ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XVI, 2 (1978)

J. KUCHARCZYK (Wrocław)

CLUSTER ANALYSIS BY THE METHODS OF HUBERT

1. INTRODUCTION

In this paper the three methods of cluster analysis proposed by Hubert in [1] are presented as a set of procedures in ALGOL 60. The set consists of 6 procedures. Procedure *divmethod* incorporates the main algorithm, and procedures *divisionA*, *divisionB* and *divisionC*, realizing step 3 of methods A, B and C, respectively, are called within its body. Two auxiliary procedures are needed: (i) procedure *sort* which sorts in increasing order the indicated part of an integer array, and (ii) real function *diameter* which calculates the greatest distance between objects of an indicated group, giving also the object numbers which realize this distance.

2. PROCEDURE DECLARATIONS

2.1. Procedure *divmethod*. This procedure realizes the main algorithm. To initialize, it should be called with $lg = 0$ first. On exit, lg will be equal to 1. Each further call of *divmethod* will give lg increased by 1 on exit, indicating that a new cluster has been formed. The calls may be continued until $lg = n$ on exit. Every call with $lg = k$ will give on exit a division of the objects into $k+1$ groups. The members of the formed groups may be printed out and all needed characteristics such as the within-group or between-group sums of squares of distances may be calculated outside the procedure body.

Data:

n — the number of objects;

$d[1: n \times (n-1) \div 2]$ — the lower half of the distance matrix; it is not changed during the calls of *divmethod*;

lg — the number of existing groups; to initialize, call *divmethod* with $lg = 0$;

$lsg[0: n]$ — integer array containing in $lsg[i]$ $(i = 1, 2, \ldots, lg)$

the cumulated numbers of objects in groups from 1 to i ($lsg[0] = 0$); thus, the number of objects in the group k is given by $lsg[k] - lsg[k-1]$ ($k = 1, 2, \ldots, lg$);

nrg[*1*: *n*] — integer array containing the object numbers; the numbers of the objects belonging to the group i are contained in $nrg[lsg[i-1]+1 : lsg[i]]$ ($i = 1, 2, \ldots, lg$); for each group the contents of *nrg* is arranged in increasing order;

diam[*1*: *n*] — real array containing the maximal distances between objects in each group; *diam*[*i*] ($i = 1, 2, \ldots, lg$) contains the maximal distance between objects belonging to the group i;

g1, g2[*1*: *n*] — integer arrays containing the numbers of objects which realize the maximal distances in groups; the numbers of objects being the most distant in the group i ($i = 1, 2, \ldots, lg$) are contained in *g1*[*i*] and *g2*[*i*] ($g1[i] < g2[i]$);

division — procedure realizing the division method chosen (see below);

sort — procedure which sorts in part the integer array *nrg* (see below);

diameter — real procedure calculating the greatest distance between objects belonging to an indicated group (see below).

Results:

lg — the number of determined groups; it is increased by 1 with respect to the value of *lg* entered;

lsg, nrg, diam, g1, g2 — arrays containing the updated information about the new groups formed (see the appropriate descriptions under data).

2.2. Procedures *divisionA*, *divisionB* **and** *divisionC*. Although performing different calculations, all those procedures have identical entry and exit parameters, therefore a common description is given.

Data:

lp, lk — the first and last elements ($lp < lk$) of *nrg* indicating the numbers of objects belonging to the group which is to be divided into two groups;

nr1, nr2 — the numbers of the objects ($nr1 < nr2$) being the most distant in this group;

