REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

P. COLLOMB

M. GONDRAN

Un algorithme efficace pour un arbre de classifications

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 11, n° 1 (1977), p. 31-49.

<http://www.numdam.org/item?id=RO_1977__11_1_31_0>

R.A.I.R.O. Recherche opérationnelle/Operations research
(vol. 11, n° 1, février 1977, p. 31 à 49)

UN ALGORITHME EFFICACE POUR UN ARBRE DE CLASSIFICATIONS (*)

par P. COLLOMB et M. GONDRAN (1)

Résumé. — *On utilise la recherche de l'arbre de longueur minimum pour définir un algorithme efficace pour déterminer l'arbre de classifications basé sur l'ultramétrique sous-dominante.*

1. INTRODUCTION

Dans tous les domaines scientifiques, que ce soit les sciences de la nature : géologie, botanique, zoologie, écologie, etc. ou les sciences de l'homme : psychologie, médecine, économie, politique, etc., nous nous trouvons souvent en face d'un grand nombre de données dont nous devons tirer le meilleur parti.

Ces données se présenteront en général comme un ensemble d'objets possédant certains caractères; la classification automatique se proposera de répondre aux questions suivantes :

(i) Définition d'une partition des objets en *classes* tel que deux objets d'une même classe soient « plus proches » que deux éléments appartenant à des classes différentes.

(ii) Définition d'un *arbre de classifications*, c'est-à-dire d'un ensemble de partitions qui soient compatibles entre elles (les classes de chaque partition sont incluses dans les classes de toute partition moins fine).

Cette dernière question englobe la question précédente.

Nous présentons ici un algorithme permettant de déterminer un arbre de classifications.

Ses particularités essentielles sont les suivantes :

(*a*) Il est basé sur la recherche de *l'arbre de longueur minimum* du graphe des données (on rappelle que l'arbre de longueur minimum d'un graphe dont les arêtes sont valuées, est un sous-ensemble de $n-1$ arêtes du graphe reliant entre eux tous les sommets et dont la somme des valuations est minimum).

Cet arbre de longueur minimum, très classique en théorie des graphes (*cf.* [1, 2]) a été introduit en classification tardivement ([3, 4, 5]). Il permet d'éclairer les approches des différents chercheurs en classification hiérarchique (*cf.* [6, 7, 8]).

(*) Reçu août 1975, révisé juin 1976.
(1) Électricité de France, Service informatique et Mathématiques appliquées, Clamart.

(*b*) *L'arbre de classifications* (ou une partie de celui-ci) lié à la *distance ultramétrique sous-dominante* est obtenu alors à partir de la connaissance de l'arbre de longueur minimum. Comme nous l'avons montré dans [9] chaque niveau de l'arbre de classifications correspond à une valeur propre, dans une structure algébrique appropriée, de la matrice des dissimilarités des objets.

Les sommets de ce niveau de l'arbre de classifications correspondent alors aux vecteurs propres générateurs du semi-module correspondant à cette valeur propre.

(*c*) *Il est très économe en place mémoire et en temps calcul.*

Si n est le nombre d'objets, il n'utilise que quelques vecteurs de dimension n et le nombre des opérations est en n^2 (*voir* § 3.3).

Au paragraphe 2, nous rappellerons certaines propriétés essentielles sur la distance ultramétrique sous-dominante et sa liaison avec l'arbre de longueur minimum et les classifications hiérarchiques indicées.

Au paragraphe 3, nous définirons l'algorithme permettant de tracer l'arbre de classifications (ou une partie de celui-ci).

Le programme de tracé en FORTRAN IV sur BENSON avec IBM 370 est donné en annexe.

2. DISTANCE ULTRAMÉTRIQUE ET ARBRE DE LONGUEUR MINIMUM

2.1. L'ultramétrique sous-dominante

Soit un ensemble de n objets. On suppose qu'entre chaque couple d'objets (i, j) on puisse calculer un indice de dissimilarité $d_{ij} \in R^+$, indice qui peut ne pas satisfaire les axiomes de distance.

(En fait il n'est pas nécessaire de supposer que cet indice existe entre tous les couples.)

La matrice $D = (d_{ij})$ peut être considérée comme la matrice d'incidence généralisée d'un graphe symétrique $G = (X, E)$ dont les arêtes seront valuées par les d_{ij}.

La *distance ultramétrique sous-dominante* $\delta(i, j)$ peut être alors définie de la façon suivante (*cf.* [7]) :

$$\delta(i, j) = \min_{\mu \in C_{ij}} \left[\max_{e \in \mu} d(e) \right],$$

où C_{ij} représente l'ensemble des chaînes de i à j; μ représente une chaîne de C_{ij}, e représente une arête de la chaîne μ.

