KBGT

A Knowledge-Based system for Group Technology

by

© Wadood M. Ibrahim

A Thesis
Submitted to the Faculty of Graduate Studies
Department of Mechanical Engineering
University of Manitoba
in partial fulfillment of the requirements
for the Degree of
Master of Science
Department of Mechanical Engineering
Industrial Engineering Program

Winnipeg, Manitoba
August 1988

ISBN 0-315-48093-9

KBGT
A KNOWLEDGE-BASED SYSTEM FOR GROUP TECHNOLOGY

BY

WADOOD M. IBRAHIM

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

To my mother,

Widad Taha Ashgah Al-Azzawi

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The author wishes to express his gratitude to Dr. Andrew Kusiak, Professor of Industrial Engineering, for his guidance and encouragement throughout the project. Acknowledgement is also due to Dr. Soad Taha Ashgah, The author's aunt for her support. Finally, the author would like to extend his thanks to his family for thier support and to Elin Hallgrimson, the author's fiance, for her help in a number of ways during the preparation for this thesis.

FIGURES

TABLES

ABSTRACT

In this thesis a knowledge-based system (KBGT) for solving the group technology problem is presented. The formulation of the group technology problem involves constraints related to machine capacity, material handling system capability, and machine cell dimension. The KBGT has been developed for an automated manufacturing environment. It takes advantage of the developments in expert systems and optimization. Two basic components of the knowledge-based system, namely the knowledge-based subsystem and the heuristic clustering algorithm are discussed. Each partial solution generated by the clustering algorithm is evaluated for feasibility by the knowledge-based subsystem which modifies search directions of the algorithm. The KBGT is illustrated with numerical examples. Application of KBGT to industrial group technology problems is also presented.

CHAPTER ONE

INTRODUCTION

Group technology (GT) is a decomposition approach to manufacturing that takes advantage of the similarity of operations to be performed on different parts. Using GT, parts that require similar operations are grouped into part families. The machines that process each part family are grouped into machine cells.

Application of GT in manufacturing has the following advantages (Kusiak and Chow, 1988):

- reduced production lead time
- reduced variety of process plans
- reduced setup time
- reduced part shortages
- reduced work-in-process
- reduced rework and scrap material
- reduced raw material stocks
- reduced labour
- reduced production floor space
- reduced tooling
- reduced order time delivery
- reduced paper work
- increased reliability of cost estimates
- improved staff relations

The above advantages were justified and discussed in detail in Durie (1970), Edwards (1971), Fazakerlay (1974), Holtz (1978a, 1978b), Schaffer (1981), Ingram (1982), Hyer and King (1984), and Kusiak and

Chow (1988). The group technology concept can be applied in a number of manufacturing areas such as: product design, process planning, programming, machining, inventory and management.

The thesis is divided into five chapters. In Chapter two the existing approaches to modeling and solving the group technology problem are reviewed, namely: classification approach and cluster analysis approach. Moreover, knowledge-based systems are also introduced.

In Chapter three a formulation of the group technology problem in automated manufacturing environment is presented. To solve the GT problem a Knowledge-Based System for Group Technology (KBGT) that has been developed is discussed in detail. The KBGT is based on the tandem system architecture proposed by Kusiak (1987).

In Chapter four, the performance of KBGT is demonstrated. First, an illustrative example of the operation of KBGT is presented. Then, an application of KBGT to group technology problems from the literature and two industrial case studies are also discussed.

Conclusions are drawn in Chapter five.

CHAPTER TWO

LITERATURE REVIEW

In this chapter, a review of the existing Group Technology (GT) literature is presented. Two approaches to modeling and solving Group Technology problems are discussed. In the first section, the classification approach is presented, where three methods of classification are outlined. They are the visual method, the nomenclature/function method, and finally the coding method. The following three basic code structures are used in the coding method: hierarchical, chained, and hybrid. In the second section, the cluster analysis approach is discussed. Three basic formulations of the clustering problem in group technology are presented, namely, the matrix formulation, the mathematical programming formulation, and other formulations. The last section of this chapter introduces basic concepts of knowledge-based systems. The tandem knowledge-based system architecture proposed by Kusiak (1987) is also presented.

2.1 CLASSIFICATION APPROACH

The term classification is used to refer to grouping parts into part families based on similarities and/or dissimilarities of predetermined part characteristics (Eckert, 1975, Ingram, 1982, Ham 1985). There are three methods of the classification approach:

- visual method
- nomenclature/function method
- coding method.

2.1.1 Visual Method

The visual method is a semi-systematic procedure where parts are grouped based on similarity of geometric shape as shown in Figure 1, where 11 parts have been grouped into four part families.

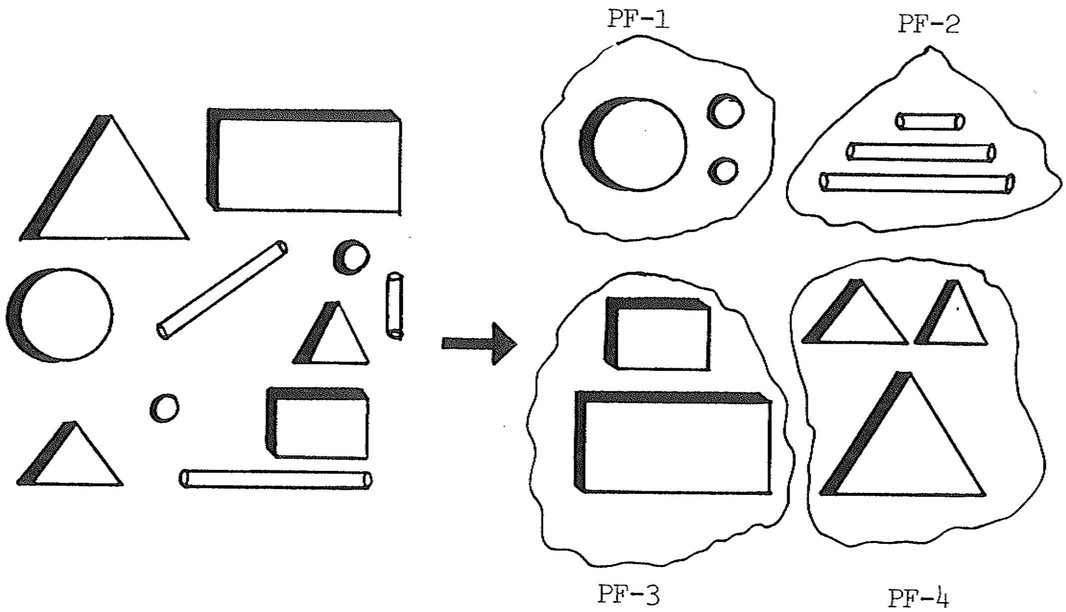


Figure 1. Grouping of parts using a visual method

2.1.2 Nomenclature/Function Method

This method is also a semi-systematic procedure where parts are grouped based on given names that designate their functions (Ham, 1985). Both the visual and nomenclature/function methods are manual procedures and are dependent on personal preference. Therefore, these two methods are applicable in cases where the number of parts is rather limited.

2.1.3 Coding Method

In the coding method each part is assigned a code that consists of numbers, letters, or a combination of both, based on predetermined part characteristics. The most common part characteristics used are:

- geometric shape
- complexity
- operational processes
- dimensions
- type of material
- shape of raw material
- required tolerance.

The above list may be extended to include additional characteristics dependent on the type of parts coded.

In the literature a system that uses a coding method is called a classification and coding system. The currently available classification and coding systems differ with respect to the depth of coverage of part characteristics mentioned above. For example, a classification and coding system may provide more information on the shape and dimension of a part whereas another may emphasize more on the tolerance of a part.

There are three basic types of code structures:

- hierarchical
- chained
- hybrid.

2.1.3.1 Hierarchical Code

Hierarchical codes have been used in areas other than manufacturing. For example, in biology, a lineage chart take this form and it is usually called a family tree (Eckert, 1975). Another form is a company's organizational chart.

To obtain a hierarchical code, characteristics of each part are matched with the characteristics corresponding to each node of the tree. For example, the sample code 222 indicated by the bold lines shown in Figure 2. Since this structure is hierarchical the meaning of each character in a code is dependent on the meaning of the character preceding it. In order to fully interpret a part hierarchical code, all of its characters have to be known. For a given part the length of its hierarchical code is rather short compared to other coding systems (Ingram, 1982).

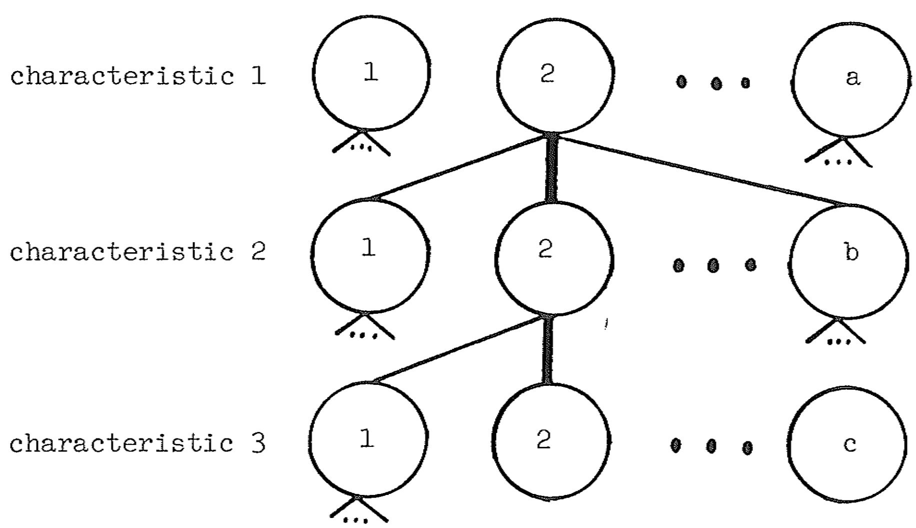


Figure 2. A tree structure of a hierarchical code

2.1.3.2 Chained Code

A chained code, also known as a polycode or feature code, is constructed in such a way that each position denotes a part's feature/characteristic. A code in a chained code system is based on a selection of digits and/or letters through a number of multiple-choice queries. To collect sufficient information describing a part, the user scans a rather large number of queries. Therefore, a chained code is typically long often more than thirty characters (Ingram, 1982). Since a chained code does not have a hierarchical structure, the meaning of a character is not dependent upon the preceding character. In practice though not all characters are totally independent (Eckert, 1975). A chained coding system and a sample code are illustrated in Figure 3. A sample code 121 is generated by selecting one digit from each of the multiple choices.

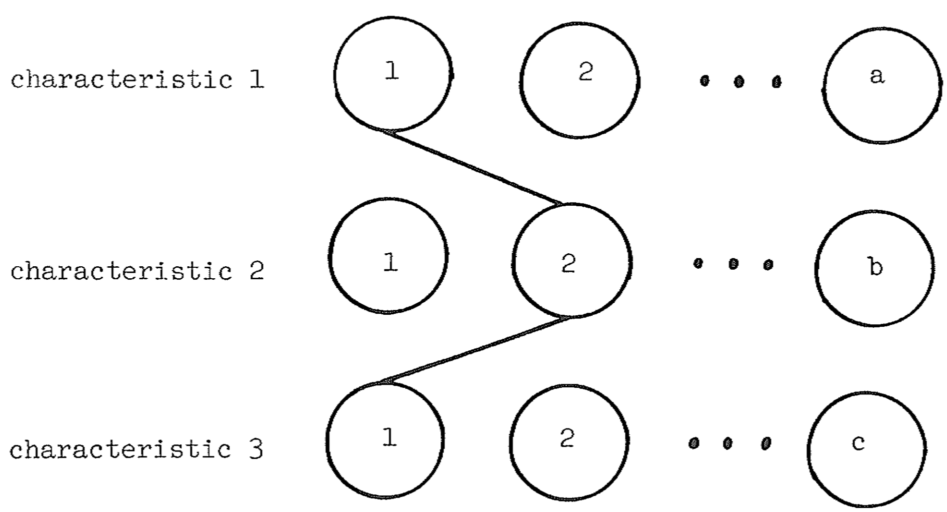


Figure 3. Structure of a chained code

2.1.3.3 Hybrid Code

The structure of a hybrid coding system is a combination of the hierarchical and chained code structures. Most current classification and coding systems employ the hybrid code structure, because it has the advantages of both structures (Schaffer, 1981). A typical structure of a hybrid system is shown in Figure 4.

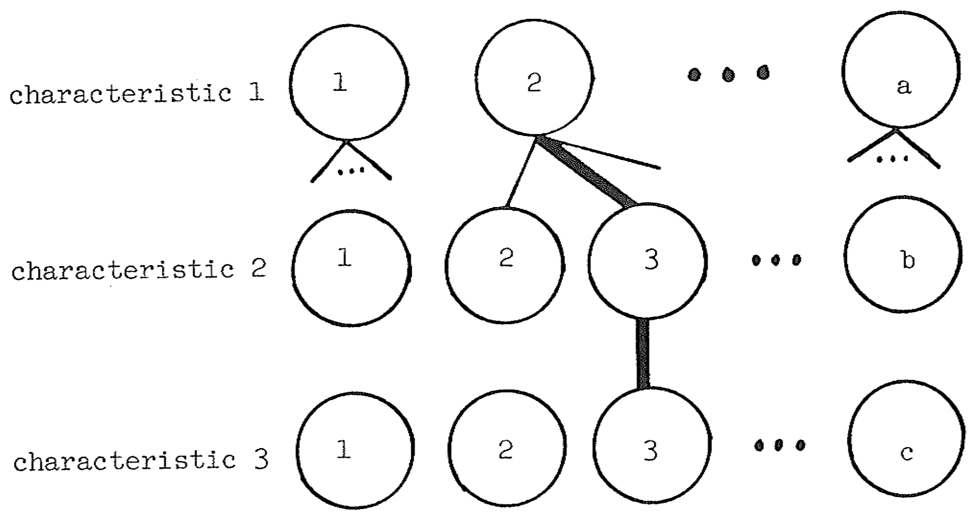


Figure 4. Structure of a hybrid code

The first two characters of the code in Figure 4 have the form of a hierarchical structure that divides parts into subgroups and the remaining characters are constituted by the chained code (Eckert, 1975).

Ham (1985) has listed 44 classification and coding systems currently in use in industry. A company that intends to employ a coding and classification system has to select and modify an existing coding system or to develop a new one so that it suits its needs. Ingram (1982) and Dunlap and Hirlinger (1983) presented several classification principles that should be considered in developing a classification and coding system. Some of the widely applied systems are (Kusiak, 1985):

1) BRISH-BIRN (United Kingdom) - based on four to six-digit primary code and number of secondary digits.
2) DCLASS (USA) - a software based system without any fixed code structure.
3) CODE-MDSI (USA) - an eight digit code.
4) MICLASS (The Netherlands) - a twelve to thirty two digit code.
5) OPTIZ (West Germany) - a five digit primary code with an extendable four digit secondary code.
6) TEKLA (Norway) - a twelve digit code.

The OPTIZ classification and coding system (Optiz and Wiendahl, 1975) is discussed below. The code in a hybrid system consists of nine digits. The five most significant digits are called a primary code, and the remaining four digits are called a supplementary code. The most significant digit is a part class code that is used to divide parts into rotational and non-rotational parts. For example, a value of three or four of the most significant digits indicates the deviation of L/D ratio, where L is the length of a part and D is the diameter. The second and third most significant digits are for a part's external shape and form. The type of surface machine and teeth formation of a part are represented by the fourth and fifth most significant digits.

The supplementary code indicates the size, material, original material shape and accuracy of a part. In the OPTIZ system, the most significant digits are used to specify the detailed structure of a part (Gallagher and Knight, 1973). One of the advantages of the OPTIZ system is that the code can be extended to include supplementary digits. This feature makes the system applicable to different companies. Moreover, the extension allows a more detailed description of a part and its process plan which makes the new system suitable for computerization (Billo et al., 1987).

2.2. CLUSTER ANALYSIS APPROACH

Cluster analysis is concerned with the separation of numerical data sets into unique clusters of data (Gongaware and Ham, 1981). It has been applied in many areas such as automated retrieval and storage systems (Hwang et al., 1988), biology (Everitt, 1980), data recognition (McCormick, et al., 1972), medicine (Klastorin, 1982), pattern recognition (Tou and Gonzalez, 1974), production flow analysis (Burbidge, 1971; King, 1980), task selection (Nagai, et al., 1980), automated manufacturing systems (Kusiak, 1985; Kumar, et al., 1986), and expert systems (Cheng and Fu, 1985). Waghodar and Sahu (1983) listed more than 400 references related to cluster analysis and group technology.

The application of cluster analysis in group technology is to group parts into part families and machines into machine cells. There are three types of formulations of the clustering problem:

- matrix formulation
- mathematical programming formulation
- formulations based on other methods.

2.2.1 Matrix Formulation

In the literature there are two matrix formulations:

- standard matrix formulation
- generalized matrix formulation.

2.2.1.1 Standard Matrix Formulation

In the standard matrix formulation a 0-1 machine-part incidence matrix $[a_{ij}]$ is constructed from production process data usually listed in operation sheets. The machine-part incidence matrix $[a_{ij}]$ consists of 0,1 entries, where an entry 1 (0) indicates that part j is (not) to be processed on machine i. Typically, when an initial machine-part incidence matrix $[a_{ij}]$ is constructed, clusters of machines and parts are not visible. Clustering algorithms are used to transform an initial machine-part incidence matrix into a more structured form, possibly a a block diagonal form. The clustering concept is illustrated in example 1.

Example 1

Consider the machine-part incidence matrix (1)

PART NUMBER

$$[a_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} & & & 1 & \\ 1 & & 1 & & 1 \\ & & 1 & & 1 \\ & 1 & & 1 & \end{bmatrix} \end{array} \quad \begin{array}{l} \text{MACHINE} \\ \text{NUMBER} \end{array} \qquad (1)$$

$$\text{where } a_{ij} = \begin{cases} 1 & \text{if machine i is used to process part j} \\ 0 & \text{otherwise} \end{cases}$$

Rearranging rows and columns in (1) results in matrix (2).

$$
\begin{array}{ll}
 & \quad \text{PF-1} \qquad \text{PF-2} \\
 & \begin{array}{ccccc} 1 & 5 & 3 & 2 & 4 \end{array} \\
\begin{array}{r} \text{MC-1} \left\{ \begin{array}{c} 2 \\ 3 \end{array} \right. \\ \text{MC-2} \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right. \end{array} & \left[\begin{array}{ccccc} 1 & 1 & 1 & & \\ & 1 & 1 & & \\ & & & 1 & 1 \\ & & & & 1 \end{array} \right]
\end{array} \qquad (2)
$$

Two machine cells MC-1 = {2,3}, MC-2 = {4,1} and two corresponding part families PF-1 = {1,5,3}, PF-2 = {2,4} are visible in matrix (2). There is no flow of parts between the two machine cells. One has to realize, that it is virtually impossible to design a cellular manufacturing system without any interaction among machine cells. A typical situation occuring in practice is illustrated in matrix (3).

$$
\begin{array}{lll}
 & \quad \text{PART NUMBER} & \\
 & \begin{array}{ccccc} 1 & 5 & 3 & 2 & 4 \end{array} & \\
\begin{array}{r} \text{MC-1} \left\{ \begin{array}{c} 2 \\ 3 \end{array} \right. \\ \text{MC-2} \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right. \end{array} & \left[\begin{array}{ccccc} 1 & 1 & 1 & & \\ & 1 & & & \\ & & 1 & 1 & 1 \\ & & & & 1 \end{array} \right] & \begin{array}{l} \text{MACHINE} \\ \text{NUMBER} \end{array}
\end{array} \qquad (3)
$$

Part 3 is to be manufactured in both machine cells.

Over the past fifteen years, the following approaches have been developed to solve the matrix formulation of the group technology problem:

- production flow analysis
- similarity coefficient methods
- sorting algorithms
- bond energy algorithm
- cost based methods
- cluster identification algorithm.

- Production Flow Analysis

The production flow analysis (PFA) was introduced by Burbidge (1971). The PFA is one of the earliest analytical methods for implementing group technology in manufacturing systems. The PFA consists of three levels of analysis:

- factory flow analysis
- group analysis
- line analysis.

In factory flow analysis level a machine-part incidence matrix is constructed based on analysis of part flows which may be obtained from operation sheets. In the group analysis level an attempt is made to identify machine cells and part families by rearranging rows and columns of the machine-part incidence matrix. This level is primarily manual and dependent on subjective evaluation. A great deal of research has been conducted in order to make this level systematic and suitable for computerization. The line analysis uses the generated clusters from the group analysis to determine machine layout, identify bottleneck machines and analyze flow patterns on the shop floor.

The PFA is primarily manual and lacks a clear-cut methodology, especially, in group formation (Oba et al., 1987). It is, therefore, not suitable for computerization.

Dekleva and Menart (1987) proposed a procedure that represents an extension to PFA. The proposed procedure deals with the first and second level of PFA and consists of three stages:

- identification of part families using clustering analysis
- identification of groups of machines using modified machine-part incidence matrix
- test of fitness.

El-Essaway and Torrance (1972) presented component flow analysis (CFA) that is similar to PFA. King and Nakornchai (1982) pointed out the two differences between CFA and PFA. The CFA first partitions the problem, whereas PFA does not. The second difference relates to the manner in which the cells are formed in the two methods.

• Similarity of Coefficient Methods

A similarity coefficient method attempts to make PFA a systematic procedure. In these methods a similarity/dissimilarity value for each pair of data elements is calculated. These values are then stored in a two-dimensional similarity array. This array is used as input to a clustering algorithm to group the data elements. The similarity values usually represent a distance between data elements. A common practice is to minimize the sum of distances of grouped elements from the calculated centroids of their respective clusters or to maximize the distance between cluster centroids (Congaware and Ham, 1981). The output is in the form of a dendogram. Clusters are generated based on a threshold value of the similarity coefficient.

McAuley (1972) introduced the Single Linkage Cluster Analysis (SLCA) which uses the similarity coefficient measure between two machines as the number of parts processed on both machines divided by the number of parts processed on either of the two machines. One of the major drawbacks of SLCA is that it fails to recognize the chaining problem resulting from the duplication of bottleneck machines (King and Nakornchai, 1982).

In order to overcome the chaining problem Seifoddini and Wolfe (1986) developed the Average Linkage Clustering (ALC) algorithm. They define the similarity coefficient between any two clusters as an average of the similarity coefficient between all members of the two clusters. The grouping obtained is dependent on the similarity threshold value used. Therefore, the SLCA and ALC algorithms generate a set of alternative solutions rather than a unique solution. Seifoddini and Wolfe (1987) suggest a threshold value based on material handling cost. Seifoddini (1986) studied the problem of improper machine assignment in machine-part grouping in group technology. The machines involved are bottleneck machines. He suggested that all bottleneck machines be reexamined after machine cells are formed and be reassigned wherever necessary in order to reduce the number of inter-cellular moves.

De Witte (1980) developed a clustering algorithm that allows some machines to be available in more than one machine cell. He divided all available machines into:

1) primary machines, where only one copy of each machine is available

2) secondary machines, where only a few copies of each machine are available

3) tertiary machines, where sufficient number of copies of each are available.

In order to analyze the interdependence between these machines, De Witte (1980) suggested three similarity coefficients:

1) absolute similarity coefficient

2) mutual similarity coefficient

3) single similarity coefficient.

To obtain the best clustering results, first start assigning machines with the absolute coefficient, the second, and then use the third coefficient to allocate the remaining unassigned machines.

- Sorting Algorithms

Many researchers have studied cluster analysis algorithms that are based on sorting rows and columns of the machine-part incidence matrix. One of them, the Rank Order Clustering (ROC) algorithm was developed by King (1980). This algorithm can be considered as an attempt to computerize the group analysis level of production flow analysis. The ROC algorithm is as follows:

STEP 1 Read each row of the machine-part incidence matrix as a binary word.

STEP 2 Sort the binary words in decreasing order.

STEP 3 If the row order of the current machine-part incidence

matrix is the same as the order of the corresponding binary words generated in Step 2, then go to Step 7 ;
else, rearrange rows of the matrix according to the order generated in Step 2 and go to Step 4.

