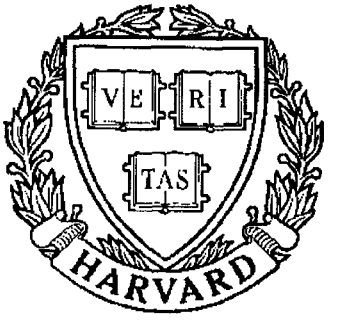
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Generalization and Implementation of the GP Method to Generate Manufacturing Cell and Part Families

by J. Hilger, G. Harhalakis and J.M. Proth

TR 89-67

GENERALIZATION AND IMPLEMENTATION OF THE GP METHOD TO GENERATE MANUFACTURING CELLS AND PART FAMILIES

Jean HILGER[1], George HARHALAKIS[2] and Jean - Marie PROTH[3]

[1] INRIA-LORRAINE, Project SAGEP, Bp 239, Campus Scientifique, Vandoeuvre-les-Nancy, France

[2] Institute of Systems Research and Department of Mechanical Engineering, University of Maryland, USA

[3] INRIA-LORRAINE, Project SAGEP, Bp 239, Campus Scientifique, Vandoeuvre-les-Nancy, France and Institute of Systems Research, University of Maryland, USA

Abstract

We present an algorithm which considers a set of product types and a set of machine types. The algorithm works out a partition of p subsets of product types, called product families, and a partition of q subsets of machine types, called production subsystems such that :

either $p = q$ and there exists a one-to-one relationship between and product families production subsystems ,

or $p = q + 1$ (or $q = p + 1$) and there exists a one-to-one relationship between r product families and production subsystems where r is the minimum value of p and q. The supplementary subset of product (or machine) types has no corresponding subset of machine (or product) types.

In both cases the partitions obtained maximize a criterion which is the weighted sum of normalized processing times of each product family in its related production subsystem and the complements of normalized processing times of each product family outside its related production subsystem. In the latter case the supplementary subset of product (or machine) types contains only products which have insignificant processing times (or machines which are only rarely or briefly involved by product transformation).

We prove the convergence of our algorithm and give some numerical results. The paper is concluded with the description of an implementation of the algorithm for large data sets.

KEY WORDS: Production Management, Group Technology, Cross-Decomposition, Clustering Techniques.

Introduction

At the design stage of a manufacturing system, we usually know the set of product types we have to manufacture along with their production process and the mean proportion of each type of product we expect to manufacture in the future. The goal is to group the machines into cells in order to reduce the production cost by designing an effective production system layout. Many projects have been carried out in this field (see [1], [4], [5], [6], [7], [9], [10], [11], [12], [13]). The process of partitioning the set of machines into cells and the set of products into product families is known as a cross decomposition process. Various approaches are available (see [2], [3], [8] for instance).

This paper extends the result proposed in [3] which is known as the GPM algorithm. This algorithm provides a partition of the set of machine types into manufacturing cells and a partition of the set of product types into product families in such a way that:

- the number of subsets in both partitions is the same,
- a one-to-one relationship between the manufacturing cells and the product types is given.

The above cross-decomposition maximizes a criterion which is the weighted sum of normalized processing times for each product family in its related production subsystem and the complements of normalized processing times of each product family outside its related production subsystem. The objective is to obtain a spatial decomposition of the production system into subsystems. Each of these subsystems manufactures the most important part of the products belonging to the corresponding product families. In addition, operations performed outside the corresponding subsystems are those which require less amount of time.

The GPM provides good results in a short computation time. With this algorithm, however we might have product types with little machine usage in a product family (as well as machine types in a production subsystem which are only briefly or rarely involved in processing product types of the corresponding product family). We call those product types (or machine types) insignificant. Our aim is to improve the GPM by extracting product types or machine types which are insignificant.

The following section briefly exposes the GPM, followed by the modified algorithm and the proof of its convergence. A numerical example is also provided. The last section presents an implementation of the algorithm for large data sets.

1. The Initial Algorithm

1.1. The Data

We consider a matrix A with the processing times of n different product types on m different machine types : $A = [a_{i,j}]$, i = 1 .. n and j = 1 .. m, where $a_{i,j}$ is the processing time of product type i on machine type j. $B = [b_{i,j}]$ is the normalized matrix A, where $b_{i,j} = a_{i,j} / \max_{i,j} (a_{i,j})$. Note that $a_{i,j} = 0$ if the product type i does not require machine type j to be manufactured.

To each product type i we assign a weight u_i which is the number of products of type i manufactured during a given time period and similarly to each machine type j we assign a weight w_j which is the number of machines of type j available for the period considered.

1.2. The Initial Algorithm and Criterion

In this paragraph we briefly revisit the GPM as well as the criterion it is based on.

We denote partitions of machine types by Y and partitions of product types by X. Partitions are computed in the following sequence: X^o, Y^1, X^1, ..., Y^t, X^t, ... where X^o is the randomly chosen initial partition of product types and $Y^t = \{ Y_1^t, Y_2^t, \ldots, Y_r^t \}$, $X^t = \{ X_1^t, X_2^t, \ldots, X_r^t \}$ are the sets of r production subsystems and r product families available at computation step t ($t > 0$).

The initial algorithm presented in [1] constructs a partition Y^t from a partition X^{t-1} and X^t from Y^t. It attemps to maximize the criterion $\Delta(X^t, Y^t)$ given by (1) (see [2]).

$$\Delta(X^t,Y^t) = h \sum_{k=r} \sum_{(i,j)\in \bigcup_{k=1} (X_k^t,Y_k^t)} u_i\, w_j\, b_{i,j}^{\lambda} + (1-h) \sum_{k=r} \sum_{(i,j)\notin \bigcup_{k=1} (X_k^t,Y_k^t)} u_i\, w_j\, (1-b_{i,j})^{1/\lambda} \qquad (1)$$

where $h \in [0, 1]$ is a parameter which gives importance to the values contained in diagonal blocks and $\lambda \in]0, \infty[$ is a parameter which modifies the influence of high or low processing times.

