





330  
3385  
1067 COPY 2

STX



# BEBR

FACULTY WORKING  
PAPER NO. 1067

Efficient Algorithms for Grouping  
Component — Processor Families

*K. Ravi Kumar*  
*Anthony Vannelli*

THE LIBRARY OF THE  
SEP 18 1984  
UNIVERSITY OF ILLINOIS  
LIBRANA-CHAMPAIGN

College of Commerce and Business Administration  
Bureau of Economic and Business Research  
University of Illinois, Urbana-Champaign



# BEBR

FACULTY WORKING PAPER NO. 1067

College of Commerce and Business Administration


University of Illinois at Urbana-Champaign

August 1984

Efficient Algorithms for Grouping Component-Processor Families

K. Ravi Kumar, Assistant Professor  
Department of Business Administration

Anthony Vannelli  
University of Toronto



Digitized by the Internet Archive  
in 2011 with funding from  
University of Illinois Urbana-Champaign

<http://www.archive.org/details/efficientalgorit1067kuma>

## Efficient Algorithms for Grouping Component - Processor Families

K. Ravi Kumar  
Department of Business Administration  
University of Illinois  
at Champaign-Urbana  
Champaign, Illinois 61820

Anthony Vannelli  
Mathematical Sciences Department  
IBM Thomas J. Watson Research Center  
Yorktown Heights, NY 10511  
and  
Department of Industrial Engineering  
University of Toronto  
Toronto, Ontario M2S 1A4

### Abstract

The problem of grouping part families is very important in the implementation of Group Technology and Flexible Manufacturing System concepts. In this paper, we opt to use existing routing sheet information to derive the component-processor groups. The actual grouping is done by modelling the problem as an optimal  $k$ -decomposition of weighted networks. Algorithms which are suitable for computer implementation and large problems are developed to find an initial solution and for refining this solution. Bounds on algorithm performance are constructed to give an estimate of the quality of the generated solution. A numerical example illustrates these new techniques.





## 1. Introduction

The trend in product preferences is towards more customization and in terms of manufacturing priorities, this implies smaller batch sizes (Reich, 1983). The conventional approach in meeting such requirements is to utilize functional or process layouts. However, statistical analysis indicate that such job shops have low machine utilization rates and very high waiting times. A more efficient means of satisfying small batch priorities is needed to alleviate the productivity lag in this sector of the manufacturing industry, which comprises approximately 75% of all manufacturing.

Two technologies, which seek to solve this problem, are Group Technology (GT) and Flexible Manufacturing Systems (FMS). They are similar in that they seek to manufacture small lot sizes (in fact, a lot size of 1 in FMS) of "parts of similar process, of somewhat dissimilar materials, geometry and size" (Mitrafanov, 1959). They differ in that GT is conceptually a dedicated cell of machines: grouped, tooled and scheduled as a unit while FMS seems to strive for flexibility in currently manufactured products and those that may arise in the future. Also FMS is aimed at total computer control while GT is satisfactorily implemented in a worker-machine environment.

Despite their differences, both GT and FMS share a common design problem - identification of those "parts of similar process" implying both components and processor types that will make up a GT cell or an FMS (Kusiak, 1984). The original approaches to this problem were based on two different philosophies:

- a) analyze a classified and coded data base of parts which reflect design shape, engineering features as well as methods of manufacture (Hyde, 1981)

b) analyze a route sheet data base which reflects the existing methods of manufacture of parts currently being produced (Burbidge, 1975).

Clearly a) is a more global approach but it suffers due to the fact that classification and coding can take thousands of man-hours without producing any productivity gains in the short run. On the other hand, b) could be implemented quite easily but has been criticized because it "simply perpetuates existing, often, poor practices" (Hyde, 1981).

Our own contention is that for productivity purposes, b) should be implemented for short-term gains and should be thought of as an intermediate step in the implementation of a). The criticism that existing practices are bad may be unfounded and the redesign, once the classification and coding is complete, may be quite simple (especially if the original grouping was a good approximation). Also the productivity gains, with some form of grouping, through reduced set-up times, material handling and waiting times, may be large enough to underwrite the classification and coding projects as well as final redesigns.

In this paper, we will discuss some aspects of efficiently implementing the grouping analysis using existing route sheet information. Specifically we will model the methodology of grouping components and processors as a network decomposition problem, and devise algorithms which aim at producing "good" solutions to the problem. We also give methods to evaluate how good the solutions are by constructing bounds on the optimal solution. These algorithms can be efficiently implemented for very large data sets, as is usually the case in such analysis, typically 2000 components on 100 processors. In Section 2, we formulate the network model for the component-processor grouping. An algorithm to obtain a good initial solution to the problem is described in Section 3. In Section 4, we devise an efficient algorithm which

seeks to improve the initial solution. A numerical example illustrates this new approach in Section 5 and Section 6 contains concluding remarks.