nrg[*1*: *n*] — integer array containing the object numbers (see procedure *divmethod*);

```
procedure divmethod(n,d,lg,lsg,nrg,diam,g1,g2,division,sort,
 diameter);
 value n;
 integer n,lg;
 integer array lsg,nrg,g1,g2;
 array d,diam;
 procedure division,sort;
 real procedure diameter;
 begin
  integer i,g,p1,p2,k1,k2,nr1,nr2;
  real maxdiam;
  if lg=0
   then
   begin
    for i:=1 step 1 until n do
     nrg[i]:=i;
    lsg[0]:=0;
    lsg[1]:=n;
    diam[1]:=diameter(1,n,nrg,d,nr1,nr2);
    g1[1]:=nr1;
    g2[1]:=nr2;
    lg:=1
   end lg=0
   else
   begin
    g:=1;
    maxdiam:=diam[1];
    for i:=2 step 1 until lg do
     if diam[i]>maxdiam
      then
```

```
      begin
       maxdiam:=diam[i];
       g:=i
      end maxdiam,i;
    division(lsg[g-1]+1,lsg[g],g1[g],g2[g],nrg,d,sort,p1,k1,p2,
    k2);
    for i:=lg step -1 until g+1 do
     begin
      lsg[i+1]:=lsg[i];
      diam[i+1]:=diam[i];
      g1[i+1]:=g1[i];
      g2[i+1]:=g2[i]
     end i;
    lg:=lg+1;
    lsg[g]:=k1;
    diam[g]:=diameter(p1,k1,nrg,d,nr1,nr2);
    g1[g]:=nr1;
    g2[g]:=nr2;
    g:=g+1;
    lsg[g]:=k2;
    diam[g]:=diameter(p2,k2,nrg,d,nr1,nr2);
    g1[g]:=nr1;
    g2[g]:=nr2
   end lg>0
end divmethod
```

```
procedure divisionA(lp,lk,nr1,nr2,nrg,d,sort,lp1,lk1,lp2,lk2);
 value lp,lk,
 integer lp,lk,nr1,nr2,lp1,lk1,lp2,lk2;
 integer array nrg;
 array d;
 procedure sort;
 begin
  integer l,l1,l2,k,k1,i,j,m,p;
  real max,dij;
  Boolean first;
  procedure exch(i,j);
   value i,j;
   integer i,j;
   begin
    integer r;
    r:=nrg[i];
    nrg[i]:=nrg[j];
    nrg[j]:=r
   end exch;
  l:=lk-lp+1;
  l1:=l2:=1;
  exch(lp,nr1);
  exch(lk,nr2);
  lp1:=lk1:=lp;
  lp2:=lk2:=lk;
  lp:=lp+1;
  lk:=lk-1;
cycle:
  if l1+l2=l
   then go to exit;
```

```
max:=.0;
for i:=lp1 step 1 until lk1 do
 begin
  k:=nrg[i];
  k1:=(k-1)×(k-2)+2;
  for j:=lp step 1 until lk do
   begin
    m:=nrg[j];
    dij:=d[if k>m then k1+m else (m-1)×(m-2)+2+k];
    if dij>max
     then
     begin
      max:=dij;
      p:=j
     end dij>max
   end j
 end i;
first:=true;
for i:=lp2 step 1 until lk2 do
 begin
  k:=nrg[i];
  k1:=(k-1)×(k-2)+2;
  for j:=lp step 1 until lk do
   begin
    m:=nrg[j];
    dij:=d[if k>m then k1+m else (m-1)×(m-2)+2+k];
    if dij>max
     then
     begin
      max:=dij;
```

```
          p:=j;
          first:=false
         end dij>max
       end j
     end i;
    if first
     then
     begin
      exch(p,lk);
      l2:=l2+1;
      lk:=lk-1;
      lp2:=lp2-1
     end first
     else
     begin
      exch(p,lp);
      l1:=l1+1;
      lp:=lp+1;
      lk1:=lk1+1
     end ¬first;
   go to cycle;
exit:
   if l1>1
    then sort(lp1,lk1,nrg);
   if l2>1
    then sort(lp2,lk2,nrg)
 end divisionA
```

```
procedure divisionB(lp,lk,nr1,nr2,nrg,d,sort,lp1,lk1,lp2,lk2);
 value lp,lk;
 integer lp,lk,nr1,nr2,lp1,lk1,lp2,lk2;
 integer array nrg;
 array d;
 procedure sort;
 begin
  integer l,l1,l2,k,k1,i,j,m,p;
  real min,dij;
  Boolean first;
  procedure exch(i,j);
   value i,j;
   integer i,j;
   begin
    integer r;
    r:=nrg[i];
    nrg[i]:=nrg[j];
    nrg[j]:=r
   end exch;
  l:=lk-lp+1;
  l1:=l2:=1;
  exch(lp,nr1);
  exch(lk,nr2);
  lp1:=lk1:=lp;
  lp2:=lk2:=lk;
  lp:=lp+1;
  lk:=lk-1;
cycle:
  if l1+l2=l
   then go to exit;
```

```
min:=99;
for i:=lp1 step 1 until lk1 do
 begin
  k:=nrg[i];
  k1:=(k-1)×(k-2)+2;
  for j:=lp step 1 until lk do
