,20
L3=PTH/LIR, INCR,2,75
MOVE/25,230
L4=PTH/LIR, INCR,2,75
#MEND
M3=M/200,300,4
MOVE/100,200
H1=HOL/50,10,10
MOVE/100,200
A1=PTH/ARC,65,0, INCR,3,AT,90
#MEND
M4=M/200,300,4
P1=PNT/80,100
P2=PNT/120,100
#MEND
#MFINI
#TASK
T1=PUNC/1,5/L1,L3,M1/L3,L4,M2
T2=PUNC/2,3/A1,M3/P1,P2,M4
T3=PUNC/3,10/H1,M3
*TFINI

```

```

1 *** *****< .BLIST >*****
2
3 *
4 *      NO.      SYMBOL.      WIDTH.      LENGTH.      THICKNESS.      NUMBER.      COST.      *
5 *      1        B1          900        900          3            5            10        *
6
7 *** *****< .BLIST END >*****
8
9
10

```

(a) B-file

```

1 *** *****< .MLIST >*****
2
3 *      NO.      SYMBOL.      WIDTH.      LENGTH.      NUMBER.      POINTER.      *
4 *      1        H1          250        300          8            1            *
5 *      2        H2          200        250          9            19           *
6 *      3        H3          200        300          4            37           *
7 *      4        H4          200        300          4            56           *
8
9 *** *****< .MLIST END >*****
10
11
12
13 *** *****< .MCANO LIST >*****
14 * N .      SYMBOL KOSU CORD  A1  A2  A3  A4  A5  A6  A7  A8  A9  A10 A11 A12 A13 A14 A15 A16 A17  *
15 *      L1  9  LIR  25  20  0  2  100  *  *  *  *  *  *  *  *  *  *  *  *
16 *      L2  9  LIR  25  280 0  2  100  *  *  *  *  *  *  *  *  *  *  *
17 *      L3  9  LIR  25  20  0  2  75  *  *  *  *  *  *  *  *  *  *  *
18 *      L4  9  LIR  25  230 0  2  75  *  *  *  *  *  *  *  *  *  *  *
19 *      H1  9  HOL  10020004 50  10  10  *  *  *  *  *  *  *  *  *  *  *
20 *      A1  10 ARC  100  200 65  0  3  90  *  *  *  *  *  *  *  *  *  *  *
21 *      P1  6  PNT  80  100  *  *  *  *  *  *  *  *  *  *  *  *  *  *
22 *      P2  6  PNT  120 100  *  *  *  *  *  *  *  *  *  *  *  *  *  *
23
24 *** *****< .MCANO END >*****
25
26
27
28
29
30

```

(b) M-file