STEP 4 Read each column of the matrix as a binary word (the most significant digit is the one at the top row).
Sort the binary words in decreasing order.

STEP 5 If the column order of the current matrix is the same as the order of the corresponding binary words generated in Step 4, then go to Step 7 ;
else, go to Step 6.

STEP 6 Rearrange the machine-part incidence matrix starting with the first column by rearranging the columns in decreasing order and go to Step 1.

STEP 7 STOP.

King and Nakornchai (1982) developed the ROC2 algorithm which is an extension of the original ROC algorithm. Chandrasekharan and Rajagopalan (1986) studied the deficiencies of the ROC algorithm and developed MODROC algorithm, that incorporates the following two methods to improve the performance of the ROC algorithm:

i. "block and slice" method
ii. hierarchical method

Another sorting algorithm that was studied by many researchers is the Direct Clustering Algorithm (DCA) which was developed by Chan and Milner (1982). The DCA incorporates the following steps:

STEP 1 Count the total number of '1's in each row and column of the machine-part incidence matrix.

STEP 2 arrange the machine-part incidence matrix with rows in increasing order of the total number of '1's and columns with decreasing order of the total number of '1's.

STEP 3 For each column of the matrix, starting with the first column, rearrange the rows, that have '1' entries in the column considered, to the top of the matrix.

STEP 4 If the matrix generated in Step 3 is the same as the one immediately preceding, then go to Step 7 else, go to Step 5.

STEP 5 For each row of the matrix, starting with the first row, rearrange the columns, that have '1' entries in the row considered, to the left-most position of the matrix.

STEP 6 If the matrix generated in Step 5 is the same as the one immediately preceding, then go to Step 7 else, go to Step 2.

STEP 7 STOP.

• Bond Energy Algorithm

The Bond Energy Algorithm (BEA) is an interchange clustering algorithm developed by McCormick et al. (1972). The BEA attempts to transform the machine-part incidence matrix to a block diagonal form by maximizing the measure of effectiveness which is defined as follows:

$$ME = 1/2 \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \, [a_{i,j+1} + a_{i,j-1} + a_{i+1,j} + a_{i-1,j}]$$

The BEA consists of the following steps:

STEP 1 Set i=1 ;

Select any column of the machine-part incidence matrix.

STEP 2 Move the remaining n-i columns, one at a time, to the i+1 positions, and calculate each column's contribution to the ME.

Place the column that gives the largest incremental contribution to the ME in its best location.

Increment i by 1 and repeat Step 2 until i=n.

STEP 3 Repeat the same procedures in Step 2 for the rows.

A clustering algorithm based on the BEA and the Shortest Spanning Path (SPP) algorithm, was developed by Slagle et al. (1975). Their concept was then extended by Bhat and Haupt (1976). They developed an algorithm where the matching between any two rows/columns of the machine-part incidence matrix is calculated as follows:

$$m = \sum_{k=1}^{n} |a_{ik} - a_{jk}|$$

The Bhat and Haupt's algorithm maximizes the total sum of matchings between rows and columns of the matrix.

- Cost-Based Method

Askin and Subramanian (1987) developed a clustering algorithm that considers the following manufacturing costs:

1) fixed and variable machining
2) setup
3) production cycle inventory

4) work-in-process inventory

5) material handling.

The algorithm consists of three stages. In the first stage, parts are classified using a coding system. In the second stage, an attempt to develop a feasible grouping between parts based on the manufacturing costs is performed. In stage three, the actual layout among a group of machine cells is analyzed.

• Cluster Identification Algorithm

Kusiak and Chow (1987a) developed the Cluster Identification (CI) algorithm. The CI algorithm decomposes the machine-part incidence matrix into separable submatrices provided that they exist. The cluster identification algorithm has a relatively low computational time complexity of O(2mn).

In practice a machine-part incidence matrix does not decompose into separable submatrices, therefore, the cost analysis algorithm was developed (Kusiak and Chow, 1987b). In the cost analysis algorithm a cost c_j is associated with each column/part of the machine-part incidence matrix. The cost c_j could be:

1) subcontracting cost

2) part flow rate.

The CI algorithm seems to be the most efficient algorithm in the literature. It has a relatively low computational time complexity of O(2mn).

2.1.1.2 Generalized Matrix Formulation

This formulation is an extension of the standard matrix formulation. The extension represents qualitative parameters and constraints (Kusiak, 1986). The parameters could be part production cost, part machining time and frequency of trips required to handle a part by a robot. The constraints usually represent production constraints such as maximum number of machines in a machine cell, maximum machining time available on a machine and maximum frequency of trips that can be handled by a robot.

2.2.2 Mathematical Programming Formulations

There are a number of mathematical programming formulations that have been developed to model the group technology problem. Most of these formulations use a distance measure d_{ij} between parts i and j. The distance measure d_{ij} is a real-valued symmetric function obeying the three following axioms (Fu, 1980):

- reflexivity $d_{ii} = 0$
- symmetry $d_{ij} = d_{ji}$
- triangle inequality $d_{iq} \leq d_{ip} + d_{pq}$

The distance measure, also known as dissimilarity measure, is defined depending on the application considered. The most commonly applied distance measures are as follows (Kusiak, 1985):

1) Minkowski distance:

$$d_{ij} = \left[\sum_{k=1}^{n} |a_{ik} - a_{jk}|^{r} \right]^{1/r}$$

where: n is the number of parts

r is a positive integer.

These two special cases of Minkowski's measure are widely used:

- absolute distance measure (for r=1)
- Euclidean distance measure (for r=2).

2) weighted Minkowski distance:

$$d_{ij} = [\sum_{k=1}^{n} w_k \mid x_{ik} - x_{jk} \mid^r]^{1/r}$$

Similar to Minkowski's distance measure, there are two special cases:

- weighted absolute distance measure (for r=1)
- weighted Euclidean distance measure (for r=2).

3) Hamming distance:

$$d_{ij} = \sum_{k=1}^{n} (x_{ik}, x_{jk})$$

where: $$(x_{ik}, x_{jk}) = \begin{matrix} 1 & \text{if } a_{ik} \neq x_{jk} \\ 0 & \text{otherwise.} \end{matrix}$$

In the following discussion three models of the mathematical programming formulation in group technology are presented:

- quadratic programming model
- p-median model
- generalized p-median model.

2.2.2.1 Quadratic programming model

Kusiak et al. (1986) developed a quadratic mathematical programming formulation in group technology. They used the following parameters and variables:

m number of machines

n number of parts

p number of part families

t_j number of parts in family j

$$a_{ij} = \begin{cases} 1 & \text{if part i can be processed on machine j} \\ 0 & \text{otherwise} \end{cases}$$

s_{ij} similarity between part i and part j

$(s_{ij} \geq 0,\ i,j=1,2,\ldots,n,\quad s_{ij}=0,\ i=j=1,2\ldots,n)$

$$s_{ij} = \sum_{k=1}^{m} d(a_{ik},a_{jk})$$

where: $$d(a_{ik},a_{jk}) = \begin{cases} 1 & \text{if } a_{ik} = a_{jk} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if part i belongs to part family j} \\ 0 & \text{otherwise} \end{cases}$$

The 0-1 quadratic programming model is as follows:

$$\max \sum_{l=1}^{p} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} s_{ij} x_{il} x_{jl} \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{p} x_{ij} = 1, \quad i=1,\ldots,p \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = t_j, \quad j=1,\ldots,p \tag{3}$$

$$x_{ij} = 0,1 \quad , \quad i=1,\ldots,n, \quad j=1,\ldots,p \tag{4}$$

Constraint (2) ensures that each part is assigned to one family. Constraint (3) specifies the required number of parts in each part family. Constraint (4) is for integrality. In this model the number of parts in each part family is restricted and to be determined a priori. Since this model is computationally complex the p-median model was developed as an approximation to this model (Kusiak and Heragu, 1987). The p-median model is discussed later.

Kumar et al. (1986) have developed the following 0-1 quadratic formulation of the group technology problem:

$$\max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{p=1}^{k} a_{ij} x_{ip} x_{jp} \tag{5}$$

$$\text{s.t.} \quad \sum_{j=1}^{k} x_{ij} = 1, \qquad i=1,\dots,n \tag{6}$$

$$1 \le \sum_{i=1}^{n} x_{ij} \le u, \qquad j=1,\dots,k \tag{7}$$

$$x_{ij} = 0,1, \qquad i,j=1,\dots,n \tag{8}$$

where: a_{ij} the volume of part i that has to be processed on machine j, or profit associated with parts i and j.

Constraint (6) ensures that each part is assigned to only one part family. Constraint (7) is to limit the number of parts in each part family, and constraint (8) is for integrality. Kumar et al. (1986) have developed a two phase polynomially bounded heuristic algorithm.

2.2.2.2 p-Median Model

The p-median model is used to group n parts into p part families such that the total sum of distances between any two parts i and j is maximized (Kusiak, 1985). The definition of the p-median model follows Mulvey and Crowder (1979):

m number of machines

n number of parts

p number of part families

$$x_{ij} = \begin{cases} 1 & \text{if part } i \text{ belong to part family } j \\ 0 & \text{otherwise} \end{cases}$$

d_{ij} distance measure between parts i and j

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{9}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad , \quad i=1,\ldots,n \tag{10}$$

$$\sum_{j=1}^{n} x_{jj} = p \tag{11}$$

$$x_{ij} \leq x_{jj} \qquad i,j=1,\ldots,n \tag{12}$$

$$x_{ij} = 0,1 \quad , \quad i,j=1,\ldots,n \tag{13}$$

Constraint (10) ensures that each part belongs to exactly one part family. Constraint (11) specifies the required number of part families. Constraint (12) ensures that part i belongs to part family j only when it is formed. Constraint (13) is to ensure integrality.
In the p-median model p, the number of part families, is specified a priori.

2.2.2.3 Generalized p-Median Model

In the p-median model and in most group technology models, the assumption is used that there exists only one process plan for each part. In automated manufacturing systems there is usually more than one process plan for each part that can be generated by Computer-Aided Process Planning (CAPP) systems. Kusiak (1987) presented the generalized p-median model. This model permits more than one process plan to be considered for each part. However, in the final clustered machine-part incidence matrix only one process plan for each part is selected. The objective of the model is to maximize the total sum of distance measures between parts.

The following notation is used to formulate the model (Kusiak, 1987):

n number of parts

q number of process plans

F_k number of process plans for part k, $k=1,2,\ldots,n$

p required number of process families

d_{ij} distance measure between process plans i and j
($d_{ij} = -\infty$ for all i and j in F_k, $k=1,2,\ldots,n$, $d_{ij}=0$,
$i,j=1,2,\ldots,q$, $d_{ij} \geq 0$ for all other i and j)

$$x_{ij} = \begin{cases} 1 & \text{if process plan } i \text{ belongs to process family } j \\ 0 & \text{otherwise} \end{cases}$$

The objective function is :

$$\max \sum_{i=1}^{q} \sum_{j=1}^{q} d_{ij} x_{ij} \qquad (14)$$

$$\text{s.t.} \sum_{i \in F_k} \sum_{j=1}^{q} x_{ij} = 1 \quad , \quad k=1,2,\ldots,n \qquad (15)$$

$$\sum_{j=1}^{q} x_{ij} = p \qquad i=1,\ldots,q \qquad (16)$$

$$x_{ij} \leq x_{jj}, \qquad i,j=1,\ldots,q \qquad (17)$$

$$x_{ij} = 0,1 \qquad i,j=1,\ldots,q \qquad (18)$$

Constraint (15) ensures that only one process plan for each part is considered. Constraint (16) specifies the number of part families required. Constraint (17) ensures that part i is included in part family j only when this part family is formed. Constraint (18) is to ensure integrality.

Note that in this model the number of machine cells is determined a priori. The generalized p-median model increases the chances of obtaining a diagonally structured incidence matrix.

2.2.3 Other Modeling Approaches

There are two other modeling approaches that have been used to formulate the group technology problem, namely a graph theoretic approach and a set theoretic approach. The graph theoretic approach of Rajagopalan and Batra (1975) uses cliques. The machines and the Jaccard similarity coefficients are represented by the vertices and the arcs of the graph, respectively. The number of cliques normally increases exponentially with the increase of the number of machines (King and Nakornchai, 1982). Therefore, this approach is applicable only when the number of machines is rather small.

In the graph formulation the bottleneck parts/machines in a graph is a non-trivial task. Lee et al. (1982) developed a heuristic algorithm in order to detect the bottleneck parts/machines. This algorithm was extended by Vannelli and Kumar (1986).

The second analytical method used to formulate the group technology problem is a set theoretic method called polyhedral dynamics, also known as q-analysis. While polyhedral dynamics is a branch of set theory dealing with the topological relationship between finite sets, q-analysis is used to study polyhedral dynamics. The two terms q-analysis and polyhedral dynamics are often used interchangeably (Robinson and Duckstein, 1986). Robinson and Duckstein (1986) applied q-analysis to group formation of machine cells and part families. They pointed out that the mathematical theory behind polyhedral dynamics is rather complex.

2.3 KNOWLEDGE-BASED SYSTEMS

In the past decade, the principles and methodologies of Artificial Intelligence (AI) have been applied to a number of areas. Knowledge-based systems are perhaps the most widely used AI application. A knowledge-based system is a computer program that uses explicit knowledge of a domain to solve problems in that domain.

There is a fundamental difference between a knowledge-based system and a standard computer program. In a conventional computer program, the knowledge of how to solve a problem is scattered within the program code which solves the problem (Miller, 1986). In a knowledge-based system the knowledge is separated from the control component of the program. Therefore, modifications and additions to the knowledge can be performed without changing the control component (Miller, 1986). Most knowledge-based systems are stand-alone knowledge-based system, as shown in Figure 5 (Kusiak, 1987). Kusiak (1987) proposed an architecture for a knowledge-based system called a tandem architecture (see Figure 6). In the tandem architecture a knowledge-based system is working jointly with a model and an algorithm. The tandem knowledge-based system is more efficient than the stand-alone system when the problems involve quantitative data because the model and the algorithm component deals with the quantitative data efficiently.

A knowledge-based system consists of three basic components. The first component is a knowledge base which contains domain-specific knowledge of how to solve problems. The second component is a general purpose control component called the inference engine (Waterman 1986).

The third component is a data base that contains facts about the problem being solved.

The reader interested in the principles of knowledge-based systems may refer to Rich (1983), Winston (1984), Nilsson (1980), Charniak and McDermott(1985), Waterman (1986), Jackson (1986), and Hayes-Roth et al. (1983).

Since the area of manufacturing is knowledge intensive, especially in the design stage, and strongly dependent on manufacture know-how, knowledge-based systems are well suited for solving manufacturing problems. Kempf (1985) discussed the computational complexities of manufacturing problems, for example part design and process planning. He pointed out that the application of AI principles and methodologies is one of the most realistic and practical approaches for dealing with manufacturing problems. The implications of using artificial intelligence for computer integrated manufacturing is presented in Kusiak (1988). Heragu and Kusiak (1987) presented an analysis of knowledge-based systems in manufacturing design. O'Conner presented Intelligent Management Assistant for Computer System manufacturing (IMACS), which is a knowledge-based system that assists in the management of the manufacturing process (Waterman, 1986). IMACS helps with the management of paper work, inventory, and capacity planning. Intelligent Scheduling and Information Systems (ISIS) was studied by Fox and Smith (1984). ISIS generates factory job shop schedules and can also assist plant schedulers to maintain schedule consistency and identify decisions that result in unsatisfied constraints. For more information related to knowledge-based systems in manufacturing refer to Gains (1987), Newman (1987), Faught (1986), and Kumara et al. (1986).

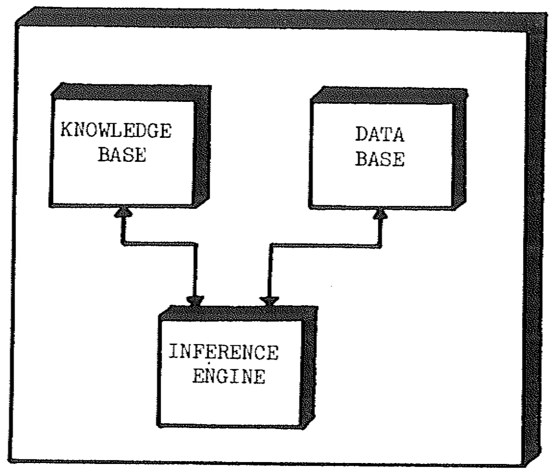


Figure 5. A stand-alone knowledge-based system

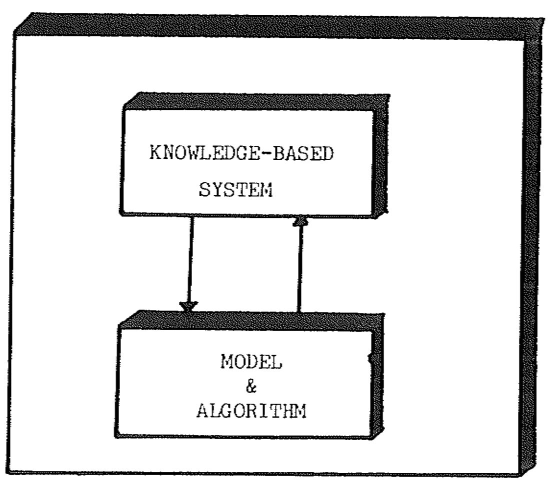


Figure 6. A tandem knowledge-based system

CHAPTER THREE

A KNOWLEDGE-BASED SYSTEM FOR GROUP TECHNOLOGY (KBGT)

3.1 FORMULATION OF THE GROUPING PROBLEM IN AUTOMATED MANUFACTURING SYSTEMS

A typical approach to cellular manufacturing is to group machines and parts based on the binary machine-part incidence matrix, usually without any constraints. Some authors, for example Stanfel (1982), Kumar et al. (1986), Kusiak (1985) have restricted the size of machine cells and part families.

The approach presented in this thesis considers two formulations of the grouping problem.

The first formulation is a generalization of the grouping problem presented in the literature. Rather than the binary matrix $[a_{ij}]$, the matrix $[t_{ij}]$, where $t_{ij} \geq 0$ is the processing time of part j on machine i is considered. This formulation involves also some constraints, which are typical for an automated manufacturing environment.

The grouping problem in automated manufacturing systems can be loosely formulated as follows (Kusiak, 1987):

Determine machine cells; for each machine cell, select a part family consisting of parts with the minimum sum of subcontracting costs and select a suitable material handling carrier with the minimum corresponding cost subject to the following constraints:

Constraint C1 : processing time available at each machine is not exceeded

Constraint C2 : upper limit on the frequency of trips of material handling carriers for each machine cell is not exceeded

Constraint C3 : number of machines in each machine cell does not exceed its upper limit or alternatively the dimension (for example the length) of each machine cell is not exceeded.

The above formulation of the GT problem is not only computationally complex, but also involves constraints which are difficult to handle by any algorithm alone. Therefore, to solve the the above problem, a knowledge-based system has been developed (Kusiak and Ibrahim, 1988).

The second formulation considered is a special case of the generalized formulation of the group technology problem. It involves a 0-1 machine-part incidence matrix (see Example 1 in Chapter 1) and constraint C3 presented above.

3.2 STRUCTURE OF THE KNOWLEDGE-BASED SYSTEM (KBGT)

A typical knowledge-based system is developed based on the knowledge elicited from experts. The elicited knowledge is represented using a suitable knowledge representation scheme in a knowledge base. A control strategy, implemented in a form of an inference engine, is employed to search the knowledge base in order to solve a problem. A knowledge-based system of this structure is suitable rather for qualitative problems, but is inefficient for solving problems of quantitative nature.

In this thesis, a tandem knowledge-based system is considered, where a knowledge-based subsystem and an algorithm closely interact (Kusiak, 1988a). The algorithm deals with the quantitative aspects of the problem while the knowledge-based subsystem deals with the qualitative aspects of the problem to be solved.

The knowledge-based system (KBGT) considered has the structure shown in Figure 7:

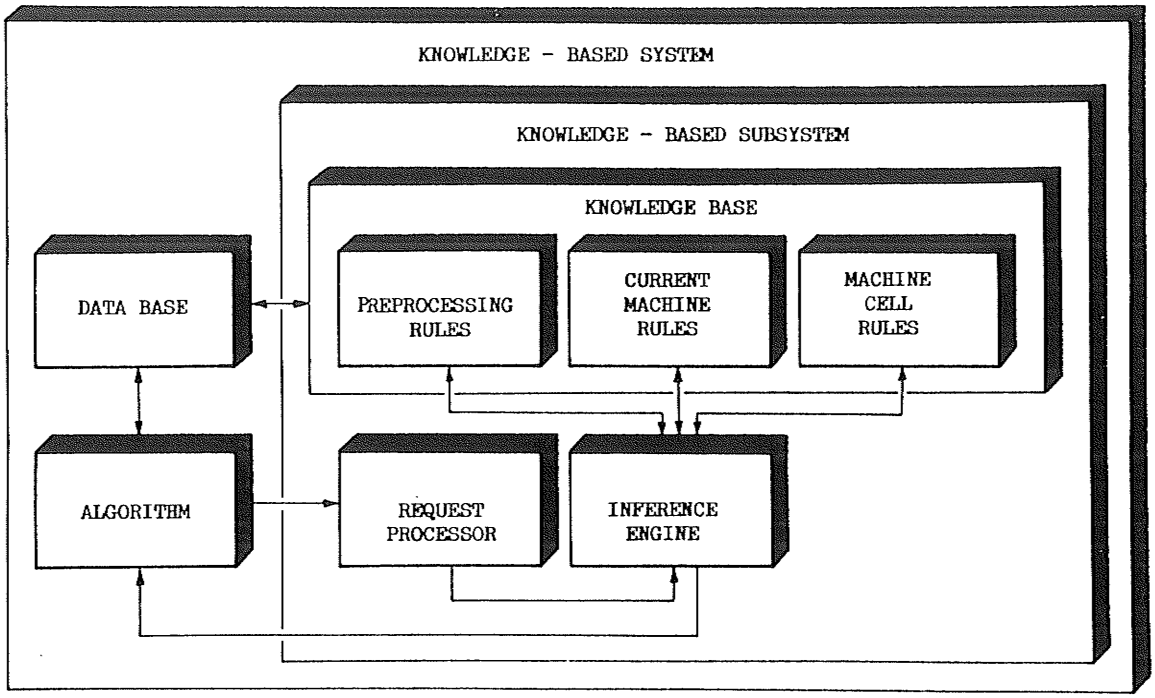


Figure 7. Structure of the knowledge-based system (KBGT)

The KBGT consists of five components :

(1) data base

(2) knowledge base

(3) inference engine

(4) request processor

(5) clustering algorithm.

One of the most tangible advantages of the tandem architecture is a relatively small knowledge base. This is because the computational effort is divided between the inference engine and the algorithm. For the same reason the tandem knowledge-based system is typically faster than the stand-alone system.

The KBGT has been implemented in Common LISP on a SPERRY MICRO IT. LISP, as a programming language for KBGT, has been selected for three reasons:

(1) it facilitates implementation of the declarative knowledge

(2) it facilitates implementation of the procedural knowledge (the clustering algorithm)

(3) it provides flexibility to define and implement the interaction between the algorithm and the knowledge-based subsystem.

3.2.1 Input data

The input data required by KBGT fall into two categories :

(i) machine data

(ii) part data

In addition to the above, depending on the characteristics of the manufacturing system, the following optional data can be provided :

(iii) maximum number of machines in a machine cell

(iv) maximum frequency of trips which can be handled by a material handling carrier (for example, robot or automated guided vehicle, AGV).

3.2.2 Grouping Process

Prior to the begining of the grouping process, KBGT constructs a machine-part incidence matrix based on the input data provided by the user. Next, the KBS initializes in the data base objects representing facts known about the manufacturing system. Then the system forms machine cells and the corresponding part families. Each machine cell is formed by including one machine at a time. A machine is first analyzed by the KBS for the possibility of inclusion in the machine cell. For example, a bottleneck machine (i.e. machine that process parts visiting more than one machine cell) is not included.

Each time a machine cell has been formed the KBS checks whether any of the constraints C1-C3 has been violated and removes all parts violating the constraints.

