



HAL
open science

Regroupements en familles de produits en fonction des outils disponibles

Jean-Marie Proth

► **To cite this version:**

Jean-Marie Proth. Regroupements en familles de produits en fonction des outils disponibles. RR-0392, INRIA. 1985. inria-00076164

HAL Id: inria-00076164

<https://inria.hal.science/inria-00076164>

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France

Tél (3) 954 90 20

Rapports de Recherche

N° 392

REGROUPEMENTS EN FAMILLES DE PRODUITS EN FONCTION DES OUTILS DISPONIBLES

Jean-Marie PROTH

Mars 1985

REGROUPEMENTS EN FAMILLES DE PRODUITS
EN FONCTION DES OUTILS DISPONIBLES

Jean Marie PROTH

PROJET SAGEP

* Travail financé par l'agence de l'informatique.

ABSTRACT

This paper gives an algorithm which leads to a partition of a set of n objects in p classes ($p \ll n$). The objective is to minimize the sum of the inertias of the p classes. We show a way to obtain a good initial situation.

RESUME

Ce papier donne un algorithme qui conduit à une partition d'un ensemble de n objets en p classes ($p \ll n$). L'objectif est de minimiser la somme des inertias des p classes. Nous montrons une méthode pour obtenir une bonne situation initiale.

I. INTRODUCTION

Les lignes qui suivent ont pour but de proposer un algorithme de classement de n objets en p classes ($p \leq n$).

Après avoir posé le problème nous montrons que, partant de p points quelconques, il est possible de construire une suite de partitions de l'ensemble des objets en p classes et que cette suite converge. Simultanément, nous montrons que la somme des moments d'inertie des classes constituant la partition diminue lorsque le rang de l'itéré augmente.

Nous utiliserons ensuite ce résultat pour bâtir un algorithme de classification automatique qui débute par un processus de génération automatique des p points initiaux.

Nous proposons ensuite les programmes (écrits en FORTRAN) et un exemple d'application.

II. ENONCE DU PROBLEME

Nous considérons n éléments de \mathbb{R}^m et une distance sur \mathbb{R}^m . Nous notons x_i , $i=1, \dots, n$, ces éléments et d la distance. u_i désignera le poids associé à x_i .

Nous souhaitons trouver une partition de $E = \{x_1, x_2, \dots, x_n\}$ en p sous-ensembles, chacun de ces sous-ensembles regroupant des points proches au sens de la distance d . Nous avons choisi d'essayer de minimiser la somme des inerties de ces sous-ensembles, c'est à dire :

$$\sum_{k=1}^p \sum_{i/x_i \in e^k} u_i d^2(x_i, y^k) \quad (1)$$

où :

y^k est le centre d'inertie de e^k , $k=1, 2, \dots, p$
 et $\{e^1, e^2, \dots, e^p\}$ est la partition cherchée.

Nous verrons dans la suite qu'il est possible de trouver un minimum local de (1) et que le choix de la partition initiale laisse espérer que ce minimum local est voisin du minimum global.

III. RESULTAT PRINCIPAL

Nous proposons d'abord un algorithme qui, partant d'une situation initiale quelconque, aboutit à un minimum local. Cet algorithme est donné dans le théorème suivant.

THEOREME

Pour $i=1, 2, \dots, n$, soit x_i un point de \mathbb{R}^m et u_i le poids associé à ce point. Soit encore d une distance sur \mathbb{R}^m .

Nous désignerons par E l'ensemble $\{x_1, x_2, \dots, x_n\}$.

Soient $x_1^0, x_2^0, \dots, x_p^0$ ($p \leq n$) des points de \mathbb{R}^m appartenant ou non à E.

On considère l'algorithme suivant, que nous désignons par A1 :

1. Faire $k=0$.
2. Construire une partition $e_1^k, e_2^k, \dots, e_p^k$ de E de la manière suivante :
 - pour $i=1, 2, \dots, n$:
 - a. Rechercher l'ensemble $J_i \subset \{1, 2, \dots, p\}$ défini comme suit :

$$J_i = \{j \in \{1, 2, \dots, p\} / d(x_i, x_j^k) = \min_{s=1, \dots, p} d(x_i, x_s^k)\}$$
 - b. Si $\text{card}(J_i) = 1$, x_i est affecté à e_j^k , sinon x_i est affecté à $e_{j_1}^k$ où :

$$j_1 = \min_{j \in J_i} j$$
3. Pour $j=1, 2, \dots, p$, on cherche x_j^{k+1} , centre d'inertie de e_j^k . On rappelle que ce point vérifie :

$$\sum_{i/x_i \in e_j^k} u_i d^2(x_j^{k+1}, x_i) = \min_{y \in \mathbb{R}^m} \left[\sum_{i/x_i \in e_j^k} u_i d^2(y, x_i) \right] \quad (3)$$
4. Test.
 - 4.1. Si $x_j^{k+1} = x_j^k$ pour $j=1, 2, \dots, p$, fin de processus.
 - 4.2. Sinon :
 - 4.2.1. Faire $k=k+1$.
 - 4.2.2. Aller en 2.