The algorithm converges with stable maximum criterion values [1] since the number of partitions X and Y is finite and we have non decreasing values of criterion $\Delta(X, Y)$ for each

successive couple of partition of r non empty subsets. Figure 1 illustrates the result of the heuristic algorithm.

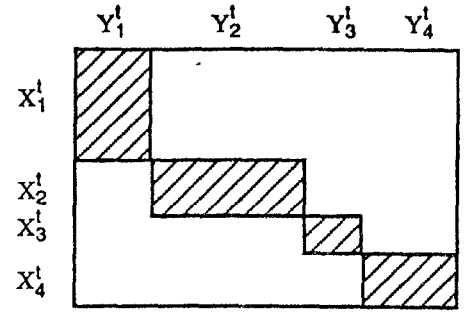


Figure 1. The diagonal blocks formed by the one-to-one related subsets of product and machine types maximize the criterion Δ.

Recall that the result of this algorithm not only depends on h and λ but also on the initial partition X^o

2. The Generalized Algorithm

At each step of the previous algorithm, we compute the additionnal value to the criterion (1) provided by the column (or the row) considered, assuming successively that this column (or row) belongs to the subsets 1, 2, ..., r of the partition. We assign the column (or the row) to the partition which leads to the maximal adding value.

In the generalized algorithm, we consider one more case : the case of the column (or the row) not belonging to any of the existing subsets of the partition. If this assumption leads for some columns (or rows) to higher adding values to the criterion, then we assign them to a supplementary subset. If we compute successively two identical partitions with the same supplementary subset then we conclude that the vectors belonging to this subset are insignificant. Insignificant vectors are excluded from the matrix under consideration and the computation continues with the remaining matrix.

In sub-section 2.1 we show how to assign a vector (column or row) to a subset of the partition. Sub-section 2.2 is devoted to the generalized cross-decomposition algorithm. In sub-section 2.3 we prove the convergence of the previous algorithm.

2.1. Assignment of Vectors to Subsets

Partitions are computed by assigning of vectors to subsets. For instance, the machine partition Y^t is computed starting from the product type partition $X^{t-1} = \{ X_1^{t-1}, ..., X_r^{t-1}\}$: we assign each machine type j (j = 1, ..., m) either to the subset Y_k^t (k = 1, ..., r) related to X_k^{t-1} or to the supplementary subset Y^t_{r+1} as follows.

We evaluate the assignment hypothesis by c^j_Y (k) : " j is assigned to subset Y_k^t ", for k = 1, ..., r.

$$c^j_Y(k) = w_j \; h \sum_{i \in X_k^{t-1}} u_i (b_{i,j})^{\lambda} + w_j \; (1 - h) \sum_{i \notin X_k^{t-1}} u_i (1 - b_{i,j})^{1/\lambda} \qquad (2)$$

We evaluate the hypothesis by c^j_Y (s) : "j is assigned to to subset Y_s^t ", with s = r+ 1.

$$c^j_Y(s) = w_j \; (1 - h) \sum_{i=1}^{n} u_i \; (1 - b_{i,j})^{1/\lambda} \qquad (3)$$

Let $$L(Y^t, j) = \{ l / \; c^j_Y(l) = \underset{k = 1, ..., s}{\text{Max}} \; c^j_Y(k) \}. \qquad (4)$$

We assign j to subset $l^*(j, t)$ such that $l^*(j, t) = \text{Min } l$. $\qquad (5)$

$$l \in L(Y^t, j)$$

If $l^*(j, t) = s$, we assign j to a new supplementary subset of machine types Y_s^t . Otherwise j is assigned to a production subsystem $l^*(j, t)$ related to a product family X_k^{t-1}. Once all machine assignments have been decided, we evaluate the resulting machine type partition Y^t by a global criterion W_Y^t.

$$W_Y^t = \sum_{j=1}^{m} c^j_Y(l^*(j, t)) \qquad (6)$$

W_Y^t is the value of the criterion and $W_Y^t = \Delta(X^{t-1}, Y^t)$ if there is no supplementary subset in Y^t (i.e. $Y^t = \phi$).

Similarly, we will use the following formulas to compute a partition X^t of product types from a partition Y^t of machine types :

$$c^i_X(k) = u_i \; h \sum_{j \in Y_k^t} w_j b_{i,j}^{\lambda} + u_i \; (1-h) \sum_{j \notin Y_k^t} w_j (1 - b_{i,j})^{1/\lambda}, \quad \text{for } k = 1, \ldots, r \qquad (7)$$

$$c^i_X(s) = u_i \; (1-h) \sum_{j=1}^{m} w_j \; (1 - b_{i,j})^{1/\lambda} \quad , \text{where } s = r + 1 \qquad (8)$$

The global criterion $W_X{}^t$ evaluates the partition X^t:

$$W_X{}^t = \sum_{i=1}^{n} c^i_X(l^*(i, t)) \qquad (9)$$

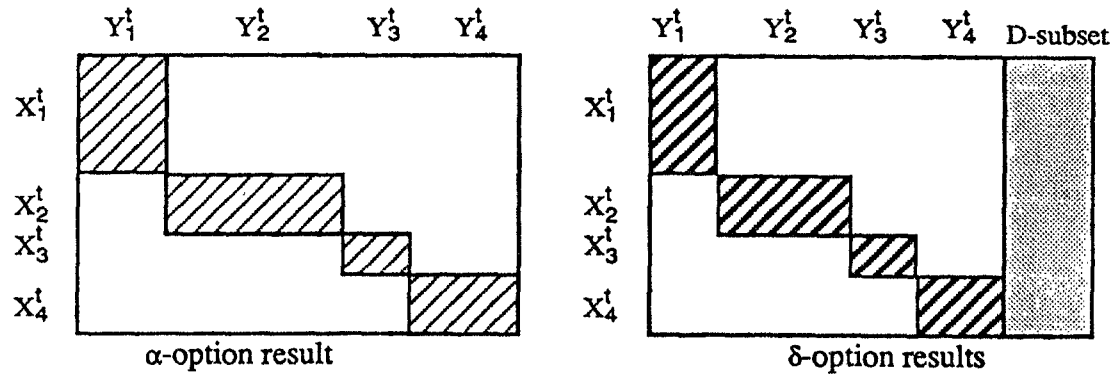
$l^*(i, t)$ being defined in a similar way to $l^*(j, t)$. (10)

