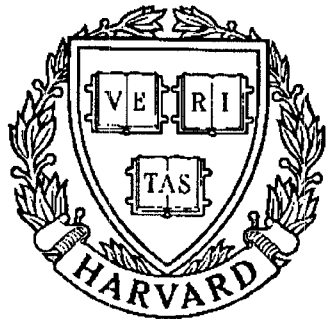
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



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

Manufacturing Cell Formation in the Case of Multi-Manufacturing Processes

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

TR 90-67

MANUFACTURING CELL FORMATION IN THE CASE OF MULTI-MANUFACTURING PROCESSES

G. HARHALAKIS [1] J. HILGER [2] J.M. PROTH [3]

[1] Systems Research Center and Department of Mechanical Engineering, University of Maryland, USA
[2]INRIA-Lorraine, Project SAGEP, BP 239, Campus Scientifique, Vandoeuvre-les-Nancy, FRANCE
[3] INRIA-Lorraine, Project SAGEP, BP 239, Campus Scientifique, Vandoeuvre-les-Nancy, FRANCE
and Associate member of CIM-LAB of the Systems Research Center and Department of Mechanical Engineering, University of Maryland, USA

ABSTRACT

The determination of a good decomposition of a manufacturing system into manufacturing cells, when various manufacturing processes are available for each type of product, is addressed in this paper. We propose a simple twofold algorithm. The first part of the algorithm aims at defining the proportion of each product type to manufacture, using each of the available manufacturing processes. The result is a mono-manufacturing process problem, i.e. a problem which consists of finding a good decomposition of a manufacturing system into cells when only one manufacturing process is available for each type of product. The second part of the algorithm uses an approach already presented by the authors to solve the mono-manufacturing process problem. We also present a numerical example to illustrate our approach.

1. INTRODUCTION

The layout problem of manufacturing systems is usually divided into three consecutive sub-problems :
(i) The first one aims at splitting the manufacturing system into sub-systems (also called manufacturing cells or, in some specific applications, departments), each of them being assigned to a subset of product types (also called product families). This problem has been extensively studied. The algorithms obtained can be classified into three major types.

The first includes algorithms which attempt to maximize the number of operations performed within cells corresponding to part families. The bound energy method [1], the rank order clustering method [2], the direct clustering algorithm [3], and the GPM method [4,5,6] belong to this first type of algorithms.

The second type of algorithms use a progressive aggregation of resources, depending on the degree of "similarity" between these resources. This type includes the single linkage algorithm [7], the algorithm derived from the method based on the Average Common Part Weighing Metric [8] and the Knowledge Based Group Technology (KBGT) system [9,10]. Note that no explicitly defined objective function appears in the last three algorithms.

The third type of algorithms concerns those which aim at minimizing the inter-cell traffic [11,12,13].

To our knowledge, the general problem (i.e. the problem which considers the case of various alternative manufacturing processes being available for some product types) has never been solved when the goal is to optimize a well defined objective function.
(ii) The second sub-problem consists in planning the overall layout of manufacturing cells within the plant. Some information about this step can be found in [14], [15], [16], and [17].
(iii) The last sub-problem addresses the detailed layout of machines within cells. In most cases, this is a highly subjective process.

The results presented in this paper concern the design of manufacturing cells and part families in the most general case of one or more manufacturing processes being available for the same product type. The paper is organized as follows. Section 2 provides the notations and definitions used in the following sections. In section 3, we propose an algorithm that defines the proportion of each product type to be manufactured, using each of the available manufacturing processes. It leads to a mono-manufacturing process problem. In section 4, we recall the general algorithm developed in [6] to solve the second part of the problem. Finally, section 5 is devoted to the presentation of a numerical example.

2. DEFINITIONS AND NOTATIONS

The following basic notations are used in this paper :
m is the number of available machines denoted by j = 1, 2, 3, ..., m ;
n is the number of product types denoted by i = 1, 2, 3, ..., n ;
k(i) is the number of manufacturing processes available to manufacture product of type i.

We also introduce a matrix A :

$[A] = [a^{s}_{i,j}]$, i = 1, 2, ..., n ; s = 1, 2, ..., k (i) ; j = 1, 2, ..., m where $a^{s}_{i,j}$ is the time spent by one unit of product type i on machine j when using manufacturing process s.