En considérant $d(e)$ comme la capacité (inférieure) de l'arête e, la capacité (inférieure) d'une chaîne sera la valeur maximale des capacités (inférieures) des arêtes de cette chaîne et $\delta(i, j)$ représente donc la *capacité de la chaîne de capacité minimale entre i et j.*

2.2. L'arbre de longueur minimum

Considérons alors *un arbre de longueur minimum* ([2]) du graphe ayant pour longueur les $d(e)$. On a alors la propriété fondamentale suivante (*cf.* [10]) : *une chaîne de capacité minimale entre les sommets i et j est obtenue en considérant le chemin liant i à j dans un arbre de longueur minimum.* Ainsi la distance ultramétrique sous-dominante entre i et j est égale au maximum des valuations des arêtes de la chaîne liant i et j dans un arbre de longueur minimum.

La recherche d'un arbre de longueur minimum permet donc d'obtenir la distance ultramétrique sous-dominante entre tous les objets.

Nous allons tirer quelques conséquences très importantes de la propriété précédente.

(*a*) La distance ultramétrique sous-dominante prendra ses valeurs parmi les p ($\leqq n-1$ puisqu'un arbre à $n-1$ arêtes) coefficients de l'arbre de longueur minimum.

Si au départ tous les indices de dissimilarité sont différents, on a $p = n-1$.

(*b*) Considérons l'algorithme de Kruskal [1] pour déterminer l'arbre de longueur minimum. Il considère les arêtes du graphe G ordonnées par ordre croissant.

La première étape de l'algorithme consiste à prendre l'arête ayant la plus petite valuation.

La r-ième étape de l'algorithme consiste à choisir l'arête ayant la plus petite valuation dans les arêtes restantes *et* ne formant pas de cycle avec les arêtes déjà choisies.

L'algorithme s'arrête à l'étape $n-1$. Les $n-1$ arêtes choisies correspondent à un arbre de longueur minimum. Cet arbre de longueur minimum et donc l'ultramétrique sous-dominante ne dépendent que de l'« ordonnance » entre les indices de dissimilarité d_{ij}.

(*c*) Ordonnons les p coefficients de l'arbre de longueur minimum :

$$\lambda_1 \geqq \lambda_2 \geqq \ldots \geqq \lambda_k \geqq \ldots \geqq \lambda_p.$$

Alors l'arbre de classifications de la classification hiérarchique liée à l'ultramétrique sous-dominante aura p niveaux et les indices de ces niveaux seront les λ_k.

L'arbre de classifications sera obtenu en connectant à chaque étape les sommets adjacents de l'arbre de longueur minimum pris dans l'ordre des λ_k croissants.

La classification de niveau λ_k correspond aux composantes connexes de l'arbre de longueur minimum auquel on a enlevé les arêtes plus grandes que λ_k [15].

([2]) Il peut exister plusieurs arbres de longueur minimum si les arêtes du graphe G ne sont pas toutes différentes.

3. L'ALGORITHME ET LE PROGRAMME

3.1. Algorithme de l'arbre de longueur minimum

On détermine l'arbre de longueur minimum à partir de l'algorithme Prim [11].

L'intérêt essentiel de cet algorithme est de ne calculer qu'une fois les $n(n-1)/2$ distances entre individus *sans avoir à les stocker.*

Rappelons cet algorithme :

Il part d'un sommet quelconque $x_1 \in X$ du graphe G qui peut être considéré comme le sous-arbre de longueur minimum de G passant par le sommet x_1 et ayant 0 arête.

A la r-ième étape de l'algorithme, on considère le sous-arbre de longueur minimum de G passant par le sommet x_1 et ayant $r-1$ arêtes. Alors le sous-arbre de longueur minimum de G passant par le sommet x_1 et ayant r arêtes est obtenu à partir du sous-arbre précédent en lui ajoutant l'arête de valuation minimum joignant ce sous-arbre aux autres sommets du graphe.

L'algorithme s'écrit donc pratiquement :

(α) Poser $i_1 = 1$, $S_1 = \{ x_{i_1} \}$, $DD(1) = 0$ et $DD(i) = +\infty$ pour les autres sommets x_i.

(β) A l'itération r, on calcule pour tous les sommets x_i non-éléments de S_r l'indice de dissimilarité $d_{i,i}$:

- si $d_{i,i} < DD(i)$, on pose $DD(i) = d_{i,i}$ et $MB(i) = i_r$;
- si $d_{i,i} \geqq DD(i)$, on ne fait rien.

(γ) On définit alors $x_{i_{r+1}}$ par

$$DD(i_{r+1}) = \min_{x_i \in X \setminus S_r} DD(i),$$

puis

$$S_{r+1} = S_r \cup \{ x_{i_{r+1}} \},$$

$$r = r+1.$$

Si $r = n$, FIN, sinon aller en β.

L'ensemble S_r correspond à l'ensemble des sommets du sous-arbre de longueur minimum de G passant par le sommet x_1 et ayant $r-1$ arêtes.