2.2. The Algorithm

Three different options are available to run the following algorithm :

- the α-option under which the algorithm behaves exactly as the initial GPM algorithm and computes one-to-one related subsets,
- the β-option which allows supplementary subsets for product types and
- the δ-option which symmetrically allows supplementary subsets for machine types.

In the following, we call D-subset the supplementary subset of vectors in the final layout.The D-subset is the union of insignificant vectors encountered along the computation.



Algorithm:

X^o is the initial product partition resulting from a random assignment of product types to r subsets, with $r \leq \min(n,m)$.

convergence = FALSE,

t = 1,

$n^t = n$, $m^t = m$,

$D^t = \phi$, $W_X{}^t = 0$, $Y = \phi$, $X = \phi$,

select between options α, β, δ.

do

s = Card(X^{t-1})+1

for j = 1 .. m^t /* *machine partitioning* Y^t *from* X^{t-1} */

if (δ) then $L(Y^t, j) = \{ l / c^j_Y (l) = \text{Max}_{k=1,\ldots,s} c^j_Y (k) \}$.

else $L(Y^t, j) = \{ l / c^j_Y (l) = \text{Max}_{k=1,\ldots,(s-1)} c^j_Y (k) \}$.

assign j to subset $Y^t_{l^*}$ such that $l^* = \text{Min}_{l \in L(Y^t, j)} l$

if (α) or (δ) then **decide_convergence_machine ()**

s = Card(Y^t)+1

for i = 1 .. n^t /* *product partitioning* X^t *form* Y^t */

if (β) then $L(X^t, i) = \{ l / c^i_X (l) = \text{Max}_{k=1,\ldots,s} c^i_X (k) \}$.

else $L(X^t, i) = \{ l / c^i_X (l) = \text{Max}_{k=1,\ldots,(s-1)} c^i_X (k) \}$.

assign j to subset $X^t_{l^*}$ such that $l^* = \text{Min}_{l \in L(X^t, i)} l$

if (α) or (β) then **decide_convergence_product ()**

t = t + 1

until convergence = TRUE

decide_convergence_machine ( )

if $Y_s^t \in Y^t$ then

if $Y^t \in Y$ then

$m^{t+1} = m^t - \text{Card}\,(Y_s^t)$

remove all column vectors $[j] \in Y_s^t$ from B

$D^t = D^t \cup Y_s^t, Y = \phi$

else $Y = Y \cup Y^t$

else $m^{t+1} = m^t$.

if $t > 1$ and $(X^{t-2}, Y^{t-1}) \equiv (X^{t-1}, Y^t)$ then convergence = TRUE

decide_convergence_product ( )

if $X_s^t \in X^t$ then

if $X^t \in X$ then

$n^{t+1} = n^t - \text{Card}\,(X_s^t)$

remove all row vectors $[i] \in X_s^t$ from B

$D^t = D^t \cup X_s^t, X = \phi$

else $X = X \cup X^t$

else $n^{t+1} = n^t$

if $t > 1$ and $(Y^{t-1}, X^{t-1}) \equiv (Y^t, X^t)$ then convergence = TRUE

Let us consider a set of partitions obtained from the above algorithm. Suppose that two of them, say Y^t and Y^l $(t < l)$, are identical and contain the same supplementary subset Y_s^t $(= Y_s^l)$. In this case, the supplementary subset is canceled and the computation restarts with the remaining matrix. The canceled subset is called a D-subset. Several supplementary subsets can be found in the computation . They are grouped in the final D-subset.

2.3. Convergence

We prove the convergence of our algorithm for all three options.

Lemma 1 Assuming that α-option has been chosen the following property holds:

$\text{card}(X^0) \geq \text{card}(Y^1) \geq \text{card}(X^1) \geq \ldots \geq \text{card}(Y^t) \geq \text{card}(X^t) \geq \ldots$ ❑ (11)

The first lemma says that the number of subsets in the partitions does not increase along with the computation.

Proof a. Let us start from X^t ($t \geq 0$). In order to compute partition Y^{t+1}, the algorithm computes for each column as many values of the criterion as the number of subsets in X^t and assigns the column to the k-th subset, if the computed k-th value is one of the highest values of the criterion. Thus, there are at most as many subsets in Y^{t+1} as in X^t. In other words, $card(X^t) \geq card(Y^{t+1})$.

b.We can use a similar argument to show that $card(Y^t) \geq card(X^t)$ for $t > 0$.

c. Combining results a. and b., we obtain relation (11). Q.E.D.

Lemma 2 Assuming that α-option has been chosen, the following property holds :

if we have $card(X^t) = card(Y^{t+p})$, $t \geq 0$, $p > 0$ then

$$W_{Y^{t+1}} \leq W_{X^{t+1}} \leq W_{Y^{t+2}} \leq \ ... \leq W_{Y^{t+p}} \leq W_{X^{t+p}}. \quad \square \qquad (12)$$

Lemma 2 shows that, if the number of subsets in the partition remains unchanged after some consecutive steps, then the value of the criterion does not decrease.