```

1 *** *****< .THORK LIST >*****
2
3 *      NO.      1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  *
4 *      1 - 10    -2  L1  L2  *  -2  L3  L4  *  -1  A1  *
5 *      11 - 20   *  -2  P1  P2  *  -1  H1  *  0  *  *
6 *      21 - 30   *  *  *  *  *  *  *  *  *  *  *
7
8 *** *****< .THORK LIST END >*****
9
10 *** * SCANNER END *****
11
12
13 *** *****< .TPREPR LIST >*****
14 * N .      SYMBOL KOSU CODE  A1  A2  A3  A4  A5  A6  A7  A8  A9  A10 A11 A12 A13 A14 A15  *
15 *      T1  10  PUNC  1  5  1  H1  5  H2  *  *  *  *  *  *  *  *
16 *      T2  10  PUNC  2  3  9  H3  12  H4  *  *  *  *  *  *  *  *
17 *      T3  8  PUNC  3  10 16  H3  *  *  *  *  *  *  *  *
18
19 *** *****< .TPREPR LIST END >*****
20
21
22
23
24
25

```

(c) T-file

Fig. 8. 9 B-file, M-file and T-file outputs

INPUT

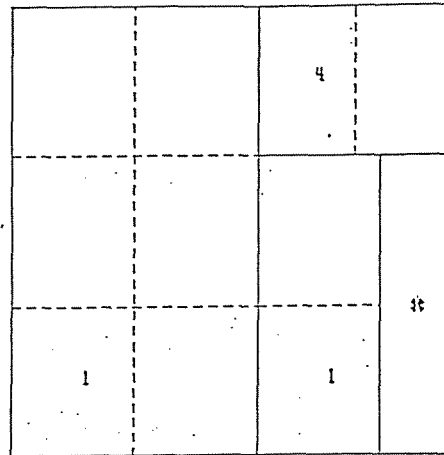
MATERIAL NO.	LEN.	WID.	REQ. NO.
1	500	500	8
2	500	500	8
3	500	500	4
4	200	100	4
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

BLANK
LENGTH 500 WIDTH 500

OUTPUT

NO.	USED NO.
1	8
2	0
3	0
4	2
5	0
6	0
7	0
8	0
9	0
10	0

WASTE
11.11 %



INPUT

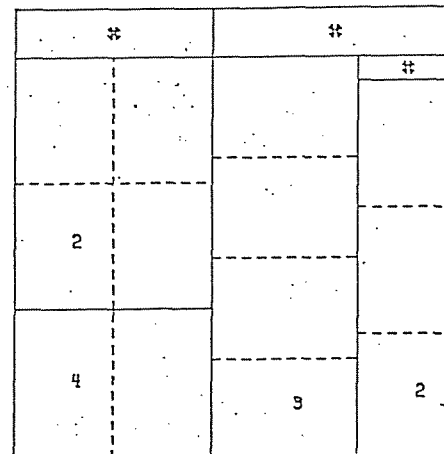
MATERIAL NO.	LEN.	WID.	REQ. NO.
1	200	200	0
2	200	200	0
3	200	200	4
4	200	200	2
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

BLANK
LENGTH 500 WIDTH 500

OUTPUT

NO.	USED NO.
1	0
2	2
3	4
4	2
5	0
6	0
7	0
8	0
9	0
10	0

WASTE
12.22 %



INPUT

MATERIAL NO.	LEN.	WID.	REQ. NO.
1	200	200	0
2	200	200	1
3	200	500	0
4	200	100	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

BLANK
LENGTH 500 WIDTH 500

OUTPUT

NO.	USED NO.
1	0
2	1
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

WASTE
23.23 %

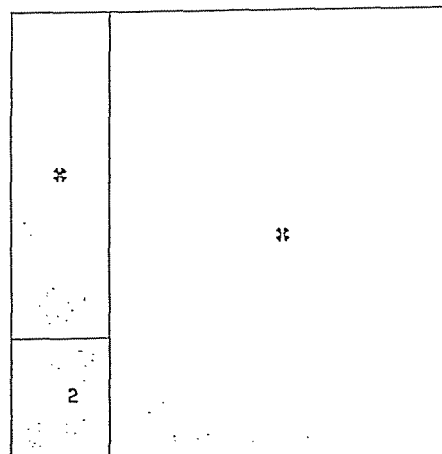


Fig. 8.11 Allocating result

*** TASK LIST ...TSL ***

WORK SYMBOL	PRIORITY	TOOL NO.	POINTER (S)	POINTER (E)
T1	1	1	1	2
T2	2	2	3	3
T3	3	3	0	0

*** WORK LIST ...WLIST ***

HAT. NO.	XAXIS NO.	YAXIS NO.	X	Y	PNT. (S)	PNT. (E)
1	2	3	0	0	1	18
1	1	2	500	0	1	18
4	2	1	500	600	19	30

*** TASK GEOMETRY LIST ...VGL

NO.	SYMBOL	KOSU CODE	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A14	A15	A16	A17
1	L1	9 LIR	25	20	0	2	100	*										
2	L2	9 LIR	25	280	0	2	100	*										
3	P1	6 PNT	80	100	*													
4	P2	6 PNT	120	100	*													

*** TASK LIST ...TSL ***

WORK SYMBOL	PRIORITY	TOOL NO.	POINTER (S)	POINTER (E)
T1	1	1	1	2
T2	2	2	3	4
T3	3	3	5	5

*** WORK LIST ...WLIST ***

HAT. NO.	XAXIS NO.	YAXIS NO.	X	Y	PNT. (S)	PNT. (E)
2	2	2	0	300	1	18
2	1	3	700	0	1	18
4	2	1	0	0	19	30
-3	1	4	400	0	31	40
-3	1	4	400	0	41	49

*** TASK GEOMETRY LIST ...VGL

NO.	SYMBOL	KOSU CODE	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A14	A15	A16	A17
1	L3	9 LIR	25	20	0	2	75	*										
2	L4	9 LIR	25	230	0	2	75	*										
3	P1	6 PNT	80	100	*													
4	P2	6 PNT	120	100	*													
5	A1	10 ARC	100	200	65	0	3	90	*									
6	H1	9 HOL	100	200	50	10	10	*										

*** TASK LIST ...TSL ***

WORK SYMBOL	PRIORITY	TOOL NO.	POINTER (S)	POINTER (E)
T1	1	1	1	1
T2	2	2	0	0
T3	3	3	0	0

*** WORK LIST ...WLIST ***

HAT. NO.	XAXIS NO.	YAXIS NO.	X	Y	PNT. (S)	PNT. (E)
2	1	1	0	0	1	18

*** TASK GEOMETRY LIST ...VGL

NO.	SYMBOL	KOSU CODE	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A14	A15	A16	A17
1	L3	9 LIR	25	20	0	2	75	*										
2	L4	9 LIR	25	230	0	2	75	*										

Fig. 8.10 TSKCLS output

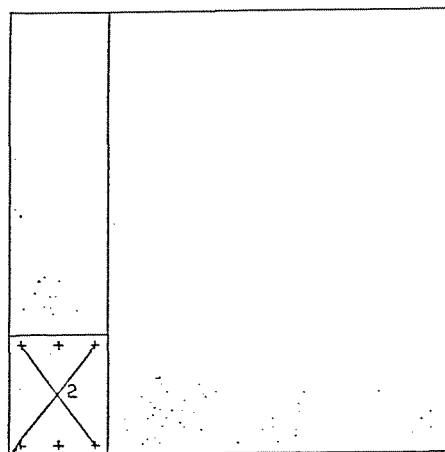
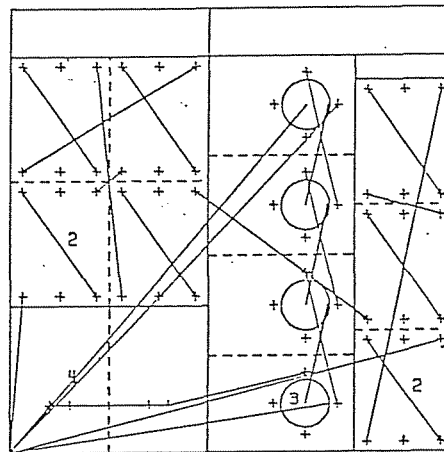
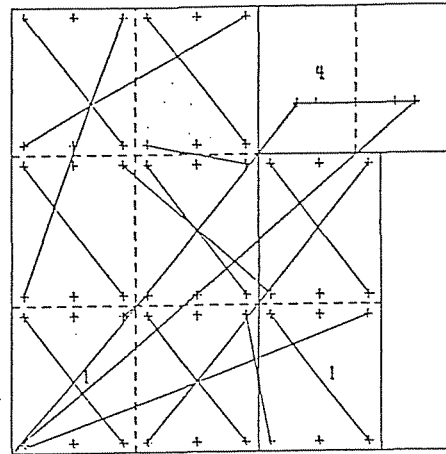


Fig. 8.12 Punching-press tool path

8.4 Conclusion

CAMPS (Computer-Aided Manufacturing for Punching-press and Shearing) system is designed and developed, based on the studies of space allocation problem. Through the system design and development, the followings are concluded.

1. The CAM system for punching-press and shearing is designed and developed in the consideration of easy input and implementation.
2. Simple language is designed as the input of system.
3. To automate planning of each processor in punching-press and shearing, the processors are developed and coded by modeling each process by the mathematical description.
4. The system validity is demonstrated by showing the output examples of CAMPS system.
5. CAMPS system will contribute transferring DCN system to sheet metal manufacturing.
6. It is expected that saving the resources and reducing the process planning time are accomplished by the use of CAMPS system.

Reference

1. Furukawa, M, "Optimum Shear and Nest Scheduling for an NC Shearing System", Jornal of Asahikawa Technical College, Vol.17, 1980.

9. Conclusion

There are many different situations and fields in which space allocation problems arise. Such problems, when encountered, must be analyzed along with their occurrence situations from the viewpoint of the common recognition of the space allocation problems. Then they should mathematically described and modeled.