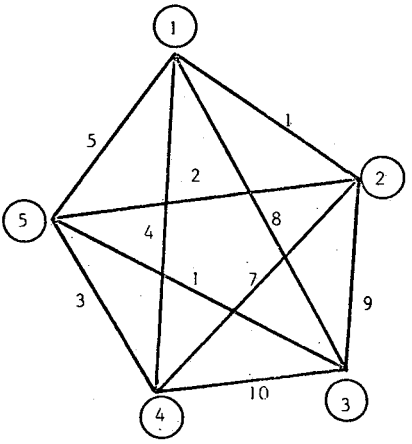
A l'étape β, chaque fois que $DD(i)$ est diminué, on garde en mémoire dans $MB(i)$ l'indice du sommet qui a permis cette diminution.

En fin d'algorithme, les arêtes de l'arbre de longueur minimum seront données par les couples de sommets $(i, MB(i))$: i sera appelé l'« aîné » et $MB(i)$ le « benjamin ». On remarque que $DD(i)$ correspond à la valuation de l'arête du graphe. Dans le programme S_r est défini par sa fonction indicatrice MASK; les vecteurs DD et MB ont le même nom dans le programme.

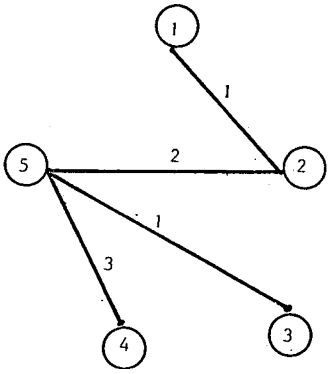
3.2. Algorithme de l'arbre de classifications

On détermine l'arbre de classifications à partir de l'arbre de longueur minimum, c'est-à-dire des vecteurs *MB* et *DD*.

Exemple : Considérons le graphe valué suivant :



L'arbre de longueur minimum est alors :



S'il a été obtenu à partir du sommet (1), on a $MB = (1, 5, 5, 2)$, $DD = (1, 1, 3, 2)$ pour $I = 2, 3, 4, 5$.

Ordonnons par distance croissante les arêtes de l'arbre de longueur minimum, donc le vecteur *DD*.

On a ici :

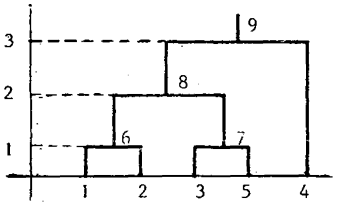
Distance.....	1	1	2	3
Aîné.....	2	3	5	4
Benjamin.....	1	5	2	5

Nous pouvons alors tracer la hiérarchie.

A chaque niveau de la hiérarchie, on affecte un numéro de « nœud », (le premier étant $n+1$), qui se substitue aux sommets qui l'ont engendré.

On a donc :

Distance.....	1	1	2	3
Aîné.....	2	3	7	4
Benjamin.....	1	5	6	8
Nœud.....	6	7	8	9



3.3. Temps et encombrement mémoire

Soit N le nombre d'individus et p le nombre de variables. La distance utilisée est la distance euclidienne classique dans $\mathbf{R}^p$. Les temps T sont en secondes, l'encombrement S en K octets.

On a les formules approchées :

$$T = 0{,}03 + 0{,}242.10^{-2} N + 0{,}13.10^{-4} N^2$$

(obtenue par régression linéaire avec $p = 6$),

$$S = 80 + 0{,}0266\, N + \frac{1}{256} p(N+2).$$

On a obtenu les valeurs suivantes (pour $p = 6$) :

N	100	200	300	400	500	600	700	800	900	1000
T	0,6	1	1,7	2,9	4,5	6,2	8,4	10,1	13,3	15,2
S	85	90	95	100	105	110	115	120	125	130

3.4. Entrées-Sorties

Les cartes paramètres sont lues dans le paragraphe 1 du programme principal.

Première carte : titre sur 80 colonnes.

Deuxième carte : sous-titre sur 80 colonnes.

Troisième carte : format de lecture des données sur 80 colonnes.

Quatrième carte : lecture des paramètres sur 4 colonnes chacun :

- NIND = nombre d'individus;
- NVAR = nombre de variables;
- IBEN = 0 si pas de tracé BENSON (1 si tracé);
- ITRØN = 0 si pas de troncage de l'arbre (1 si troncage);
- NDEB = nombre de types au départ si ITRØN = 1;
- ILECT = numéro du support de lecture des données;
- LIST = 1 si impression des données.

Cinquième carte : format d'impression des données sur 80 colonnes (si LIST = 1).

Les données sont lues sur ILECT suivant le format FØRM, avec l'identificateur en tête (alphanumérique sur 4 caractères maximum), puis NVAR valeurs réelles correspondant aux coordonnées de l'individu sur les variables.