For a machine cell which has been formed and analyzed by the KBS, the corresponding machines and parts forming a part family are removed from the machine-part incidence matrix. The system does not backtrack in the grouping process, i.e. once a machine cell is formed, the machines included in the cell are not considered for future machine cells. This irrevocable control strategy, as illustrated in Figure 8, is possible due to the nature of the algorithm and the knowledge-based analysis performed by the KBS.

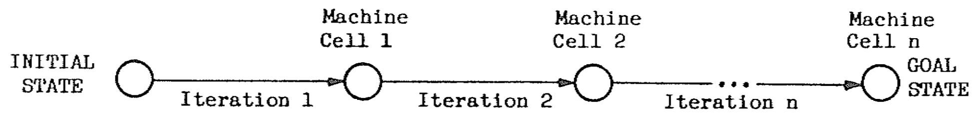


Figure 8. Illustration of the irrevocable control strategy of KBGT

3.2.3 Output Data

At the end of the grouping process, KBGT prints the following data:

(1) machine cells formed

The machine cells formed are listed in the order they have been generated. For each machine cell the following information is provided :

- machine cell number
- list of machine numbers in a machine cell
- part family number
- list of part numbers in a part family
- material handling carrier alternatives, if any.

(2) part waiting list

This list includes parts that were placed on the waiting list due to either :

- overlapping of parts in such a way that prevents grouping, or

- including them in a machine cell would violate one or more constraints.

(3) list of machines not used

The list of machines with all parts removed during the grouping process

(4) list of bottleneck machines

The list of machines that process a relatively large number of parts, which need to be processed on machines belonging to more than one machine cell. These machines should be given special consideration while determining the layout. Each of these machines should be preferably located adjacent to the machine cell that processes the same parts.

(5) maximum number of machines in a cell

This number indicates the maximum number of machines in a machine cell. It has an impact on the machine cells formed, namely if it was too small, it might result in the removal of too many parts. If this number is not supplied by the user then the system groups the machines based on their similarities only.

For any two 0-1 vectors

$$m_i = [a_{i1}, a_{i2}, \ldots, a_{ik}, \ldots, a_{in}]$$

$$m_j = [a_{j1}, a_{j2}, \ldots, a_{jk}, \ldots, a_{jn}]$$

define a similarity measure s_{ij}

$$s_{ij} = \sum_{k=1}^{n} (a_{ik}, a_{jk})$$

where:

$$(a_{ik}, a_{jk}) = \begin{cases} 1 & \text{if element } a_{ik} = a_{jk} \\ 0 & \text{otherwise} \end{cases}$$

n number of parts

In particular, vector m_i may represent parts in machine cell MC-k, and vector m_j may represent parts corresponding to the machine to be selected (see Step 2 in the grouping algorithm in section 6). In this case the distance s_{ij} is regarded as a distance between machine cell MC-k and machine m_j.

3.3 DATA BASE

The global data base contains information about the current problem in a form of objects and frames. It is a non-monotonic data base, since objects are modified by the clustering algorithm and the knowledge-based subsystem (KBS).

The contents of the data base are either provided by the user as input data, or generated by the system. A list of objects and frames in the data base is as follows:

(1) machine frame

Machine frame contains information regarding one machine and is identified by the machine number. It has the following format:

```
(m#i  ((parts ((p#1  p-time)...(p#j  p-time)...))
       (max-process-time   x)
       (multiple  y)))
```

where:

p-time: the time required to process part number p#j on machine m#i (p-time is equal to 0, if processing time is not available)

max-process-time (optional): maximum processing time available on machine m#i, i.e. the capacity of machine m#i

multiple (optional): number of the identical machines available.

(2) part frame

Part frame contains information regarding each part and is identified by the part number. It has the following format :

```
(part#j  ((primary-pp  (m#1 ... m#i...))
           (fr    y)
           (fa    z)))
```

where:

primary-pp: primary process plan for part#j

fr : frequency of trips required by a robot to handle part#j

fa : frequency of trips required by an AGV to handle part#j

(3) matrix-t (machine-part incidence matrix)

The machine-part incidence matrix is constructed by the system based on the input data provided in the following format :

```
( (m#1  ((p#1  p-time) (p#2  p-time)...))
   .
   .
   .
  (m#i  ...                           )
   .
   .
   .                                      )
```

In case when processing times are not available, then by default the matrix is a 0-1 incidence matrix, i.e. p-time is 0.

(4) current machine

A machine that the system has selected for possible inclusion in the machine cell being formed.

(5) list of candidate machines

A list of candidate machines to be included in the machine cell being formed.

(6) list of temporary candidate machines

A list of all machines such that the parts that are processed on the current machine are also processed on one or more of these machines. Moreover, these machines are not on the list of candidate machines.

(7) part waiting list

A list of parts that have been removed from the machine-part incidence matrix.

(8) list of bottleneck machines

A list of all bottleneck machines.

(9) list of temporary bottleneck machines

A list of machines that are considered to be temporary bottleneck machines. These machines may become non-bottleneck machines after some parts have been removed from the machine-part incidence matrix.

(10) list of machines not used

Same as discussed earlier in the subsection on the output of KBGT.

(11) MC-k (machine cell k)

A list of machine numbers in the current machine cell.

(12) PF-k (part family k)

A list of part numbers in the current part family.

3.4 THE KNOWLEDGE-BASED SUBSYSTEM (KBS)

As illustrated in Figure 7, the knowledge-based subsystem (KBS) consists of three components:

(1) knowledge base

(2) inference engine

(3) request processor.

3.4.1 Knowledge base

The knowledge base in KBGT contains production rules, which have been acquired from three experts in group technology and the literature. In the current implementation of KBGT, the knowledge base consists of three classes of production rules :

(a) preprocessing rules

(b) current machine rules

(c) machine cell rules.

The preprocessing rules deal with the initialization of objects in the data base that are not provided by the user. The current machine rules check the appropriateness of a current machine to the machine cell being formed, for example whether the current machine is a bottleneck machine. The machine cell rules deal with each machine cell which has been formed. Machine cell rules check for violation of constraints and remove parts violating them. The structuring of rules into separate classes has two advantages. First, the search for applicable rules is more efficient since the inference engine attempts to fire only rules that are relevant to the current context. Second, the modularity of the rule base makes it more understandable, and easy for modification.

Each rule has the following format:

(rule-number (IF conditions
THEN actions))

The rule number is used for identification by the inference engine. The most significant digit represents a class, and the other two digits represent a rule number in a class. The actions of the rule are carried out, only if the conditions in the IF part are true. Each condition in a rule can have one of the following three forms :

(i) a straightforward checking of values in the data base,

(ii) procedure calls to calculate the values required , or

(iii) a combination of (i) and (ii).

An example of (i) is comparing the size of the current MC-k with the maximum number of machines allowed per machine cell. An example of (ii) is a call of the procedure calculating the number of parts that a machine shares within the current MC-k. The action part consists of procedure calls and/or modifications of values of some objects in the data base.

Combining a rule-based representation paradigm with procedural representation paradigm improves the efficiency of the KBS. Values that can not be obtained directly from the data base are calculated only when necessary.

Sample production rules that have been implemented in KBGT are listed below:

Rule-102 (preprocessing rule)

IF **the intercellular movement (icm) level is not specified**

AND the total number of machines is greater than 50

THEN **set icm to 3**

Note that the intercellular movement level is defined as $icm = \frac{n_1}{n} 100\%$,

where: n_1 is the number of overlapping parts

n is the total number of parts considered.

Alternatively the value of icm can be set by a group technology analyst.

Rule-103 (preprocessing rule)

IF **the maximum number of machines in a machine cell is specified**

THEN **remove from the machine-part incidence matrix all parts that require more machines than the maximum number of machines in a machine cell**

AND place them on the part waiting list.

Rule-201 (current machine rule)

IF no machine has been included in MC-k

AND the number of temporary candidate machines plus the current machine is greater than the maximum number of machines in a machine cell

THEN add the current machine number to the list of temporary bottleneck machines

AND go to Step 1 of the algorithm.

Rule-202 (current machine rule)

IF the maximum number of machines in a machine cell is not specified

AND the similarity between the current machine and MC-k is less than the similarity of the next machine to be selected as the current machine

AND the number of parts that are processed by the current machine and machines in MC-k is less than or equal to the intercellular movement (icm) level

THEN place the parts mentioned above in the part waiting list

AND set the list of temporary candidate machines to empty

AND set the list of candidate machines to empty

AND set the current machine to null

AND go to Step 6 of the algorithm.

Rule-203 (current machine rule)

IF the total of the number of machines in MC-k and machines in the list of temporary candidate machines and machines in the list of candidate machines and the current machine is greater than the maximum number of machines in a machine cell

AND the number of parts, which are processed by the current machine and machines in MC-k, is less than or equal to the intercellular movement (icm) level

THEN place the parts mentioned above in the part waiting list

AND set the list of temporary candidate machines to empty

AND set the current machine to null

and go to Step 5 of the algorithm.

Rule-301 (machine cell rule)

IF there are machines where constraint C1 is violated

THEN remove parts from the machines violating constraint C1

AND place the removed parts on the part waiting list.

Rule-302 (machine cell rule)

IF constraint C2 is violated for a robot or AGV

THEN select a robot or AGV such that C2 is not violated.

3.4.2 Inference engine

One of the greatest advantages of the tandem system architecture is the simplicity of the inference engine. The inference engine in KBS employs a forward-chaining control strategy. In a given class of rules it attempts to fire all the rules which are related to the context considered. If a rule is triggered, i.e. the conditions are true, then the actions of the triggered rule are carried out. However, some rules, for example Rule-201, Rule-202 and Rule-203, stop the search of the knowledge base and send a message to the algorithm.

The inference engine maintains a list of the rules which have been fired. This list is called "explain". The rules in "explain" are placed in the order that they were fired. The list forms a basis for building an explanation facility.

3.4.3 Request processor

The request processor facilitates the interaction between the algorithm and KBS. Based on each request of the algorithm, the request processor calls the inference engine and selects a suitable class of rules to be searched by the inference engine.

3.5 CLUSTERING ALGORITHM

The clustering algorithm presented is an extension of the algorithm presented in Kusiak and Chow (1987a). It takes advantage of two simple observations:

Observation 1

A horizontal line h_i drawn through any row i (machine number i) of matrix $[t_{ij}]$ indicates parts to be manufactured on machine i. This observation is illustrated in matrix (4)

$$[t_{ij}] = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \overset{\text{PART NUMBER}}{\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0.5 & 3.1 & & 2.8 & \\ - & - & 7.4 & - & 1.6 \\ 3.9 & & 4.2 & & 4.3 \end{bmatrix} \end{array}} \begin{array}{l} \\ \\ -h_2 \\ \\ \end{array} \quad \text{MACHINE NUMBER} \qquad (4)$$

The horizontal line h_2 crosses elements (2,3) and (2,5) in matrix(4). Parts 3 and 5 are to be manufactured on machine 2.

Observation 2

A vertical line v_j drawn through any column of matrix $[t_{ij}]$ indicates machines to be used for manufacturing of the corresponding parts.

Based on the two observations the clustering algorithm is developed.

3.5.1 The Algorithm

Step 0 : Set iteration number k=1 .

Construct machine-part incidence matrix.

Send a request to KBS for preprocessing.

Step 1 : Select a machine (row of machine-part incidence matrix) such that it processes the maximum number of parts and is not in the list of temporary bottleneck machines.

Place the selected machine in the list of candidate machines.

Step 2 : From the list of candidate machines, select a machine, which is the most similar to machine cell MC-k. If machine cell MC-k is empty, then choose the machine selected in Step 1.

Draw a horizontal line h_i , where i is the selected machine number.

Step 3 : For each entry crossed once by the horizontal line h_i draw a vertical line v_j . Parts indicated by the vertical lines are potential candidates for part family PF-k.

For each entry $t_{ij} > 0$ crossed by a vertical line v_j, add the corresponding machines, which are not in the list of candidate machines to the list of temporary candidate machines.

Remove the current machines from the list of candidate machines.

Step 4 : KBS analyzes the current machine selected, and takes one of following two actions :

- go to Step 5 (include the current machine in MC-k)
- go to Step 1 (do not include it).

Step 5 : Add the machine considered to machine cell MC-k. Add the corresponding part numbers to part family PF-k. If the list of candidate machines is empty, then go to Step 6, otherwise, go to Step 2 .

Step 6 : KBS analyzes machine cell MC-k for violations of constraints C1-C3 and attempts to satisfy the constraints. Remove machine cell MC-k and part family PF-k from the machine-part incidence matrix.

Step 7 : If the machine-part incidence matrix is not empty, then increment k by 1 and go to Step 1 ; otherwise, STOP.

CHAPTER FOUR

PERFORMANCE OF KBGT

4.1 ILLUSTRATIVE EXAMPLE

Given the machine-part incidence matrix (5), vector fa (frequency of AGV trips required for handling each part), vector fr (frequency of robot trips required to handle each part), max-fa=40 (maximum frequency of trips that can be handled by an AGV), max-fr=100 (maximum frequency of trips that can be handled by a robot), and vector T (the column outside of matrix (5)) represents the maximum processing time available on each machine, solve the group technology problem. The maximum number of machines in a machine cell to be used is 3.

	01	02	03	04	05	06	07	08	09	10	11	12	
fa	11	30	2.5	-	6	10	-	6	7	15	18	14	max-fa (40)
fr	11	30	5	3	6	15	10	12	7	-	36	28	max-fr (100)
PART NUMBER	01	02	03	04	05	06	07	08	09	10	11	12	
MACHINE NUMBER 1		4		21				8					40
2	26		5			10							40
3			20		10					22		8	40
4		35					2	6					50
5	5					6			25				50
6		16		10			3				1		60
7					1					7		7	20

(5)

Iteration 1

Step 0 Iteration number is set to k=1 .

The machine-part incidence matrix is constructed from the input data presented in matrix (5).

A request is sent to KBS for preprocessing. KBS initializes the following lists to empty : MC-k, PF-k, candidate machines, temporary candidate machines, bottleneck machines, temporary bottleneck machines, current machine, waiting parts and machines not used list.

Step 1 Machine 3 is selected since it processes the maximum number of parts and it is not in the list of temporary bottleneck machines. It is placed in the list of candidate machines.

Step 2 Machine 3 is selected from the list of candidate machines.

A horizontal line h_3 is drawn as shown in matrix (6) .

fa [11 30 2.5 - 6 10 - 6 7 15 18 14] max-fa (40)

fr [11 30 5 3 6 15 10 12 7 - 36 28] max-fr (100)

			v 3		v 5					v 10		v 12		
PART NUMBER	01	02	03	04	05	06	07	08	09	10	11	12		
MACHINE NUMBER 1		4		21				8					40	
2	26		5			10							40	
3			20		10					22		8	40	h_3
4		35					2	6					50	
5	5					6			25				50	
6		16		10			3				1		60	
7					1					7		7	20	

(6)

Step 3 Vertical lines v_3, v_5, v_{10}, and v_{12} are drawn (see matrix (6)).

Machine 2 and 7 are added to the list of temporary candidate machines.

Machine 3 is removed from the list of candidate machines.

Step 4 Since the total of the number of machines in MC-k, machines in the list of candidate machines, machines in the list of temporary candidate machines, and the current machine equals 3, constraint C3 is not violated. No current machine rule is fired.

Go to Step 5.

Step 5 Machine 3 is added to MC-k and parts 3, 5, 10, and 12 are included in PF-k.

Since the list of candidate machines is not empty, then go to Step 2.

Step 2 Machine 7 is selected since it is the most similar machine to MC-k.

A horizontal line h_7 is drawn.

Step 3 Vertical lines v_5, v_{10} and v_{12} have already been drawn.

Machine 7 is removed from the list of candidate machines.

Step 4 Since the total of the number of machines in MC-k, machines in the list of candidate machines, machines in the list of temporary candidate machines, and the current machine equals 3, constraint C3 is not violated.

Go to Step 5.

Step 5 Machine 7 is added to MC-k.

Since the list of candidate machines is not empty, then go to Step 2.

Step 2 Machine 2 is selected from the list of candidate machines.

A horizontal line h_2 is drawn (see matrix (7)).

	01	02	03	04	05	06	07	08	09	10	11	12		
fa	11	30	2.5	-	6	10	-	6	7	15	18	14	max-fa (40)	
fr	11	30	5	3	6	15	10	12	7	-	36	28	max-fr (100)	
	v_1		v_3		v_5	v_6				v_{10}		v_{12}		
MACHINE NUMBER \ PART NUMBER	01	02	03	04	05	06	07	08	09	10	11	12		
1		4		21				8					40	
2	26		5			10							40	h_2
3			20		10					22		8	40	h_3
4		35					2	6					50	
5	5					6			25				50	
6		16		10			3				1		60	
7					1					7		7	20	h_7

(7)

Step 3 Vertical lines v_1 and v_6 are drawn (see matrix (7)).

Machine 5 is added to the list of temporary candidate machines.

Machine 2 is removed from the list of candidate machines.

Step 4 Since the total of the number of machines in MC-k and machines in the list of candidate machines and machines in the list of temporary candidate machines and the current machine equals 4, constraint C3 is violated.

Rule-203 is fired :

- part 3 is placed in the part waiting list
- the list of temporary candidate machines is set to empty

- the current machine is set to empty.

Now constraint C3 is no longer violated. Go to Step 5.

Step 5 Since the current machine is empty then no element is added to MC-k and PF-k.

Since the list of candidate machines is empty, then go to Step 6.

Step 6 Constraint C2 is violated, because of part 10 which can not be handled by an AGV. Therefore, rule-303 is fired and a robot is selected as the material handling alternative.

Machine 3 and 7 and part 5, 10 and 12 are removed from the machine-part incidence matrix (see matrix (8)).

Iteration number k is incremented by 1.

Step 7 Since machine-part incidence matrix is not empty, then go to Step 1.

Two more iterations of the algorithm produces the results shown in Figure 10.

$$
\begin{array}{ll}
fa & \begin{bmatrix} 11 & 30 & - & 10 & - & 6 & 7 & 18 \end{bmatrix} \ \text{max-fa } (40) \\
fr & \begin{bmatrix} 11 & 30 & 3 & 15 & 10 & 12 & 7 & 36 \end{bmatrix} \ \text{max-fr } (100)
\end{array}
$$

$$
\begin{array}{ll}
\text{PART NUMBER} & \begin{array}{cccccccc} 01 & 02 & 04 & 06 & 07 & 08 & 09 & 11 \end{array} \\
\text{MACHINE NUMBER} \begin{array}{c} 1 \\ 2 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} & 4 & 21 & & & 8 & & \\ 26 & & & 10 & & & & \\ & 35 & & & 2 & 6 & & \\ 5 & & & 6 & & & 25 & \\ & & 10 & & 3 & & & 1 \end{bmatrix} \begin{array}{c} 40 \\ 40 \\ 50 \\ 50 \\ 60 \end{array}
\end{array}
\qquad (8)
$$

4.2 APPLICATION OF KBGT TO GT PROBLEMS

To illustrate and test performance of the KBGT a number of problems have been considered. The first is a generalized group technology problem represented by matrix (5). The KBGT input represention of matrix (5) is shown in Figure 9.

```
(setq machine-db  '( (m1 ( (parts ((p2 4)(p4 21)(p8 8)))
                           (max-process-time 40)))
                     (m2 ( (parts ((p1 26)(p3 5)(p6 10)))
                           (max-process-time  40)))
                     (m3 ( (parts ((p3 20)(p5 10)(p10 22)(p12 8)))
                           (max-process-time  40)))
                     (m4 ( (parts ((p2 35)(p7 2)(p8 6)))
                           (max-process-time  50)))
                     (m5 ( (parts ((p1 5)(p6 6)(p9 25)))
                           (max-process-time  50)))
                     (m6 ( (parts ((p2 16)(p4 10)(p7 3)(p11 18)))
                           (max-process-time  60)))
                     (m7 ( (parts ((p5 1)(p10 7)(p12 7)))
                           (max-process-time  20)))
                                                              ))

(setq part-db  '( (p1  ((primary-pp (m2 m5))
                        (fr 11) (fa 11)))
                  (p2  ((primary-pp (m1 m4 m6))
                        (fr 30) (fa 30)))
                  (p3  ((primary-pp (m2 m3))
                        (fr 5)  (fa 2.5)))
                  (p4  ((primary-pp (m1 m6))
                        (fr 3)  (fa 0)))
                  (p5  ((primary-pp (m3 m7))
                        (fr 6)  (fa 6)
                  (p6  ((primary-pp (m2 m5))
                        (fr 15) (f 10)))
                  (p7  ((primary-pp (m4 m6))
                        (fr 10) (fa 0)))
                  (p8  ((primary-pp (m1 m4))
                        (fr 12) (fa 6)))
                  (p9  ((primary-pp (m5))
                        (fr 7)  (fa 7)))
                  (p10 ((primary-pp (m3 m7))
                        (fr 0)  (fa 15)))
                  (p11 ((primary-pp (m6))
                        (fr 36) (f 18)))
                  (p12 ((primary-pp (m3 m7))
                        (fr 28) (f 14)))          ))

 (setq max-mc-k-size  0)
 (setq max-fr 100)
 (setq max-fa  40)
```

Figure 9. Input of matrix (5) in KBGT format

The output from KBGT for matrix (5) is shown in Figure 10.

```
((MACHINE-CELL  1)    M3  M7)
((PART-FAMILY   1)    P5 P10 p12)
(M-H-S-ALTERNATIVE   (AGV))
    +++++++++++++++++++

((MACHINE-CELL  2)    M6 M1 M4)
((PART-FAMILY   2)    P2 P4 P7 P11 P8)
(M-H-S-ALTERNATIVE   (ROBOT))
    +++++++++++++++++++

((MACHINE-CELL  3)    M5  M2)
((PART-FAMILY   3)    P1 P6 P9)
(M-H-S-ALTERNATIVE   (ROBOT OR AGV))
    +++++++++++++++++++

-------------------------------------
(PART-WAITING-LIST============>  (P3))
(MACHINES-NOT-USED============>    0 )
(BOTTLENECK-MACHINES==========>    0 )
--------------------------------------
(MAXIMUM-MACHINE-CELL-SIZE-USED   3)
```

Figure 10. KBGT output generated form matrix (5)

As shown in Figure 10, three machine cells and part families have been generated. MC-1 is served by an AGV, MC-2 is served by a robot, and MC-3 can be served by a robot or an AGV. The overlapping part 3 is placed on the part waiting list. The computation was performed for the maximum cell size equal 3.

The second problem is a special case of the generalized GT problem. It is based on 0-1 machine-part incidence matrix as shown in Figure 11.

PART NUMBER

MACHINE NUMBER	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1																																					1					1	
2		1								1																		1				1					1	1		1		1	
3							1										1																	1	1	1							
4					1				1					1					1		1		1						1														
5					1			1	1					1	1	1			1		1		1						1				1								1		1
6	1	1				1	1	1				1	1	1			1		1				1									1	1	1			1		1	1		1	1
7	1												1												1																		
8	1	1	1					1	1		1	1			1				1	1	1		1	1			1	1			1						1	1			1		1
9		1		1						1								1										1				1					1	1		1		1	
10	1											1	1												1	1					1								1				
11			1						1											1				1			1			1													
12											1											1		1			1			1													
13			1																					1																			
14		1				1											1																		1								
15					1									1					1			1											1								1		1
16		1					1			1								1														1					1	1				1	

Figure 11. Machine-part incidence matrix (Burbidge 1973)

For the incidence matrix in Figure 11, the KBGT provides the solution in Figure 12.

```
((MACHINE-CELL  1)  M5 M4 M15)
((PART-FAMILY   1)  P5 P8 P14 P15 P16 P19 P21
                    P23 P29 P33 P41 P43)
     ++++++++++++++++++

((MACHINE-CELL  2)  M9 M2 M16 M1 M14 M3)
((PART-FAMILY   2)  P2 P4 P10 P18 P28 P32 P37
                    P38 P40 P6 P7 P17 P34
                    P35 P36 P42)
     ++++++++++++++++++

((MACHINE-CELL  3)  M10 M7)
((PART-FAMILY   3)  P1 P12 P13 P25 P26 P31 P39)
     ++++++++++++++++++

((MACHINE-CELL  4)  M12 M11 M13)
((PART-FAMILY   4)  P11 P22 P24 P27 P30 P3 P20)
     ++++++++++++++++++

-----------------------------------------------
(PART-WAITING-LIST==========>  (P9))
(MACHINES-NOT-USED===========>  0 )
(BOTTLENECK-MACHINES=========> M8  M6)
-----------------------------------------------
(MAXIMUM-MACHINE-CELL-SIZE-USED    6)
```

Figure 12. KBGT output for the machine-part incidence matrix in Figure 11

To date a large number of clustering algorithms have been developed mostly for solving the 0-1 group technology problem and only a few of them have been tested.