Proof

a. According to lemma 1, $card(X^t) = card(Y^{t+p})$ leads to:

$$card(X^t) = card(Y^{t+1}) = card(X^{t+1}) = ... = card(Y^{t+p}) = card(X^{t+p}) = ... \qquad (13)$$

b. Because α-option has been chosen :

$$W_{Y^{k+1}} = \Delta\,(X^k, Y^{k+1}) \text{ for } k = t, t+1, ..., t+p-1 \qquad (14)$$

On the other hand $\Delta\,(X^k, Y^{k+1}) \leq \Delta\,(X^{k+1}, Y^{k+1})$ because partition X^{k+1} is computed such that:

$$\Delta\,(X^{k+1}, Y^{k+1}) = \underset{X \in E}{\text{Max}}\ \Delta\,(X, Y^{k+1}) \qquad (15)$$

where E is the set of all partitions of the matrix rows whose dimension is $card(X^{k+1})$.

But $$\Delta\,(X^{k+1}, Y^{k+1}) = W_{X^{k+1}} \qquad (16)$$

From (14), (15) and (16) we derive :

$$W_{Y^{k+1}} \leq W_{X^{k+1}} \text{ for } k = t, t+1, ..., t+p-1. \qquad (17)$$

c. We can use a similar argument to prove that

$$W_{X^k} \leq W_{Y^{k+1}} \text{ for } k = t+1, ..., t+p-1. \qquad (18)$$

d. Finally, from (17) and (18) we derive :

$$W_{Y^{t+1}} \leq W_{X^{t+1}} \leq W_{Y^{t+2}} \leq \ ... \leq W_{Y^{t+p}} \leq W_{X^{t+p}}. \quad \text{Q.E.D.}$$

Lemma 3 Assuming that the α-option has been chosen and that:

$$card(X^t) = card(Y^{t+p}),\ t \geq 0,\ p > 0 \qquad (19)$$

and $$W_{Y^{t+1}} = W_{X^{t+p}},\ t \geq 0,\ p > 0 \qquad (20)$$

then $l^*(j, k+1) \leq l^*(j, k)$, for $j = 1, .., m$ and $t+1 \leq k \leq t+p-1$ (21)

and $l^*(i, k+1) \leq l^*(i, k)$, for $i = 1, .., n$ and $t+1 \leq k \leq t+p-2$. ❑ (22)

As a consequence, if (19) and (20) hold, then :

- either $(X^t, Y^{t+1}) \equiv (X^{t+p-1}, Y^{t+p})$
- or $(X^t, Y^{t+1}) \not\equiv (X^{t+p-1}, Y^{t+p})$ and in that case (X^t, Y^{t+1}) cannot be obtained again.

Proof

a. From lemma 1 and 2, relations (19) and (20) lead to:

$$card(X^t) = card(Y^{t+1}) = card(X^{t+1}) = \ldots = card(X^{t+p}) \quad (23)$$

and $$W_{Y^{t+1}} = W_{X^{t+1}} = \ldots = W_{X^{t+p-1}} = W_{Y^{t+p}} \quad (24)$$

b. Because the α-option has been chosen and (23) holds:

$$W_{Y^{k+1}} = \Delta\,(X^k, Y^{k+1}) \text{ for } k = t, t+1, \ldots, t+p-1 \quad (25)$$

and $$W_{X^{k+1}} = \Delta\,(Y^{k+1}, X^{k+1}) \text{ for } k = t, t+1, \ldots, t+p-2 \quad (26)$$

Taking into account (24), relations (25) and (26) lead to:

$$\Delta\,(X^k, Y^{k+1}) = \Delta\,(Y^{k+1}, X^{k+1}) \quad (27)$$

Thus, partitions X^{k+1} and X^k lead to the same value of the criterion when Y^{k+1} is fixed. But according to (5), X^{k+1} is obtained by assigning the rows of the matrix to the subsets with the lowest rank.

From (22) : $l^*(i, k+1) \leq l^*(i, k)$, for $i = 1, \ldots, n$ and $t+1 \leq k \leq t+p-2$.

c. Using a similar proof, we can write successively:

$$W_{X^{k+1}} = \Delta\,(Y^{k+1}, X^{k+1}), \text{ for } k = t, t+1, \ldots, t+p-2$$

and $$W_{Y^{k+2}} = \Delta\,(X^{k+1}, Y^{k+2}), \text{ for } k = t-1, t, t+1, \ldots, t+p-1$$

Consequently (see (24)):

$$\Delta(X^{k+1}, Y^{k+2}) = \Delta(X^{k+1}, Y^{k+1}) \text{ for } k = t, t+1, \ldots, t+p-2$$

Remember that Y^{k+2} is derived from X^{k+1} in the algorithm.

Thus: $l^*(j, k+2) \leq l^*(j, k+1)$, for $k= t, t+1, \ldots, t+p-2$

or $l^*(j, k+1) \leq l^*(j, k)$, for $k= t, t+1, \ldots, t+p-1$, which is relation (21).

d. Let us first assume that

$$l^*(j, k+1) = l^*(j, k) \text{ for } j = 1, \ldots, m \text{ and } k = t, t+1, \ldots, t+p-2$$

and $$l^*(i, k+1) = l^*(i, k) \text{ for } i = 1, \ldots, n \text{ and } k = t, t+1, \ldots, t+p-1.$$

In that case $(X^t, Y^{t+1}) \equiv (X^{t+p-1}, Y^{t+p})$.

Assume that $l^*(j, k+1) < l^*(j, k)$ for at least one $j \in \{1, \ldots, m\}$ and one $k \in \{t, t+1, \ldots, t+p-2\}$ or $l^*(i, k+1) < l^*(i, k)$ for at least one $i \in \{1, \ldots, n\}$ and one $k \in \{t, t+1, \ldots, t+p-1\}$.