L'algorithme A1 converge.

Démonstration.

a. Nous notons :

$I(y, E) = \sum_{x \in E} u_x d^2(y, x)$, où u_x est le poids associé à x et E , un ensemble fini.

Si $x_j^{k+1} \neq x_j^k$ pour au moins un $j \in \{1, 2, \dots, p\}$, alors

$$\sum_{j=1}^p I(x_j^{k+1}, e_j^k) < \sum_{j=1}^p I(x_j^k, e_j^k) \quad (\text{voir (3)}) \quad (4)$$

et :

$$\sum_{j=1}^p I(x_j^k, e_j^k) \leq \sum_{j=1}^p I(x_j^k, e_j^{k-1}) \quad (5)$$

(voir (2) et la définition du moment d'inertie)

Finalement :

$$\sum_{j=1}^p I(x_j^{k+1}, e_j^k) < \sum_{j=1}^p I(x_j^k, e_j^{k-1}) \quad (6)$$

Cette inégalité montre que si, pour deux pas d'itération successifs, l'ensemble des centres d'inertie est modifié, alors la somme des inerties des éléments de la partition décroît, donc la partition est modifiée et la nouvelle partition est différente de toutes les partitions précédemment rencontrées.

b. E étant fini, l'ensemble de ses partitions est fini. Associée à la conclusion de a., cette remarque achève la démonstration. \square

Remarques :

1. Le théorème que nous venons de présenter indique la convergence de l'algorithme proposé, mais n'affirme pas que la partition obtenue est celle qui assure le minimum absolu de la somme des inerties. Le contre exemple suivant nous en convaincra. Les points x_1 , x_2 et x_3 sont de poids 1. Les distances sont indiquées sur la figure 1. Nous considérons la partition constituée des sous-ensembles $\{x_1, x_3\}$ et $\{x_2\}$ ($p=2$).

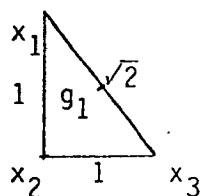


Fig. 1

Les centres d'inertie de ces sous-ensembles sont respectivement g_1 et x_2 et la somme des inerties est $1/2$. Il est facile de vérifier que cette partition ne sera pas modifiée par l'algorithme et, en même temps, que la partition $\{x_1\}, \{x_2, x_3\}$ conduirait à une somme d'inerties égale à $1/4$.

2. L'approche précédente est du type "nuées dynamiques" (voir en particulier [2]).
3. L'algorithme ci-dessus suppose fixé le nombre de classes. C'est une hypothèse forte.
4. Les résultats obtenus sont très dépendants des points initiaux choisis dans \mathbb{R}^m et de la distance.

Nous proposons dans le paragraphe suivant un algorithme que conduit à une situation initiale raisonnable.

IV. RECHERCHE D'UNE SITUATION RAISONNABLE

Choisissons un réel positif r et appelons densité d'un point x_i ($i \in \{1, 2, \dots, n\}$) de \mathbb{R}^m le nombre de points dont la distance à x_i est inférieure ou égale à r .

Nous pensons qu'un choix raisonnable des p points initiaux doit porter sur des points :

- 1) de forte densité, ce qui assure qu'un grand nombre de points sont proches des points choisis.
- 2) convenablement répartis dans l'ensemble des points à classer.

A noter que ces deux critères peuvent être incompatibles.

Nous proposons l'algorithme suivant, noté A2, pour le choix des points initiaux :

1. Choix de r .
2. Choix de $\delta \geq 2r$.
3. Choix (éventuellement au hasard) de $x_1^0 = \{x_1, x_2, \dots, x_n\} \in E$.
4. Simultanément, par balayage des points de E :
 - 4.1. Recherche de la densité de x_1^0 .
 - 4.2. Recherche de E_1 , ensemble des points de E distants de x_1^0 de plus de δ .
5. Itérations :
 - 5.1. Faire $k=1$.
 - 5.2. Choix (au hasard) de $x_{k+1}^0 \in E_k$.
 - 5.3. Simultanément, par balayage des points de E_k :
 - 5.3.1. Recherche de la densité de x_{k+1}^0 .
 - 5.3.2. Recherche de E_{k+1} , ensemble des points de E_k distants de x_{k+1}^0 de plus de δ .
 - 5.4. Test.
 - 5.4.1. Si $E_{k+1} = \emptyset$.
 - 5.4.1.1. Si $k+1 \geq p$, aller en 6.
 - 5.4.1.2. Sinon :
 - 5.4.1.2.1. Edition d'un message : "Nombre de points initiaux insuffisants".
 - 5.4.1.2.2. Décision de l'utilisateur :
 - . soit retour en 1.
 - . soit retour en 2.
 - . soit fin de processus.
 - 5.4.2. Si $E_{k+1} \neq \emptyset$.
 - 5.4.2.1. Faire $k=k+1$.
 - 5.4.2.2. Retour en 5.2.
6. Classement des x_i^0 , $i=1,2,\dots,k+1$, par ordre décroissant de leur densité.
7. Les p premiers points obtenus sont les points initiaux.

