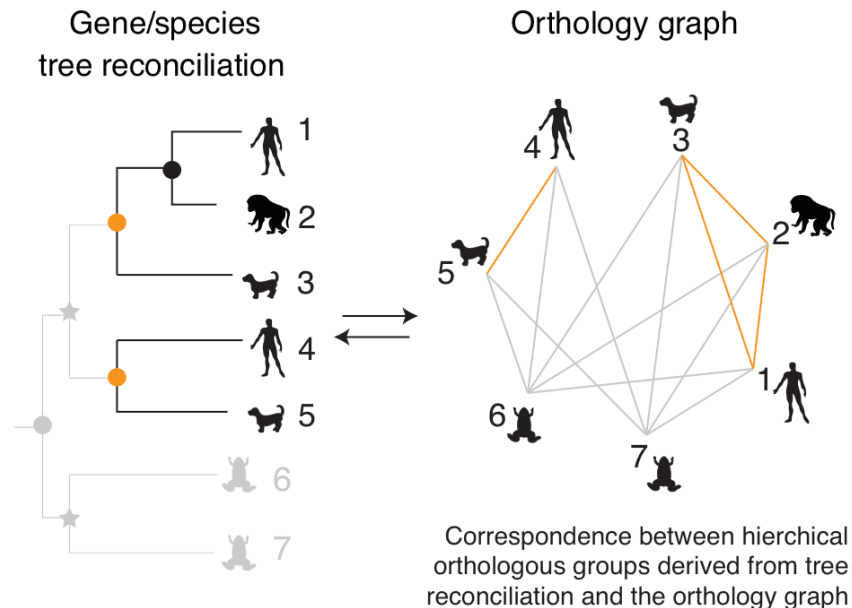


# Traitement des graphes et réseaux biologiques

## Master 1 MABS



- Plan
  - ◆ Evolution des génomes et importance de l'orthologie
  - ◆ Définitions et hypothèse(s) sous jacente(s)
  - ◆ Approches pour réaliser la construction de groupes de gènes orthologues
  - ◆ Méthodes basées sur des partitionnement de graphes

## Pourquoi étudier l'évolution des génomes ?

- ♦ **Structure et organisation** des génomes
    - Lien avec l'expression du programme génétique
  - ♦ **Transmission verticale/horizontale**: impact de la reproduction sexée, de la recombinaison, des transferts de gènes.
  - ♦ **Coévolution** : entre organismes différents, hôte/symbiote ou hôte/pathogène (microbiote intestinal).
  - ♦ **Adaptation des organismes à leur environnement** : rôles des duplications, des délétion et des éléments génétiques mobiles (domestication).
  - ♦ **Phylogénie des espèces**, : différentes échelles, ex. l'histoire des populations (ADN ancien/fossile).
  - ♦ **Origine de la vie**
- Ces différentes analyses reposent sur l'identification de liens **d'orthologies**

En pratique: utilisations de l'orthologie

- 1. Reconstruction d'arbre phylogénétiques** basés sur un ensemble de gènes orthologues (super arbre, super matrice), sous l'hypothèse que ces gènes suivent la divergence des espèces.
  - 2. Inférence fonctionnelle** : établir des relations fonctionnelles entre différents génomes, sous l'hypothèse que les gènes orthologues conservent la même fonction.
  - 3. Phylogénomique** : prérequis pour l'analyse comparative et évolutive des génomes.
- Importante communauté internationale s'est structurée autour de la question de l'identification des gènes orthologues.

[http://questfororthologs.org/orthology\\_databases](http://questfororthologs.org/orthology_databases)

MEETINGS  
COMMUNITY STANDARDS  
ORTHOLOGY DATABASES  
DOCUMENTS (INTRANET)  
MAILING-LIST & CONTACT

## QUEST FOR ORTHOLOGS

**QfO 7 will take place in Frankfurt, Germany, in summer 2021. Stay tuned!**

### Welcome

This is the site of the Quest for Orthologs consortium. Proteins and functional modules are evolutionarily conserved even between distantly related species, and allow knowledge transfer between well-characterized model organisms and human. The underlying biological concept is called 'Orthology' and the identification of gene relationships is the basis for comparative studies.

More than 30 phylogenomic databases provide their analysis results to the scientific community. The content of these databases differs in many ways, such as the number of species, taxonomic range, sampling density, and applied methodology. What is more, phylogenomic databases differ in their concepts, making a comparison difficult – for the benchmarking of analysis results as well as for the user community to select the most appropriate database for a particular experiment.

The Quest for Orthologs (QfO) is a joint effort to benchmark, improve and standardize orthology predictions through collaboration, the use of shared reference datasets, and evaluation of emerging new methods.

The main sections of this site are:

- [Meetings](#)
- [Community Standards](#) (Reference proteome, standardized formats, benchmarking, etc..)
- [Working groups](#)
- [Orthology databases](#)
- [Documents \(Intranet\)](#)
- [Mailing-List and Contact](#)

To contribute to this website, please create an account (see below) and [contact](#) us!

Orthology   
Benchmarking

## QUEST FOR ORTHOLOGS

MEETINGS  
COMMUNITY STANDARDS  
ORTHOLOGY DATABASES  
DOCUMENTS (INTRANET)  
MAILING-LIST & CONTACT

### List of orthology databases

*If you know of any other database, please edit this page directly or contact us.*

1. [COGs/TWOGa/KOGs](#)
2. [COGs-COCO-CL](#)
3. [COGs-LOFT](#)
4. [eggNOG](#)
5. [EGO](#)
6. [Ensembl Compara](#)
7. [Gene-Oriented Ortholog Database](#)
8. [GreenPhylDB](#)
9. [HCOP](#)
10. [HomoloGene](#)
11. [HOGENOM](#)
12. [HOVERGEN](#)
13. [HOMOLENS](#)
14. [HOPS](#)
15. [INVHOGEN](#)
16. [InParanoid](#)
17. [KEGG Orthology](#)
18. [MBGD](#)
19. [MGD](#)
20. [OMA](#)
21. [OrthoDB](#)
22. [OrthologID](#)
23. [ORTHOLOGE](#)
24. [OrthoInspector](#)
25. [OrthoMCL](#)
26. [Panther](#)
27. [PhIGs](#)
28. [PHOG](#)
29. [PhylomeDB](#)
30. [PLAZA](#)
31. [P-POD](#)
32. [ProGMap](#)
33. [Proteinortho](#)
34. [RoundUp](#)
35. [TreeFam](#)
36. [YOGY](#)

## Définitions (Fitch, 2000)

- **Homologie**: relation entre deux entités biologiques (caractères anatomique, séquences...) qui dérivent d'un ancêtre commun.
- **Orthologues**: séquences qui dérivent par un événement de **spéciation** de leur dernier ancêtre commun.
- **Paralogues**: séquences qui dérivent par un événement de **duplication** de leur dernier ancêtre commun.

Concepts définies dans le cadre de la théorie de l'évolution,

- pas directement lié à la **fonction** des séquences impliquées!

Remarque, les protéines homologues ont généralement conservées la même structure 3D.

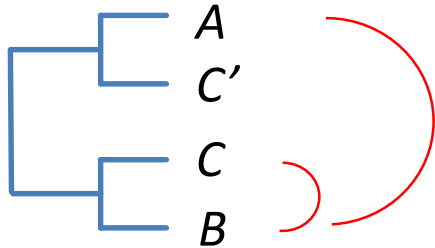
Duplication:

- néo ou sub fonctionnalisation.

## Problèmes

La relation d'orthologie **n'est pas transitive**:

- Le gène *A* peut être orthologue au gène *B* et le gène *B* orthologue au gène *C* mais les gènes *A* et *C* ne sont pas nécessairement orthologues.

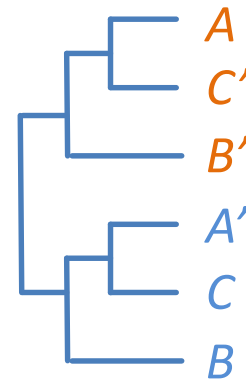
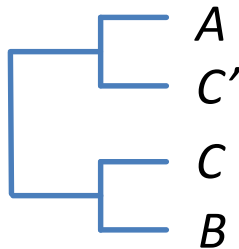




## Problèmes

La relation d'orthologie **n'est pas transitive**:

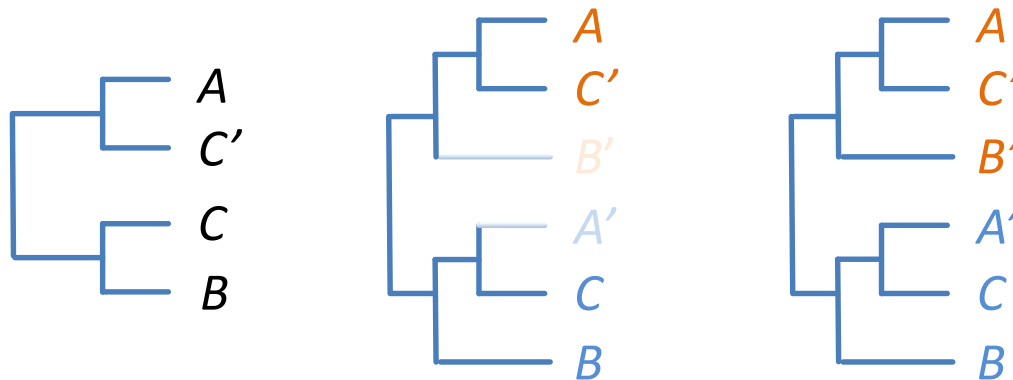
- Le gène *A* peut être orthologue au gène *B* et le gène *B* orthologue au gène *C* mais les gènes *A* et *C* ne sont pas nécessairement orthologues.