   begin
    m:=nrg[j];
    dij:=d[if k>m then k1+m else (m-1)×(m-2)+2+k];
    if dij<min
     then
     begin
      min:=dij;
      p:=j
     end dij<min
   end j
 end i;
first:=true;
for i:=lp2 step 1 until lk2 do
 begin
  k:=nrg[i];
  k1:=(k-1)×(k-2)+2;
  for j:=lp step 1 until lk do
   begin
    m:=nrg[j];
    dij:=d[if k>m then k1+m else (m-1)×(m-2)+2+k];
    if dij<min
     then
     begin
      min:=dij;
```

```
            p:=j;
            first:=false
           end dij<min
         end j
       end i;
      if first
       then
       begin
        exch(p,lp);
        l1:=l1+1;
        lp:=lp+1;
        lk1:=lk1+1
       end first
       else
       begin
        exch(p,lk);
        l2:=l2+1;
        lk:=lk-1;
        lp2:=lp2-1
       end ¬first;
      go to cycle;
    exit:
      if l1>1
       then sort(lp1,lk1,nrg);
      if l2>1
       then sort(lp2,lk2,nrg)
     end divisionB
```

```
procedure divisionC(lp,lk,nr1,nr2,nrg,d,sort,lp1,lk1,lp2,lk2);
 value lp,lk;
 integer lp,lk,nr1,nr2,lp1,lk1,lp2,lk2;
 integer array nrg;
 array d;
 procedure sort;
 begin
  integer l,l1,l2,p,p1,i,j,k,m,q;
  real min,max,dij;
  Boolean first,prim;
  procedure exch(l1,l2);
   value l1,l2;
   integer l1,l2;
   begin
    integer r;
    r:=nrg[l1];
    nrg[l1]:=nrg[l2];
    nrg[l2]:=r
   end exch;
  l:=lk-lp+1;
  l1:=l2:=1;
  exch(lp,nr1);
  exch(lk,nr2);
  lp1:=lk1:=lp;
  lp2:=lk2:=lk;
  lp:=lp+1;
  lk:=lk-1;
cycle:
  if l1+l2=l
   then go to exit;
```

```
max:=.0;
for i:=1p step 1 until 1k do
 begin
  min:=10 99;
  p:=nrg[i];
  p1:=(p-1)×(p-2)÷2;
  for j:=1p1 step 1 until 1k1 do
   begin
    k:=nrg[j];
    dij:=d[if k>p then (k-1)×(k-2)÷2+p else p1+k];
    if dij<min
     then min:=dij
   end j;
  first:=true;
  for j:=1p2 step 1 until 1k2 do
   begin
    k:=nrg[j];
    dij:=d[if k>p then (k-1)×(k-2)÷2+p else p1+k];
    if dij<min
     then
     begin
      min:=dij;
      first:=false
     end dij<min
   end j;
  if max<min
   then
   begin
    max:=min;
```

```
        q:=i;
        prim:=first
      end max<min
    end i;
   if prim
    then
    begin
     exch(q,lp);
     l1:=l1+1;
     lp:=lp+1;
     lk1:=lk1+1
    end prim
    else
    begin
     exch(q,lk);
     l2:=l2+1;
     lk:=lk-1;
     lp2:=lp2-1
    end ¬prim;
   go to cycle;
exit:
   if l1>1
    then sort(lp1,lk1,nrg);
   if l2>1
    then sort(lp2,lk2,nrg)
 end divisionC
```

```
real procedure diameter(lp,lk,nrg,d,nr1,nr2);
 value lp,lk;
 integer lp,lk,nr1,nr2;
 integer array nrg;
 array d;
 begin
  integer i,j,k;
  real max,dij;
  max:=.0;
  for i:=lp+1 step 1 until lk do
   begin
    k:=nrg[i];
    k:=(k-1)×(k-2)+2;
    for j:=lp step 1 until i-1 do
     begin
      dij:=d[k+nrg[j]];
      if dij>max
       then
       begin
        max:=dij;
        nr1:=j;
        nr2:=i
       end dij>max
     end j
   end i;
  diameter:=max
 end diameter
```

```
procedure sort(p,k,a);
 value p,k;
 integer p,k;
 integer array a;
 begin
  integer i,j,l,m,r,s,t;
  m:=p-k-1;
  for m:=m+2 while m<0 do
   begin
    l:=k+m;
    for j:=p step 1 until l do
     begin
      for i:=j step m until p do
       begin
        t:=i-m;
        r:=a[i];
        s:=a[t];
        if r≤s
         then go to endj
         else
         begin
          a[i]:=s;
          a[t]:=r
         end a[i]>a[i-m]
       end i;
endj:end j
   end m
 end sort
```