Remarques :

1. A chaque pas d'itération, nous choisisons au hasard un point suffisamment éloigné de tous les points précédemment retenus et nous calculons sa densité. Ce faisant, nous limitons le nombre de densités à calculer.
2. L'importance de la densité apparaît au cours de la seconde phase du processus (classement), qui conduit à ne retenir que les points de plus forte densité parmi les points éloignés les uns des autres de plus de δ .
3. Le nombre de points retenus doit être supérieur ou égal au nombre de points initiaux. Si ce nombre est proche du nombre de points initiaux, on privilégie la répartition des points initiaux dans l'ensemble des points à classer au détriment des densités. S'il est très grand, on privilégie les densités.

V. LE LOGICIEL DE CLASSIFICATION

Le logiciel de classification que nous présentons maintenant est structuré de manière à pouvoir être complété pour autoriser le choix de plusieurs distances et de plusieurs processus de classification à partir de ces distances.

Les données à traiter sont dans le fichier 70 qui est structuré comme suit :

Premier enrégistrement :

Nombre d'observations	I4
Nombre de paramètres	I4

Enrégistrement suivants :

Chaque enrégistrement contient une suite de 50 observations (sauf peut être le dernier).

Chaque observation est composée :

-de la suite des paramètres	F2.0
-du poids de l'observation	F2.0

Le programme est écrit pour recevoir sans modification des observations composées de 19 paramètres au maximum, plus le poids, l'ensemble étant exprimé en format F2.0.

Une modification du format entraînerait une modification du recl (record length) du fichier 70.

Une modification du nombre maximal de paramètres modifierait en conséquence le "recl" du fichier 70 ainsi que :

les dimensions des tableaux actuellement égales à 20 dans le programme principal et ses sous-programmes dist et rep (cette dimension doit toujours être supérieure de 1 au nombre de paramètres).

Une modification du nombre maximal d'observations modifierait en conséquence le "recl" du fichier 70 ainsi que :

1. la dimension du tableau iz dans le programme principal et ses sous-programmes.
2. plus indirectement, la dimension du tableau it qui doit être supérieure ou égale au nombre de points candidats au rôle de points initiaux pour la constitution des classes.

La dimension 50 correspond au nombre d'observations par enrégistrement logique.

Nous donnons dans les pages suivantes le programme principal et les sous-programmes.

Les résultats des calculs figurent dans le fichier 50.

A l'entrée du programme, le nombre d'observations et de paramètres sont affichés.

L'utilisateur fournit :

1. le nombre de classes souhaitées.
2. la distance utilisée. Pour l'instant, la distance euclidienne et la distance égale au nombre de non correspondances (cas d'observations linéaires) sont seules disponibles.

3. les représentants des classes. Actuellement, le choix se réduit aux centres d'inertie.
4. le rayon servant au calcul des densités. C'est le rayon des boules ayant comme centre les points-observation et qui servent à la détermination de la densité des points-observation : la densité d'un point-observation est égale au nombre de points contenus dans la boule dont il est le centre.
5. la distance minimale entre deux points initiaux. Cette distance est déterminante. Trop importante, elle interdit la détermination d'un nombre suffisant de points initiaux. Trop faible, elle peut conduire à choisir des points initiaux trop proches.

Le programme fournit, dans le fichier 50 et pour chaque classe :

1. le représentant de la classe.
2. la liste des éléments de la classe.

Nous donnons le programme principal, les deux sous-programmes auxquels il fait appel :

1. le sous-programme "dist" qui calcule la distance entre deux observations.
2. le sous-programme "rep" qui calcule les représentants des classes.