Since the generalized formulation of the GT problem is new, we could not compare performance of the KBGT for this problem. We have identified four 0-1 problems in the GT literature and solved them with KBGT. The solutions obtained are of better quality than ones generated by the four algorithms considered (see Table 1). The computational time complexity of the heuristic clustering algorithms available in the literature is high, for example $O(m^2n+n^2m)$, where m is the number of rows and n is the number of columns in a machine-part incidence matrix (see Kusiak 1985). The algorithm presented in this thesis is an extention of the clustering identification algorithm (Kusiak and Chow 1987a, and 1987b) and has a computational time complexity slightly higher than $O(2mn)$. The CPU time reported in Table 1 is for a SPERRY MICRO IT (an IBM-PC compatible). In addition some of the traditional algorithms listed in Table 1 required human interaction, while KBGT does not. The machine-part incidence matrices for each of the four problems presented in Table 1 and the corresponding KBGT outputs are shown in Appendix I.

Table 1. Solutions of four group technology problems

Problem			Solution					
Problem Reference	Number of Machines	Number of Parts	Solution Method	Maximum Machine Cell Size	Number of Machine Cells and Part Families	Number of Overlapping Parts	Bottleneck Machines	KBGT CPU Time
King and Nakornchai(1982)	16	43	ROC2 Algorithm[1]					
			• solution 1	n/a	-	-	2	
			• solution 2	n/a	4	3	2	
			KBGT	6	4	1	2	4 sec.
Seifoddini(1986)	5	12	SLCA	n/a	2	5	0	
			KBGT	3	2	4	0	1 sec.
Kumar and Vannelli(1987)	30	41	Kumar and Vannelli Algorithm					
			• solution 1	n/a	2	4	0	
			• solution 2	n/a	2	5	0	
			• solution 3	n/a	3	6	0	
			KBGT[2]					
			• solution 1	14	3	7	0	5 sec.
			• solution 2	n/s	3	6	0	7 sec.
Chandrasekharan and Rajagopalan(1987)	40	100	ZODIAC	n/a	10	33	0	
			KBGT					
			• solution 1	11	5	23	0	23 sec.
			• solution 2	10	6	32	0	24 sec.
			• solution 3	9	7	31	0	40 sec.
			• solution 4	n/s	9	32	0	50 sec.

n/a not applicable

n/s not specified

1 solution 1 has been generated by the ROC2 algorithm

2 subcontracting the overlapping part 38 in solution 2 makes machine 24 redundant

4.3 APPLICATION OF KBGT TO TWO INDUSTRIAL CASE STUDIES

The performance of KBGT has been evaluated using industrial data. In the second case study KBGT has been applied to data obtained from Standard Aero Ltd. (overhauling aeroplane engines company in Winnipeg, Manitoba, Canada) to solve a GT problem involving 28 distinct machines and 51 parts. For some machines multiple copies were available. The parts selected represented all process plans in the company.

In the second case study KBGT has been applied on data obtained from Fraunhofer Institut of Industrial Engineering (F. R. of Germany) to solve an industrial GT problem involving 128 machines and 187 parts. The solution results obtained for both case studies are presented in Table 2.

The input and output machine-part incidence matrices for the two case studies are shown in Appendix II.

Table 2. Solutions of two industrial case studies

GT Case Study			Solution						
Case Study Number	Number of Machines	Number of Parts	Solution Number	Maximum Machine Cell Size	ICM[1]	Number of Machine Cells and Part Families	Number of Overlapping Parts	Number of Bottleneck Machines	KBGT CPU Time
Case Study 1	28	51	1	19	1	2	0	5	20 sec.
			2	19	3	2	2	4	20 sec.
			3	9	3	5	20	1	30 sec.
			4	8	3	5	20	2	30 sec.
			5	6	3	5	25	0	32 sec.
Case Study 2	128	187	1	128	3	3	0	0	3:40 min.
			2	90	3	4	48	0	3:50 min.
			3	43	3	6	64	3	4:00 min.

[1] Inter Cellular Movement level

4.4 QUALITY OF SOLUTIONS

In order to present the quality of the solutions provided by KBGT the measure of effectiveness (ME) defined in McCormick et al. (1972) is used:

$$ME = 1/2 \sum_i \sum_j a_{ij} [a_{i,j+1} + a_{i,j-1} + a_{i+1,j} + a_{i-1,j}]$$

where a_{ij} is an element of the 0-1 machine-part incidence matrix.

The measure of effectiveness computed for the solutions provided by KBGT and the solutions existing in the literature is presented in Table 3. The measure of effectiveness for the two industrial case studies is also shown in Table 3. As we can see in Table 3 the quality of solutions provided by KBGT is better than the existing solutions. It can be further improved by changing parameters of the knowledge-based subsystem. In the calculation of ME the overlapping parts as well as parts to be processed on bottleneck machines were excluded.

Table 3. Measure of effectiveness of six group technology problems

Group Technology Problem			Measure of Effectiveness of Solution			
	Number of			Output Matrix Generated by		
Reference	Machines	Parts	Input Matrix	Reference Algorithm	KBGT	Maximum[1] Machine Cell Size
King and Nakornchai(1982)	16	43	54	55	88	6
Seifoddini(1986)	5	12	23	24	23[2]	3
Kumar and Vannelli(1987)	30	41	50	•solution 1 123 •solution 2 110 •solution 3 124	•solution 1 124 •solution 2 108	14 n/s
Chandrasekharan and Rajagopalan(1987)	40	100	133	381	386	n/s
Case Study 1	28	51	139	-	•solution 1 199[3] •solution 2 186 •solution 3 101 •solution 4 75 •solution 5 78	19 19 9 8 6
Case Study 2	128	187	374	-	•solution 1 880 •solution 2 712 •solution 3 551	128 90 43

n/s not specified

[1] used by KBGT only

[2] changing the sequence of the parts in a part family increases the value the measure of effectiveness

[3] the solution obtained for icm=1

CHAPTER FIVE

CONCLUSION

In this thesis a generalized formulation of the problem of grouping machines and parts in an automated manufacturing system was presented. The formulation involves a matrix of processing times and three constraints related to the availability of processing time at each machine, the requirement for material handling carriers, and the maximum number of machines allowed per machine cell. A special case of the grouping problem involving 0-1 machine-part incidence matrix was also considered. To solve the grouping problem a knowledge-based system (KBGT) was developed. The KBGT involves a heuristic algorithm and a knowledge-based subsystem. To demonstrate performance of the knowledge-based system four problems available in the literature have been solved. The solutions obtained are superior to ones presented in the literature. This is due to the clustering algorithm presented and the group technology knowledge included in the knowledge base. Application of KBGT to two industrial case studies was also presented. The approach presented involving an optimization algorithm and a knowledge-based system can be applied to solving other problems as well.

REFERENCES

ANDENBERG, M.R. 1973, __Cluster Analysis for Applications__ (New York: Academic Press).

ASKIN, R. and SUBRAMANIAN, S., 1987, A cost-based heuristic for group technology configuration, __International Journal of Production Research,__ Vol. 25, No. 1, pp. 101-114.

BHAT, M.V. and HAUPT, A., 1976, An efficient clustering algorithm, __IEEE Transactions on Systems, Man and Cybernetics,__ Vol. SMC-6, No. 1, PP. 61-64.

BILLO, R.E., RUCKER, R., and SHUNK, D.L., 1987, Integration of a group technology classification and coding system with an engineering database, __Journal of Manufacturing Systems,__ Vol. 6, No. 1, pp. 37-45.

BURBIDGE, J.L., 1971, Production flow analysis, __The Production Engineer,__ April, pp. 139-152.

BURBIDGE, J.L., 1973, Production Flow Analysis on the computer, The Institute of Production Engineers, Group Technology Division, Third Annual Conference.

CHAN, H.M., and MILNER, D.A., 1982, Direct clustering algorithm for group formation in cellular manufacturing, __Journal of Manufacturing Systems,__ Vol. 1, No. 1, pp. 65-74.

CHANDRASEKHARAN, M.P. and RAJAGOPALAN, R., 1986, MODROC: an extension of rank order clustering for group technology, __International Journal of Production Research,__ Vol. 24, No. 5, pp. 1221-1233.

CHANDRASEKHARAN, M.P. and RAJAGOPALAN, R., 1987, ZODIAC-an algorithm for concurrent formation of part-families and machine-cells, __International Journal of Production Research,__ Vol. 25, No.6, pp. 835-850.

CHENG, Y., and FU, K.S., 1985, Conceptual clustering in knowledge organization, __IEEE Transactions on Pattern Analysis and Machine Intelligence,__ Vol. PAMI-7, No. 5, pp. 592-598.

CHERNIAK, E. and McDERMOTT, D., 1985, __Introduction to Artificial Intelligence,__ (Reading, Mass.: Addison-Wesley).

DE WITTE, J., 1980, The use of similarity coefficients in production flow analysis, __International Journal of Production Research,__ Vol. 18, No. 4, pp. 503-514.

DEKLEVA, J, and MENART, D., 1987, Extentions of Production Flow Analysis, __Journal of Manufacturing Systems,__ Vol. 6, No. 2, pp. 93-105.

DUNLAP, G.C. and HIRLINGER, C.R, 1983, Well planned coding and classification system offers company-wide synergistic benefits, __Industrial Engineering,__ Vol. 15, No. 1, pp. 78-83.

DURIE, F.R.E., 1970, A survey of group technology and its potentail for user application in the U.K., **The Production Engineer,** February, pp. 51-61.

ECKERT, R.L., 1975, Codes and classification systems, **American Machinist,** December, pp. 88-92.

EDWARDS, G.A.B., 1971, The family grouping philosophy, **International Journal of Production Research,** Vol. 9, No. 3, pp. 337-352.

EL-ESSAWY, I.G.K. and TORRANCE, J., 1972, Component flow analysis: an effective approach to production systems' design, **The Production Engineer,** May, pp. 165-176.

EVERITT, B., 1980, **Cluster Analysis.** (New York, N.Y.: Halsted Press).

FAUGHT, W.S., 1986, Applications of AI in engineering, **IEEE Computer,** Vol. 19, No. 7, pp. 17-27.

FAZAKERLAY, G.M., 1974, Group technology: Social benefits and social problems, **The Production Engineer,** October, pp.383-386.

FOX, M.S., and SMITH, S.F., 1984, ISIS: a knowledge-based system for factory scheduling, **Expert Systems Journal,** Vol. 1, No. 1, pp. 25-49.

FU, K.S., 1980, Recent developments in pattern recognition, **IEEE Transactions on Computers,** Vol. C-29, No. 10, pp. 845-854.

GAINS, B.R., 1987, Expert systems in integrated manufacturing: Structure, development and applications, in KUSIAK, A. (Ed.), **Artificial Intelligence: Computer Integrated Manufacturing,** (Kempston, Bedford, U.K.: IFS Publications) and (New York, N.Y.: Springer-Verlag).

GALLAGHER, C.C., and KNIGHT, W.A., 1973, **Group Technology,** (London, U.K.: Butterworths).

GONGAWARE, T.A. and HAM, I., 1981, Cluster Analysis Applications for Group Technology Manufacturing Systems, SME Ninth North American Metalworking Research Conference.

HAM, I. HITOMI, K. and YOSHIDA, T. 1985, **Group Technology.** (Boston, MA: Kluwer-Nijhoff Publishing).

HAYES-ROTH, F., WATERMAN, D.A., LENAT, D.B., (Eds.), 1983, **Building Expert Systems,** (Reading, Mass.: Addison-Wesley).

HERAGU, S. and KUSIAK, A., 1988, Machine layout problem in flexible manufacturing systems, **Operations Research,** Vol. 36, No. 2.

HERAGU, S. and KUSIAK, A., 1987, Analysis of expert systems in manufacturing design, **IEEE Transactions on Systems, Man and Cybernetics,** Vol. SMC-17, No. 6, pp. 899-912.

HOLTZ, R.D., 1978a, GT and CAPP cut work-in-process time 80%, part 1, **Assembly Engineering,** Vol. 21, No. 6, pp. 24-27.

HOLTZ, R.D., 1978b, GT and CAPP cut work-in-process time 80%, part 2, **Assembly Engineering,** Vol. 21, No. 7, pp. 16-19.

HWANG, H., BAEK, W., and LEE, M-K., 1988, Clustering algorithms for order picking in an automated storage and retrieval system, **International Journal of Production Research,** Vol. 26, No. 2, pp. 189-201.

HYER, N.L. Ed., 1984, **Group Technology at Work,** (Dearborn, MI: Society of Manufacturing Engineers).

INGRAM, F.B., 1982, Group Technology, in HYER, N.L. and KING, R.E., (Eds.), **Group Technology at Work,** (Michigan: Society of Manufacturing Engineers).

JACKSON, P., 1986, **Introduction to Expert Systems,** (Reading, Mass.: Addison-Wesley).

KEMPF, K.G., 1985, Manufacturing and artificial intelligence, **Robotics,** Vol. 1, No. 1, pp. 13-25.

KING, J.R., 1980, Machine-component group formation in production flow analysis: An approach using a rank order clustering algorithm, **International Journal of Production Research,** Vol. 18, No. 2, pp. 213-232.

KING, J.R., and NAKORNCHAI, V., 1982, Machine-component group formation in group technology: review and extention. **International Journal of Production Research,** Vol. 20, pp. 117 133.

KLASTORIN, T.D., 1982, The p-median problem for cluster analysis: A comparison test using the mixture model approach, **Management Science,** Vol. 31, No. 1, pp.1134-1146.

KUMAR, K.R., KUSIAK, A. and VANNELLI, A. 1986, Grouping of parts and components in flexible manufacturing systems. **European Journal of Operational Research,** Vol. 24, pp. 387-397.

KUMAR, K.R., and VANNELLI, A. 1987, Strategic subcontracting for efficient disaggregated manufacturing, **International Journal of Production Research,** Vol. 25, No. 12, pp. 1715-1728.

KUMARA, S.R.T., JOSHI, S., KASHYAP, R.L., MOODIE, C.L. and CHANG, T.C., 1986, Expert systems in industrial engineering, **International Journal of Production Research,** Vol. 24, No. 5, pp. 1107-1125.

KUSIAK, A., 1985, The part families problem in flexible manufacturing systems. **Annals of Operations Research,** Vol. 3, pp.279-300.

KUSIAK, A., 1986, Formation of Machine Cells and Part Families in Flexible Manufacturing Systems, Proceedings of the 2nd International Conference on Production Systems, INRA, Paris, France, April, Vol. 1, pp. 15-28.

KUSIAK, A., 1987a, Artificial intelligence and operations research in flexible manufacturing systems. **Information Systems and Operational Research (INFOR)**, Vol. 25, No. 1, pp. 2-12.

KUSIAK, A., 1987b, The generalized group technology concept, **International Journal of Production Research**, Vol. 25, No. 4, pp.561-569.

KUSIAK, A., 1988, **Artificial Intelligence: Implications for CIM**, (New York, N.Y.: Springer-Verlag).

KUSIAK, A., 1988a, EXGT-S: A knowledge-based system for group technology, **International Journal of Production Research**, Vol. 26, No. 5, pp. 887-904.

KUSIAK, A. and CHOW, W.S., 1987a, An algorithm for cluster identification, **IEEE Transactions on Systems, Man and Cybernetics**, Vol. SMC-17, No. 4, pp. 696-699.

KUSIAK, A. and CHOW, W.S., 1987b, Efficient solving of the group technology problem, **Journal of Manufacturing Systems**, Vol. 6, No. 2, pp. 117-124.

KUSIAK, A. and CHOW, W.S., 1988, Decomposition of manufacturing systems, **IEEE Journal of Robotics and Automation**, Vol. 4, No. 5 (forthcoming).

KUSIAK, A. and HERAGU, S., 1987, Group Technology, **Computers in Industry**, Vol. 9, pp. 83-91.

KUSIAK, A. and IBRAHIM, W.M., 1988, KBGT: A Knowledge-Based system for Group Technology, 1988 International Conference on Computer Integrated Manufacturing, Troy, New York, May 25-25, pp. 184-193.

KUSIAK, A., VANNELLI, A., and KUMAR, K.R., 1986, Cluster analysis: Models and algorithms, **Control and Cybernetics**, Vol. 15, No. 2, pp. 139-154.

LAWLER, E.L., LENSTRA, J.K., RINNOOY KAN, A.H.G., and SHOMYS, D.B. Eds., 1985, **The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization**, (New York, N.Y.: Wiley)

LEE, J.L., VOGT, W.G. and MICKLE, M.H., 1982, Calculation of shortest paths by optimal decomposition, **IEEE Transactions on Systems, Man and Cybernetics**, Vol. SMC-12, pp. 410-423.

McAULEY, J., 1972, Machine grouping for efficient production, **The Production Engineer**, February, pp. 53-57.

McCORMICK, W.T., SCHWEITZER, P.J. and WHITE, T.W., 1972, Problem decomposition and data reorganization by clustering technique, **Operations Research,** Vol. 20, pp. 992-1009.

MILLER, A., 1986, Expert Systems, **IEEE Potentials,** October, pp. 12-15.

MULVEY, J.M. and Crowder, H.P., 1979, Clustering analysis: An application of Lagrangian relaxation, **Management Science,** Vol. 25, No. 4, pp. 329-340.

NAGAI, Y., TENDA, S. and SHINGU, T., 1980, Determination of similar task types by the use of the multidimensional classification methods: Towards improving quality of work life and job satisfaction, **International Journal of Production Research,** Vol. 18, No. 3, pp. 307-332.

NEWMAN, P.A., (1987), Scheduling in CIM, in KUSIAK, A., (Ed.), **Artificial Intelligence: Computer Integrated Manufacturing,** (Kempston, Bedford, U.K.: IFS Publications) and (New York, N.Y.: Springer-Verlag).

NILSSON, N., 1980, **Principles of Artificial Intelligence,** (Los Altos, CA: Morgan Kaufmann Publishers).

OBA, F., KATO, K., YASUDA, K. and TSUMURA, T., 1987, Review and extension of cell formation problems in flexible manufacturing systems, **Computer Applications in Production and Engineering,** in BO, K., ESTENSEN, L., FALSTER, P. and WARMAN, E.A. (Eds.), (North-Holland: Elsevier Science Publishers).

OPTIZ, H. and WIENDAHL, H.P., 1971, Group technology and manufacturing systems for small and medium quantity production, **International Journal of Production Research,** Vol. 9, No. 1, pp 181-203.

RAJAGOPALAN, R. and BATRA, J.L., 1975, Design of cellular production systems: A graph theoretic approach, **International Journal of Production Research,** Vol. 13, No. 6, pp. 567-579.

RICH, E., 1983, **Artificial Intelligence,** (New York, N.Y.: McGraw-Hill).

ROBINSON, D. and DUCKSTEIN, L., 1986, Polyhedral dynamics as a tool for machine-part group formation, **International Journal of Production Research,** Vol. 24, No. 5, pp. 1255-1266

SCHAFFER, H., 1981, Implementing CIM, **American Machinist,** August, pp. 151-174.

SEIFODDINI, H. 1986, Improper Machine Assignment in Machine-Component Grouping in Group Technology, Proceedings of the Fall Industrial Engineering Conference, Boston, MA, December 7-10, pp. 406-409.

SEIFODDINI, H. and WOLFE, P.M., 1986, Application of the similarity coefficient method in group technology, **IIE Transactions,** Vol. 18, No.3, pp.271-277.

SEIFODDINI, H. and WOLFE, P.M., 1987, Selection of a Threshold Value Based on Material Handling Cost in Machine-Component Grouping, **IIE Transactions,** Vol. 19, No. 3, pp. 266-270.

SLAGLE, J.L., CHANG, C.I. and HELLER, S.R., 1975, A clustering and data reorganization algorithm, **IEEE Transactions on Systems, Man and Cybernetics,** Vol. SMC-5, pp.125-128.

STANFEL, L.E., 1982, An algorithm using Lagrangean relaxation and column generation for one-dimensional clustering problems, in S.H. Zanakis and J.S. Rustagi (Eds.), **Optimization in Statistics,** (Amsterdam: North Holland), pp. 165-185.

TOU, J.T. and GONZALEZ, R.C., 1974, **Pattern Recognition Principles,** (Reading, Mass.: Addison-Wesley).

VANNELLI, A. and KUMAR, K.R., 1986, A method for finding minimal bottleneck cells for grouping part-machine families, **International Journal of Production Research,** Vol. 24, No. 2, pp. 387-400.

WAGHODEKAR, P.H., and SAHU, S., 1983, Group technology: A research bibliography. **OPSEARCH,** Vol. 29, pp. 225-249.

WATERMAN, D.A., 1986, **A Guide to Expert Systems,** (Reading, Mass.: Addison-Wesley).

WINSTON, P.H., 1984, **Artificial Intelligence,** (Reading, Mass.: Addison-Wesley).

APPENDIX I

Input and output matrix from KBGT for the group technology problem presented in King (1982)

Input Matrix

```
   0000000001111111111222222222233333333334444
   1234567890123456789012345678901234567890123
 1                                     1    1
 2  1       1                 1   1    11 1 1
 3       1         1                111
 4     1   1    1    1 1     1
 5     1  11    111  1 1     1   1       1 1
 6 11   111   111  1 1   1        111  1 11 11
 7 1           1           1
 8 111    11 11  1   111 11  11  1     11  1 1
 9  1 1     1       1        1   1    11 1 1
10 1          11           11    1       1
11   1     1          1   1  1  1
12           1          1 1  1  1
13   1                    1
14  1   1          1                 1
15     1        1    1                     1 1
16  1    1  1       1             1    11   1
```

Output Matrix

```
   0011112223440011233344001333011223312223020
   5845691393132408827802767546123561912470309
 5 111111111111                              1
 4 1 1  1111                                 1
15 1 1  1    11
 9             1111111111
 2             1 1 111111
16             1 11 111 11
 1                   1  1
14             1          111
 3                       1 1111
10                             1111111
 7                             1 11
12                                    11111
11                                      111111
13                                      1  1
 8  1 1 111  111   1 11        11   1 1 11 111
 6  11  1 1 1 11    11 11111 1 111   1
```

Machine Cells : MC-1 = {5,4,15}, MC-2 = {9,2,16,1,14,3}, MC-3 = {10,7}

MC-4 = {12,11,13}

Part Families : PF-1 = {5,8,14,15,16,19,21,23,29,33,41,43}

PF-2 = {2,4,10,18,28,32,37,38,40,42,7,6,17,35,34,36}

PF-3 = {1,12,13,25,26,31,39}

PF-4 = {11,22,24,27,30,3,20}

Overlapping parts : {9}

Bottleneck Machines : {8,9}

Input and output matrix form KBGT for the group technology problem presented in Seifoddini (1986)

Input Matrix

```
  000000000111
  123456789012
1  1 11   1
2 1 1  111 111
3 1 1  111 1
4    11   1  1
5   1111 111 1
```

Output Matrix

```
  000001101000
  136780122459
3 111111
2 1111111 1
5  11 11  1111
1        1 111
4         1111
```

Machine Cells : MC-1 = {3,2,5}, MC-2 = {1}

Part Families : PF-1 = {1,3,6,7,8,10,11}, PF-2 = {2}

Overlapping Parts : {12,4,5,9}

Input and output matrix (solution 1) from KBGT for the group technology problem presented in Kumar and Vannelli (1987)

Input Matrix

```
   00000000011111111112222222222333333333344
   12345678901234567890123456789012345678901
 1          1                    111     1 1
 2           11     1             11     1 1
 3            1          1       11      11
 4       1       1        11  1      1
 5     1           1                1 1
 6      1                   1
 7                1          1      1 1
 8 1      11    1              1
 9   1         1       1
10  1         1       1  1               1
11          1                      1     1 1
12           11      11  11              11
13                   1     1            1
14    11            1       1          1
15                 1                   1
16      1                     1
17       1        1          1        1
18              1 1          1      1
19 1 1                 11       1
20 1 1                 11
21          1 1                  111      1
22          1 1          1       11      11
23  1         1          1       1       1
24                                      1
25    1
26                1          1      1 1
27        1      1
28        1                    1     1
29 1 1     1   1       1        1
30 1 1                 11
```

Output Matrix

```
   11122340331340001232023101331230201322021
   12903902120311393102895557466774668845784
12 1111111                             1
10  1 111 1
23  1  11 11
 3  1  111 11
22  1  111 111
21  1    1 1111
 1      1  11111
 2 11   1   1 11                     1
11      1    111
13   1                                1 1
29              111111
 9               1 11
19              11  111
20              11  1 1
30              11  1 1
 8              1 1    11                  1
28                     111
27                     1  1
 4                       11            1111
 5                         1111
 7                           1111
26                           1111
17                            111        1
18                           1 11          1
15                          1    1
14                         1     111 1
25                                1
 6                                 11
16                                  1     1
24                                    1
```

Machine Cells : MC-1 = {12,10,23,3,22,21,1,2,11,13}

MC-2 = {29,9,19,20,30,8,28,27,4}

MC-3 = {5,7,26,17,18,15,14,25,6,16}

Part Families : PF-1 = {11,12,19,20,23,39,40,2,31,32,10,33,41}

PF-2 = {1,3,9,13,21,30,22,8,29,35,15}

PF-3 = {5,17,34,36,16,27,37,4,26,6}

Overlapping parts : {18,38,24,25,7,28,14}

If overlapping part 38 is subcontracted then machine 24 is redundant.

