Implémentation de l'algorithme de clustering MCL

Benabderrahmane Mohammed

1. Introduction

Le projet consiste à implémenter en R l'algorithme MCL de partitionnement de graphe (clustering) à partir d'un graphe et d'un paramètre nommé inflate factor (noté IF par la suite), l'algorithme découpe le graphe en clusters.
Un graphe est un objet mathématique composé de sommets (vertex) reliés par des arètes (edge). Une des représentations informatique consiste en une matrice carrée nommée matrice d'adjacence. Une cellule de cette matrice contient le poids de l'arète reliant deux sommets ou zéro en absence de lien.

On distingue les graphes orientés (directed graph) des graphes non orientés (undirected graph). Dans un graphe orienté, les arètes sont appelées arcs et ont un sens (de A vers B)

La matrice d'adjacence n'est donc plus obligatoirement symétrique et un choix arbitraire est fait concernant son interprétation: par exemple, les sommets sources correspondent aux lignes et les sommets destinations correspondent aux colonnes.

2. Principes et algorithme de MCl

Une bonne méthode de clustering maximisela similarité des objets appartenant à un même cluster (cohésion) la dissemblance entre objets appartenant à des clusters différents (séparation).

En ce qui concerne le partitionnement de graphes, une bonne méthode vise à ce que les clusters obtenus

aient un maximum d'arètes ou d'arcs reliant les sommets d'un même cluster aient un minimum d'arètes ou d'arcs reliant les sommets de clusters différents

Pour identifier les clusters, MCL se base sur la remarque suivante : si on place un marcheur aléatoire sur un sommet d'un cluster, il a plus de chances (en se promenant aléatoirement d'un sommet à un autre) de rester dans le cluster (plus grande densité de liens) que d'en sortir.

Le procédé s'appuie donc sur le calcul des probabilités de marches aléatoires (probabilité de passer par un sommet et probabilité d'emprunter un arc) dans un graphe donné. Le calcul s'effectue sur des matrices de Markov qui représentent les probabilités de transition d'un sommet à un autre ; il s'agit donc d'une matrice d'adjacence dont la somme des lignes vaut 1 (la somme des probabilités de sortie d'un sommet vaut 1).

L'algorithme MCL simule des marches aléatoires dans un graphe en alternant deux oprations appelées expansion et inflation. * L'expansion correspond au calcul de marches aléatoires de grande longueur et coincide à élever une matrice stochastique à une certaine puissance. Concrètement, l'expansion consiste à multiplier la matrice avec elle-même avec le produit matriciel. * L'inflation consiste à exagérer les probabilités de marches au sein d'un même cluster et à atténuer les probabilités de marches inter-clusters

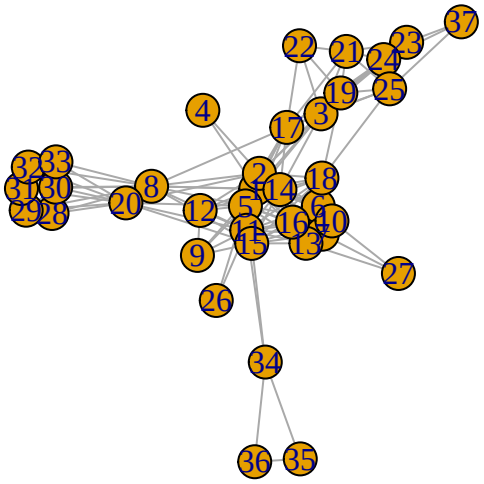
En pratique, l'inflation consiste à élever chaque cellule de la matrice à une certaine puissance (inflate factor) puis à normaliser les valeurs obtenues afin que la matrice soit stochastique.

Au final, à force de répéter les opération d'expansion et d'inflation sur la matrice de transition (et donc sur le graphe), celui-ci est décomposé en différentes composantes (composantes connexes) déconnectées les unes des autres et qui correspondent aux clusters. En d'autres termes, l'algorithme converge et la matrice n'évolue plus.

L'algorithme est donc le suivant :