PROGRAMME DE CHARGEMENT

```

1      dimension a(50,20)
2      open(70,access="direct",form="formatted",recl=10000)
3      open(55,form="formatted")
4      write(0,35)
5 35   format(1x,"Nouveau fichier=1,modif.=2")
6      write(0,36)
7 36   format(1x"Impression=3,adjonction=4")
8      read(0,2)ind
9      go to (500,600,700,800),ind
10 500 write(0,1)
11     1 format(2x,"_Nombre d'individus ? ")
12     read(0,2)n
13     2 format(v)
14     write(0,3)
15     3 format(2x,"_Nombre de parametres ? ")
16     read(0,2)m
17     m=m+1
18     write(70,4,rec=1,err=5)n,m
19     4 format(2i4)
20     kk=0
21 c
22 c
23     kind=1
24     do 10 i=1,n
25     kk=kk+1
26     write(0,11)i
27 11 format(/,15x,24("="),/,15x,"I individu numero",i4," I",/,15x,2
\c4("="),/)
28     write(0,30)
29 30 format(2x,"_Le poids se tape en derniere position ! ")
30     read(0,2)(a(kk,j),j=1,m)
31     if(kk.lt.50)go to 10
32     kind=kind+1
33     write(70,12,rec=kind,err=6)((a(k,j),j=1,m),k=1,kk)
34 12 format(1000f10.3)
35     kk=0
36 10 continue
37 c
38 c
39     if(kk.eq.0)go to 9
40     kind=kind+1
41     write(70,12,rec=kind,err=6)((a(k,j),j=1,m),k=1,kk)
42     goto 9
43 600 write(0,601)
44 601 format(2x,"Individu a modifier ou 0")
45     read(0,2)num
46     if(num.eq.0) go to 9
47     k1=(num-0.5)/50+2
48     k2=num-(k1-2)*50
49     k3=n-(k1-2)*50
50     if(k3.gt.50)k3=50
51     read(70,12,rec=k1,err=6)((a(k,j),j=1,m),k=1,k3)
52     num=num+1
53     write(0,602)(a(k2,j),j=1,m)
54 602 format(2x,5f10.3)

```

```

55      read(0,2)(a(k2,j),j=1,m)
56      write(70,12,rec=k1,err=6)((a(k,j),j=1,m),k=1,k3)
57      go to 600
58 700   read(70,4,rec=1,err=5)n,m

59      m1=m-1
60      write(55,60)n,m1
61 60    format(2x,"Nombre d'individus",i4,2x,"Nombre de parametres",i4)
62      write(55,62)
63 62    format(2x,"Le poids est en derniere position")
64      k=2
65      iu=0
66 702   read(70,12,rec=k,err=9)((a(kk,j),j=1,m),kk=1,50)
67      kk=(k-2)*50
68      jj=n-kk
69      if(jj.le.0)go to 9
70      if(jj.gt.50)jj=50
71      do 701 i=1,jj
72      iu=iu+1
73 701   write(55,61)iu,(a(i,j),j=1,m)
74      k=k+1
75      go to 702
76 61    format(1x,i4/5(1x,5f10.3))
77 800   read(70,4,rec=1,err=5)n,m
78      write(0,803)
79 803   format(2x,"Adjonction=1,stop=0")
80      read(0,2)ia
81      if(ia.eq.0)go to 9
82      k1=(n-0.5)/50+2
83      k2=n-(k1-2)*50
84      read(70,12,rec=k1,err=6)((a(k,j),j=1,m),k=1,k2)
85      n=n+1
86      if(k2.eq.50)go to 805
87      k2=k2+1
88 808   write(0,875)
89 875   format(2x,"Tapez l'observation suivante ")
90      read(0,2)(a(k2,j),j=1,m)
91      write(70,12,rec=k1,err=6)((a(k,j),j=1,m),k=1,k2)
92      go to 806
93 805   k2=1
94      k1=k1+1
95      go to 808
96 806   write(70,4,rec=1,err=5)n,m
97      go to 800
98      5 write(0,7)
99      7 format("erreur d'ecriture des parametres n et m dans le fichier
\c70")
100     goto 9
101     13 write(0,14)
102     14 format("erreur d'ecriture ds dernier enr a(j) ds 70")
103     goto 9
104     6 write(0,8)
105     8 format("erreur d'ecriture des a(j) dans le fichier 70")
106     9 close(70)
107     close(55)
108     stop
109     end

