Détection de communautés

Graphes – 2e partie

Raphaël Fournier-S'niehotta

CNAM Paris, fournier@cnam.fr

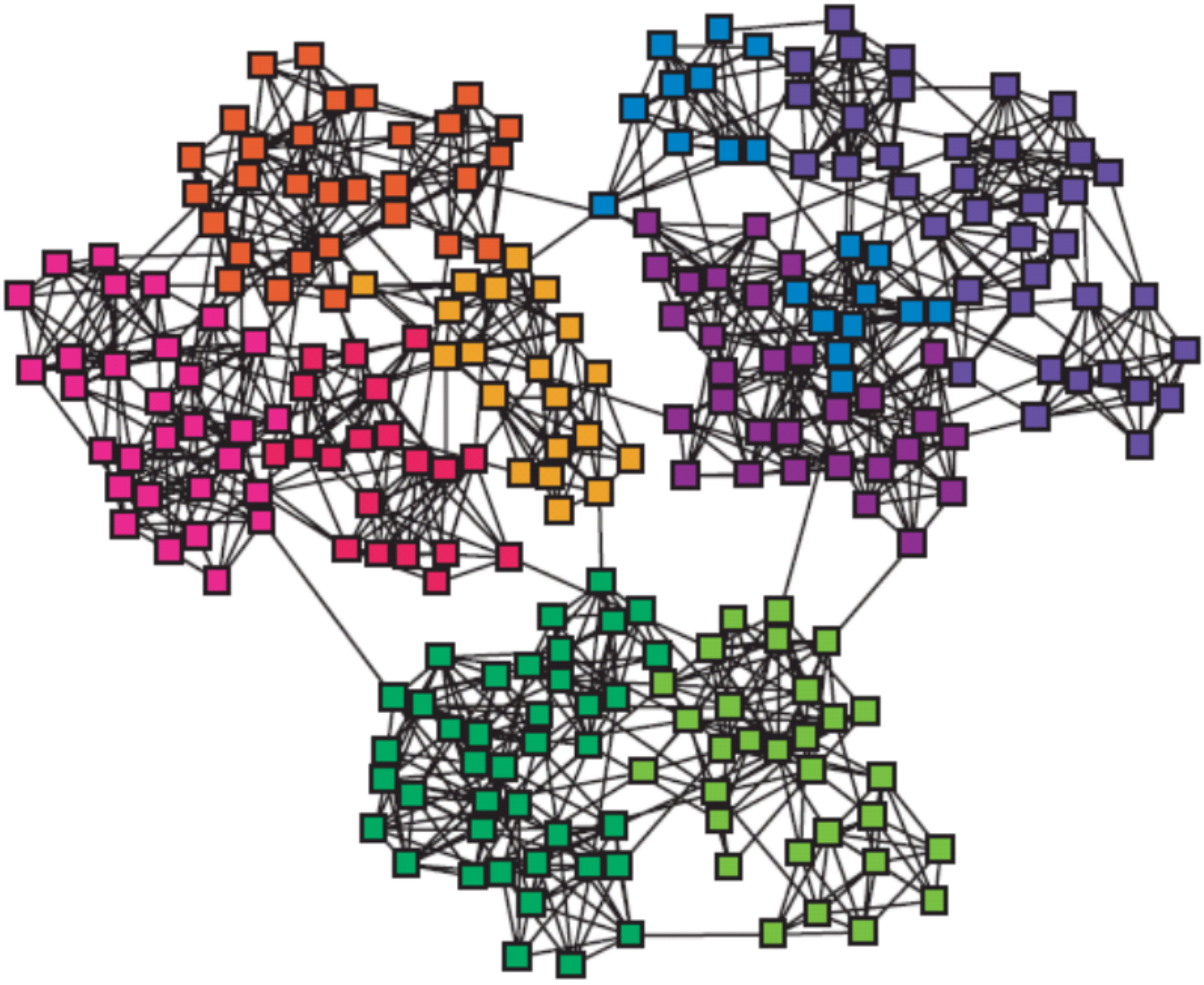
HTT-FOD
RCP216
2020-2021

le cnam

Plan

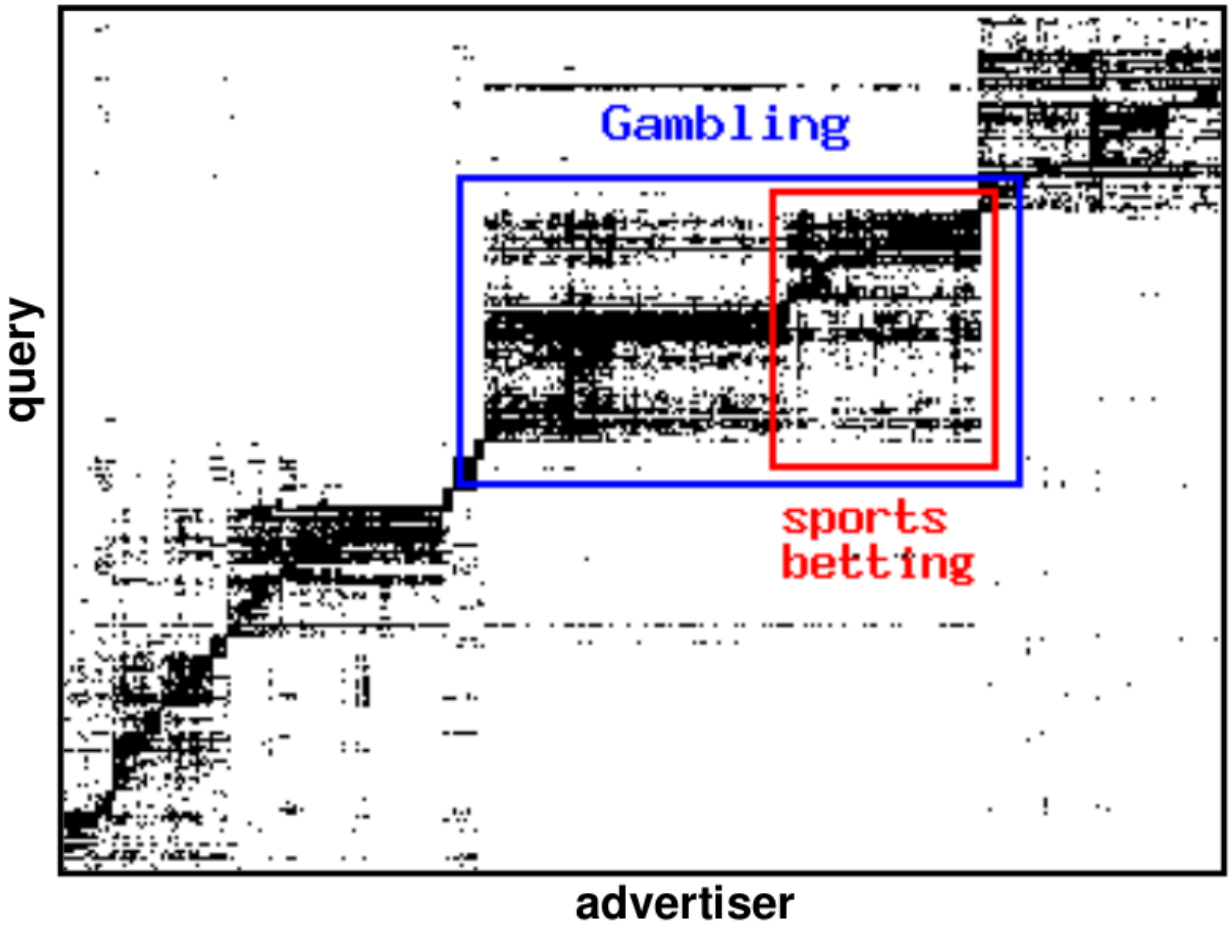
Détection de communautés

Graphes et communautés



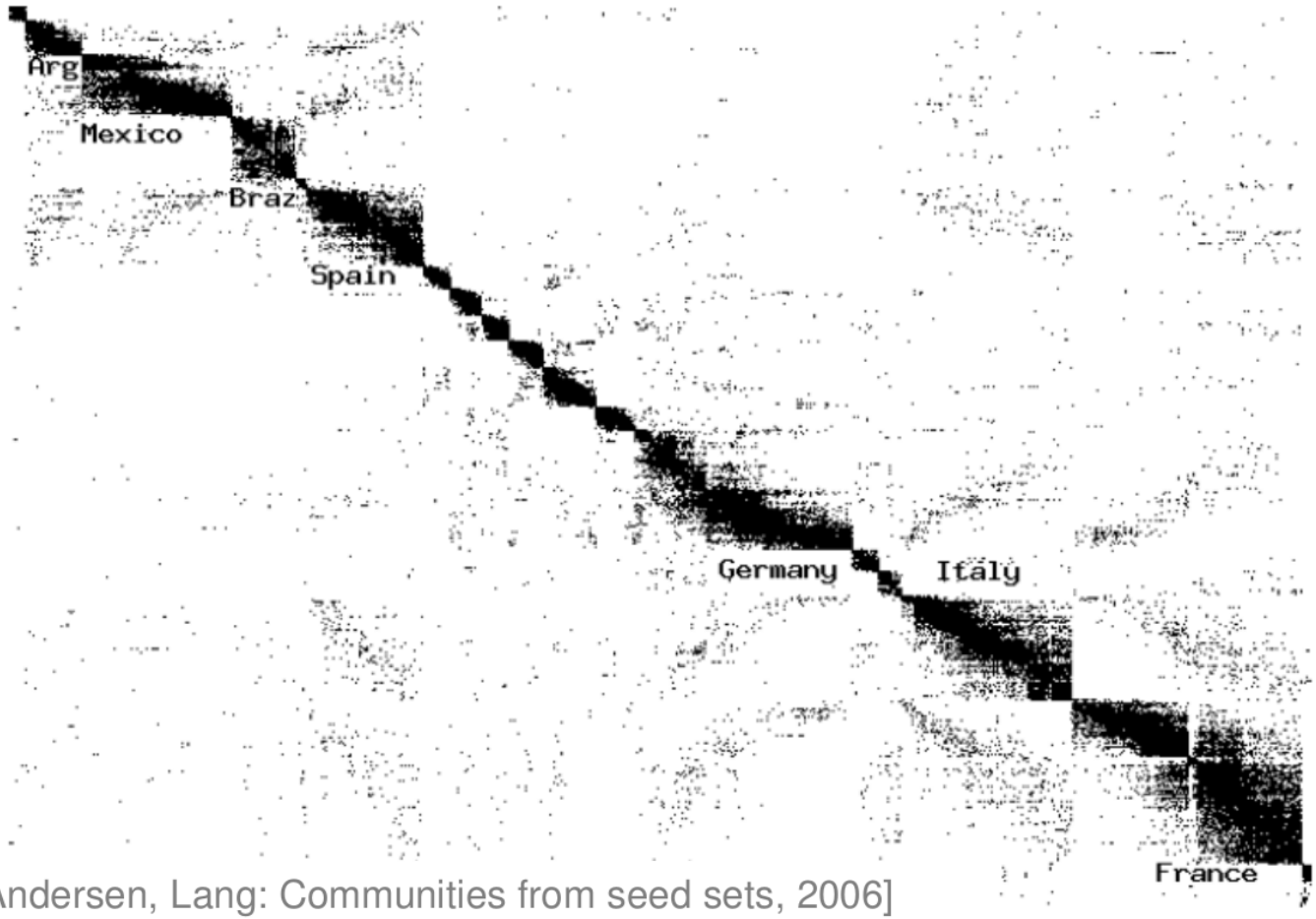
Exemple de communautés

Requête / publicité



[Andersen, Lang: Communities from seed sets, 2006]

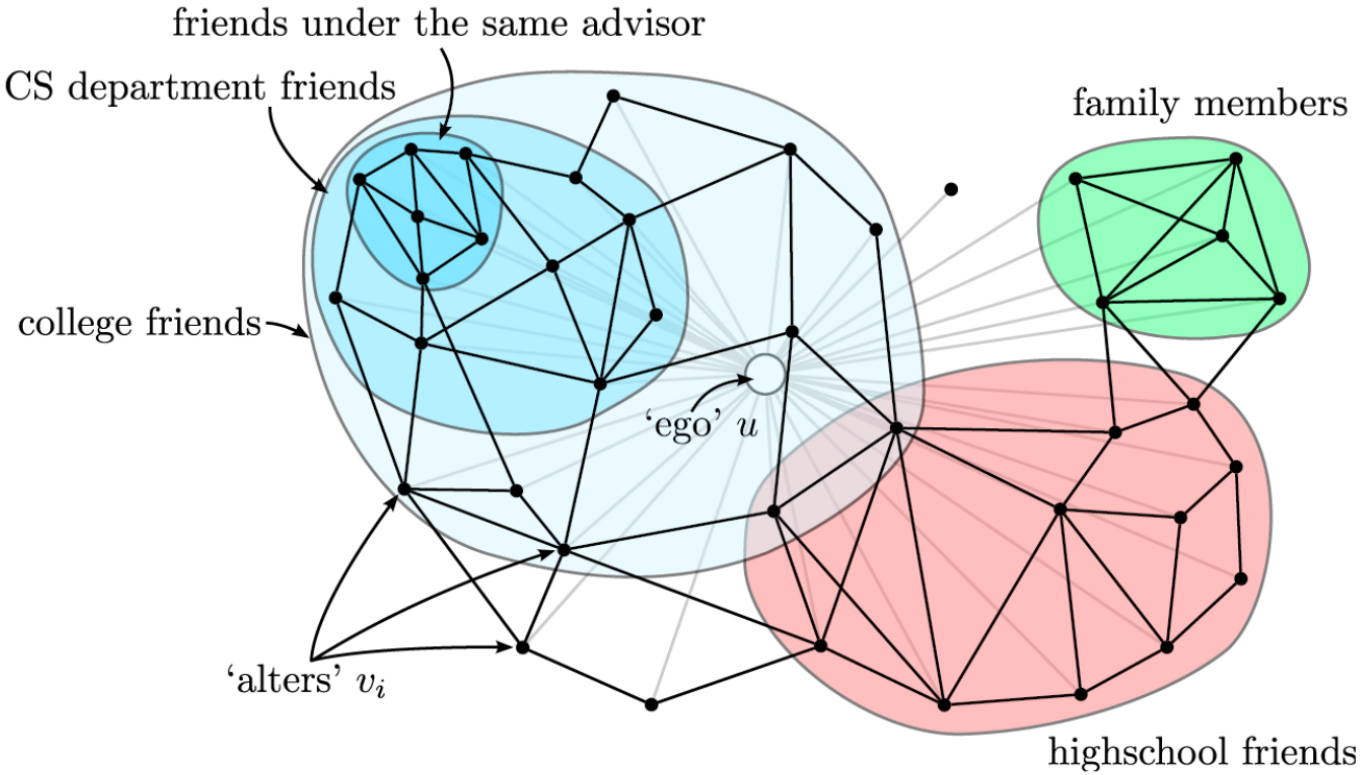
Exemple de communautés



[Andersen, Lang: Communities from seed sets, 2006]

Exemple

Réseau social (communautés recouvrantes !)



