Les Cahiers de l'Analyse des Données
Vol. XX - 1995 - n°3 - pp. 321-358.

SUR L'INSTAURATION D'UN CODE : (2) PROGRAMME DE SIMULATION ET RÉSULTATS TYPIQUES

[INST. CODE (2)]

J.-P. BENZÉCRI

4 Programme de simulation

Le programme 'Icode', met en œuvre le modèle du §3, sans s'y conformer exactement. Écrit en langage PASCAL, 'Icode' se compose d'une suite de procédures appelées par un programme principal.

On expliquera d'abord, au §4.1, les procédures d'initialisation, qui mettent en place la description des ensembles d'objets et de signaux et la matrice des liens; (sans préciser, toutefois, l'utilisation des types et des pointeurs pour ranger les informations).

Ceci fait, on interprétera, au §4.2, le programme principal en termes de psychologie du comportement: comme mettant en jeu renforcement et généralisation; par des procédures dont l'effet sera suivi dans ses détails, au §4.3.

Le §4.4 présente une version du programme Icode qui offre à l'utilisateur de densifier les ensembles en correspondance - A, objets; et B, signaux: les bons résultats ainsi obtenus (cf *infra*, §5.4) attestent qu'on peut concevoir un processus d'instauration de code dont l'efficacité s'accroisse progressivement en abaissant les seuils de discrimination des objets et des signaux.

4.1 Procédures d'initialisation

4.1.1 Choix des données et des règles d'une partie: procédure 'choisir'

Le programme commence par la procédure 'choisir' qui permet à l'utilisateur, d'une part, de désigner, par son nom, le fichier décrivant l'ensemble A des objets et l'ensemble B des signaux, avec leur structure métrique; et, d'autre part, de fixer les modalités du jeu. Désormais, pour simplifier les notations, l'on attribue pour nom au fichier de description: Disq:tit.

Le format du fichier des ensembles A et B sera expliqué avec la lecture de celui-ci. Nous notons seulement ici que 'choisir' vérifie (par 'dialof')

Les cahiers de l'analyse des données 0339-3097/95/03/$ 4.00/© Gauthier-Villars

```
procedure choisir; begin repc:='N';er1:=0;
while not((repc='O') or (er1=6)) do begin
   write('le nom du fichier des ensembles B et A est ');readln(nomba);
   repc:=dialof(stringptr(@nomba));if not(repc='O') then er1:=er1+1;
   if (repc='O') then begin
     write('le nombre des joueurs sera (2 ou 3) ');readln(no);
     if (no<2) then no:=2;if (3<no) then no:=3;rps:='N';
     write('faut-il multiplier les renforcements par (1, 2, ou 3) ');
     readln(rfr);if (rfr<1) then rfr:=1;if (3<rfr) then rfr:=3;
     if (no=3) then begin
       write('les joueurs font-ils des séries par deux(D) ou non(N) ');
       readln(rps);end;
     write(nomba,';',no:2,' joueurs; renforcé',rfr:2);
     if not(rps='N') then write(';séries par Deux');writeln;
     if (no=2) then no:=1;
     write('ces choix sont-ils confirmés O ou N ');readln(repc) end;end;end;
```

l'existence du fichier; de telle manière que si le nombre des noms erronés, proposés pour le fichier, dépasse 5, le paramètre de confirmation, 'repc' reste sur 'N' (Non); et l'exécution du programme s'arrête.

Les modalités du jeu sont au nombre de trois.

Le nombre 'no' des joueurs ne peut être que 2 ou 3: on remarquera que, si no=2, la valeur numérique conservée est 1. Voici pourquoi: chaque joueur code et décode suivant une matrice de codage qu'il modifie, par une procédure de renforcement, en fonction des résultats de la communication. En l'état du programme, les paramètres de renforcement sont les mêmes pour tous les joueurs (ce qui exclut l'étude de l'effet de dominance qui pourrait résulter de ce qu'un sujet modifie peu son code...): il en résulte que, s'il n'y a que deux joueurs, ceux-ci, participant aux mêmes essais et en tirant les mêmes conclusions, ont toujours le même code; il n'y a donc qu'une seule matrice à garder en mémoire. Avec trois joueurs, au contraire, on a trois matrices distinctes; lesquelles, toutefois, dans nos expériences, convergent au cours du jeu, pour définir un comportement uniforme des sujets.

Un coefficient 'rfr' est introduit dans la formule de renforcement: le but est d'apprécier dans quelle mesure des renforcements de grande amplitude nuisent à la découverte d'un code efficace et à la convergence des comportements. Nous estimions, *a priori*, que rfr=3 compromettrait l'instauration du code: rien de tel n'est apparu dans nos essais.

L'éventualité rps≠'N' ne concerne que les expériences avec trois joueurs. Il semble que, dans la vie réelle, il soit plus facile de parvenir à se comprendre entre deux qu'entre plusieurs. Dans la simulation, le cas de deux joueurs offre la particularité supplémentaire que ceux-ci se trouvent toujours avoir le même comportement; se pose seulement la question de l'efficacité à laquelle peut conduire un processus local, ne prenant pas explicitement en compte la forme globale des ensembles A et B. Avec trois sujets, on peut attendre qu'une

longue série de coups entre deux des joueurs suffise à instaurer un code cohérent; lequel sera ensuite imposé au troisième lorsque celui-ci jouera une série avec l'un des deux premiers.

Le programme offre donc le choix entre deux règles pour les rencontres entre les trois joueurs: ou bien, pour chaque essai de communication, tirer au sort, entre 3, les numéros, nn et np, du sujet qui formule la demande et de celui qui la décode; ou bien procéder par séries: une série comprenant 10*cardA essais, (i.e. dix fois autant qu'il y a d'objets à désigner,) joués entre deux mêmes sujets {s1, s2}; lesquels prennent, alternativement, les rôles de nn et np. Il est apparu qu'un code s'instaure bien sans faire de séries par deux; mais l'analogie géométrique entre espace des objets et espaces des signaux est mieux comprise dans le code, s'il y a des séries par deux.

4.1.2 Espace des objets et espace des signaux

4.1.2.1 La procédure 'grapher'

```
procedure grapher; begin
carac:=' ';lecnombre;carj:=ffl;readln(ftu,linea);
if (carj<3) or (250<carj) then repc:='N';
if (repc='O') then begin pzg:=pzgi(newptr(6*carj));
  for j:=1 to carj do if (repc='O') then begin
    lecsig;pzg^[j]:=sig;
    readln(ftu,linea);id:=1;iz:=length(linea);
    while (linea[id]=' ') and (id<iz) do id:=id+1;
    iz:=iz+1-id;linea:=copy(linea,id,iz);id:=1;
    while not((linea[id]=' ') or (id=iz)) do id:=id+1;
    iz:=id;if (linea[id]=' ') then iz:=iz-1;
    linea:=copy(linea,1,iz);
    gr[j]:=stringptr(newptr(256));gr[j]^:=linea;
    if not(iz=carj) then begin repc:='N';
      for jp:=1 to j do dispose(gr[jp]);dispose(pzg);
      if (blc='a') then begin dispose(pzgb);
        for ib:=1 to cardB do dispose(grb[ib]);end;
      write('ERREUR ',sigler(sig),' ',linea);readln(rpv);
end;end;end;end;
```

La procedure 'grapher' a pour objet la lecture du fichier de description des ensembles A et B, 'Disq:tit'. Nous expliquerons d'abord la structure de ce fichier. En bref, après une ligne de titre, (dans notre cas: 'communication',) sont présentés successivement, sous le même format, l'ensemble B des signaux et l'ensemble A des objets.

On imaginerait que chaque ensemble soit décrit, d'emblée, par un tableau carré symétrique de distances; mais, ici, notamment afin d'éviter des erreurs de calcul, on part d'une simple matrice de contiguïté décrivant un graphe: avec k(i, i') = 1 si et seulement si i est contigu à i'; un calcul de distance sur le graphe est effectué ensuite par la procédure 'normer'.

Ici, les matrices de description des graphes et les matrices de distances sont présentées face à face.

```
Disq:tit  communication
20 signaux
b01 01==================
b02 101=================
b03 =101================
b04 ==101===============
b05 ===101==============
b06 ====101=============
b07 =====101============
b08 ======101===========
b09 =======101==========
b10 ========101=========
b11 =========101========
b12 ==========101=======
b13 ===========101======
b14 ============101=====
b15 =============101====
b16 ==============101===
b17 ===============101==
b18 ================101=
b19 =================101
b20 ==================10

10 objets
a01 01=======1
a02 101======
a03 =101=====
a04 ==101====
a05 ===101===
a06 ====101==
a07 =====101=
a08 ======101
a09 =======10
a10 1=======0
```

```
Disq:titΔis  communication
20 signaux
b01 0123456789==========
b02 10123456789=========
b03 210123456789========
b04 3210123456789=======
b05 43210123456789======
b06 543210123456789=====
b07 6543210123456789====
b08 76543210123456789===
b09 876543210123456789==
b10 9876543210123456789=
b11 =9876543210123456789
b12 ==987654321012345678
b13 ===98765432101234567
b14 ====9876543210123456
b15 =====987654321012345
b16 ======98765432101234
b17 =======9876543210123
b18 ========987654321012
b19 =========98765432101
b20 ==========9876543210

10 objets
a01 0123454321
a02 1012345432
a03 2101234543
a04 3210123454
a05 4321012345
a06 5432101234
a07 4543210123
a08 3454321012
a09 2345432101
a10 1234543210
```

À gauche : Matrices de contiguïté ; *À droite* : Matrices de distance

Une matrice de contiguïté (comme, ensuite la matrice de distance) est donnée et comprise comme un texte. Après une ligne, où la seule information relevante est le nombre des éléments de l'ensemble, (qui doit seulement être suivi d'un blanc et d'au moins un signe quelconque,) viennent une suite de lignes, chacune afférente à un élément; l'interposition de lignes blanches ne nuisant pas à la lecture.

La ligne commence par le sigle de l'élément j ($j \in J$; J = A ou B); puis vient, après au moins un blanc, un segment de texte comptant exactement cardJ caractères (non blancs): où seuls sont interprétés les chiffres '1', expression de la contiguïté: e.g., si le 7-ème caractère du texte de la 4-ème ligne est '1', le 4-ème et le 7-ème élément sont contigus. L'intégration des données de contiguïté est laissée à la procédure 'normer': 'grapher' vérifie seulement qu'existent cardJ lignes donnant, après le sigle, un segment de texte ayant pour longueur cardJ. Toute irrégularité commande l'arrêt, avec libération (par 'dispose') des zones de mémoire pointées et envoi d'un commentaire d'erreur.

4.1.2.2 La procédure 'normer'

```
procedure normer; begin
if (repc='0') then begin
  for j:=1 to carj do begin
    for jp:=1 to carj do begin
      if (gr[j]^[jp]='1') then gr[jp]^[j]:='1';
      if not (gr[j]^[jp] in ['0','1']) then gr[j]^[jp]:='=';end;
    gr[j]^[j]:='0';end;
  for iz:=1 to 8 do begin rpv:=chr(48+iz);rpw:=chr(49+iz);
    for j:=1 to carj-1 do for jp:=1 to carj do
      if (gr[j]^[jp]='1') then for js:=j+1 to carj do
         if ((gr[jp]^[js]=rpv) and (gr[j]^[js]='=')) then begin
            gr[j]^[js]:=rpw;gr[js]^[j]:=rpw;end;end;end;end;
```

La procédure 'normer' met en forme cohérente les informations de contiguïté. D'abord, la matrice lue est symétrisée: i.e. si on a lu k(i, i')=1, on impose aussi k(i', i)=1; les termes diagonaux k(i, i) sont mis à 0; dans les autres places, on met uniformément le caractère '='. Pour plus de clarté, nous avions déjà fait cette régularisation dans le fichier de description 'Disq:tit', publié ici.

Ensuite, on pose k(i, i') = n, si existe, de i à i', une suite de n+1 éléments dont chaque élément intermédiaire est, sur le graphe, contigu au précédent et au suivant; et qu'il n'y en a pas de plus courte. On notera, d'une part, que le calcul est fait en format de texte: ainsi, chr(49+iz) signifie: le chiffre représentant le nombre iz+1; et, d'autre part, que, sous ce format, on a limité le calcul itératif à la distance maxima n=9.

Une procédure 'dister', non publiée ici, crée un fichier Disq:titΔis contenant les matrices de distance (calculées par 'normer' pour les deux ensembles). On voit que, dans le cas considéré ici, l'ensemble A comprend dix objets, de a01 à a10, liés en un cycle; tandis que B est un segment, ou suite de 20 signaux, de b01 à b20.

4.1.3 La matrice des liens: création en mémoire vive et édition sur fichier

4.1.3.1 La procédure 'tabler'

```
procedure tabler; begin
if (repc='0') then begin
  for j:=1 to cardA do begin pra[j]:=1;
    xab[j]:=pti(newptr(no*4*cardB));for ib:=1 to no*cardB do xab[j]^[ib]:=1;
end;end;end;
```

D'une part, la procédure 'tabler' met à 1 les cases du tableau `pra`; ce qui assigne à tous les objets de A une égale probabilité d'être demandés dans le jeu de communication.