In modeling, it must be confirmed that the problems to be modeled are already turned into the subproblems of the real problems and the obtained optimum solutions of the problems modeled are not the optimum solutions of the real problems. This means the subproblems of the real problems are solved and the suboptimum solution is obtained instead of the optimum solution.

The space allocation problems to be mathematically modeled are combinatorial in most cases. Hence the proof of the optimum solution is guaranteed by searching all the feasible solutions. However, it is impossible to execute such searching procedure because of an enormous combinatorial number of solutions. This directs the problem-solution method towards suboptimum-optimum solutions.

The basic strategy for developing such a method in this paper is to "divide the problems into some solvable subproblems and obtain the optimum-suboptimum solution". From experience, the optimization of subproblems approaches to the optimization of the whole problem although it does not compose the optimization of the whole problems. The methods developed in Chapters 2, 3 and 4 stand on this strategy.

Meanwhile, there is a question as to whether a large scale computer is really needed to solve the problems when most of the problems occur daily or in every hour. And the problems belong to "NP-Complete Problem". Economically the use of a large scale computer is not a good choice because of high costs if the practical solution is reached by the use of a mini-computer. Thus the algorithms are established for a midium-small sized computer.

A recursive procedure is designed so as to realize the algorithms for such a computer. If the algorithm does not consist of the recursive procedure, many procedures must be prepared corresponding to many cases of the subproblems. This implies that the method which has the above-mentioned function is rather said to be a kind of "artificial intelligence". It will be needed in the near future perhaps, but it costs too much at present.

The space geometries treated in this paper have almost regular shapes. Although the graphic processing techniques for the space allocation problems are developed so as to deal with irregular shapes in Chapter 7, these are not applied sufficiently to the problems. However, as the graphic processing is essential for the treatment of the space allocation problems, especially when irregular shapes of the spaces are represented, the graphic processing techniques discussed will be of great use.

In closing the paper, it must be emphasized that the optimization of the problems means the optimization of the subproblems as the result of modeling the real problems. Therefore, even if the complete optimization of the problem is accomplished, it is the partial optimization of the real problem. Thus, we have already accepted that the partial optimization will approach to the optimization of the real problem. In order to accomplish the complete optimisation, we must know the whole system of the problem. A couple of hundred years ago, economist Adam Smith said, "An invisible hand will lead the partial optimization to the optimization of the whole system". We may say that one of the final goals of the optimization is to establish the "invisible hand of the system".

Acknowledgement

Works for the paper are supported by many people. The author's special thanks go to Professor N. Okino, of Hokkaido University, for his direction and guidance in completion of this work. Since his under-graduate student days, Professor N. Okino has taught him what is CAD and CAM, and Professor Okino's enthusiastic lectures and discussions have impressed him and encouraged him to make a study and complete this work. The author wishes to thank Associate Professor Y. Kakazu, of Hokkaido University, for his invaluable suggestions and discussions. In addition, he would like to thank Dr. K. Hoshi and Dr. R. Miura, of the previous and current president ministers of Asahikawa Technical College, for their genial supports. He also wishes to thank Professor K.K. Wang, of Cornell University, for his training and education of a software system engineer in 1977 - 78 in U.S.A. His thanks go to members of Combinatorial Research Groups Dr. H. Kubo, Dr. K. Sekiguchi, Dr. A. Ouchi of Hokkaido University, and Mr. N. Wakabayashi of Otaru Commercial College, for their innumerable helpful discussions. He wishes to thank his colleagues of Asahikawa Technical College, specially Mr. H. Konno,