[McAuley, Leskovec: Discovering social circles in ego networks, 2012]

Détection de communautés

- Objectif : identifier automatiquement des groupes pertinents
- Applications
  - Identifier automatiquement des groupes pertinents.
  - Comprendre la structure des réseaux.
  - Détecter des communautés spécifiques (Pages web, influenceurs politiques, groupes d'amis, ...)
  - Visualisation.
  - Amélioration de moteurs de recherche, systèmes P2P, ...
- Challenges
  - Nombre de communautés inconnu.
  - Communautés de taille variable.
  - Passage à l'échelle : milliards de sommets.

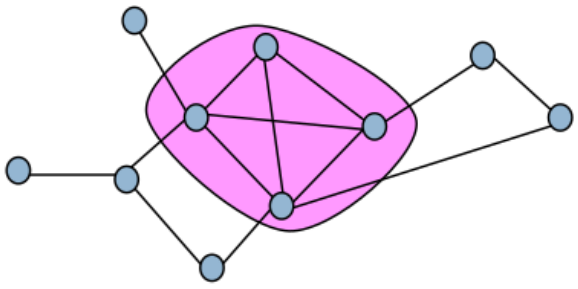
Qu'est-ce qu'une communauté ?

- Un ensemble de sommets d'un graphe qui partagent quelque chose :
  - Amis, collègues, ...
  - Personnes avec des intérêts similaires.
  - Pages web avec un même contenu.
  - etc.

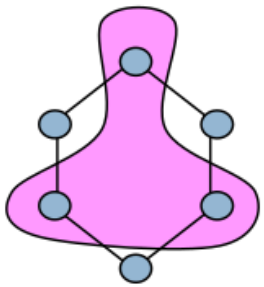
  - On peut donc chercher le lien avec la structure du réseau
  - Il existe de nombreuses manières de caractériser des sous-graphes intéressants

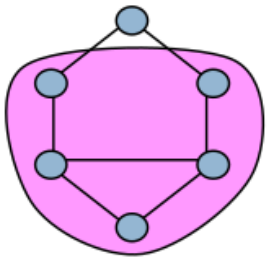
Sous-graphes cohésifs

- Composantes connexes ou k-connexes :
  - Au moins (1) k chemins disjoints entre chaque paire de sommets
- Cliques :
  - sous-graphe complètement connecté



- n-cliques
  - Sous-graphe G' avec distance inférieure ou égale à n dans G.
  - peut être déconnecté ou de diamètre $> n$ (exemples avec $n = 2$) :





Sous-graphes cohésifs (suite)

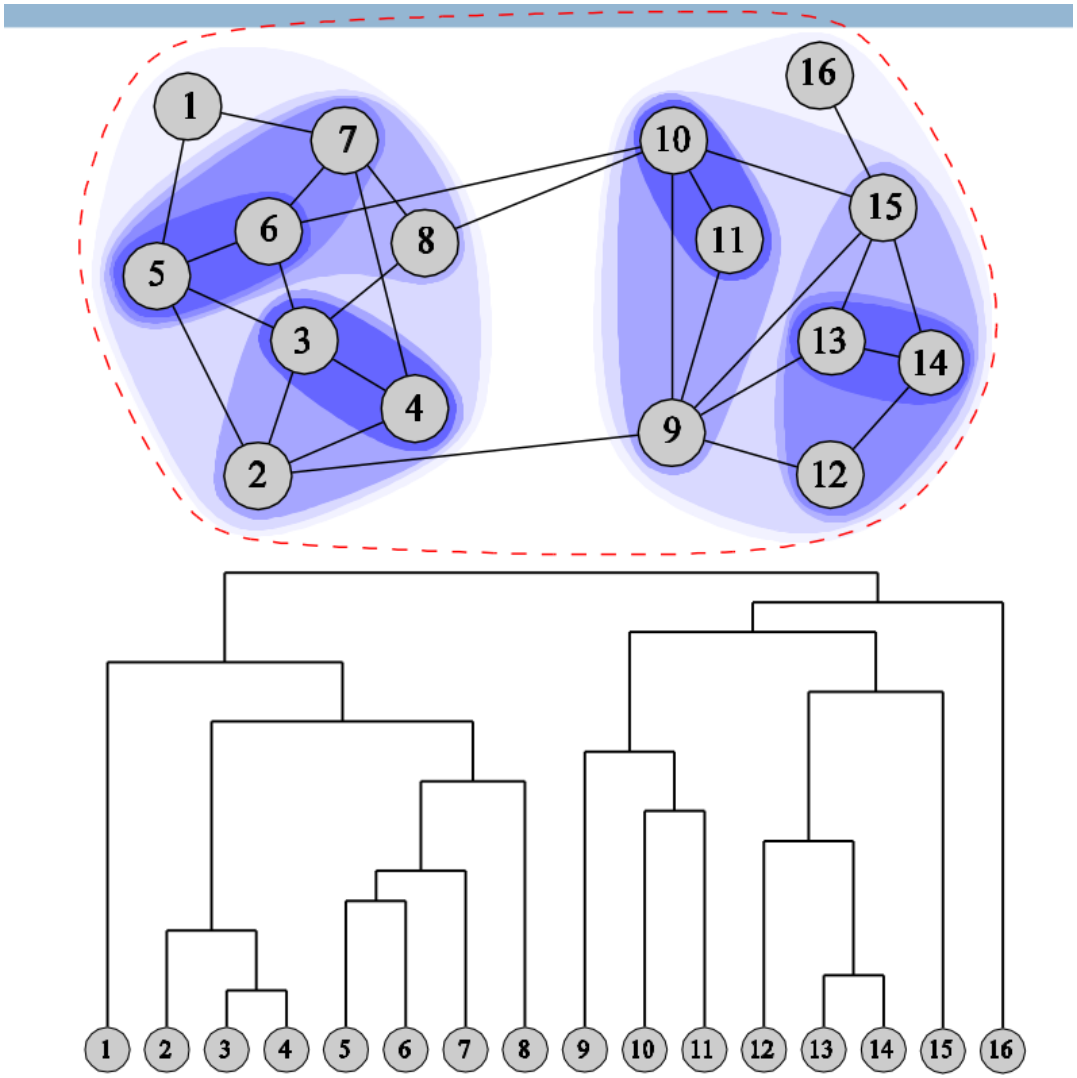
- k-degree set : Sous-graphe dans lequel tous les sommets ont degré au moins k.
- k-outdegree set : Sous-graphe tel qu'aucun sommet n'a plus de k liens sortants.
- k-plex : Sous-graphe maximal H t.q. chaque sommet est connecté à au moins $|H| - k$ sommets
- LS set : Sous-graphe minimal pour le nombre de liens sortants.
- alpha set : Sous-graphe t.q. chaque sommet a plus de liens internes que sortants.

Calcul

- La recherche de ces zones sur un graphe donné est en général un problème coûteux
- Complexité
  - Souvent NP-complet : Cliques (Bron Kerbosch), k-plex, ...
  - Parfois polynomial : LS, Lambda sets (n^4 ou moins, ...)
  - Passage à l'échelle : *web* $\sim 10^{10}$ sommets !
- Communautés recouvrantes
  - Un sommet peut être dans plusieurs communautés.
  - Certaines définitions sont recouvrantes : Clique, ...
  - Ou pas : K-cores

On doit trouver d'autres approches

Clustering hiérarchique

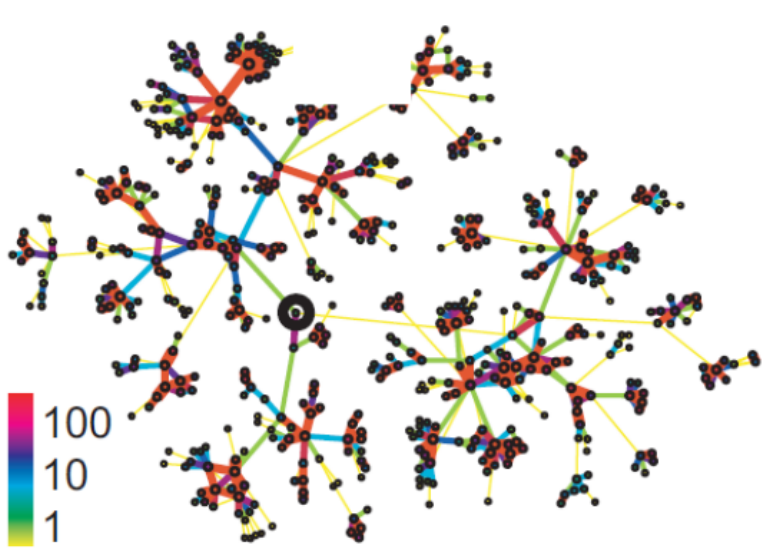


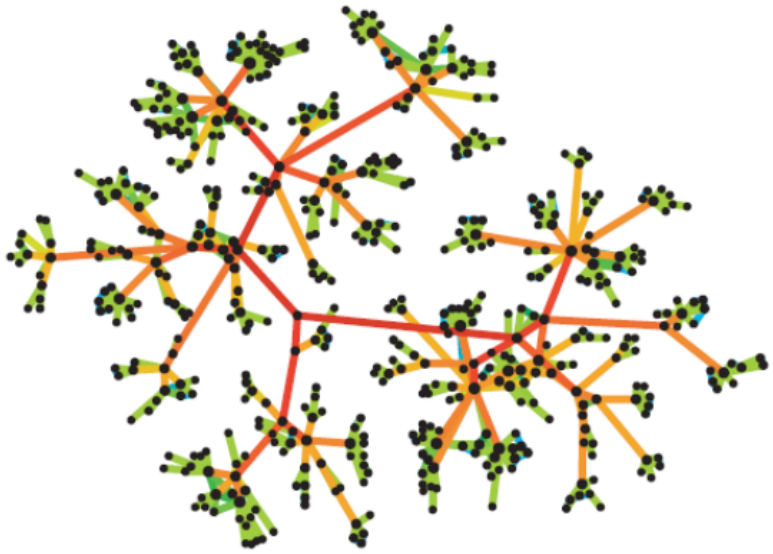
Clustering hiérarchique

- Algorithme générique
  1. Chaque sommet est dans une communauté.
  2. Calculer une distance entre chaque paire de communautés.
  3. Fusionner les deux plus proches.
  4. Revenir à 2.
- Quelle distance prendre entre deux sommets ? longueur des chemins ?
- Pour les distances entre communautés : Min, max, moyenne des distances entres paires de sommets

Centralité et clustering hiérarchique “divisif”

- On calcule la centralité de chaque lien : c’est le nombre de plus courts chemins passant par ce lien
- Plus elle est élevée, plus on suppose que le lien n’est pas “au sein d’une communauté”, mais plutôt “entre des communautés différentes”
- intuition :

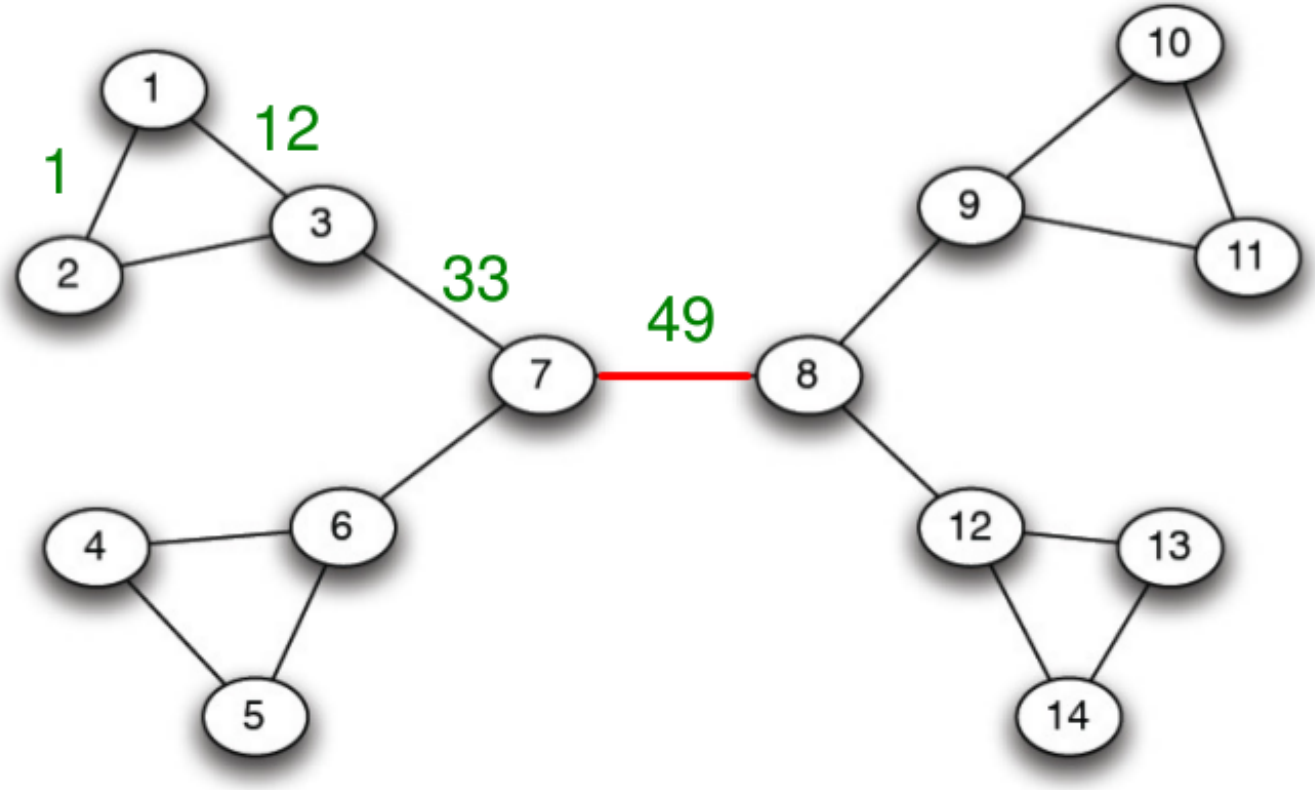




Algorithme de Girvan Newman

- Clustering hierarchique avec approche divisive
- On calcule la centralité (*betweenness*) de chaque lien
- On enlève le lien de plus forte centralité
- On recommence jusqu'à ce qu'il n'y ait plus de lien
- les composantes connexes restantes sont les communautés
- On obtient une décomposition hiérarchique du réseau