## 2. Component-Processor Grouping and the Network Decomposition Model

Burbidge (1975) proposes Production Flow Analysis as a technique to implement GT. Within this analysis, there is a phase, called Group Analysis, which takes the matrix of components (or packs of components) and processors needed by the components and tries to rearrange this matrix such that block diagonalization (or approximately so) is achieved. For example consider Figure 1 which records the routing requirements of 4 packs consisting of 7 parts. Pack 1, which is just one part, requires machines A and C which pack 2, consisting of three parts, requires machines B and D. This processor/pack information is depicted, in Figure 2, in matrix form with packs as rows and machines as columns. Looking at this matrix, we can see packs 1 and 3 require only machines A and C while packs 2 and 4 need only machines B and D. Thus, a family grouping of packs and machines is possible in which each pack in the family can be processed by the group of machines in that family. This can be seen in Figure 3, where we have exchanged rows 2 and 4 and also columns B and C, to create a partition of the original matrix into diagonal blocks. Each block represents a family consisting of packs and machines, with each pack of parts being processed using only the machines in this family. Any off-diagonal entry, after this block diagonalization process, represents interdependencies of the processing of a pack in one family with the machines in another family. And, of course, the idea is to minimize the interdependencies.

<u>Packs</u>	<u>Part Numbers</u>	<u>Machines Required</u>
1	5	A, C
2	10, 15, 20	B, D
3	25	D
4	30	C

Figure 1

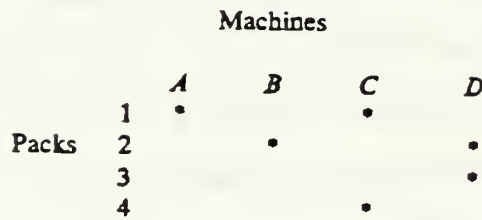


Figure 2

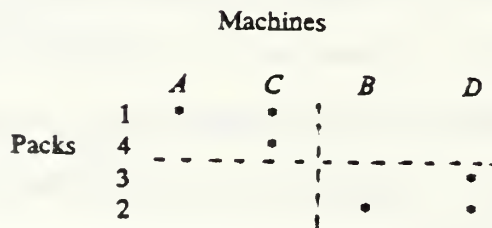


Figure 3

A variety of researchers have found that the process of block diagonalization is not easy to implement on a computer. Burbidge (1975) points out that it is comparatively simple to find the families for a small sample using "pattern recognition, application of production know-how and intuition. It has proved surprisingly difficult to find a method suitable for the computer...". Groover (1980) recognizes that "this is the most subjective and most difficult step in production flow analysis, yet it is the most crucial step in the procedure". El-Essawy and Torrance (1972) indicate that this grouping process requires "an unjustifiably sophisticated procedure." Also, in real world situations, the problem size can be very large, a typical value being 1400 parts on 150 machines. The use of computers become increasingly necessary and efficient algorithms even more so.

Very few attempts to grapple with this problem of grouping components and processors have taken an analytical approach. King (1979, 1980) uses a Rank Order Clustering (ROC) algorithm, while McCormick *et al.* (1972) use a sub-optimization procedure on a restricted quadratic assignment model. King and Nakornchai (1982) briefly review the approaches to this problem and extend the ROC algorithm to perform more efficiently on the computer, regarding storage and CPU time. A major problem in their algorithm is identification of bottleneck machines - this step is quite arbitrary but is very crucial to the development of subsequent grouping. Also, the methods discussed above do not take into account the annual volume of production that is required of each component on the processor i.e. the material handling cost. In their formulations, each component is equally important in terms of cost irrespective of volume.

We formulate this problem as an optimal k-decomposition problem in graph theoretic terms. Instead of looking at block diagonalizations of matrices, we will,

equivalently, look at decompositions of networks. For example, Figure 4 represents the same information as Figures 1 and 2 but in network form. Also, Figure 5 represents the block diagonalized matrix Figure 3, which implies that the optimization problem can also be formulated as : find a decomposition of the packs/machines network such that there are minimal dependencies between the sub-networks.