## Problèmes

La relation d'orthologie **n'est pas transitive**:

- Le gène *A* peut être orthologue au gène *B* et le gène *B* orthologue au gène *C* mais les gènes *A* et *C* ne sont pas nécessairement orthologues.



-> Difficulté de définir un **'groupe de gènes orthologues'**.

Autres définitions basées sur une paire de gènes et un **événement** de spéciation  
(Remm *et al.*, 2001 )

- **in-paralogues :**

- ♦ séquences paralogues issues d'une duplication **postérieur** à un événement de spéciation (paralogues récents)  
-> **fonction similaire?**

- **out-paralogues :**

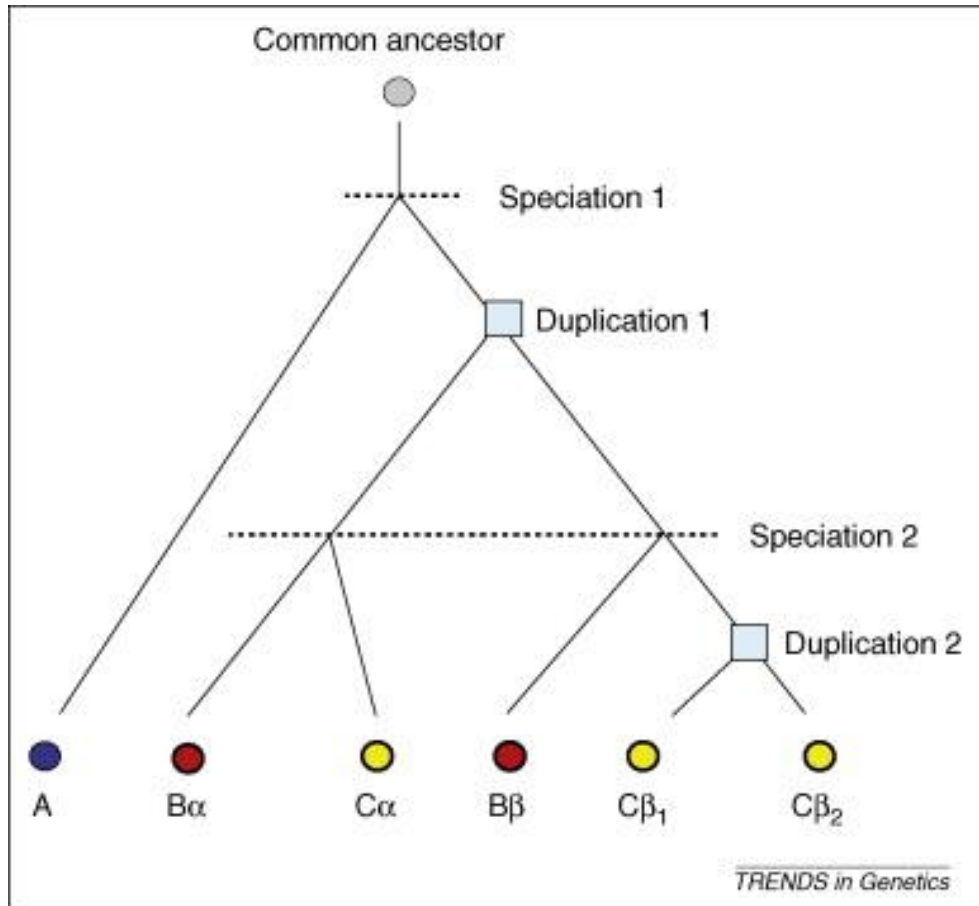
- ♦ séquences paralogues issues d'une duplication **antérieur** à un événement de spéciation (paralogues anciens)  
-> **fonction différente?**

Le gros problème!

Les définitions précédentes se rapportent à des événements de spéciation et de duplication observés sur l'arbre **modélisant** l'évolution de ces séquences (structure hiérarchique).

Et cet arbre est inconnu!!!

# Illustration



Studer et Robinson-Rechavi, 2009

Arbre décrivant l'évolution de 6 gènes homologues appartenant à trois espèces (A, B et C).

Duplication 1 :  $\alpha$  et  $\beta$  chez le dernier ancêtre commun de B and C,

Duplication 2 :  $\beta_1$  et  $\beta_2$  dans la lignée C.

Seul le gène de l'espèce A n'a pas de ?.

Tous les gènes de B et C sont ? au gène de A.

Les gènes  $\alpha$  et  $\beta$  sont ? / spéciation 1 et ? / spéciation 2.

Les gènes  $\beta_1$  et  $\beta_2$  sont ? / spéciations 1 et 2.

Les gènes B $\alpha$  et C $\alpha$  sont ?.

## Groupe de gènes orthologues (OG)

Ensemble de gènes homologues dont chaque paire possible est une relation **d'orthologie** ou de **in-paralogie** par rapport au dernier événement de spéciation (en pratique: une duplication dans la même espèce!).

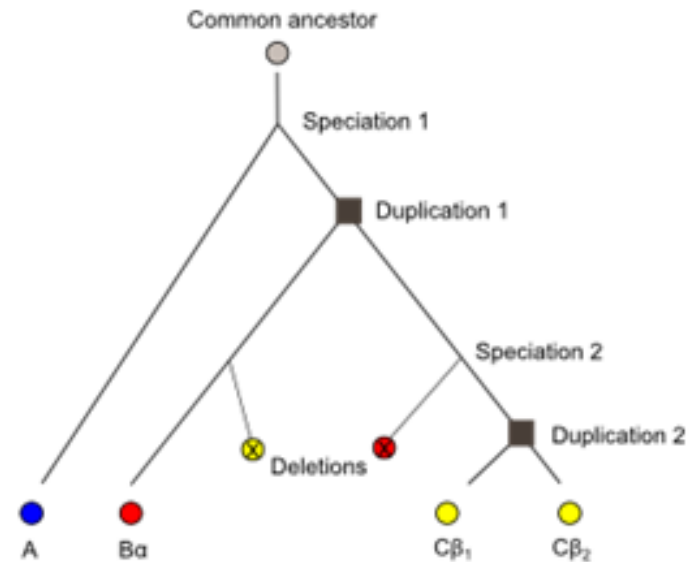
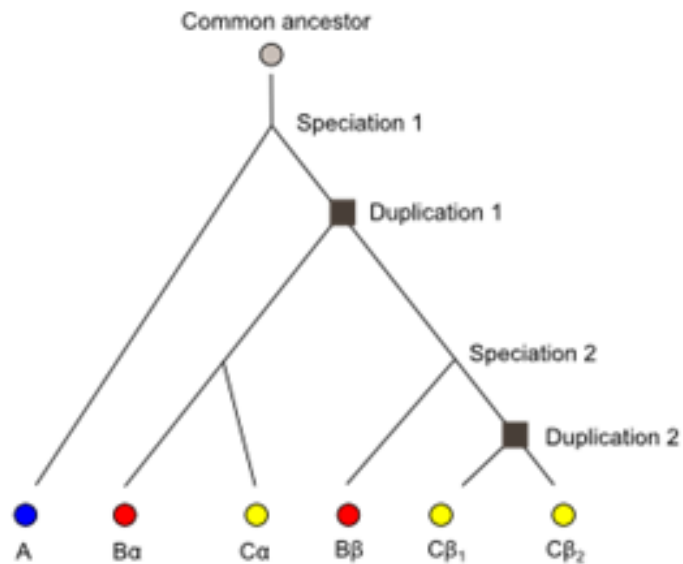
Dans un OG les paires de gènes sont

- **orthologues** si les gènes appartiennent à des espèces différentes
- **paralogues** si les gènes appartiennent à la même espèce.

Pièges dans la construction d'OG?

# Pièges dans la construction d'OG

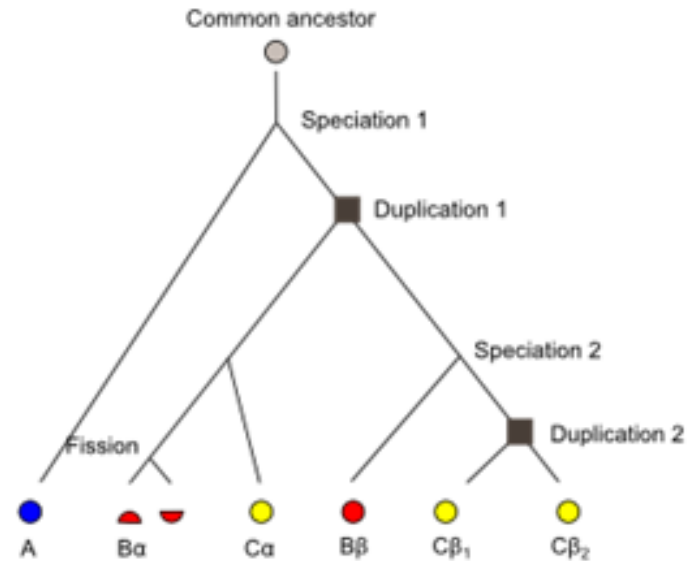
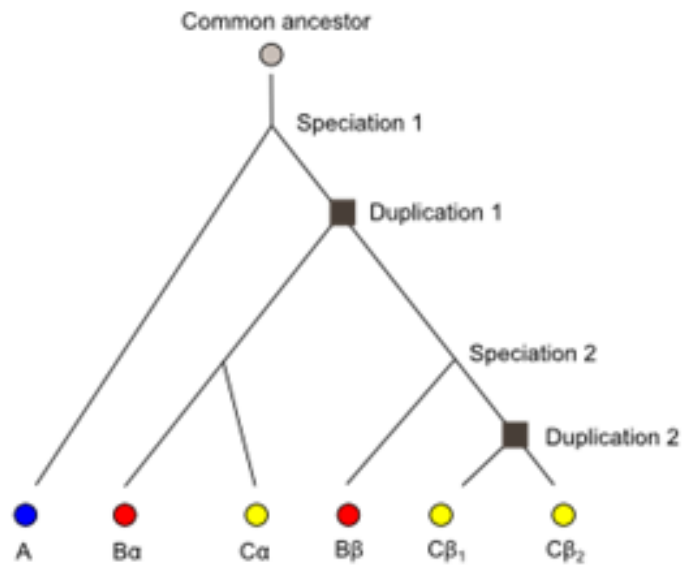
Pièges dans la construction d'OG:  
on ne connaît pas l'histoire des séquences!