D'autre part, 'tabler' crée un tableau `xab` dont chacune des cardA colonnes contient no*cardB nombres réels. S'il n'y a que deux joueurs, on pose, cf. *supra*, no = 1; il y a une seule matrice de codage; et celle-ci occupe le tableau `xab`.

Avec trois joueurs, no=3, il a trois matrices de codage distinctes (qui toutefois, selon ce que nous avons remarqué, tendent à converger): ces matrices sont superposées dans xab. Ainsi, les sujets étant numérotés de n=0 à n=2, xab[ia]^[((n)*cardB)+ib], donne, pour le sujet n, le coefficient d'association entre l'objet ia et le signal ib. Initialement toutes les cases du tableau xab sont à 1; elles sont modifiées, au cours du jeu, par la procédure 'renforcer' qui appelle 'diffuser' (cf. *infra*, §4.3.2).

4.1.3.2 La procédure 'ouvcoder'

```
procedure ouvcoder;begin
  nomf:=concat(nomba,'code');
  if (no=3) then nomf:=concat(nomf,'n3');
  if not(rps='N') then nomf:=concat(nomf,'S');
  if not(rfr=1) then nomf:=concat(nomf,'r',chr(48+rfr));
  rewrite(ftq,nomf);end;
```

La procédure 'ouvcoder' ouvre un fichier de texte destiné à recevoir les états successifs des matrices de codage.

De façon précise, le processus de jeu est découpé en cycles, ou séries, comprenant chacun 10*cardA essais; au cours d'une partie, les séries se poursuivent jusqu'à ce que l'utilisateur interrompe le jeu; une procédure 'critaber', non publiée ici, inscrit dans le fichier l'état des matrices de codage (dites encore: matrice des liens), à la fin de la 1-ère série; puis toute les 10 séries; et à la fin de la partie.

On voit que 'ouvcoder' donne au fichier le nom de 'disq:titcode', avec, éventuellement, un suffixe pour signaler qu'il y a trois joueurs, que chaque série n'intéresse que deux joueurs, qu'on a introduit un coefficient de renforcement; c'est-à-dire, en bref, pour enregistrer les réponses que l'utilisateur a données aux questions posées par la procédure 'choisir'.

4.1.3.3 La procédure 'initier'

```
procedure initier;begin
writeln('Il est possible d''initialiser le code des sujets');
for nc:=1 to no do begin
 write('faut-il lire un tableau donné pour le code du sujet',nc:2,' O ou N ');
 readln(rpi);if not(rpi='N') then rpi:='O';noc:=nc-1;
 if (rpi='O') then begin
  litab(s2j,@t1k,@nom,@tit,rpi,carti,cartj,s1i);
  if (rpi='O') and not((cartj=carda) and (carti=cardb)) then rpi:='P';
  if (rpi='O') then begin for j:=1 to carda do yab[j]:=pti(t1k[j]);
    for j:=1 to carda do for ib:=1 to cardb do
      xab[j]^[(noc*cardb)+ib]:=yab[j]^[ib];
    writeln(ftq,'le code du sujet',nc:2,' est initialisé par ',nom);end;
  if (rpi in ['O','P']) then begin
    for j:=0 to cartj do dispose(t1k[j]);dispose(s1i);dispose(s2j);end;
end;end;end;
```

La procédure 'initier' offre à l'utilisateur d'imposer à chacun des sujets un code initial déterminé, lu dans un tableau, au format usuel du logiciel

MacSAIF. La procédure se borne à vérifier les dimensions de ce tableau; lequel, après lecture dans yab, se substitue, dans xab, au bloc rempli de chiffres 1, afférent au sujet considéré. La référence au fichier du code initial, 'disq:codinit', est alors inscrite en tête du fichier 'disq:titcode' (créé par 'ouvcoder').

Dans la pratique, les codes imposés peuvent être des schémas de modèles simples, entre lesquels le processus simulera un conflit; ou des états de code obtenus par simulation, dans une partie antérieure; ce qui, à la suppression près des liens négatifs, permet de reprendre un jeu interrompu. On notera qu'il n'y a pas de risque majeur à introduire une matrice de code absurde, nulle, ou encombrée de liens négatifs: car la fonction 'tirer' (cf. *infra*, §4.3.1.1) pourvoit, si nécessaire, à transformer la suite de nombres qui lui est transmise en une loi de probabilité acceptable.

4.2 Le programme principal 'Icode'

Les procédures que nous achevons de décrire effectuent toutes les opérations préalables au jeu proprement dit; elles sont appelées dans les premières instructions du programme principal. Nous pouvons maintenant considérer ce programme dans son ensemble.

4.2.1 La boucle générale des parties

L'instruction initiale 'while not (rpr='N')' place l'ensemble du programme principal, initialisation et jeu, dans une boucle globale où chaque passage constitue une partie, dont les spécifications sont fixées par 'choisir'. Une partie se termine quand l'utilisateur répond 'N' (Non) à la question:

```
faut-il poursuivre le tirage O ou N ;
```

la procédure 'disposer' libère alors la mémoire attribuée dynamiquement (pour loger la description des ensembles et le code); et offre à l'utilisateur l'occasion de sortir de la boucle globale, en répondant 'N' à la question:

```
faut-il reprendre avec d'autres données O ou N .
```

4.2.2 L'appel des procédures d'initialisation

Ainsi qu'on l'a dit ci-dessus, si existe le fichier de description, 'Disq:tit', demandé dans la prodédure 'choisir', la réponse de confirmation 'repc' peut être mise à 'O' (Oui).

En lisant 'Disq:tit', on enregistre alors, successivement, chacun des ensembles A et B par les procédures 'grapher' et 'normer'. À la réserve près que toute irrégularité de format interrompt cet enregistrement; et met 'repc' à 'N' (Non); ce qui termine l'exécution du programme.

Si le fichier 'Disq:tit' est accepté, 'dister' crée le fichier 'Disq:titΔis' des matrices de distances; 'tabler' met en place le tableau xab des liens entre

```
begin rpr:='O';
while not (rpr='N') do begin
 writeln('ce programme joue suivant une loi');repc:='N';rpr:='N';
 choisir;
 if (repc='O') then begin
   reset(ftu,nomba);
   readln(ftu,titre);
   blc:='b';grapher;normer;
   cardB:=carj;pzgb:=pzg;for j:=1 to 250 do grb[j]:=gr[j];
   blc:='a';grapher;normer;
   cardA:=carj;pzga:=pzg;for j:=1 to 250 do gra[j]:=gr[j];
   close(ftu);end;
 if (repc='O') then begin
   dister;tabler;ouvcoder;initier;cy:=0;end;

 while not(repc='N') do begin
  quijoue;cy:=cy+1;
  if (cy=1) and (no=3) then begin nn:=0;np:=1;end;
  for im:=1 to 10*cardA do begin
   j:=tirer(cardA,@pra[1]);
   if (rps='N') then quijoue else begin ibb:=nn;nn:=np;np:=ibb;end;
   ib:=tirer(cardB,@xab[j]^[(nn*cardB)+1]);ibb:=(np*cardB)+ib;
   for ia:=1 to cardA do prpa[ia]:=xab[ia]^[ibb];
   jp:=tirer(cardA,@prpa[1]);renforcer;end;
  if (cy mod 10=1) then critaber;
  write('faut-il poursuivre le tirage O ou N ');readln(repc);
  if (repc='N') then disposer;end;end;end.
```

objets et signaux; 'ouvcoder' prépare l'inscription répétée de ce tableau sur 'Disq:titcode': la partie de jeu proprement dite peut commencer.

4.2.3 Structure des cycles de jeu

Une partie est une boucle 'while not(repc='N')', dont chaque passage, repéré par l'indice 'cy', produit un cycle, ou série, de 10*cardA essais de communication. On sait qu'à chaque essai participent, en principe, deux sujets: celui qui formule la demande et celui qui la décode; dont les numéros respectifs sont nn et np. Plus précisément, s'il y a, en tout, trois joueurs, no=3, nn et np sont deux nombres différents, pris de 0 à 2:

$$nn \in \{0, 1, 2\} \ ; \ np \in \{0, 1, 2\} \ ; \ nn \neq np.$$

Dans le cas no=3, rps='N', nn et np sont tirés à chaque essai; avec rps≠'N', nn et np alternent seulement, au cours d'une série, par le jeu des instructions: ibb:=nn;nn:=np;np:=ibb. On notera que le premier essai a toujours lieu entre les joueurs nn=0 et np=1: de cette façon, si, par 'initier', on a imposé, pour code initial, à chacun de ces deux joueurs, des matrices déterminées, on est assuré (avec rps='O') que la partie commence par une première série qui montre un conflit entre les deux codes choisis.

Quand on dit qu'il y a deux joueurs, il n'y a, en fait, qu'une seule matrice de codage et décodage:

no=1; nn=np=0.

Pour chaque essai, un objet j de A est tiré suivant la loi uniforme 'pra' (cf. *supra* 'tabler'). On détermine s'il y a lieu, nn et np. La matrice de liens du demandeur nn sert à choisir, aléatoirement, le signal ib dans B; ce signal est décodé comme désignant un objet jp, d'après la matrice du sujet np. En conclusion les matrices de liens des sujets nn et np sont renforcées; c'est-à-dire modifiées en fonction du succès de l'essai, d'autant meilleur qu'est plus faible la distance (j, jp) entre objet demandé et objet répondu.

Enfin, dans le fichier 'disq:titcode', (ouvert par la procédure 'ouvcoder'; cf. *supra*, §4.1.3.2,) s'inscrit, tous les dix cycles, par la procédure 'critaber', un état des matrices de codage: tel est bien l'effet de l'instruction:

```
if (cy mod 10=1) then critaber;
```

placée en fin de cycle.

Reste à voir comment s'effectuent les tirages; et quelles sont les règles de renforcement.

4.3 Les procédures de jeu

4.3.1 Tirage des partenaires, des objets et des signaux

4.3.1.1 La fonction 'tirer'

```
function tirer(caru:integer;ploi:ptr):integer;
  var loi:pti;totu,brnu:single;
  begin loi:=pti(ploi);totu:=0;
  for iu:=1 to caru do if loi^[iu]<0 then loi^[iu]:=0;
  for iu:=1 to caru do totu:=totu+loi^[iu];
  if (totu=0) then begin totu:=caru;
    for iu:=1 to caru do loi^[iu]:=1;end;
  brnu:=randoner;brnu:=(brnu+32768)*totu/65535;totu:=0;iu:=0;
  while not((iu=caru) or (brnu<totu)) do begin
    iu:=iu+1;totu:=totu+loi^[iu];end;
  if (iu=0) then iu:=1;tirer:=iu;end;
```

La fonction 'tirer' a pour arguments un entier 'caru' et un pointeur 'ploi'. L'entier caru est le cardinal de l'ensemble U, où s'effectue le tirage; U est assimilé à la suite d'entiers {1, 2, ..., caru}. En typant le pointeur 'ploi' on accède à une suite 'loi' de caru nombres réels; on pourvoit à ce qu'aucun de ceux-ci ne soit négatif et à ce que leur somme, totu, soit strictement positive; de telle sorte qu'on ait toujours une loi de probabilité acceptable.

Le tirage proprement dit passe par le choix d'un entier 'brnu', uniformément distribué entre -32768 et +32767; la fonction 'randoner' qui renvoie brnu, ne diffère de la fonction 'random' fournie par le constructeur qu'en ce que l'on y utilise l'état de l'horloge interne afin que plusieurs exécutions du programme sur les mêmes donnés n'aient aucune chance de produire la même suite d'essais (ce qui serait le cas avec 'random', prise telle quelle).

Finalement, 'tirer' renvoie un entier iu, distribué de 1 à caru suivant la loi spécifiée par le pointeur ploi.

Selon les cas, dans le cours d'une partie, 'tirer' renvoie un numéro de sujet; le numéro d'un objet à coder; le signal qui codera cet objet; ou l'objet conjecturé par décodage du signal. Pour les sujets, il s'agit toujours de tirer suivant une loi uniforme, entre 2 ou entre 3 possibles. Pour l'objet à coder, on a déjà dit que les cardA objets sont équiprobables.

Quant au codage et au décodage, ne prétendant pas imposer au lecteur de déchiffrer, sans aide, le programme principal, nous dirons qu'on utilise, respectivement, une colonne ou une ligne de la matrice des liens: une colonne est un système de cardB nombres réels, donc une loi sur B; de même, une ligne, avec cardA nombres, fournit une loi sur A.

4.3.1.2 La procédure 'quijoue'

```
procedure quijoue;begin
  if (no=1) then begin nn:=0;np:=0;end;
  if (no=3) then begin
     nn:=tirer(no,@pra[1]);np:=tirer(no-1,@pra[1]);
     nn:=nn-1;np:=(nn+np) mod no;end;end;
```

Nous donnons, pour mémoire, la brève procédure 'quijoue'; laquelle, en appelant 'tirer', distribue les rôles selon des règles déjà expliquées plus haut. Si no=3 et rps='N', nn est tiré dans {0, 1, 2}; puis np, entre les deux sujets restant dans $\{0, 1, 2\} - \{nn\}$.