$a^s_{i,j} = 0$ if machine j does not belong to process s of product type i.

Thus, $a^s_{i,j}$ is the element of row $\sum_{v=1}^{i-1} k(v) + s$ and column j.

Matrix A is shown in figure 1.

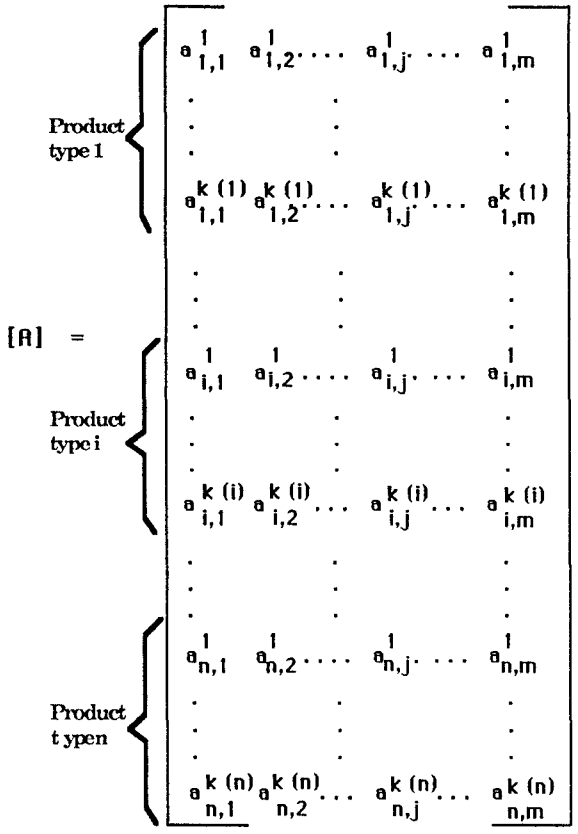


Fig. 1 : Matrix A

To each product type i, we associate a weight q_i which can be interpreted as the average number of products of type i to manufacture during a given period of time T.

The algorithm which computes the quantity of each product type to manufacture, using each of the available manufacturing processes, is called the translation algorithm, because it transforms the multi-manufacturing process problem into a mono-manufacturing process problem.

3. THE TRANSLATION ALGORITHM

3.1. Problem Definition

The goal of the translation algorithm is to assign a quantity to manufacture by each of the available production processes.

We define q^s_i (i = 1, 2, ..., n ; s = 1, 2, ..., k (i)) as the number of products of type i which have to be manufactured using manufacturing process s.

A set $\{q^s_i\}$; i = 1, 2, ..., n : s = 1, 2, ..., k (i) is said to be feasible if the following constraints are satisfied :

a. the required average number of products of each type (q_i) is manufactured within time T, that is :

$$\sum_{s=1}^{k(i)} q^s_i = q_i \qquad \text{for } i = 1, 2, ..., n \qquad (1)$$

b. the capacity constraints are verified :

$$\sum_{i=1}^{n} \sum_{s=1}^{k(i)} a^s_{ij}\, q^s_i \le T \qquad \text{for } j = 1, 2, ..., m \qquad (2)$$

Let us consider the $\sum_{i=1}^{n} k(i)$ points of $\mathcal{R}^m$ derived from matrix A. The coordinates of one of these points are the elements of a row of matrix A, and the weight associated to this point is the q^s_i-value associated to that row.

Let R be an integer. The feasible set $\{q^s_i\}$; i = 1, 2, ..., n : s = 1, 2, ..., k (i) is said to be optimal if and only if it is possible to group the previous points into r clusters ($r \le R$) such that the sum of the inertia of those clusters is minimal. In other words, to be impossible to find another partition of the set of points and another feasible set of weights having a smaller sum of inertia.

This approach aims at selecting manufacturing processes which can be grouped into subsets as compact as possible, in order to favour the second stage of the algorithm. Indeed, this compactness reduces the variety of the manufacturing processes to consider at the second stage of the algorithm.

3.2. Definition Of An Initial Feasible Solution

The problem consists in finding a set $\{q^s_i(0)\}$; i = 1, 2, ..., n : s = 1, 2, ..., k (i) verifying constraints (1) and (2).