M : matrice d'adjacence
ajouter les boucles à M
M_1 la matrice stochastique obtenue à partir de M
tant que on observe un changement dans la matrice
faire :
M_2 <- expansion(M_1)
M_2 <-inflation(M_2, inflate_factor)
déterminer s'il y a eu un changement
M_1 <- M_2

```
library(igraph)
# Acquisition des données (format edglist)
g= read.graph('http://silico.biotoul.fr/enseignement/m1/math/projet/toy.tgr',directed=FALSE)
#head(g)
plot(g) # Plot du graph de depart ...
title(main='Graph initial',cex.main=0.8, font.main=3,col.main="Red")
```



```
# Matrice d'adjacence du graph
M = get.adjacency(g)
M= as.matrix(M)
```

```
#Ajouter les boucles ... "
add_loops <- function(M){
  diag(M) = 1
  return(M)
}
```

```
#Rendre la matrice stochastique ==> calculer la matrice de markov
Make_stochastic <- function(M) {
  sum_row=apply(M,1,sum)
  M = M / sum_row
  return ( M )
}
```

```
#Operation d'expension
Expansion = function (M ) {
  M = M %*% M
  return (M)
}
```

```
#Operation d'inflation
inflation = function(M,inflate) {
  M = M^inflate
  M = M / apply(M,1,sum)
  return (M)
}
```

```
# Fonction Chaos, qui calcule si un changement a eu lieu ..
Chaos = function(M){
  max=apply(M,1,max) # max de chaque ligne
  sum_sq=apply( M^2, 1, sum ) # on élève chaque élément de la matrice(par ligne) au carré et on en fait
  row_chaos= max-sum_sq
  return(row_chaos)
}
```

```
# Fonction MCL
MCL= function(M,inflate) {
  M2=add_loops(M)
  M2=Make_stochastic(M2)
  change = 1
  while (change > 0.001) {
    M2=Expansion(M2)
    M2=inflation(M2,inflate)
    change=Chaos(M2)
  }
  return(round(M2,3))
}
```

Tests : Graphes avec différentes valeurs inflate : 2 , 4 , 6 , 8 , 10, 15, 20 , 100

```
# Ignore warnings ...
options(warn=-1)
```

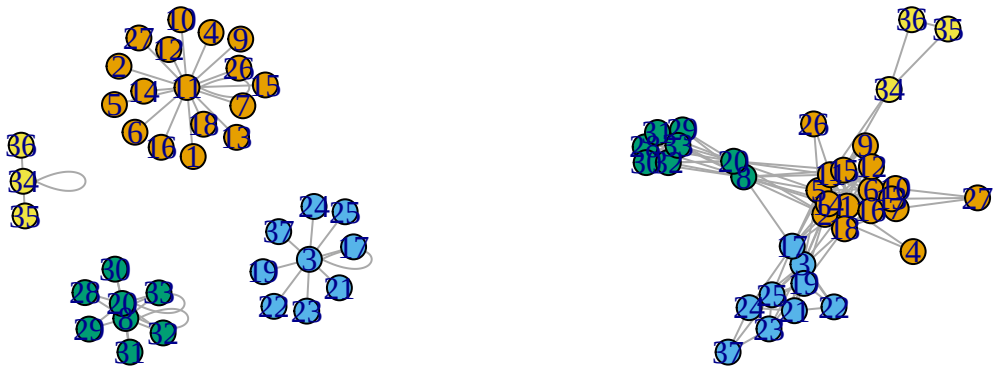
```r
#1. inflate = 2
m = MCL(as.matrix(get.adjacency(g)), inflate=2)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

```
## $membership
##  [1] 1 1 2 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 2 3 2 2 2 2 2 1 1 3 3 3 3 3 3 4 4
## [36] 4 2
##
## $csize
## [1] 17  9  8  3
##
## $no
## [1] 4
```

```r
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe

Clustering sur le graph initial, communautés

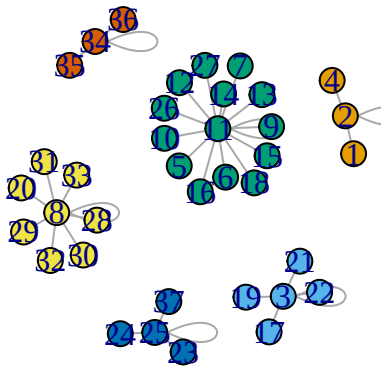


```r
#2. inflate = 4
m = MCL(as.matrix(get.adjacency(g)), inflate=4)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

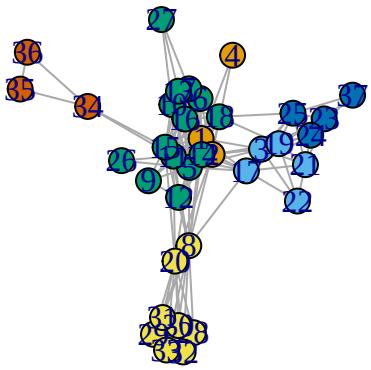
```
## $membership
##  [1] 1 1 2 1 3 3 3 4 3 3 3 3 3 3 3 3 2 3 2 4 2 2 5 5 5 3 3 4 4 4 4 4 4 6 6
## [36] 6 5
##
## $csize
## [1]  3  5 14  8  4  3
##
## $no
## [1] 6
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés



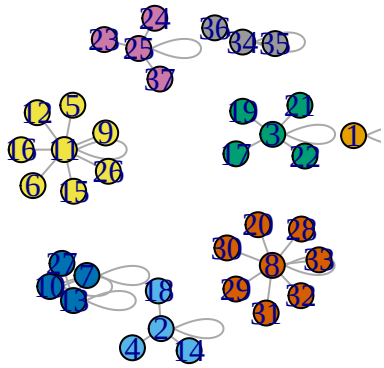
```
#3. inflate = 6
m = MCL(as.matrix(get.adjacency(g)), inflate=6)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

```
## $membership
##  [1] 1 2 3 2 4 4 5 6 4 5 4 4 5 2 4 4 3 2 3 6 3 3 7 7 7 4 5 6 6 6 6 6 6 8 8
## [36] 8 7
##
## $csize
## [1] 1 4 5 8 4 8 4 3
```

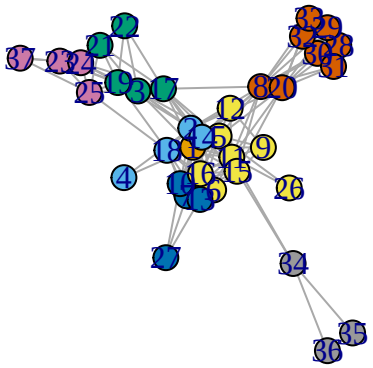
```
##
## $no
## [1] 8
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés



```
#4. inflate = 8
m = MCL(as.matrix(get.adjacency(g)), inflate=8)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

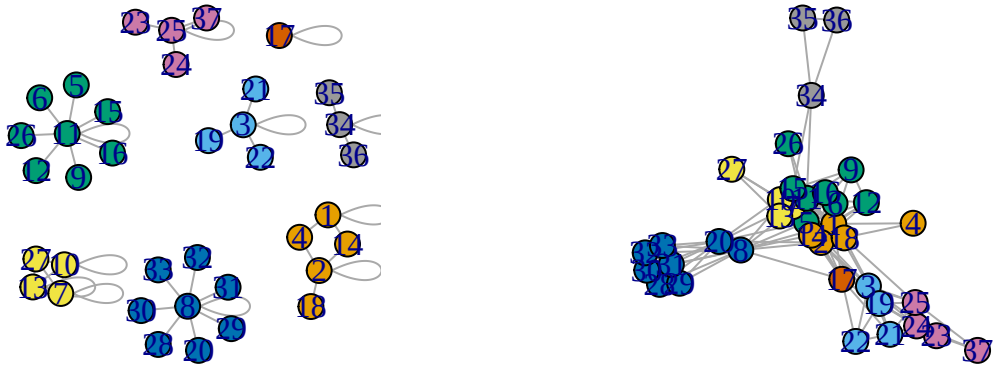
```
## $membership
##  [1] 1 1 2 1 3 3 4 5 3 4 3 3 4 1 3 3 6 1 2 5 2 2 7 7 7 3 4 5 5 5 5 5 5 8 8
## [36] 8 7
##
## $csize
## [1] 5 4 8 4 8 1 4 3
##
## $no
## [1] 8
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
```

```
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe

Clustering sur le graph initial, communautés

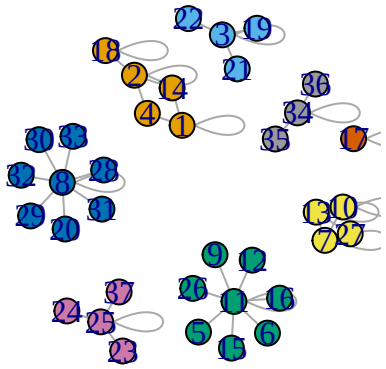


```
#5. inflate = 10
m = MCL(as.matrix(get.adjacency(g)), inflate=10)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

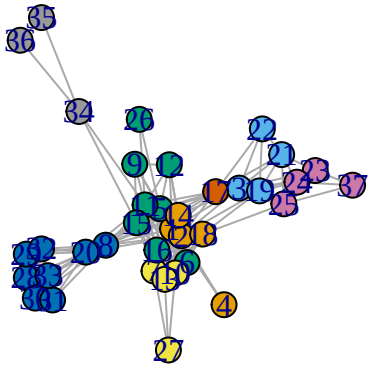
```
## $membership
##  [1] 1 1 2 1 3 3 4 5 3 4 3 3 4 1 3 3 6 1 2 5 2 2 7 7 7 3 4 5 5 5 5 5 8 8
## [36] 8 7
##
## $csize
## [1] 5 4 8 4 8 1 4 3
##
## $no
## [1] 8
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés

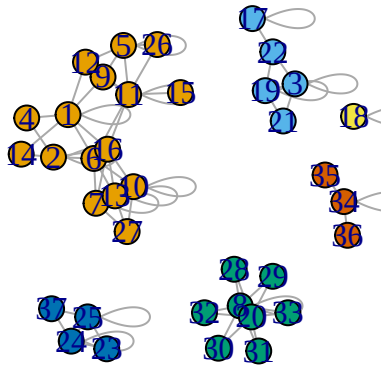


```
#6. inflate = 15
m = MCL(as.matrix(get.adjacency(g)), inflate=15)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

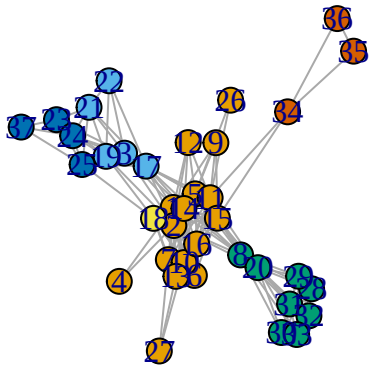
```
## $membership
##  [1] 1 1 2 1 1 1 1 3 1 1 1 1 1 1 1 1 2 4 2 3 2 2 5 5 5 1 1 3 3 3 3 3 3 6 6
## [36] 6 5
##
## $csize
## [1] 16  5  8  1  4  3
##
## $no
## [1] 6
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés

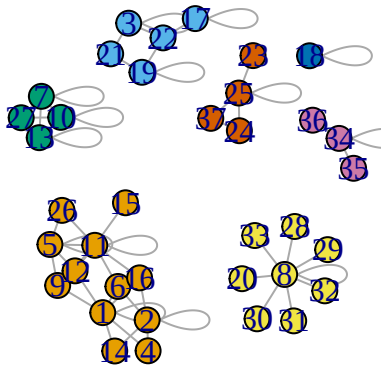


```
#7. inflate = 20
m = MCL(as.matrix(get.adjacency(g)), inflate=20)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

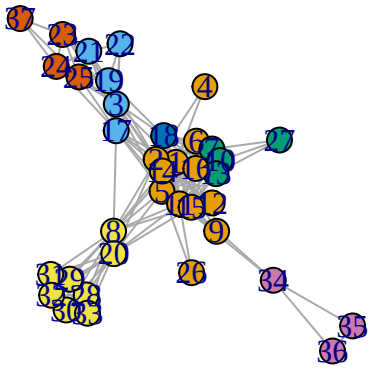
```
## $membership
##  [1] 1 1 2 1 1 1 3 4 1 3 1 1 3 1 1 1 2 5 2 4 2 2 6 6 6 1 3 4 4 4 4 4 4 7 7
## [36] 7 6
##
## $csize
## [1] 12  5  4  8  1  4  3
##
## $no
## [1] 7
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés

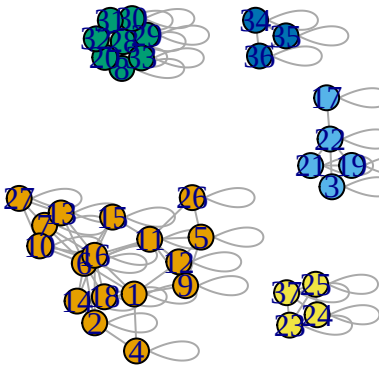


```
#8. inflate = 100
m = MCL(as.matrix(get.adjacency(g)), inflate=100)
g2=graph.adjacency(m, mode='undirected', weighted=T)
clusters(g2)
```

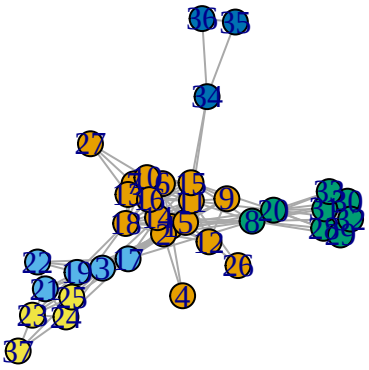
```
## $membership
##  [1] 1 1 2 1 1 1 1 3 1 1 1 1 1 1 1 1 2 1 2 3 2 2 4 4 4 1 1 3 3 3 3 3 3 5 5
## [36] 5 4
##
## $csize
## [1] 17  5  8  4  3
##
## $no
## [1] 5
```

```
par(mfrow=c(1,2))
plot(g2, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='composantes connexes du graphe',cex.main=0.8, font.main=3,col.main="Red")
plot(g, vertex.color=clusters(g2)$membership, edge.width=E(g)$weight)
title(main='Clustering sur le graph initial, communautés',cex.main=0.8, font.main=3,col.main="Red")
```

composantes connexes du graphe



Clustering sur le graph initial, communautés



R:

L'augmentation du inflate factor a pour effet d'augmenter le nombre de clusters .
Le seuil minimal pour le inflate factor est : 2 (inflate > 1)