$d[1: n \times (n-1) \div 2]$ — the lower half of the distance matrix (see procedure *divmethod*);

sort — procedure which sorts in part the integer array *nrg* (see below).

Results:

$nrg[1: n]$ — integer array containing the updated object numbers after formation of the new groups (see procedure *divmethod*);

lp1, lk1, lp2, lk2 — integers indicating the lower (*lp1, lp2*) and upper (*lk1, lk2*) bounds of the object numbers of the two new groups in *nrg*; these numbers are needed in procedure *divmethod*.

2.3. Real procedure *diameter*. This real function finds the maximal distance between objects contained in the group with object numbers from $nrg[lp]$ to $nrg[lk]$.

Data:

lp, lk — the first and last elements ($lp \leqslant lk$) of *nrg* indicating the object numbers of the group for which the calculations are to be done;

$nrg[1: n]$ — integer array containing the object numbers (see procedure *divmethod*);

$d[1: n \times (n-1) \div 2]$ — the lower half of the distance matrix (see procedure *divmethod*).

Results:

diameter — the greatest distance between objects in the given group;

nr1, nr2 — integers indicating the elements of *nrg* which realize the greatest distance.

2.4. Procedure *sort*. This procedure sorts in increasing order a part of an integer array by the Shell method (see [2], p. 61). Any other similar procedure can be used instead.

Data:

p, k — the first and last indices of the part of *a* which is to be sorted ($p \leqslant k$);

$a[1: n]$ — integer array containing the numbers to be sorted.

Results:

$a[1: n]$ — the sorted array in which elements from $a[p]$ to $a[k]$ are now sorted in increasing order, and the remaining elements are unchanged.

3. METHOD USED

All three procedures are of the hierarchical divisive type, i.e. one starts with all objects forming a single group which is then divided into two groups, afterwards one of the two groups is divided again into two new groups giving altogether a division of the objects into three groups, etc., until there are as many groups as objects. The procedures differ only in the way the division of one of the existing groups into two new groups is performed. They work as follows:

Assume that k groups $G_1, G_2, \ldots, G_k$ have already been formed.

Step 1. For each existing group determine the most distant objects and among them choose that pair, say N_r and N_s belonging to the group G_i, for which this distance is maximal.

Step 2. The group G_i is divided into two new groups G' and G''. Assign the object N_r to G' and the object N_s to G'', removing both of them from the group G_i.

Step 3. This step is different for the three methods.

Method A. Among the objects remaining in G_i determine the most distant from all objects being already in G' and G''. Assign it to G'' (G', respectively) if it is the most distant object from G' (G'', respectively). Remove this object from G_i.

Method B. Among the objects remaining in G_i determine the one which has the smallest distance from all objects being already in G' and G''. Assign it to the group G' (G'', respectively) if this smallest distance is realized by an object already belonging to G' (G'', respectively). Remove this object from G_i.

Method C. Among the objects remaining in G_i determine this one for which the smallest distance from all objects being already in groups G' and G'' is maximal. Assign it to that group which contains the object for which the maximum of minimal distances is realized, and remove it from the group G_i.

Step 3 is performed until the objects of the group G_i are exhausted. After performing steps 1-3 of the algorithm, $k+1$ groups are formed.

4. CERTIFICATION AND REMARKS

All procedures have been run on the Odra 1204 computer with the Algol MT compiler. The obtained results were correct.

The computer memory requirements of the methods, as they are presented here, are rather small. For the distance matrix and the greatest within-group distances altogether $n(n+1)/2$ locations for holding of

real numbers are needed. In addition, $4n+1$ locations for integers are necessary. From this all necessary information about the groups formed can be retrieved and the distance matrix remains unchanged during the execution of the procedures.

References

[1] L. Hubert, *Monotone invariant clustering procedures*, Psychometrika 38 (1973), p. 47-62.

[2] K. Jerzykiewicz and J. Szczepkowicz, *Algol 1204*, Warszawa 1973.

INSTITUTE OF COMPUTER SCIENCE
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 25. 5. 1976

J. KUCHARCZYK (Wrocław) ALGORYTMY 56-59

ANALIZA SKUPIEŃ METODAMI HUBERTA

STRESZCZENIE

Praca zawiera 4 algorytmy, za pomocą których można realizować metody analizy skupień podane przez Huberta w [1]. Dla kompletności podano także dwie procedury pomocnicze. Zasadniczy algorytm zawarty jest w procedurze *divmethod*, która wywołuje jedną z trzech procedur *divisionA*, *divisionB* i *divisionC*, opisujących trzy metody podane w [1].