Let :

$$[Q] = [q^1_1, q^2_1, ..., q^{k(1)}_1, q^1_2, q^2_2, ..., q^{k(2)}_2, ..., q^1_n, q^2_n, ..., q^{k(n)}_n]$$

$$[B] = \begin{bmatrix} 1 & 1 \ldots 1 & 0 & 0 \ldots & \ldots & \ldots \\ 0 & & & & & \\ 0 & 0 \ldots 0 & 1 & 1 \ldots 1 & 0 & 0 \ldots \\ \vdots & & & & & \end{bmatrix}$$

k (1) k (2) k (n)

[B] has n rows and $\sum_{i=1}^{n} k(i)$ columns.

Let us also define :

$$[U] = [q_1, q_2, ..., q_n]$$

Relation (1) can be rewritten as :

$$[B]\,[Q]^T = [U]^T, \qquad (1')$$

where T denotes the transposition

Relation (2) becomes :

$$[Q]\,[A] \leq T\,[1, 1, ..., 1] \qquad (2')$$

Furher, we define :

$[S] = [x_1, x_2, ..., x_n]$ as a matrix of slack variables

[O] a n x n zero-matrix

[I] a n x n unit matrix

The feasible solution (if any) is obtained by solving the following LP-problem :

$$\text{Minimize} \sum_{i=1}^{n} x_i \qquad (3)$$

s.t.

$$[B,O]\,[Q,S]^T = [U]^T \qquad (4)$$

$$[Q,S] \begin{bmatrix} A \\ I \end{bmatrix} = T\,[1, 1, ..., 1] \qquad (5)$$

$$q_i^s \geq 0 \; ; \; i = 1, 2, ..., n \; ; \; s = 1, 2, ..., k(i) \qquad (6)$$

$$x_i \geq 0 \; ; \; i = 1, 2, ..., n \qquad (7)$$

3.3. Minimization Of Total Intra-cluster Inertia

The process starts with an initial feasible solution $\{q_i^s\}$; i = 1, 2, ..., n ; s = 1, 2, ..., k (i). We first compute a set of clusters using the K-mean approach. We then improve the total intra-cluster inertia by modifying the q_i^s-values in an adequate manner with respect to constraints (1) and (2). We then return to the computation of clusters, and so on until it is impossible to further reduce the total intra-cluster inertia.

We detail the previous two points in the following subsections.

3.3.1. The K-mean approach

We start by setting r = R points of $\mathcal{R}^m$, say G (1), G (2), ..., G (r), at random. Let $(x_v^1, x_v^2, ..., x_v^m)$ be the coordinates of G (v), v = 1, 2, ..., r. We can, for instance, choose these points from the set $\{P_i^s\}$; s = 1, 2, ..., k (i) ; i = 1, 2, ..., n, where P_i^s is the point whose coordinates are $(a_{i1}^s, a_{i2}^s, ..., a_{im}^s)$.

P_i^s is associated with G (u) if :

$$d(G(u), P_i^s) = \min_{v=1,2,...,r} d(G(v), P_i^s) \qquad (8)$$

where d is the Euclidian distance.

Relation (8) can be rewritten as :

$$\sum_{j=1}^{m} (x_u^j - a_{ij}^s)^2 = \min_{v=1,2,...,r} \sum_{j=1}^{m} (x_v^j - a_{ij}^s)^2 \qquad (8')$$

If several G (v) lead to the minimum in (8), we choose the G (u) with the smallest value of v.

At the end of the process, a cluster is associated with each G (v), v = 1, 2, ..., r. Some of these clusters may be empty : in that case, we remove the corresponding points G (v), reorder the remainding points and reduce r by the number of empty clusters. We denote K (v) the cluster corresponding to G (v) for v = 1, 2, ..., r.

The process continues by computing the gravity centers H (v) of clusters K (v), v = 1, 2, ..., r. They are obtained by solving:

$$\sum_{P_i^s \in K(v)} q_i^s \overrightarrow{H(v), P_i^s} = 0 \qquad (9)$$

which can be rewritten as :

$$\sum_{P_i^s \in K(v)} q_i^s (a_{ij}^s - y_v^j) = 0, \quad j = 1, 2, ..., m$$

or

$$y_v^j = \Big(\sum_{P_i^s \in K(v)} q_i^s a_{ij}^s \Big) \Big/ \Big(\sum_{P_i^s \in K(v)} q_i^s \Big), \; j = 1, 2, ..., m \qquad (9')$$

where $(y_v^1, y_v^2, ..., y_v^m,)$ are the coordinates of H (v).