Input and output matrix (solution 2) from KBGT for the group technology problem presented in Kumar and Vannelli (1987)

Input Matrix

```
   00000000011111111112222222222333333333344
   12345678901234567890123456789012345678901
 1          1                    111     1 1
 2           11     1             11     1 1
 3            1          1       11      11
 4       1       1        11  1      1
 5     1           1                1 1
 6      1                   1
 7                1          1      1 1
 8 1      11    1              1
 9   1         1       1
10  1         1       1  1               1
11          1                      1     1 1
12           11      11  11              11
13                   1     1            1
14    11            1       1          1
15                 1                   1
16      1                     1
17       1        1          1        1
18              1 1          1      1
19 1 1                 11       1
20 1 1                 11
21          1 1                  111      1
22          1 1          1       11      11
23  1         1          1       1       1
24                                      1
25    1
26                1          1      1 1
27        1      1
28        1                    1     1
29 1 1     1   1       1        1
30 1 1                 11
```

Output Matrix

```
   11122234033134123123020031000123212030231
   12903490212031858585664577139310267768944
12 11111111
10  1 11 1 1
23  1  1 1 11
 3  1  1 11 11
22  1  1 11 111
21  1     1 1111
 1       1  11111
 2 11    1   1 111
11       1    111
13   1            11
24                 1
 4      1         1 111               1
16                   1 1
27                  1                   1
28                    1                 11
 6                     11
14               1      1111
25                       1
15                         11
 5                        1 1          1  1
29                           111111
 9                            1 11
19                           11  111
20                           11  1 1
30                           11  1 1
 8                           1 1        11 1
 7                                  11 1  1
17                                  1111
26                                  11 1  1
18                                  11    11
```

Machine Cells : MC-1 = {12,10,23,3,22,21,1,2,11,13,
24,4,16,27,28,6,14,25,15,5}

MC-2 = {29,9,19,20,30,8}

MC-3 = {7,17,26,18}

Part Families : PF-1 = {11,12,19,20,23,24,39,40,2,31,32,
1,33,41,18,25,38,15,28,35,6,26,4,5,37,17}

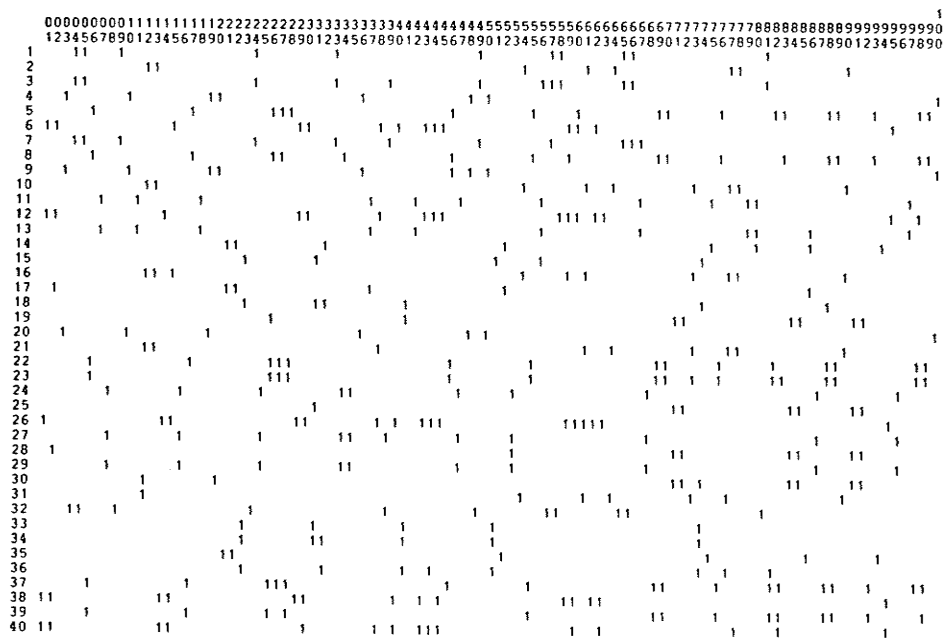
PF-2 = {1,3,9,13,21,30,22}

PF-3 = {16,27}

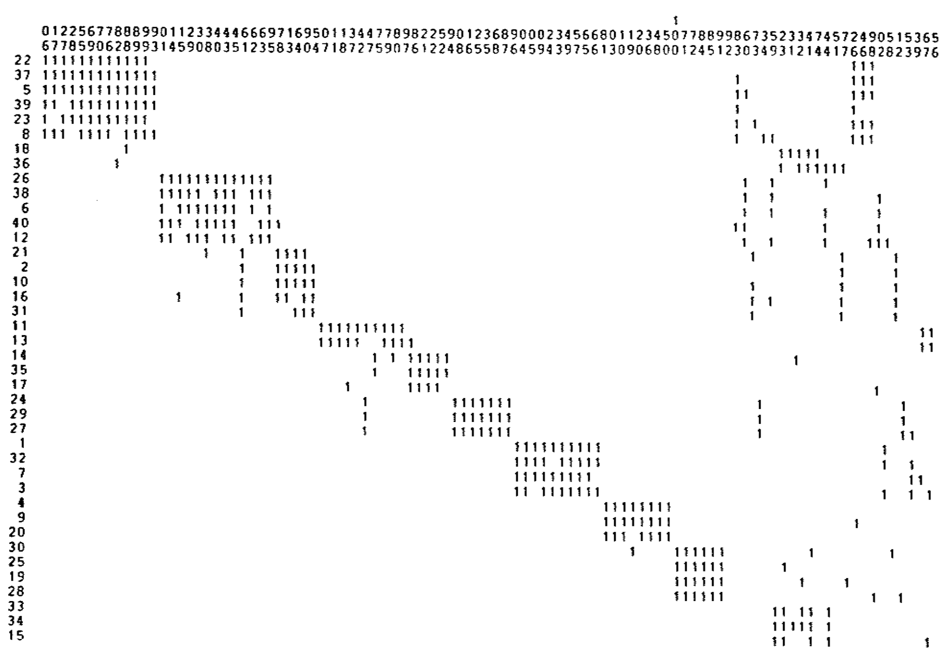
Overlapping parts : {7,36,8,29,34,14}

Input and output matrix (solution 1) from KBGT for the group technology problem presented in Chandrasekharan and Rajagopalan (1987)

Input Matrix



Output Matrix



Machine Cells : MC-1 = {22,37,5,39,23,8,18,36}

MC-2 = {26,38,6,40,12,2,10,16,31}

MC-3 = {11,13,14,35,17,24,29,27}

MC-4 = {1,32,7,3}

MC-5 = {4,9,20,30,25,19,28}

Part Families : PF-1 = {6,17,27,28,55,69,70,76,82,88,89,99,93}

PF-2 = {1,14,15,29,30,38,40,43,45,61,62,63,95,
78,13,64,90,54}

PF-3 = {7,11,18,37,42,47,75,79,80,97,86,21,22
52,94,8,16,25,35,68,87,96}

PF-4 = {4,5,9,24,33,49,57,65,66,81}

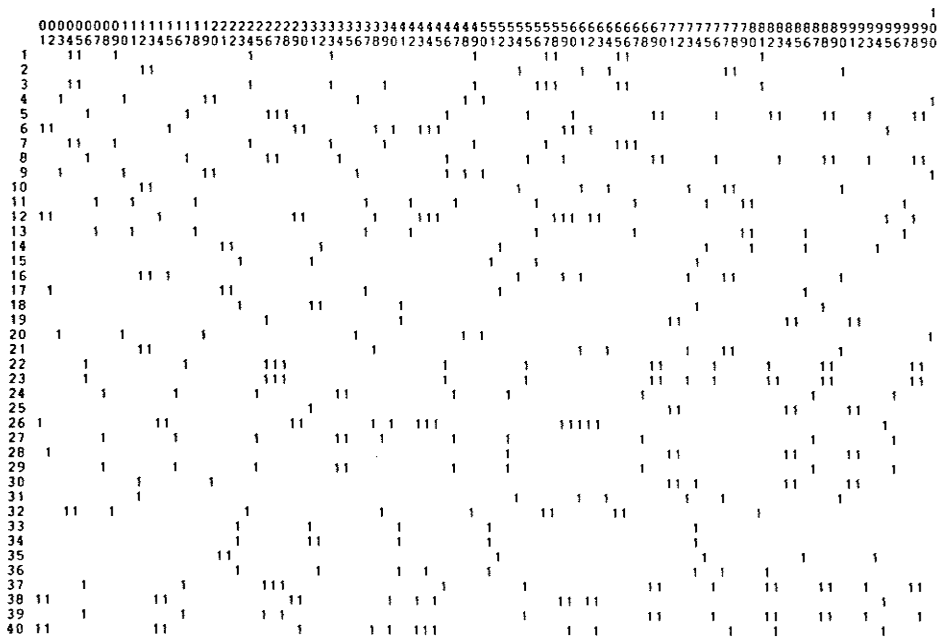
PF-5 = {3,10,19,20,36,48,50,100,71,72,84,85,91,92}

Overlapping Parts : {83,60,73,34,59,23,31,32,41,74,44,51,77,26,
46,98,2,58,12,53,39,67,56}

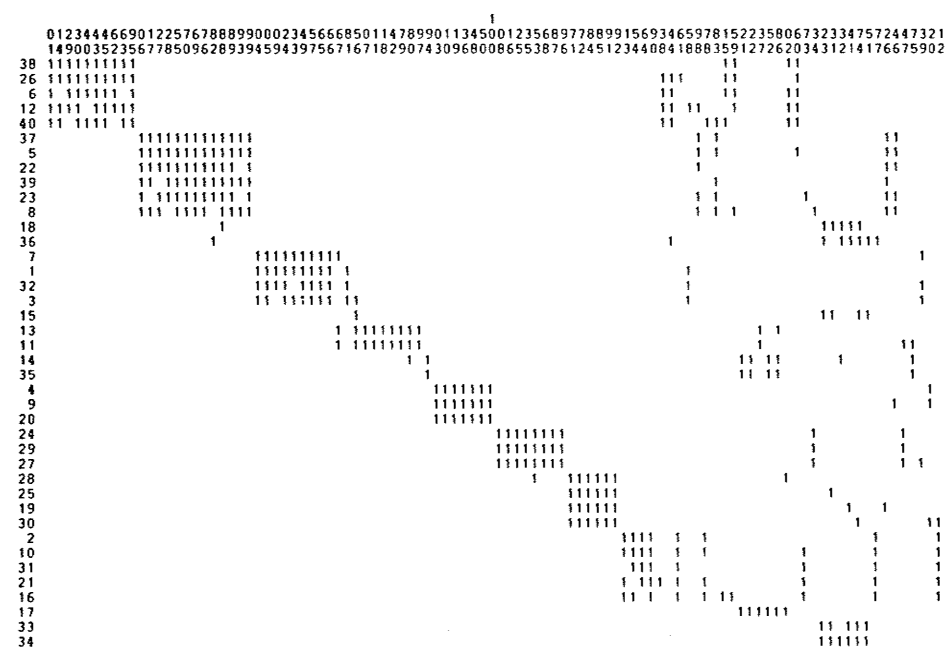
If overlapping parts {23,31,32,41,74,51,56} are subcontracted then machines 33, 34, and 15 are redundant.

Input and output matrix (solution 2) from KBGT for the group technology problem presented in Chandrasekharan and Rajagopalan (1987)

Input Matrix



Output Matrix



Machine Cells : MC-1 = {38,26,6,12,40}

MC-2 = {37,5,22,39,23,8,18,36}

MC-3 = {7,1,32,3,15,13,11,14,35}

MC-4 = {4,9,20}

MC-5 = {24,29,27,28,25,19,30}

MC-6 = {2,10,31,21,16}

Part Families : PF-1 = {1,14,29,30,40,43,45,62,63,95}

PF-2 = {6,17,27,28,55,70,69,76,82,88,89,99,93,99}

PF-3 = {4,5,9,24,33,49,57,65,66,67,81,56,7,11,18 42,79,94}

PF-4 = {3,10,19,36,48,50,100}

PF-5 = {8,16,25,35,53,68,87,96,71,72,84,85,91,92}

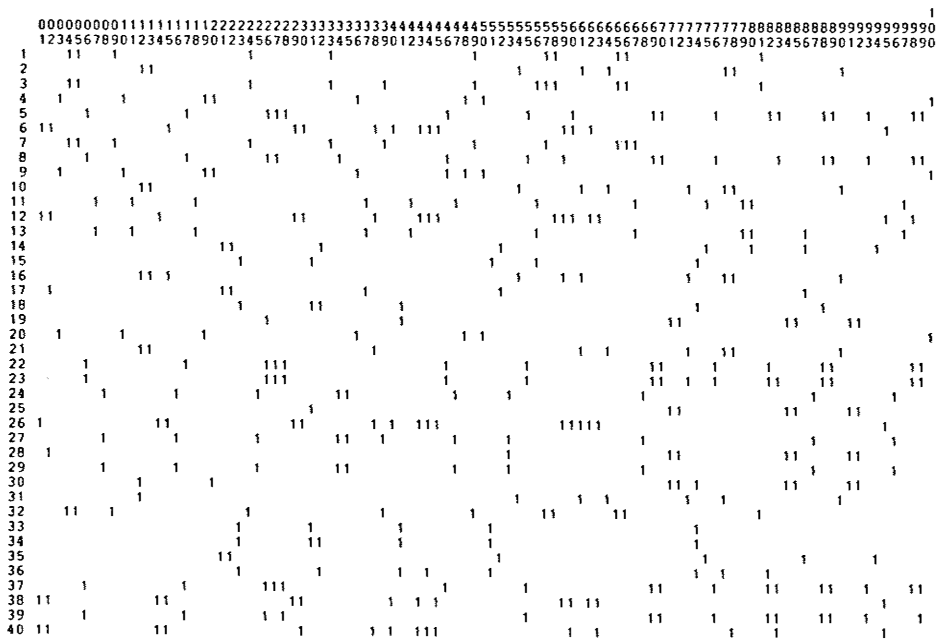
PF-6 = {13,54,64,90}

Overlapping Parts : {38,44,61,58,98,78,83,15,59,21,22,37,52,86,2 60,73,34,23,31,32,41,74,51,77,26,46,47,75,39 20,12}

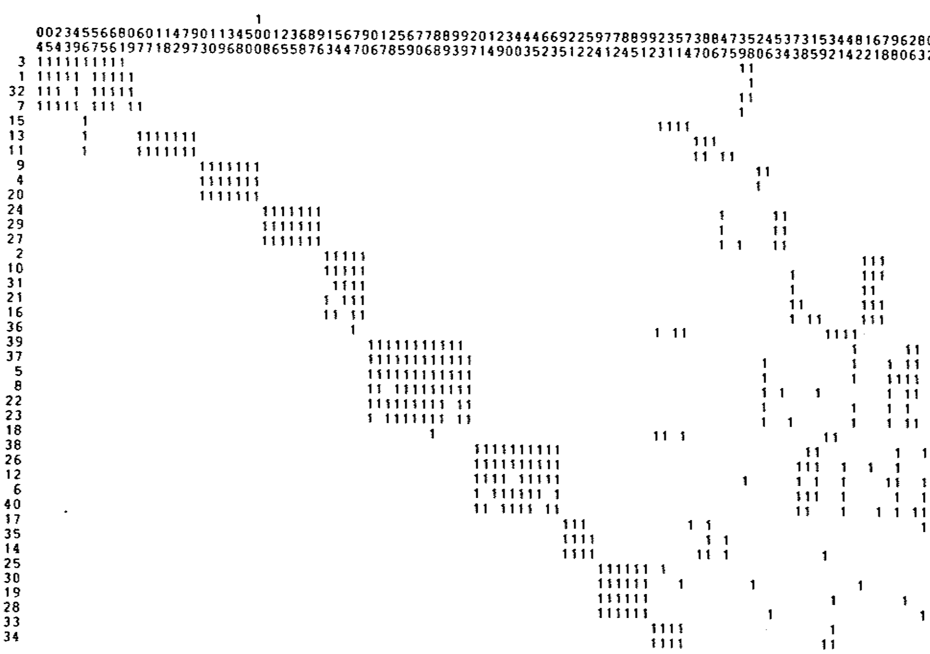
If overlapping parts {21,22,37,52,86,2,23,31,32,41,74,51} are subcontracted then machines 17, 33, and 34 are redundant.

Input and output matrix (solution 3) from KBGT for the group technology problem presented in Chandrasekharan and Rajagopalan (1987)

Input Matrix



Output Matrix



Machine Cells : MC-1 = {3,1,32,7,15,13,11}

MC-2 = {9,4,20}

MC-3 = {24,29,27}

MC-4 = {2,10,31,21,16,36}

MC-5 = {39,37,5,8,22,23,18}

MC-6 = {38,26,12,6,40,17,35,14}

MC-7 = {25,30,19,28}

Part Families : PF-1 : {4,5,24,33,49,56,57,65,66,81,9,67,7,11,
18,42,79,97}

PF-2 : {3,10,19,36,48,50,100}

PF-3 : {8,16,25,35,68,87,96}

PF-4 : {13,54,64,77,90}

PF-5 : {6,17,28,55,69,70,76,88,89,93,99,27}

PF-6 : {1,14,29,30,40,43,45,62,63,95,21,22,52,94}

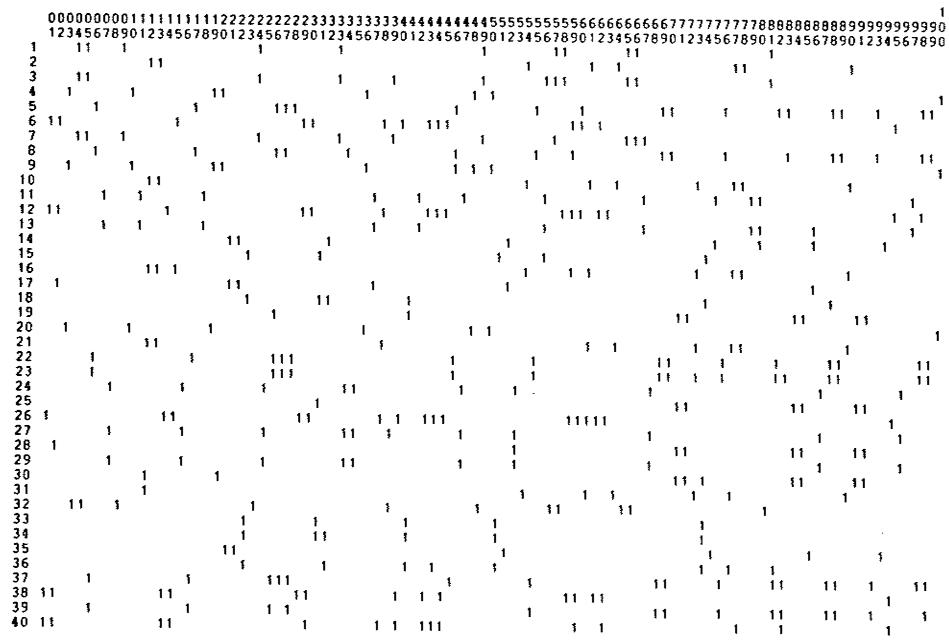
PF-7 : {71,72,84,85,91,92}

Overlapping Parts : {23,31,51,74,37,80,86,47,75,39,58,20,46,53,34,
73,38,15,59,32,41,44,82,12,61,78,98,60,,26,
83,2}

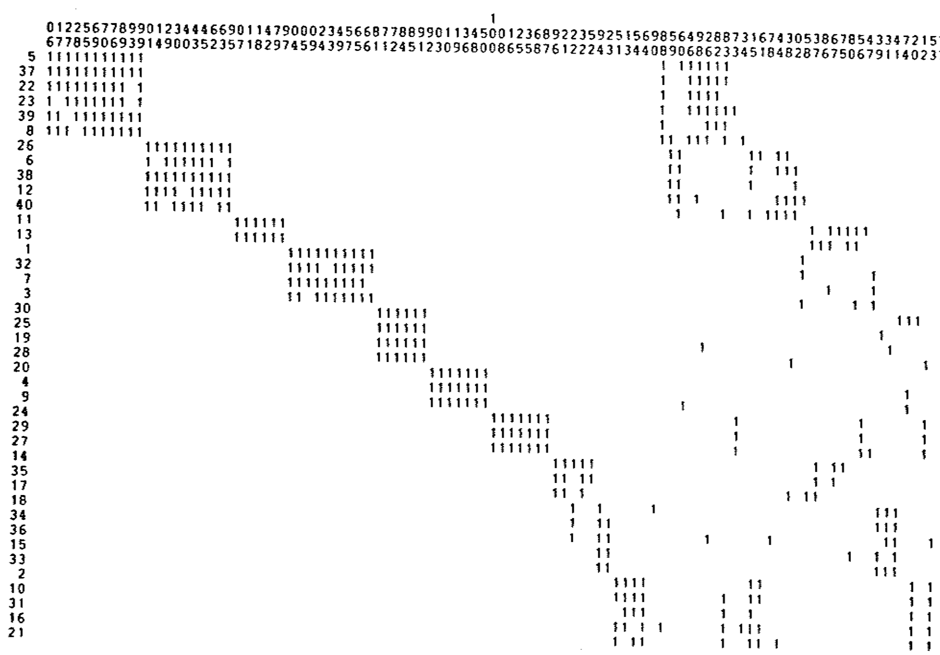
If overlapping parts {23,31,32,41,74,51} are subcontracted then machines 33 and 34 are redundant.