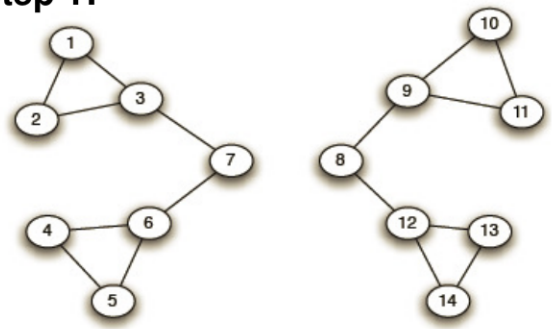
Girvan Newman



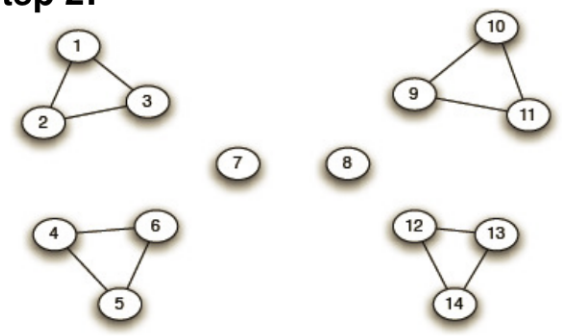
- On doit recalculer la centralité à chaque étape

Girvan Newman

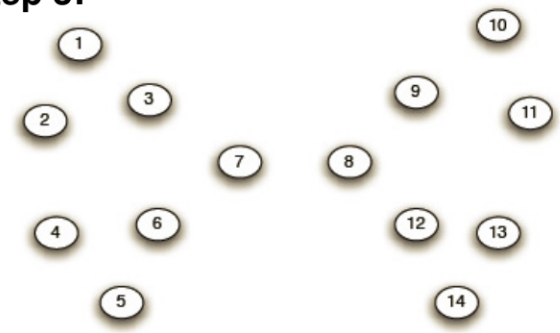
Step 1:



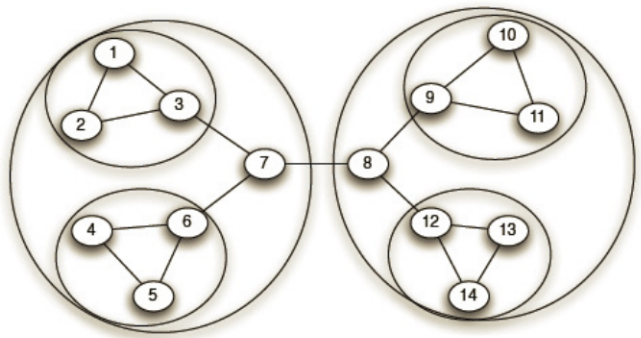
Step 2:



Step 3:



Hierarchical network decomposition:

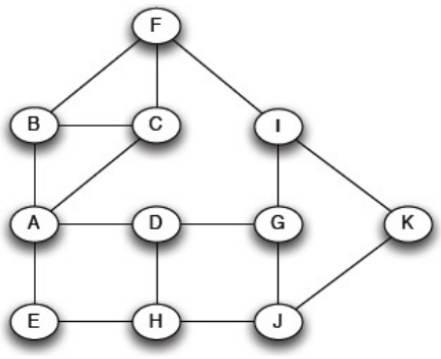


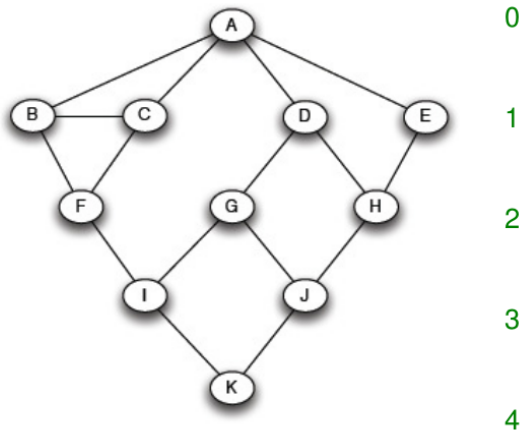
2 questions

- Comment calculer la centralité ?
- Comment choisir le nombre de clusters ?

Calcul de la centralité

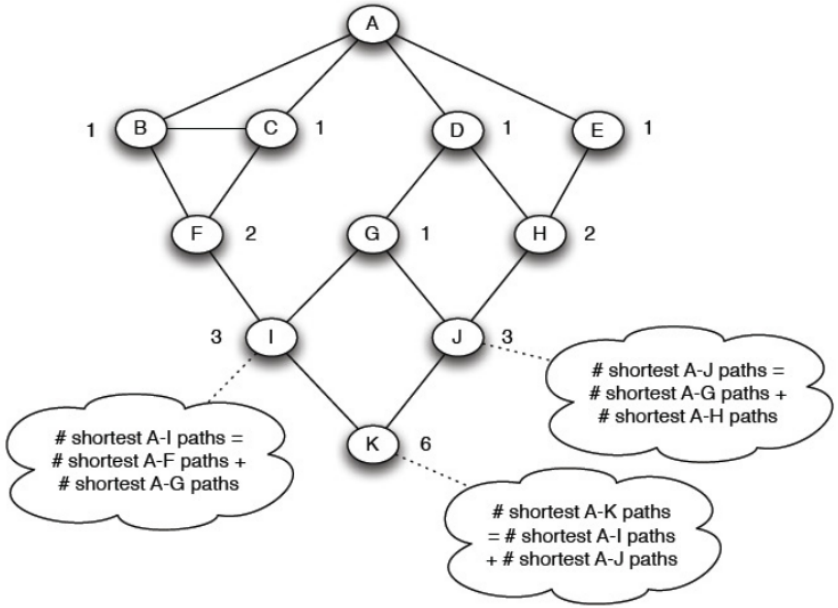
- On veut calculer la centralité des chemins commençant à A
- On fait un parcours en largeur à partir de A





Calcul de la centralité

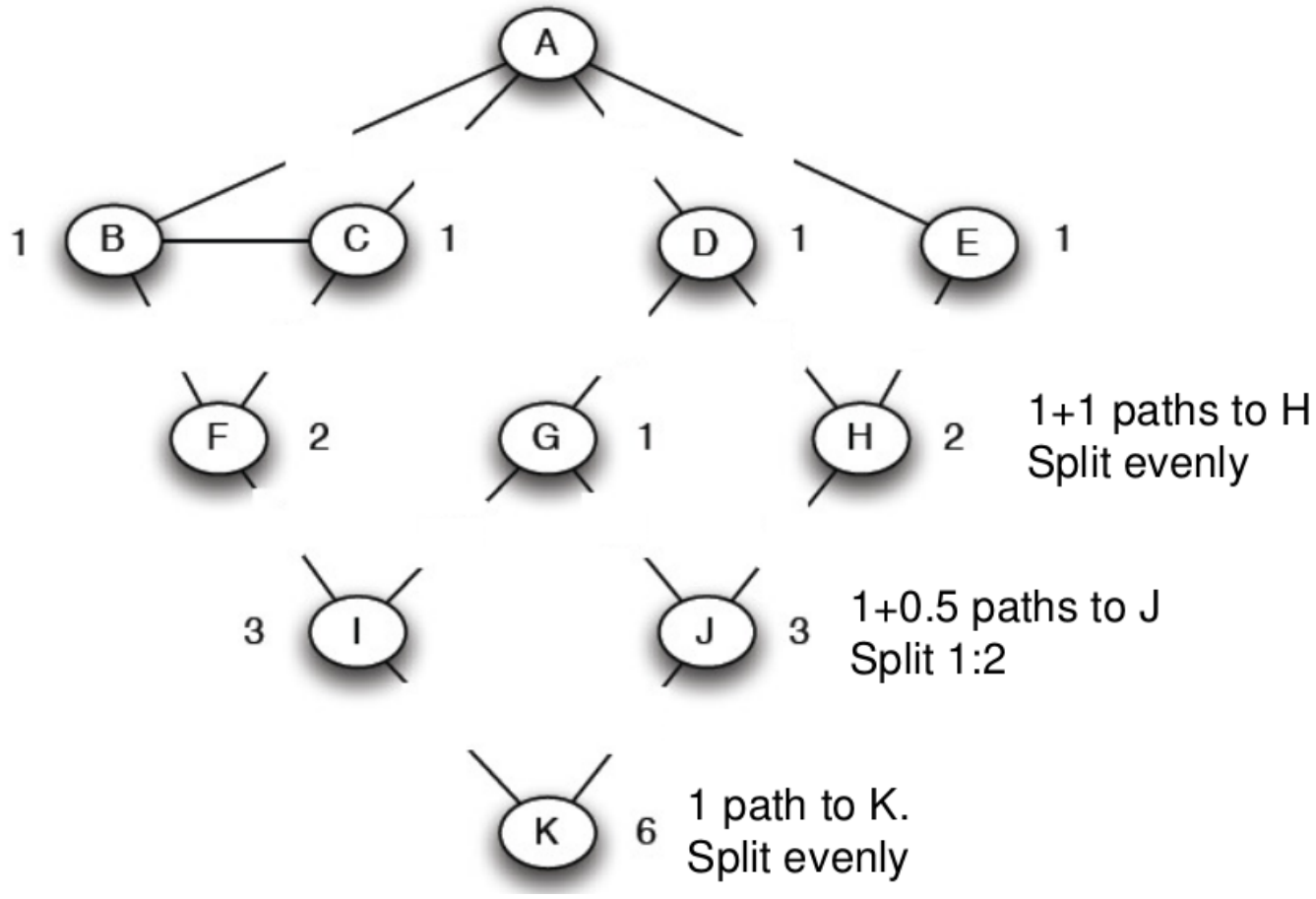
- la racine reçoit 1
- On donne à chaque nœud la somme des valeurs de ses parents
- (c'est le nombre de plus courts chemins arrivant à ce nœud)



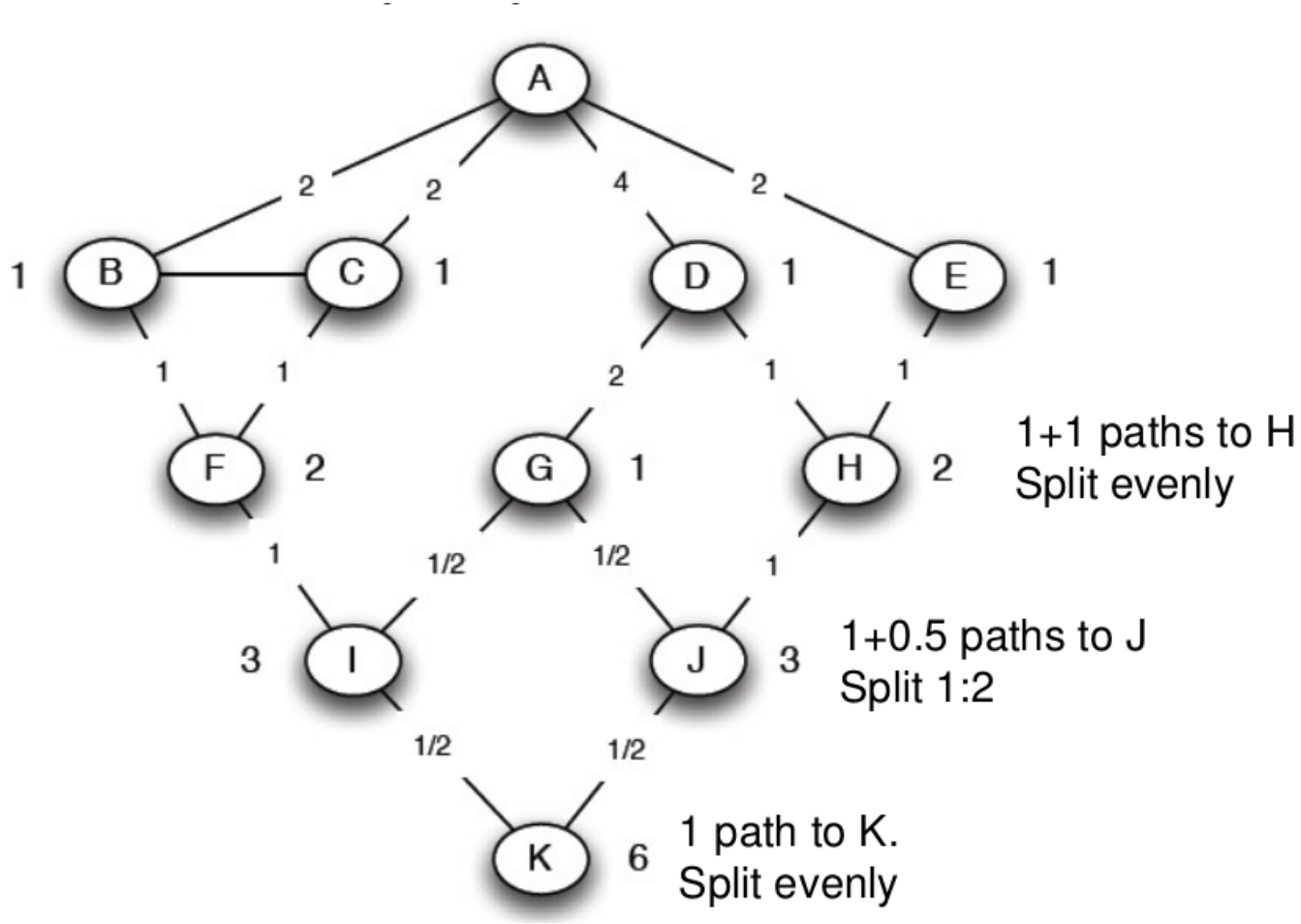
Calcul de la centralité

- Ensuite, en **remontant** l'arbre, on calcule la centralité
- chaque nœud reçoit 1 si c'est une feuille
- le "flot" d'un nœud vaut 1+$\sum$ "liens enfants"
- on le divise en fonction des "valeurs" des parents

Calcul


A
1
B
C
1
D
1
E
1
F
2
G
1
H
2
1+1 paths to H
Split evenly
3
I
J
3
1+0.5 paths to J
Split 1:2
K
6
1 path to K.
Split evenly

Calcul


A
2
2
4
2
1
B
C
1
D
1
E
1
1
1
2
1
1
F
2
G
1
H
2
1+1 paths to H
Split evenly
1
1/2
1/2
1
3
I
J
3
1+0.5 paths to J
Split 1:2
1/2
1/2
K
6
1 path to K.
Split evenly

Évaluer la structure communautaire

- Définition : Une communauté est un sous-graphe dont les sommets sont plus liés entre eux qu'avec le reste du réseau.
- Modularité : une mesure pour comparer le nombre de liens dans un groupe de sommets à ce que l'on attendrait pour un graphe aléatoire similaire.
- Différence entre nombre de liens dans un module et nombre attendu de liens dans un graphe similaire