Le troncage de l'arbre permet de ne tracer la hiérarchie qu'à partir d'un certain niveau.

Le programme fait un troncage automatique à 100 types si le nombre d'individus est supérieur à 100 (les distances étant ramenées à 0 au plus bas niveau).

Remarques : 1° Le programme est écrit en FORTRAN IV pour IBM 370.

2° Les temps de calcul concernent IBM 370-168.

3° Pour les installations ne possédant pas de tracé graphique, il suffit de mettre IBEN = 0, d'enlever la carte CALL CHENE en fin de programme principal et d'enlever le sous-programme CHENE.

BIBLIOGRAPHIE

1. J. B. KRUSKAL, *On the Shortest Spanning Subtree of a Graph*, Proc. Amer. Math. Soc., vol. 7, 1956, p. 48-50.
2. C. BERGE, *Théorie des graphes et ses applications*, Dunod, Paris, 1958.
3. J. C. GOWER et G. J. S. ROSS, *Minimum Spanning Trees and Single Linkage Cluster Analysis*, Appl. Statist., vol. 18, n° 1, 1969, p. 54-64.
4. C. T. ZAHN, *Graph Theoretical Methods for Detecting and Describing Gestalt Clusters*, I.E.E.E. Trans. and Comp., vol. C-20, 1971, p. 68-86.
5. R. L. PAGE, *Algorithm 479—A Minimal Spanning Tree Clustering Method*, Comm. ACM, vol. 17, n° 6, 1974, p. 321-323.
6. S. C. JOHNSON, *Hierarchical Clustering Schemes*, Psychometrica, vol. 32, 1967, p. 241-245.
7. M. ROUX, *Un algorithme pour construire une hiérarchie particulière*, Thèse de 3e cycle, (L.S.M. I.S.U.P.), 1968.
8. G. LERMAN, *Les basses de la classification automatique*, Gauthier-Villars, Paris, 1970.

9. M. GONDRAN, *Valeurs propres et vecteurs propres en classification hiérarchique*, R.A.I.R.O. Informatique Théorique, vol. 10, n° 3, 1976 (à paraître en anglais dans les *Actes du Congrès européen des Statistiques*, North Holland, avril 1977).
10. T. C. HU, *The Maximum Capacity Route Problem*, Ops. Res., vol. 9, 1961, p. 898-900.
11. R. C. PRIM, *Shortest Connexion Networks and Some Generalizations*, Bell. Syst. Tech. J., 1957, p. 1389-1401.
12. M. JAMBU, *Techniques de classification automatique*, Thèse de 3e cycle, (L.S.M. I.S.U.P.), 1972.
13. CARISTAN-GAUJARD-DELBOS, *Présentation de programmes de traces graphiques*, Note E.D.F. HI 1633/02 du 2 août 1974.
14. M. GONDRAN, *La structure algébrique des classifications hiérarchiques*, Note E.D.F. HI 1888/02 du 27 juin 1975 (à paraître dans les *Annales de l'I.N.S.E.E.*, n° 22-23, 1976).
15. C 3 E, *Analyse des données multidimensionnelles*, t. III.
16. P. COLLOMB, *Théorie des graphes et classification hiérarchique*, Note E.D.F. HI 1942/02 du 22 septembre 1975.

MAIN 1

PROGRAMME DE CLASSIFICATION HIERARCHIQUE
DISTANCE EUCLIDIENNE
ULTRAMETRIQUE INFERIEURE MAXIMALE (SOUS-DOMINANTE)
METHODE DE RECHERCHE DE L'ARBRE MINIMUM

```
C     NIND=NOMBRE D'INDIVIDUS
C     NVAR=NOMBRE DE VARIABLES
C     DIMENSION DES TABLEAUX :
C                              TITRE,METHD,FORM,FOR : 20
C                              XMOY,SIG : NVAR
C                              MASK,DD,MB,B,LIP : NIND
C                              X : NIND*SUP(2,NVAR)
C                              NOM,Y : 2*NIND
```