We then set

$x_v^j = y_v^j$ for j = 1, 2, ..., m

and restart at the beginning of the process until it converges. The convergence is proven (see for instance [5] and the related references).

3.3.2. Improvement of the total intra-cluster inertia

At this point of the process, we have obtained r ($r \leq R$) clusters K (1), K (2), ..., K (r) and the related gravity centers H (1), H (2), ..., H (r). The notations are the same as those of the previous sub-section.

We try to reduce the total intra-cluster inertia by modifying the q_i^s-values with respect to constraints (1) and (2). In the following, we restrict ourselves to the rows of [A].

a. We first freeze the rows of matrix [A] corresponding to product types having only one production process. The related quantities cannot be used to reduce the total intra-cluster inertia because the quantity assigned to the unique manufacturing process is the total quantity required and cannot be modified.

b. We compute the point P_{i1}^{s1} belonging to a cluster K (u) such that :

$$d(P_{i1}^{s1}, H(u)) = \max_{v=1,...,r} \max_{(i,s) \in E(v)} d(P_i^s, H(v)) \qquad (10)$$

where

$$E(v) = \{(i,s) / P_i^s \in K(v)\}$$

P_{i1}^{s1} is the point which leads to the maximal intra-cluster inertia variation when P_{i1}^{s1} varies by one unit.

We compute P_{i1}^{s2} belonging to a cluster K (w) such that :

$$d(P_{i1}^{s2}, H(w)) = \mathit{Min}_{v=1,2,...,r} \; \mathit{Min}_{s \in F(v)} d(P_{i1}^{s}, H(v)) \qquad (11)$$

where

$$F(v) = \{s / P_{i1}^{s} \in K(v)\}$$

P_{i1}^{s2} is the point corresponding to the product type i1 which leads to the minimal intra-cluster variation when the related quantity varies by one unit.

c. We now consider the following two cases :

c.1. $|d(P_{i1}^{s1}, H(u)) - d(P_{i1}^{s2}, H(w))| < \eta$

where h is a small real positive value given by the user.

In that case, it is impossible to significantly reduce the total inertia of the system by reducing q_{i1}^{s1} and increasing q_{i1}^{s2} by the same value (in order to verify constraint (1)).

The rows of matrix [A] related to q_{i1}^{s1} and q_{i1}^{s2} are frozen, and the algorithm continues as shown in paragraph d., below.

c.2. $|d(P_{i1}^{s1}, H(u)) - d(P_{i1}^{s2}, H(w))| > \eta$

In that case, we first compute

$$\Delta 1 = \mathit{Min}_{j \in X} \left[\left(T - \sum_{i=1}^{n} \sum_{s=1}^{k(i)} (a_{ij}^{s} \; q_i^{s}) \right) / (a_{i1\,j}^{s2} \; a_{i1\,j}^{s1}) \right]$$

where

$$X = \{j / (a_{i1\,j}^{s2} \; a_{i1\,j}^{s1}) > 0\}$$

$\Delta 1$ is the maximal value we can subtract from q_{i1}^{s1} and add to q_{i1}^{s2} with respect to constraints (2).

Thus, $\Delta = \mathit{Min} \{q_{i1}^{s1}, q_{i1} - q_{i1}^{s2}, \Delta 1\}$ is the maximal value we can subtract from q_{i1}^{s1} and add to q_{i1}^{s2} with respect to constraints (1) and (2).

If $\Delta > 0$, we set

$$q_{i1}^{s1} = q_{i1}^{s1} - \Delta$$

and

$$q_{i1}^{s2} = q_{i1}^{s2} + \Delta$$

Then, the rows of [A] related to q_{i1}^{s1} and q_{i1}^{s2} are frozen and the algorithm continues as shown hereafter.

d. If the number of non-frozen rows of [A] is equal to zero, it is no longer possible to further decrease the total inertia with regard to the previous gravity center, and the algorithm stops. Otherwise, we return to b., after having removed the points P_i^s corresponding to the rows of [A] previously frozen.