Modularité Q

- On cherche à partitionner notre graphes en groupes $s \in S$
- On veut une mesure évaluant la qualité de ce partitionnement (est-ce qu'on a "cassé" des communautés)
- $$Q \propto \sum_{s \in S} \#\text{ liens dans s} - \#\text{ liens attendu dans s}$$
- il faut un modèle aléatoire pour comparer

Modèle configurationnel

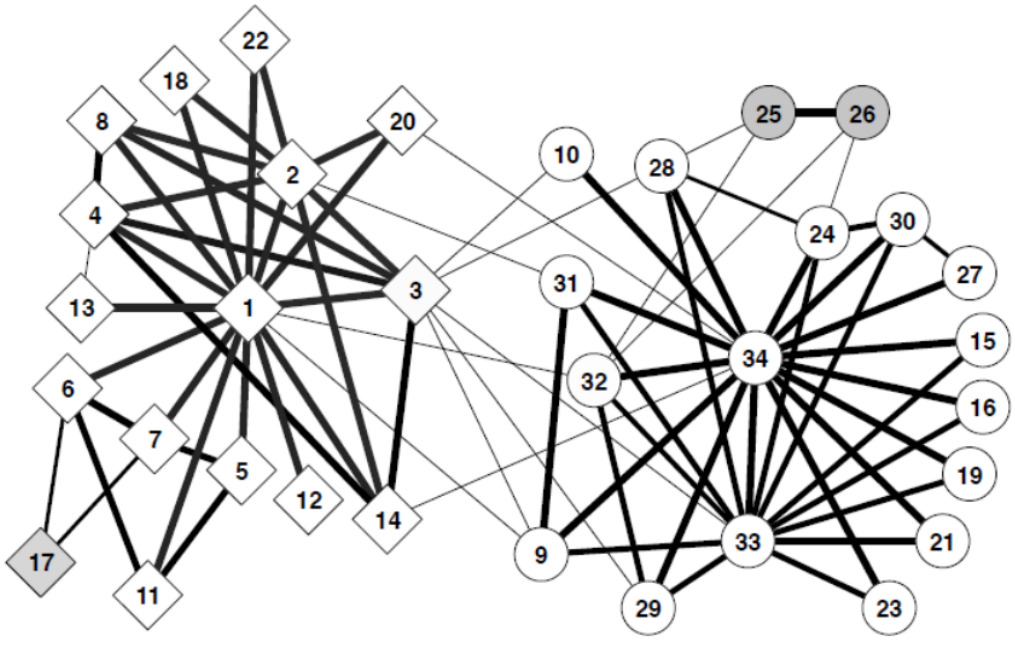
- À partir de $G(n, m)$, on construit un multigraphe G', en recablant les liens
- Même distribution de degrés
- Mais connexion aléatoire
- Le nombre de liens attendus entre les nœud i et j, de degrés k_i et k_j est : $\frac{k_i \times k_j}{2m}$

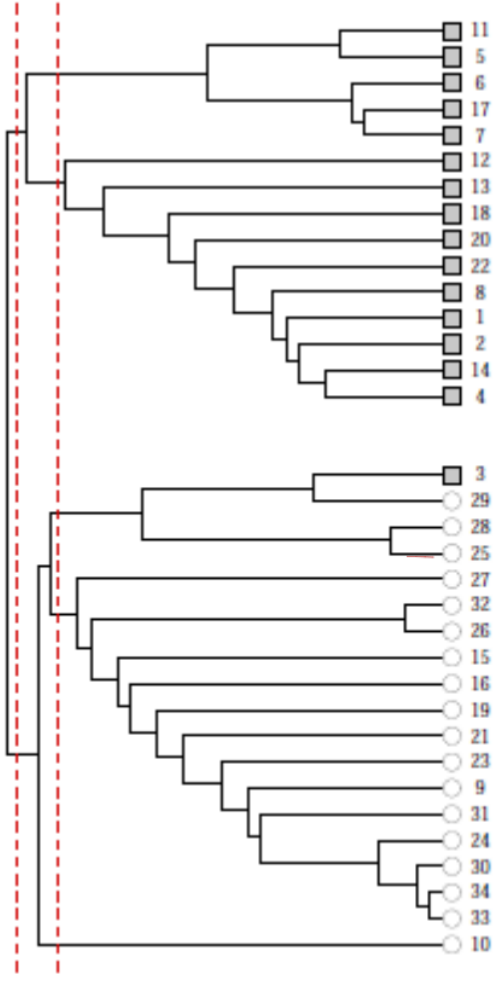
Modularité

- La modularité de la partition S de G vaut donc :
  - $Q(G,S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in S} \sum_{j \in S} [A_{ij} - \frac{k_i k_j}{2}]$
  - avec $A_{ij} = 1$ si le lien $i \to j$ existe, 0 sinon
- le premier facteur sert à normaliser, de façon que Q prenne des valeurs dans $[-1;1]$
- la modularité est positive si le nombre de nœud dans les groupes est supérieur à ce qu'on pourrait attendre
- Q au-dessus de 0.3/0.7 indique qu'on a une structure communautaire significative

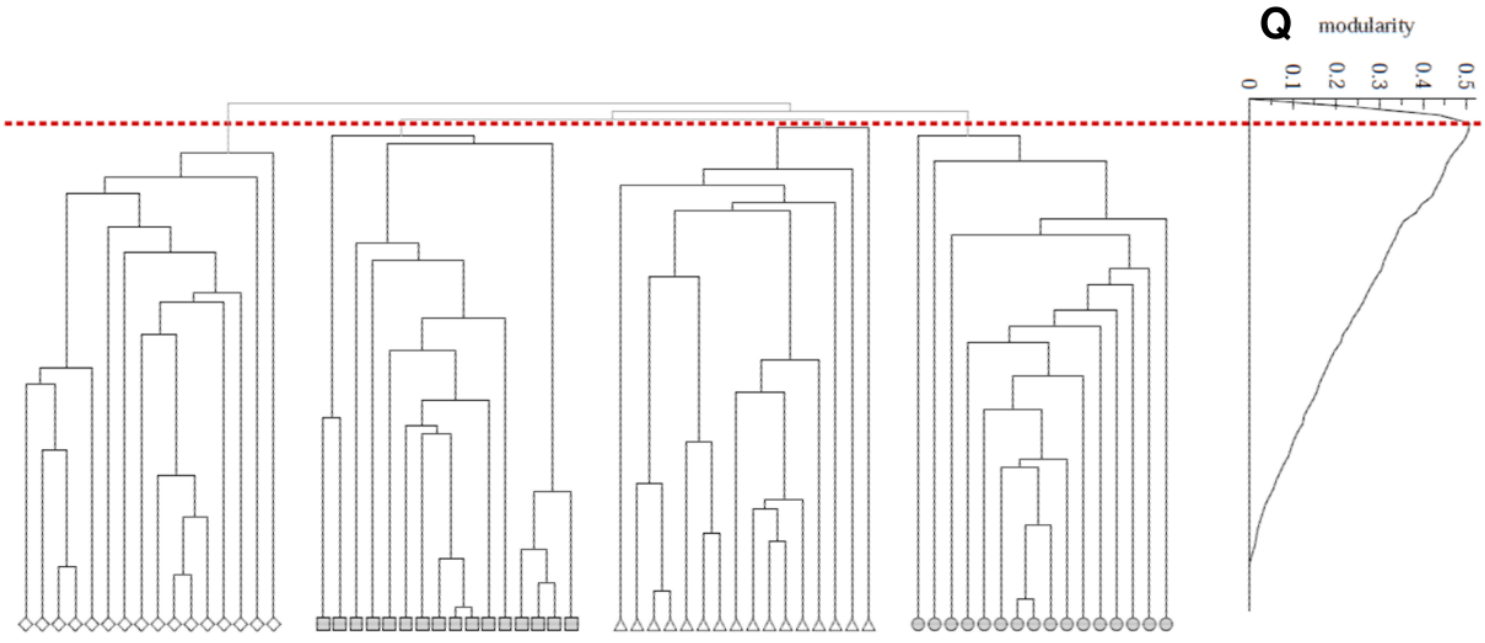
Karaté club

- Le graphe du club de Karaté étudié par W.Zachary (1972)





Sélectionner avec la modularité



Il peut être intéressant d'optimiser la modularité autrement...

L'algorithme de Louvain

- Travail et slides de Jean-Loup Guillaume (U. La Rochelle)

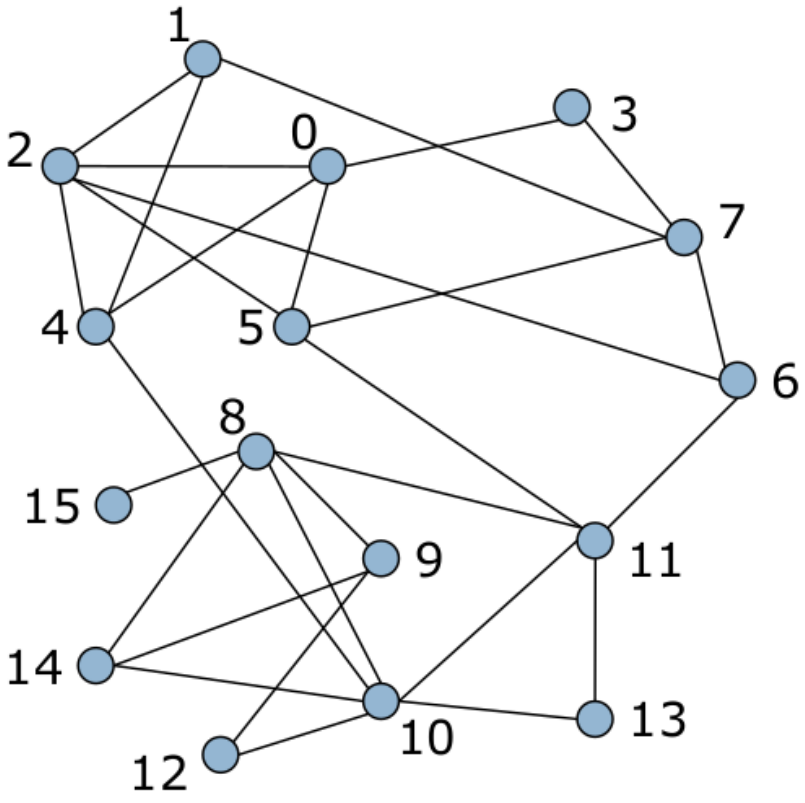
L'algorithme, formellement

- Suite de passes :
  - Chacune calcule un niveau de la hiérarchie.
  - Entrée : graphe (pondéré).
  - Sortie : graphe pondéré dont les sommets sont les communautés du graphe en entrée.
- les passes sont appliquées récursivement
- Arrête quand la modularité ne peut plus être améliorée

L'algorithme, formellement

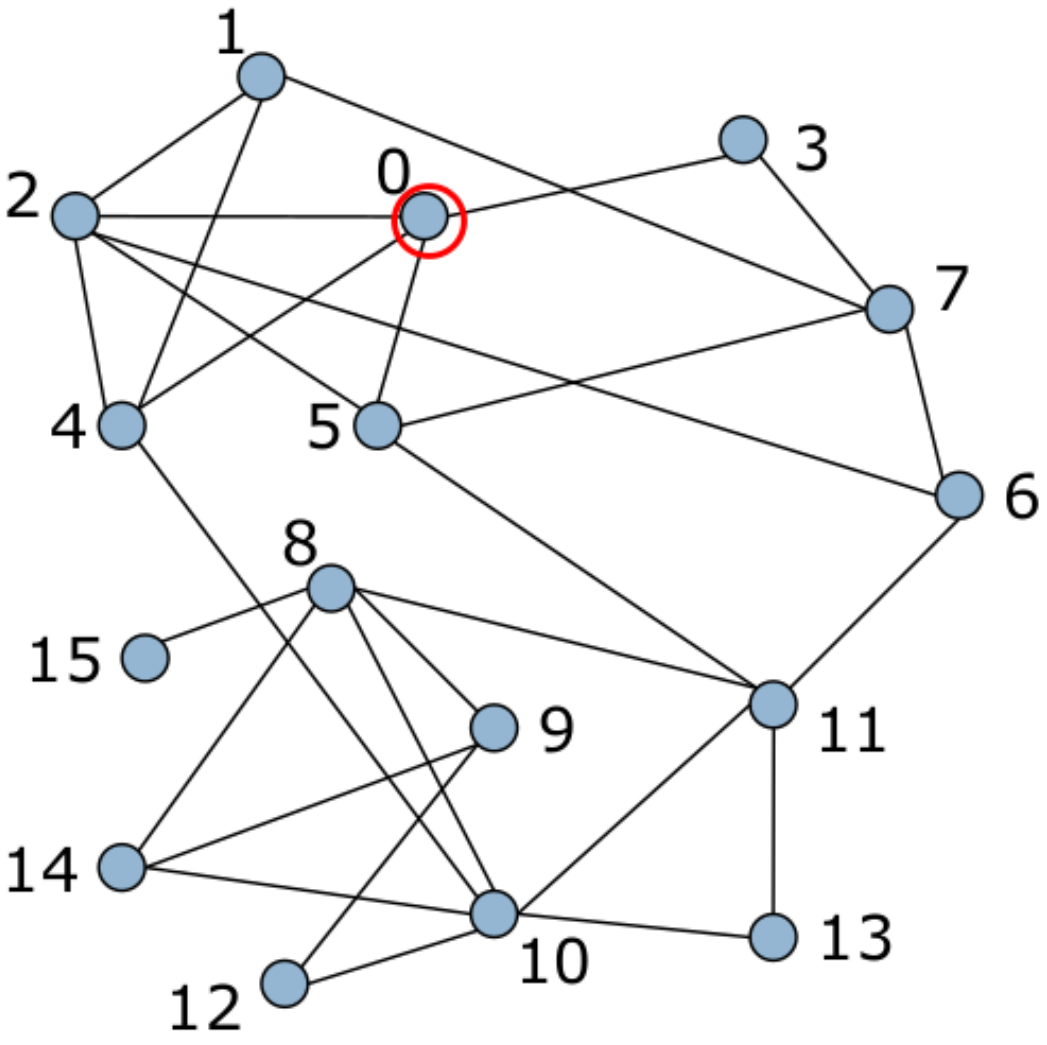
- Chaque passe :
  - Au début chaque sommet forme une communauté.
  - Répéter de manière itérative :
    - Supprimer i de sa communauté.
    - Insérer i dans la communauté voisine qui maximise la modularité (approche locale gloutonne).
  - Arrêter quand un maximum local est atteint