```

PROGRAMME PRINCIPAL

```

1 c
2 c
3 c      On part du fichier 35
4 c      Le fichier 70 est construit a partir du fichier 35 apres
5 c      une eventuelle normalisation des donnees
6 c      le fichier 50 -> resultats
7 c
8 c  CONTRAINTES ACTUELLES DU PROGRAMME
9 c----Nombre maxi. d'observations : 1999
10 c----Nombre maxi. de classes      : 50
11 c----Nombre maxi. de parametres   : 19
12      dimension a(50,20),aini(50,20),iz(2000),ds(50),it(500),b1(20),b
\c2(20)
13      dimension sg(20),sm(20)
14      dimension ires(300)
15      open(70,access="direct",form="formatted",recl=10000)
16      open(50,form="formatted")
17      open(35,access="direct",form="formatted",recl=10000)
18 c
19 6666 format(1000f10.3)
20      read(35,1,rec=1,err=27)n,m
21      write(70,1,rec=1,err=27)n,m
22      eps=.001
23      write(0,70)n,m
24      70 format("_Nombre d'observations",i4," _Nombre de parametres",i4
\c)
25 c
26      write(0,47)
27 47 format(2x,"Normalisation:oui=1,non=0")
28      read(0,3)nk
29      if(nk.eq.1) go to 48
30      k=1
31 66 k1=(k-.5)/50+2
32      k2=k-(k1-2)*50
33      k3=n-(k1-2)*50
34      if(k3.gt.50)k3=50
35      read(35,6666,rec=k1,err=21)((a(i5,j),j=1,m),i5=1,k3)
36      write(70,6666,rec=k1,err=21)((a(i5,j),j=1,m),i5=1,k3)
37      k=k+50
38      go to 66
39 48 do 49 j=1,m-1
40      sg(j)=0
41 49 sm(j)=0
42      do 50 k=1,n,50
43      k1=(k-.5)/50+2
44      k2=k-(k1-2)*50
45      k3=n-(k1-2)*50
46      if(k3.gt.50) k3=50
47      read(35,6666,rec=k1,err=13)((a(i5,j),j=1,m),i5=1,k3)
48      do 51 i5=1,k3
49      do 51 j=1,m-1
50      sg(j)=sg(j)+a(i5,j)**2./n
51 51 sm(j)=sm(j)+a(i5,j)/n
52 50 continue

```



```

53      do 52 j=1,m-1
54      write(0,*)sm(j),sg(j)
55 52    sg(j)=(sg(j)-sm(j)**2.)**.5
56      do 55 k=1,n,50
57      k1=(k-.5)/50+2
58      k2=k-(k1-2)*50
59      k3=n-(k1-2)*50
60      if(k3.gt.50) k3=50
61      read(35,6666,rec=k1,err=13)((a(i5,j),j=1,m),i5=1,k3)
62      do 56 i5=1,k3
63      do 56 j=1,m-1
64      a(i5,j)=(a(i5,j)-sm(j))/sg(j)
65      if((j.eq.1).or.(j.eq.5).or.(j.eq.9).or.(j.eq.13)) a(i5,j)=a
\c(i5,j)*3
66 56      continue
67      write(70,6666,rec=k1,err=13)((a(i5,j),j=1,m),i5=1,k3)
68 55      continue
69      1 format(2i4)
70      21 write(0,5)
71      5 format(2x,"_Nombre de classes ? ")
72      read(0,3)ip
73      3 format(v)
74      write(0,200)
75 200 format(12x,"<< CHOIX DES DISTANCES POUR LA CONSTITUTION DES CLA
\cSSES
76      &>>",//)
77      write(0,201)
78 201 format(2x,"_Distance euclidienne ;.....taper 1",//)
79      read(0,3)idis
80      write(0,400)
81 400 format(12x,"<< CHOIX DES REPRESENTANTS >>")
82      write(0,401)
83 401 format(2x,"centre d'inertie; tapez 1")
84      read(0,3)irep
85 c
86 c
87 c      *****
88 c      *          PHASE D'INITIALISATION          *
89 c      *****
90 c
91 c      Recherche du premier point.
92 c
93      write(0,4444)
94 4444 format(2x,"Voulez-vous donner vous-memes les points initiaux?")
95      write(0,4445)
96 4445 format(2x,"Oui=1,non=0")
97      read(0,3)nk
98      if(nk.eq.0) go to 4499
99      do 4446 k=1,ip
100     write(0,3347)k
101 3347 format(2x,"Point initial num.",i4,"?")
102     read(0,3)nk
103     k1=(nk-.5)/50+2
104     k2=nk-(k1-2)*50
105     k3=n-(k1-2)*50
106     if(k3.gt.50) k3=50

```

```

107      read(70,6666,rec=k1,err=30)((a(i5,j),j=1,m),i5=1,k3)
108      do 4449 j=1,m
109 4449  aini(k,j)=a(k2,j)
110 4446  continue
111      go to 333
112 4499  write(0,2)
113      2 format(2x,"_Choix du rayon pour le calcul des densites. ")
114      read(0,3)r
115      8 write(0,4)
116      4 format(2x,"_Choix de la distance minimale entre points initiaux
\c. ")
117      read(0,3)d1
118      if(d1.lt.r) goto 8
119      call random_$uniform(x)
120      if(x.gt.(1.-1.e-4))x=1.-1.e-4
121      k=x*n+1
122      k1=(k-.5)/50+2
123      k2=k-(k1-2)*50
124      k3=n-(k1-2)*50
125      if(k3.gt.50) k3=50
126      read(70,6666,rec=k1,err=13)((a(k,j),j=1,m),k=1,k3)
127      6 format(1000f2.0)
128      jj=1.
129      do 7 j=1,m
130      b1(j)=a(k2,j)
131      7 aini(1,j)=a(k2,j)
132 c
133 c      Recherche de la densite du premier
134 c      point et des points suivants.
135 c
136      do 12 j=1,50
137 12 ds(j)=0
138      ii=0
139      do 9 k=1,n
140      k1=(k-.5)/50+2
141      k2=k-(k1-2)*50
142      if(k2.gt.1)go to 10
143      k3=n-(k1-2)*50
144      if(k3.gt.50) k3=50
145      read(70,6666,rec=k1,err=26)((a(i,j),j=1,m),i=1,k3)
146 10 do 11 j=1,m
147      b2(j)=a(k2,j)
148 11 continue
149      call dist(b1,b2,m,idis,y)
150      if(y.le.r) ds(jj)=ds(jj)+1
151      if(y.lt.dl) goto 9
152      ii=ii+1
153      it(ii)=k
154      9 continue
155 c
156 c