PARAGRAPHE 1
LECTURE DES PARAMETRES
IMPRESSION DES TITRES

```
      DIMENSION TITRE(20),METHD(20),FORM(20),FOR(20)
      DIMENSION XMOY(6),SIG(6)
      DIMENSION MASK(200),DD(200),MB(200),B(200),LIP(200)
      DIMENSION X(1200)
      DIMENSION NOM(400),Y(400)
      INTEGER TITRE,FORM,FOR
      INTEGER*2 MASK,MB,B
      LOGICAL LIP
      READ(5,60)(TITRE(I),I=1,20)
      READ(5,60)(METHD(I),I=1,20)
      READ(5,60)(FORM(I),I=1,20)
   60 FORMAT(20A4)
      WRITE(6,501)(TITRE(I),I=1,20)
      WRITE(6,501)(METHD(I),I=1,20)
  501 FORMAT(/T50,20A4//)
C     NIND= NOMBRE D'INDIVIDUS
C     NVAR= NOMBRE DE VARIABLES
C     IBEN= 0 SI PAS DE TRACE BENSON
C     ITRON= 0 SI PAS DE TRONCAGE DE L'ARBRE
C     NDEB= NOMBRE DE TYPES AU DEPART SI ITRON.NE.0
C     ILECT= NUMERO DU SUPPORT DE LECTURE DES OBSERVATIONS
C     LIST= 1 SI IMPRESSION DES OBSERVATIONS
      READ(5,2)NIND,NVAR,IBEN,ITRON,NDEB,ILECT,LIST
    2 FORMAT(20I4)
      IF(LIST.EQ.1) READ(5,60)(FOR(I),I=1,20)
      DO 3 I=1,NVAR
      XMOY(I)=0.
    3 SIG(I)=0.
```

MAIN 2

PARAGRAPHE 2
LECTURE ET IMPRESSION (EVENTUELLE) DES DONNEES
CALCUL ET EDITION DES MOYENNES ET ECART-TYPE

```
      K=1
      DO 13 J=1,NIND
      KK=K+NVAR-1
      READ(ILECT,FORM) NOM(J),(X(KJ),KJ=K,KK)
      IF(LIST.EQ.1) WRITE(6,FOR) NOM(J), (X(KJ),KJ=K,KK)
      DO 13 I=1,NVAR
      XMOY(I)=XMOY(I)+X(K)
      SIG(I)=SIG(I)+X(K)*X(K)
   13 K=K+1
      DN=NIND
      DO 5 I=1,NVAR
      XMOY(I)=XMOY(1)/DN
    5 SIG(I)=SQRT(SIG(I)/DN-XMOY(I)*XMOY(I))
      WRITE(6,502)
  502 FORMAT(1H1)
      WRITE(6,500)((I,XMOY(I),SIG(I)),I=1,NVAR)
  500 FORMAT(1X,'VARIABLE',I3,5X,'MOYENNE',E12.4,5X,'ECART-TYPE',E12.
     14)
      K=1
      DO 4 J=1,NIND
      DO 4 I=1,NVAR
      X(K)=X(K)/SIG(I)
    4 K=K+1
```

PARAGRAPHE 3
CALCUL ET EDITION DE L'ARBRE MINIMUM

```
      CALL CODE(NIND,NOM)
      WRITE(6,502)
      DO 6 I=2,NIND
      MASK(I)=0
      DD(I)=0.
      MB(I)=1
      K=(I-1)*NVAR
      DO 6 J=1,NVAR
      DD(I)=DD(I)+(X(J)-X(K+J))**2
    6 CONTINUE
      J=1
      DO 7 KK=2,NIND
      MASK(J)=1
      DS=1.0E+60
      KJ=NVAR*(J-1)
      DO 8 I=1,NIND
      IF(MASK(I))8,9,8
    9 T=0.
      KI=NVAR*(I-1)
      DO 14 L=1,NVAR
   14 T=T+(X(KI+L)-X(KJ+L))**2
      IF(DD(I)-T)11,11,12
   12 DD(I)=T
      MB(I)=J
```

MAIN 3

```
      IF(DS.LE.DD(I)) GO TO 8
        DS=DD(I)
        K=I
    8 CONTINUE
        J=K
    7 CONTINUE
        WRITE(6,504)
  504 FORMAT(1X,13HARBRE MINIMUM///)
        NIN=NIND-1
        DO 51 I=1,NIN
        DS=1.0E+60
        DO 52 J=2,NIND
        IF(DS-DD(J))52,52,53
   53 DS=DD(J)
        K=J
   52 CONTINUE
        B(I)=K
        MASK(I)=MB(K)
        Y(I)=DD(K)
   51 DD(K)=1.0E+60
        DO 54 I=1,NIND
   54 DD(I)=Y(I)
        NINI=NIN/10
        NINO=NIN-10*NINI
        IF(NINI)15,15,16
   16 DO 55 I=1,NINI
        JJ=10*(I-1)+1
        JK=JJ+9
        WRITE(6,505)(NOM(B(J)),J=JJ,JK)
        WRITE(6,506)(NOM(MASK(J)),J=JJ,JK)
        WRITE(6,507)(DD(J),J=JJ,JK)
   55 WRITE(6,509)
   15 IF(NINO)56,56,57
   57 JJ=JK+1
        JK=JJ+NINO-1
        WRITE(6,505)(NOM(B(J)),J=JJ,JK)
        WRITE(6,506)(NOM(MASK(J)),J=JJ,JK)
        WRITE(6,507)(DD(J),J=JJ,JK)
        WRITE(6,509)
  505 FORMAT(1X,'AINE     ',10(4X,A4,3X))
  506 FORMAT(1X,'BENJAMIN',10(4X,A4,3X))
  507 FORMAT(1X,'DISTANCE',10(E11.3))
  509 FORMAT(1X,120(1H*))
   56 CONTINUE
```