3.3.3. Summary

3.3.3.1. Synthesis The algorithms presented in the previous sub-sections can be summarized as follows.

1. Stage 1 : Computation of a feasible solution

At this stage the LP-problem, given by relations (3) to (7), is solved. If this problem does not have a solution, the initial problem has no solution either and the algorithm stops ; else, we obtain an initial feasible solution $\{q_i^s\}$; i = 1, 2, ..., n ; s = 1, 2, ..., k (i).

2. Stage 2 : The K-mean approach (see sub-section 3.3.1.)

2.1. We choose G (1), G (2), ..., G (r) at random

2.2. We compute clusters K (1), ..., K (r) corresponding to points G (1), G (2), ..., G (r)

2.3. We compute the gravity centers H (1), H (2), ..., H (r) of K (1), K (2), ..., K (r)

2.4. If H (s) = G (s) for s = 1, 2, ..., r, go to stage 3, else set G (s) = H (s) and return to 2.2.

3. Stage 3 : Improvement of the total intra-cluster inertia

In this second part of the algorithm, we reduce the total intra-cluster inertia as explained in the previous section. The gravity centers H (s), s = 1, 2, ..., r remain unchanged during this stage.

4. Stage 4 : Decision

If no improvement has been achieved during the previous stage, the algorithm stops and clusters K (1), K (2), ..., K (r) are kept as the solution of the problem ; else, we return to 2.3.

Note :

As the reader can see, the computation has been made assuming the q_i^s-values are real instead of integer. In practice, the horizon T is very large compared to manufacturing times and the number of parts involved in the computation is great enough to derive a near-optimal solution to the problem from the outputs of the translation algorithm, by adjusting these values to the nearest integer value.

3.3.3.2. Convergence The first stage of the algorithm consists of finding a feasible solution to a LP-problem. As it is well known, the solution of a LP-problem is obtained within a reasonable amount of time, despite the fact that it is theoretically NP-hard.

It is also well known that the K-mean approach converges and that the limit is reached after a very small number of computations, specially when the starting points needed in this type of approach are defined at random.

The computational burden of the third stage of the algorithm is upper bounded by :

$$\sum_{i \in S} C_{k(i)}^{2}$$

where S = {i/ i Œ (1, 2, ..., n) and k (i) ≥ 2}

because the points corresponding to a given product are, at most, examined two-by-two.

Furthermore, the step composed of stage 2 followed by stage 3 is executed only a finite number of times because :

(i) the initial intra-cluster inertia is finite

(ii) at each step, the intra-cluster inertia decreases at least by $\Delta\eta > 0$

Thus the algorithm stops after a finite number of iterations.

4. THE MONO-MANUFACTURING PROCESS ALGORITHM

The translation algorithm developed in the previous section assigns a portion of the total quantity of a product type required within the horizon T to each of the available manufacturing processes related to that product type . In other words, the quantity q_i (i = 1, 2, ..., n) is split into quantities q_i^1, q_i^2, ..., $q_i^{k(i)}$ which have to be produced following their respective manufacturing processes 1, 2, ..., k (i) of product type i. Note that some of the quantities q_i^j may be equal to zero. In that case, the corresponding manufacturing processes are removed and k (i) is reduced by the number of manufacturing processes removed.

Thus, the translation algorithm transforms the initial multi-manufacturing process problem of n product-types into a mono-manufacturing process problem of $N = \sum_{j=1}^{n} k(i)$ product types. The translation has been performed in order to favour the design of manufacturing cells along with their related part families using the mono-product algorithm already presented in (7). This algorithm is summarized in this section.

4.1. Notations And Problem Definition

In order to simplify the notations, let 1, 2, ..., N be the new mono-manufacturing process product types, b_{ij} (i = 1, ..., N ; j = 1, ..., m) the time spent by a product of type i on machine j, and q_i the total quantity of product type i which has to be manufactured within the horizon T. Note that the b_{ij} values are the a_{ij}^s values of the general problem and that the quantities q_i are the quantities q_i^s obtained as outputs of the translation algorithm.

We also assume that the b_{ij} values have been normalized (i.e. divided by the greatest b_{ij}-value) in order to handle only values belonging to [0,1]. In the following, these normalized values are still denoted by b_{ij}.