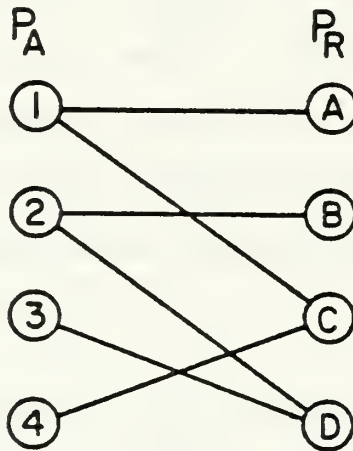


Figure 4

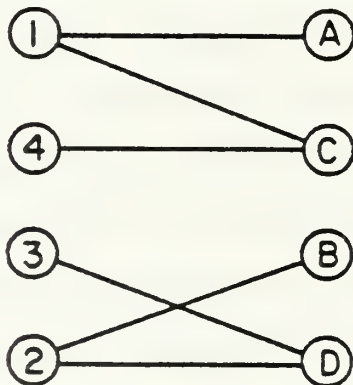


Figure 5

In network theory terms, let  $G = (V, E)$  be an undirected graph, where  $V$  is the set of vertices (or nodes) and  $E$  is the set of edges (or arcs). In this case, let  $V = \{P_A, P_R\}$  where  $P_A$  is the set of packs of parts e.g. final products, subassemblies, spare parts and  $P_R$  is the set of processes e.g. heat treatment, deburring, painting. The arc set  $E$  contains all the interconnections between node set  $P_A$  and node set  $P_R$  where each arc  $(i, j)$  represents the requirement (at least, currently) of processor  $j$  for pack  $i$ .

A  $k$ -decomposition of the graph  $G$  is obtained by deleting edges of  $G$  to obtain  $k$  disconnected subgraphs  $G_i = (V_i, E_i)$ ,  $i = 1, 2, \dots, k$ , and each of the vertices of  $G$  is contained in exactly one of the node sets  $V_i$ . Then, an optimal  $k$ -decomposition of a graph  $G$  is a  $k$ -decomposition that minimizes the weight on the interconnections (or edges) between the  $k$  subgraphs (Vannelli and Vidyasagar, 1984). The factor  $k$  remains under management control and could be a policy variable. For instance,  $k = 1$  would not partition the graph at all while a large  $k$  (in comparison to the total number of nodes) will tend to partition the graph very finely. In the first case, the number of interdependencies is zero while in the latter, it will be quite high. So, there is a trade-off to be made between the number of groups in the decomposition and the amount of interdependencies.

To mathematically model the problem given a fixed  $k$ , let

$$x_{ij} = \begin{cases} 0, & \text{if node } i \text{ is not in subgraph } j \\ 1, & \text{if node } i \text{ is in subgraph } j. \end{cases}$$

Then,

$$\sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, 2, \dots, n$$

implying that each node (and there are  $n$  of them in total i.e.  $\{\# \text{ of elements in } P_A\} + \{\# \text{ of elements in } P_R\} = n$ ) can only be in one subgraph.

Also, one could add a congestion constraint which restricts the number of nodes in each subgraph; i.e.,

$$l \leq \sum_{i=1}^n x_{ij} \leq \alpha, \quad \forall j = 1, 2, \dots, k.$$

We can represent each arc of subgraph  $p$  ( $p = 1, 2, \dots, k$ ) by the node product  $(x_{ip} * x_{jp})$ . Then, arc  $(i, j)$  is in subgraph  $p$  if and only if  $x_{ip} = x_{jp} = 1$ . Let  $a_{ij}$  be the volume of component  $i$  that has to be processed through processor  $j$  (or even the profit or productivity potential associated with  $i$  and  $j$ ). We can now represent the sum of all the arcs that belong within the  $k$  subgraphs by

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^n \left[ \sum_{p=1}^k a_{ij} x_{ip} x_{jp} \right].$$

Note that maximizing this quantity is equivalent to minimizing the sum of the interdependencies of the  $k$  weighted subgraphs.

The optimal  $k$ -decomposition problem can now be stated as:

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^n \left[ \sum_{p=1}^k a_{ij} x_{ip} x_{jp} \right] \\ \text{s t} \quad & \sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, 2, \dots, n \\ & l \leq \sum_{i=1}^n x_{ij} \leq \alpha, \quad \forall j = 1, 2, \dots, k \end{aligned} \tag{1}$$

$$x_{ij} = 0 \text{ or } 1.$$



This is the 0-1 quadratic programming problem with linear constraints which is referred to as the quadratic assignment problem by King and Nakornchai (1982) and this formulation is considered difficult to solve, since it is an NP-complete problem.

In the next two sections we develop algorithms to solve approximations of (1). A starting k-decomposition and improved k-decomposition are found by these techniques. We also develop bounds that convey how good the solution generated is, which is a factor that other analytical approaches have failed to consider.

### 3. A Heuristic Technique for Decomposing Undirected Graphs

In this section, we develop an algorithm for approximating the global solution of the optimal k-decomposition problem (1). A modification of an eigenvector approach introduced by Barnes (1982b) is used to accomplish this.