4.3.2 Renforcement et généralisation

Les deux procédures suivantes, 'diffuser' et 'renforcer', celle-ci appelant celle-là, modifient les matrices de liens, en conclusion à un essai. Après une communication réussie le poids relatif des liens que celle-ci a mis en jeu est renforcé. De plus les appels de 'renforcer' à 'diffuser' définissent une généralisation de l'objet; tandis que 'diffuser' procède, en outre, à une généralisation du signal. Renforcements et généralisations pourraient se faire de bien des manières; et, dans le présent algorithme lui-même, le choix des seuils et coefficients ne s'impose pas. Sans calcul théorique, nous expliquerons les effets cherchés et les ordres de grandeur adoptés, *a priori*, ou d'après des résultats.

4.3.2.1 La procédure 'diffuser'

Après chaque essai, la procédure 'diffuser' modifie la matrice des liens, en fonction de multiples appels reçus de la procédure 'renforcer', expliquée plus bas. Selon les cas, le paramètre d'appel, ii, est positif ou négatif. L'amplitude des modifications apportées est donnée par 'res'.

Il est naturel que le coefficient de renforcement 'rfr' (cf. *supra* 'choisir') soit l'un des facteurs de res. Le signe de res est celui du paramètre d'appel ii,

```
procedure diffuser(ii:integer);begin res:=rfr*cardB/ii;
xab[js]^[(nn*cardB)+ib]:=xab[js]^[(nn*cardB)+ib]+res;
if not(no=1) then xab[js]^[(np*cardB)+ib]:=xab[js]^[(np*cardB)+ib]+res;
res:=res/2;
for iu:=1 to cardB do if (grb[ib]^[iu]='1') then begin
   xab[js]^[(nn*cardB)+iu]:=xab[js]^[(nn*cardB)+iu]+res;
   if not(no=1) then xab[js]^[(np*cardB)+iu]:=xab[js]^[(np*cardB)+iu]+res;end;
end;
```

ce qui signifie qu'il y a renforcement proprement dit si ii>0; et, sinon, effacement de liens. De plus, ii étant en dénominateur, l'effet est d'autant plus faible que liil est plus élevée; en l'état présent du programme, les seuls appels prévus dans 'renforcer' sont avec ii ∈ {-3, -2, 1, 2}.

C'est en introduisant le facteur cardB qu'on fixe proprement l'ordre de grandeur de 'res': initialement, dans une matrice de lien, la colonne afférente à un objet js contient cardB nombres, tous égaux à 1; ajouter, au premier essai, cardB à l'un de ces nombres revient donc à donner à celui-ci un poids supérieur au total de tous les autres.

Dans la suite des renforcements, la matrice des liens n'est jamais remise à l'échelle; les renforcements s'y accumulent, sans risque de débordement, compte tenu de ce qu'un réel peut aller jusqu'à 3e38... L'accumulation des renforcements a pour conséquence que décroît, au cours du temps, l'importance relative du renforcement qui suit un essai individuel; ce qui favorise la convergence du code.

C'est seulement dans le fichier 'disq:titcode' que la procédure 'critaber' écrit des matrices de liens normalisées pour avoir un terme maximum égal à 1000. Quant aux nombres négatifs, 'critaber' les met à zéro; mais ils sont conservés dans la matrice 'xab' sans compromettre les tirages, puisque la procédure 'tirer' traite comme des zéros les nombres négatifs qui lui sont transmis dans une loi.

On voit que les cases modifiées par 'diffuser' sont déterminées relativement à ib, numéro du signal effectivement utilisé dans l'essai, et à js, numéro d'objet fixé par 'renforcer' avant l'appel de 'diffuser'.

La variation maxima, égale à res, affecte le lien entre ib et js; une variation moitié, res/2, est apportée aux liens entre js et tout signal iu qui, sur le graphe de B, est contigu à ib. Ainsi, le renforcement, qui concerne au premier chef le lien (ib, js), s'étend, comme par diffusion, aux signaux voisins de ib: de là vient le nom de la procédure.

À l'usage, le terme res/2 est apparu trop fort quand cardB est faible relativement à cardA: on a donc adopté l'instruction suivante (avec succès, cf. *infra* §5.2.2):

```
res:=res/2;if (cardB<2*cardA) then res:=res*cardB/(2*cardA);
```

où le rapport (cardB/(2*cardA)) a été introduit parce que, en passant de {cardA=10, cardB=20} à {cardA=10, cardB=15}, on avait constaté une détérioration du code limite. Sans s'arrêter à cette formule, on conçoit qu'une forte généralisation du signal ait pour effet de rendre solidaires des blocs de signaux voisins ce qui diminue la capacité du code à transmettre l'information. On verra au §4.4 que la généralisation sur A et B peut être diminuée au cours du processus d'instauration de code.

Si no=1, on a nn=np=0, il n'y a qu'une seule matrice de liens à modifier; au contraire, si no=3, on apporte des modification identiques aux matrices de liens afférentes aux deux joueurs nn et np. Les instructions d'affectation, utilisées pour cela, se comprennent si l'on se souvient de la manière dont les matrices de liens sont rangées dans un unique tableau 'xab', créé par la procédure 'tabler' (cf. *supra*).

4.3.2.2 La procédure 'renforcer'

```
procedure renforcer;begin
if (j=jp) then
  for js:=1 to cardA do begin rpv:=gra[j]^[js];
    if (rpv='0') then diffuser(1);
    if (rpv='1') then diffuser(2);
    if not(rpv in ['0','1','2']) then diffuser(-2);end;
if (gra[j]^[jp]='1') then
  for js:=1 to cardA do begin rpv:=gra[j]^[js];rpw:=gra[jp]^[js];
    if ((rpv='1') or (rpw='1')) then diffuser(3);
    if not((rpv in ['0','1','2']) or (rpw in ['0','1','2']))
      then diffuser(-3);end;
end;
```

Ainsi qu'on l'a dit, la procédure 'renforcer' régit les appels à la procédure 'diffuser': un appel comporte un paramètre d'intensité (noté ci-dessus: ii) et affecte fondamentalement le couple (ib, js), avec diffusion autour de ib. Afin de simplifier l'exposé, nous ferons abstraction de cette diffusion; et ferons comme si le lien (ib, js) était seul affecté.

Nous considérerons les deux éventualités distinguées par 'renforcer'.

Si (j=jp) le succès de la communication a été parfait: par diffuser(1) un renforcement positif maximum affecte le lien de ib à j=jp (cas rpv='0'); un renforcement moindre affecte les liens entre ib et tout signal js contigu à j=jp.

Au contraire, pour un objet js dont la distance à j=jp dépasse le seuil ds=2 (not(rpv in ['0','1','2'])), le lien avec ib est affaibli. On réduit ainsi la probabilité que, dans un essai ultérieur, le signal ib (déjà fortement lié à j) désigne ou évoque un objet js éloigné de j=jp; ce qui contituerait une sorte de polysémie de ib. (Certes, la valeur même du seuil, ds=2, ne s'impose pas...).

Si j et jp sont contigus sur le graphe A des objets, on considère que, le succès de la communication ayant été mitigé, les renforcements doivent être

moins prononcés que si j=jp. On accroît faiblement les liens entre ib et un objet contigu à j ou à jp; on réduit faiblement les liens entre ib et un objet éloigné, à la fois, de j et de jp.

On doit répéter ici ce qu'on a dit avant tout commentaire, que le choix des seuils ne s'impose pas.

Voilà comment les appels de 'renforcer' à 'diffuser' comportent une généralisation de l'objet; 'diffuser', procèdant, en outre, à une généralisation du signal.

4.4 Densification des ensembles A et B

Les graphes A et B représentent chacuns, dans le modèle, un espace dont il est le schéma. On peut concevoir ce schéma comme la décomposition simpliciale associée à un recouvrement par des zones dont le rayon est égal au seuil de discrimination choisi: à chaque zone - intervalle, disque, boule, selon la dimension -, correspond un point, ou sommet, du graphe; deux points étant dits contigus, ou reliés par une arête, si les zones correspondantes ont une intersection non vide. Diminuer les seuils (cf. §3.1.1), c'est subdiviser les zones, donc densifier le graphe.

Nous ne tenterons pas, en toute généralité, d'associer à un espace donné, A ou B, une suite de schémas de plus en plus fins; la version modifiée de Icode, présentée ci-dessous, ne prend pas directement en compte l'espace, mais permet de densifier un graphe. La procédure proposée est acceptable dans le cas unidimensionnel, il serait facile de la généraliser à une triangulation quelconque.

En bref, partant d'un graphe, J, ayant carJ sommets, avec, entre ceux-ci, des relations de contiguïté; on crée un graphe Jd, dont les cardJ sommets sont, d'une part, les carJ sommets de J et, d'autre part, des sommets nouveaux, jd, dont chacun peut être considéré comme le milieu d'une arête, j.j', de J. Alors que dans J les points j et j' sont contigus (|jj'|=1); dans Jd, j et j' ne sont pas contigus entre eux, mais le sont tous deux à jd: |jj'|=2 ; |jjd|=|j'jd|=1. Le graphe Jd sera nommé: le densifié de J.

Les seules additions importantes à Icode concernent les procédures 'grapher' et 'initier': il faut, d'une part, densifier les graphes A et B lus dans 'disq:tit'; d'autre part, étendre en une correspondance entre les graphes densifiés, Ad et Bd, une matrice initiale de liens, 'disq:codinit', donnée entre A et B. Les autres procédures, de l'initialisation ou du jeu, opèrent sur Ad et Bd comme elles le feraient sur A et B. On introduit seulement le caractère '∂' dans le nom des fichiers créés: 'disq:tit∂Δis', pour les graphes de distance; 'disq:tit∂code', pour les tableaux d'états du code.

4.4.1 L'option 'densifier' dans 'choisir'

```
write('faut-il densifier les graphes A et B oui(O) ou non(N) ');readln(rp2);
```

Une instruction suffit pour offrir à l'utilisateur de conduire le processus d'instauration de code non sur les graphes décrits dans le fichier 'disq:tit', mais sur des graphes obtenus par densification de ceux-ci.

4.4.2 Complément à la procédure 'grapher'

```
if ((repc='O') and (rp2='O')) then begin
  for j:=1 to carj-1 do for jp:=j+1 to carj do
    if ((cardj<250) and ((gr[j]^[jp]='1') or (gr[jp]^[j]='1'))) then begin
    cardj:=cardj+1;gr[j]^[jp]:='=';gr[jp]^[j]:='=';
    gr[cardj]:=stringptr(newptr(256));
    for js:=1 to carj do gr[cardj]^[js]:='=';
    gr[cardj]^[jp]:='1';gr[cardj]^[j]:='1';
    sig[0]:=3;sig[1]:=ord(blc);
    sig[2]:=65+((j-1) mod 26);sig[3]:=65+((jp-1) mod 26);
    pzg^[cardj]:=sig;end;
  for j:=1 to cardj do begin
    gr[j]^[0]:=chr(cardj);for js:=carj+1 to cardj do gr[j]^[js]:='=';end;end;
```

Un graphe à carj sommets a été lu: en cherchant par une double boucle les arêtes de ce graphe, on crée les nouveaux sommets, numérotés de carj+1 à cardj; (chaque nouveau sommet étant créé avec pour numéro la nouvelle valeur, augmentée de 1, qu'il donne à cardj). Initialement, la matrice d'incidence est un bloc carré, de carj lignes de longueur carj; en créant un sommet nouveau, cardj≈{j, jp}, on supprime dans les lignes j et jp leur lien mutuel en posant:

```
gr[j]^[jp]:='=';gr[jp]^[j]:='=';
```

et on crée une ligne, gr[cardj]^, formée de carj signes '='; à ceci près qu'on y note la contiguïté avec j et jp:

```
gr[cardj]^[jp]:='1';gr[cardj]^[j]:='1';
```

le tableau de cardj lignes de longueur carj est ensuite complété en un carré cardj × cardj par des signes '='.

On notera qu'il a fallu donner un sigle à chaque sommet créé. Pour l'expérience qu'on avait en vue (cf. §5.4), il a suffi d'un sigle de trois caractères: le premier est, en minuscule, la lettre 'blc', caractéristique de l'ensemble; les deux suivants représentent par des capitales le rang des extrémités j et jp de l'arête. Ainsi, dans le cercle densifié Ad, le sommet interposé entre a04 et a05 est noté aDE; et dans le segment Bd, bRS se place entre b18 et b19.

4.4.3 Complément à la procédure 'initier'

Pour suivre cette procédure il faut prendre garde à la distinction entre carA, nombre des sommets du graphe lu, A; et cardA, nombre des sommets du graphe densifié, Ad; et de même carB et cardB pour B et Bd.