Let $X = (X_1, X_2, ..., X_K)$ be a partition of the set of N product types into K product families and $Y = (Y_1, Y_2, ..., Y_K)$ a partition of the set of m machines into K manufacturing cells. We also denote $B_k = X_k \times Y_k$, k = 1, 2, ..., K and $B = \bigcup_{k=1}^{K} B_k$.

The basic problem consists of finding K and a set $B_1, B_2, ..., B_K$ of blocs which minimize :

$$\Delta(X,Y) = h \sum_{(i,j) \in \supseteq B} q_i \, b_{ij}^{\lambda} + (1-h) \sum_{(i,j) \notin B} q_i \, (1 - b_{ij})^{1/\lambda} \qquad (12)$$

where h Œ [0,1] is a parameter giving more or less importance to the positive values contained in B and $\lambda \in (0, +\infty)$ is a parameter modifying the influence of high or low processing times.

4.2. The Basic Algorithm

The basic algorithm is a heuristic aiming at minimizing Δ (X, Y). It can be summarized as follows :

1. Step 1

Generate K and a partition $X^0 = (X_1^0, X_2^0, ..., X_K^0)$ of the set of product types at random.

2. Step 2

2.1. For j = 1, 2, ... m

2.1.1. For k = 1, 2, ... K

Compute C (X^0, k) =

$$h \sum_{i \in X_K^0} q_i \, b_{ij}^{\lambda} + (1-h) \sum_{i \notin X_K^0} q_i \, (1 - b_{ij})^{1/\lambda} \qquad (13)$$

2.1.2. End of loop k

2.1.3. Compute the smallest k_1 such that :

$$C(X^0, k_1) = \max_{k=1,2,...,K} C(X^0, k)$$

2.1.4. Assign machine j to manufacturing cell $Y_{k_1}^1$

2.2. End of loop j

2.3. If some Y_k^1 are empty for k Œ (1, 2, ..., K), they are removed and K is reduced by the number of subsets removed.

3. Step 3

3.1. For i = 1, 2, ... N

3.1.1. For k = 1, 2, ...K

Compute C (Y^0, k) =

$$h \sum_{j \in Y_{k_1}^1} q_i \, b_{ij}^{\lambda} + (1-h) \sum_{j \notin Y_{k_1}^1} q_i \, (1 - b_{ij})^{1/\lambda} \qquad (14)$$

3.1.2. End of loop k

3.1.3. Compute the smallest K_1 such that :

$$C(Y^1, k_1) = \max_{k=1,2,...,K} C(Y^1, k)$$

3.1.4. Assign product type i to part family $X_{k_1}^1$

3.2. End of loop i

3.3. If some X_k^1 are empty for k Œ (1, 2, ..., K), they are removed and K is reduced by the number of subsets removed.

4. Step 4 (Test)

4.1. If $X^1 \equiv X^0$, (X1, Y1) is the near optimal solution, and the algorithms stops.

4.2. If $X^1 \neq X^0$

4.2.1. Set $X^0 = X^1$

4.2.2. Go to 2

It has been proved in [4] that the previous algorithm converges. This algorithm has been improved in two different ways.

4.3. Improved Algorithm

The previous algorithm is known as the α-option algorithm. In [6], two new options have been introduced :

a. the β-option which allows a product family not to be assigned to a manufacturing cell.

b. the γ-option which allows a manufacturing cell not to correspond to a product family.

Figure 2 represents possible results when applying α-, β- or γ-options.

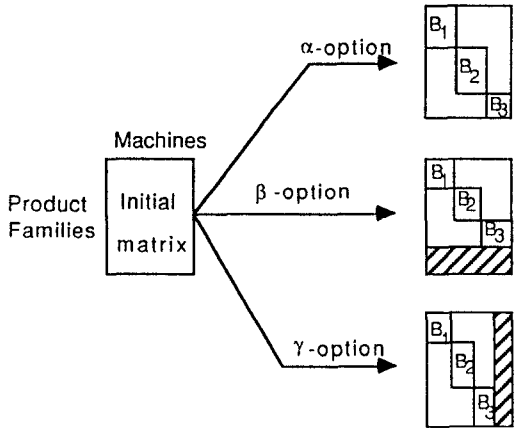


Fig. 2 : α-, β- and γ-options

The β-option is obtained by computing the value :

$$C(Y^1, k+1) = (1-h) \sum_{j=1}^{m} q_i \, (1 - b_{ij})^{1/\lambda}$$

in addition to the values defined by (14), and by computing k_1 as :

$$C(Y^1, k_1) = \max_{k=1,2,\ldots,K+1} C(Y^1, k)$$

If $k_1 \in \{1, 2, \ldots, K\}$, the product type i is assigned to part family $X^1_{k_1}$. If $k_1 = K + 1$, i is assigned to a supplementary subset X^1_{K+1}.