Barnes formulates an algorithm for approximating the optimal k-decomposition problem with fixed subgraph size.

$$\begin{aligned}
 & \text{Max} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^n \left( \sum_{p=1}^k a_{ij} x_{ip} x_{jp} \right) \\
 & \text{s.t.} \quad \sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, 2, \dots, n \\
 & \quad \quad \sum_{i=1}^n x_{ij} = m_j, \quad \forall j = 1, 2, \dots, k \\
 & \quad \quad x_{ij} = 0 \text{ or } 1, \quad \sum_{j=1}^k m_j = n.
 \end{aligned} \tag{2}$$

The algorithm is a two-step procedure which first finds the k largest eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$  and their corresponding eigenvectors  $u_i$  of the admittance matrix A (Cullum and Donath, 1974). An approximation of problem (2) is then found by

solving the transportation problem

$$\begin{aligned}
 \text{Max} \quad & \sum_{j=1}^k \sum_{i=1}^n \left( \frac{u_{ij}}{\sqrt{m_j}} \right) x_{ij} \\
 \text{s.t.} \quad & \sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, 2, \dots, n \\
 & \sum_{i=1}^n x_{ij} = m_j, \quad \forall j = 1, 2, \dots, k \\
 & x_{ij} \geq 0.
 \end{aligned} \tag{3}$$

Lawler (1976) shows that problem (3) is solved in  $O(n^3)$  time in the worst case. However, this situation rarely arises in practice.

We now proceed to develop an algorithm for approximating the solution of problem (1). We begin by first constructing upper and lower bounds on the number of edges cut,  $E_c$  for problem (1). Define

$$E_u = \text{sum of weighted edges cut by a } k\text{-decomposition}$$

then,  $E_c \leq E_u$ .

It is also desirable to estimate how far  $E_u$  is from  $E_c$ . One would like to construct a lower bound on  $E_c$  in this case. Donath and Hoffman (1973) construct a simple lower bound on  $E_c$  for problem (2). Consider the matrix  $\bar{A}$ , where

$$\begin{aligned}
 \bar{a}_{ii} &= - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} \\
 \bar{a}_{ij} &= a_{ij} \quad (i \neq j)
 \end{aligned}$$

and calculate the  $k$  largest eigenvalues of  $\bar{A}$ ; that is,  $\bar{\lambda}_1 \geq \bar{\lambda}_2 \geq \dots \geq \bar{\lambda}_k$ . A lower bound on  $E_c$  is  $-\frac{1}{2} \sum_{i=1}^k \bar{\lambda}_i m_i$  (Donath and Hoffman, 1973). Thus,  $E_c$  can always be bounded for problem (1) by

$$-\frac{1}{2} \sum_{i=1}^k \bar{\lambda}_i m_i \leq E_c \leq E_a. \quad (4)$$

The upper and lower bounds (4) on  $E_c$  allow us to investigate the optimal  $k$ -decomposition problem (1). Note that one fixes the number of nodes that belong to each subgraph in the number of nodes that belong to each subgraph in the optimal  $k$ -decomposition problem (2). Since one does not know a priori the number of nodes to be fixed in each subgraph, a local optimum of problem (1) may be found by solving *all* the transportation problems (3) where

$$n \geq m_1 \geq m_2 \geq \dots \geq m_k \geq l$$

$$\sum_{i=1}^k m_i = n \quad (5)$$

$$m_i \geq 0, \quad m_i \text{ integer.}$$

Clearly, this approach is unsuitable for problems containing many  $m_i$ 's satisfying (5). However, the lower bound on  $E_c$  given in (4) allows us to develop a more tractable procedure for approximating the solution to problem (1). Note that the optimal

solution of problem (1) is bounded by

$$\begin{aligned}
 \text{Min} \quad & -\frac{1}{2} \sum_{i=1}^k \bar{\lambda}_i m_i \\
 \text{s.t.} \quad & \omega \geq m_1 \geq m_2 \geq \dots \geq m_k \geq \ell \\
 & \sum_{i=1}^k m_i = n \\
 & m_i \geq 0, m_i \text{ integer.}
 \end{aligned} \tag{7}$$

The solution of problem (7) can be obtained in greedy fashion as the following result shows (Vannelli, 1984).

**Theorem 1:** The optimal solution of problem (7) is

$$\begin{aligned}
 m_i^* &= \text{Min}\{m_i; m_i \geq m_{i+1}^*, \omega \geq m_1 \geq m_2 \geq \dots \geq m_k \geq \ell \\
 & \text{and } \sum_{j=1}^{i-1} m_j = n - \sum_{j=i}^k m_j^*\} \text{ for } i = k, k-1, \dots, 1
 \end{aligned} \tag{8}$$

where  $m_{k+1}^* = \ell$ .