```

```

157 c *****
158 c * ITERATIONS *
159 c *****
160 c
161 c Choix du point suivant
162 c
163 18 call random_$uniform(x)
164 if(x.gt.(1.-1.e-4))x=1.-1.e-4
165 ii1=x*ii+1
166 k=it(ii1)
167 k1=(k-.5)/50+2
168 k2=k-(k1-2)*50
169 k3=n-(k1-2)*50
170 if(k3.gt.50) k3=50
171 read(70,6666,rec=k1,err=28)((a(k,j),j=1,m),k=1,k3)
172 jj=jj+1
173 do 14 j=1,m
174 b1(j)=a(k2,j)
175 aini(jj,j)=a(k2,j)
176 14 continue
177 c
178 c
179 c Recherche de la densite du point
180 c et des points a retenir.
181 c
182 ii1=0
183 do 15 k5=1,ii
184 k=it(k5)
185 k1=(k-.5)/50+2
186 k2=k-(k1-2)*50
187 k3=n-(k1-2)*50
188 if(k3.gt.50) k3=50
189 read(70,6666,rec=k1,err=29)((a(i,j),j=1,m),i=1,k3)
190 do 17 j=1,m
191 17 b2(j)=a(k2,j)
192 call dist(b1,b2,m,idis,y)
193 if(y.le.r) ds(jj)=ds(jj)+1
194 if(y.lt.d1) goto 15
195 ii1=ii1+1
196 it(ii1)=k
197 15 continue
198 ii=ii1
199 if(ii.gt.0) goto 18
200 if(jj.ge.ip) goto 22
201 write(0,19)
202 19 format(2x,"_Nombre de points initiaux insuffisant.")
203 write(0,20)
204 20 format(2x,"_Abandon = 0",/,2x,"_Reprise = 1",/)
205 read(0,3)is
206 if(is.eq.0) goto 1664
207 goto 21
208 c
209 c

```

```

210 c *****
211 c * CLASSEMENT DES POINTS INITIAUX *
212 c * POSSIBLES *
213 c *****
214 c
215 c
216 22 do 23 i=1,jj-1
217 do 23 j=i+1,jj
218 if(ds(i).gt.ds(j))goto 23
219 do 24 k=1,m
220 x=aini(i,k)
221 aini(i,k)=aini(j,k)
222 24 aini(j,k)=x
223 x=ds(i)
224 ds(i)=ds(j)
225 ds(j)=x
226 23 continue
227 c
228 c
229 c *****
230 c * FIN DE L'INITIALISATION *
231 c *****
232 c
233 c Edition des points initiaux.
234 c
235 index=0
236 c
237 c
238 c *****
239 c * CONSTITUTION DES CLASSES *
240 c *****
241 c
242 c
243 kuw=0
244 333 do 40 i=1,n
245 40 iz(i)=0
246 do 41 k=1,n
247 k1=(k-.5)/50+2
248 k2=k-(k1-2)*50
249 if(k2.gt.1) goto 42
250 k3=n-(k1-2)*50
251 if(k3.gt.50) k3=50
252 read(70,6666,rec=k1,err=39)((a(is,j),j=1,m),is=1,k3)
253 42 do 43 j=1,m
254 43 b1(j)=a(k2,j)
255 do 44 i=1,ip
256 do 45 j=1,m
257 45 b2(j)=aini(i,j)
258 call dist(b1,b2,m,idis,y)
259 if(i.gt.1) goto 46
260 z=y
261 iz(k)=i
262 goto44
263 46 if(y.gt.z) goto 44
264 z=y
265 iz(k)=i
266 44 continue
267 41 continue