Dr. T. Akiyama for their help of study and Mr. T. Takeuchi for his frequent correction of English. The author owes his students for some program codings as graduate studies. His thanks go to Mr. Y. Tomooka, of Mitsubishi Electric Company, for his presentation of industrial experiment data, and Mr. M. Hashimoto and Mr. T. Kani, of Murata Machinery Company, for their realization of the author's theory in the industry. Finally, the author wishes to thank Miss Sato, of Fuji High School, for her English correction and typewriting of the manuscript.

Appendix A Mitsubishi Metal Sheet Production System

A.1 Introduction

A development of an automated metal sheet production line system was intended for the sake of rationalization of a factory when a construction of a new factory had been planned by Mitsubishi Electric Company. In this new system for the metal sheet production, N/C shearing machines and N/C turret punching-press machines are equipped and they are directly controlled by a host computer. A feature of the new system is that decision making of the metal sheet products allocation in large metal sheets is executed automatically and it dominates the system. P.B.M. (Pair to Block Method) proposed in chapter three was adopted for this allocation decision making. This appendix introduce of P.B.M. adoption for the metal sheet production in the factory.

A.2 Hardware System

A product flow diagram of the designed metal sheet production system is shown in Fig. A.1.

Corresponding to each production process, hardware equipments are assigned in the production line. Hardware equipments assigned and their functions are as follows;

(Note: a is for Process, b is for Equipment and c is for Function.)

1. a. Initial shearing b. Shearing machine 1
 c. Shearing coiled metal sheet with desired length
2. a. Levelling b. Leveller machine
 c. Making a metal sheet flat
3. a. Edge trimming b. Shearing machine 2
 c. Trimming edges of metal sheet
4. a. Transportation 1 b. Turn table
 c. Selecting a shear machine to be used
5. a. Transprotation 2 b. Convayer
 c. Transporting a metal sheet and setting it to a turret punching-press machine
6. a. Punching-press 1 b. Turret punching-press machine 1
 c. Punching out holes
7. a. Punching-press 2 b. Turret punching-press machine 2
 c. Punching out holes

8. a. Shearing

b. Shearing machine 1

c. Guillotine shearing

9. a. Shearing

b. Shearing machine 2

c. Right angle shearing

The production line of the system becomes as shown in Fig. A.2.

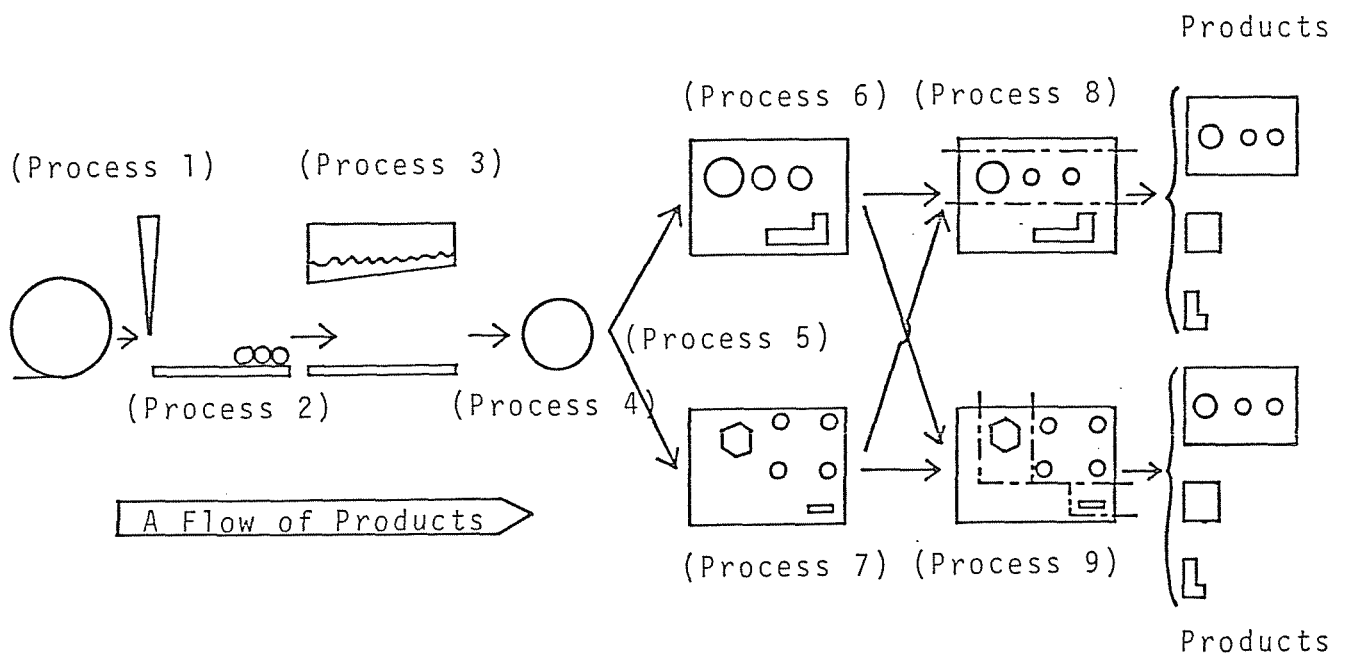


Fig. A.1 Products flow line

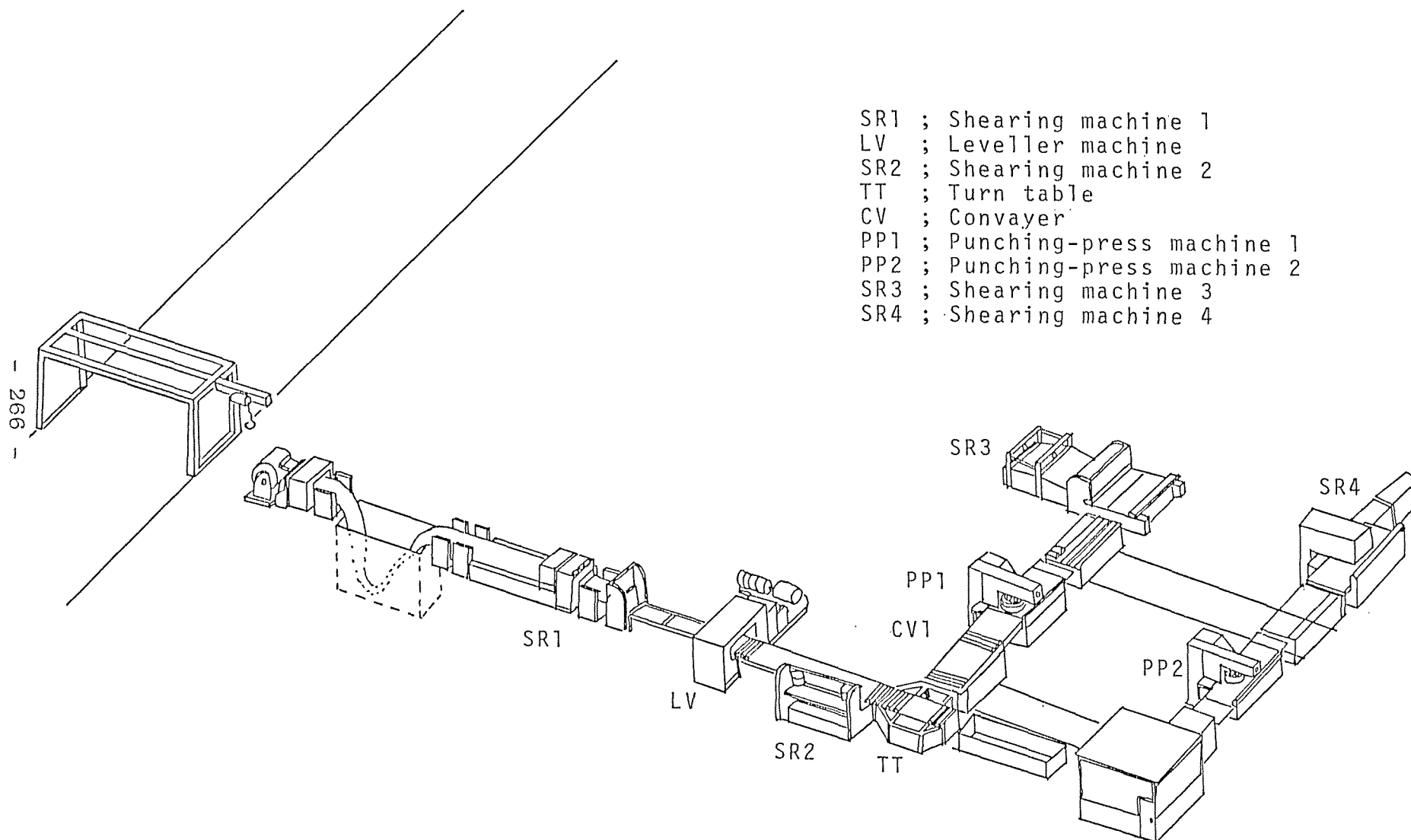


Fig. A.2 Hardware system

A.3 Software System

The software system developed consists of three procedures: input data processing, processing for auto-allocation of metal sheet products, and punching-press processing. We only describe the procedure concerning to P.B.M., processing for auto-allocation.

In processing auto-allocation, three algorithms are prepared for the metal sheet product allocation.

Algorithm A: P.B.M. This is prepared for the allocation of many different-sized products. Both types of shearing machines (Guillotine type and Right angle type) are available.

Algorithm B: Pyramid building method. This is adopted for complements of P.B.M. Some cases occur on P.B.M. in which a block built up by more than two products brings a better solution than a block built up by pyramid products. In such cases, this algorithm works.

Algorithm C: If a number of the same products are required, this algorithm works. This algorithm allocates products

to the shape of mesh.

By using three algorithms, the auto-allocation procedure is composed. Allocation procedure is described by the following 8 steps.

Step 1: Product data are input. If there is no datum as input, the procedure is terminated.

Step 2: Input data are sorted in accordance with metal materials, priorities and so on.

Step 3: Algorithm C is employed to allocate the same kinds of products. In this procedure, if a waste ratio resulted from the allocation is bigger than the pre-set one, go on to the next step. If not, go to step 8.

Step 4: An algorithm employed is selected between A and B by the adoption of shearing method. If a right angle shearing, go to step 7. If not, go to the next step.

Step 5: Algorithm B is employed.

Step 6: If a waste ratio resulted from the allocation is bigger than the pre-set one, go to the next step. If not, go to step 8.

Step 7: Algorithm A is employed.

Step 8: The allocation results such as punch-pressed hole coordinate determination and an arrangement of the nested products are edited.

A process flow diagram is shown in Fig. A.3. Fig. A.4 is some results of the allocation employed by the developed procedure.

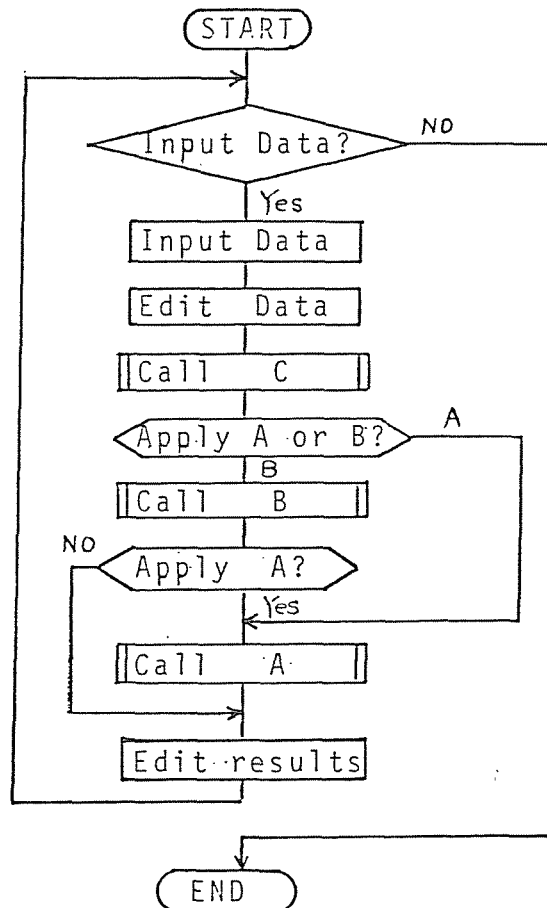


Fig. A.3 A flow diagram