**Proof:** We prove this result by contradiction. Assume that we can find  $\{m_i; i = 1, 2, \dots, k\}$  such that  $\omega \geq m_1 \geq m_2 \geq \dots \geq m_k \geq \ell$  and

$$-\frac{1}{2} \sum_{i=1}^k m_i \bar{\lambda}_i < -\frac{1}{2} \sum_{i=1}^k m_i^* \bar{\lambda}_i.$$

where the  $m_i^*$ 's are obtained using formula (8). Letting  $\bar{\lambda}_i = -\frac{1}{2} \bar{\lambda}_i$ , we have

$$\sum_{i=1}^k (m_i^* - m_i) \bar{\lambda}_i > 0$$

and

$$\sum_{i=1}^k (m_i^{\circ} - m_i) = 0.$$

By (9), there exists  $\ell \in \{k, k-1, \dots, 1\}$  such that

$$m_{\ell}^{\circ} - m_{\ell} > 0.$$

In addition, we claim

$$m_i^{\circ} - m_i \geq 0 \quad \forall i = 1, 2, \dots, \ell. \quad (10)$$

To show that (10) is true, assume that  $m_i^{\circ} - m_i < 0$  for some  $i \in \{1, 2, \dots, \ell-1\}$ . Then, there exists a partition  $\{m_i^{\circ\circ}\}$  such that

$$\begin{aligned} m_j^{\circ\circ} &= m_j^{\circ} + 1 && \text{for some } j \in \{1, 2, \dots, \ell-1\} \\ m_{\ell}^{\circ\circ} &= m_{\ell}^{\circ} - 1 \\ m_i^{\circ\circ} &= m_i^{\circ} && \text{otherwise.} \end{aligned}$$

This implies  $m_{\ell}^{\circ}$  has not been chosen according to (8). Therefore

$$\sum_{i=1}^{\ell} (m_i^{\circ} - m_i) \bar{\lambda}_i > - \sum_{i=\ell+1}^k (m_i^{\circ} - m_i) \bar{\lambda}_i \geq 0.$$

Since  $\bar{\lambda}_1 \leq \bar{\lambda}_2 \leq \dots \leq \bar{\lambda}_k$ , then

$$\sum_{i=1}^{\ell} (m_i^{\circ} - m_i) \bar{\lambda}_i \leq - \sum_{i=\ell+1}^k (m_i^{\circ} - m_i) \bar{\lambda}_i.$$

We have a contradiction. ■

Problem (7) is easily solved in the optimal 2-decomposition case. Since the largest eigenvalue of  $\bar{A}$  is  $\lambda_1 = 0$ ,  $m_2$  is chosen to be the smallest integer value greater than or equal to  $\ell$  and satisfying  $m_1 + m_2 = n$ . This result shows that the best

weighted cut decomposes the graph into one large subgraph and one smaller one. In general, we find that the solution of problem (7) by (8) yields a weighted cut which decomposes the original graph into an *equal* number of large and small subgraphs containing almost  $\alpha$  and  $l$  nodes respectively.

The previous discussion leads to the following heuristic technique for solving problem (1) and for obtaining bounds on  $E_c$ .

#### Algorithm 1

- Step 1: Choose the number of desired subgraphs  $k$ , an upper bound  $\alpha$  and lower bound  $l$  on subgraph size.
- Step 2: Solve problem (7) by formula (8) to obtain  $\alpha \geq m_1^* \geq m_2^* \geq \dots \geq m_k^* \geq l$
- Step 3: Find the  $k$  largest eigenvalues  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$  and corresponding eigenvectors  $u_j$  of the admittance matrix  $A$ . A local optimum for problem (1) is found by solving the *one* transportation problem

$$\begin{aligned}
 \text{Min} \quad & - \sum_{j=1}^k \sum_{i=1}^n \left( \frac{u_{ij}}{\sqrt{m_j^*}} \right) x_{ij} \\
 \text{s.t.} \quad & \sum_{j=1}^k x_{ij} = 1 \quad \forall i = 1, 2, \dots, n \\
 & \sum_{i=1}^n x_{ij} = m_j^* \quad \forall j = 1, 2, \dots, k \\
 & x_{ij} \geq 0.
 \end{aligned} \tag{11}$$

- Step 4: Upper and lower bounds on the sum of the edges cut  $E_c$  for problem (1)

are

$$-\frac{1}{2} \sum_{i=1}^k \bar{\lambda}_i m_i \leq E_c \leq E_u(m_1^*, m_2^*, \dots, m_k^*)$$

where  $E_u(m_1^*, \dots, m_k^*)$  is the sum of the edges cut by the  $k$ -decomposition solution obtained by solving problem (11) and  $\bar{\lambda}_1 \geq \bar{\lambda}_2 \geq \dots \geq \bar{\lambda}_k$  are the  $k$  largest eigenvalues of  $\bar{A}$ .

Tighter bounds on  $E_c$  for problem (1) can be obtained in (Barnes, 1982a).

#### 4. Improving an Existing $k$ -Decomposition

In general, a  $k$ -decomposition obtained by Algorithm 1 will not be optimal, even locally. For such a partition, it may be possible to decrease  $E_u(m_1^*, \dots, m_k^*)$  by *interchanging* nodes in the  $k$  subgraphs. Kernighan and Lin (1970) describe an  $O(n^2)$  routine for performing this. However, this technique assumes that the subgraph sizes remains *fixed* and that only *two* subgraphs are considered at one time. This can be very limiting when applied to the bounded subgraph constraints considered in problem (1).