Les gènes  $B\alpha$  et  $C\beta_1$ ,  $C\beta_2$  sont ?

# Pièges dans la construction d'OG

Pièges dans la construction d'OG: on ne connaît pas l'histoire des séquences!

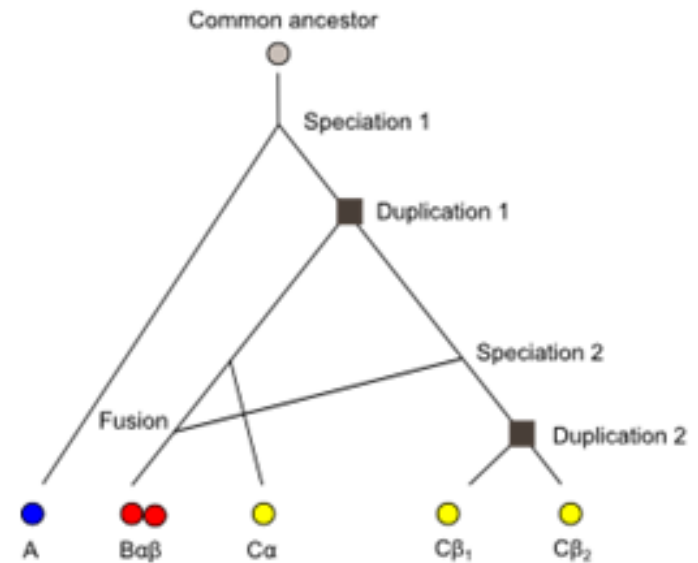
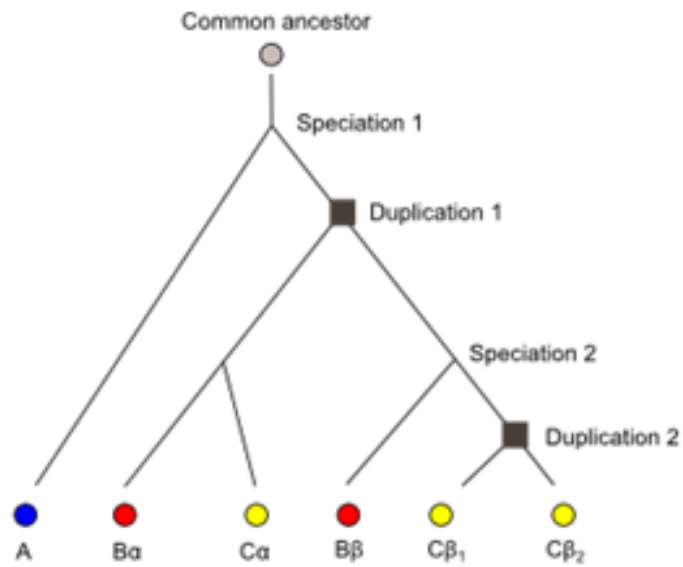


Les gènes Ba<sub>1</sub> et Ba<sub>2</sub> sont ? de Ca



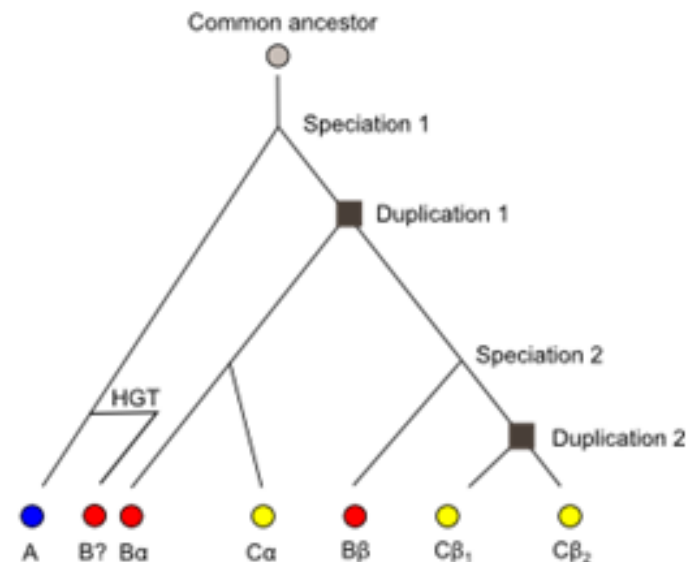
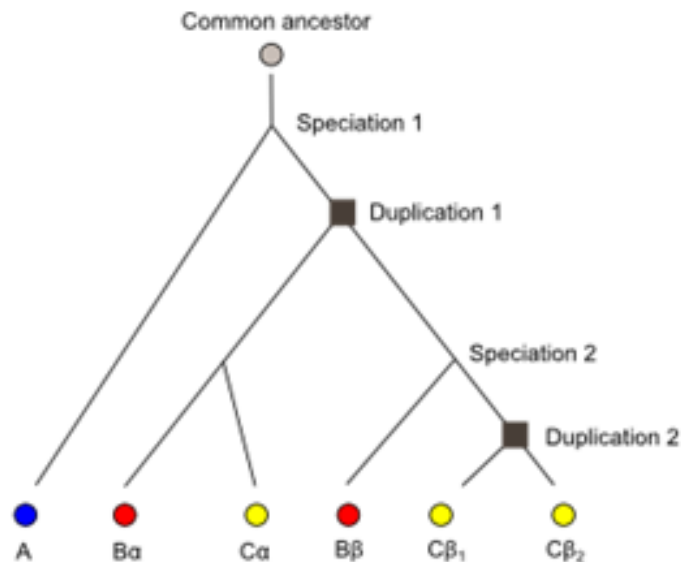
# Pièges dans la construction d'OG

Pièges dans la construction d'OG: on ne connaît pas l'histoire des séquences!



Le gène  $Ba\beta$  est ? à  $Ca$  et ? à  $C\beta_1$ ,  $C\beta_2$

Pièges dans la construction d'OG: on ne connaît pas l'histoire des séquences!



- + erreurs liées à la méthodologie
  - Envisager la construction d'OG pouvant se recouvrir!

Comment établir les relations d'orthologie (Kuzniar *et al.*, 2008)?

Comment établir les relations d'orthologie (Kuzniar *et al.*, 2008)

Approche classique à l'échelle d'une famille de gènes:

- reconstruction d'arbres et identification des événements de spéciation et de gain et perte de gènes.

A l'échelle des génomes

1. Méthodes basées sur la reconstruction d'un **arbre** phylogénétique (référence) et d'une analyse de l'évolution des gènes homologues / référence.
2. Méthodes basées sur l'analyse du **graphe** constitué de l'ensemble de paires de gènes orthologues.
3. Méthodes hybrides (arbres et graphes).

Utilisation d'autres informations que la séquences (synthénie, ontologie...)

## Observations

Les arbres obtenus sur les gènes ne sont pas nécessairement **congruents**

- entre eux
  - et avec l'**arbre** phylogénétique (des espèces)
- méthode de **réconciliation d'arbres**: prédire le nombre minimum de duplications et de pertes de gènes pour expliquer les données (Dufayard *et al.*, 2005; Zmasek et Eddy 2001; Goodman *et al.*, 1979; Page et Charleston 1997)

## Observations

### Limitations

- Les arbres obtenus doivent être fiables or c'est rarement le cas (prise en compte des nœuds incertains)!
- Les arbres doivent être correctement enracinés (manuellement?).
- Les alignements multiples utilisés pour construire les arbres doivent être de très bonnes qualités.
- Les logiciels d'alignements multiples et de reconstruction d'arbres voient leurs performances décroître avec l'augmentation du nombre de séquences (heuristiques, parallélisations...).
- L'ensemble de la procédure (contrôle qualité) est difficile à automatiser.

## Méthodes basées sur les graphes

Repose sur l'identification de liens d'orthologie entre paires de séquences.

- Approche indirecte: les liens d'orthologie sont inférés à partir de l'alignement de paires de séquences.
- Alignement global ou local?

## Méthodes basées sur les graphes

Méthode la plus répandue:

Programme : BlastP ou équivalent

Critères utilisés : score de similarité (ou e-value), pourcentage de séquence alignée.

Notation :

$BP(A, A')$ , si  $A$  et  $A'$  appartiennent à la même souche (Best Paralogs)

$BH(A, B)$ , si  $A$  et  $B$  appartiennent à des souches différentes (Best Homologs)

- Réaliser les BlastP pour chaque paire de souches
- Pour chaque paire de souches
  - ♦ Retenir pour chaque séquence soumise  $A$ , la séquence  $B$  maximisant le critère (Score  $> S_{\text{seuil}}$ ) et présentant une couverture (%ali)  $> S_{\text{ali}}$ .
    - si  $BH(A, B) = BH(B, A)$  alors  $A$  et  $B$  sont **Best Bidirectional Hits (BBH)**.
    - si  $BBH(A, B) > BP(A, A')$  et  $> BP(B, B')$  alors  $A$  et  $B$  sont **orthologues 1:1**.