PARAGRAPHE 4

APPEL DU SOUS-PROGRAMME TRONC

```
      IF(NDEB-NIND)100,100,101
  101 NDEB=NIND
      ITRON=0
      WRITE(6,502)
      WRITE(6,102) NIND
  102 FORMAT(1X,'LE NOMBRE DE TYPES AU DEPART EST RAMENE  ',I3)
  100 IF(ITRON.EQ.0.AND.NDEB.NE.NIND) NDEB=NIND
      IF(IBEN.EQ.0.ØR.NIND.LE.100) GO TO 21
      NDEB=100
      ITRON=1
   21 CALL TRONC(NIND,B,MASK,LIP,MB,NDEB,ITRON,NOM,X,Y,DD)
```

MAIN 4

```
      IF(ITRON.NE.0) NIN=NDEB-1
```

PARAGRAPHE 5
IMPRESSION DE L'ARBRE HIERARCHIQUE

```
      WRITE(6,502)
      WRITE(6,510)
      NINI=NIN/10
      NINØ=NIN-10*NINI
      DO 45 I=1,NINI
      JJ=10*(I-1)+1
      JJD=JJ+NIND
      JK=JJ+9
      JKD=JK+NIND
      WRITE(6,505)(NOM(B(J)),J=JJ,JK)
      WRITE(6,506)(NOM(MASK(J)),J=JJ,JK)
      WRITE(6,511)  (J,J=JJD,JKD)
      WRITE(6,507)(DD(J),J=JJ,JK)
   45 WRITE(6,509)
      IF(NINO)46,46,47
   47 JJ=JK+1
      JJD=JJ+NIND
      JK=JJ+NINO-1
      JKD=JK+NIND
      WRITE(6,505)(NOM(B(J)),J=JJ,JK)
      WRITE(6,506)(NOM(MASK(J)),J=JJ,JK)
      WRITE(6,511)  (J,J=JJD,JKD)
      WRITE(6,507)(DD(J),J=JJ,JK)
      WRITE(6,509)
  510 FORMAT(1X,'ARBRE HIERARCHIQUE'///)
  511 FORMAT(1X,'NOEUD     ',10(3X,I4,4X))
   46 CONTINUE
      IF(IBEN.EQ.0) GO TO 9999
```

PARAGRAPHE 6
APPEL DU PROGRAMME DE TRACE BENSON SI IBEN.NE.0

```
      N=NIN+1
      CALL CHENE(N,B,MASK,DD,TITRE,METHD,NOM,LIP,MB,X,Y,NIND)
 9999 STOP
      END
```

TRONC 1

```
      SUBROUTINE TRONC(N,A,B,LIP,IP,ND,ITRON,NOM,X,Y,D)
      DIMENSION A(1),B(1),LIP(1),IP(1),NOM(1),X(1),Y(1),TIT(20)
      DIMENSION D(1)
      LOGICAL LIP
      INTEGER*2 A,B,IP,TIT,NTIT,MTIT
      DO 1 I=1,N
      LIP(I)=.FALSE.
    1 IP(I)=I
      K=N-ND
      IF(K)2,2,3
```

PARAGRAPHE 1

CALCUL ET IMPRESSION DES REGROUPEMENTS SI ITRON.NE.0

```
    3 CALL ARDU(K,A,B,LIP,IP,X,Y,N,ITRON,IAK,N)
      WRITE(6,502)
  502 FORMAT(1H1)
      WRITE(6,503) ND
  503 FORMAT(1X,'REGROUPEMENT EN',I4,'TYPES'///)
      KJ=0
      DO 9 I=1,IAK
      IF(KJ.GT.IAK) GO TO 10
      KK=0
      IF(Y(I).EQ.0.) GO TO 9
      KK=KK+1
      TIT(KK)=X(I)
      NTIT=TIT(KK)
      CALL IMP(NTIT,NOM)
      KJ=KJ+1
      II=I+1
      DO 7 J=II,IAK
      IF(Y(J).EQ.0.)GO TO 7
      IF(Y(I).NE.Y(J)) GO TO 7
      KK=KK+1
      TIT(KK)=X(J)
      MTIT=TIT(KK)
      CALL CHANGE(A,B,NTIT,MTIT,N,ND)
      IF(KK.EQ.20) CALL IMPRES(KK,TIT,NOM)
      Y(J)=0.
      KJ=KJ+1
    7 CONTINUE
      IF(KK.NE.0) CALL IMPRES(KK,TIT,NOM)
      Y(I)=0.
    9 CONTINUE
```