In general, one would like to know if it is possible to interchange subsets of the  $k$  subgraphs to *decrease* the number of interconnections between subgraphs. In this section, we describe a new technique for performing this sequence of interchanges. We take advantage of the bipartite graph structure of GT to accomplish this. The new  $k$ -decomposition improvement method is solved in polynomial time. A linear transportation problem is again solved at each step.

An improved  $k$ -decomposition of the existing  $k$ -decomposition is generated as follows. For each row element  $i \in P_A$ , let  $s(i,j)$  denote the sum of the elements in row  $i$  that are in subgraph  $S_j$ . If row  $i$  is in the  $p$ th subgraph, then let  $a(i) = s(i,p)$ .

We now calculate the change in the sum of the interconnections that result in moving node  $i$  in subgraph  $S_p$  to subgraph  $S_j$ ,  $j \neq p$ . Equivalently, we calculate the net gain in off-diagonal elements in moving row  $i$  in block  $p$  to block  $j$ . This is done by

$$\Delta_R(i,j) = s(i,j) - a(i)$$

Note that  $\Delta_R(i,j) < 0$  implies that by moving node  $i$  to subgraph  $S_j$ , the number of interconnections  $E_u$  is reduced by  $\Delta_R(i,j)$ . Since the graph representation of GT is a bipartite graph, we can determine which nodes can be moved to other subgraphs so that the existing  $k$ -decomposition is improved by keeping the columns fixed. This is obtained by solving the following problem

$$\begin{aligned} \text{rowmin}(\ell) = \quad & \text{Min} \quad \sum_{j=1}^k \sum_{i=1}^m \Delta_R(i,j)x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^k x_{ij} = 1, \quad \forall i = 1, 2, \dots, m \quad (m = |P_A|) \\ & \sum_{i=1}^m x_{ij} \leq a - m_{jR}(\ell), \quad \forall j = 1, 2, \dots, k \\ & x_{ij} \geq 0 \end{aligned} \tag{12}$$

where  $m_{jR}(\ell)$  is the number of nodes in  $P_R$  that are in subgraph  $S_j$  at iteration  $\ell$ . A similar procedure can be found for the columns of  $\bar{B}$ . Letting  $\Delta_c(i,j)$  represent the net gain in off-diagonal elements in moving column  $i$  in block  $p$  to block  $j$ , we can determine which nodes can be moved to other subgraphs so that the existing  $k$ -decomposition is improved by keeping the rows fixed. In this case, we solve the linear



transportation problem

$$\begin{aligned}
 \text{colmin}(\ell) = & \text{Min} \sum_{j=1}^k \sum_{i=1}^n \Delta_c(i,j)y_{ij} \\
 \text{s.t.} & \sum_{j=1}^k y_{ij} = 1, \quad \forall i = 1, 2, \dots, n (n = |P_R|) \\
 & \sum_{i=1}^n y_{ij} \leq n - m_{jA}(\ell), \quad \forall j = 1, 2, \dots, k \\
 & y_{ij} \geq 0
 \end{aligned} \tag{13}$$

where  $m_{jA}(\ell)$  is the number of nodes in  $P_A$  that are in subgraph  $S_j$  at iteration  $\ell$ .

Note that problems (12) and (13) are easy to solve transportation problems. Alternating row and column changes, which most decrease the number of interconnections are performed at each iteration. After a finite number of steps, no improvement will be possible and a local optimum is obtained. The constraints imposed on problems (12) and (13) assume that the bounded subgraph constraints of problem (1) are satisfied. We tie these ideas together in the following algorithm.

#### Algorithm 2

Step 0:  $\ell = 1$

Step 1: Given a  $k$ -decomposition  $\{S_1^{(\ell)}, \dots, S_k^{(\ell)}\}$ , determine its matrix  $\bar{B}$  representation. Calculate

$$\Delta_R(i,j) \triangleq \text{net gain in interconnections in moving row } i \text{ to subgraph } S_j^{(\ell)}$$

$$\Delta_c(i,j) \triangleq \text{net gain in interconnections in moving column } i \text{ to subgraph } S_j^{(\ell)}$$

Step 2: Solve problems (12) and (13).

Step 3: Calculate  $\min \{\text{rowmin}(\ell), \text{colmin}(\ell)\} = c(\ell)$ . If  $c(\ell) = 0$ , go to Step 6.

Step 4: If  $c(l) = \text{rowmin}(l) < 0$ , use solution of problem (12) to permute the appropriate rows of  $\bar{B}$ , otherwise permute the appropriate columns of  $\bar{B}$ .

Step 5:  $l \leftarrow l + 1$

Go to Step 1.