La procédure 'tabler', a créé un tableau de liens xab, comportant, pour chaque sujet, un bloc cardB × cardA, où tous les liens sont mis à 1.

```
if ((rpi='0') and (rp2='0')) then begin
  for j:=1 to carA do for ib:=carB+1 to cardB do xab[j]^[(noc*cardB)+ib]:=0;
  for j:=carA+1 to cardA do for ib:=1 to cardB do xab[j]^[(noc*cardB)+ib]:=0;
  for ib:=carB+1 to cardB do for iz:=1 to carB do if (grb[ib]^[iz]='1') then
      for j:=1 to carA do xab[j]^[(noc*cardB)+ib]:=
            xab[j]^[(noc*cardB)+ib]+(0.5*xab[j]^[(noc*cardB)+iz]);
  for j:=carA+1 to cardA do for js:=1 to carA do if (gra[j]^[js]='1') then
      for ib:=1 to cardB do xab[j]^[(noc*cardB)+ib]:=
            xab[j]^[(noc*cardB)+ib]+(0.5*xab[js]^[(noc*cardB)+ib]);end;
```

Si, pour un sujet numéroté 'noc', l'utilisateur a demandé d'initialiser le code par 'disq:codinit', le sous-bloc carB × carA du bloc cardB × cardA de xab, afférent à ce sujet, a été rempli des liens - entre sommets initiaux - lus dans 'disq:codinit'.

Par les deux premières boucles 'for' du listage ci-dessus, on commence par mettre à zéro le reste du bloc cardB × cardA. Ensuite pour ces couples, où interviennent un ou deux des sommets créés par densification, on calcule, par moyenne, de nouveaux liens: c'est un sorte d'interpolation.

Dans la boucle 'for ib:=carB+1 to cardB', on attribue à chaque sommet - signal - ib de Bd, créé, par densification entre deux sommets {iz, izp} de B, une ligne de liens avec A, qui est la moyenne des lignes de liens lus pour iz et izp.

De même, dans la boucle 'for j:=carA+1 to cardA', on attribue à chaque sommet j - objet - de Ad, créé, par densification entre deux sommets {js, jsp} de A, une colonne de liens avec Bd, qui est la moyenne des colonnes de liens, lus ou créés pour js et jsp.

Par une telle initialisation, les sommets insérés prennent des positions moyennes entre celles de sommets préexistants: la précision du code n'est pas améliorée. Chaque nouveau signal b est associé à un domaine de l'espace des objets, densifié certes, mais de même largeur que ceux afférents aux signaux préexistants. Il s'agit à peu près, des mêmes domaines de l'espace; mais avec davantage de points, du fait de l'interpolation. Reste à voir dans quelle mesure, par le jeu des renforcements ce code initial se spécialisera pour gagner en précision: c'est ce que le §5.4 montre sur un exemple.

5 Exemples de processus d'instauration de code

En l'état présent des calculs de simulation et des modèles mathématiques, le processus d'instauration de code offre matière à des études indéfinies. Ici, on se bornera à présenter, sur des exemples de simulation, des phénomènes généraux qui pourraient se reproduire dans tous les processus où une structure s'édifie au cours d'une coopération, traversée par des conflits.

Nous considérerons exclusivement des cas où, comme dans la présentation du programme, l'ensemble - ou graphe - des objets est un cercle; et celui des signaux, un segment.

5.1 Structure analogique d'un code et structure discrète

Partons du cas où les matrices de liens des sujets sont initialisées de façon triviale par la procédure 'tabler', laquelle donne à tous les liens la même valeur, 1. Les actes de communication réussis se distribuent d'abord équiprobablement dans le champs des possibles; mais, par les procédures de renforcement, les liens mis en jeu dans ces premiers succès se consolident rapidement; un code s'établit sur quelques points d'ancrage.

Pour apprécier la qualité d'un code, il vaut la peine d'analyser les matrices conservées par la procédure 'critaber'. Ces matrices, (cf. *supra*, §§4.1.3.2 & 4.2.3), ne diffèrent de la matrice interne des liens qu'en ce que les valeurs négatives sont mises à zéro; et les valeurs positives, changées d'échelle, de sorte que le maximum soit 1000.

On dira que le code a une structure analogique si, particulièrement dans le plan (1, 2), la représentation des ensembles en correspondance est fidèle à la structure des graphes - A, objets; et B, signaux - et suggère un recouvrement de A par B. L'analogie parfaite consiste, dans notre cas, en ce que A soit un polygone régulier sur lequel B pose une chaine aux maillons égaux. Au contraire, on a un code discret si A et B sont partagés en îlots non coordonnés; s'associant (e.g.) deux à deux.

Sur la matrice des liens, l'analogie se révèle par des bandes obliques; dont une permutation circulaire des objets peut faire une bande diagonale; tandis qu'un code discret s'exprime par des blocs rectangulaires; chacun intersection de bandes (bande horizontale, de lignes; et verticale, de colonnes) qui sont quasi vides, en dehors du bloc. Bandes obliques et blocs se mêlent généralement dans les matrices issues de processus de simulation.

À partir de données identiques, l'algorithme, étant régi par une séquence de nombres pseudo-aléatoires, réalise de multiples parties; aboutissant à des codes limite où la part de l'analogique et du discret peut être très variable. Sans prétendre démontrer de théorèmes, nous proposerons ici, d'après notre expérimentation, des suggestions générales illustrées de quelques exemples et corroborées par des raisonnements plausibles.

Le §5.2 offre des exemples d'instauration de code à partir d'une matrice de liens homogène entre A et B; avec ou sans séries d'essais entre deux joueurs. Au §5.3, on considère le conflit entre codes initiaux différents. Le §5.4 montre, après densification des ensembles A et B, l'adaptation efficace du code initialement introduit.

Ne pouvant soumettre au lecteur la totalité des résultats, nous publions, selon les exemples, des matrices de liens ou des plans issus de l'analyse de celles-ci.

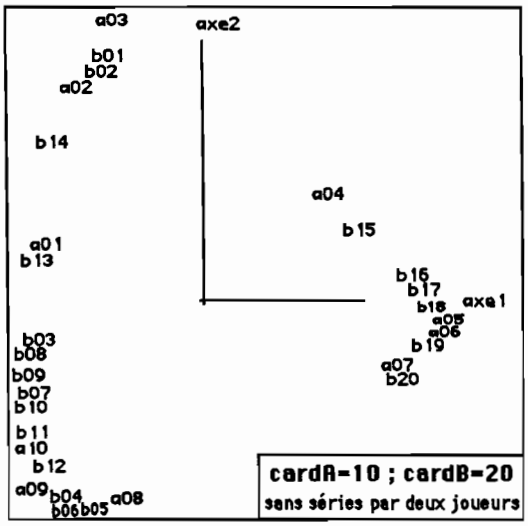
5.2 Instauration d'un code à partir d'une matrice de liens homogène

Au 5.2.1, avec {cardA=10; cardB=20}, on compare la structure des codes obtenus selon qu'il y a, ou non, des séries d'essais entre deux joueurs. Au §5.2.2, avec un ensemble restreint de signaux (cardB=15), on rétablit l'adéquation entre espace des signaux et espace des objets, en réduisant la généralisation du signal.

5.2.1 Distribution des rôles et structure du code

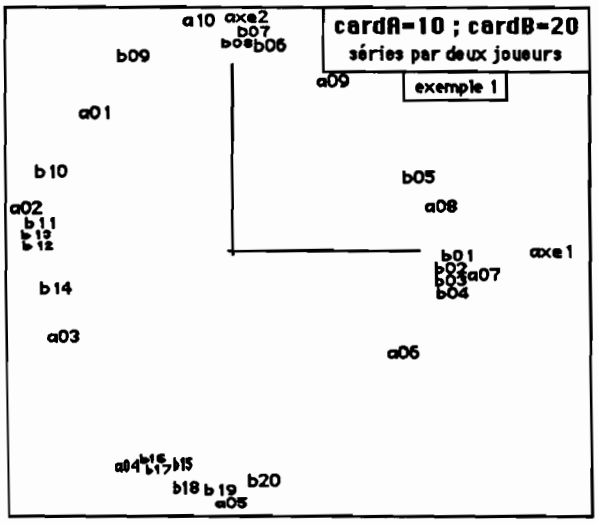
Il apparaît que, s'il n'y a que deux joueurs, qui ont une même matrice de liens, ou, plus généralement, si la partie commence par une série entre deux joueurs, le code tissé entre ceux-ci a plus volontier une structure analogique que si trois joueurs prennent simultanément, en ordre aléatoire, l'initiative des essais; comme si, dans ce dernier cas, la multiplicité des partenaires, compatible avec l'ancrage du code sur des essais réussis, perturbait l'instauration, plus fragile, de zône de gradients entre ceux-ci. Choisis pour illustrer ce phénomène, les exemples considérés dans ce § comportent tous 3 joueurs, (avec cardA=10, cardB=20); mais diffèrent, quant à la présence éventuelle de séries d'essais entre deux joueurs formulant alternativement demandes et réponses.

5.2.1.1 Distribution aléatoire des rôles entre trois joueurs



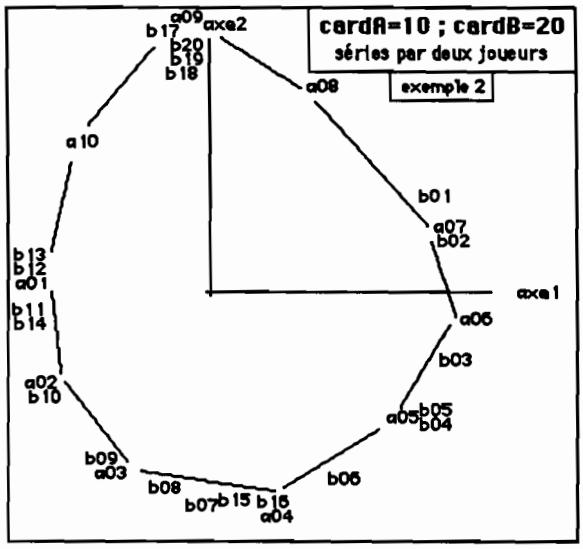
Le cercle A des objets se voit sur le plan (1, 2) issu de l'analyse de la matrice éditée par 'critaber' à l'issue du 11-ème cycle; mais l'espacement des objets a est très irrégulier; et l'ensemble des signaux qui forme plusieurs agrégats resserrés, ne suggère aucunement un enroulement de B sur A.

5.2.1.2 Distribution des rôles par séries d'essais entre deux joueurs: exemple 1

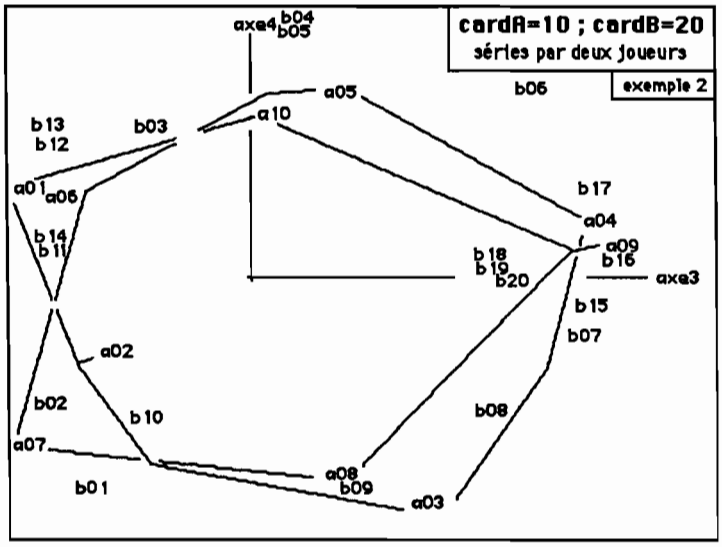


Dans le plan (1, 2), le cercle A des objets présente une image régulière; quant à l'ensemble B des signaux, il apparaît enroulé sur A, à ceci près que {b01 b02 b03 b04} sont étroitement agrégés.

5.2.1.3 Séries d'essais entre deux joueurs: exemple 2



Dans le plan (1, 2), l'ensemble B des signaux, est assez bien étalé (malgré quelques agrégats); mais on ne reconnaît pas sans irrégularité la structure du segment: en décrivant sur A, dans le sens de la montre, l'ensemble B, de b01 à b20, on rencontre deux exceptions, {b15 b16}, qui s'insèrent entre {b06 b07}.



Pour le cercle A des objets, on a, non seulement, dans le plan (1, 2), une image parfaitement régulière; mais encore, dans le plan (3, 4), un double enroulement. Ceci revient à dire qu'en interpolant A pour en faire un cercle continu, paramétré par un angle a, on a pour quatre premiers facteurs: {sin(a), cos(a), sin(2a), cos(2a)}.