Input and output matrix (solution 4) from KBGT for the group technology problem presented in Chandrasekharan and Rajagopalan (1987)

Input Matrix



Output Matrix



Machine Cells : MC-1 = {5,37,22,23,39,8}

MC-2 = {26,6,38,12,40}

MC-3 = {11,13}

MC-4 = {1,32,7,3}

MC-5 = {30,25,19,28}

MC-6 = {20,4,9}

MC-7 = {24,29,27}

MC-8 = {14,35,17,18,34,36,15,33}

MC-9 = {2,10,31,16,21}

Part Families : PF-1 = {6,17,27,28,55,69,70,76,89,93,99}

PF-2 = {1,14,29,30,40,43,45,62,63,95}

PF-3 = {7,11,18,42,79,97}

PF-4 = {4,5,9,24,33,49,57,65,66,81}

PF-5 = {71,72,84,85,91,92}

PF-6 = {3,10,19,36,48,50,100}

PF-7 = {8,16,25,35,68,87,96}

PF-8 = {21,22,32,52,94,23,51}

PF-9 = {13,54,64,90}

Overlapping parts : {88,59,60,46,98,26,82,83,73,34,15,61,78,44,38,
2,58,37,86,67,75,80,56,47,39,31,41,74,20,12,
53,77}

APPENDIX II

```
       000000000111111111122222222223333333333444444444455
       123456789012345678901234567890123456789012345678901
(1)  1      1    1            1 1        1
     2             1        1                1
(1)  3 1 1  11 1111 11111 1 1  11  1   11  1  11 11111 111
     4            1
(1)  5  1           1            1            1           1
(1)  6  1   1 1                     1   1
     7  1         1   1    1     1
(1)  8  111111  1111 11 111 111 1   1  1   111  1       11
     9  1   1                    1            1 1111 11 11
(1) 10      1                       11    1
    11 11 1 1 11 111     11 1   1            1
    12  1 1111   1 1                         1  1
    13       1                1     1
    14       1
    15        1
(1) 16 11   11   111      1 1   1            1
    17                                          1       11
(3) 18 1 1111   1111  1 1111111 1        1   1  1     1 1
    19                1
(1) 20 111111 1 1111  111111111 1      111  11  111 11 111
(1) 21        11 1  11               1   1   1 11 11111 1
    22        1
    23                      1    1
    24    1    1 11   1 1      1                       1
    25    1   11 1 1            1   1        1  1
    26                1
    27                  1
    28                                 1
```

Input Matrix for Case Study 1

Note: (1) indicates a multiple copy of the corresponding machine
      (3) indicates 3 multiple copies of the corresponding machine

```
       001113334444444403323020011223001224230111122334552
       891450480134567961645701307597459138822236826392017
    21 1111111111111111
    15 1
    22 1
     9          111 1111
    10      1          111
     1   1             1  11
    13      1             1 1
    14                      1
     6 1               1                   11
     3  1111 1 111111111    111111111
    20 1 1   11  11 1111  11 11111   11111
     8   1 11 1        1  1 11 11   1111 1
    18   1             1  11 1111    111111
    16   1             1    111
(1) 20                                       111111111111
(1)  8                                       111111111111
(1) 18                                        111111 111
(1)  3                                        1 11111  11
(1) 16                                       111  11 1
     5                                       1 1   1 1  1
(1)  9                                       1     1 111
     2                                         1  1  1
    17                                                111
     4                                        1
    19                                          1
    26                                          1
    27                                           1
    28                                              1
(1)  1                                             1
(1)  6                                       1
    23                                            1      1
     7   1 1                 1               1     1
    25 111  1                        1         1   1 11
    11 111             1     11      1 1     111  11 1
    24  11                        1  1    1   1 11
    12   1             1    1        11      1 1     11
```

Output Matrix (solution 1) for Case Study 1

maximum machine cell size equal 19 and icm=1

Note: (1) indicates the 1-st multiple copy of the corresponding machine

```
       001113334444444403323020011223001224231111233455220
       891450480134567961645701307597459138822368239201762
    21 1111111111111111
    15 1
    22 1
     9          111 1111                                11
    10      1          111
     1   1             1  11                            1
    13      1             1 1
    14                      1
     7   1 1                 1                          11
     3  1111 1 111111111    111111111                   1
    20 1 1   11  11 1111  11 11111   11111              11
     8   1 11 1        1  1 11 11   1111 1              11
    18   1             1  11 1111    111111             1
    16   1             1    111                         11
     6 1               1                   11            1
(1) 20                                       1111111111
(1)  8                                       1111111111
(1) 18                                       11111 111
(1)  3                                       1 1111  11
(1) 16                                       11  1 1
     2                                        1  1 1
    17                                              111
     5                                        1    1  1 11
(1)  9                                             111
     4                                       1
    19                                         1
    26                                         1
    27                                          1
    28                                            1
    23                                           1     1
    11 111             1     11      1 1     11  1 1    11
    12   1             1    1        11       1    11    1
    24  11                        1  1    1  1 11
    25 111  1                        1        1    11   1
```

Output Matrix (solution 2) for Case Study 1

maximum machine cell size equal 19 and icm=3

Note: (1) indicates the 1-st multiple copy of the corresponding machine

```
       550444444133440112322332112432301230100400123312120
       015134679448053079713167685838261692347289540520921
    17 11                                     1
     5  1                               1111
    12   1                            11 111111
     9 1  111111                      1 111    1
    21    11111111111
     3 11 11111111 1111111
    20 111 11111 11   11  11
     8 111        1  11  1 1                1
    18 1 1           11   11                1
    10                      11        1            1
    23                        1                         1
    24                         1111    1    1   1    1
(1) 18                         11 1
(1)  3                         111 1  111    1  11   11 11
(1)  8                         11  1  111111 11  111 1111
    19                         1
    26                         1
    27                          1
    28                             1
     6                              111   1    1
    14                                       1
    15                                         1
    22                                         1
    13                                       1    11
    25                                 111 11 111  1
    34                                 1       111 1
     1                                111         1 1
     4                                               1
     7                                 11 1      1    1
     2                                   1 1            1
    16                                111111 1       11 11
    11                                1111111  11    11111
(2) 18                                1111 1  1   1 111111
(1) 20 111 11111 11  111  11   11  1  1111111  11 1 111111
```

Output Matrix (solution 3) for Case Study 1

maximum machine cell size equal 9 and icm=3

Note: (1) indicates the 1-st multiple copy of the corresponding machine
(2) indicates the 2-nd multiple copy of the corresponding machine

```
       233322010122312134444444401134200112300231200451335
       821627193013779440134567956858526136987402049205831
     6 11                             11    1
    10   11                            1       1
    23     11
     2     1                             1 1
    16     1 1                        111111 1  11
    11     1 11                       1111111   1111
    18     1 111111
     8     1  111 11
    20     1 111111 1
     3     1 1 11  111                           1
    21                1111111111        1   1  1   1  11
     9                  111 111      11  11        11
(1) 18                           11111  11111  1 111 11
    24                            11 11  1       1 11
    19                            1
    26                            1
    27                             1
     1                              1   11 1   1
    12                           1     1111 1 1    1 1
(1)  8                           111   111111 111111 111111
    15                                       1
    22                                       1
    14                                        1
    13                                        111
     4                                           1
    25                                   11111  1  111
     7                                 1 1 1      1    1
    28                                                   1
     5                                 1  111             1
    17                                               11   1
(1)  3                1111111111 11  1 11 1  1  1  1 11 11
(1) 20     1 111111 1  1  11 1111111  1111111 1 111 11 111
```

Output Matrix (solution 4) for Case Study 1

maximum machine cell size equal 8 and icm=3

Note: (1) indicates the 1-st multiple copy of the corresponding machine

```
       331133444444441232301012220000111122234002351243503
       164548013456797978219303172468123602692974008583155
    10 11                          1             1
    21   111111111111               11        1  1
     9        111 111            1 1       111    1
     7    1                      1   1   1 1
     3   111 11111111111
    20     11  11 1111
     8    1 1          1
     6                  11       1 11
    11                   11     1111111 1111 1
    16                   1      1 1 111 1111  1
(1)  3                   1 11     1 11 1111  11  111 11
(1)  8                    1111  111 111111111 11111  111
(1) 20                   111111 1111111111111  1 11  1111
    18                   111111  11 111111111  1 11 1  11
    23                         1         1
    15                             1
    22                             1
     4                               1
    19                                 1
    26                                 1
     2                                1  1 1
    14                                        1
    13                                        111
    27                                            1
    24                            1  11 1     1    111
    28                                               1
     5                           1     1   11          1
    17                                      1    1    1
    12                           111 1 1    11 1        1
     1                             1 1     1    1        1
    25                            1 11 1   1111  1
```

Output Matrix (solution 5) for Case Study 1

maximum machine cell size equal 6 and icm=3

Note: (1) indicates the 1-st multiple copy of the corresponding machine

```
                                                                                                      1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
   0000000001111111111222222222233333333334444444444555555555566666666667777777777888888888899999999990000000000111111111122222222223333333333444444444455555555556666666666777777777788888888
   1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
 1                                                                                                                                                                                 1
 2                                                                                                                                                             11       11
 3                1                                                                                                                                                           1
 4    1                                                                                                                       1                 1                             1
 5      1
 6                                   1                                                                                             1
 7                                                                                                                       111
 8                    11  1      1     11  11
 9                1                                                                 1        1 11 111         11   1  1             1                                 1
10                           1      1                                                                                                                                             1
11         1
12                                                                 1                                           1
13                1                                                                                                                                                           1
14       11  11  1                                      11
15                                                              1 1                                                                      1  1 1       1    1
16                                                                                                                                                                              1  1
17  1                                                1                                                                         1
18                                1                                                         1                                                                  1
19                                                                                                                                                                          1
20                                                                                          1
21                                               1                                                                                       1    111     1
22                                                                                                                                                        11       11
23       11                 11    111           11  1                          11  11   11   1 1      1           1
24                         1                              1                                                 1      1 1                                 1                             11  1
25                                              1          1   1   1 1111111111                                                       11  1 1     111   11
26           1          11      11        11                                                              1                     1                          1   11      11
27                                                                                                       1                 1                                                                1
28 1 1      1  11    1               1       111   1  11    111  1                1    1  11      1     11 1     1    1 1    1   11                                          1 1 1 1    11   1
29     1            1         1                   1                              1    1                                       1 1                                                          11
30                                             1            1    1  1                                       1          1                                     1        1
31                1                                                                 1        1 11 111         11   1  1       1     1                                 1
32                                                                                  1          11     1
33                           1      1                                                                                                                                             1
34                                                                                  1          11     1
35       1                                      1
36      1                                                      1                                                 1
37                                                                                                                                                            1
38                1                                                                                                                                                           1
39                       1      1
40                                                                                                                               1
41                       1      1
42                       1      1
43                                                                                                                                      1    1  1
44                                                             1                                                                1
45                                                                                                            11
46                  1                                                                                                                            1    1
47                                                                                                                                                        11       11  1
48                                                                                                                                                            11        1
49                                                                                                                                     1  1 1
50                           1      1
51                                                                                                                                     1  1 1       1    1
```

Input Matrix for Case Study 2

```
                                                                                                       1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
    000000000111111111122222222223333333333444444444455555555556666666666777777777788888888889999999999000000000011111111112222222222333333333344444444445555555555666666666677777777778888888888
    1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
 52      1                                                                                                              1
 53                                                                     111111111                            1        1 1
 54        11             1      1   1     1      11  1                                        1 1      1     1               1           1    1  1           1    1       1
 55   1 1    1  11   11         1     1       111    1       111  1               11   11  11      1     11 1     1    1      1  111 1                                       1 1 1 1  1 11 111
 56                                                              1 1
 57                 1                                                                  1       1 11 111 1            1                1
 58                                                                                             11     1                             1
 59                                                                                                       1
 60                                               1  1   1
 61                                                                                            1 1     1
 62       11                  1     1             1          1   1   1 1111111111                              1     1 1               11  1 1     111   11
 63                                  1
 64                                                                                 1 1  1     1 11 111 1            1  1                                     1         1
 65                                                                                                                                1
 66       11                   1      1           1          1   1   1      11                                 1       1               11  1 1     111   11
 67                                                1  1    1                                                            1      1                               1                    1
 68         1       1        1                                         111111  11   1 1  1     1 11 111         11  11          1     1                                 1                     1
 69 1  1                                               11                                                               1                              1                       1
 70                                                                                                                                                                           1
 71                                                                                                       1
 72       11        1                1            1          1   1   1 1111111111      1       11   1111   1   1     11
 73                           1     1                                                          1
 74                                                                                 1 1  1     1111 11111            1  1             1                       1
 75                                                                                            1                                      1
 76 1 1 1   11  11   11                          1  1  1    1 1    1                                                           1                                                 1 1         1
 77                 1          1      1                           1                 1 1  1     1 11 111 1     1 11   1  1             1                                11
 78                                                                                                                                              1
 79                                                                    111 111111
 80                                                                                             1       1      1                       11  1 1     1
 81                         1      1                                                                       1   1     1 1
 82                                                                                                                                    11  1 1     111  111
 83                          1          1
 84                                                                                                                                    11  1 1     111   11
 85                           1     11                       1   1   1 1111111111              1 1      1  1   1                       111 1111    1111  111
 86                                                                                                                  1
 87      111                                      11  1   1  1   1   1 1111111111               1       1  1   1       1
 88                                                                                                                                      1  1 1       1    1
 89                                                                                                                                    11  1 1     111   11
 90 1
 91                1
 92      1
 93                    11                                                                                                                                                                1
 94                1
 95                 1          1      1                                                                         11              1                              1                1
 96                                                1
 97                1     11     11        11                                                                                                                                   1       1
 98                1
 99                                                                                                                                              1
100                                     11  11
101      1
102                                                          1   1   1                                               1  1
```

Input Matrix for Case Study 2 (continued)

```
                                                                                                       111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
    000000000111111111122222222223333333333444444444455555555556666666666777777777788888888889999999999000000000011111111112222222222333333333344444444445555555555666666666677777777778888888
    1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234560123456
103                                                                                         1                                            1  1 1       1    1
104                                                                                                                                                                    1
105 1                       1                                         1111111111      1       11111 11111  1  1                       111  1 1     111  111
106       11        1        1     1             11  1   11                                                         111 1
107      1                          1
108              1                              1            1                                                          1
109      111        1        1     11            1                                                                  111 1                                                     1
110                           1      1                      1   1   1                                               1
111      111        1    1   11 1  11     1      11  1   1  1   1   1 1111111111    1 1  1    11111 11111  1 11 11  111 1       1     111111111111111111 111  11   1    1  1    1    1    1
112                           1      1 1          11 1           1                                                           1
113    1                                       1       11                                   1                                                                                  1
114                                   1             1  1111
115                                                                                                                                      1  1 1       1    1
116                                                                                                                1
117                                                                   1111111111    1 1  1     1 11 111 1  1        111 1                                     1
118                                                                                                                                 1                                               1
119                1   11               11  11
120                                                                                                                                                                                       1
121                          1     11                                                                                   1
122                  1                                                                                                                   1            1    1
123                                                                                                                                      1  1 1       1    1                               11
124                                                                                                                                               1
125      1    11  1    11               11  11                                                                                 11         1  1 1       1    1                          1 11 11
126            1             1      1                    11      1       1                    1 1     11  1                                         11   11
127                                                                  1                                       1               1                                1         1
128                1                                                                                                                                              1
```

Input Matrix for Case Study 2 (continued)

```
                                                        1111111111111111111111111111111111111111  1 1 1 1  1    1     1111         111                   1111 1     111   1           111111111  1  11111111111
    000122223334455566666777777888999999900000011111233333333344444444455555666778024083266517778816834578886011234266145623112233442288570011177827823444568889000112237755109705556665561
    678726792396704715789012345613612345789014679034575123456789012345678012560583831599742245217827249686904564153009196583414801781234121735039466894653499005862351678002238205893423678790
111 11111111111111111111111111111111111111111111111111111111
 85      1   11      1111111111111    1 1      1 1 1         111 1111     1111 111
 62   11    1    1    1     1111111111111                 1   1 1    11   1 1     111   11
105                         1111111111 1 11111111111 1        111   1 1     111   11          111
 72   111      1 1   1111111111111 1  11  1111 1 ;   11
117                  1111111111111 1 11111 11     1111                        1
 87 111        111111111111111111     1      1 1 1     1
 68     1              111111  11111 1 11111     1111  11                        1     1 11
 25            1   1111111111111                        11   1 1     111   11
 66   11    1    1   111        11            1     1   11   1 1     111   11                1
 54   11 1  1 11111                  1 1     1  1          1     1   1          1 1 1         1
 77     1  1                       111 1 11111 1 1 11 1 1 1                     1         1 11
 74                                111 111111111     1 1 1                    1
 31     1                           1  1 11111     11 1 111                       1
  9     1                           1  1 11111     11 1 1 1                       1
 64                                111 1 11111 1     1 1                      1   1
 53                    1111111111            1     1 1
106   111 1  1  1111                              1111                                      1
126     1   1    1     1          1 1    111                      11  11                   1 11
 57     1                          1  1 11111 1      1  1
109 1111 1  11 1                                  1111                                       1
 23   11  11 11 111                1 11 1     1                                          1     11111
 82                                                 11  1 1     111  11            1
 79                    111 111111
 84                                                 11  1 1     111  11
 89                                                 11  1 1     111  11
 80                                   1     1   1   11  1 1     1
 95     1  1                                  11    1                         1   1       1
110        1      111                              1                                     11
 21            1                                     1   111    1
 15                                                   1  1 1      1  1                  1        1
 51                                                   1  1 1      1  1
 88                                                   1  1 1      1  1
102               111                               1  1
115                                                   1  1 1      1  1
 32                                 1   11   1
 34                                 1   11   1
 58                                      11    1       1
103                                                   1  1 1      1  1                            1
112        1    11                                                                       111         11
121       1  11                                         1
 10        1                                                                         1    1
 24                                           1   1 1                                 1 11          111
 27                                          1                                           1 1
 33        1                                                                         1    1
 36 1                                                                                       1      1
 43                                                    1   1 1
 49                                                   1  1 1
 60             111
 61                               1 1    1
 73       1  1                    1
```

Output Matrix (solution 1) for Case Study 2

maximum machine cell size is 128

```
                                         1111:11111111111111111111111111111111111111   1 1 1 1  1    1    1111        111           1111 1      111    1          111111111  1  1111111111111
    000122223334455566666777777788899999999900000011111233333333344444444455555666778024083266517778816834578886011234266145623112233442288570011177827823444568889000112237755109705556665561
    6787267923967047157890123456136123457890146790345751234567890123456780125605838315997422452178272496869045641530091965834148017312341217350394668946534990058623516780022382058934236787
123                                                     1  1 1      1   1                              11
127                                         1                            1  1         1                  1
  4                                                 1           1             1                           1
 14  11           1                                                                      11                11
 26     1  1  1                              1                            11 1                             1 111111
 97     1  1  1                                                                11 1                               111   1
 30                                         1      1                     1              1                1          111
 35  1         1
 45                                           11
 50       1                                                                              1
 52 1                                              1
 75                               1                  1
 81                                        1 1  1 1                                                                    11
107 1        1
108                                                1                                                                11  1
122                                                     1           1   1                                                1
 46                                                                     1   1                                                1
125 1                                                   1  1 1      1   1                 1          1 11  11              1111111111
  8                                                                                                                    11 111111
119                                                                                                                1      111111
100                                                                                                                        1111
 67                                                1                      1                        11                            1  11
 76                                                                              1 1               11               111 11       1  1 11111111
 93                                                                                                      1                    11
 83                                                                              1                1
 44                                                                                     1                        1
 56                                                                                     1         1
 29                                                                                                1   11                1       1     1     1111
 55                                                                                              1  1 11           111 11      1 11  111111111111111111111111111111
 28                                                                              1                1                 111 1       1 11  1 111111   1111111111111111 11111
 69                                                                           1  1 1                      1                                                         111
114               1                                                                      1                                                       1  1               11
113                                                                                              1       1                                        1               111
  6                                                                                                1                                                             1
118                                                                                                             1                    1
 39                                                                                                                    11
 41                                                                                                                    11
 42                                                                                                                    11
  5 1
 17                                                 1                                                                                               1                                  1
 18         1                                                             1                                                                                              1
 20                               1
 59                                        1
 63          1
 71                                        1
 78                                                             1
 86                                            1
 92 1
 99                                                             1
101 1
120                                                                            1
```

Output Matrix (solution 1) for Case Study 2 (continued)

maximum machine cell size is 128

```
                                           11111111111111111111111111111111111111111  1 1 1 1  1    1     1111       111              1111 1     111   1          1111111111  1  11111111111111
    000122223334455566666777777788899999999000000111112333333333444444444555556667780240832665177788168345788860112342661456231122334422885700111778278234445688890001122377551097055566655 6122
    678726792396704715789012345613612345789014679034575123456789012345678012560583831599742245217827249686904564153009196583414801781234121735039466894653499005862351678002238205893423678790 1
124                                                                1
 90                                                                                 1
 11                                                                                    1
104                                                                                         1
 19                                                                                            1
 70                                                                                            1
116                                                                                                 1
 96                                                                                                     1
 65                                                                                                                  1
 91                                                                                                                     1
 94                                                                                                                     1
 98                                                                                                                     1
  1                                                                                                                                        1
 40                                                                                                                                                                     1
 16                                                                               1                                                                                              1
 12                                                                                                  1                                                                            1
  3                                                                                                                     1                                                          1
 13                                                                                                                     1                                                          1
 38                                                                                                                     1                                                          1
128                                                                                                                     1                                                          1
 47                                                                                                                                                                                 11111
 22                                                                                                                                                                                 1111
  2                                                                                                                                                                                     1111
 48                                                                                                                                                                                      111
 37                                                                                                                                                                                      1
  7                                                                                                                                                                                         111
```

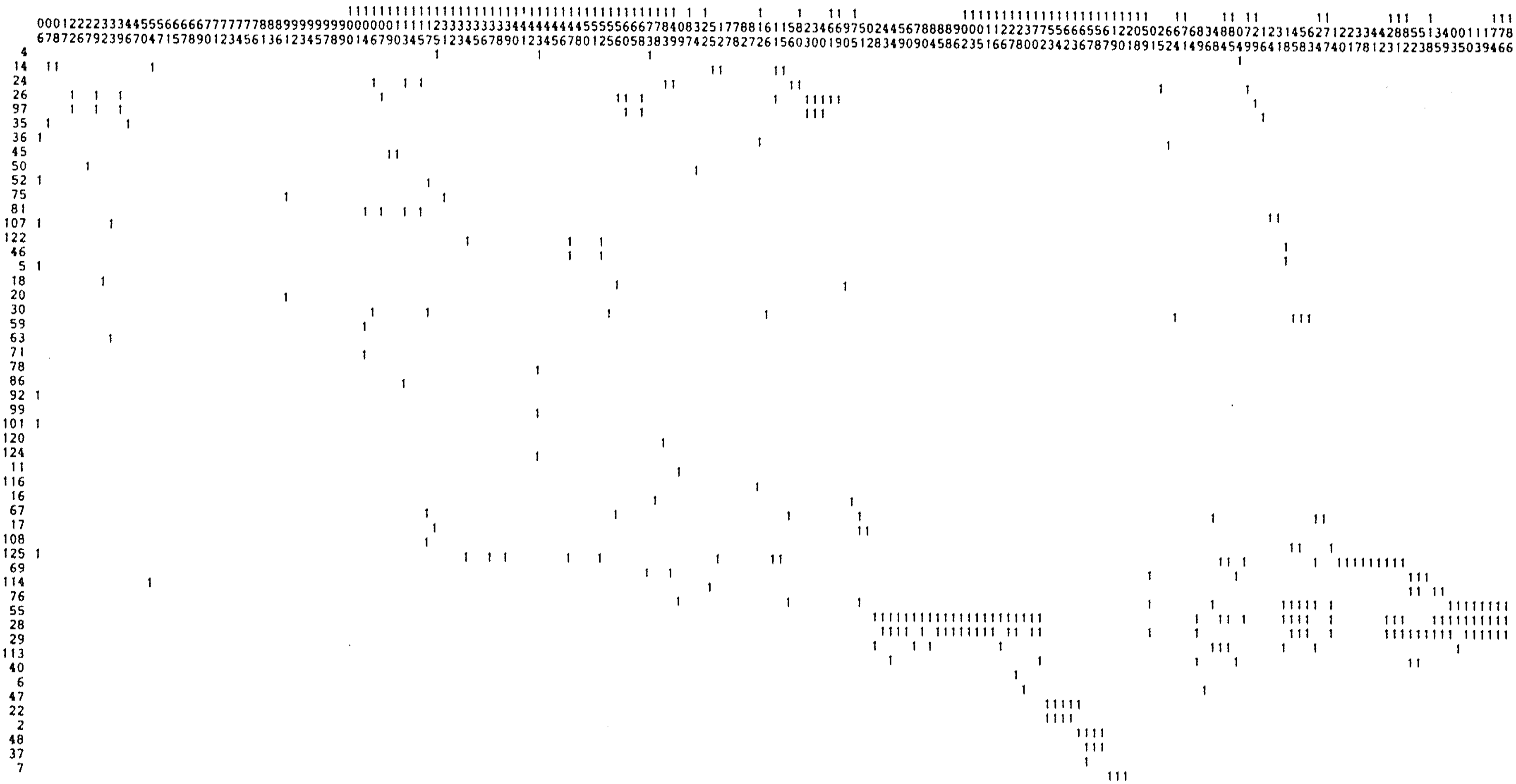
Output Matrix (solution 1) for Case Study 2 (continued)

maximum machine cell size is 128

```
                                           111111111111111111111111111111111111111111 1 1      1    1   11 1             111111111111111111111111   11    11 11       11       111  1       111
    00012222333445556666677777778889999999900000011111233333333444444444555556667784083251778816115823466975024456788889000112223775566655612205026676834880721231456271223344288551340011177 8
    67872679239670471578901234561361234578901467903457512345678901234567801256058383997425278272615603001905128349090458623516678002342367879018915241496845499641858347401781231223859350394 66
111 111111111111111111111111111111111111111111111111111111111111111111111111111111111
 85      1  11     11111111111111   1 1     1 1 1        111 1111    1111 111
 62  11  1  1   1  11111111111111               1  1 1   11  1 1     111  11
105                    1111111111 1 11111111111 1       111  1 1     111  11         1                                                            11
 72  111     1  1  11111111111111 1  11  1111 1 1   11
117                    1111111111111 1 11111 11    1111                      1
 87 111         11111111111111111     1     1 1 1    1
 25             1  11111111111111                        11  1 1     111  11
 68    1               111111  11111 1 11111     1111  11                       1     11                                                           1
 66  11   1     1  111       11                 1    1   11  1 1     111  11            1
 54  11 1  1 11111                  1 1     1  1            1    1  1        1 1 1       1
 77    1  1                      111 1 11111 1 1 11 1 1 1                       1       1                                                           11
 74                              111 111111111      1 1 1                    1
 31    1                          1  1 11111     11 1 111                       1
  9    1                          1  1 11111     11 1 1 1                       1
 64                              111 1 11111 1      1 1                      1  1
 53                    1111111111             1    1 1
106  111 1  1   1111                               1111                                   1
126      1   1     1      1         1 1     111                       11  11              11                                                        1
 57    1                          1  1 11111 1      1   1
109 1111 1  11  1                                  1111                                                                                               1
 23  11  11 11  111              1 11 1     1                                           1   11111
 82                                                      11  1 1     111  11         1
 79                    111 111111
 84                                                      11  1 1     111  11
 89                                                      11  1 1     111  11
 80                                   1     1   1        11  1 1     1
 95    1  1                                      11    1                      1   1     1
110       1        111                             1                                   11
 21              1                                          1    111     1
 51                                                        1  1 1       1   1
 88                                                        1  1 1       1   1
102                111                             1  1
115                                                        1  1 1       1   1
 15                                                        1  1 1       1   1                                                                       1  1
 32                               1    11    1
 34                               1    11    1
 58                                    11    1          1
103                                                        1  1 1       1   1                                                                           1
121      1  11                                        1
 10       1                                                                       1     1
 27                                           1                                        1 1
 33       1                                                                       1     1
 43                                                         1    1  1
 49                                                        1  1 1
 60              111
 61                                 1 1     1
 73      1  1                       1
112       1      11                                                                     11                                                          1    11
123                                                        1  1 1       1   1                                                                              11
127                                            1                             1  1        1       1
```