Step 6: The new cut is  $E_u + \sum c(l)$ . The best local  $k$ -decomposition is  $\{S_1^{(l)}, \dots, S_k^{(l)}\}$ .

Algorithm 2 is illustrated on the following example.

#### Example 4.1

Consider the following 7-node example.

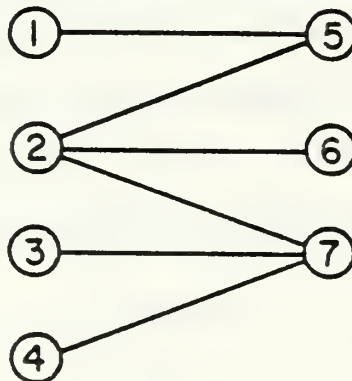


Figure 6

We seek the optimal 2-decomposition where  $l = 2$  and  $n = 4$  in problem 1. Assume that the initial 2-decomposition is:

$$S_1^{(1)} = \{1,4,7\} \text{ and } S_2^{(1)} = \{2,3,5,6\}$$

Step 1:

$$\bar{B} = \begin{array}{cccc|c} & 7 & 5 & 6 & \\ & & \downarrow & & \\ & 0 & 1 & 0 & 1 \\ \hline & 1 & 0 & 0 & 4 \\ & 1 & 1 & 1 & 2 \\ \hline & 1 & 0 & 0 & 3 \end{array}$$

$$\Delta_R = \begin{bmatrix} 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 \end{bmatrix}$$

$$\Delta_c = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}$$

Step 2: The solution of problem (12) is to interchange rows 1 and 4. The solution of problem (13) is to move column 2 to  $S_1^{(1)}$

Step 3: rowmin (1) = -2

colmin (1) = 0

c(1) = -2.

Step 4: Permute rows 1 and 4 of  $\bar{B}$

$$\bar{B} = \begin{array}{cccc|c} & 7 & 5 & 6 & \\ & & \downarrow & & \\ & 1 & 0 & 0 & 3 \\ \hline & 1 & 0 & 0 & 4 \\ & 1 & 1 & 1 & 2 \\ \hline & 0 & 1 & 0 & 1 \end{array}$$

Step 5:  $l \leftarrow 2$ . Go to Step 1.

Step 1:  $S_1^{(2)} = \{3,4,7\}$ ,  $S_2^{(2)} = \{1,2,5,6\}$

$$\Delta_R = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\Delta_c = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Step 2: No rows or columns can be moved by solving (12) and (13)

Step 3: c(2) = 0. Go to Step 6.

Step 6: The new cut is  $E_2 - 2 = 3 - 2 = 1$  and  $S_1^{(2)} = \{3,4,7\}$   $S_2^{(2)} = \{1,2,5,6\}$  is a local optimal 2-decomposition satisfying problem (1) with  $\alpha = 4$ ,  $l = 2$ .

## 5. A Numerical Example

The most difficult aspect of implementing Algorithms 1 and 2 is to determine the linear coefficients of the transportation problems. This is accomplished in Algorithm 1 by finding the  $k$  largest eigenvalues and their corresponding eigenvectors. In the case of very large sparse symmetric matrices (dimension 2000 or greater), an available FORTRAN code (Cullum and Willoughby, 1984b) allows one to handle such problems. An APL code called PUSH is currently being used to find the linear coefficients  $\Delta_R(i,j)$  and  $\Delta_c(i,j)$  for any partition  $\{S_1^{(l)}, \dots, S_k^{(l)}\}$  in Algorithm 2.

We illustrate the use of Algorithms 1 and 2 on the following 23 machine by 20 pack GT problem given in Figure 7 (Groover, 1980). We attempt to find an optimal 2-decomposition of this problem where  $n = 29$  and  $l = 14$ . We solve problem (3) to decompose the graph into two subgraphs containing 19 and 24 nodes. This yields the matrix representation  $B_1$  (20 edges cut). This is shown in Figure 8. Using Algorithm 2, machines 4, 16, 19 and 20 are moved to subgraph  $S_2$  yielding matrix representation B (14 edges cut). Finally, note that pack 42 can be moved to subgraph  $S_2$  yielding 13 edges cut; see Figure 9. We can bound the number of edges cut,  $E_c$  by Algorithm 1

$$6 \leq E_c \leq 13.$$



	1		1	1		1								1															5	
	1		1	1	1	1	1								1														6	
			1	1																									7	
			1	1																									8	
			1	1																									9	
	1		1			1																							13	
	1																												14	
			1					1																					17	
							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
																													1	
							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	
																													3	
																													4	
																													10	
																													11	
																													12	
																													15	
																													16	
																													18	
																													19	
																													20	
																													21	
																													22	
																													23	
25	30	33	34	39	40	42	24	26	27	28	29	31	32	35	36	37	38	41	43											

13 edges cut  
Figure 9

## 6. Conclusions

One of the fundamental design problems faced, when a manufacturing plant is deciding to implement the new technologies of flexible manufacturing systems or group technology, is the question of what parts to produce on which machines. In this paper, using the philosophy of production flow analysis (Burbidge, 1975), we have modelled the grouping of parts/packs and machines as a weighted bipartite graph decomposition problem.