L'algorithme de Louvain

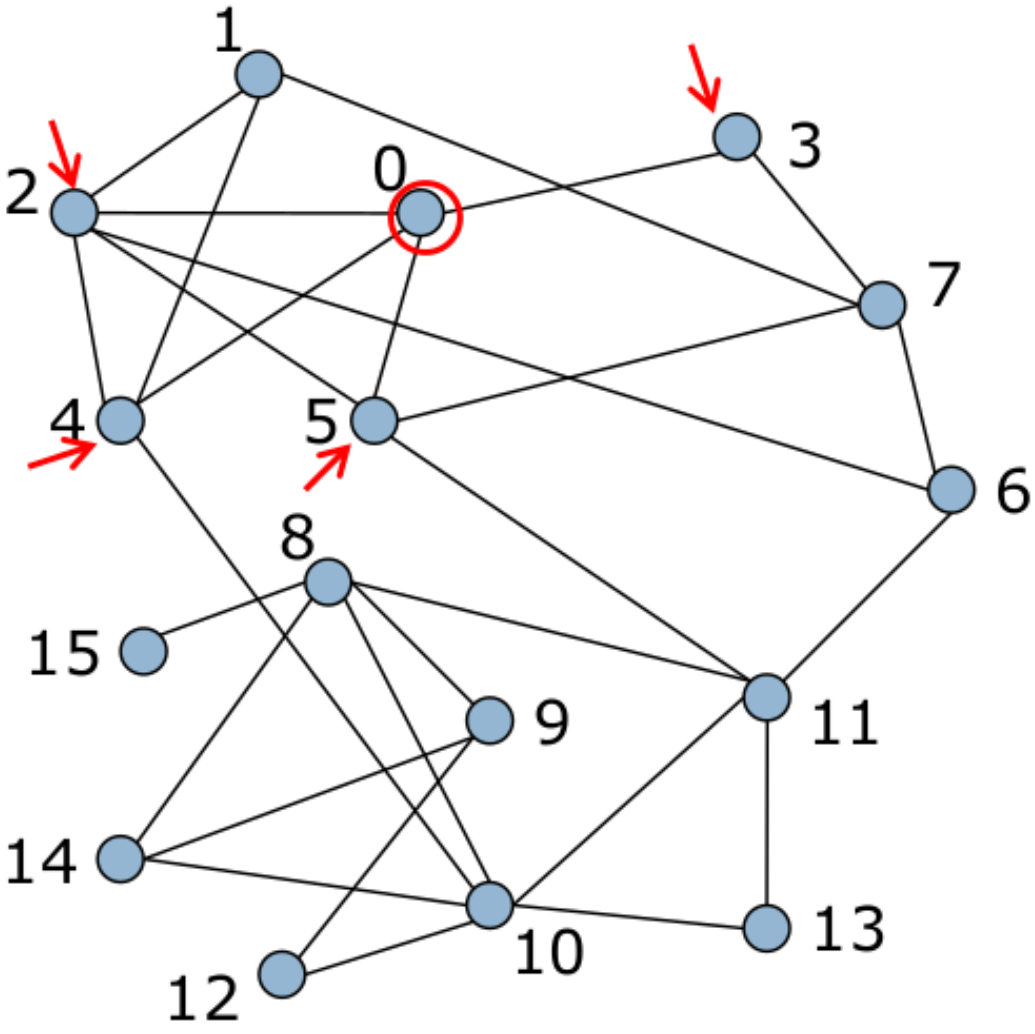


Passe 1 – Itération 1
chaque sommet est isolé

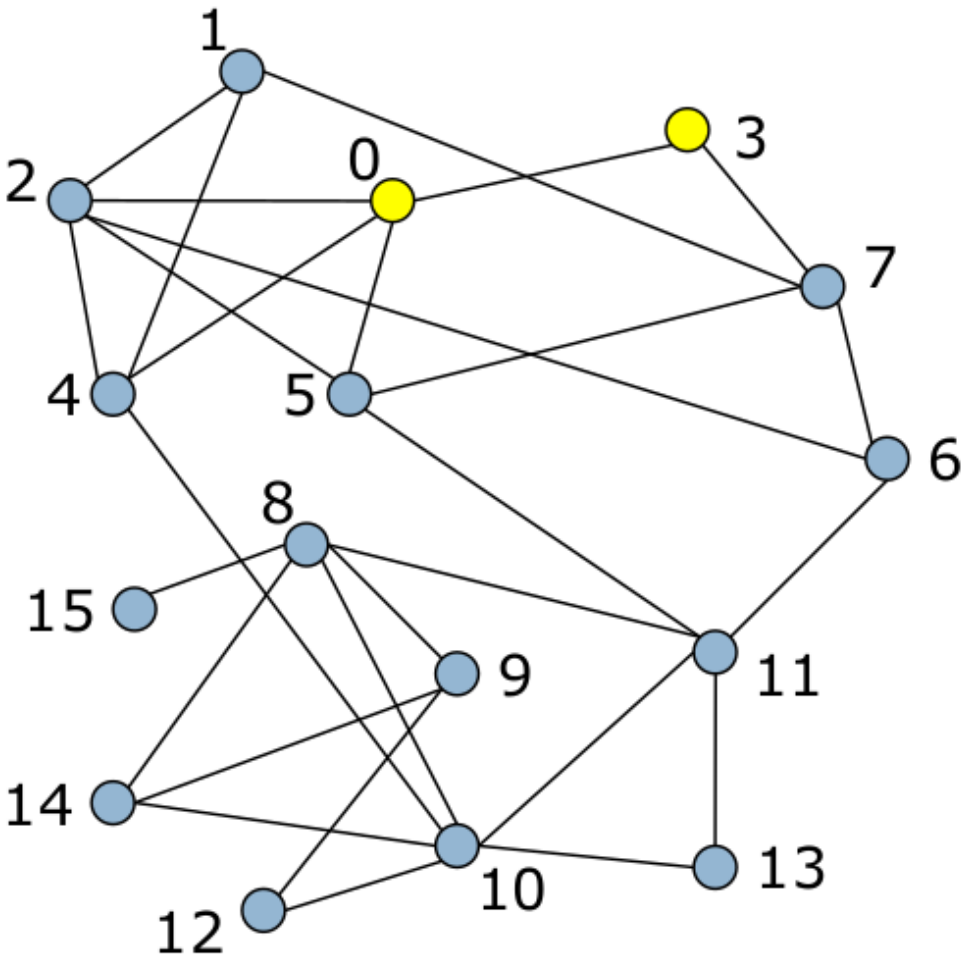
L'algorithme de Louvain



L'algorithme de Louvain

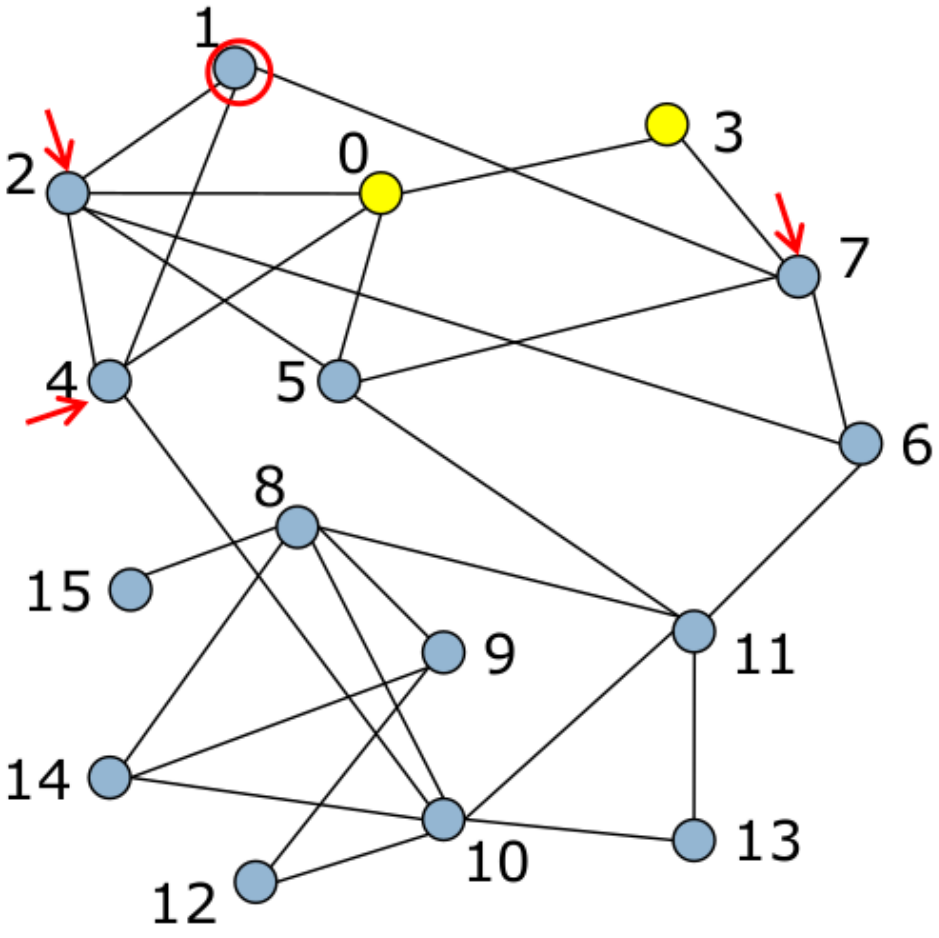


L'algorithme de Louvain



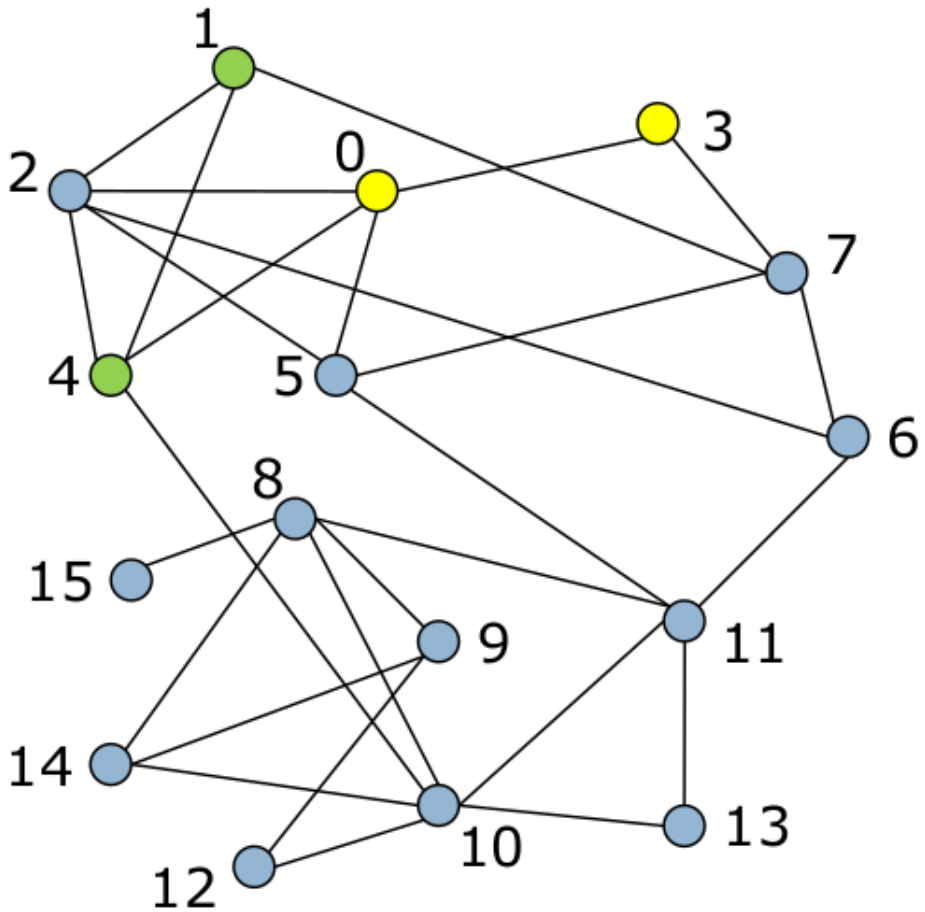
0 -> c[3]

L'algorithme de Louvain



0 -> c[3]

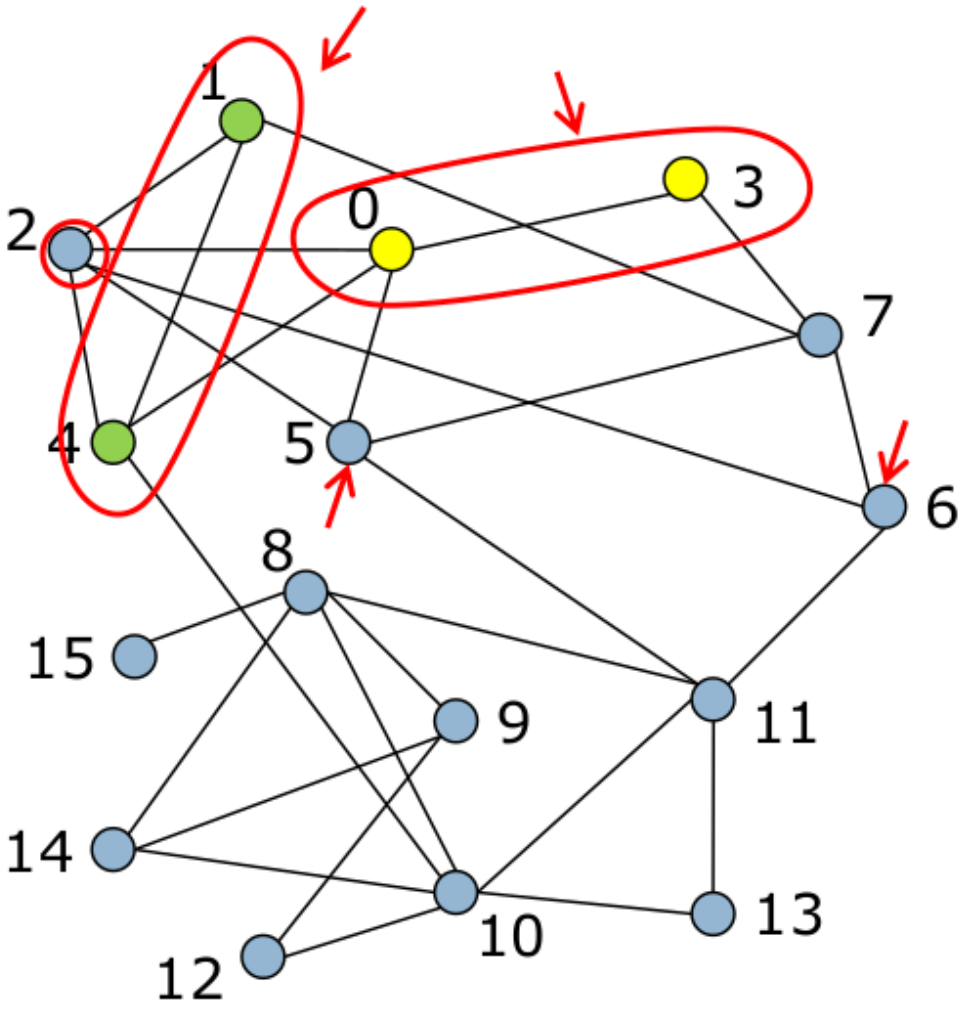
L'algorithme de Louvain



0 -> c[3]