The γ-option is obtained by computing the value :

$$C(X^0, k+1) = (1-h) \sum_{i=1}^{N} q_i \, (1 - b_{ij})^{1/\lambda}$$

in addition to the values (13), and by computing k_1 as :

$$C(X^0, k_1) = \max_{k=1,2,\ldots,K+1} C(X^0, k)$$

If $k_1 \in \{1, 2, \ldots, K\}$, the machine j is assigned to the cell $Y^1_{k_1}$. If $k_1 = K + 1$, j is assigned to the supplementary subset Y^1_{K+1}.

It has been proved in Harhalakis (1989) that both options converge.

5. NUMERICAL EXAMPLE

In this section, we present an example of 7 machines M_1, M_2, ..., M_7 and 6 product types P_1, P_2, ..., P_6. In this example, M_2 and M_3 are identical, as well as M_6 and M_7.

The manufacturing processes are given below (machines are followed by manufacturing times) :

P_1 : (M_5, 1), (M_6 or M_7, 8), (M_1, 3), (M_2 or M_3, 2)
P_2 : (M_5, 2), (M_6 or M_7, 6)
P_3 : (M_4, 7), (M_5, 2), (M_6 or M_7, 3), (M_2 or M_3, 2)
P_4 : (M_1, 5), (M_4, 2), (M_2 or M_3, 1)
P_5 : (M_4, 5), (M_1, 2), (M_6 or M_7, 2)
P_6 : (M_4, 1), (M_5, 9), (M_1, 8), (M_6 or M_7, 2)

The following quantities of products have to be manufactured within a 500 time unit period.

P_1 (50), P_2 (20), P_3 (30), P_4 (20), P_5 (20), P_6 (20)

Consequently, the machine loads are :

M_1	: 450 time units
M_2 or M_3	: 180 time units
M_4	: 370 time units
M_5	: 330 time units
M_6 or M_7	: 690 time units

As seen above, 4 manufacturing processes are available for manufacturing product P_1, 2 for P_2, 4 for P_3, 2 for P_4, 2 for P_5, and 2 for P_6.

In this example, many feasible solutions are obvious. We chose the feasible solution proposed in figure 3. The numbers in parentheses represent the number of products manufactured using a particular process.

The translation algorithm statisfies the total quantity requirements :

P^1_1 (27), P^3_1 (23), P^1_2 (20), P^3_3 (30), P^1_4 (20), P^1_5 (20), P^1_6 (20)

Applying the algorithm described in section 4 for the mono-manufacturing process, we finally derive the manufacturing cells and part families as shown in figure 4.

		M_1	M_2	M_3	M_4	M_5	M_6	M_7
P_1	$P_1^1(25)$	3	2			1	8	
	$P_1^2(0)$	3		2		1	8	
	$P_1^3(25)$	3	2			1		8
	$P_1^4(0)$	3		2		1		8
P_2	$P_2^1(20)$					2	6	
	$P_2^2(0)$					2	6	
P_3	$P_3^1(10)$		2		7	2	3	
	$P_3^2(0)$			2	7	2	3	
	$P_3^3(20)$		2		7	2		3
	$P_3^4(0)$			2	7	2		3
P_4	$P_4^1(20)$	5	1		2			
	$P_4^2(0)$	5		1	2			
P_5	$P_5^1(20)$	2			5		2	
	$P_5^2(0)$	2			5			2
P_6	$P_6^1(20)$	8			1	9	2	
	$P_6^2(0)$	8			1	9		2

Fig. 3 : A feasible solution

	M_2	M_4	M_7	M_1	M_5	M_6	M_3
P_3^1 (23)	2	0	8	3	1	0	0
P_3^3 (30)	2	7	3	0	2	0	0
P_5^1 (20)	0	5	0	2	0	2	0
P_1^1 (27)	2	0	0	3	1	8	0
P_2^1 (20)	0	0	0	0	2	6	0
P_4^1 (20)	1	2	0	5	0	0	0
P_6^1 (20)	0	1	2	8	9	0	0