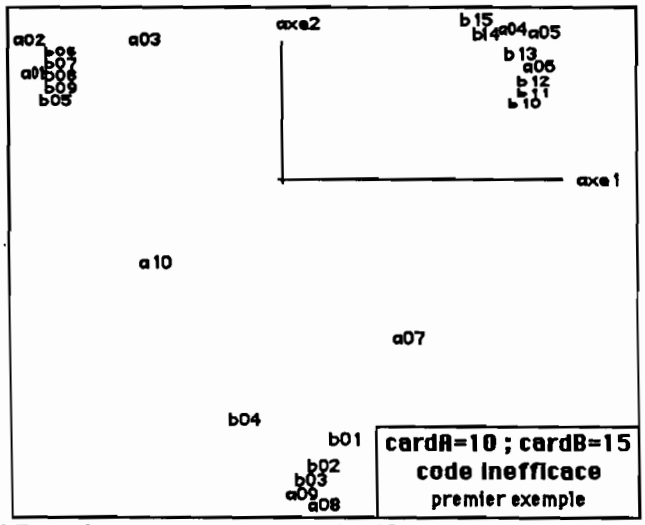
5.2.2 Adéquation entre espace des signaux et espace des objets

Dans les expériences rapportées jusqu'ici, A est un cercle de 10 objets; B, un segment de 20 signaux.

On considère maintenant des expériences où B est un segment de 15 signaux.

Ainsi qu'on l'a dit au §4.3.2.1, on a d'abord obtenu, entre A et B, des codes très peu efficaces: la généralisation, sur un ensemble B restreint, a pour effet de rendre solidaires des blocs de signaux voisins ce qui diminue la capacité du code à transmettre l'information.

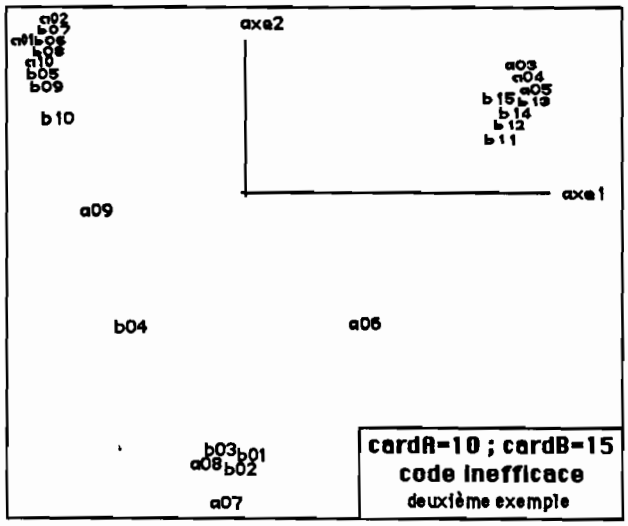
Pour obtenir des codes satisfaisants, il a suffi de réduire la généralisation du signal, en introduisant un coefficient cardB/(2*cardA) dans la version initiale de la procédure 'diffuser'.



5.2.2.1 Premier exemple de code inefficace avec cardB=15

Comme on le voit sur le plan (1, 2) issu de l'analyse de la matrice éditée par 'critaber' à l'issue du 11-ème cycle, les signaux sont concentrés en trois groupes {{b01 b02 b03 b04}, {b05 b06 b07 b08 b09}, {b10 b11 b12 b13 b14 b15}; et, dans le schéma triangulaire obtenu, l'ensemble A des objets, bien que disposé suivant l'ordre du cercle, n'a pas d'espacement régulier.

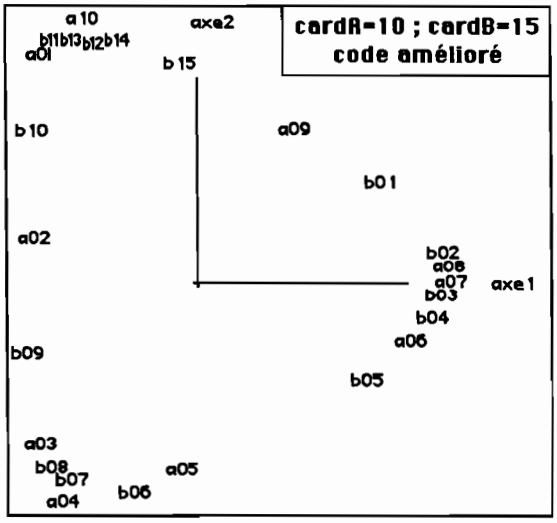
5.2.2.2 Deuxième exemple de code inefficace avec cardB=15



On publie le plan (1, 2) obtenu par analyse de la matrice éditée à l'issue du 21-ème cycle d'une partie jouée sur les mêmes données et avec la même procédure 'diffuser' qu'au § précédent.

Les résultats ne sont pas meilleurs: ici encore, schéma triangulaire; et, puisque l'ensemble A des objets n'occupe que 2 côtés sur 3, le cercle n'est pas véritablement représenté.

5.2.2.3 Exemple de code efficace obtenu avec cardB=15 en réduisant la généralisation du signal



Après avoir modifié la procédure 'diffuser', on obtient un meilleur code: même si la répartition des objets et signaux n'est pas uniforme, le cercle est bien représenté.

5.3 Conflit entre codes initiaux différents

Nous ne considérerons qu'une seule situation expérimentale, avec trois sujets, dont deux reçoivent des codes initiaux contradictoires. Une partie, que nous appellerons "X", a abouti à un code limite commun, par un processus semblable à ceux décrits au §5.2.

Mais, dans une autre partie, notée "Y", un code limite commun ne s'instaure qu'en plusieurs étapes; où l'on peut voir une image des conflits et des coalitions qui naîtraient entre des sujets humains. Bien que le détail des nombres importe peu, il paraît nécessaire de publier des états succesifs des matrices de liens, pour montrer l'instauration de la structure de code.

Disq:tiz1cod : code initial pour le sujet nn=0;

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	20	0	0	0	0	0	0	0	0	0
b02	10	10	0	0	0	0	0	0	0	0
b03	0	20	0	0	0	0	0	0	0	0
b04	0	10	10	0	0	0	0	0	0	0
b05	0	0	20	0	0	0	0	0	0	0
b06	0	0	10	10	0	0	0	0	0	0
b07	0	0	0	20	0	0	0	0	0	0
b08	0	0	0	10	10	0	0	0	0	0
b09	0	0	0	0	20	0	0	0	0	0
b10	0	0	0	0	10	10	0	0	0	0
b11	0	0	0	0	0	20	0	0	0	0
b12	0	0	0	0	0	10	10	0	0	0
b13	0	0	0	0	0	0	20	0	0	0
b14	0	0	0	0	0	0	10	10	0	0
b15	0	0	0	0	0	0	0	20	0	0
b16	0	0	0	0	0	0	0	10	10	0
b17	0	0	0	0	0	0	0	0	20	0
b18	0	0	0	0	0	0	0	0	10	10
b19	0	0	0	0	0	0	0	0	0	20
b20	10	0	0	0	0	0	0	0	0	10

5.3.1 Les données initiales

Disq:tiz2cod : code initial pour le sujet np=1;

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	0	0	20
b02	0	0	0	0	0	0	0	0	10	10
b03	0	0	0	0	0	0	0	0	20	0
b04	0	0	0	0	0	0	0	10	10	0
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
b18	10	10	0	0	0	0	0	0	0	0
b19	20	0	0	0	0	0	0	0	0	0
b20	10	0	0	0	0	0	0	0	0	10

Le code initial attribué au sujet nn=0 a une forme diagonale parfaite: en bref, il enroule le segment B des 20 signaux sur le cercle A des 10 objets. Le code initial attribué au sujet np=1 a la même structure, mais suivant l'autre diagonale; ce qui revient à enrouler B sur A avec un sens de rotation opposé.

Il semble que soit ainsi créée, entre les deux structures analogiques, de nn et np, une opposition qui ne laisse place à aucune structure analogique intermédiaire; mais ne peut se résoudre que par la dominance de l'une sur l'autre; ou une rupture en blocs; esquissant, éventuellement, des fragments de bande oblique. Le troisième sujet, quant à lui, ne recevant pas de code initial (sinon les liens uniformes créés par 'tabler') est susceptible de se conformer soit à l'un, soit à l'autre, des deux premiers.

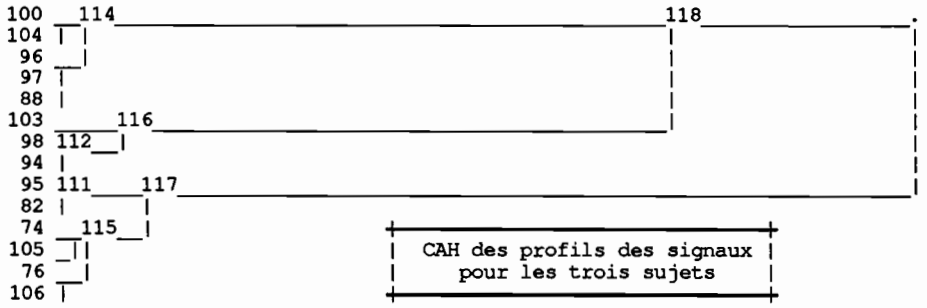
5.3.2 Un exemple de convergence rapide: la partie "X"

Les joueurs font des séries par deux. Dans la partie "X", dès la première série, les joueurs 0 et 1 ont élaboré des codes similaires; à la dixième série, ne subsistaient entre les trois joueurs que des différences mineures.

"X": con it entre codes initiaux: état à cy 21;n=2

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	316	257	86	0	0	0	0	4	47	230
b02	399	291	78	0	0	0	0	4	94	329
b03	304	230	76	0	0	0	0	7	88	268
b04	155	129	49	0	0	0	0	7	48	140
b05	47	22	0	0	0	0	0	8	43	60
b06	2	0	0	0	2	5	35	53	49	17
b07	0	0	0	3	72	120	112	58	4	0
b08	0	0	0	119	256	270	152	8	0	0
b09	0	0	41	316	462	409	162	0	0	0
b10	0	0	66	474	654	580	217	0	0	0
b11	0	0	174	689	854	675	154	0	0	0
b12	0	0	471	937	941	533	4	0	0	0
b13	0	2	502	776	690	279	2	0	0	0
b14	0	0	10	132	239	241	148	28	1	2
b15	0	0	0	0	2	420	742	729	414	19
b16	0	0	0	0	0	337	869	1000	705	108
b17	93	0	0	0	0	2	247	585	669	481
b18	785	487	3	0	0	0	0	9	393	774
b19	996	835	216	0	0	0	0	0	107	734
b20	682	606	220	0	0	0	0	0	17	442

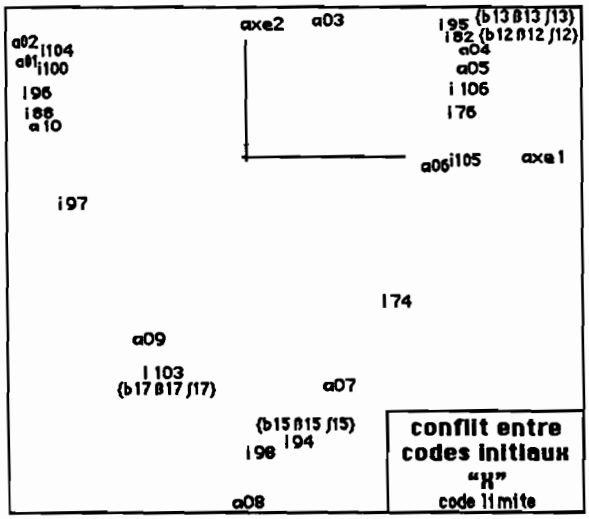
Nous donnons un état de la matrice après la 21-ème série, pour l'un des sujet. Quant à la précision du code, on voit que, du fait de la loi de généralisation choisie, chaque signal est lié à quatre ou cinq objets consécutifs: cette largeur se retrouve dans tous les autres exemples.



Afin d'apprécier l'accord des trois codes, on a analysé un tableau (60 × 10), (bB+ßB+∫B) × A, superposant les matrices de liens des trois sujets après 21 séries d'essais; les profils sur A d'un même signal, e.g. le 13-ème, étant notés {b13 ß13 ∫13}, selon qu'il s'agit du 1-er, du 2-ème ou du 3-ème sujet.

Dans la partition en 14 classes extraite de la CAH de (bB+ßB+∫B), 4 classes s'identifient à un signal dont s'agrègent les 3 profils: e.g., {b13 ß13 ∫13} constitue la classe i95.

Pour les autres signaux, l'accord entre les sujets paraîtra bon, si l'on tient compte de la disposition des centres des classes dans le plan (1, 2).