TRONC 2

PARAGRAPHE 2
REPOSITIONNEMENT DES TABLEAUX PERMETTANT LE CALCUL DE L'ARBRE HIERARCHIQUE SI ITRON.NE.0

```
 10 M=ND-1
    DIST=D(K)
    DØ 11 I=1,M
    A(I)=A(I+K)
    D(I)=D(I+K)-DIST
 11 B(I)=B(I+K)
    DØ 12 I=1,N
    LIP(I)=.FALSE.
 12 IP(I)=I
```

PARAGRAPHE 3
IMPRESSION DE L'ARBRE MINIMUM TRONQUE

```
    WRITE(6,502)
    WRITE(6,504)
504 FORMAT(1X,'ARBRE MINIMUM TRONQUE'///)
    NINI=M/10
    NINO=M-10*NINI
    DO 55 I=1,NINI
    JJ=10*(I-1)+1
    JK=JJ+9
    WRITE(6,505)(NOM(A(J)),J=JJ,JK)
    WRITE(6,505)(NOM(B(J)),J=JJ,JK)
    WRITE(6,507) (D(J),J=JJ,JK)
 55 WRITE(6,509)
    IF(NINO)56,56,57
 57 JJ=JK+1
    JK=JJ+NINO-1
    WRITE(6,505)(NOM(A(J)),J=JJ,JK)
    WRITE(6,505)(NOM(B(J)),J=JJ,JK)
    WRITE(6,507)(D(J),J=JJ,JK)
    WRITE(6,509)
505 FORMAT(1X,'AINE     ',10(4X,A4,3X))
506 FORMAT(1X,'BENJAMIN',10(4X,A4,3X))
507 FORMAT(1X,'DISTANCE',10(E11.3)
509 FORMAT(1X,120(1H*))
 56 CONTINUE
```

PARAGRAPHE 4
CALCUL DE L'ARBRE HIERARCHIQUE

```
  2 IF(ITRON.EQ.0) M=N-1
    INDICE=0
    CALL ARDU(M,A,B,LIP,IP,X,Y,M,INDICE,IAK,N)
    RETURN
    END
```

ARDU 1

```
      SUBROUTINE ARDU(K,A,B,LIP,IP,X,Y,N,INDICE,IAK;NB)
      DIMENSION A(1), B(1), LIP(1), IP(1), X(1),Y(1)
      LOGICAL LIP
      INTEGER*2 A,B,IP
```

PARAGRAPHE 1
CALCUL DE L'ARBRE HIERARCHIQUE

```
      IAL=1
      DO 1 I=1,K
      IA=0
      IB=0
      IF(LIP(A(I))) GO TO 2
      LIP(A(I))=.TRUE.
      IP(A(I))=I+NB
      X(IAL)=A(I)
      Y(IAL)=I
      IAL=IAL+1
      GO TO 3
    2 U=A(I)
      IA=1
      J=IP(A(I))
      DO 4 L=1,NB
      IF(IP(L).EQ.J) IP(L)=I+NB
    4 CONTINUE
      A(I)=J
    3 CONTINUE
      IF(LIP(B(I))) GO TO 5
      LIP(B(I))=.TRUE.
      IP(B(I))=I+NB
      X(IAL)=B(I)
      Y(IAL)=I
      IAL=IAL+1
      GO TO 6
    5 V=B(I)
      IB=1
      J=IP(B(I))
      DO 7 L=1,NB
      IF(IP(L).EQ.J) IP(L)=I+NB
    7 CONTINUE
      B(I)=J
    6 CONTINUE
```

PARAGRAPHE 2
CALCUL DES REGROUPEMENTS SI INDICE.NE.0

```
      IAK=IAL-1
      IF(INDICE.EQ.0) GO TO 1
      IF(IA.EQ.0.AND.IB.EQ.0) GO TO 1
      IF(IA.EQ.0.OR.IB.EQ.0) GO TO 8
      DO 10 M=1,IAK
      IF(X(M)-U)10,11,10
   10 CONTINUE
   11 JA=Y(M)
      DO 12 M=1,IAK
      IF(X(M)-V)12,9,12
   12 CONTINUE
```

ARDU 2

```
    9 JB=Y(M)
      IF(JA-JB)13,14,14
   13 TA=JA
      TB=JB
      GO TO 15
   14 TA=JB
      TB=JA
   15 DO 16 M=1,IAK
      IF(Y(M).EQ.TB) Y(M)=TA
   16 CONTINUE
      GO TO 1
    8 IF(IA.EQ.0) U=V
      DO 17 M=1,IAL
      IF(X(M)-U)17,18,17
   17 CONTINUE
   18 Y(IAK)=Y(M)
    1 CONTINUE
      RETURN
      END
```