Procedurę *divmethod* należy początkowo wywołać z $lg = 0$. Po wyjściu z procedury będzie $lg = 1$. Każde następne wywołanie procedury da na wyjściu wartość lg zwiększoną o 1, co dowodzi, że zostało utworzone nowe skupienie. Proces ten należy kontynuować do momentu, gdy $lg = n$. Każde wywołanie procedury z wartością $lg = k$ daje po wyjściu z procedury podział obiektów na $k+1$ grup.

Dane:

n — liczba obiektów;

$d[1 : n \times (n-1) \div 2]$ — dolna półmacierz macierzy odległości; nie zmienia się ona podczas obliczeń;

lg — liczba istniejących grup; początkowo należy wywołać procedurę *divmethod* z wartością $lg = 0$;

$lsg[0 : n]$ — tablica całkowita zawierająca w $lsg[i]$ $(i = 1, 2, \ldots, lg)$ skumulowane liczby obiektów w grupach od 1 do i $(lsg[0] = 0)$; liczba obiektów w grupie k wynosi zatem $lsg[k] - lsg[k-1]$ $(k = 1, 2, \ldots, lg)$;

$nrg[1 : n]$ — tablica całkowita zawierająca numery obiektów; numery obiektów należących do grupy i zawarte są w $nrg[lsg[i-1]$

$+1 : lsg[i]]$ $(i = 1, 2, \ldots, lg)$; numery te są uporządkowane w kolejności rosnącej wewnątrz każdej grupy;

diam $[1 : n]$ — tablica rzeczywista zawierająca największe odległości między obiektami każdej grupy; *diam*$[i]$ $(i = 1, 2, \ldots, lg)$ zawiera największą odległość między obiektami grupy i;

g1, g2 $[1 : n]$ — tablice całkowite zawierające numery obiektów, dla których odległość w każdej grupie jest największa; numery obiektów najbardziej odległych w grupie i zawarte są w *g1*$[i]$ oraz *g2*$[i]$ $(g1[i] < g2[i])$;

division — nazwa procedury realizującej wybraną metodę podziału (patrz dalej);

sort — nazwa procedury, która porządkuje część tablicy *nrg*;

diameter — funkcja rzeczywista, obliczająca największą odległość między obiektami należącymi do określonej grupy.

Wyniki:

lg — liczba wyznaczonych grup; jest ona większa o 1 od wartości *lg* na wejściu do procedury;

lsg, nrg, diam, g1, g2 — tablice zawierające informacje o utworzonych grupach (patrz odpowiednie opisy w danych).

Procedury *divisionA*, *divisionB* i *divisionC* mają identyczne wykazy parametrów.

Dane:

lp, lk — pierwszy i ostatni $(lp < lk)$ element *nrg*, wskazujący numery obiektów należących do grupy, która ma być podzielona na dwie grupy;

nr1, nr2 — numery obiektów $(nr1 < nr2)$ najbardziej odległych w tej grupie;

nrg $[1 : n]$ — tablica całkowita zawierająca numery obiektów;

d $[1 : n \times (n-1) \div 2]$ — dolna półmacierz macierzy odległości;

sort — nazwa procedury sortującej część tablicy *nrg*.

Wyniki:

nrg $[1 : n]$ — tablica całkowita zawierająca numery obiektów po utworzeniu nowych grup;

lp1, lk1, lp2, lk2 — liczby całkowite wskazujące dolne $(lp1, lp2)$ i górne $(lk1, lk2)$ granice numerów obiektów w *nrg*, odnoszące się do nowo utworzonych grup; numery te są potrzebne we wnętrzu procedury *divmethod*.

Funkcja rzeczywista *diameter* oblicza największą odległość między obiektami grupy, której numery zawarte są w *nrg* $[lp : lk]$.

Procedury te potrzebują stosunkowo niewiele pamięci operacyjnej. Macierz odległości i największe odległości wewnątrz grup wymagają zapamiętania $n(n+1)/2$ liczb rzeczywistych; ponadto wymagane jest pamiętanie $4n+1$ liczb całkowitych. Na tej podstawie można otrzymać wszystkie informacje dotyczące utworzonych grup i w razie potrzeby obliczyć zmienności wewnątrz grup i między grupami.