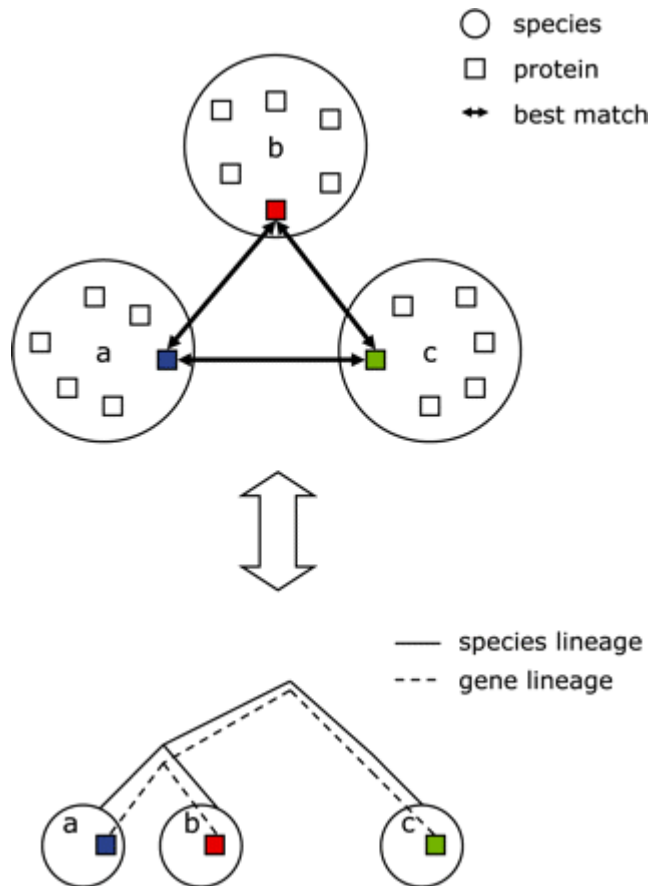


Relation entre graphe et arbre phylogénétique.  
 Graphe G:

sommets: gènes/protéines

arrêtes: liens d'orthologies

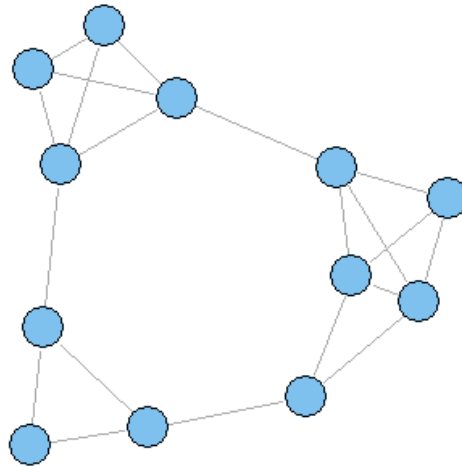
- Un sous graphe de G est un triangle entre a, b et c si (a, b), (b, c) et (a, c) sont orthologues (génomomes différents).



Version original de la BD COG

- Chaque groupe de gènes orthologues est un sous-graphe de G initialisé par un triangle auquel on ajoute les triangles ayant un coté en communs.

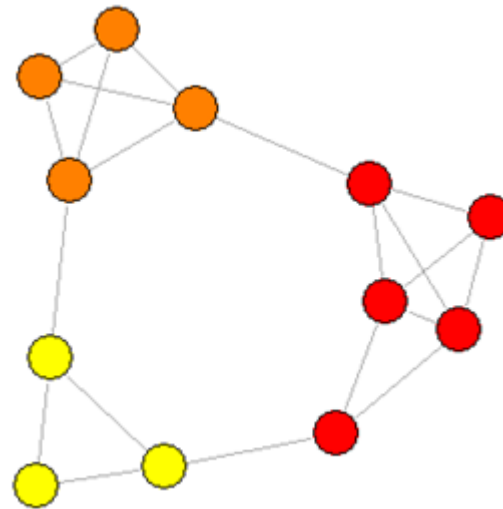
Groupes d'orthologues = communautés dans les graphes?



Combien de communautés?

## Structure en communautés dans les graphes

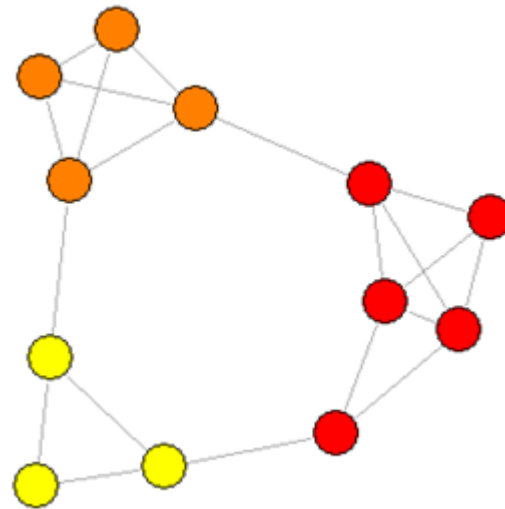
- Graphes aléatoires :
  - ♦ la distribution des arêtes entre les sommets est très homogène.
- Dans les graphes réels (biologiques) :
  - ♦ forte inhomogénéité, traduisant une organisation complexe,
  - ♦ un grand nombre de sommets de faible degré coexistent avec un petit nombre de sommets de degré élevé (scale-free network, graphes dont les degrés des nœuds  $\sim$  loi de puissance).
- Cas des relations d'orthologies  $\Rightarrow$  structure en **communautés**.
- Exemple : un graphe avec trois communautés



- Approche basée sur les graphes :
  - Identifier des communautés uniquement sur la base de la topologie du graphe.
- Questions toujours débattues:
  - ◆ Quelle est le meilleur partitionnement d'un graphe en communautés?
  - ◆ Communautés chevauchantes ou non?
  - ◆ Une partition ou une organisation hiérarchique des partitions?
  - ◆ Comment comparer des partitions réalisées sur le même graphe?
  - ◆ Mesurer la qualité d'une partition?

## Structure de communautés dans les graphes

- **Remarque:** Identifier des communautés n'est possible que si le graphe est peu dense.
- Intuitivement, une communauté se caractérise par une plus grande densité d'arêtes *intra* / *inter*.



- **Remarque:** Identifier des communautés n'est possible que si le graphe est peu dense.
- Intuitivement, une communauté se caractérise par une plus grande densité d'arêtes *intra / inter*.
- Trois façons de considérer le problème:
  - ♦ **localement:** focaliser sur les arêtes du voisinage immédiat.
    - Exemple, une communauté = sous graphe pour lequel chaque nœud à plus de voisins dans, qu'en dehors du sous graphe .
  - ♦ **globalement:** considérer le graph dans son ensemble.
    - On a recours à un modèle nul: un graphe aléatoire (pas de structure en communautés).
      - ♦ Exemple, un graphe version randomisée du graphe original (les nœuds conservent leur degré).
  - ♦ **similarité entre les sommets:** les communautés sont des groupes de sommets similaires. Reste à définir la similarité entre les sommets!

## Evaluer la qualité d'une partition : modularité Q

- Problème : existe un grand nombre de partitions possibles!

- Problème : existe un grand nombre de partitions possibles!
  - Utiliser des heuristiques et une fonction objective à minimiser ou maximiser
    - Un plus : estimer la **qualité de la partition** obtenue



## Evaluer la qualité d'une partition : modularité Q

- Estimer la **qualité de la partition** obtenue
- La fonction de modularité  $Q$  (Newman et Girvan, 2004) est la plus populaire.

$$Q = \frac{1}{2m} \sum_{ij} \left( \underset{\text{observé}}{A_{ij}} - \underset{\text{attendu}}{\frac{k_i k_j}{2m}} \right) \delta(C_i, C_j)$$

- avec
  - ♦  $A_{ij}$  : matrice d'adjacence du graphe,
  - ♦  $k_i$  : degré du sommet  $i$  (aussi noté  $d(i)$ )
  - ♦  $m$  : nombre total d'arêtes dans le graphe
- La fonction  $\delta$  est égale à 1 si  $i$  et  $j$  appartiennent à la même communauté et 0 sinon.
  - ♦ Ainsi, seules les contributions provenant de sommets appartenant à la même communauté seront prises en compte.