Fig. 4 Manufacturing cells and part families

The partition in figure 4 has been computed applying the g-option with parameter values h = 0.9 and l = 1.0. We obtained two product families :

$X_1^* = (P_3^1, P_3^3, P_5^1)$ $X_2^* = (P_1^1, P_2^1, P_4^1, P_6^1)$

and two manufactruing cells :

$Y_1^* = (M_2, M_4, M_7)$ $Y_2^* = (M_1, M_5, M_6)$

Note that machine M_3 belongs to the supplementary subset.

6. CONCLUSION

The fast translation algorithm presented in this section tends to split production volumes in a way that the clusters obtained have a total inter-cluster inertia as small as possible. This leads to using manufacturing processes which are close to each other (i.e. grouped in clusters in which the points representing the manufacturing processes are close to each other according to the Euclidian distance). This situation favours the use of mono-manufacturing process algorithm described in section 4.

Unfortunately, the translation algorithm does not necessarily favour the application of the algorithm which aims at reducing inter-cell traffic.

REFERENCES

[1] McCORMICK W.T., SCHWEITZER P.J. AND WHITE T.E., "Problem Decomposition and Data Re-Organization by a Clustering Technique", Operations Research, 20 (993), 1972.

[2] KING J.R., "Machine-Component Grouping Using ROC Algorithm", International Journal of Production Research, 26 (1), 1980.

[3] CHAN H.M. AND MILNER D.A., "Direct Clustering Algorithm for Group Formation in Cellular Manufacturing", Journal for Manufacturing Systems, 1, (1), 1982.

[4] GARCIA H. AND PROTH J.M., "Group Technology in Production Management : The Short Horizon Planning Level", Applied Stochastic Models and Data Analysis, 1, 1985.

[5] GARCIA H. AND PROTH J.M., "A New Cross-Decomposition Algorithm : The GPM Comparison with the Bond Energy Method", Control and Cybernetics, 15 (2), 1986.

[6] HARHALAKIS G., HILGER J. AND PROTH J.M., "Generalization and Implementation of the GP Method to Generate Manufacturing Cells and Part Families", submitted for publication to the International Journal of Production Research, 1989.

[7] McAULEY J., "Machine Grouping for Efficient Production", Production Engineering, 51 (53), 1972.

[8] LESKOWSKY Z., LOGAN L. AND VANNELLI A., "Group Technology Decision Aids in an Expert System for Plant Layout", Modern Production Management Systems, Elsevier Science Publisher, 1987.

[9] KUSIAK A., "EXGT-S : A Knowledge-Based System for Group Technology", International Journal of Production Research, 26 (1), 1988.

[10] KUSIAK A. AND WADOOD I., "Knowledge-Based System for Group Technology", 1st International Conference on CIM,RPI, Troy, NY, May 22-25, 1988.

[11] HARHALAKIS G., MINIS I., NAGI R. AND PROTH J.M., "A Comprehensive Group Technology System for Cellular Manufacture", 10-th ICPR, August 14-18, 1989, NOTTINGHAM (UK).

[12] HARHALAKIS G., NAGI R. AND PROTH J.M., "An Efficient Heuristic in Manufacturing Cell Formation for Group Technology Applications", accepted for publication in the International Journal of Production Research, 1988.

[13] HARHALAKIS G., HILGER J., NAGI R. AND PROTH J.M., "Formation of Manufacturing Cells : An Algorithm for Minimizing the Inter-Cell Traffic", submitted for publication to Applied Stochastic Models and Data Analysis, 1988.

[14] ARMOUR G.C. AND BUFFA E.S., "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities", Management Science, 9, 294, 1963.

[15] BUFFA E.S., ARMOUR G.C. AND VOLLMANN T.E., "Allocating Facilities with CRAFT", Harvard Business Research, 42, 136, 1964.

[16] FOULDS L.R., "Techniques for Facilities Layout : Deciding which Pairs of Activities should be Adjacent", Management Science, 29, 12, 1983.

[17] MOORE J.M., "Facilities Design with Graph Theory and Strings", Omega, 4, 193-203, 1976.