```

```

268 c
269 c
270 c
271 c *****
272 c * RECHERCHE DES REPRESENTANTS *
273 c *****
274 c Recherche des representants suivants.
275 c
276 call rep(ip,iz,aini,irep,n,m,idis,itel,eps)
277 index=index+1
278 kuw=kuw+1
279 if(index.lt.5) goto 600
280 write(0,1025)kuw
281 1025 format(2x,i4,"essais Voulez-vous continuer? oui=1;non=0")
282 read(0,3)nk
283 if(nk.eq.0)go to 700
284 index=0
285 go to 600
286 600 if(itel.eq.1) goto 333
287 c
288 c
289 c *****
290 c * IMPRESSION DES RESULTATS *
291 c *****
292 c
293 700 do 701 i=1,ip
294 write(50,801)i
295 801 format(15x," Classe numero ",i4)
296 write(50,899)
297 899 format(5x,"Representant")
298 do 57 j=1,m-1
299 if((j.eq.1).or.(j.eq.5).or.(j.eq.9).or.(j.eq.13))aini(i,j)=aini
\c(i,j)/3
300 57 aini(i,j)=aini(i,j)*sg(j)+sm(j)
301 write(50,898)(aini(i,j),j=1,m-1)
302 898 format(5(1x,e14.7))
303 ipp=0
304 do 702 k=1,n
305 k1=(k-.5)/50+2
306 k2=k-(k1-2)*50
307 if(k2.gt.1) goto 704
308 k3=n-(k1-2)*50
309 if(k3.gt.50) k3=50
310 read(70,6666,rec=k1,err=30)((a(i5,j),j=1,m),i5=1,k3)
311 704 if(iz(k).ne.i) goto 702
312 ipp=ipp+1
313 ires(ipp)=k
314 702 continue
315 write(50,3100)(ires(i9),i9=1,ipp)
316 3100 format(18(1x,i3))
317 701 continue
318 go to 1664
319 30 write(0,32)
320 32 format("erreur label 30")
321 goto 1664
322 29 write(0,33)
323 33 format("erreur label 29")
324 goto 1664

```

```
325     28 write(0,34)
326     34 format("erreur label 28")
327         goto 1664
328     26 write(0,35)
329     35 format("erreur label 26")
330         goto 1664
331     27 write(0,36)
332     36 format("erreur label 27")
333         goto 1664
334     13 write(0,37)
335     37 format("erreur label 13")
336         go to 1664
337     39 write(0,38)
338     38 format("erreur label 39")
339 1664 close(70)
340         close(50)
341     close(35)
342     stop
343     end
```

SOUS-PROGRAMME "dist"

```
283 c
284 c
285 c
286     subroutine dist(b1,b2,m,idis,y)
287     dimension b1(20),b2(20)
288     y=0
289     goto (1,2),idis
290     1 do 10 j=1,m-1
291     10 y=y+(b1(j)-b2(j))**2
292     y=y**0.5
293     goto 100
294     2 do 20j=1,m-1
295     i5=1
296     i5=b1(j)+b2(j)
297     if(i5.eq.2) i5=0
298     20 y=y+i5
299     100 return
300     end
```

SOUS-PROGRAMME "rep"

```

418 c
419 c
420 c
421     subroutine rep(ip,iz,aini,irep,n,m,idis,itel,eps)
422     dimension a(50,20),aini(50,20),iz(2000),b1(20),b2(20),co(50,20,3)
423     goto (1,100),irep
424     1 do 2 i=1,50
425       do 2,j=1,m-1
426         do 2 k=1,3
427           2 co(i,j,k)=0
428           do 12 i=1,50
429             do 12 j=1,20
430 12      aini(i,j)=0.
431           do 3 i=1,n
432             k1=(i-.5)/50+2
433             k2=i-(k1-2)*50
434             if(k2.gt.1) goto 4
435             k3=n-(k1-2)*50
436             if(k3.gt.50) k3=50
437 c
438             read(70,5,rec=k1,err=9)((a(k,j),j=1,m),k=1,k3)
439             5 format(1000f2.0)
440             4 i5=iz(i)
441             do 6 j=1,m-1
442               co(i5,j,1)=co(i5,j,1)+a(k2,m)
443               co(i5,j,2)=co(i5,j,2)-2*a(k2,m)*a(k2,j)
444             6 co(i5,j,3)=co(i5,j,3)+a(k2,m)*a(k2,j)**2
445             3 continue
446             itel=0
447             do 7 i=1,ip
448               if(co(i,1,1).gt.eps)go to 17
449               write(0,18)i
450 18      format(2x,"Representant num.",i4,"non utilise")
451             go to 7
452 17      do 7 j=1,m-1
453               a1=co(i,j,1)
454               a2=co(i,j,2)
455               a3=co(i,j,3)
456               pp=-a2/2./a1
457               uu=abs(pp-aini(i,j))
458               if(uu.gt.eps) itel=1
459               aini(i,j)=pp
460             7 continue
461             goto 1000
462 1000 write(0,15)
463 15 format(2x,"_Coefficient de dissimilarite non prevu !")
464             go to 1000
465             9 write(*,8)
466             8 format(2x,"_Erreur lecture fichier 70 dans sous-programme rep.")
467 1000 return
468     end

```


VI. UN EXEMPLE D'APPLICATION

Nous considérons un atelier composé de m machines et susceptible de traiter n produits différents. Nous sommes dans le cas d'un job-shop, c'est à dire dans le cas où chaque type de produit passe sur une partie seulement des machines de l'atelier, et dans des ordres variables suivant le type de produit.