Thus there exists at least one column and/or one row of the matrix classified in a subset of the partition Y^{t+p} (or X^{t+p-1}) whose rank is lower than the one of the subset containing the same

column (and/or row) in partition Y^{t+1} (or X^t). Knowing that the rank of the subset of a partition containing a given row (or column) does not increase along with the computation when (19) and (20) hold, the proof is completed. Q.E.D.

Theorem 1 Algorithm converges when α-option is chosen. □

Proof

The algorithm ends with $W_X{}^{k-1} = W_X{}^k$ or $W_Y{}^k = W_Y{}^{k+1}$, for $k > 1$.

We call step of the algorithm the computational process which leads from a pair (X^k, Y^{k+1}) to the next one (X^{k+1}, Y^{k+2}). At each step of the algorithm, the number of subsets in the partition either do or do not change.

a. In the first case, the value of the criterion either increases or not

a1. When the value of the criterion increases, it increases by a strictly positive value.

a2. When the value of the criterion remains unchanged, partitions do or do not change.

a11. When partitions change, theorem 1 shows that it is impossible to find again the same couple of partitions.

a12. When partitions remain unchanged the algorithm stops.

b. If the number of subsets in the partition changes, they can only decrease, because we are in the α-option situation.

Remembering that the value of the criterion is bounded and that the number of different partitions with a given number of subsets is limited and that we always reach the state described in a12, the previous items show that the algorithm converges when the α-option is chosen. Q.E.D.

Theorem 2 Algorithm converges when the β and δ-options are chosen. □

Proof

We have the following two cases :

a. Two consecutive pair of partitions are found to be identical : the algorithm stops.

b. Else, since the number of pairs of partitions is finite, we necessarily find two identical pairs (X^{t-1}, Y^t) and (X^{l-1}, Y^l) $(l > t)$ containing a supplementary subset. In this case, we cancel the supplementary subset and continue the computation. The number of rows and columns being finite, the algorithm stops either because we reach the above case a or because the matrix vanishes.

Q.E.D.

3. Numerical examples

In this section, we compute with data generated at random. The values of the elements of the matrix belong to [0, 1] . They are randomly generated with a probability of 0.5 that the value of the

element is 0. Thus, the number of 0 values is close to the number of 1 values in the matrix.

We solve this problem using the α-option and the δ-option for various values of parameters h and λ. For each pair of those values, we generate a random initial partition and compute the number of one-to-one related subsets (column P) and the criterion value (column Δ) obtained with the α-option. Also, we generate the number of one-to-one related subsets (column Q), the size of the D-subset (column D) and the criterion value (columns W, CR and T) obtained using the δ-option. In column Δ we determine criterion Δ (X,Y) defined by relation (1). In column W we determine criterion W_Y (see relation 6). In column CR we introduce the values of the criterion for the D-subset computed as follows

$$CR(D) = (1-h) \sum_{j=1}^{md} \sum_{i=1}^{nd} u_i w_j (1- d_{i,j})^{1/\lambda} \qquad (28)$$

where $D = [d_{i,j}]$, $i = 1 .. n_d$ and $j = 1.. m_d$ is the matrix reduced to the D-subset. Column T concerns the sum W+CR.

These results were computed for a matrix with 100 rows and 45 columns:

		α–option		β– option				
h	λ	P	Δ	Q	D	W	CR	T
0.2	0.2	20	2136.0	20	1	2084.4	51.6	2136.1
0.2	0.5	24	2424.6	24	0	2424.6	0	2424.6
0.2	0.8	21	2614.7	21	4	2375.2	241.6	2616.8
0.2	1.1	21	2752.7	0	45	0	2744.0	2744.0
0.2	1.4	1	187.0	0	45	0	2851.1	2851.1
0.2	1.7	1	166.2	0	45	0	2934.3	2934.3
0.4	0.2	15	1666.6	15	0	1666.6	0	1666.6
0.4	0.5	24	1860.6	24	0	1860.6	0	1860.6
0.4	0.8	26	1998.0	27	1	1951.9	44.8	1996.7
0.4	1.1	29	2097.8	29	0	2097.8	0	2097.8
0.4	1.4	26	2171.1	25	2	2070.7	99.7	2170.4
0.4	1.7	24	2226.3	22	7	1874.3	350.5	2224.8
0.6	0.2	7	1262.5	7	0	1262.5	0	1262.5
0.6	0.5	13	1326.0	13	0	1326.0	0	1326.0
0.6	0.8	21	1386.4	21	0	1386.4	0	1386.4
0.6	1.1	24	1453.9	24	0	1453.9	0	1453.9
0.6	1.4	24	1495.5	24	0	1495.5	0	1495.5
0.6	1.7	24	1526.5	24	0	1536.5	0	1526.5
0.8	0.2	1	1508.3	1	0	1508.3	0	1508.3
0.8	0.5	1	1201.2	1	0	1201.2	0	1201.9
0.8	0.8	3	920.5	3	0	920.5	0	920.5
0.8	1.1	9	853.7	9	0	853.7	0	853.7
0.8	1.4	12	849.1	12	0	849.1	0	849.1
0.8	1.7	14	851.9	14	0	851.9	0	851.9

4. Implementation of the Algorithm

In this paragraph we present a dynamic storage technique for the processing time matrix as well as the required data structure for simultanous computation of S solutions.