D'autre part, le conflit une fois arbitré, le cercle A des objets présente une image régulière. Toutefois, au haut du graphique du plan (1, 2) on voit, entre les objets {a02, a04}, un grand espace où a03 n'est accompagné d'aucun signal.

c	Partition de B en 14 classes : Sigles des signaux de la classe c
100	b01 b02 ∫02 ß19 b19 b20 ∫04 ß20 ∫03
104	b03 ∫01 ∫19 b04 b05 ∫20
96	ß01 ß02 ß03 ß18
97	∫05 ß05
88	ß04 b18 ∫18
103	ß17 b17 ∫17
98	∫06 ß06 ß16 b16 ∫16
94	ß15 ∫15 b15
95	b13 ß13 ∫13
82	ß12 b12 ∫12
74	∫07 ß07
105	b06 b07 ß08 ∫14 b14 b08 ß10 ∫08 ß09
76	∫09 ∫10 ß11
106	ß14 b09 b10 ∫11 b11

"X": conflit entre codes initiaux : état à cy 21 ; (bB+ßB+∫B) × A;

trace	:	1.542e+0								
rang	:	1	2	3	4	5	6	7	8	9
lambda	:	8435	5600	912	405	27	19	9	6	3 e-4
taux	:	5472	3633	591	262	17	12	6	4	2 e-4
cumul	:	5472	9105	9696	9958	9976	9988	9994	9998	10000 e-4

5.3.3 Une partie sans convergence après 12 séries: la partie "Y"

"Y": conflit entre codes initiaux: code à cy 12; n°1

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	12	8	12	8	4	4	0	0	0	0
b02	26	63	67	41	0	0	0	0	0	0
b03	84	237	249	154	4	0	0	0	0	0
b04	182	450	455	247	0	0	0	0	0	0
b05	229	439	419	186	0	0	0	0	0	0
b06	136	227	236	120	0	0	0	0	0	0
b07	0	7	36	30	0	15	36	43	8	0
b08	0	0	0	0	8	147	259	277	116	1
b09	0	0	0	0	112	354	442	387	107	0
b10	0	0	0	48	416	559	463	197	6	0
b11	0	0	0	259	565	576	342	21	0	0
b12	0	0	0	289	537	493	270	4	0	0
b13	0	0	0	167	434	456	335	21	0	0
b14	0	0	0	47	242	300	266	66	0	0
b15	0	0	0	12	56	65	54	21	0	0
b16	70	29	3	0	0	0	0	10	77	102
b17	237	15	0	0	0	0	1	106	353	387
b18	501	43	0	0	0	0	0	194	686	793
b19	701	150	0	0	0	0	0	149	792	1000
b20	594	197	0	0	0	0	0	58	541	748

L'initialisation est identique à celle de la partie "X". Mais il se trouve qu'ici, dans la partie "Y", au terme de la première série, qui comprend 10*cardA=100 essais, joués entre deux sujets dont les codes initiaux s'opposent, les sept signaux {b01 ... b07}, n'ont reçu aucun renforcement positif, ni chez l'un ni chez l'autre des joueurs: la place de ces signaux dans le code reste donc à fixer.

"Y": conflit entre codes initiaux: code à cy 12; n°2

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	0	0	23
b02	0	0	0	0	0	0	0	0	11	11
b03	0	0	0	0	0	0	0	0	23	0
b04	0	0	0	0	0	0	0	11	11	0
b05	0	0	0	0	0	0	0	23	0	0
b06	0	0	0	0	0	0	11	11	0	0
b07	0	0	0	0	4	36	85	73	32	0
b08	0	0	0	0	0	203	362	403	190	0
b09	0	0	0	0	90	520	685	637	213	0
b10	0	0	0	58	569	753	655	305	6	0
b11	0	0	23	394	778	684	367	0	0	0
b12	0	0	62	390	588	480	207	0	0	0
b13	0	0	6	220	345	345	224	2	0	0
b14	0	0	0	68	162	192	168	34	0	0
b15	0	0	17	6	24	21	30	19	6	0
b16	73	15	0	0	0	0	0	41	113	124
b17	198	0	0	0	0	0	0	132	356	358
b18	524	89	0	0	0	0	0	181	637	721
b19	872	326	0	0	0	0	0	87	740	1000
b20	744	369	0	0	0	0	0	45	507	802

Or, quant on interrompt la partie, après douze séries, jouée chacune entre deux sujets, un code commun s'est établi entre le 1-er et le 3-ème sujet; mais

ce code ne s'est pas imposé au 2-ème sujet. Dans le code de celui-ci, le signal b07 a reçu quelques renforcements; mais les six premiers {b01 ... b06} n'en ont pas reçu. Le lecteur appréciera, par lui-même, sur les deux matrices de code, la différence qui sépare les sujets 1 et 2, au terme de la série cy=12.

5.3.4 Reprises de la partie "Y"

La partie "Y" a été interrompue après douze essais: pour la reprendre dans les conditions mêmes où elle se serait poursuivie, il faudrait mettre le tableau xab des liens dans son état après cy=12. À la suppression près des liens négatifs, le listage des matrices de code nous a conservé ce tableau. À la question:

```
faut-il lire un tableau donné pour le code du sujet … O ou N ,
```

posée par la procédure 'initier', (cf. *supra* §4.1.3.3,) on répond donc trois fois 'O'; en donnant pour chaque sujet le nom d'un fichier séparé contenant sa matrice de code à cy=12.

De plus, afin de brasser les codes, on a répondu 'N' à la question:

```
les joueurs font-ils des séries par deux(D) ou non(N) ,
```

posée par la procédure 'choisir'.

"Y": première reprise de partie: code à cy 21; n°2

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	0	0	9
b02	0	0	0	0	0	0	0	0	4	4
b03	0	0	0	0	0	0	0	0	9	0
b04	0	0	0	0	0	0	0	4	4	0
b05	0	0	0	0	0	0	0	9	0	0
b06	0	0	0	0	0	3	9	11	5	0
b07	0	0	0	0	0	66	126	131	69	2

Au cours de cette reprise de la partie "Y", poursuivie jusqu'à sa 21-ème série, les codes des sujets n°1 et n°3 sont peu modifiés. Le sujet n°2 ne s'y est pas encore rallié; mais le signal b06 reçoit des renforcements; les cinq premiers {b01 ... b05} restant seuls hors du jeu. On en jugera d'après les 7 premières lignes de la matrice de code, seules publiées ici.

"Y": deuxième reprise de partie: code à cy 21; n°2

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	0	0	4
b02	0	0	0	0	0	0	0	0	2	2
b03	0	0	0	0	0	0	0	0	4	0
b04	0	0	0	0	0	0	0	2	2	0
b05	0	0	0	0	0	0	0	4	0	0
b06	0	0	0	0	1	22	38	26	9	1
b07	0	0	0	0	13	127	194	163	70	0

La partie "Y" est donc reprise une deuxième fois, de la même manière que la première. Entre la 21-ème et la 31-ème série, les signaux {b01 ... b05} s'intègrent au code actif du sujet n°2.

"Y": deuxième reprise de partie: code à cy 31; n°2

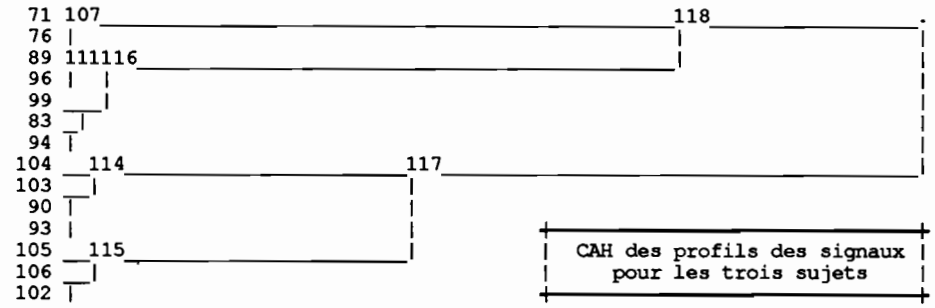
10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	4	6	3	1	0	0	0	0	0	5
b02	11	16	9	3	0	0	0	0	0	5
b03	11	21	16	7	0	0	0	0	0	0
b04	5	19	17	10	1	0	0	0	0	0
b05	1	7	7	4	1	0	0	0	0	0
b06	0	0	0	0	1	24	40	28	8	0
b07	0	0	0	0	11	124	184	159	66	0

Comme pour la première reprise, nous publions ici les 7 premières lignes des états intermédiaires de la matrice de code.

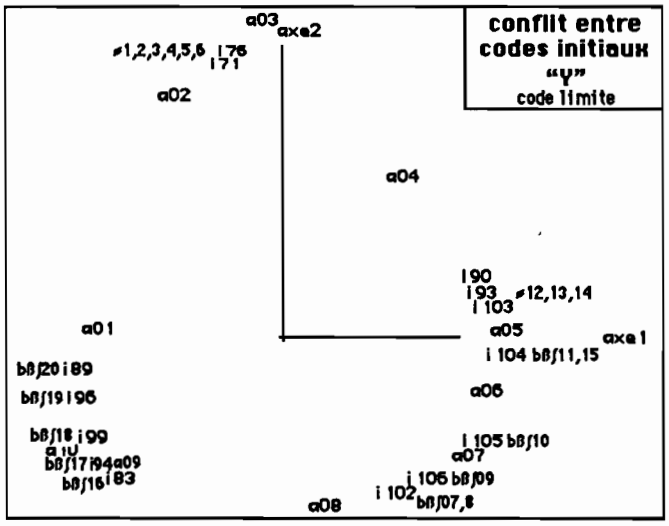
"Y": deuxième reprise de partie: code à cy 111; n°2

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	32	63	61	31	0	0	0	0	0	0
b02	80	167	178	104	0	0	0	0	0	1
b03	145	307	338	210	0	0	0	0	0	0
b04	187	389	417	257	0	0	0	0	0	0
b05	147	307	322	197	0	0	0	0	0	0
b06	40	110	118	72	0	0	0	0	0	1
b07	0	0	0	0	2	126	199	187	86	0
b08	0	0	0	0	4	356	528	508	240	0
b09	0	0	0	0	103	585	765	645	206	0
b10	0	0	0	1	370	686	695	397	2	0
b11	0	0	2	356	659	667	381	1	0	0
b12	0	0	173	641	825	639	156	0	0	0
b13	0	0	171	595	752	572	121	0	0	0
b14	0	0	71	344	467	381	121	0	0	0
b15	0	0	3	56	112	105	63	5	0	0
b16	126	0	0	0	0	0	2	157	269	256
b17	390	3	0	0	0	0	0	266	585	618
b18	705	96	0	0	0	0	0	241	782	934
b19	868	303	0	0	0	0	0	76	717	1000
b20	614	305	0	0	0	0	0	5	403	645

Finalement, par généralisation et renforcement, le sujet n°2 a rempli le bloc {b01...b06} × {a01...a04} de la matrice de code. Une classification, analogue à celle qui illustre le §5.3.2, atteste que les codes des trois sujets ont convergé.



Dans la partition en 14 classes extraite de la CAH de (bB+ßB+JB), 7 classes s'identifient à un signal dont s'agrègent les 3 profils. Pour les autres signaux, l'accord entre les sujets est bon, si l'on tient compte de la proximité



c	Partition de B en 14 classes : Sigles des signaux de la classe c
71 | 76 |	b01 ∫01 ß01 b05 b06 ∫06 ∫05 b04 ß02 ∫02 ß05 ∫04 ß04 ß06 b02 b03 ß03 ∫03
89 | 96 |	∫20 ß20 b20 ß19 b19 ∫19
99 | 83 | 94 |	∫18 ß18 b18 ∫16 ß16 b16 ∫17 ß17 b17
104 |	ß15 ß11 b15 ∫15 b11 ∫11
103 | 90 | 93 |	b12 b14 ∫14 ß13 ß12 ß14 b13 ∫13 ∫12
105 |	∫10 ß10 b10
106 | 102 |	b09 ß09 ∫09 b07 ∫07 ß07 ∫08 ß08 b08

des centres des classes dans le plan (1, 2). Le conflit une fois arbitré, le cercle A des objets présente une image régulière; mais l'espace B des signaux est scindé en plusieurs segments; avec des gradations: {07 08 09 10}, {16 17 18 19 20}; et des classes étroitement agrégées: {1 2 3 4 5 6} et {12 13 14}.

5.4 Adaptation après densification des ensembles A et B

Nous considérerons successivement deux exemples. Le premier est présenté en détail dans les §§5.4.1 à 5.4.3: dans cet exemple, le code est

initialisé, avant dédoublement, par une matrice de liens choisie pour produire un code dont la structure analogique est assez nette; et la densification améliore cette structure.

Dans le deuxième exemple, objet du §5.4.4, on densifie successivement deux fois les ensembles A et B; ainsi partant d'un code discret on aboutit à un code qui rend parfaitement compte de la structure circulaire de l'ensemble des objets.

Afin de distinguer les graphiques afférents aux deux exemples, on écrit pour le premier '∂ensification'; et pour le second: 'Δensification'.