IMP

```
      SUBROUTINE IMP(NTIT,NOM)
```

IMPRESSION DU NOM DES TYPES

```
      DIMENSION NOM(1)
      INTEGER*2 NTIT
      WRITE(6,1) NOM(NTIT)
    1 FORMAT(/////,1X,'LES INDIVIDUS SUIVANTS SONT REGROUPES SOUS LE NOM
     1 : ',A4/56(1H*)//)
      RETURN
      END
```

CHANGE

```
      SUBROUTINE CHANGE(A,B,NTIT,MTIT,N,ND)
```

ATTRIBUTION DU NOM DU TYPE A TOUT INDIVIDU REGROUPE

```
      DIMENSION A(1),B(1)
      INTEGER*2 A,B,MTIT,NTIT
      J=N-ND+1
      K=N-1
      DO 1 I=J,K
      IF(A(I).EQ.MTIT) A(I)=NTIT
      IF(B(I).EQ.MTIT) B(I)=NTIT
    1 CONTINUE
      RETURN
      END
```

IMPRES

```
      SUBROUTINE IMPRES(KK,TIT,NOM)

      IMPRESSION DES INDIVIDUS REGROUPES DANS UN TYPE

      DIMENSION TIT(1),NOM(1)
      INTEGER*2 TIT
      LOGICAL NOM
      WRITE(6,1) (NOM(TIT(I)),I=1,KK)
    1 FORMAT(1X,20(A4,1X))
      KK=0
      RETURN
      END
```

CODE

```
      SUBROUTINE CODE(N,NOM)

      CODIFICATION DU NOM DES NOEUDS EN FORMAT A

      DIMENSION NOM(1)
      INTEGER ZERO
      DATA ZERO/ZF0
      LL=N+1
      NN=2*N
      DO 1 L=LL,NN
      M=L
      I=M/1000
      M=M-I*1000
      J=M/100
      M=M-J*100
      K=M/10
      M=M-K*10
      NOM(L)     =((((I+ZERO)*2**8+(J+ZERO))*2**8+(K+ZERO))*2**8+(M+ZERO)
     1)
    1 CONTINUE
      RETURN
      END
```

CHENE 1

```
      SUBROUTINE CHENE(CARDJ,A,B,D,TITRE,METHD,NOM,LIP,IP,X,Y,NIND)

      TRACE DE L'ARBRE HIERARCHIQUE SUR TABLE BENSON

      INTEGER CARDJ,TITRE,AI,BI
      INTEGER*2 A,B,IP
      LOGICAL LIP
      DIMENSION A(1),B(1),D(1),TITRE(1),METHD(1),NOM(1),LIP(1),IP(1),
     1 X(1),Y(1)
      DATA IMAT/2HBN/
      NTRACE=33
      CALL BNCHOI(IMAT,NTRACE)
      XTJ=100.* (CARDJ+2)
      CALL BNCADA(0.,0.,XTJ,12000.,1)
      AMAX=30.
      XIN=-2
      XSU=CARDJ+2
      YIN=-5
      YSU=AMAX
      CALL BNCADS(XIN,YIN,XSU,YSU)
      CALL BNCARA(1,3.,3.,2,1,2,0)
      CALL BECARS(5.,-4.,0,METHD,80)
      CALL BECARS(0.,-3.,0,TITRE,80)
      CALL BNCARA(1,3.,3.,2,1,2,90)
      IS=NIND+CARDJ-1
      I=1
      N=1
      LIP(1)=.FALSE.
      IP(1)=IS
  100 II=I
      IS=A(IS-NIND)
  120 IF(IS.LE.NIND) GO TO 130
      I=II+1
      IP(I)=IS
      LIP(I)=.FALSE.
      GOTO 100
  130 X(IS)=N
      Y(IS)=0
      N=N+1
      IF(N.GT.CARDJ)GOTO 199
      GOTO 140
  110 II=II-1
  140 IF(LIP(II))GOTO 110
      LIP(II)=.TRUE.
      JS=IP(II)-NIND
      IS=B(JS)
      GOTO 120
  199 CONTINUE
      DMAX=AMAX-3.
      LCARD=CARDJ-1
      DO 200 I=1,LCARD
      J= I+NIND
      Y(J)=(D(I)*DMAX)/D(LCARD)
      AI=A(I)
      BI=B(I)
      IF(AI.GT.NIND) GO TO 210
```

CHENE 2

```
      CALL BECARS(X(AI),Y(AI)-1.,0,NOM(AI),4)
  210 CONTINUE
      CALL BETRAS(X(AI),Y(AI),0)
      CALL BETRAS(X(AI),Y(J),1)
      CALL BETRAS(X(BI),Y(J),1)
      CALL BETRAS(X(BI),Y(BI),1)
      IF(BI.GT.NIND) GO TO 220
      CALL BECARS(X(BI),Y(BI)-1,0,NOM(BI),4)
  220 CONTINUE
      X(J)=(X(AI)+X(BI))*0.5
  200 CONTINUE
      CALL BOSTOP
      RETURN
      END
```