- D'où une autre formulation en fonction des liens dans la communauté:

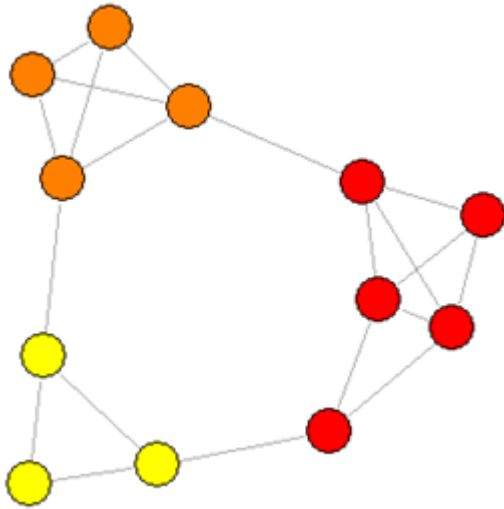
$$Q = \sum_{s=1}^{n_m} \left[ \frac{l_s}{m} - \left( \frac{d_s}{2m} \right)^2 \right]$$

- avec
  - ♦  $n_m$  : le nombre de communautés dans le graphe
  - ♦  $m$  : le nombre total d'arêtes dans le graphe
  - ♦  $l_s$  : le nombre total d'arêtes dans la communauté  $s$
  - ♦  $d_s$  : la somme des degrés des sommets de la communauté  $s$

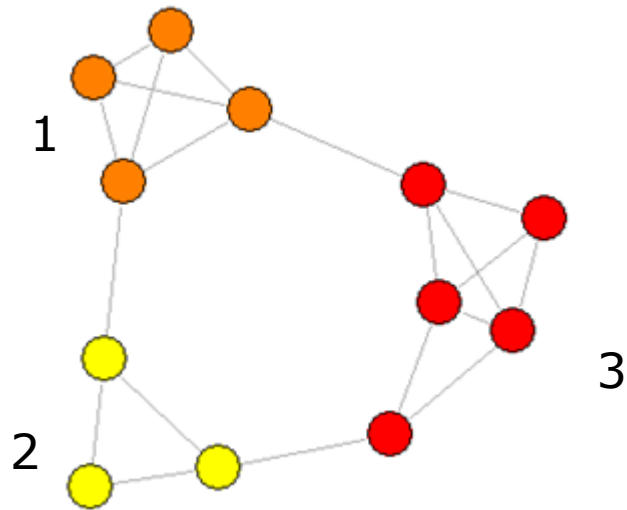
$$Q = \sum_{s=1}^{n_m} \left[ \underbrace{\frac{l_s}{m}}_{\text{observé}} - \underbrace{\left( \frac{d_s}{2m} \right)^2}_{\text{attendu}} \right]$$

- Premier terme:
  - fraction d'arêtes **observées** dans la communauté  $s$ .
- Second terme,
  - fraction des arêtes **attendues** dans la communauté  $s$ , si le graphe était aléatoire, sous la contrainte de la conservation du degré des sommets.

Un sommet peut être lié à n'importe quel autre sommet du graphe et la probabilité d'un lien entre deux sommets est proportionnel au produit de leur degré.
- Un écart important entre observé et attendu conduit à des valeurs positives  $Q$ , signalant une structure en communautés ( $\max(Q) = 1$ ).
- $Q \leq 0$ , -> absence de structure.



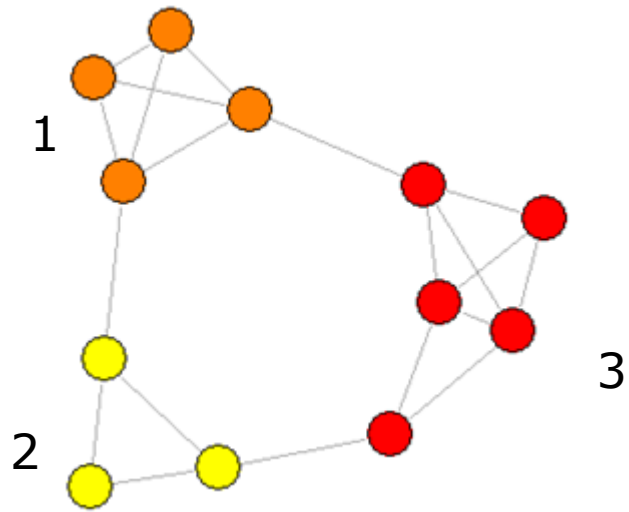
Modularité de la partition?



Modularité de la partition?

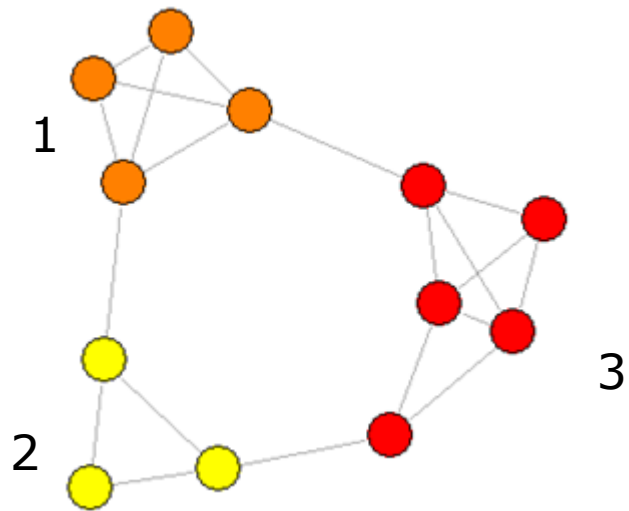
communauté	arêtes		$l_s/m$	$(d_s/2m)^2$	$l_s/m - (d_s/2m)^2$
	communautaires ( $l_s$ )	sommes des degrés ( $d_s$ )			
1					
2					
3					
m					

## Exemple



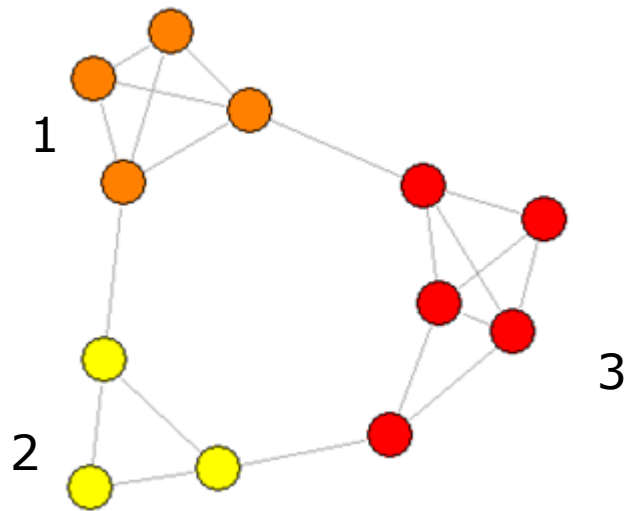
communauté	arêtes		$l_s/m$	$(d_s/2m)^2$	$l_s/m - (d_s/2m)^2$
	communautaires ( $l_s$ )	sommes des degrés ( $d_s$ )			
1	6				
2	3				
3	8				
m	20				

## Exemple



communauté	arêtes		$l_s/m$	$(d_s/2m)^2$	$l_s/m - (d_s/2m)^2$
	communautaires ( $l_s$ )	sommes des degrés ( $d_s$ )			
1	6	14			
2	3	8			
3	8	18			
m	20				

## Exemple



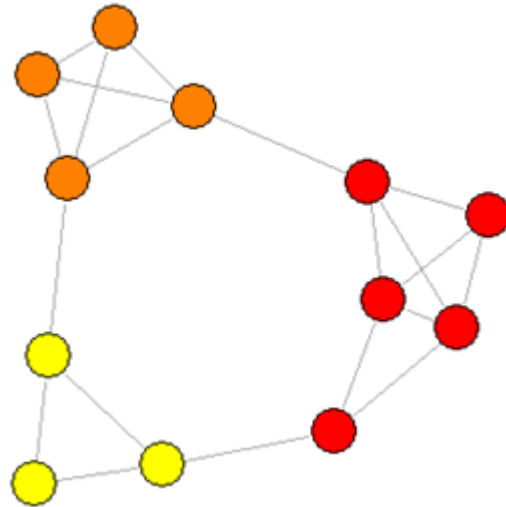
Modularité de la partition = 0.485

communauté	arêtes		$l_s/m$	$(d_s/2m)^2$	$l_s/m - (d_s/2m)^2$
	communautaires ( $l_s$ )	sommes des degrés ( $d_s$ )			
1	6	14	0,3	0,1225	0,1775
2	3	8	0,15	0,04	0,11
3	8	18	0,4	0,2025	0,1975
m	20				0,485



- Inconvénients de la modularité  $Q$ :
  - ◆ A partir de quelle valeur de *modularité* a t'on une structure en communautés dans le graphe?
    - Comme des graphes aléatoires peuvent avoir de grandes valeurs de  $Q$ , un graphe a une structure en communautés si sa modularité est significativement plus grande que celle de graphes aléatoires équivalents.
  - ◆ La *modularité*  $Q$  est sensible à la **taille du graphe**
    - pas comparable entre différents graphes.
  - ◆ Des graphes aléatoires peuvent avoir des modularités  $\gg 0$ .
  - ◆ **Mesure globale**, donc pas prise en compte d'hétérogénéités locales!

# Méthodes?



- Première approche pour identifier des communautés:
  - détecter les arêtes qui **connectent** des sommets appartenant à des communautés différentes et les supprimer.

- Première approche pour identifier des communautés:
  - détecter les arêtes qui connectent des sommets appartenant à des communautés différentes et les supprimer.
- Le point crucial est de définir une propriété qui caractérise les arêtes intercommunautaires (**centralité**).

<https://fr.wikipedia.org/wiki/Centralité>

- Première approche pour identifier des communautés:
  - détecter les arêtes qui connectent des sommets appartenant à des communautés différentes et les supprimer.
- Le point crucial est de définir une propriété qui caractérise les arêtes intercommunautaires (**centralité**).
- Remarque: ces méthodes sont, par construction, **hiérarchiques descendantes** (partitions représentées par un dendrogramme).
- Principe:
  1. Calculer la centralité de chaque arête.
  2. Supprimer l'arête de plus grande centralité.
  3. Recalculer la centralité des arêtes de chaque sous-graphes.
  4. Répéter l'étape 2 jusqu'à #communautés > seuil fixé par l'utilisateur.