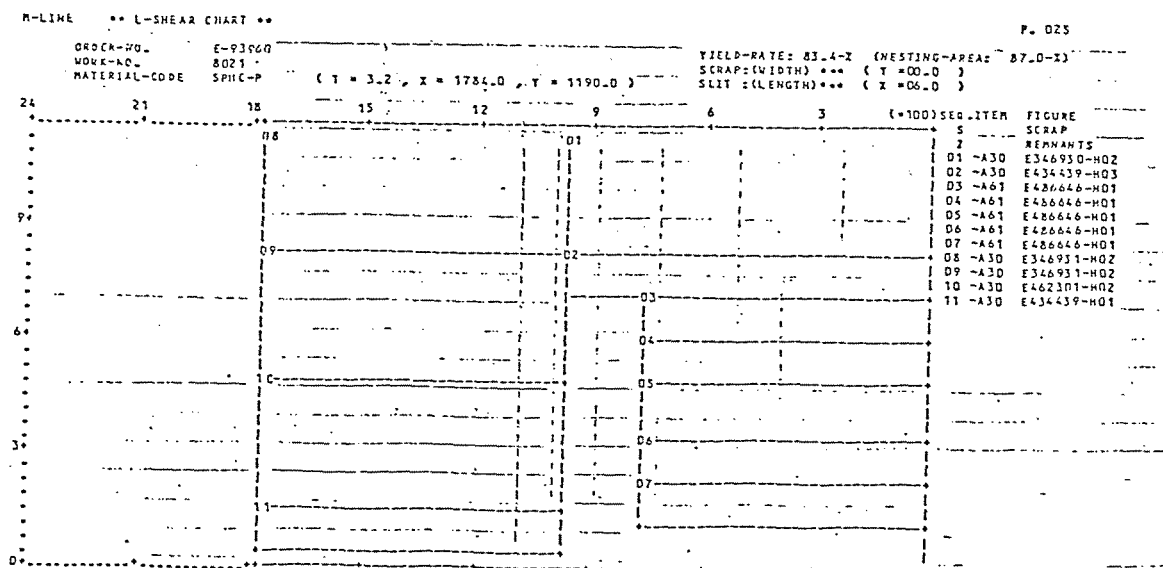
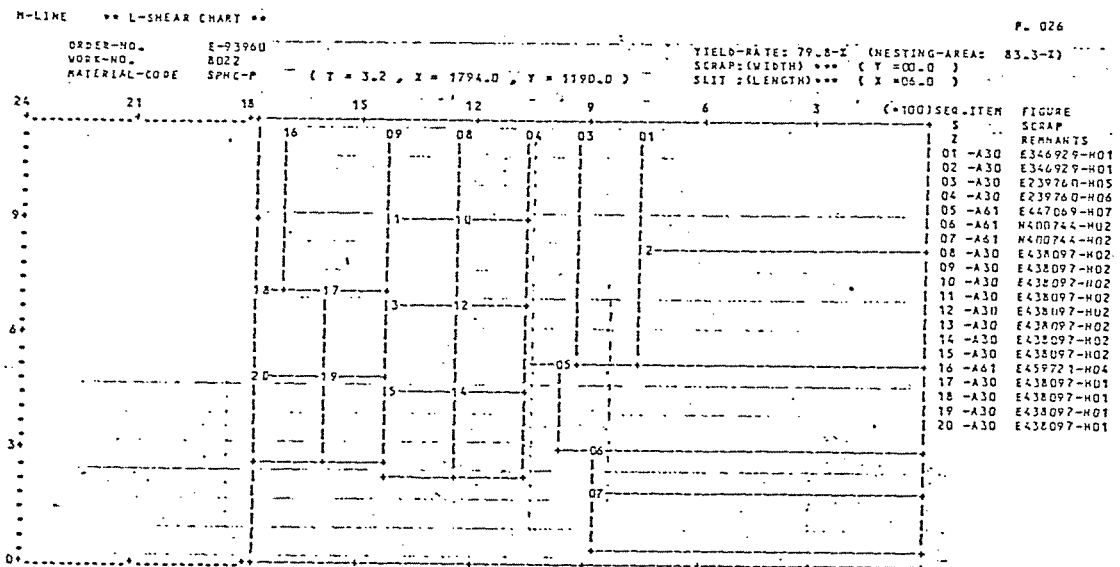
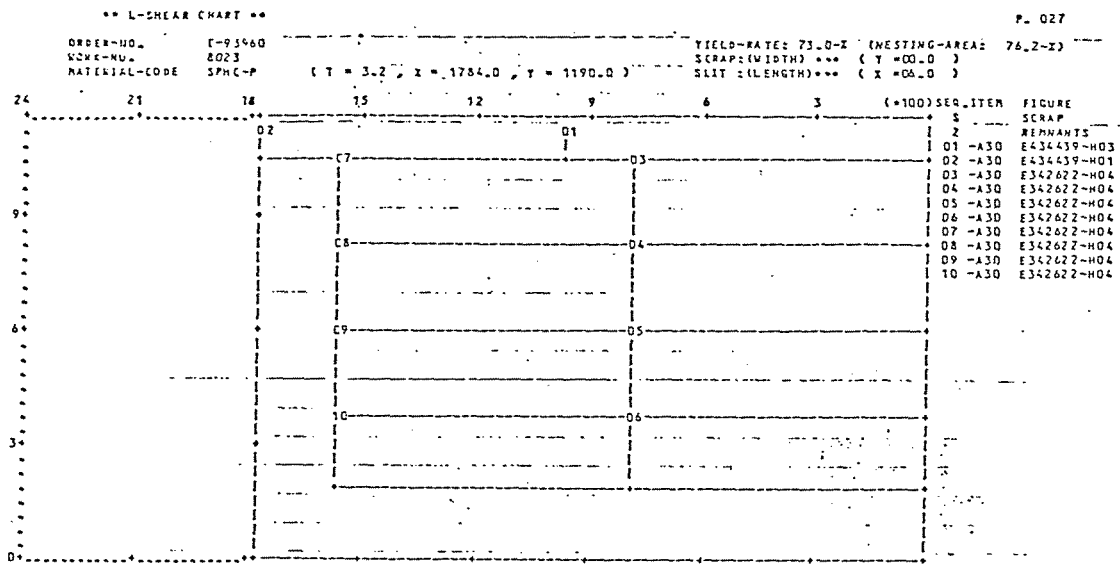


Fig. A.4 Examples of product allocation

A.4 Effects of System Development

After the introduction of the developed system, the productivity is widely improved. This effect is shown in Table A.1. Before the development of the system, metal sheet products are sorted to large area products and small ones. It becomes unnecessary to sort them in this system.

By estimating manufacturing time hour/year in Table A.1 as workers, 9.1 workers per year is reduced to 2 workers per year after the development of the system.

Table A.1 Effect of system development

	Pre-system development		Post-system development
	No. of products	hr./yr.	hr./yr.
Large area of products	14728	3535	3782
Small area of products	41052	11412	
Total	55780	14947	3782

Appendix B OPTI-CUT System Mannual

A method proposed in chapter four is applied to metal sheet cutting at Murata Machinery Company. A Program coded is modified to meet a practical problem that occurs by the implemented NC shearing machines and products tolerance. Such problems are repositioning and edge trimming.

When the area of machine tables available for shearing is smaller than the area of blanks (raw metal sheets), the blanks must be set and fixed at least twice. This operation is called "repositioning". As frequent repositioning makes undesirable effects on the product tolerance, repositioning is usually operated only one time if necessary. A modified nesting algorithm conquers this problem. Edge trimming means that four edges of the product is cut down. When three products are allocated in the blank as shown in Fig. B.1, a kind of "Burr" is generated. This prevent the products from keeping tolerance. This problem is also overcome by the addition of the offsets to the product.

The modified processor is named "OPTI-CUT". A manual for OPTI-CUT will explain how these problems are remedied and the proposed method works.

The manual of OPTI-CUT is described as follows.

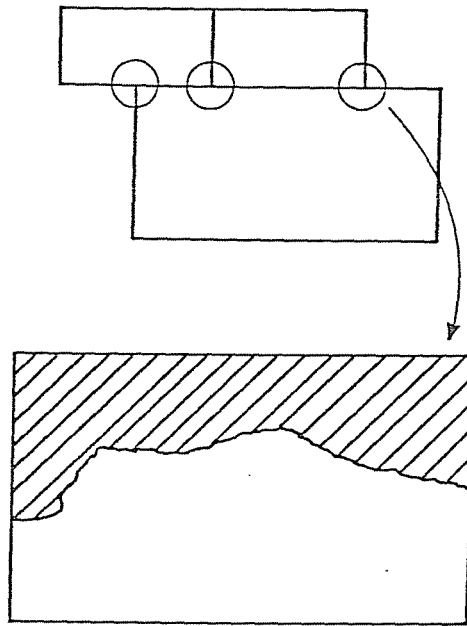


Fig. B.1 "Burr"

OPTI-CUT manual

General Descriptions

The OPTI-CUT enable you to easily and rapidly produce the NC tape for shearing machine, which has L-shape blade, with input sizes, quantities and codes of the sheet metal to be sheared. The OPTI-CUT provides you productivity, profitability and easy inventory.

The OPTI-CUT system is implemented on General Electric Time Sharing Service MARK III. GE provides two types of computer services which are Foreground and Background. The OPTI-CUT system is only accessed in Foreground service at this time.

Fig 1 below shows you OPTI-CUT system diagram.
In this manual, BLANK means source sheet metal to be sheared, and MATERIAL means requested sheet metal to be sheared.

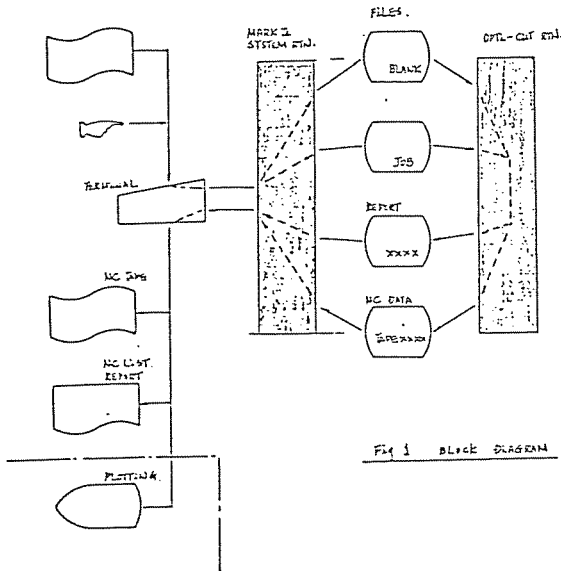


Fig 1 BLOCK DIAGRAM

2-2. "JOB" File

"JOB" file is consisted of PARAMS and MATERIALS, it must be entered PARAMS first as shown in Fig 3. "JOB" file will normally be generated every OPTI-CUT runs.

It is allowable to input several different codes of material at random in one "JOB" file. OPTI-CUT will arrange them and process them in order. Maximum quantity of materials is 50 kinds for each code and 150 kinds in total.

Registration format is following.

(N),(X),(Y),(C),(V)

Where

- N: Material number. Any number can be used in integer up to 3 digit.
- X: Length of material along X-axis in mm. It may be grained to Y-axis by the OPTI-CUT.
- Y: Length of material along Y-axis in mm. It may be grained to X-axis by the OPTI-CUT.
- C: Code. Refer to chapter 2-1.
- V: Quantity. Maximum quantity is limited 9999.

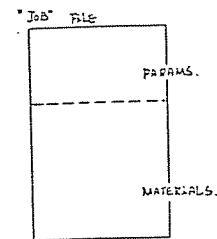


Fig 3 STRUCTURE "JOB" FILE

2. Generating a File

Two files named "BLANK" and "JOB" are required for access the OPTI-CUT as shown in Fig 1. Those two files could be generated by use of MARK III system routine.

2-1. "BLANK" File

"BLANK" is a inventory file which contents will be equal to the blanks currently exist in your factory. The OPTI-CUT finds the appropriate blanks from the "BLANK" file, and quantity of blanks used in the computation is automatically subtracted from the "BLANK" file. To print "BLANK" file by LIS command shows you current quantities of blank in your factory. It is user's obligation to adjust the contents of "BLANK" file unless your manufacture the blanks resulted by OPTI-CUT.

Registration format is following

(N),(X),(Y),(C),(V),(CR),(YD),(XR)

Where

- N: Sheet(Blank) number. Any number can be used in integer up to 3 digit.
- X: Length of blank along X-axis in mm.
- Y: Length of blank along Y-axis in mm.
- C: Code. Any number can be used in following format