While this formulation is an NP-complete problem, we have developed a two phase polynomially bounded algorithm for approximating the optimal solution. Phase one approximates the original graph partitioning problem by an easily solved linear transportation problem. The output of this algorithm is to be viewed as a good starting solution to the grouping problem, which can be improved by the phase two

algorithm. The phase two algorithm takes advantage of the bipartite graph structure of the GT problem to accomplish this. Both these algorithms have been designed with the large-scale nature of these problems in mind, and are easily implemented on computers.

Given that these algorithms may not be optimal, we have derived bounds on the optimal solution which indicates how good these algorithms perform. This allows the users to decide whether to seek a better solution. Also imbedded in our algorithms is the flexibility for users to perform sensitivity analysis on the number of groups that may be decided.

Further issues that have to be dealt with in this line of research are the consequences of the bottleneck machines (those that are needed by more than one group) and efficient methods to analyze the costs of perfect decomposition or decoupling of these groups. These problems are currently being investigated.

## References

- Barnes, E.R., "Partitioning, Spectra and Linear Programming", *Proceedings of the 25th Anniversary Waterloo Conference in Combinatorics*, (June 1982).
- Barnes, E.R., "An Algorithm for Partitioning the Nodes of a Graph", *SIAM J. of Algebraic Discrete Methods*, 3, pp. 541-550 (1982).
- Burbidge, J.L., *The Introduction of Group Technology*, Halsted Press, John Wiley and Sons, New York; Chapter 9 (1975).
- Cullum, J. and Donath, W.E., "A Block Lancos Algorithm for Computing the  $q$  Algebraically Largest Eigenvalues and a Corresponding Eigenspace of Large, Sparse, Real Symmetric Matrices", *Proc. of the 1974 IEEE Conference on Decision and Control*, pp. 505-509 (1974).
- Cullum, J. and Willoughby, R.A., *Lancos Algorithms for Large Symmetric Matrices, Volume 1*, Birkhouser, Boston, (to appear 1984).
- Donath, W.E. and Hoffman, A.J., "Lower Bounds for the Partitioning of Graphs", *IBM J. Res. and Dev.*, 17, pp. 420-425 (1973).
- El-Essawy, I.F.K. and Torrance, J., "Component Flow Analysis: An Effective Approach to Production Systems", *Production Engineer*, 51, 165 (1972).
- Groover, M.P., *Automation, Production Systems and Computer-Aided Manufacturing*, Prentice-Hall, Englewood Cliffs, N.J.; Chapter 18 (1980).
- Hyde, W.F., *Classification, Coding and Data Base Standardization*, Marcel Dekker Inc., New York; Chapter 7 (1981).
- Kernighan, B.W. and Lin, S., "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell Systems Tech. J.*, 49, pp. 291-307 (1970).
- King, J.R., "Machine-Component Group Formation in Group Technology", *Fifth International Conference on Production Research*, pp. 40-44 (1979).
- \_\_\_\_\_, "Machine-Component Grouping in Production Flow Analysis: An Approach Using a Rank Order Clustering Algorithm", *Int. J. Prod. Res.*, 18, 213 (1980).
- \_\_\_\_\_ and Nakornchai, V., "Machine-Component Group Formation in Group Technology: Review and Extension", *Int. J. Prod. Res.*, 20, 117 (1982).
- Kusiak, A., "The Part Families Problem in Flexible Manufacturing Systems", Working Paper # 06/84, Dept. of Industrial Engineering, Technical University of Nova Scotia (1984).
- Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.



McCormick, W.T., Schweitzer, P.J. and White, T.E., "Problem Decomposition and Data Recognition by a Clustering Technique", *Oper. Res.*, 20, 993 (1972).

Mitrafanov, S.P., *The Scientific Principles of Group Technology*, Leningrad (1959); translated by the National Lending Library, (1966).

Reich, R., "The Next American Frontier", *Fortune*, pp. 97-108, March (1983).

Vannelli, A., "Approximating a Class of Graph Decomposition Problems by Linear Transportation Problems", *TIMS/ORSA Meeting, San Francisco*, May (1984); IBM Research Report RC 10584 (#47380), June (1984).

Vannelli, A. and Vidyasagar, M., "Three Approximate Solution Techniques for the Optimal k-Decomposition Problem", *International Conference on Systems, Man and Cybernetics*, Bombay, India (1984).











HECKMAN  
BINDERY INC.



**JUN 95**

Bound-To-Pleas<sup>®</sup> N. MANCHESTER,  
INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA



3 0112 060296115