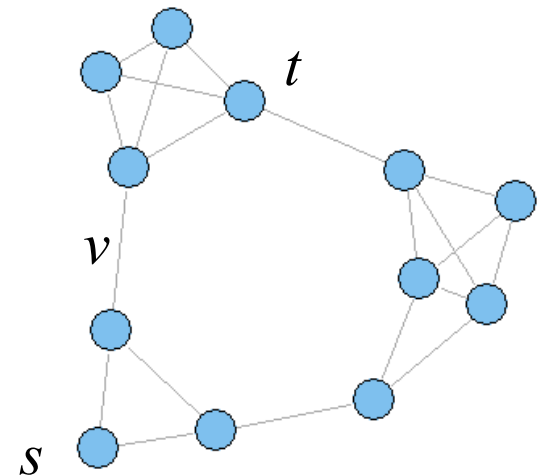
Exemple: Edge-betweenness (Girvan et Newman 2002).

- ♦ *Betweenness* : mesure de centralité.

*betweenness* d'une arête: somme sur les paires de sommets,

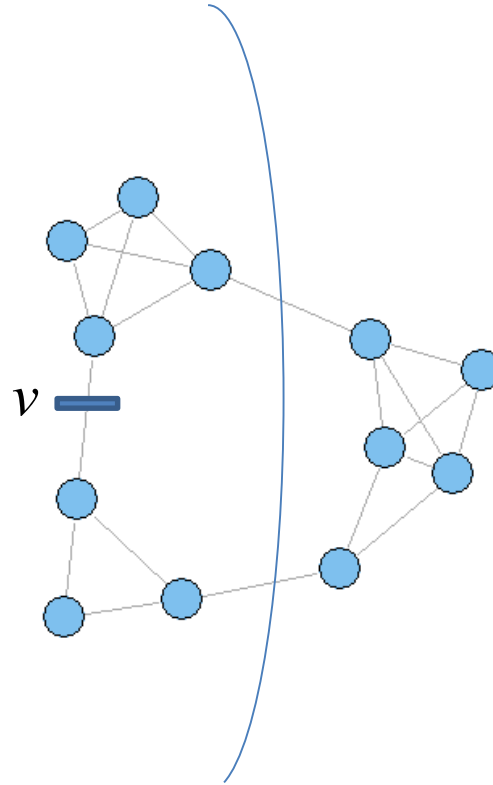
- ♦ du nombre de chemins les plus courts entre les sommets  $s$  et  $t$  qui passent par cette arête  $v$ :  $\sigma_{st}(v)$
- ♦ divisé par le nombre total de chemins les plus courts entre les sommets  $s$  et  $t$ :  $\sigma_{st}$

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

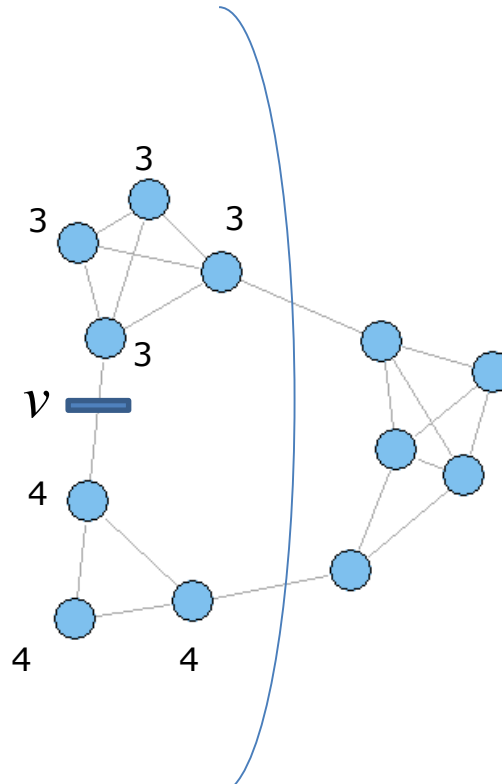


- ♦ La modularité  $Q$  peut être utilisée pour sélectionner la meilleur partition.

Nombre de chemins les plus courts entre les sommets  $s$  et  $t$  qui passent par l'arête  $v$  ?

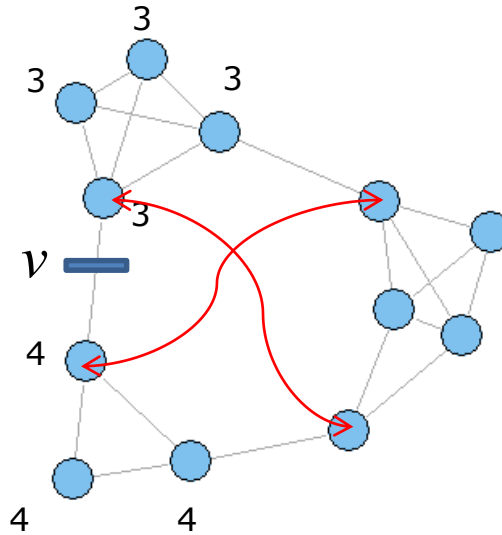


Nombre de chemins les plus courts entre les sommets  $s$  et  $t$  qui passent par cette arête ?



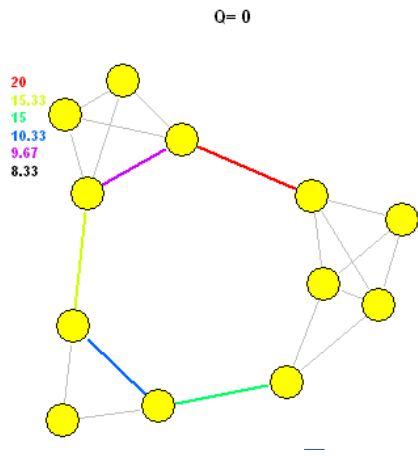


Arête de plus grande betweenness ?



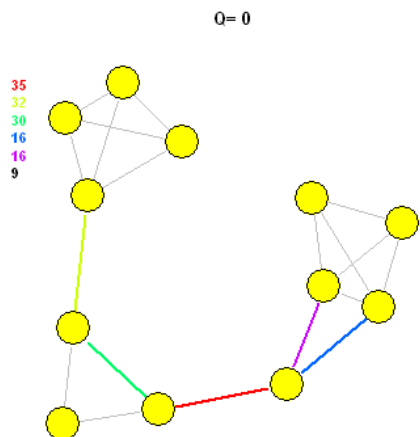
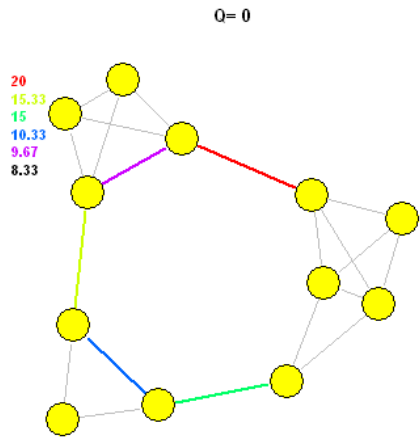
# Edgebetweenness, déroulement pas à pas

- Les *betweenness* des arêtes sont triées par ordre décroissant de *betweenness*.
- 20, 15.33, 15, 10.33, 9.676, 8.33...



# Edgebetweenness, déroulement pas à pas

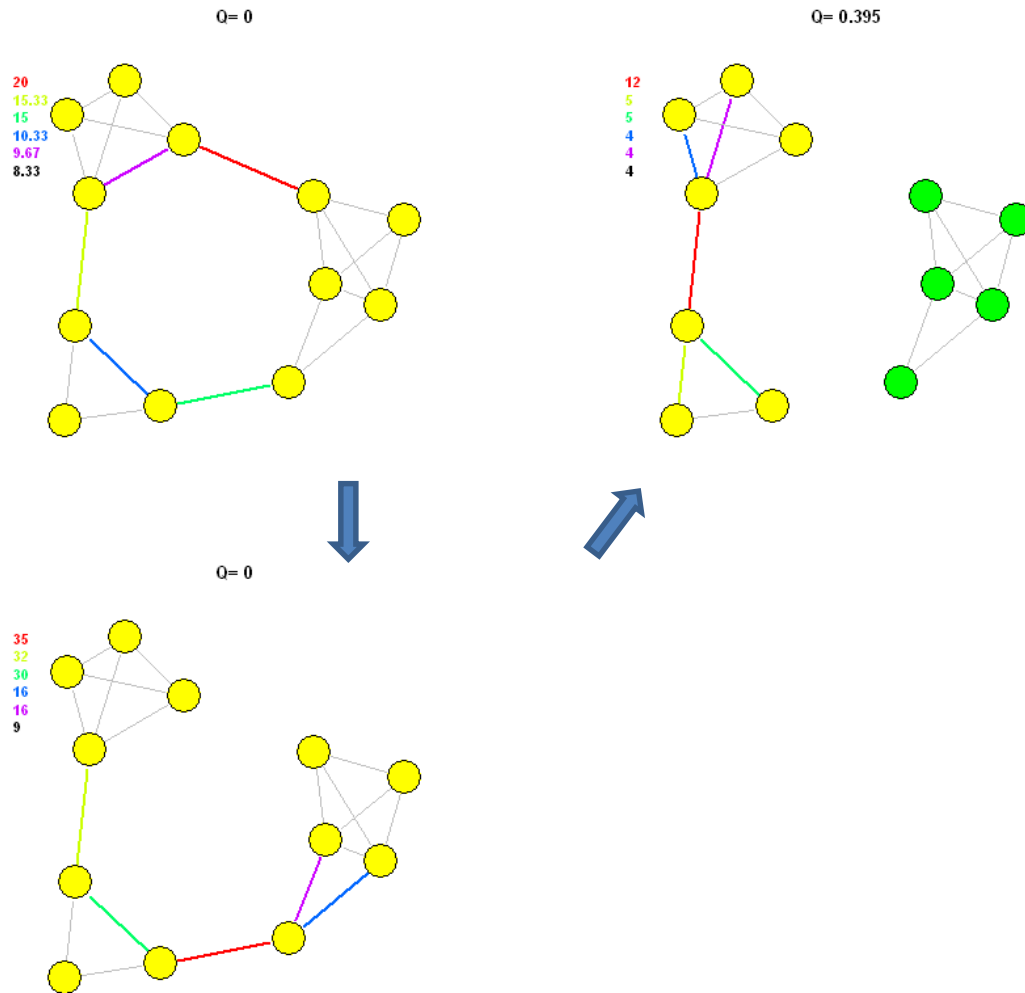
- les *betweenness* des arêtes sont triées par ordre décroissant.