XXX.X (X=0~9)

Creating the code as follow is recommended.

- 1The left most 2 digits are corresponded to sheet metal code.
- 1The right most 3 digits with floating point are corresponded to thickness of the blank.

For example,
Steel = 01
Stainless steel = 05
Aluminum = 10

then

- 013.2 means the blank of steel with 3.2 thickness.
- 051.2 means the blank of stainless steel with 1.2 thickness.
- 102.5 means the blank of aluminum with 2.5 thickness.

C: Quantity. Maximum quantity is limited 9999.

Following three variables CR, YD and XR are related with PARAM 10 describe later. Only when PARAM 10 is set to 2, these variables are referenced by OPTI-CUT. That is, inputting these variables will not be necessary unless PARAM 10 is set to 2.

- CR: Width of strip reserved for gripping blank.
- YD: Salvage allowance in X-direction of blank.
- XR: Salvage allowance in Y-direction of blank at shear end.

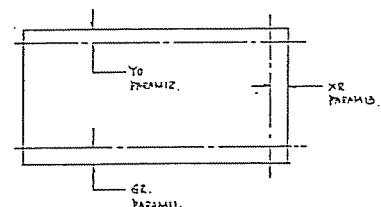


Fig 2 SALVAGE ALLOWANCES

If these variables are not set in both "BLANK" and PARAM,
CR (PARAM 11) = 40. mm
YD (PARAM 12) = 0. mm
XR (PARAM 13) = 8. mm
are defaulted.

Cautions

- 1The delimiter could be comma or spaces, comma is used in this manual.
- 1Although up to 10 blanks in each code are referenced by OPTI-CUT some of blanks will be ignored if registered more than 10 blanks.
- 1File could be generated by both tape or key-board input.

3. Parameter (PARAM)

Parameter is prepared making a function of OPTI-CUT flexible, you will not assign them if defaulted parameter is used.

Registration format is following.

(PARAM),(C),(V)

Where

- C, Classification. Detail is described below.
- V, Variable. The number must include a decimal point.

Classification	Pre-set	Descriptions
25	0.	0.1 Produce repositionning automatically. 1.1 Repositionning is not considered.

(Not used at this time)

(30)	0.	Width of duncy cut along X-axis in mm.
(31)	0.	Width of duncy cut along Y-axis in mm.

Classification	Pre-set	Descriptions
4	0.	0.1 Report requires the listing of blanks which are referenced by OPTI-CUT. 1.1 Not require.
	0.	0.1 Report requires the listing of materials which are processed by OPTI-CUT. 1.1 Not require.
3	0.	0.1 Priority is set by the square of materials when OPTI-CUT simulates layout. 1.1 Priority is set by the length.

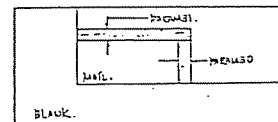


Fig 4 DUNCY AREA

(Not used at this time)

(PARAM 30 & 31 not available yet)

10	0.	Refer to chapter 2-1 0.1 Both "BLANK" (CR,YO,XR) and PARAM 11-13 are not referenced by OPTI-CUT. It means following values are required. PARAM 11 (NR) = 40. mm PARAM 12 (YU) = 0. mm PARAM 13 (XR) = 0. mm 1.1 PARAM 11-13 are referenced. Unless otherwise PARAM 11-13 are required in the "JOB" file, defaults will automatically set. NR,YO and XR are not effect if required. 2.1 CR,YO and XR are referenced. Unless otherwise CR,YO and XR are required in the "BLANK" file, error message will be appeared in the report. PARAM 11-13 are not effect if required.
----	----	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

50	TAPE	Name of the output file for NC data (refer to Fig 1) File name is acceptable from TAPE1 to TAPE9999. Decimal point is not necessary only for this parameter.
51	0.	0.1 Sequence number is not required in NC data. 1.1 Sequence number required.

Classification	Pre-set	Descriptions
11	40.	Width of strip reserved for gripping blank. Refer to Fig 2.
12	0.	Salvage allowance in X-direction of blank. Refer to Fig 2.
13	0.	Salvage allowance in Y-direction of blank. Refer to Fig 2.

(Not used at this time)

15	570.	Maximum length on half-shwar.
----	------	-------------------------------

(Not used at this time)

20	680.	Maximum cut of blade X-axis in mm.
21	680.	Maximum cut of blade Y-axis in mm.
22	1830.	Maximum distance of table travel X-axis in mm.
23	1400.	Maximum distance of table travel Y-axis in mm.

4. GC's System Commands

This chapter intentionally omitted

5. Access the OPTI-CUT

The OPTI-CUT can be accessed by independent run, which is one of service in Foreground, to decrease the running cost. Following procedure can be taken if you need the optimized result urgently.

Independent run has three kind of priorities such as

1. PRI(DEF) — Express
2. PRI(DEF) — Within 3 hours
3. PRI(QVE) — Overnight

5-1. Sign-on

After your terminal is connected to the GE's system, the system begins a formal sign-on sequence.

1. Input user number.
 2. Input pass words.
 3. Input ID.
 4. Input the language to be used.
- F77 must be required for OPTI-CUT.

5-2. Generating the file

Files named "BLANK" and "JOB" are prepared by user. Format for each file is previously described, refer to chapter 2 if necessary.

1. Generating the file from key-board