5.4.1 Ensembles {A, B} et ensembles densifiés {Ad, Bd}

L'ensemble A des objets est un cercle, ou polygone régulier à 10 sommets; par densification, on a un polygone Ad à 20 sommets. Dans l'ordre de création par 'grapher', les 10 sommets nouveaux se placent après les sommets préexistants:

```
a01 a02 a03 a04 a05 a06 a07 a08 a09 a10 aAB aAJ aBC aCD aDE aEF aFG aGH aHI aIJ
```

l'ordre cyclique naturel serait:

```
a01 aAB a02 aBC a03 aCD a04 aDE a05 aEF a06 aFG a07 aGH a08 aHI a09 aIJ a10 aAJ.
```

De même, par densification du segment B, de 20 sommets on a un segment Bd de 39 sommets; où l'ordre naturel est:

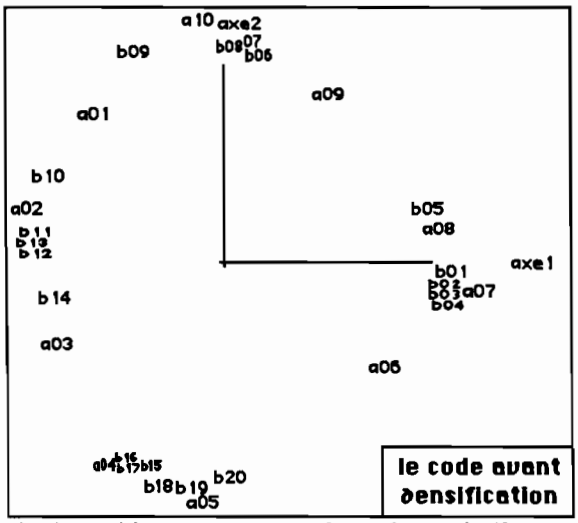
```
b01 bAB b02 bBC b03 bCD b04 bDE b05 ... ... ... ... b16 bPQ b17 bQR b18 bRS b19 bST b20
```

5.4.2 Premier exemple: initialisation du code

Les codes des deux sujets 1 et 2 sont initialisés par un même tableau carB × carA, 20 × 10 qui est l'état du code obtenu dans une partie antérieure. Nous donnons quelques lignes de la matrice de code; ainsi que le plan (1,2) issu de l'analyse de cette matrice, avec le tableau des valeurs propres.

```
disq:titcodinit : initialisation avant densification
10    a01  a02  a03  a04  a05  a06  a07  a08  a09  a10
b01     0    0    0    0   45  499  756  630  279    0
b02     0    0    0    0  135  714 1000  865  334    0
b03     0    0    0    0  147  634  824  677  194    0
b04     0    0    0    0   86  448  576  440  101    0
b05     0    0    0    0    1  123  247  274  186   71
b06   249    7    0    0    0    0   10  216  433  426
b07   431   30    0    0    0    0    0  254  616  665
.....  ..........................................................................
```

Quant à la précision du code, on voit sur la matrice que, comme dans les exemples précédents, chaque signal est notablement lié à quatre ou cinq objets consécutifs. Il en résulte que, sur la matrice de code initial Bd × Ad, 39 × 20, complétée par interpolation (cf. *supra* §4.4.3), la largeur du profil de chaque signal est à peu près double.



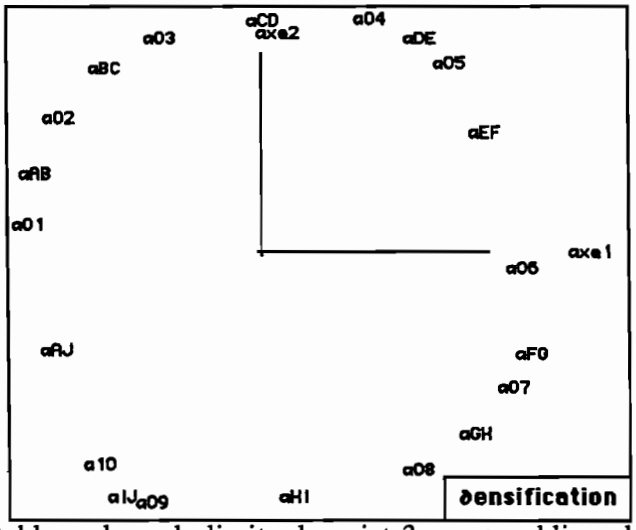
Le cycle de A est bien reconnu par le code; mais l'enroulement de B sur A n'est pas parfait. Il y a des amas de points de B dont les profils sur A sont similaires, e.g. {b01 b02 b03 b04}; entre cet amas et l'autre extrémité b20 de B, dans le plan (1, 2), le quadrant (F1>0; F2<0) du cycle, est vide de tout signal; et, de b01 à b05, apparaît un rebroussement en b04.

5.4.3 Premier exemple: le code limite après densification

```
[titð3S]: code limite après densification
20 a01 aAB a02 ..... aCD a04 aDE a05 aEF a06 aFG a07 aGH a08 aHI a09 aIJ a10 aAJ
b01   0   0   0 .....   0   0   0   0   0   0   0 130 528 686 549 128   0   0   0
bAB   0   0   0 .....   0   0   0   0   0   0   0 379 770 800 491   0   0   0   0
b02   0   0   0 .....   0   0   0   0   0   1 262 686 796 543  68   0   0   0   0
bBC   0   0   0 .....   0   0   0   0   0 161 657 852 676 183   0   0   0   0   0
b03   0   0   0 .....   0   0   0   0   0 551 924 862 436   0   0   0   0   0   0
bCD   0   0   0 .....   0   0   0   0 184 767 963 726 164   0   0   0   0   0   0
b04   0   0   0 .....   0   0   0   0 246 613 662 434  11   0   0   0   0   0   0
bDE   0   0   0 .....   0   0   0   1  70 221 251 191  38   0   0   0   0   0   0
b05   0   0   0 .....   0   0   0   0   0   0   2  26 113 153 130  39  21   0   0
```

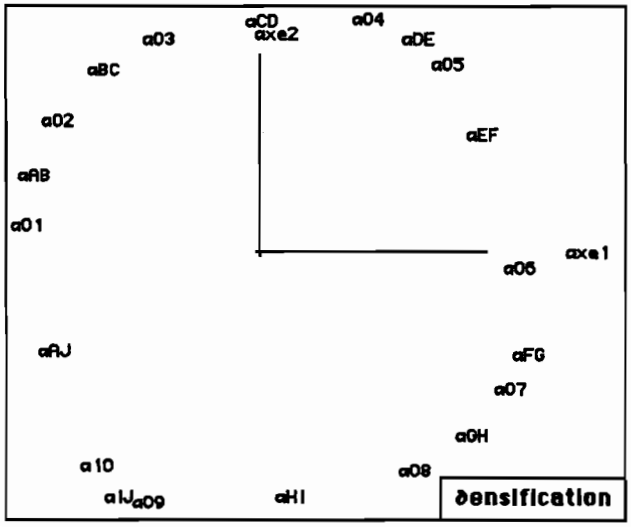
Du fait du grand nombre d'objets et de signaux, la convergence du code, sans présenter d'anomalies (telles que celles créées au §5.3 par le conflit entre deux codes analogiques incompatibles), n'a été achevée qu'après de longues séries d'essais: la place de certains signaux, peu utilisés étant finalement fixée par généralisation.

De façon précise, au lieu des séries de 10*cardA essais de communication, prévues dans la version du programme commentée au §4.2.3, on a permis à l'utilisateur de demander des séries plus longues. Et, ici, après une série de 10*cardA=200 essais, on a fait dix séries de 100*cardA=2000 essais.



Du tableau du code limite du sujet 3, nous publions les 9 premières lignes: b01 bAB b02 bBC b03 bCD b04 bDE b05... De b01 à b04, on voit, sur la matrice, un glissement régulier; mais ensuite le sens du glissement s'inverse: ce rebroussement se voit bien dans le plan (1, 2) issu de l'analyse du code.

Quant à la précision du code, comme dans tous les exemples précédents, chaque signal n'est notablement lié qu'à quatre ou cinq objets qui sont consécutifs. Ainsi, le processus simulé de communication a amélioré



l'efficacité du code initial: en densifiant A et B, on a doublé la précision du code, en ce sens que si l'on considère l'espace A des objets comme un cercle paramétré par un angle, la largeur angulaire de la zone liée à chaque signal se trouve divisée par 2.

La structure géométrique globale du code limite apparaît dans le plan (1, 2). Pour l'ensemble Ad des 20 objets, la forme du cycle est parfaite; et la distribution des points est à peu près uniforme. Pour l'ensemble Bd des 39 signaux, on a d'abord un mouvement régulier, de b01 à b04, sur 1/8 de cercle parcouru dans le sens trigonométrique; puis, après rebroussement en b04, l'enroulement du segment sur le cercle, dans le sens inverse, est bon, de b04 jusqu'à b20.

5.4.4 Un exemple de double densification

code (0) : trois joueurs ser=100;cy 11;n°3;par Deux;

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	1	0	0	0	0	3	339	737	780	491
b02	1	0	0	0	0	2	441	931	1000	634
b03	0	0	0	0	0	2	417	765	771	441
b04	0	0	0	0	0	5	319	491	435	191
b05	2	1	2	1	0	1	56	104	80	24
b06	168	364	385	215	3	0	0	0	0	0
b07	403	748	763	398	2	0	0	0	0	0
b08	592	902	836	348	0	0	0	0	0	14
b09	697	821	603	76	0	0	0	0	0	233
b10	658	612	291	2	0	0	0	0	0	409
b11	419	323	64	1	0	0	0	0	96	350
b12	117	60	2	1	0	0	0	0	49	124
b13	0	0	2	74	187	224	138	0	0	0
b14	0	0	0	233	488	541	328	1	0	0
b15	0	0	0	417	767	772	442	2	0	0
b16	0	0	5	560	890	822	397	1	0	0
b17	0	0	49	517	719	627	216	0	0	0
b18	0	0	87	330	421	354	88	0	0	0
b19	0	0	39	141	191	162	45	0	0	0
b20	0	0	2	33	59	53	20	0	0	0

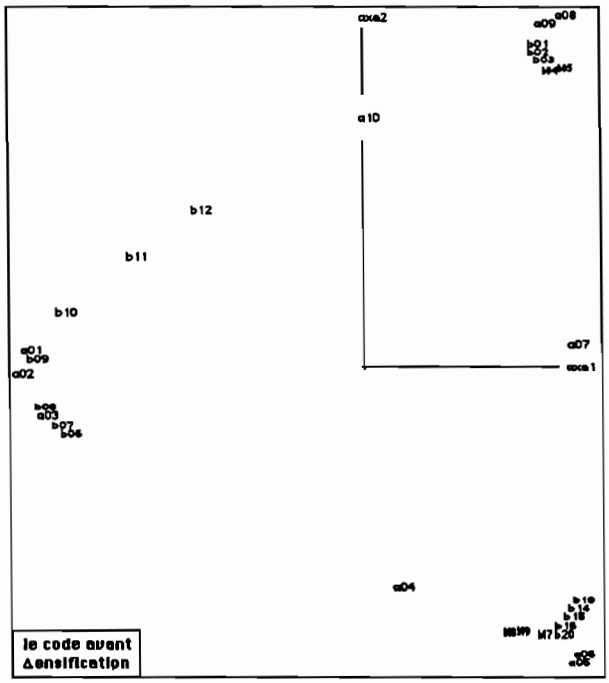
5.4.4.0 Processus initial avant densification: code (0)

Comme au §5.4.1, on part d'un cercle A de 10 objets, et d'un segment B de 20 signaux. Le processus d'instauration se déroule jusqu'à produire une matrice de liens, en trois blocs; structure à laquelle répond, sur le plan (1, 2) issu de l'analyse factorielle, un aspect triangulaire; où toutefois l'ordre cyclique de A se montre par quelques points intermédiaires entre les amas qui sont aux sommets du triangle. Tel est le code (0), avant 'Δensification'.

5.4.4.1 Processus après première densification: code (1)

Avec une première densification, le processus reprend entre un cercle Ad de 20 objets et un ensemble Bd de 19 signaux: les notations pour Ad et Bd étant les mêmes qu'au §5.4.1.