Output Matrix (solution 2) for Case Study 2

maximum machine cell size is 90



Output Matrix (solution 2) for Case Study 2 (continued)

maximum machine cell size is 90

```
                                            111111111111111111111111111111111111111111 1 1      1    1   11 1             111111111111111111111111   11    11 11       11       111  1       111
     0001222233344555566666777777788899999999900000011111233333333344444444445555556667784083251778816115823466975024456788889000112223775566655612205026676834880721231456271223344288551340011177 8
     6787267923967047157890123456136123457890146790345751234567890123456780125605838399742527827261560300190512834909045862351667800234236787901891524149684549964185834740178123122385935039466
  12                                                                                                                                            1      1
   3                                                                                                                                             1              1
  13                                                                                                                                             1              1
  38                                                                                                                                             1              1
 128                                                                                                                                             1              1
  90                                                                                                                                              1
 104                                                                                                                                                 1
  19                                                                                                                                                  1
  70                                                                                                                                                  1
  56                                                                                                                                                1  1
  96                                                                                                                                                      1
  83                                                                                                                                               1     1
  65                                                                                                                                                           1
  44                                                                                                                                                1          1
  91                                                                                                                                                            1
  94                                                                                                                                                            1
  98                                                                                                                                                            1
  39                                                                                                                                                             11
  41                                                                                                                                                             11
  42                                                                                                                                                             11
   1                                                                                                                                                                    1
 118                                                                                                                                                           1        1
 100                                                                                                                                                                        1111
  93                                                                                                                                                          1           11
 119                                                                                                                                                            1         111111
   8                                                                                                                                                             11       111111
```

Output Matrix (solution 2) for Case Study 2 (continued)

maximum machine cell size is 90

```
                            11111111111111111111  1    1  1             1111111111111111111    1111   11     11111111    1111 1 11     1111 11111     11 111         1   111111   1         111
    0011111234444556678888890001122223777778888855104552008223666777777700333334444444556671234126672385133445566655677881122609571553605640453332223101211856623345620041712408890989999099367
    3503489853459890390458962351636780024691245623898164420299789012345647235680124568010886300191954179578123423678778272901480672456665436724790174790535371562370227864175911630132789145153
 55 11111111111111111111111111111111111111111111
 28 1 111 1 111111111 1 1111111111 11 1111 11  1111
 76 1111111    1 1  1                   11     1   11111                                                                                                                                 1
 29  1   1 1         1 1          1          11     1  1
113           1           1            1        11      1
 69                                             111     1                                                                                                                    1           1    1
108     1      1 1                                                                                                                                                         1
 67                                                 1111                                                                        11                                         1
 40                                 1
114         1   1                               11                                                                                 11
 11                                                1
 96                                                 1
  6                                  1                                                                                               1
 30            1 1  1                                                                                                                 1111                                 1
  4                                                     1                                                                                 1                                 1                 1
 17                                                  1   1                                                                                                                  1
 46      1                                                                                                                                  11
122      1                                                                                                                                  111
123                                          11                                                                                             11111
103                       1                                                                                                                 11111
 93                                       1                                                                                                      11
 24                                       1           1   1                                                                            1                1111                 1
112                                                 1                                                                                1             11             1111
111                                                        11111111111111111111111111111111                                     1  1   11 1111111  1 111 11111111111  1111 11 11111111111 11111
 85                                                           11111111111111 11   111 11                                                    11111           111111               111
105                                                           11111111111111 11   111 11                                                                1                    11  111111111111
 62                                                           1111111111 111 11   111 11                                                                 11 11111     111
 25                                                           1111111111  11 11   111 11                                                                    111         1
 66                                                                 11   111 11   111 11                                                           11      1 111       111
 87                                                           111111111111                                                         1       1              1 111   11  111        11
 72                                                           111111111111                                                                           1    1 111  1    1111       11 11111
 53                                                           11111111111                                                                           11
 79                                                           111 111111
 84                                                                       11 11   111 11
 89                                                                       11 11   111 11
 82                                                                       11 11   111 11                                                                                     1
 54                                                        111              1  1        11                                             11 1                      111 1111        111
 80                                                                     111 11   1                                                                                               11
126                                                              1      1          11 11                                          111                          1 1  1         1  111
 21                                                                         1  111   1                                                                            1
 43                                                                         1  1                                                          1
 59                                                                     1
 71                                                                     1
 97                                                        111                          11 1111
 26                                                        111           1              11  1111111                             1
 91                                                                                           1
 94                                                                                           1
 98                                                                                           1
 65                                                                                                1
118                                                                                                1                                1
 44                                                                                                1                                                                   1
```

Output Matrix (solution 3) for Case Study 2

maximum machine cell size is 43

```
                            11111111111111111111  1    1  1             1111111111111111111     1111  11     11111111    1111 1 11     1111 11111     11 111         1   111111   1         11
    00111112344445566788888900011222237777788888551045520082236667777777003333344444445566712341266723851334455666556778811226095715536056404533322231012118566233456200417124088909899990993 6
    35034898534598903904589623516367800246912456238981644202997890123456472356801245680108863001919541795781234236787782729014806724566654367247901747905353715623702278641759116301327891451 5
 16                                                                                       1        1
 81                                                                     11                          11                                                   11
 39                                                                                                 11
 41                                                                                                 11
 42                                                                                                 11
 27                                                                     1                             1                                                              1
110                                                                                                   1                                            11    1  111
  3                                                                                        1           1
 13                                                                                        1           1
 38                                                                                        1           1
128                                                                                        1           1
 14                                                                                            1        1                         111                                 11
  8                                                                                                 11   1111                                    11
119                                                                                        1             1111                                    11
100                                                                                                      1111
 47                                                                                                          11111
 22                                                                                                          1111
  2                                                                                                              1111
 48                                                                                                               111
 37                                                                                                               1
 23                                                                                                                  11111                         11          11111  111      11111
116                                                                                                                       1
 36                                                                                                                       1                 1                        1
  7                                                                                                                         111
 12                                                                                                                            11
 56                                                                                                                            1                                      1
 15                                                                                                                            1              11111                   1
 18                                                                                                                              11                               1
125                              1        11111        1                                       1        11111                     1        11111111
 68                                                1          111111  11                              1                                               11111               1  1  11   11111 1111
117                                                           11111111111                                                               1                11              1 1  111   11111 11
  1                                                                                                                              1
104                                                                                                                                      1
 78                                                                                                                                       1
 99                                                                                                                                       1
124                                                                                                                                       1
  5                                                                                                                                        1
 92                                                                                                                                        1
101                                                                                                                                        1
 49                                                                                                                                           111
 51                                                                                                                                         11111
 88                                                                                                                                         11111
115                                                                                                                                         11111
 50                                                                                                                                                11
 45                                                                                                                                                   11
 86                                                                                                                                                      1
120                                                                                                                                                        1
 83                                                                                                                                  1                  1
 63                                                                                                                                                              1
107                                                                                                                                         1                     1
 60                                                                                                                                1                              11
```

Output Matrix (solution 3) for Case Study 2 (continued)

maximum machine cell size is 43

```
                            11111111111111111111  1    1  1             1111111111111111111     1111   11     11111111    1111 1 11     1111 11111     11 111         1  111111   1         111
    00111112344445566788888900011222237777788888551045520082236667777777003333344444445566712341266723851334455666555677881122609571553605640453332223101211856623345620041712408890989999099367
    3503489853459890390458962351636780024691245623898164420299789012345647235680124568010886300191954179578123423678778272901480672456665436724790174790535371562370227864175911630132789145153
 35                                                                                                                                                                       1 1
 19                                                                                                                                                                           1
 70                                                                                                                                                                           1
 52                                                                                                                                        1                               1
102                                                                                                                                                      1  111            1
121                                                                                                                                                            111         1
106                                                                                                                                11                1   11    11 11  1111 1
109                                                                                                                                        1         1   11    111    111111
 20                                                                                                                                                                                   1
 73                                                                                                                                                            11                  1
 61                                                                                                                                                                                 111
 90                                                                                                                                                                                         1
 57                                                                                                                                                  1                   1    1     11111 111
 74                                                                                                                                    1                                1 1  11111 11111 111
 32                                                                                                                                                                           1     1     11
 34                                                                                                                                                                           1     1     11
 58                                                                                                                                                                           1           111
 75                                                                                                                                                                                1        1
 10                                                                                                                                               11                                         1
 33                                                                                                                                               11                                         1
 95                                                                                                                            1                  11111                    1                 1
  9                                                                                                                                                 111                 1 1        11111 1111
 31                                                                                                                                                 111                 1 11       11111 1111
 77                                                                                                                                   1 1         11111            1    1 1  111   11111 1111
 64                                                                                                                                    1                                1 1  111   11111 11 1
127                                                                                                                                  111                            1                       1
```

Output Matrix (solution 3) for Case Study 2 (continued)