- Suppression de l'arête de plus grande betweenness,
- mise à jour de la betweenness.

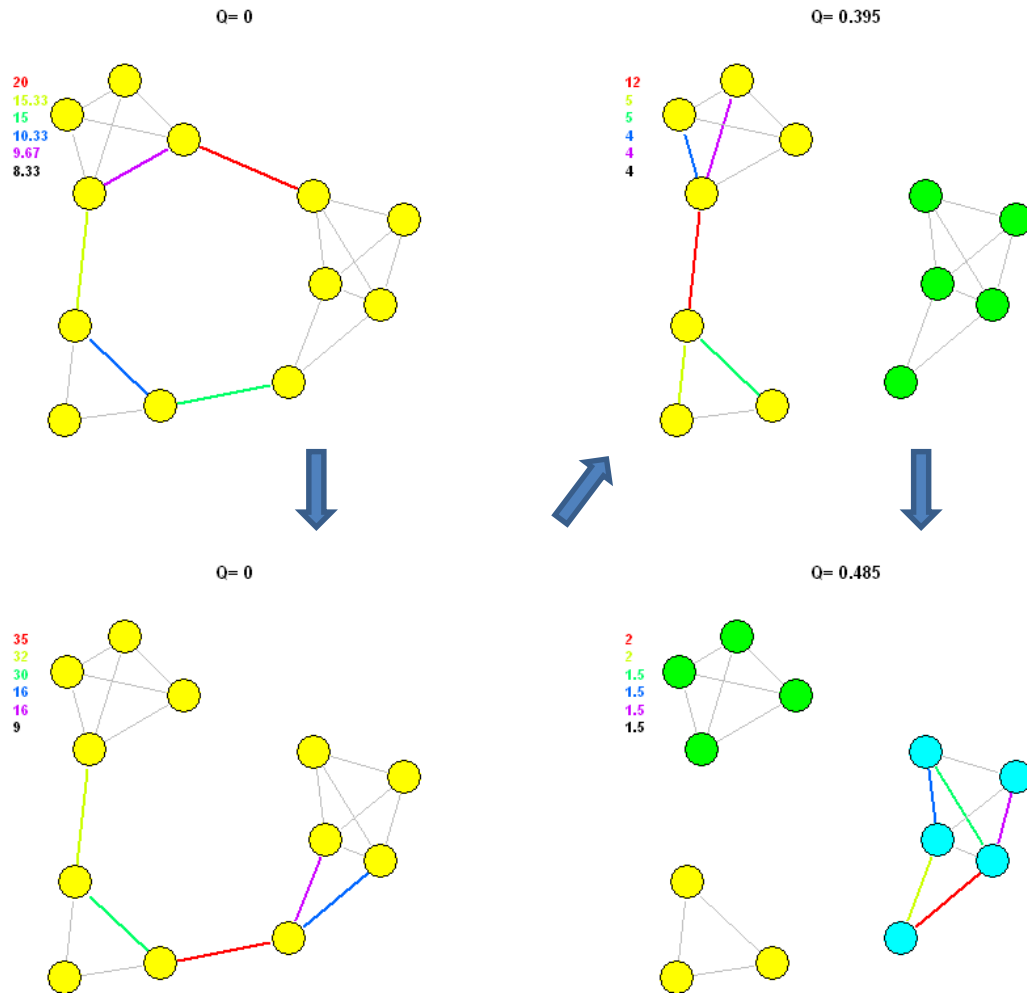
# Edgebetweenness, déroulement pas à pas

- On peut suivre l'évolution de  $Q$ .



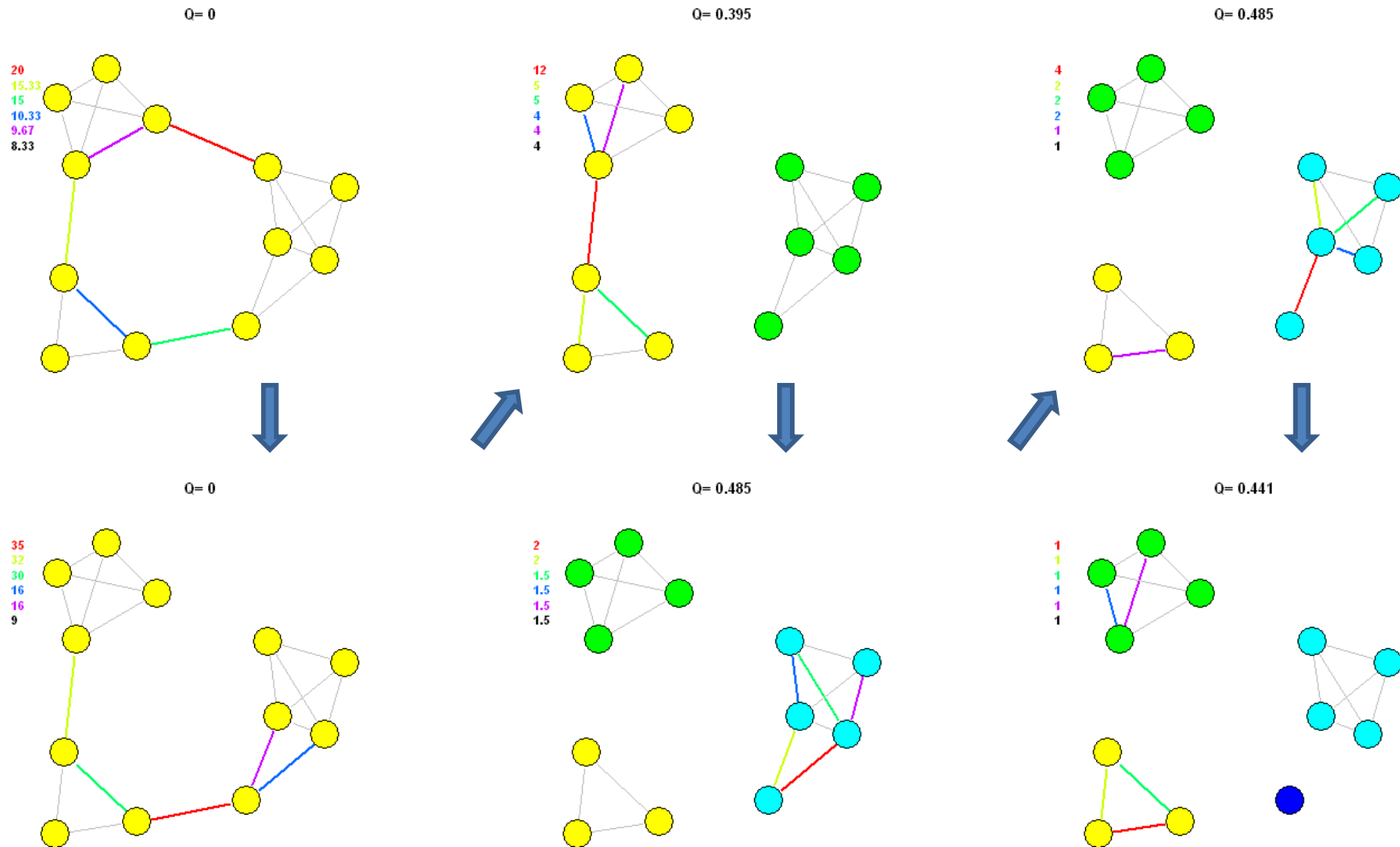
# Edgebetweenness, déroulement pas à pas

- On peut suivre l'évolution de  $Q$ .



# Edgebetweenness, déroulement pas à pas

- On peut suivre l'évolution de  $Q$ .





Si la modularité  $Q$  est un bon indicateur de la qualité d'une partition, alors la partition qui maximise  $Q$  doit être la meilleur!

- ♦ Problème: maximiser  $Q$  est impossible pour la grande majorité des graphes => approximations en un temps raisonnable.
- ♦ Méthode **hiérarchique ascendante** (agglomérative)
  - Exemple: Fastgreedy (Newman, 2004)

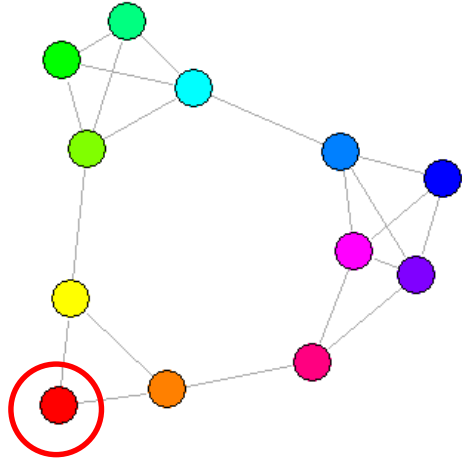


### Algorithme

1.  $n$  communautés (une communauté = 1 sommet), pas d'arête,  $Q=0$ ,
2. addition de l'arête qui maximise l'augmentation de  $Q \rightarrow n-1$  communautés,
3. addition des arêtes qui ne changent pas  $Q$  (arêtes intra-communautés)
4. répéter les étapes 2-3 jusqu'à obtenir tous les sommets dans une seule communauté.