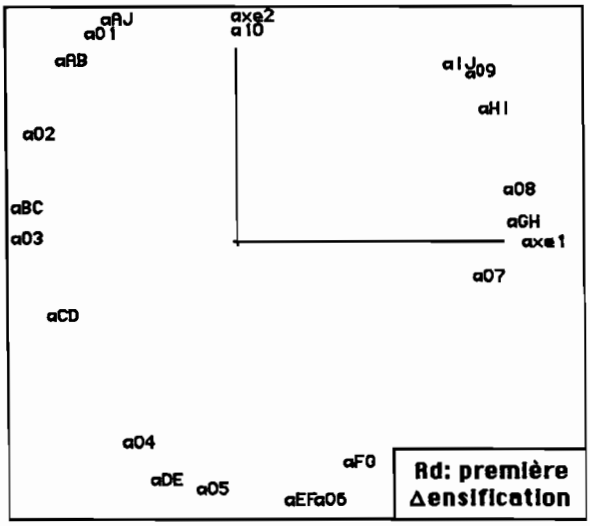
On s'arrête à un code (1) où le caractère analogique est nettement plus marqué que dans le code (0), utilisé pour l'initalisation. Nous publions, sur



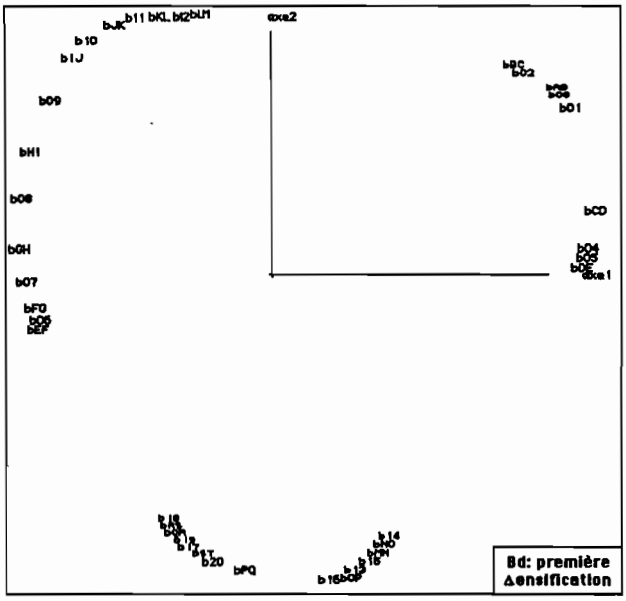
des graphiques séparés, la représentation de Ad et Bd dans le plan (1, 2); avec un extrait de la matrice des liens, restreint à A × B. Comme au §5.4.3, il apparaît qu'en densifiant A et B, on a doublé la précision du code.

code (1) : trois joueurs ser=100;cy 11;n°3;par Deux;

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	220	341	1
b02	0	0	0	0	0	0	0	40	812	280
b03	0	0	0	0	0	0	0	193	518	35
b04	0	0	0	0	0	0	750	830	0	0
b05	0	0	0	0	0	0	260	277	0	0
b06	0	16	433	166	0	0	0	0	0	0
b07	0	186	740	93	0	0	0	0	0	0
b08	0	534	433	0	0	0	0	0	0	0
b09	328	525	1	0	0	0	0	0	0	0
b10	652	240	0	0	0	0	0	0	0	33
b11	593	0	0	0	0	0	0	0	0	349
b12	277	0	0	0	0	0	0	0	0	305
b13	1	0	0	0	48	146	5	0	0	1
b14	0	0	0	0	49	481	145	0	0	0
b15	0	0	0	0	147	590	81	0	0	0
b16	0	0	0	0	300	381	0	0	0	0
b17	0	0	0	480	502	0	0	0	0	0
b18	0	0	0	510	366	0	0	0	0	0
b19	0	0	0	163	156	0	0	0	0	0
b20	0	0	0	50	80	0	0	0	0	0

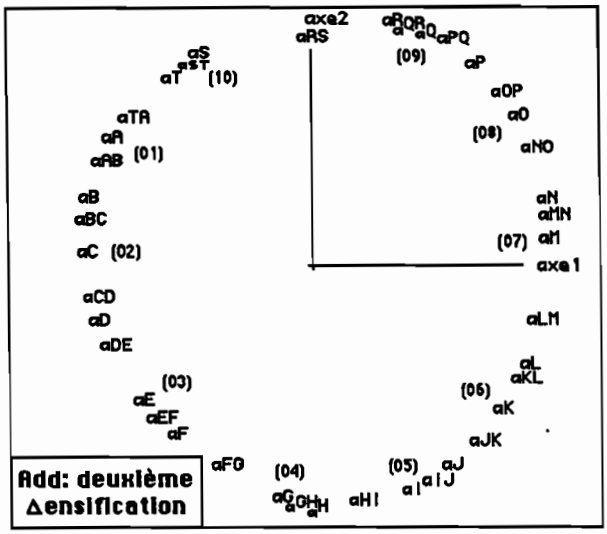


De plus la valeur analogique du code est améliorée.



état de code(1);ser=100;cy 11;n°3;par Deux; trace : 4.320e+0

rang	:	1	2	3	4	5	6	7	8	9	10	
lambda	:	9273	9174	7475	6885	5944	2657	748	517	279	130	e-4
taux	:	2147	2124	1730	1594	1376	615	173	120	65	30	e-4
cumul	:	2147	4270	6001	7594	8970	9585	9758	9878	9943	9973	e-4



5.4.4.2 Processus après deuxième densification: code (2)

On procède alors à une deuxième densification, qui produit des ensembles Add, cercle de 40 objets; et Bdd, segment de 37 signaux; ensembles entre lesquels le processus se poursuit, à partir de la matrice de liens où on était parvenu après la première densification.

Ici, pour plus de clarté, on utilise, sur le graphique, des sigles pour Ad et Add créés suivant de nouveaux principes. Les objets de Ad sont chacuns désignés par la minuscule 'a', suivie d'une capitale de A à T, prise dans l'ordre du cercle; soit:

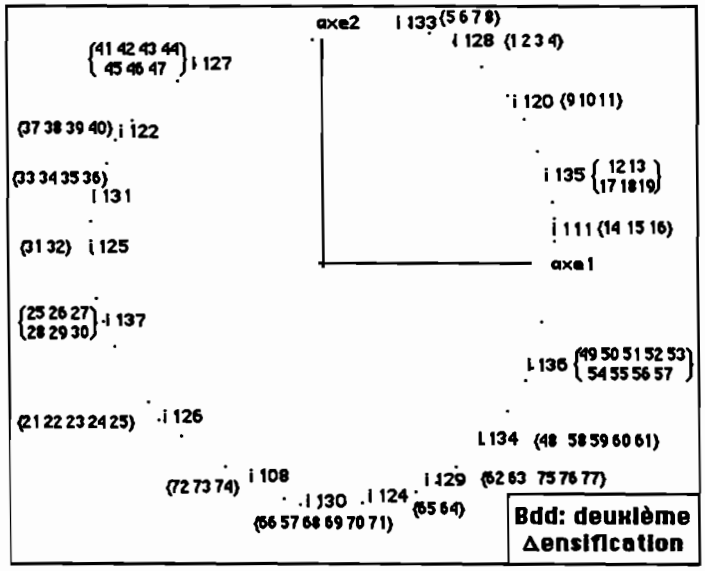
```
Ad = {aA aB aC aD aE aF aG aH aI aJ aK aL aM aN aO aP aQ aR aS aT} ;
 A = {aA    aC    aE    aG    aI    aK    aM    aO    aQ    aS   } ;
```

en particulier, les objets de A sont désignés par les lettres de rang impair, de A à S. Les objets nouveaux de Add, créés par densification de Ad, reçoivent des sigles à trois caractères; avec, après 'a', deux capitales qui désignent les objets de Ad entre lesquels s'insère le nounel objet; soit:

```
Add = {aA aAB aB aBC aC aCD aD aDE … … … aP aPQ aQ aQR aR aRS aS aST aT} ;
```

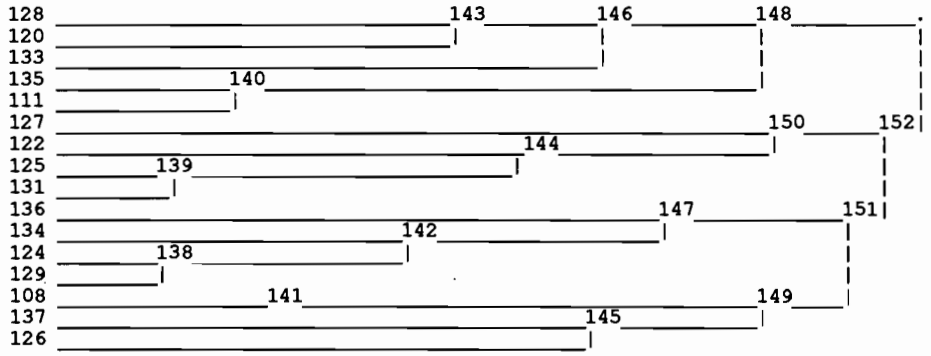
de plus, dans le plan (1, 2) issu de l'analyse de la matrice de liens, chaque élément de A est accompagné de son numéro initial: (01) pour aA, (02) pour aC, …, (10) pour aS.

Ainsi on voit aisément que le code (2), obtenu après la deuxième densification, améliore encore la représentation du cercle des objets.



Quant au segment Bdd, on en a renuméroté les signaux de b01 à b77: ainsi les signaux initiaux, de B, sont numérotés de 4 en 4 à partir de 1: {b01, b05, ..., b77}; les signaux complémentaires de Bd sont {b03, b07, ..., b75}; enfin, les signaux de Bdd-Bd ont les numéros pairs de b02 à b76.

c	Partition de Bdd en 17 classes : Sigles des signaux de la classe c
*128	b01 b02 b03 b04
*120	b11 b09 b10
*133	b05 b06 b08 b07
135	b17 b13 b12 b19 b18
*111	b16 b14 b15
*127	b41 b47 b46 b45 b44 b43 b42
*122	b37 b38 b39 b40
*125	b32 b31
*131	b36 b35 b34 b33
*136	b49 b57 b50 b56 b51 b52 b55 b53 b54
134	b48 b58 b59 b61 b60
*124	b65 b64
129	b62 b63 b75 b77 b76
*108	b72 b74 b73
*130	b71 b68 b67 b70 b66 b69
*137	b30 b29 b25 b26 b28 b27
*126	b24 b20 b23 b21 b22



Nous donnons une CAH des signaux, avec, dans le plan (1, 2) les centres des classes de la partition définie par les 16 nœuds les plus hauts; et le contenu des classes.

On voit que la plupart de ces classes correspondent à un intervalle de Bdd; mais, sur le cycle, l'enroulement de Bdd présente des rebroussements.

code (2) : trois joueurs ser=100 ; cy 21;n°3;par Deux;

10	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
b01	0	0	0	0	0	0	0	0	159	0
b02	0	0	0	0	0	0	0	0	716	0
b03	0	0	0	0	0	0	0	449	0	0
b04	0	0	0	0	0	0	297	1	0	0
b05	0	0	0	0	0	0	118	58	0	0
b06	0	0	410	0	0	0	0	0	0	0
b07	0	0	451	0	0	0	0	0	0	0
b08	0	317	0	0	0	0	0	0	0	0
b09	0	499	0	0	0	0	0	0	0	0
b10	591	0	0	0	0	0	0	0	0	0
b11	2	0	0	0	0	0	0	0	0	234
b12	0	0	0	0	0	0	0	0	0	333
b13	0	0	0	0	0	89	0	0	0	0
b14	0	0	0	0	0	75	104	0	0	0
b15	0	0	0	0	0	428	0	0	0	0
b16	0	0	0	0	65	173	0	0	0	0
b17	0	0	0	1	201	0	0	0	0	0
b18	0	0	0	332	1	0	0	0	0	0
b19	0	0	0	502	0	0	0	0	0	0
b20	0	0	0	0	436	0	0	0	0	0

Comme pour le code (1), nous publions, pour le code (2), un extrait de la matrice des liens, restreint à A × B : ici, on conserve les sigles originaux des éléments de A et B.

La précision du code a encore augmenté; et les trois blocs initiaux du code (0) sont devenus trois bandes diagonales {b01... b07} {b08... b12} {b13... b20}; dont la dernière présente toutefois des rebroussements, visibles sur la CAH.

5.5 Conclusion: capacité du programme à traiter l'information

5.5.1 Hasard et déterminisme

Le seul fait que les graphes A et B sont orientables implique que le code limite ne peut être déterminé par les données initiales mais est, en partie, l'effet du hasard. Ainsi, dans le conflit de codes initiaux étudié au §5.3, il y a, au départ, symétrie entre deux orientations du cercle A: {a01... a10} et {a10... a01}. Cette symétrie ne subsiste pas dans le code limite qui montre, notamment, dans la matrice de liens du §5.3.4, une bande partielle orientée suivant la deuxième diagonale du tableau.

Cependant après une phase initiale marquée par des ruptures, l'algorithme semble conduire le code vers sa limite comme le ferait un système différentiel. Dans ce même §5.3.4, l'assimilation du sujet n°2 au code des deux autres se fait par une diffusion de masse vers le bloc {b01...b06} × {a01...a04} de la matrice des liens. Et, au §5.4, la densification, éventuellement répétée, semble équilibrer le code comme par la force d'expansion des divers objets et signaux.

5.5.2 Local et global

Un programme d'analyse factorielle détermine de façon certaine la structure globale d'un objet A d'après un tableau de relations, mieux que ne le fait l'algorithme d'instauration de code, mettant A en relation avec B.

D'ailleurs, un algorithme d'approximation stochastique permet d'obtenir les résultats d'une analyse factorielle au terme d'un processus.

Mais l'intérêt du programme d'instauration de code vient de la pauvreté des informations qu'il traite: des relations de contiguïté. On peut dire que, par le code, il range les objets suivant une forme qu'on ne lui a pas indiquée globalement, qu'il s'élève du local au global. Alors que le sujet humain voit A et B - par exemple la carte d'Europe et un segment - et conçoit une stratégie de codage qu'il tente de suggérer à son partenaire, le programme, qui est vide de tout système de formes, n'en parvient pas moins à découper des zones et même à tracer des lignes.

5.5.3 Perception et finalité

D'autre part, il ne s'agit pas d'optimiser une objet mathématique complexe (le système des matrices de liens) d'après une quantité critère, mais de réaliser une fonction par une technique. Nous l'annoncions au §1: la forme des données, ici A et B, par la finalité, (la tâche à accomplir: communiquer,) donne structure au comportement. Pour un tel automate finalisé, on a peine à ne pas user d'un langage anthropomorphique. Nous y reviendrons au § suivant.