1 -> c[4]

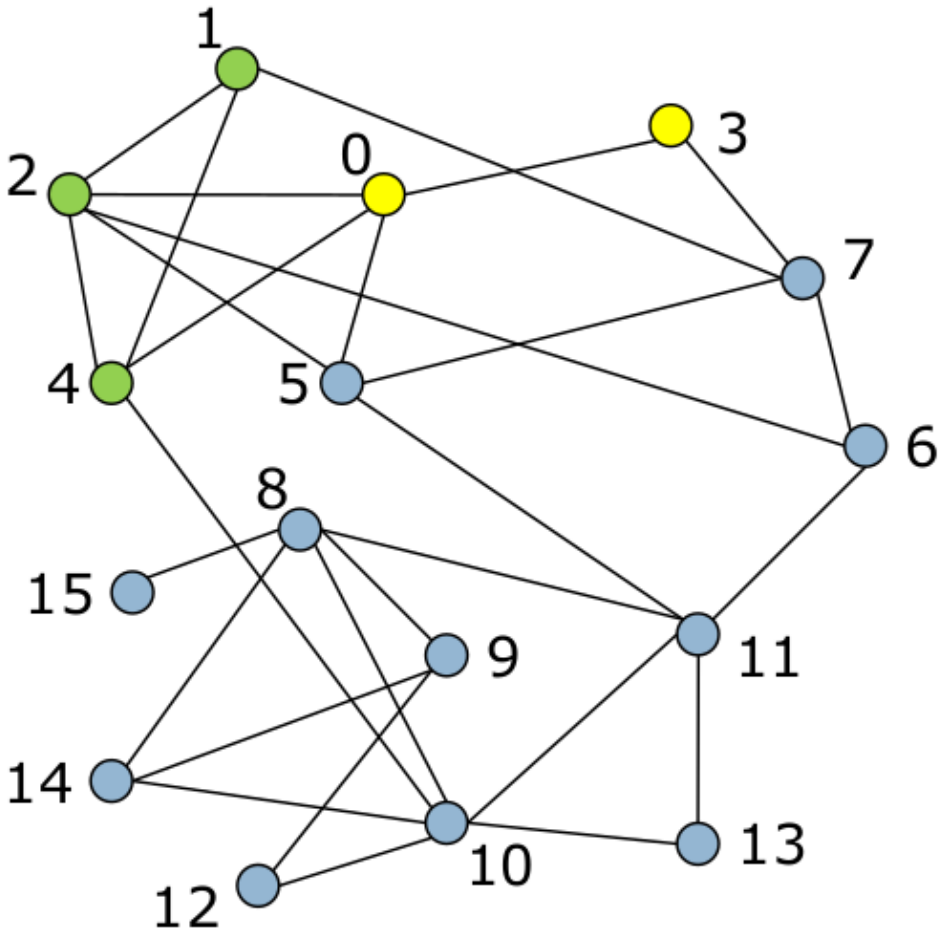
L'algorithme de Louvain



0 -> c[3]

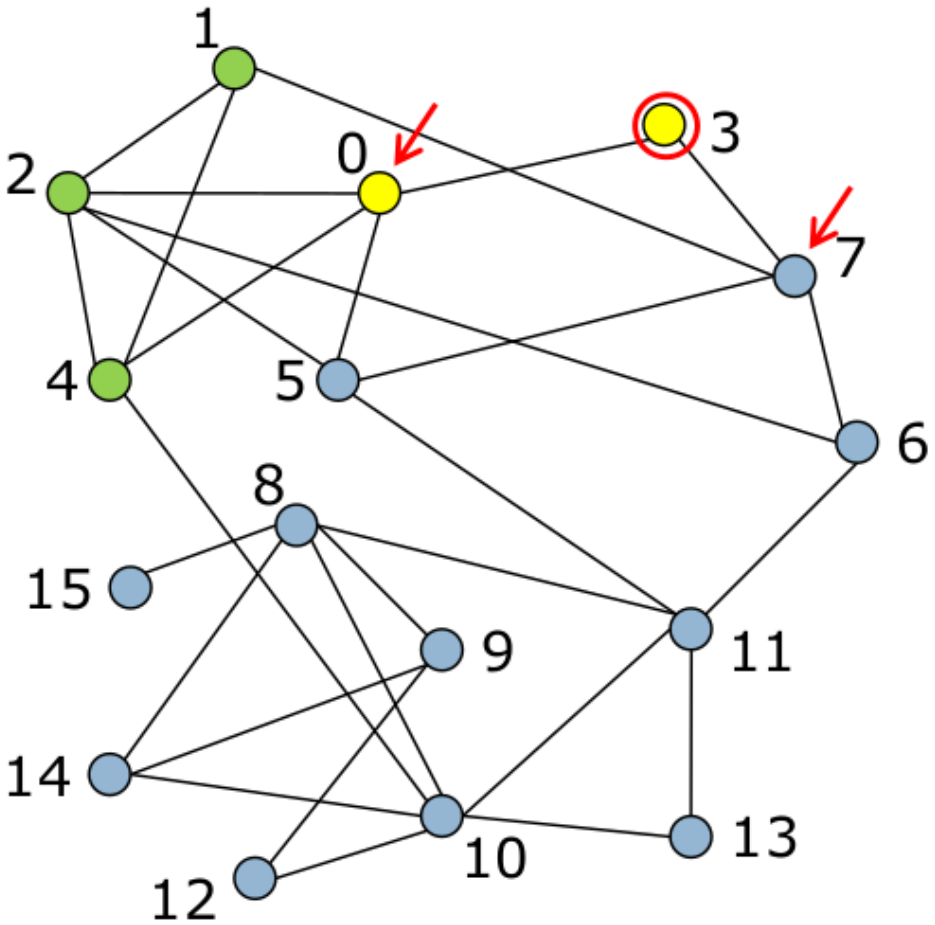
1 -> c[4]

L'algorithme de Louvain



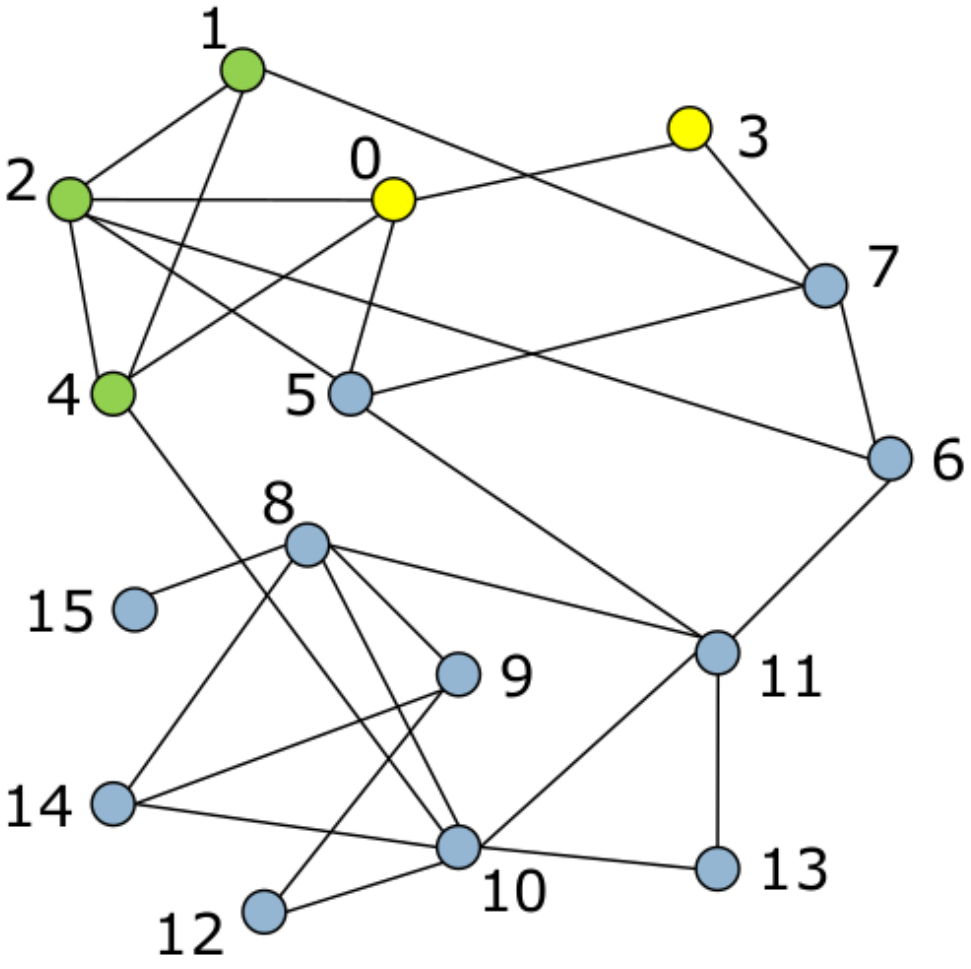
0 -> c[3]
1 -> c[4]
2 -> c[1,4]

L'algorithme de Louvain



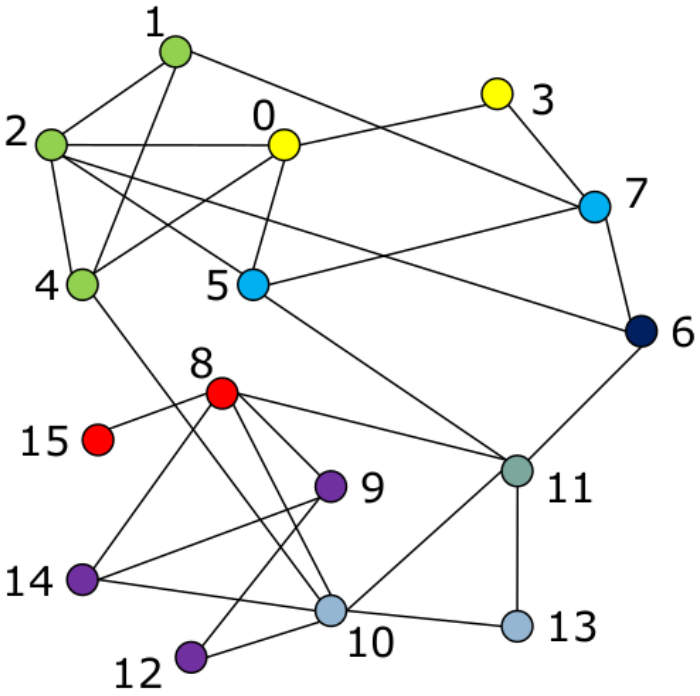
0 -> c[3]
1 -> c[4]
2 -> c[1,4]

L'algorithme de Louvain



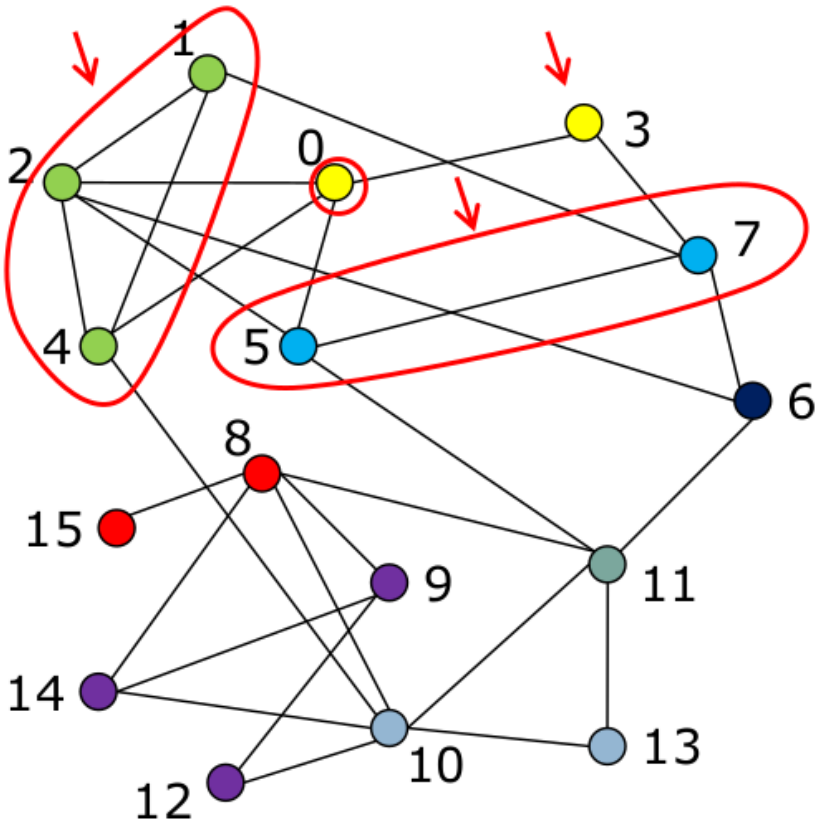
0 -> c[3]
1 -> c[4]
2 -> c[1,4]
3 -> c[0]

L'algorithme de Louvain



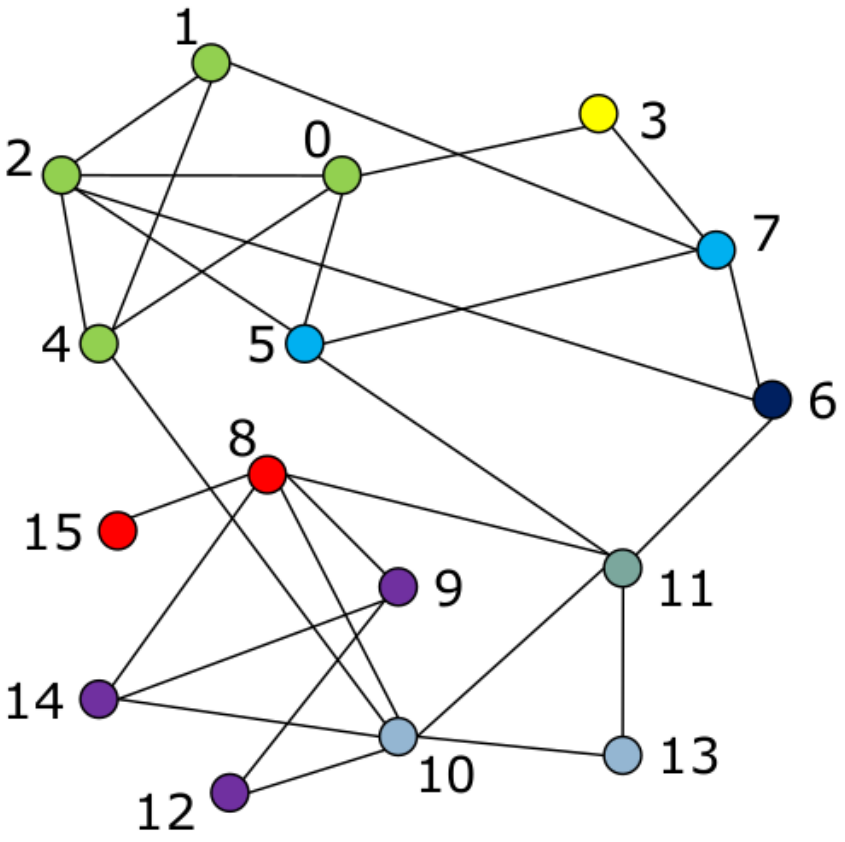
0 -> c[3]
1 -> c[4]
2 -> c[1,4]
3 -> c[0]
4 -> c[1]
5 -> c[7]
6 -> c[11]
7 -> c[5]
8 -> c[15]
9 -> c[12]
10 -> c[13]
11 -> c[10,13]
12 -> c[9]
13 -> c[10,11]
14 -> c[9,12]
15 -> c[8]