```
NEW OR OLD-
NEW JOB
READY
=====
SAV ( or REP )
READY
```

2. Generating the file from paper tape

```
NEW OR OLD-
NEW JOB
READY
DSMT
READY FOR INPUT
( turn tape reader on )
=====
( strike break key )
READY
SAV ( or REP )
READY
```

RUN OPTICUT

This is report.

```
PROGRAM STOP AT XXXX
USED XXXX UNITS
READY
```

Differences between this type of run and independent run is to print the report immediately and automatically after end of run instead of generating the report file. NC data file is generated when OPTI-CUT is terminated as well as independent run.

5-3. Access the OPTI-CUT

After prepared two files such as "BLANK" and "JOB", OPTI-CUT could be accessed to input following format.

```
IND-100 OPTICUT,,RPRT,,PRI(DEF)
(1)      (2)      (3)
```

Where

- (1): Maximum CRU units.
- (2): Name of the report file (refer to Fig 1) up to 8 characters.
- (3): Setting priority.

To set the priority other than EXP, system will normally be sign-off.

5-4. Getting optimized results

OPTI-CUT produces two output files as a result which are named by user GE's system command enable you to print and punch out listings and/or NC tapes.

1. Listing the report

```
OLD RPRT
READY
LIS
=====
READY
```

2. Output NC tape and listing

```
OLD TAPE
LIS
( turn punch on )
=====
( turn punch off )
READY
```

Although plural of tapes are output simultaneously, approximately 10 inches of null codes is put on front and tail of each NC data.

NC tapes you generate with above procedure are ISO code. If you need to create EIA code of tapes, use one of NC's library routine "NUTAP".

6. EXAMPLES

```
IND-100 OPTICUT,,RPRT,,PRI(DEF)
NEW OR OLD-
NEW JOB
READY
=====
SAV ( or REP )
READY
```

BLANK Q7140JST 04/10/80

```
F11 1029. 914. 11.2 100 20. 20. 10.
F11 1029. 914. 12.1 100 20. 15. 20.
F11 1029. 914. 13.1 100 0. 0. 5.
F11 2438. 1217. 21.1 100 20. 0. 15.
F11 1029. 914. 21.1 100 20. 20. 10.
F11 2438. 1217. 22.1 100 35. 0. 0.
```

```
READY
END JOB
```

```
READY
=====
```

```
===== FOR INPUT
=====
100 1 00. 100. 21.1 1
101 100. 200. 21.1 1
102 100. 300. 21.1 1
103 100. 400. 21.1 1
104 100. 500. 21.1 1
105 100. 600. 21.1 1
106 100. 700. 21.1 1
107 100. 800. 21.1 1
108 100. 900. 21.1 1
109 100. 1000. 21.1 1
110 100. 1100. 21.1 1
111 100. 1200. 21.1 1
112 100. 1300. 21.1 1
113 100. 1400. 21.1 1
114 100. 1500. 21.1 1
115 100. 1600. 21.1 1
116 100. 1700. 21.1 1
117 100. 1800. 21.1 1
118 100. 1900. 21.1 1
119 100. 2000. 21.1 1
120 100. 2100. 21.1 1
```