maximum machine cell size is 43

APPENDIX III

```
;---------------------------------------------------------------------
;     KBGT : A KNOWLEDGE-BASED SYSTEM FOR GROUP TECHNOLOGY            |
;---------------------------------------------------------------------


;*********************************************************************
;*************  BEGINING OF  KNOWLEDGE-BASE  ************************
;*********************************************************************


(setq pre-processing-rules '(

   (rule-101
         ( ((equal t t))

           ((progn
             (initialize-objects-in-db)
             (list 'continue)))))

   (rule-102
         ( ((zerop  max-mc-k-size))

           ((progn
             (setq max-mc-k-size  (-  (length machine-db)  1))
             (list 'continue)))))

   (rule-103
         ( ((>  max-mc-k-size  0))

           ((progn
             (delete-parts-pp-large)
             (list 'continue)))))


   (rule-104
         ( ((< (length part-db)  20))

           ((progn
             (setq icm  33)
             (list 'continue)))))
   (rule-105
         ( ((and
             (< (length part-db)  50)
             (greater-or-equal (length part-db)  20)))

           ((progn
            (setq icm 5)
            (list 'continue)))))
```

```
   (rule-106
          ( ((> (length part-db)  50))

            ((progn
             (setq icm 3)
             (list 'continue)))))
                                       ))

;-------------------------------------------------------------------------------

(setq curr-machine-rules '(

   (rule-201
          ( ((and (zerop (length mc-k))
                  (> (+ (length candidate-machines)
                        (length temp-candidate-machines)
                        1)  max-mc-k-size)))

            ((progn (setq temp-bottleneck-machines (append
                                                     temp-bottleneck-machines
                                                  (list (car curr-machine))))
                    (setq matrix-t (append matrix-t (list curr-machine)))
                    (setq temp-candidate-machines  nil)
                    (setq temp-pf-k  nil)
                    (list 'select-new-machine)))))

(rule-202
      ( ((and
            (> (+ (length mc-k) 1)  max-mc-k-size)
            (> (setq num-shared (car (get-shared-parts  curr-machine pf-k)))
               0)
            (less-or-equal  (* (/ num-shared (length part-db))  100)   icm)
            (setq shared-parts (cadr (get-shared-parts curr-machine pf-k)))))
        ((progn
            (delete-p-in-candidate-machines  shared-parts)
            (delete-p-in-mc-k  shared-parts)
            (delete-p-in-curr-machine  shared-parts)
            (delete-p-in-pf-k  shared-parts)
            (setq part-waiting-list (append part-waiting-list shared-parts))
            (remove-candidate-machines)
            (cond ((not (null curr-machine))
                   (setq matrix-t (append matrix-t (list curr-machine)))))
            (setq temp-bottleneck-machines  nil)
            (setq temp-candidate-machines nil   temp-pf-k nil
                  curr-machine nil   num-shared nil  shared-parts nil)
            (list 'continue)))))

(rule-203
      ( ((and
            (> (+ (length mc-k) 1)  max-mc-k-size)
            (> (setq num-shared (car (get-shared-parts curr-machine pf-k)))
               0)
            (> (* (/ num-shared (length part-db))  100)   icm)))
        ((progn
```

```
                (setq bottleneck-machines (append  bottleneck-machines
                                                    (list (car curr-machine))))
                (remove-candidate-machines)
                (setq temp-bottleneck-machines nil  curr-machine nil
                      temp-pf-k nil  temp-candidate-machines nil num-shared nil)
                (list 'continue)))))

(rule-204
       ( ((and
                (less-or-equal (+ (length mc-k) 1)  max-mc-k-size)
                (=  max-mc-k-size (- (length machine-db) 1))
                (> (setq next-machine-similarity (get-next-machine-similarity))
                   curr-machine-similarity)
                (> (setq num-shared (car (get-shared-parts curr-machine pf-k)))
                   0)
                (less-or-equal (* (/ num-shared (length part-db)) 100)   icm)
                (setq shared-parts (cadr (get-shared-parts curr-machine pf-k)))))
          ((progn
                (delete-p-in-mc-k  shared-parts)
                (delete-p-in-candidate-machines  shared-parts)
                (delete-p-in-curr-machine  shared-parts)
                (delete-p-in-pf-k  shared-parts)
                (setq part-waiting-list (append part-waiting-list shared-parts))
                (remove-candidate-machines)
                (cond ((not (null curr-machine))
                       (setq matrix-t (append matrix-t (list curr-machine)))))
                (setq temp-bottleneck-machines nil   temp-candidate-machines nil
                      temp-pf-k nil   curr-machine nil
                      num-shared nil  shared-parts nil)
                (list 'continue)))))


(rule-205
       ( ((and (< (length mc-k)  max-mc-k-size)
                 (> (+ (length mc-k)
                       (length candidate-machines)
                       (length temp-candidate-machines)
                       1)  max-mc-k-size)
                 (> (setq num-not-shared (car (get-not-shared-parts
                                                         curr-machine))) 0)
                 (< (* (/ num-not-shared (length  part-db)) 100)  icm)
                 (setq not-shared-parts (cadr (get-not-shared-parts
                                                         curr-machine)))))
          ((progn
               (delete-p-in-temp-candidate-m  not-shared-parts)
               (delete-p-in-candidate-machines  not-shared-parts)
               (delete-p-in-curr-machine  not-shared-parts)
               (delete-p-in-temp-pf-k  not-shared-parts)
               (setq part-waiting-list (append part-waiting-list
                                                   not-shared-parts))
               (setq temp-candidate-machines  nil)
               (setq not-shared-parts nil  num-not-shared nil)
               (list  'continue)))))
```

```
(rule-206
      ( ((and (< (length mc-k) max-mc-k-size)
              (> (+ (length mc-k) (length candidate-machines)
                    (length temp-candidate-machines) 1) max-mc-k-size)
              (> (setq num-shared (car (get-shared-parts curr-machine pf-k)))
                 0)
              (< (* (/ num-shared (length  part-db)) 100)  icm)
              (setq shared-parts (cadr (get-shared-parts  curr-machine
                                                          pf-k)))))
        ((progn
           (delete-p-in-mc-k  shared-parts)
           (delete-p-in-candidate-machines  shared-parts)
           (delete-p-in-curr-machine  shared-parts)
           (delete-p-in-pf-k  shared-parts)
           (setq part-waiting-list (append part-waiting-list
                                           shared-parts))
           (cond ((not (null curr-machine))
                  (setq matrix-t (append matrix-t (list curr-machine)))))
           (setq temp-bottleneck-machines (remove (car curr-machine)
                                            temp-bottleneck-machines))
           (setq temp-candidate-machines nil)
           (setq temp-pf-k nil  curr-machine  nil)
           (setq num-shared nil  shared-parts  nil)
            (list  'continue)))))

(rule-207
      ( ((and (< (length mc-k) max-mc-k-size)
              (> (+ (length mc-k) (length candidate-machines)
                    (length temp-candidate-machines) 1) max-mc-k-size)
              (greater-or-equal  (* (/ (car (get-shared-parts  curr-machine
                                        pf-k)) (length  part-db)) 100)   icm)
              (greater-or-equal  (* (/ (car (get-not-shared-parts
                   curr-machine)) (length  part-db)) 100)   icm)
              (check-multiple-machine  curr-machine)
              (setq not-shared-parts (cadr (get-not-shared-parts
                                                      curr-machine)))))
        ((progn
           (setq matrix-t (append matrix-t (construct-multiple-machine
                                      curr-machine not-shared-parts)))
           (setq temp-bottleneck-machines (remove  (car curr-machine)
                                               temp-bottleneck-machines))
           (delete-p-in-curr-machine  not-shared-parts)
           (delete-p-in-temp-pf-k  not-shared-parts)
           (setq  temp-candidate-machines nil   not-shared-parts  nil)
           (list  'continue)))))

(rule-208
      ( ((and (< (length mc-k)  max-mc-k-size)
              (> (+ (length mc-k) (length candidate-machines)
                    (length temp-candidate-machines) 1) max-mc-k-size)
              (greater-or-equal
                 (* (/ (setq num-shared (car (get-shared-parts
                    curr-machine pf-k))) (length  part-db)) 100)
```

```
            icm)
         (greater-or-equal
            (* (/ (setq num-not-shared (car (get-not-shared-parts
                     curr-machine))) (length  part-db)) 100)
            icm)
         (not (check-multiple-machine  curr-machine))))
     ((progn
       (setq bottleneck-machines (append bottleneck-machines
                                      (list (car curr-machine))))
       (setq temp-bottleneck-machines (remove (car curr-machine)
                                          temp-bottleneck-machines))
       (setq curr-machine  nil)
       (setq temp-candidate-machines nil  temp-pf-k nil
            num-shared nil  num-not-shared nil)
       (list  'continue)))))


                                       ))

;------------------------------------------------------------------


(setq machine-cell-rules '(

   (rule-301
        ( ((and (not (zerop  (cadar (cadar mc-k))))
              (setq capacity-violated-machines
                           (violated-capacity   mc-k))))

          ((progn
            (setq parts-deleted (removed-parts-capacity-violation
                                     mc-k  capacity-violated-machines))
            (delete-p-in-mc-k   parts-deleted)
            (delete-p-in-pf-k   parts-deleted)
            (setq part-waiting-list  (append part-waiting-list
                                          parts-deleted))
            (setq capacity-violated-machines nil   parts-deleted nil)
            (list 'continue)))))


   (rule-302
        ( ((not (zerop max-fr)))

          ((progn
             (check-m-h-s  mc-k pf-k)
             (list 'continue)))))
                                     ))


;****************** END OF RULES *************************************

;-----------------------------------------------*
```

```
;  PROCEDURES INVOKED BY RULES                         *
;----------------------------------------------------*


(defun  initialize-objects-in-db  ()
    (setq matrix-t (build-matrix-t  machine-db))
    (setq curr-machine nil)
    (setq bottleneck-machines  nil    temp-bottleneck-machines  nil)
    (setq candidate-machines nil    temp-candidate-machines nil)
    (setq machines-not-used  nil    part-waiting-list  nil)
    (setq pf-k nil    temp-pf-k nil    all-pf-k nil)
    (setq mc-k nil    all-mc-k nil)
    (setq all-m-h-s nil   m-h-s nil)
    (setq explain  nil    part-pp-pairs nil))

(defun delete-parts-pp-large ()
  (let ((parts-deleted))
    (do ((parts  part-db))
        ((null parts) parts-deleted)   ;test
        (cond ((> (length (cadr (assoc 'primary-pp (cadr (car parts)))))
                  max-mc-k-size)
               (setq parts-deleted (append parts-deleted
                                           (list (caar parts))))))
        (setq parts (cdr parts)))
    (cond ((not (null parts-deleted))
           (delete-p-in-matrix-t  parts-deleted)
           (setq part-waiting-list (list parts-deleted))))
    (print parts-deleted)
    (print (length parts-deleted))))


(defun delete-p-in-matrix-t (parts)
    (do ((machines))
        ((null parts) t)  ;test
        (setq machines (cadr (assoc 'primary-pp (cadr (assoc (car parts)
                                                             part-db)))))
        (do ((m)(temp-parts)(curr-part))
            ((null machines)  t)  ;test
            (setq m (car (horizental-line (car machines))))
            (setq matrix-t (remove m  matrix-t))
            (setq temp-parts (cadr m))
            (setq curr-part (assoc (car parts) temp-parts))
            (setq temp-parts (remove curr-part temp-parts))
            (cond ((not (null temp-parts))
                   (setq m (list (car m) temp-parts))
                   (setq matrix-t (append matrix-t (list m))))
                  (t
                   (setq machines-not-used (append machines-not-used
                                                   (list (car m))))))
            (setq machines (cdr machines)))
        (setq parts (cdr parts))))


(defun delete-p-in-candidate-machines  (parts)
```

```
(do ((temp-candidates  candidate-machines)
     (machine-parts) (still-candidate) (m))
    ((null temp-candidates)  t)    ;test
    (setq m (car temp-candidates))
    (setq machine-parts (cadar (horizental-line  m)))
    (setq matrix-t (remove (car (horizental-line m))  matrix-t))
    (do ((temp-machine-parts  machine-parts))
        ((null temp-machine-parts)  t)   ;test
        (cond ((member (caar temp-machine-parts) parts)
                (setq machine-parts (remove (car temp-machine-parts)
                                             machine-parts)))
              ((member (caar temp-machine-parts)  pf-k)
                (setq still-candidate  t)))
        (setq temp-machine-parts (cdr temp-machine-parts)))
    (cond ((null machine-parts)
            (setq machines-not-used (append machines-not-used  (list m)))
            (setq candidate-machines (remove m candidate-machines)))
          ((null still-candidate)
            (setq candidate-machines (remove m  candidate-machines))
            (setq matrix-t (append matrix-t
                                   (list (list m machine-parts)))))
          (t
            (setq matrix-t (append matrix-t (list (list m machine-parts))))
                                             ))
    (setq temp-candidates (cdr temp-candidates))))


(defun delete-p-in-temp-candidate-m  (parts)
   (do ((temp-candidates  temp-candidate-machines)
        (machine-parts) (m))
       ((null temp-candidates)  t)    ;test
       (setq m (car temp-candidates))
       (setq machine-parts (cadar (horizental-line  m)))
       (setq matrix-t (remove (car (horizental-line m))  matrix-t))
       (do ((temp-machine-parts  machine-parts))
           ((null temp-machine-parts)  t)   ;test
           (cond ((member (caar temp-machine-parts)  parts)
                   (setq machine-parts (remove (car temp-machine-parts)
                                                machine-parts))))
           (setq temp-machine-parts (cdr temp-machine-parts)))
       (cond ((null machine-parts)
               (setq machines-not-used (append machines-not-used  (list m)))
               (setq temp-candidate-machines (remove m
                                                 temp-candidate-machines)))
             (t
               (setq matrix-t (append matrix-t (list
                                               (list m machine-parts))))))
       (setq temp-candidates (cdr temp-candidates))))

(defun delete-p-in-mc-k  (parts)
   (do ((temp-mc-k  mc-k) (machine-parts) (m))
       ((null temp-mc-k)  t)     ;test
       (setq m (car temp-mc-k))
       (setq mc-k (remove  m  mc-k))
```

```
        (setq machine-parts (cadr m))
        (do ((temp-machine-parts  machine-parts))
            ((null temp-machine-parts)  t)     ;test
            (cond ((member (caar temp-machine-parts)  parts)
                   (setq machine-parts(remove (car temp-machine-parts)
                                              machine-parts))))
            (setq temp-machine-parts (cdr temp-machine-parts)))
        (cond ((null machine-parts)
               (setq machines-not-used (append machines-not-used
                                               (list (car m))))) ;<==obj in DB
              (t
               (setq mc-k (append mc-k (list (list (car m) machine-parts))))))
        (setq temp-mc-k (cdr temp-mc-k))))

(defun delete-p-in-curr-machine  (parts)
 (let ((machine-parts  (cadr  curr-machine)))
    (do ((temp-machine-parts  (cadr curr-machine)))
        ((null temp-machine-parts)  t)    ;test
        (cond ((member (caar temp-machine-parts)  parts)
               (setq machine-parts (remove (car temp-machine-parts)
                                           machine-parts))))
        (setq temp-machine-parts (cdr temp-machine-parts)))
    (cond ((null machine-parts) (setq curr-machine  nil))
          (t  (setq curr-machine (list (car curr-machine) machine-parts))))))

(defun delete-p-in-pf-k  (parts)
    (do ()
        ((null parts)  t)
        (setq pf-k (remove (car parts) pf-k))
        (setq parts (cdr parts))))

(defun delete-p-in-temp-pf-k  (parts)
    (do ()
        ((null parts) t)
        (setq temp-pf-k (remove (car parts) temp-pf-k))
        (setq parts (cdr parts))))


(defun remove-candidate-machines ()
    (do ((parts)(machines  candidate-machines))
        ((null machines) (setq candidate-machines nil)   t)
        (setq parts (cadr (get-shared-parts (car (horizental-line
                                                  (car machines)))
                                            pf-k)))
        (cond ((zerop (length parts))  t)
              ((> (length parts)  icm)
               (setq matrix-t (remove (car (horizental-line (car machines)))
                                      matrix-t))
               (setq bottleneck-machines (append bottleneck-machines
                                                 (list (car machines)))))
              ((less-or-equal (length parts)  icm)
               (setq part-waiting-list (append part-waiting-list  parts))
               (delete-p-in-candidate-machines  parts)
               (delete-p-in-mc-k  parts)
```

```
            (delete-p-in-pf-k  parts)
            (delete-p-in-curr-machine  parts)))
       (setq machines  (cdr machines))))


(defun greater-or-equal  (a b)
    (cond ((< a b) nil)
          (t   t)))

(defun less-or-equal  (a b)
    (cond ((> a b)  nil)
          (t   t)))


(defun get-next-machine-similarity ()
    (cadr (select-most-similar-machine (append candidate-machines
                                               temp-candidate-machines)
                                       (add-temp-pf-k  temp-pf-k pf-k))))


(defun check-multiple-machine (machine)
   (let ((multiple (assoc 'multiple (cadr (assoc (car machine)
                                                  machine-db)))))
       (cond ((null multiple)  nil)
             ((> (cadr multiple) 0)  t)
             (t  nil))))


(defun construct-multiple-machine (machine parts)
   (let ((machine-attributes (cadr (assoc (car machine) machine-db)))
         (n))
       (setq n (cadr (assoc 'multiple machine-attributes)))
       (setq machine-attributes (remove (assoc 'multiple machine-attributes)
                                        machine-attributes))
       (setq machine-attributes (append machine-attributes
             (list (list 'multiple (- n 1)))))
       (setq machine-db (remove (assoc (car machine) machine-db)
                                machine-db))
       (setq machine-db (append machine-db
                   (list (list (car machine) machine-attributes))))
       (do ((all-parts (cadr machine))
            (new-machine-parts))
           ((null parts) (list (list (car machine) new-machine-parts)))
           (setq new-machine-parts (append new-machine-parts
                                   (list (assoc (car parts) all-parts))))
           (setq parts (cdr parts)))))



(defun sum-process-time (parts)
    (apply '+ (mapcar #'(lambda (p) (cadr p))  parts)))


(defun  violated-capacity  (mc-k)
```

```
    (do ((machines))
        ((null mc-k)  machines)  ;test
        (cond ((> (sum-process-time  (cadar mc-k))
                  (cadr (assoc 'max-process-time
                               (cadr (assoc (caar mc-k) machine-db)))))
               (setq machines (append machines (list (caar mc-k))))))
        (setq mc-k (cdr mc-k))))


(defun  removed-parts-capacity-violation  (mc-k  machines)
    (do ((parts))
        ((null machines)  parts)  ;test
        (setq parts (append parts (lowest-time-part-capacity
                                     (cadr (assoc (car machines) mc-k))
                                     (cadr (assoc 'max-process-time
                                       (cadr (assoc (caar mc-k)
                                                    machine-db)))))))
        (setq machines (cdr machines))))


(defun  lowest-time-part-capacity  (parts  max-process-time)
    (do ((part-deleted) (min-time  max-process-time)
         (sum  (sum-process-time  parts)))
        ((null parts)  (list part-deleted))   ;test
        (cond ((and (less-or-equal (- sum (cadar parts))  max-process-time)
                    (< (cadar parts)  min-time))
               (setq min-time (cadar parts))
               (setq part-deleted (caar parts))))
        (setq parts (cdr parts))))


(defun  check-m-h-s  (mc-k  pf-k)
    (let ((r-violation) (agv-violation))

         (setq r-violation (> (sum-frequency 'fr pf-k) max-fr))
         (setq agv-violation (> (sum-frequency 'fa pf-k) max-fa))
         (cond ((and r-violation agv-violation)
                 (setq m-h-s '(none is suitable)))
               ((and (null r-violation) (null agv-violation))
                 (setq m-h-s '(robot or agv)))
               ((null r-violation)  (setq m-h-s '(robot)))
               ((null agv-violation)  (setq m-h-s '(agv))))))


(defun sum-frequency (freq pf-k)
   (let ((frequencies))
        (setq frequencies (mapcar #'(lambda (p) (cadr
                               (assoc freq (cadr (assoc p part-db))))) pf-k))
        (cond ((member '-  frequencies)
                 (+  (max  max-fa  max-fr) 1))
              (t
               (apply '+ frequencies)))))
```

```
(defun match-mc-pp (mc  pp)
   (do ((temp (append mc bottleneck-machines)))
       ((null pp)  t)
(princ "mc-pp==> ") (print pp) (princ "mc-pp==> ") (print temp)
       (cond ((not (member (car pp) temp))
              (return nil)))
       (setq pp (cdr pp))))


(defun match-all-mc-pp (part  pp)
   (do ((temp  all-mc-k))
       ((null temp) nil)
       (cond ((match-mc-pp (cdar temp)  pp)
              (setq part-waiting-list (remove  part  part-waiting-list))
              (setq part-pp-pairs (append part-pp-pairs (list
                                         (list part pp))))
              (add-p-to-group  part  (car (cdaar temp)))
              (return t)))
       (setq temp (cdr temp))))


(defun add-p-to-group (part  k)
   (do ((left) (temp) (right all-pf-k))
       ((null right)  (print "function add-p-to-group has rigth=null"))
       (cond ((= k (car (cdaar right)))
              (setq temp (append (car right) (list part)))
              (setq all-pf-k (append left (list temp) (cdr right)))
              (return t))
             (t
              (setq left (append left (list (car right))))
              (setq right (cdr right))))))


(defun match-alt-pp-groups (plist)
   (do ((part) (pps))
       ((null plist)  t)
       (setq part (car plist))
       (setq pps (cadr (assoc 'alternative-pps (cadr (assoc part part-db)))))
(princ "alt-groups==> ") (print pps)
       (do ((pp))
           ((null pps) t)
           (setq pp (car pps))
           (cond ((match-all-mc-pp  part  pp) (return t)))
           (setq pps (cdr pps)))
       (setq plist (cdr plist))))


(defun use-alt-pp-for-part-waiting-list  ()
   (cond ((not (null part-waiting-list))
(princ "use=> ") (print part-waiting-list)
          (cond ((listp (car part-waiting-list))
                 (match-alt-pp-groups (cdr part-waiting-list)))
                (t
```

```lisp
          (match-alt-pp-groups part-waiting-list)))
     (print-results  all-mc-k  all-pf-k))))


;***************************************************************************
;*****************  END  OF KNOWLEDGE-BASE  *********************************
;***************************************************************************



;*-----------------------------------------------*
;*      INFERENCE  ENGINE                        *
;*-----------------------------------------------*

(defun carry-out-rule-actions (actions)
    (eval (car actions)))


(defun eval-rule-conditions  (conditions)
    (do ()
        ((null conditions)  t)
        (cond ((not (eval (car conditions))) (return nil)))
        (setq conditions (cdr conditions))))


(defun try-fire-rule (rule)
    (let ((temp-rule) (rule-number) (conditions) (actions)
          (msg) (cond-result) (action-result))

         (setq rule-number (car rule))
         (setq temp-rule (cadr rule))
         (setq conditions (car temp-rule))
         (setq actions (cadr temp-rule))
         (setq cond-result (eval-rule-conditions  conditions))
         (cond ((equal cond-result  t)
                 (setq action-result (carry-out-rule-actions  actions))
                 (setq msg (list rule-number  (car action-result))))
               (t
                 (setq msg (list rule-number  'does-not-apply))))))


(defun kbs-inference-engine  (rules)
    (do ((msg))
        ((null rules) (setq msg '(continue)))
        (setq msg (try-fire-rule (car rules)))
        (cond ((equal (cadr msg) 'select-new-machine)
                (return (cons 'select-new-machine msg))
                (setq explain (append explain (list (car msg))))))
        (setq rules (cdr rules))))


;*-------------------------------------*
```

```
;*    REQUEST  PROCESSOR                  *
;*---------------------------------------*


(defun kbs (request)
    (cond ((equal request 'pre-process)
             (kbs-inference-engine  pre-processing-rules))

          ((equal request 'check-curr-machine)
            (kbs-inference-engine   curr-machine-rules))

          ((equal request 'check-curr-group)
             (kbs-inference-engine  machine-cell-rules))))



;*----------------------------------------------------------------*
;*                 ALGORITHM                                       *
;*----------------------------------------------------------------*


(defun build-matrix-t (machine-db)
    (do ((current-machine) (matrix-t))
        ((null machine-db)  matrix-t)
        (setq current-machine (car machine-db))
        (setq matrix-t (append matrix-t (list (list (car current-machine)
                 (cadr (assoc 'parts (cadr current-machine)))))))
        (setq machine-db (cdr machine-db))))

(defun get-machines-remain  ()
    (do ((machines-remain) (m  matrix-t))
        ((null m)  machines-remain)     ;test
        (cond ((not (or (member (caar m) candidate-machines)
                        (member (caar m) temp-candidate-machines)))
              (setq machines-remain (append machines-remain
                                            (list (caar m))))))
        (setq m (cdr m))))

(defun machine-with-most-p ()
    (do ((machine)(machine-parts)(maximum  0)
         (temp-machines-remain  (get-machines-remain)))

         ((null temp-machines-remain)  (list machine))        ;test
         (cond ((not (member (car temp-machines-remain)
                                             temp-bottleneck-machines))
               (setq machine-parts (cadr (assoc (car temp-machines-remain)
                                            matrix-t)))
               (cond ((> (length machine-parts)  maximum)
                      (setq machine (car temp-machines-remain))
                      (setq maximum (length machine-parts))))))
        (setq temp-machines-remain (cdr temp-machines-remain))))
```

```
(defun select-machines ()
    (machine-with-most-p))


(defun get-not-shared-parts (machine)
    (do ((number 0) (machine-parts (cadr machine))
         (not-shared-parts)
         (parts-in-temp-candidate-m (p-processed-on-machines
                                              temp-candidate-machines)))
        ((null machine-parts) (list number  not-shared-parts))     ;test
        (cond ((and (not (member (caar machine-parts) pf-k))
                    (member (caar machine-parts) parts-in-temp-candidate-m))
                (setq number (+ number 1))
                (setq not-shared-parts (append not-shared-parts
                                          (list (caar machine-parts))))))
        (setq machine-parts (cdr machine-parts))))


(defun p-processed-on-machines (machines)
    (do ((parts))
        ((null machines) parts)
        (setq parts (add-new-parts  parts
                                    (cadar (horizental-line (car machines)))))
        (setq machines (cdr machines))))


(defun get-shared-parts (machine  pf-k) .
    (do ((number 0) (parts-shared)
         (curr-machine-parts  (cadr machine)))

        ((null curr-machine-parts) (list number parts-shared))

        (cond ((member (caar curr-machine-parts)  pf-k)
               (setq number (+ number 1))
               (setq parts-shared (append parts-shared
                                     (list (caar curr-machine-parts))))))
        (setq curr-machine-parts  (cdr curr-machine-parts))))


(defun select-most-similar-machine  (machines  pf-k)
    (let ((num-parts  (length part-db)))
         (cond ((null mc-k) (list (car machines) 2000))
               (t
                (do ((max-similarity  0) (curr-similarity)
                     (similar-machine  (car machines))
                     (num-shared)(num-not-shared)(machine-parts)(m))
                    ((null machines)
                            (list similar-machine  max-similarity))   ;test

                    (setq m (car machines))
                    (setq machine-parts (cadar (horizental-line  m)))
                    (setq num-shared (car (get-shared-parts
                                        (car (horizental-line  m)) pf-k)))
```

```
                (setq num-not-shared (- (length machine-parts)
                                        num-shared))
                (setq curr-similarity (- num-parts
                          (+ num-not-shared (- (length pf-k)
                                               num-shared))))
                (cond ((> curr-similarity  max-similarity)
                       (setq similar-machine  m)
                       (setq max-similarity  curr-similarity)))
                (setq machines (cdr machines)))))))


(defun horizental-line (machine)
    (list (assoc machine matrix-t)))


(defun get-temp-pf-k  (machine)
    (do ((machine-parts  (cadr machine))  (t-pf-k))
        ((null machine-parts)    t-pf-k)  ;test
        (setq t-pf-k (append t-pf-k (list (caar machine-parts))))
        (setq machine-parts (cdr machine-parts))))


(defun get-pf-k (mc-k pf-k)
    (let ((curr-machine-parts   (cadar (last mc-k))))
         (setq pf-k (add-new-parts  pf-k  curr-machine-parts))))


(defun add-new-parts (pf-k curr-machine-parts)
    (do ()
        ((null curr-machine-parts) pf-k)  ;got all parts of curr-machine

        (cond ((not (member (caar curr-machine-parts)  pf-k))
                   (setq pf-k (append pf-k (list (caar curr-machine-parts))))))
        (setq curr-machine-parts (cdr curr-machine-parts))))


(defun delete-candidate-from-temp  (candidates  temps)
    (do ()
        ((null candidates)  temps)     ;test
        (cond ((member (car candidates)  temps)
                   (setq temps (remove (car candidates) temps))))
        (setq candidates  (cdr candidates))))


(defun crossed-once (pf-k candidate-machines)

    (do ( (temp-machines  (get-machines-remain))
          (curr-machine-parts) )

        ((null temp-machines)  candidate-machines)        ; test

        (setq curr-machine-parts  (cadr (assoc (car temp-machines) matrix-t)))
        (do ()
            ((null curr-machine-parts)   t)   ;test
```

```
          (cond ((member (caar curr-machine-parts) pf-k)
                 (setq candidate-machines (append candidate-machines
                                 (list (car temp-machines))))
                 (return  t))

                (t (setq curr-machine-parts  (cdr curr-machine-parts)))))

        (setq temp-machines (cdr temp-machines))))


(defun add-temp-pf-k  (temp-pf-k  pf-k)
    (do ()
        ((null temp-pf-k)  pf-k)
        (cond ((not (member (car temp-pf-k)  pf-k))
               (setq pf-k (append pf-k (list (car temp-pf-k))))))
        (setq temp-pf-k (cdr temp-pf-k))))


(defun add-curr-machine-to-mc  ()
  (cond ((not (null curr-machine))
         (setq mc-k (append mc-k (list curr-machine)))
         (cond ((member (car curr-machine)  temp-bottleneck-machines)
           (setq temp-bottleneck-machines (remove (car curr-machine)
                                 temp-bottleneck-machines)))))))


(defun add-mc-k (all-mc-k  mc-k  k)
   (let ((curr-mc-k  (list (list 'machine-cell  k))))
    (do ()

        ((null mc-k) t)     ; all machines included
        (setq curr-mc-k (append curr-mc-k (list (caar mc-k))))
        (setq mc-k (cdr mc-k)))

    (setq all-mc-k  (append all-mc-k (list curr-mc-k)))))


(defun add-pf-k (all-pf-k  pf-k  k)
    (let ((curr-pf-k (append  (list (list 'part-family  k))   pf-k)))
         (setq all-pf-k (append all-pf-k (list curr-pf-k)))))

(defun add-m-h-s (all-m-h-s  m-h-s  k)
    (cond ((not (null m-h-s))
           (setq all-m-h-s (append all-m-h-s (list (list
                                 'm-h-s-alternative   m-h-s))))
           (setq m-h-s  nil)
           all-m-h-s)
          (t  nil)))
(defun print-results (all-mc-k  all-pf-k)
    (do ()
        ((null all-mc-k) t)
        (print (car all-mc-k))
        (print (car all-pf-k))
```

```
      (cond ((not (null all-m-h-s))
             (print (car all-m-h-s))))
    (print "    ++++++++++++    ")
    (print "                    ")
      (setq all-mc-k (cdr all-mc-k))
      (setq all-pf-k (cdr all-pf-k))
      (cond ((not (null all-m-h-s))
             (setq all-m-h-s (cdr all-m-h-s)))))
  (print '-------------------------------------------------------------------)
  (print (list 'parts-on-waiting-list=====>  part-waiting-list))
  (print (list 'machines-not-used-list====>  machines-not-used))
  (print (list 'bottleneck-machines=======>  bottleneck-machines))
  (print '-------------------------------------------------------------------)
  (print (list 'maximum-machine-cell-size==> max-mc-k-size)))


;-------------------------------------------------------*
;      ALGORITHM  MAIN-LINE                              *
;-------------------------------------------------------*

(defun gt-algorithm ()
    (prog ((kbs-msg))

      STEP0
         (kbs  'pre-process)
         (setq  k  0)

      STEP1
         (setq candidate-machines  (append candidate-machines
                                           (select-machines)))


      STEP2
        (setq curr-machine (select-most-similar-machine  candidate-machines
                                                          pf-k))
         (setq curr-machine-similarity  (cadr curr-machine))
         (setq curr-machine (car curr-machine))
         (cond ((null curr-machine)  (go presult)))
         (setq curr-machine (car (horizental-line curr-machine)))
         (setq temp-pf-k (get-temp-pf-k  curr-machine))
         (setq matrix-t  (remove  curr-machine  matrix-t))

      STEP3
         (setq candidate-machines (remove (car curr-machine)
                                          candidate-machines))
         (setq temp-candidate-machines (crossed-once
                                        temp-pf-k  temp-candidate-machines))
         (setq temp-candidate-machines (delete-candidate-from-temp
                                         candidate-machines
                                         temp-candidate-machines))



      STEP4
```

```
    (setq kbs-msg (kbs  'check-curr-machine))
    (cond ((equal (car kbs-msg)  'continue) t)
          ((equal (car kbs-msg) 'select-new-machine) (go step1))
          (t
            (print "*** algorithm does not understand kbs message")
            (go stop)))



 STEP5
    (setq pf-k (add-temp-pf-k  temp-pf-k pf-k))
    (add-curr-machine-to-mc)
    (setq candidate-machines (append candidate-machines
                                     temp-candidate-machines))
    (setq temp-candidate-machines  nil)


    (cond ((null candidate-machines) (go formgroup))
          (t  (go step2)))

 STEP6
  formgroup
    (cond ((not (null pf-k))
           (setq k (+ k 1))
           (kbs 'check-curr-group)
           (setq all-mc-k (add-mc-k  all-mc-k  mc-k  k))
           (setq all-pf-k (add-pf-k  all-pf-k  pf-k  k))
           (setq all-m-h-s (add-m-h-s  all-m-h-s  m-h-s  k))))


    (setq mc-k nil pf-k nil   temp-bottleneck-machines nil)


 STEP7
  presult
    (cond ((null matrix-t) (print-results all-mc-k  all-pf-k)
                      (go  stop))
          ((= (length matrix-t) (length temp-bottleneck-machines))
             (setq bottleneck-machines (append bottleneck-machines
                                         temp-bottleneck-machines))
             (print "too many bottleneck machines")
             (print "change machine cell size")
             (go stop))
          (t
             (go step1)))


 stop  ))

;*****************************************************************************
;*****************  END  OF  ALGORITHM   *************************************
;*****************************************************************************
```

```
;*------------------------------------------------------*
;*    FUNCTION TO INVOKE  THE KNOWLEDGE-BASED SYSTEM    *
;*------------------------------------------------------*

(defun kbgt ()
    (gt-algorithm)
    (print "                                              ")
    (print "                                              ")
    (print "************   END OF PROCESSING   ************"))


;************************************************************************
;*  PROCEDURES FOR USER INTERFACE     ***********************************
;************************************************************************

(defun create-machine-db  (machine-numbers)
   (do ((parts) (machine-db) (machine-parts) (m))
       ((null machine-numbers)  machine-db)
       (setq machine-parts  ())
       (setq parts  part-db)
       (setq m (car machine-numbers))

       (do ()
           ((null parts)  t)
           (cond ((member m (cadar (cadar parts)))
                  (setq machine-parts (append machine-parts
                                       (list (list (caar parts) 0))))))
           (setq parts (cdr parts)))
       (setq machine-db (append machine-db  (list (list m (list
                                 (list 'parts machine-parts))))))
       (setq machine-numbers (cdr machine-numbers))))


(defun print-group (l num)
   (let ((l2))
        (setq l2 (do ()
                     ((null l)   nil)
                     (cond ((equal num (car (cdaar l)))
                            (return (cdar l))))
                     (setq l (cdr l))))
        (do ((blank  " "))
            ((null l2) (print "****end***"))
            (princ (car l2))
            (princ blank)
            (setq l2 (cdr l2)))))

; ***********************************************************************
; ****************    END    OF    CODE    ******************************
; ***********************************************************************
```