L'algorithme de Louvain



Passe 1 – Itération 2

L'algorithme de Louvain

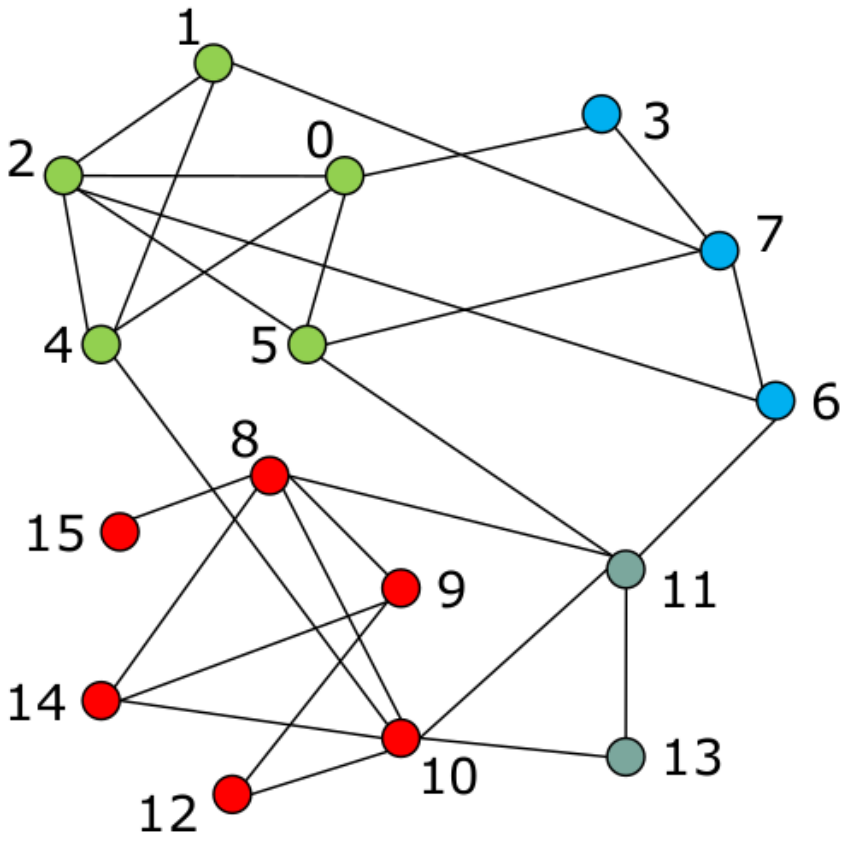


Passe 1 – Itération 2

0 -> c[4]

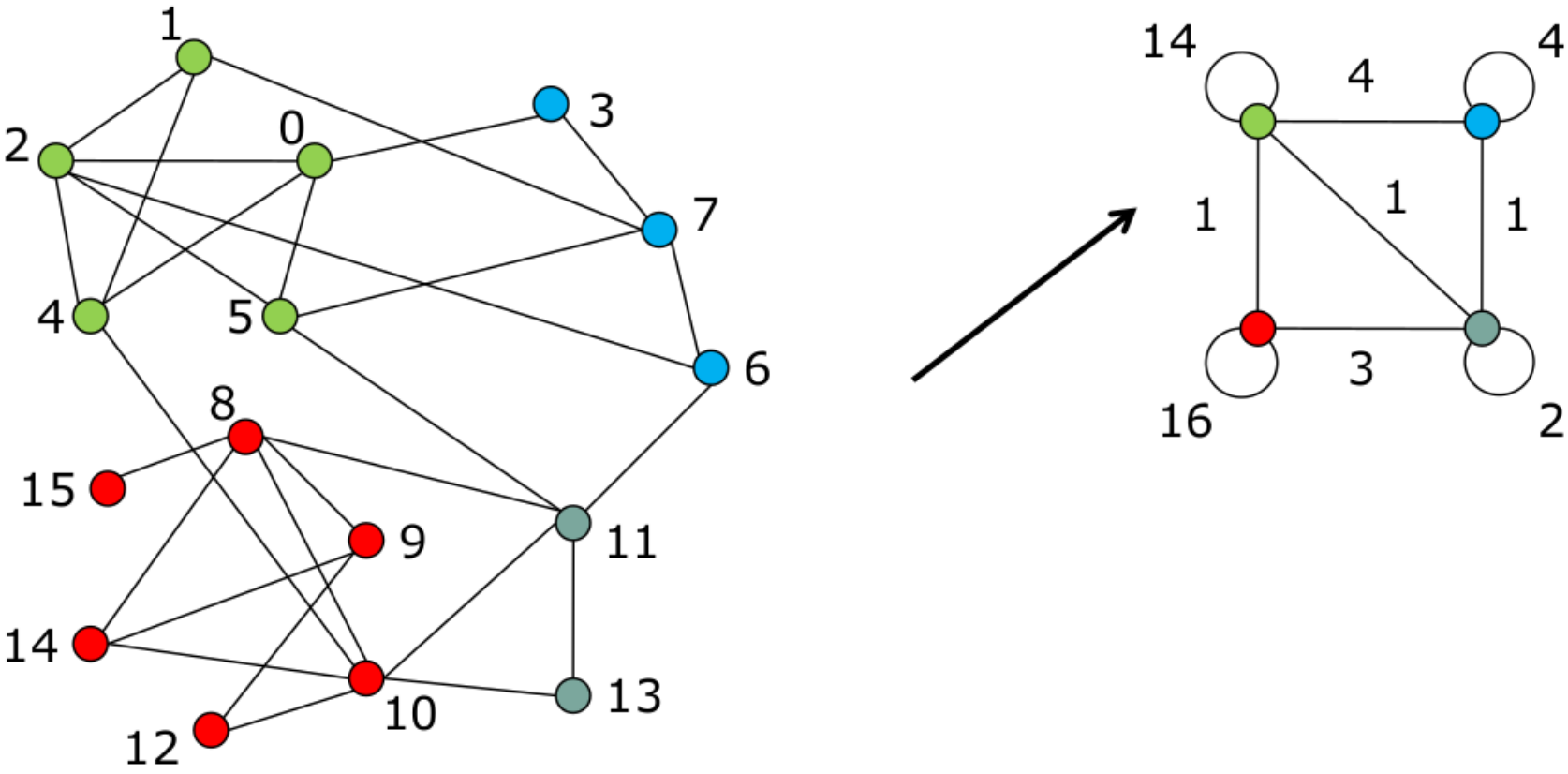
...

L'algorithme de Louvain

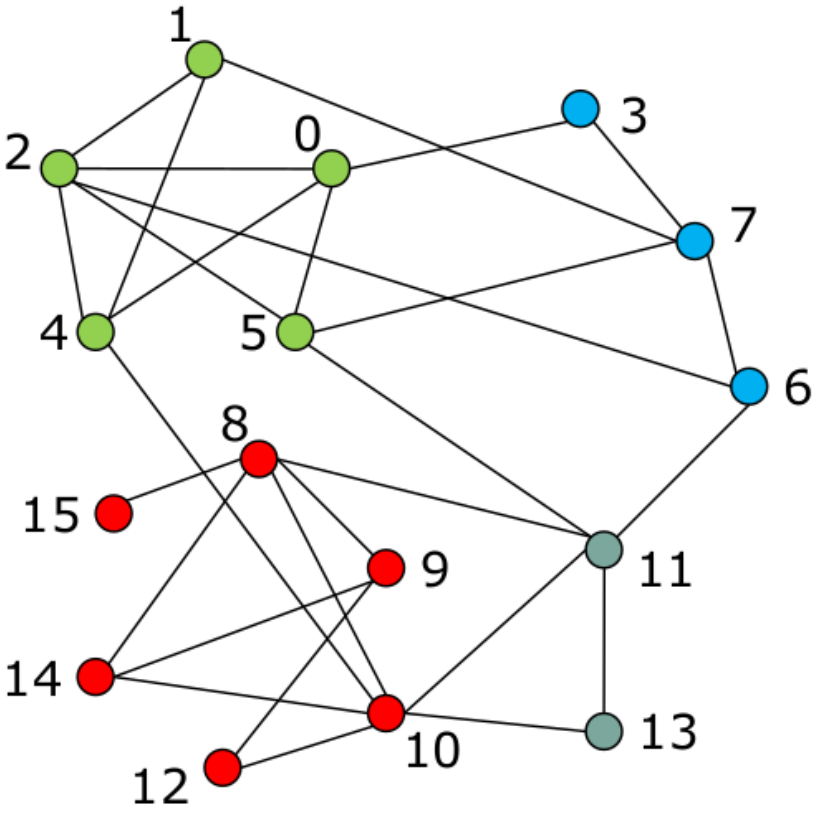


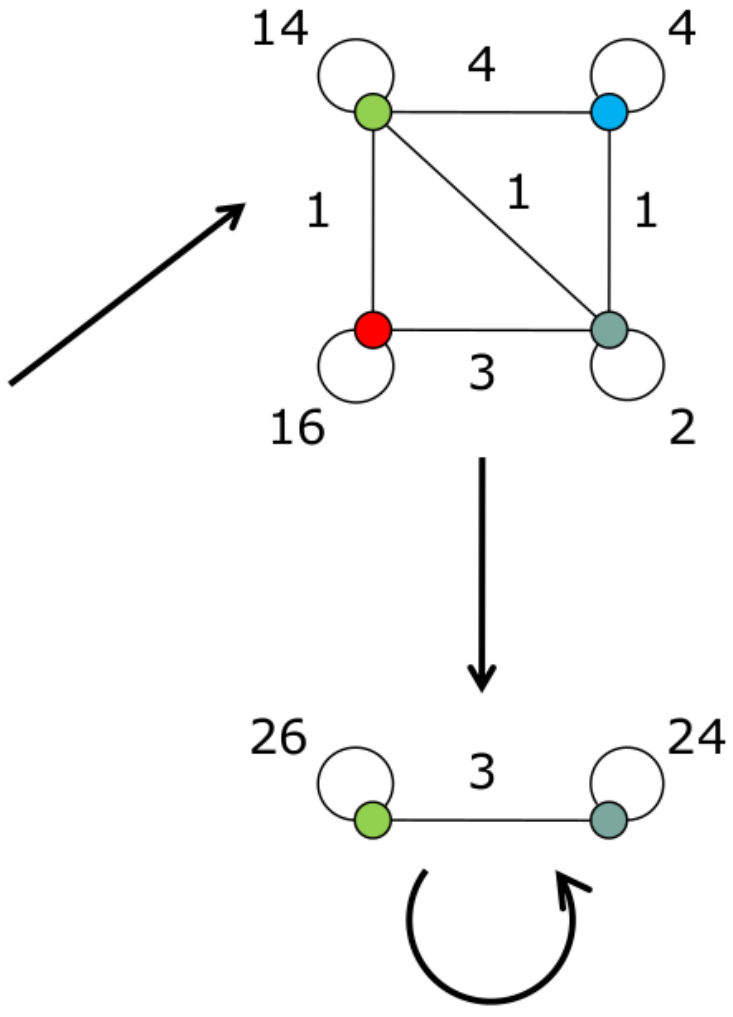
Après 4 itérations,
plus de changement

L'algorithme de Louvain



L'algorithme de Louvain





Performances

- en 5 passes et 22 minutes, un graphe de 118M/1MM est traité !
- gain très important par rapport aux algos précédents
- bonnes performances pour optimiser la modularité
- Les graphes aux niveaux supérieurs sont petits :
  - Les premières passes sont les plus couteuses.
  - En général : première passe > 90% du temps de calcul.
- Il y a peu d'itérations par passe :
  - Seules les itérations de la première passe sont couteuses.
  - Moins de 33 pour tous les réseaux testés.
- Traiter un sommet est simple.