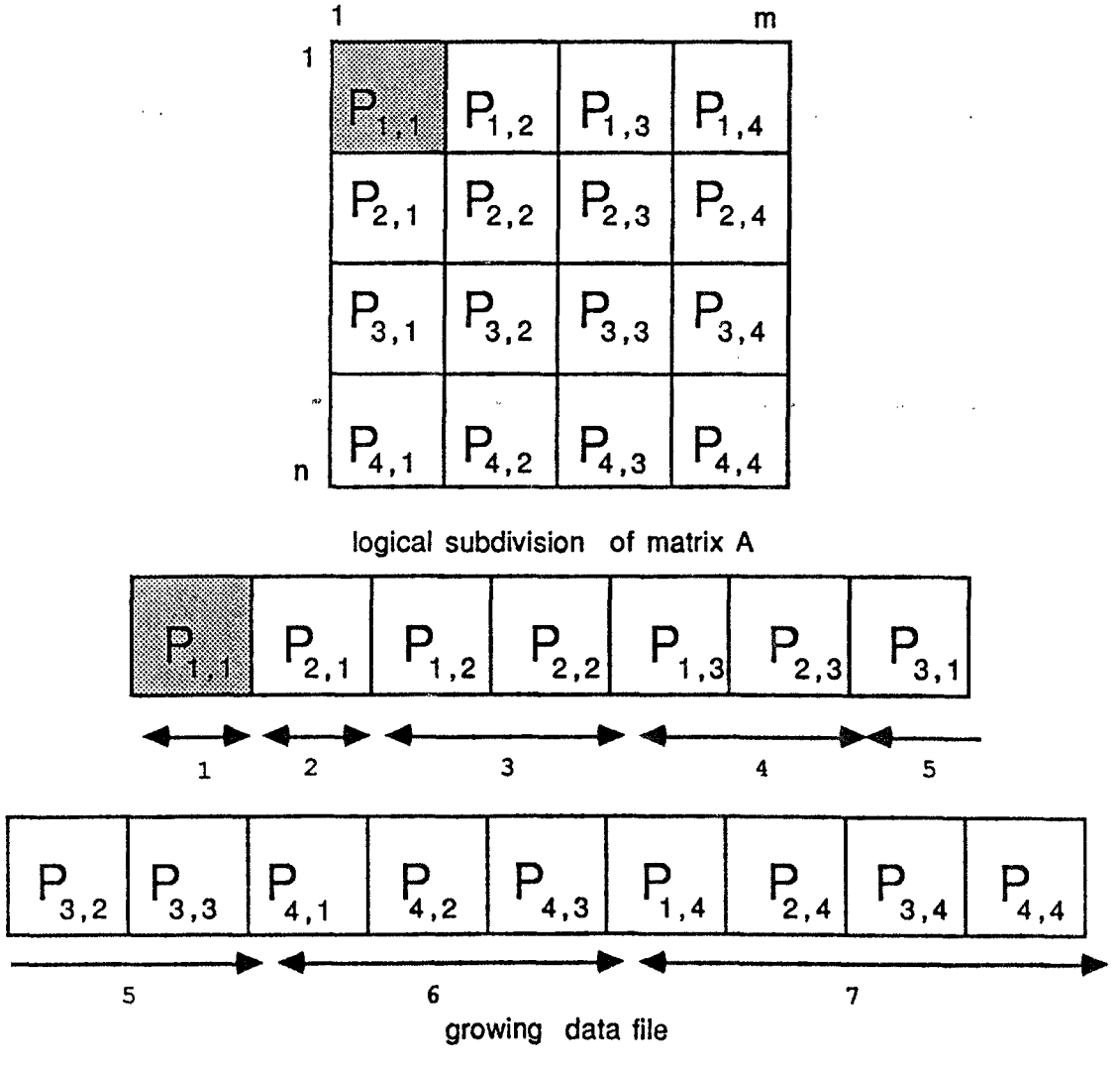
4.1. Data

4.1.1. Dynamic storage technique

The storage technique of the processing time matrix permits dynamical growth of the matrix, allows rapid access to vector data and limits overdimensioning of the stored matrix to a constant factor r.

The processing time matrix $A = [a_{i,j}]$, with $i = 1 \ldots n$ and $j \ldots m$ is subdivided into blocks. Let us call P the matrix of blocks where each block has r rows and r columns:

$P = [p_{l,k}]$ with $l = 1 \ldots ln$, $k = 1 \ldots km$, $ln = n/r$ and $km = m/r$

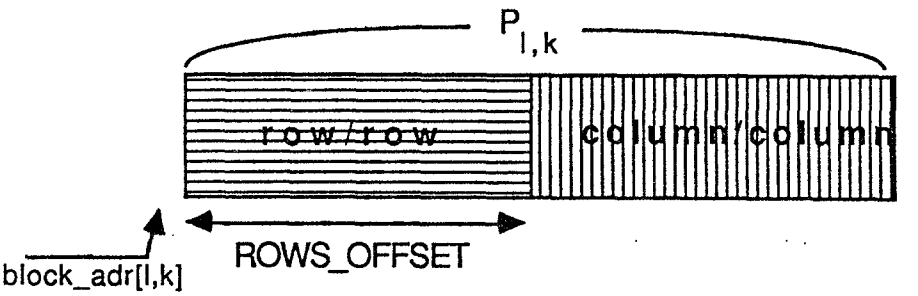


At the beginning of the data storage we create block $p_{1,1}$ in the disk-file. A matrix of r rows and columns is now available. If we add columns to the data set and pass beyond the limit of the r available columns then block $p_{1,2}$ is added to the file. An overun of the limit of r available rows implies the creation of two successive blocks, $p_{2,1}$ and $p_{2,2}$ in order to complete the matrix.

The above figures show the logical subdivision of matrix A and its storage in a disk file when columns and rows are added to the matrix in an arbitrary order. The series 1, 2, 3, 4, 5, 6, 7 of blocks are created in order to complete the matrix, when overruns of limits of available vectors occur.

3.1.2. Data Access

In order to work efficiently the algorithm needs direct access to row and column vectors of the matrix. Each block of r^2 elements is first stored first row-wise and then column-wise.