```
READY
END
```

```
IND-100 OPTICUT,,RPRT,,PRI(DEF)
```

```
READY
```

```
=====
0000..20 CRU 0000.01 ICH .0000. KI
OPT AT 09:47JST 04/10/80
```


HERRON
 UIC: 00030000
 PASSWORD

 ID: 0000
 SYSTEM: F77
 HCM OK OLD-
 LIS RPRT

KFRT 10:11VJST 04/10/80

LFICUT 10:17JST 04/10/80

*** SHEAR OPTIMIZATION PROGRAM ***
 AUTHORIZED BY HUKATA MACHINERY CO

— EXECUTION FOR CODE 21.1 —

21.1 BLANKS YOU HAVE NOW ARE

P.NO	SIZE	NUMBER
NO.141	2430.00 # 1219.00	100
NO.151	1829.00 # 914.00	100

*** MATERIAL REGISTERED

P.NO	SIZE	NUMBER
NO.100	1000.00 # 300.00	4
NO.101	400.00 # 200.00	2
NO.102	100.00 # 350.00	8
NO.103	300.00 # 550.00	2
NO.104	600.00 # 200.00	8
NO.105	470.00 # 330.00	6
NO.106	400.00 # 220.00	2

* BLANK NO.141 LONGER THAN THROAT DEPTH.

*** OPTIMIZED RESULT 1

.BLANK USED NO.141
 .NUMBER OF BLANK 1 S
 .WASTE PERCENTAGE 9.688 %
 .FINISHED PARTS PER ONE NO.141 BLANK

P.NO	SIZE	NUMBER
NO.100	1000.00 # 300.00	4
NO.102	100.00 # 350.00	8
NO.103	300.00 # 550.00	2
NO.104	600.00 # 200.00	2
NO.105	470.00 # 330.00	2
NO.106	400.00 # 220.00	2

*** OPTIMIZED RESULT 2

.BLANK USED NO.151
 .NUMBER OF BLANK 1 S
 .WASTE PERCENTAGE 5.824 %
 .FINISHED PARTS PER ONE NO.151 BLANK

P.NO	SIZE	NUMBER
NO.101	400.00 # 200.00	2
NO.104	600.00 # 200.00	6
NO.105	470.00 # 330.00	4

*** TOTAL USAGE OF BLANK ARE

NO.141	2430.00 # 1219.00	1
NO.151	1829.00 # 914.00	1

*** TOTAL PRODUCE OF MATERIALS ARE

NO.100	1000.00 # 300.00	4
NO.101	400.00 # 200.00	2
NO.102	100.00 # 350.00	8
NO.103	300.00 # 550.00	2
NO.104	600.00 # 200.00	8
NO.105	470.00 # 330.00	6
NO.106	400.00 # 220.00	2

LIS TAPE0

TAPE0 10:24JST 04/10/80

*** RESULT NO = 1

NO01G00X121150Y63950M03
 NO02M41
 NO03X61850
 NO04M42
 NO05X183000Y67500
 NO06Y13100
 NO07Y6000
 NO08M22
 NO09X95000Y91900
 NO10M25
 NO11X81000
 NO12M22
 NO13X95000Y61900
 NO14M25
 NO15X03000
 NO16M22
 NO17X95000Y31900
 NO18M25
 NO19X83000
 NO20X123000Y11900
 NO21Y6000
 NO22X81000Y9900
 NO23Y6000
 NO24X29650Y63950M03
 NO25M41
 NO26X86950
 NO27M42
 NO28M22
 NO29X54300Y91900
 NO30M25
 NO31X42300
 NO32X2300Y99900
 NO33X112300Y36900
 NO34X02300
 NO35X0Y99900
 NO36X82300Y16900
 NO37X0Y91900
 NO38X2300Y6000
 NO39X3300Y28900
 NO40Y25700
 NO41Y6000
 NO42X300Y81900
 NO43Y71900
 NO44Y61900
 NO45Y51900
 NO46Y41900
 NO47Y31900

NO48Y21900
 NO49Y11900
 NO50X0Y37500
 NO51Y11900
 NO52Y6000
 NO53X180000Y-1000
 NO54X80000Y4000M03
 NO55M41
 NO56X170000
 NO57M42
 NO58X181500Y-1000
 NO59X-1000M03
 NO60M05
 NO61X183000Y14000M02

Appendix C Comparison of Boundary Evaluators

For the features of boundary evaluators, iso-contour displacement of boundary evaluators, Penalty function and a new evaluators are plotted by X-Y plotter. Calculation of the evaluators are run by OKITAC-4500-C. The results prove that the discussion in chapter seven is verified.

Experiment 1) Aspects of Penalty function and the new boundary evaluator are examined and plotted for the following simple geometry shape.

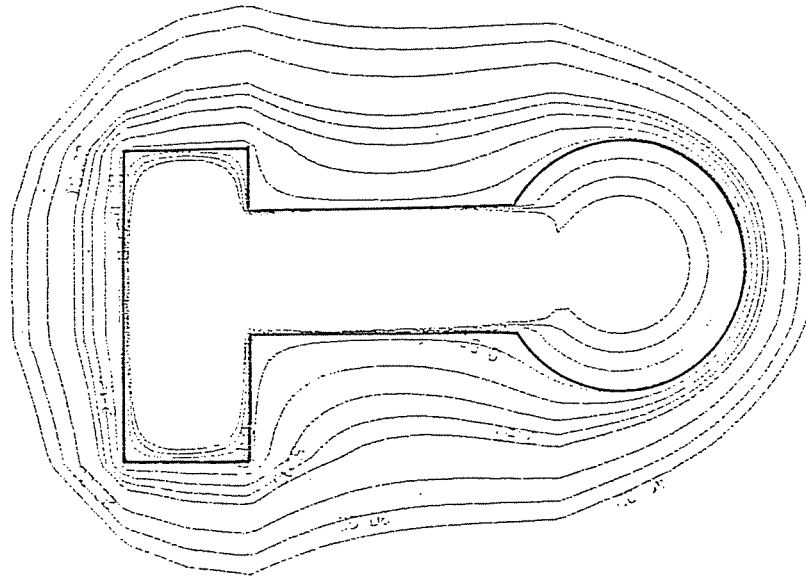
$$\begin{aligned}f_{11}(x, y) &= x - 4 \geq 0, & f_{21}(x, y) &= 5 - y \geq 0, \\f_{12}(x, y) &= 3 - y \geq 0, & f_{22}(x, y) &= 11 - x \geq 0, \\f_{13}(x, y) &= 10 - x \geq 0, & f_{23}(x, y) &= y \geq 0, \\f_{14}(x, y) &= y - 1 \geq 0, & f_{31}(x, y) &= x - 9 \geq 0, \\f_{31}(x, y) &= 2 - \{(x - 3)^2 + (y - 2)^2\}^{\frac{1}{2}} \geq 0. \\P &= \left[\bigcap_{i=1}^4 (f_{1i}(x, y) \geq 0) \right] \cup \left[\bigcap_{i=1}^4 (f_{2i}(x, y) \geq 0) \right] \\&\cup [f_{31}(x, y) \geq 0]\end{aligned}$$

Experiment 2) The same experiment as the experiment 1 is attempted. In this geometry shape, a different operation is examined. The geometry shape is described as follows.

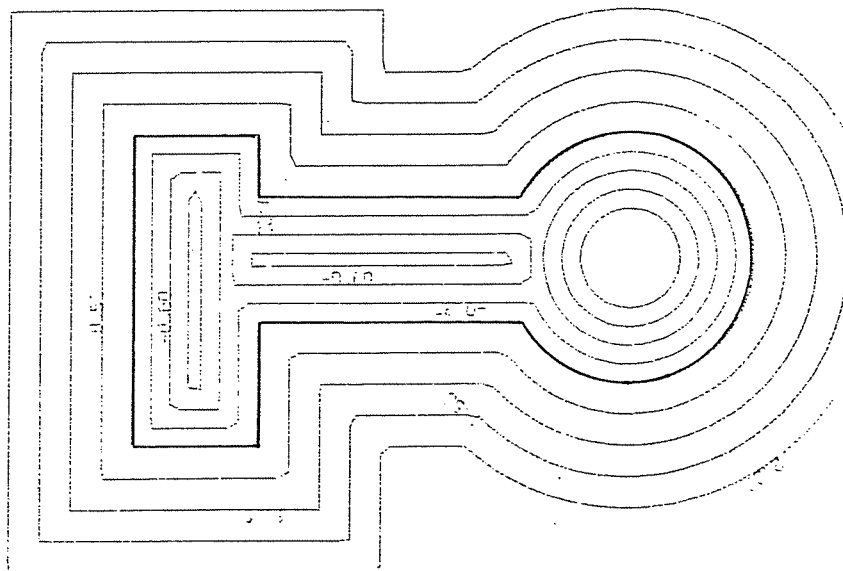
$$\begin{aligned}
f_{11}(x, y) &= x - 1 \geq 0, & f_{21}(x, y) &= x - 5 \geq 0, \\
f_{12}(x, y) &= 4 - x \geq 0, & f_{22}(x, y) &= 4 - y \geq 0, \\
f_{13}(x, y) &= 4 - y \geq 0, & f_{23}(x, y) &= 14 - x \geq 0, \\
f_{14}(x, y) &= y - 2 \geq 0, & f_{24}(x, y) &= y - 2 \geq 0, \\
f_{31}(x, y) &= 2 - \{(x - 4)^2 + (y - 3)^2\}^{\frac{1}{2}} \geq 0.
\end{aligned}$$

$$\begin{aligned}
P &= [\overbrace{[\bigcap_{i=1}^4 f_{i1}(x, y) \geq 0]} \cap [\{\bigcap_{i=1}^4 (f_{2i}(x, y) \geq 0)\} \\
&\quad \cup \{f_{31}(x, y) \geq 0\}],
\end{aligned}$$

$$\begin{aligned}
\text{where } [\overbrace{[\bigcap_{i=1}^4 (f_{1i}(x, y) \geq 0)]} \text{ shows } [\bigcup_{i=1}^4 (-f_{1i}(x, y) \\
\geq 0)].
\end{aligned}$$



(a) Aspect of Penalty function



(b) Aspect of the new evaluator

Fig. B.1 Result of experiment 1