Déplacer un sommet

- Un sommet isolé i peut être déplacé dans C avec un gain ne dépendant que de i et C
- complexité linéaire avec k_i
- Gain :

$$\Delta Q(C,i) = \left[\frac{e_C + k_{i,C}}{2m} - \left(\frac{a_C + k_i}{2m}\right)^2\right] - \left[\frac{e_C}{2m} - \left(\frac{a_C}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

Structure de données

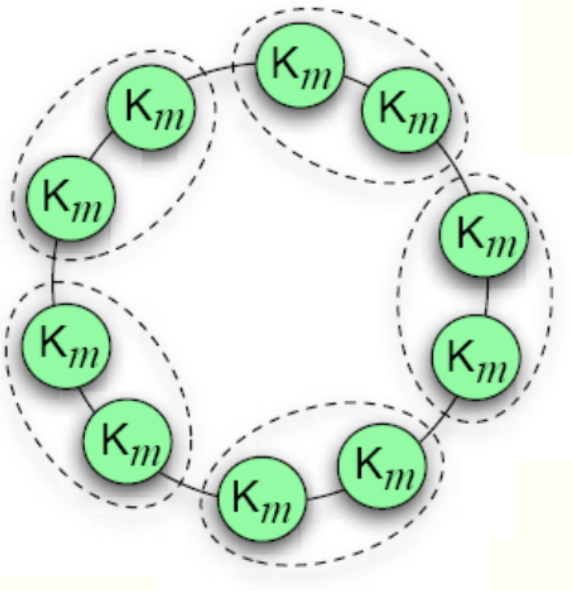
- À conserver dans la mémoire :
  - les listes d'adjacence (2m +n)
  - les vecteurs e, a, node2comm (n chacun)
  - total : 2m + 4n
  - 100 M nœuds, 1 G liens : env 10 Go
- algorithme itératif :
  - les listes d'adjacences peuvent être lues sur le disque séquentiellement
  - permet de gérer de très grands graphes sur des machines portables

heuristiques

- Il est possible d'accélérer encore
  - les dernières passes apportent un gain faible
  - arrêter dès que le gain est inférieur à un seuil
- Sommets de degrés peuvent être supprimés avant les calculs
- Seuls quelques sommets sont déplacés à chaque itération, un sommet statique peut être laissé de côté à l'étape suivante

Limite de résolution

- Anneau de cliques (n cliques de taille m)
- $Q_{single} = 1 - \frac{2}{m(m-1)+2} - \frac{1}{n}$
- $Q_{pairs} = 1 - \frac{2}{m(m-1)+2} - \frac{2}{n}$
- $Q_{single} > Q_{pairs} \iff$
  $m(m-1)+2 > n$



Limite de résolution

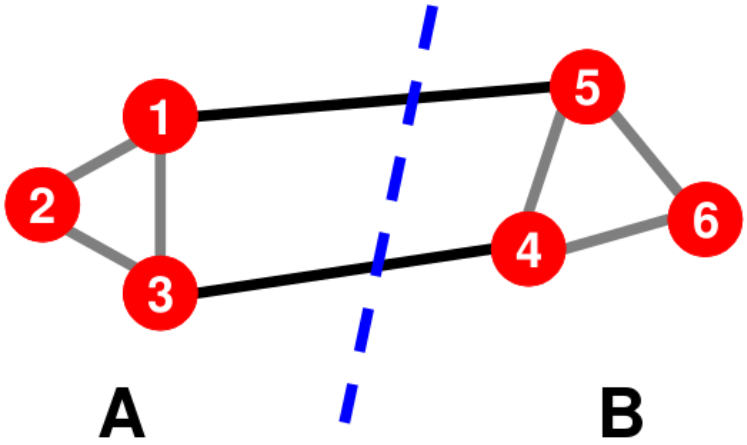
- Avec 30 cliques de tailles 5
- 30 > 22
- $Q_s = 0.876$, $Q_p = 0.888$
- contre-intuitif (on s'attend à ce que les cliques seules soient des zones plus denses que les paires de cliques)
- Peut apparaître à n'importe quelle échelle
- si l'on est confronté à des communautés de tailles différentes à l'intérieur d'un même graphe, certaines communautés, même bien définies, pourront ne pas être distinguées dans la partition de modularité optimale

Recuit simulé

- À chaque étape, mises à jour aléatoires
  - déplacements individuels
  - déplacements collectifs : fusion séparation
  - accepté avec probabilité :
    - $p = 1$ si ça fait augmenter la modularité
    - $p = exp\left(\frac{-\Delta Q}{T}\right) sinon$
  - le système est refroidi, $T' = 0.995T$

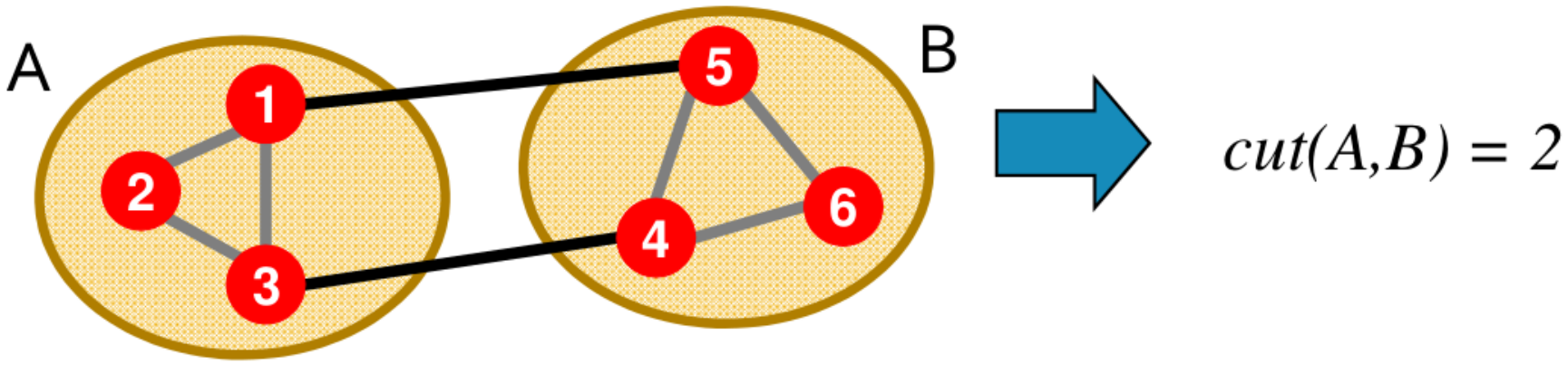
Approches spectrales

- On cherche à maximiser le nombre de connexions **intra-groupes**, à minimiser le nombre de connexions **inter-groupes**



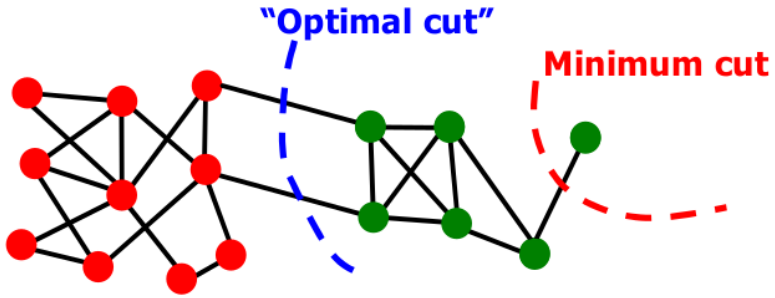
Approches spectrales

- On exprime le partitionnement en cherchant à optimiser une fonction calculant le nombre de “cut”
- “cut” = nombre de liens avec avec seulement un sommet par groupe



Minimum-cut

- Minimiser les connexions inter-groupes : $argmin_{A,B} cut(A, B)$
- Cas dégénéré



- Problème : ne considère pas les connexions intra

Normalized-cut

- Normalized-cut (Shi-Malik, 1997)
- Connectivité entre groupes relative à la densité de chaque groupe

$$ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

- $vol(A)$: total des liens qui ont une extrémité dans A
- Donne des partitions plus équilibrées
- Problème : optimiser cette valeur est NP-difficile (mais il y a des heuristiques, voir Fiedler '73)

Autres approches spectrales

- Laplacien de G
  - $L = D - A$ (où D : matrice de degrés, A : matrice d'adjacence)
- Si les communautés sont claires, les vecteurs propres permettent de trouver les communautés
- sinon il faut une distance entre vecteurs
- Approches à base de matrices de similarité K
  - Matrices plus complexes
  - Distances simples à partir de K

Algorithmes génétiques

- Chromosomes
  - Partition en communautés
  - initialement : communautés de sommets connectés
- Cross-over (enjambement)
  - Sommets qui appartiennent à c dans C1 vont vers c dans C2
- Mutation
  - Un sommet est déplacé aléatoirement
- Clean-up
  - Un sommet dont les voisins sont ailleurs est mis avec eux

D'autres algorithmes

- Équations de Kirchhoff
- Wu and Huberman, Eur Phys B 38, 2004
- Modèle de Potts
- Reichardt and Bornholdt, Phys Rev Lett 93, 2004

- complexité assez variable

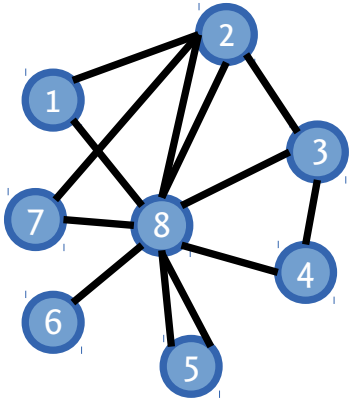
Voir Fortunato 2009

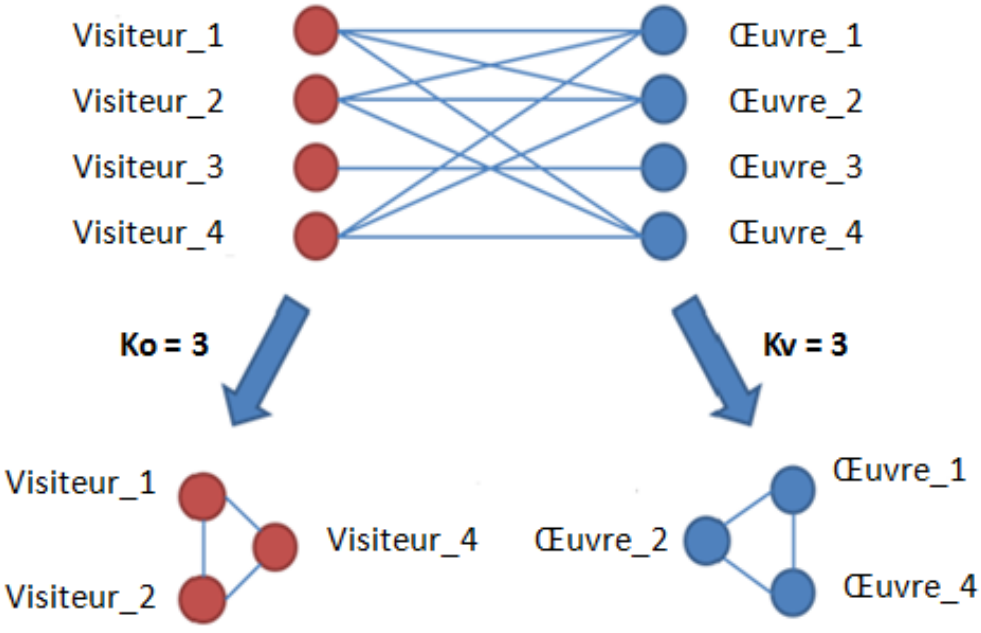
D’autres communautés

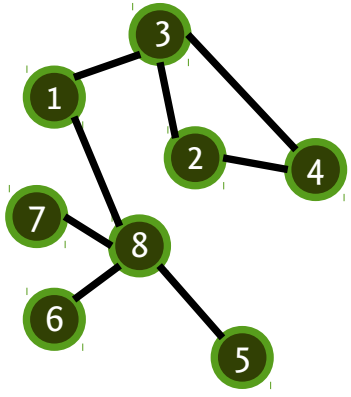
- Recouvrantes
- Égo-centrées
- Dans des graphes bipartis
- etc.

Systèmes de recommandation et graphes

Approche







graphes de données *implicites* ou *explicites*

Formalisme unifié

- un lien dans le projeté (u_1, u_2) peut être vu comme une règle d'association $u_1 \rightarrow u2$
- poids $\propto$ support de la règle

$$Supp(u_1 \rightarrow u2) = \frac{\#ItemsViewedBy\ u_1\ and\ u_2}{\#Items}$$

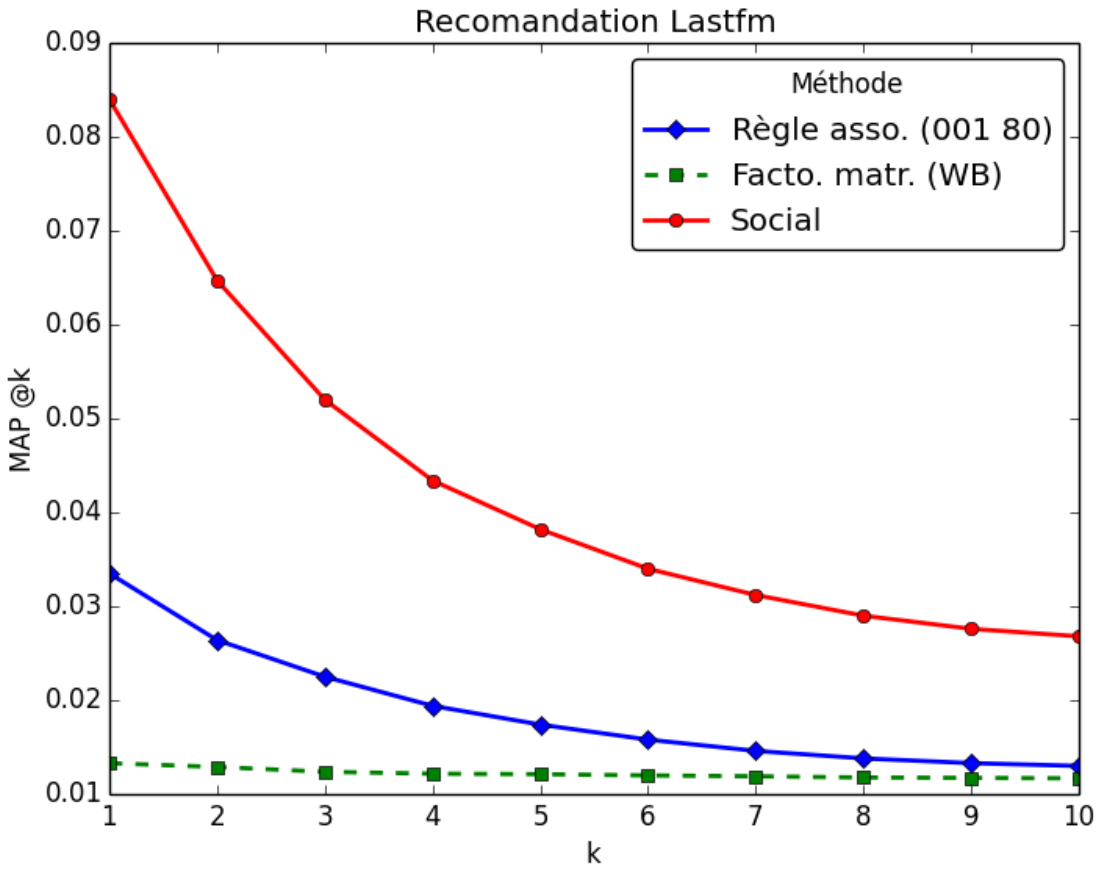
- dans le cas où la matrice est binaire, on a :

$$Supp(u_1 \rightarrow u2) = \frac{1}{C} \sum_{i}^{C} r_{u_1 i} \times r_{u_2 i}$$

- équivalent à la similarité cosinus

Comme voisinage, on s'en tient à $K(u)$, 1er cercle dans le graphe

Apport de l'information sociale



Gain significatif avec information sociale

Conclusion

Références

- Ce cours repose sur les travaux de :
  - l'équipe ComplexNetworks du LIP6 (UPMC), http://www.complexnetworks.fr (membres passés et présents)
  - en particulier les cours de Jean-Loup Guillaume (PR, U. de La Rochelle) et de Clémence Magnien
  - le livre *Mining Massive datasets* (http://www.mmds.org), de Jure Leskovec, Anand Rajaraman, Jeff Ullman