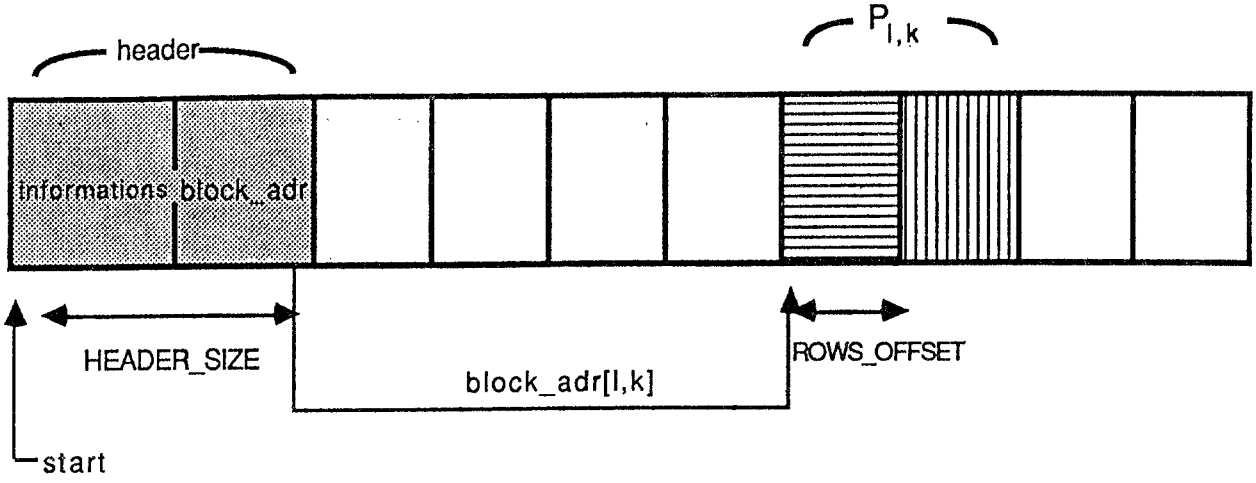
The location of a block $p_{l,k}$ in the file is a relative address block_adr[l, k] from the beginning of the block series. The location of the columns of block $p_{l,k}$ is given by the sum of the block address (block_adr[l, k]) and the storage size of rows in the block (ROWS_OFFSET).

We define:

block_adr [l, k] = - 1 if block $p_{l,k}$ does not exist
else relative address of block $p_{l,k}$

The relative address block_adr [l, k] is known at block creation at the end of the file. Relative block addresses are retained in a matrix of addresses stored in the file header. Beneath the addresses of blocks the file header contains the following information : a data identifier, the last partition obtained and its parameter values, the number of rows and columns of matrix A, the maximum processing time, the machine and product type names, the machine and product type weights and comments on the data set.

During the computation the header data are loaded into memory.



We define :
- ELE_SIZE : storage size of floating point numbers
- COLUMN_SIZE : r * ELE_SIZE
- ROWS_OFFSET : r^2 * ELE_SIZE
- HEADER_SIZE : storage size of file header

Access to column j of matrix A is performed by the following algorithm.

```
k = j / r, l = 1
column_offset = ((j mod r) * COLUMN_SIZE)+ ROWS_OFFSET
while block_adr[l, k] ≠ - 1 do
    position_in (block_adr[l, k] + column_offset + HEADER_SIZE)
    read_vector (column_vector[r * (l - 1) ... r * l])
    l = l + 1
done
```

The presented data access method permits the reading of a vector by direct access to vector segments stored at constant offset in blocks in the disk file. By choosing factor r such that r * ELE_SIZE corresponds to the size of system input / output buffers as well as to the file system block size, we are able to optimize data access under the given constraints of data representation.

4.2. The Implemented Algorithm

The implemented algorithm computes first S different initial product partitions X^o. Row vectors of matrix A are randomly assigned to p subsets, where p is either user defined or equivalent to the minimum dimension of matrix A.

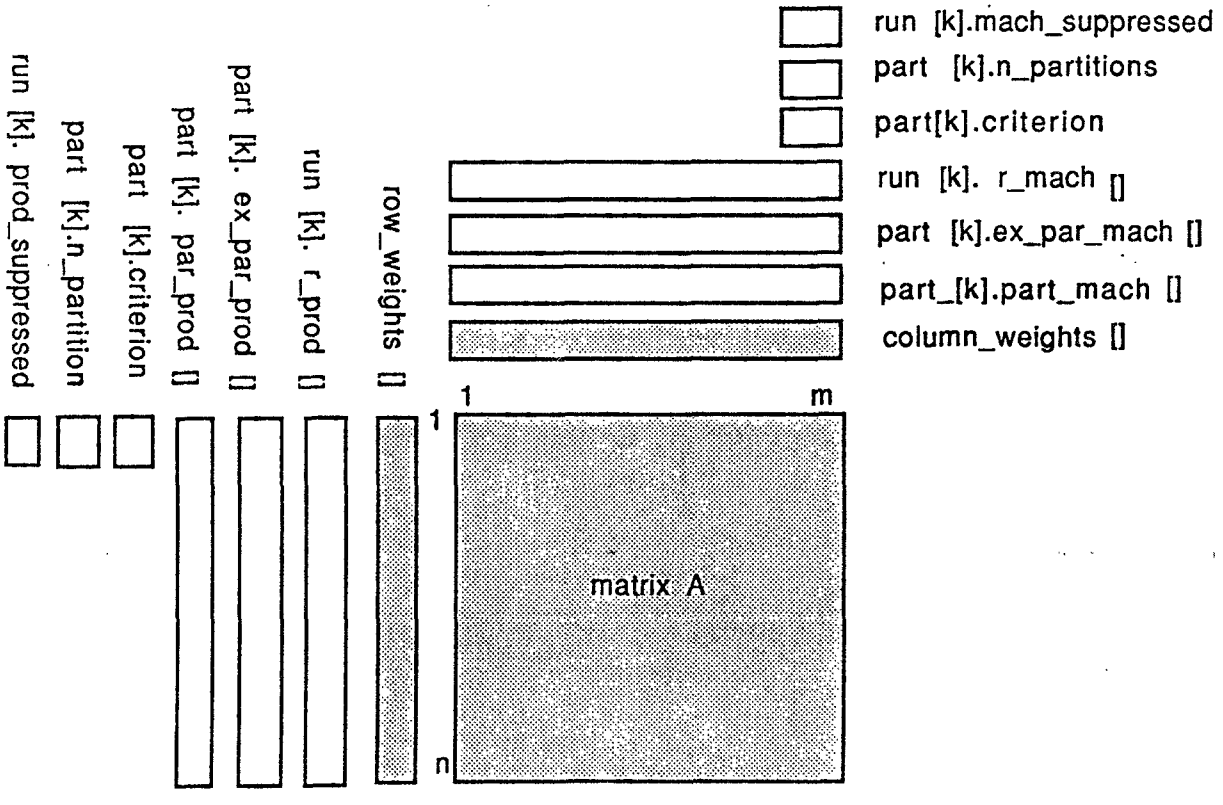
intial_partition()

```
/* no row vector is marked */
for i = 1to p
      select randomly j ∈ [1 ... n]
      if j is marked    then   select next j which is not marked
      mark j  by i

for j = 1 to n
      if j is not marked then   select randomly r ∈ [1 ... p] mark j  by  r

/* all row vectors are assigned to subsets, none of the p subsets are empty */
```

Then we simultaneously run S computations of partitions on choice under α, β or δ-option. For each vector read from the disk-file we compute its criterion values for the S different partitionings.



Required data structures for S simultanious computations

The data structures represented in the above figure are defined as follows :

For each partition k = 1 ... S we define:

part [k].n_partitions	number of existing one-to-one related subsets
run [k].prod_suppressed	number of rows belonging to the D-subset

run [k].mach_suppressed	number of columns belonging to the D-subset
part [k].criterion	value of working criterion W
run [k].r_prod [i]	0 if product type i belongs to the D-subset, 1 else
run [k].r_mach [j]	0 if machine type j belongs to the D-subset, 1 else
part [k].ex_par_prod [i]	1 if product subset i exists, 0 else
part [k].ex_par-mach [j]	1 if machine subset j exists, 0 else
part [k].part_prod [i]	r > 0 if product type i belongs to subset r, 0 if product type i does not belong to any existing subset
part [k].part_mach [j]	r > 0 if machine type j belongs to subset r, 0 if machine type j does not belong to any existing subset

4. Conclusion

In this paper we have proposed a method providing a set of machine subsets and a set of product type subsets as well as a one-to-one relationship between these subsets. Insignificant machines or product types are isolated in a so-called supplementary subset.

The aim of this algorithm used in group technology is to simplify scheduling problems by dividing these problems into smaller subproblems.

The following two issues are subject to further study. The simplest issue is to acquire a machine as many times as it is used by different computed product families. Otherwise we have to look for a way to integrate information about processing sequences of products in the criterion evaluation such that links (or product moves) between production subsystems are minimized.

References

[1] ASKIN R. and SUBRAMANIAN S.B., "A Cost-Based Heuristic for Group Technology Configuration", International Journal of Production Research, 25, 1, pp. 101-113, 1987.

[2] GARCIA H. and PROTH J.M., "A New Cross-Decomposition Algorithm : the GPM. Comparison with the Bond Energy Method", Control and Cybernetics, n° 2, vol. 15, pp. 115-165, 1986.

[3] GARCIA H. and PROTH J.M., "Group Technology in Production Mangement : the Short Horizon Planning Level", Applied Stochastic Models and Data Analysis, n° 1, pp. 25-34,

1985.

[4] KING J.R., "Machine-Component Group Formation in Group Technology", OMEGA The International Journal of Management Science, 8, 2, pp. 193-199, 1979.

[5] KUMAR R.K., KUSIAK A. and VANNELLI A., "Grouping of Parts and Components in Flexible Manufacturing Systems", European Journal of Operations Research, 24, pp. 387-397, 1986.

[6] KUSIAK A., "The Part Families Problem in Flexible Manufacturing Systems", Annals of Operations Research, 3, pp. 279-300, 1985.

[7] Mc AULEY J., "Machine Grouping for Efficient Production", The Production Engineer, pp. 53-57, Feb. 1972.

[8] Mc CORMICK W.T., SCHWEITZER P.J. and WHITE T.E., "Problem Decomposition and Data Reorganization by a Cluster Technique", Operations Research, 20, 1972.

[9] HARHALAKIS G., NAGI R. and PROTH J.M., "An Efficient Heuristic in Manufacturing Cell Formation for Group Technology Applications", accepted for publication by International Production Research .

[10] PORTMANN M.C. and PROTH J.M., "Spatial and Temporal Decomposition Methods in Production Management", International Conference on Computer Integrated Manufacturing, CMP/CIM Computer Society of the IEEE, May 23-25, 1988.

[11] PORTMANN M.C. and PROTH J.M., "A Cross-Decomposition Method for Layout Systems and Scheduling Problem", Third International Conference on CAD/CAM, Robotics and Factories of the Future, CARS and FOF Conference and Exhibits, August 14-17, 1988, Southfield, Mich.

[12] PORTMANN M.C.,"Methodes de décomposition spatiale et temporelle en ordonnancement", Thèse d'Etat ès-sciences mathématiques, Université Nancy I, sept. 1987.

[13] HARHALAKIS G., HILGER J., NAGI R., PROTH J.M., "Formation of Manufacturing Cells : an Algorithm for Minimizing Inter-cell Traffic", proposed for publication to the Journal of Applied Stochastic Models and Data Analysis.