Nous cherchons à regrouper les produits qui utilisent des moyens "proches". Un tel regroupement sera appelé "famille".

Les données du problème sont donc regroupées dans un tableau à n lignes et m colonnes :

$$A = (a_{ij}), \quad i=1,2,\dots,n \text{ et } j=1,2,\dots,m$$

avec :

$$a_{ij} = \begin{cases} 1 & \text{si le produit } i \text{ passe sur la machine } j. \\ 0 & \text{sinon.} \end{cases}$$

La figure 2 donne le tableau de l'exemple que nous avons traité. Il comporte 20 produits et 12 paramètres.

Nombre d'individus 20 Nombre de parametres 12
 Le poids est en derniere position

1	1.1.1.1.0.0.0.0.0.0.0.2.
2	0.0.0.0.1.1.1.0.0.0.0.3.
3	0.0.0.0.0.0.0.1.1.0.0.0.4.
4	0.0.0.0.0.0.0.0.0.0.1.1.1.4.
5	0.0.0.0.0.0.0.0.1.1.0.0.0.3.
6	0.0.0.0.1.1.1.0.0.0.0.0.2.
7	1.1.1.1.0.0.0.0.0.0.0.0.3.
8	0.0.0.0.0.0.0.0.0.0.1.1.1.3.
9	1.1.1.1.0.0.0.0.0.0.0.0.2.
10	0.0.0.0.0.0.0.1.1.0.0.0.3.
11	0.0.0.0.0.0.0.0.0.1.1.1.2.
12	1.1.1.1.0.0.0.0.0.0.0.0.4.
13	0.0.0.0.1.1.1.0.0.0.0.0.5.
14	0.0.0.0.0.0.0.1.1.0.0.0.2.
15	0.0.0.0.0.0.0.0.1.1.1.2.
16	0.0.0.0.0.0.0.1.1.0.0.0.3.
17	1.1.1.1.0.0.0.0.0.0.0.0.4.
18	0.0.0.0.1.1.1.0.0.0.0.0.4.
19	0.0.0.0.0.0.0.0.0.1.1.1.3.
20	1.1.1.1.0.0.0.0.0.0.0.0.2.

Fig. 2

A l'exécution, nous avons fourni un rayon de 1 et une distance minimale entre points initiaux de 2. Nous avons demandé 3 classes.

Compte tenu de la simplicité des données, le résultat a été obtenu en une itération.

Le système a retenu les points initiaux :

1 1 1 1 0 0 0 0 0 0 0 0	pour la classe 1.
Ce point a une densité de 6.	
0 0 0 0 0 0 0 1 1 0 0 0	pour la classe 2.
Ce point a une densité de 5.	
0 0 0 0 0 0 0 0 1 1 1	pour la classe 3.
Ce point a une densité de 5.	

Finalement, la classification automatique nous a conduit à (en utilisant la distance euclidienne) :

Classe 1

Représentant : 1 1 1 1 0 0 0 0 0 0 0 0

Numéros des produits de la classe : 1, 7, 9, 12, 17 et 20.

Classe 2

Représentant : 0 0 0 0 0,48 0,48 0,48 0,52 0,52 0 0 0

Numéros des produits de la classe : 2, 3, 5, 8, 10, 13, 14, 16, 18.

Classe 3

Représentant : 0 0 0 0 0 0 0 0 0 1 1 1

Numéros des produits de la classe : 4, 8, 11, 15, 19.

CONCLUSION

Ce travail est un premier pas dans la voie de la technologie de groupe. Le programme que nous présentons a été écrit pour obtenir rapidement des familles de produits en fonction des outils utilisés. Le poids associé à chaque produit représente le nombre moyen de produits de ce type à fabriquer durant une période de temps donnée. Bien entendu, on peut remplacer les machines par des opérations. Les produits sont alors regroupés en fonction de la similitude de transformations qu'ils subissent.

Notons encore que la nécessité de fixer à priori le nombre de classes est une contrainte importante. A notre connaissance, personne n'a encore réussi à se dégager de cette contrainte même si, dans certains cas, le problème est déplacé.

La démarche suivante, que nous présenterons dans un prochain rapport, aura pour but de regrouper les machines (ou opérations) dont l'activité essentielle est consacrée à une même famille de produits afin d'obtenir des îlots de fabrication.

BIBLIOGRAPHIE

- [1] J.MINOT - Y.LEMOINE - B.MUTEL,
"Implantation assistée par ordinateur de la Technologie de Groupe",
Congrès AFCET Productique et Robotique Avancée -Besançon -Novem-
bre 83.
- [2] Ensemble des travaux de E.DIDAY et collaborateurs.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique