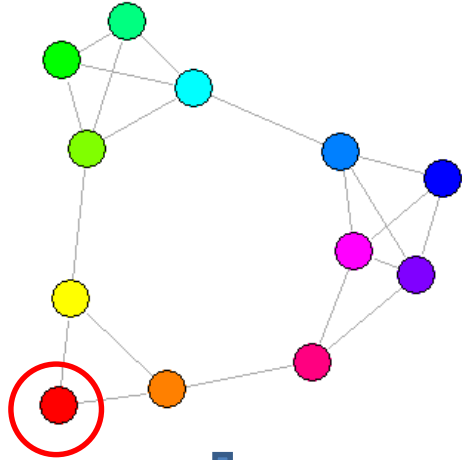
- Avantages:
  - simplicité et rapidité (grâce à des heuristiques!),
  - traitement de graphes de très grande taille.
- Inconvénients majeurs:
  1.  $Q$  peut s'écarter de l'optimum (séquentiel),
  2.  $Q$  est estimée sur la globalité du graphe.

$Q = -0.086$

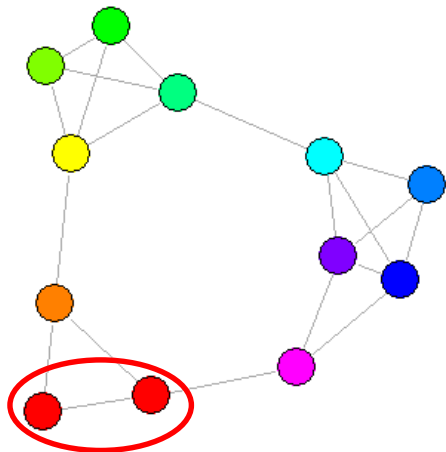


# Fastgreedy, déroulement pas à pas

$Q = -0.086$

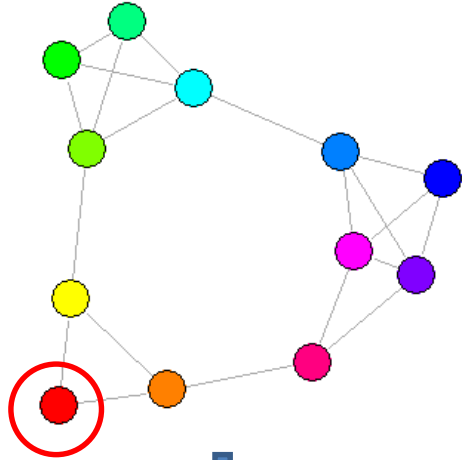


$Q = -0.044$

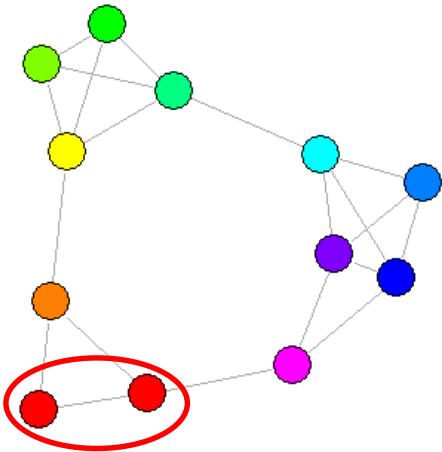


# Fastgreedy, déroulement pas à pas

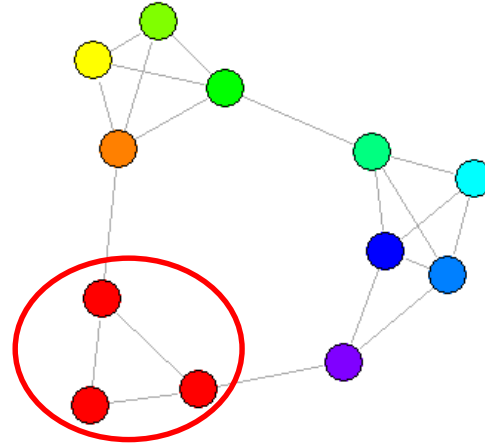
$Q = -0.086$



$Q = -0.044$

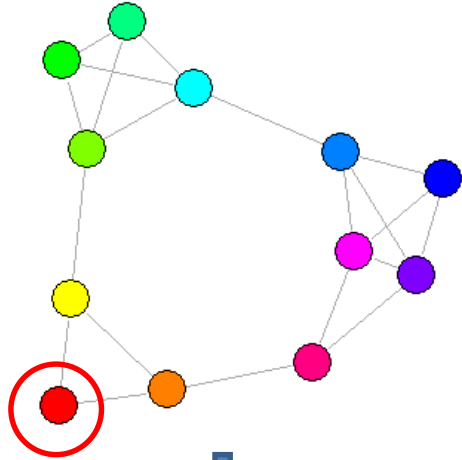


$Q = 0.037$

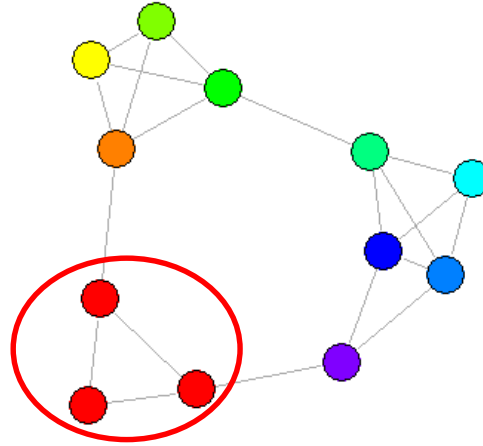


# Fastgreedy, déroulement pas à pas

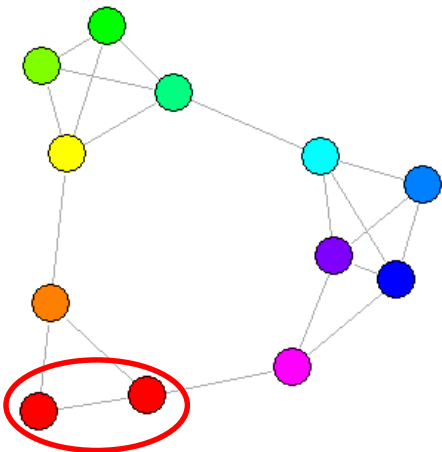
$Q = -0.086$



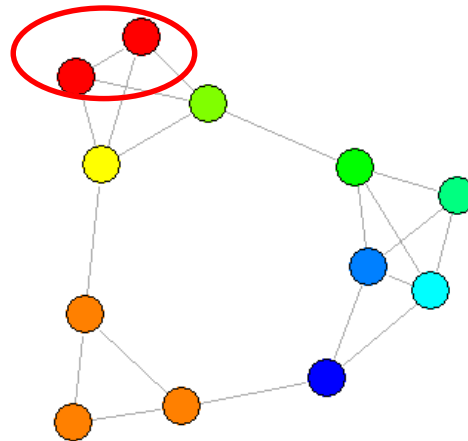
$Q = 0.037$



$Q = -0.044$

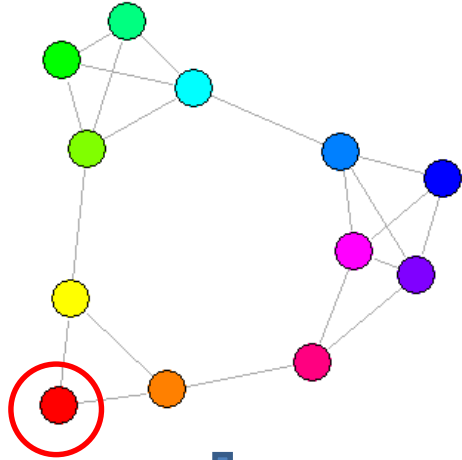


$Q = 0.076$

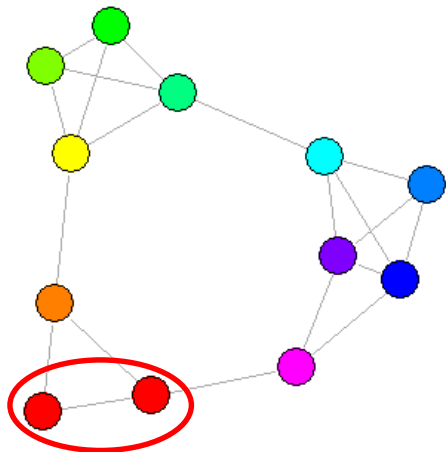


# Fastgreedy, déroulement pas à pas

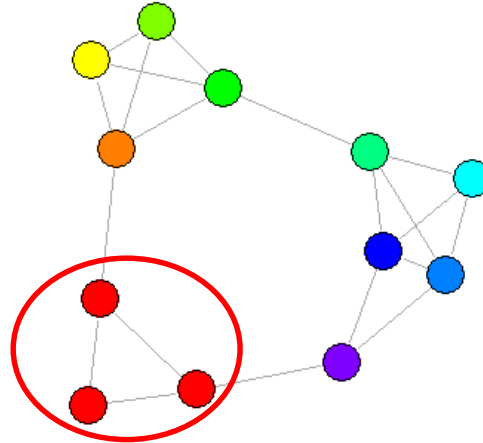
$Q = -0.086$



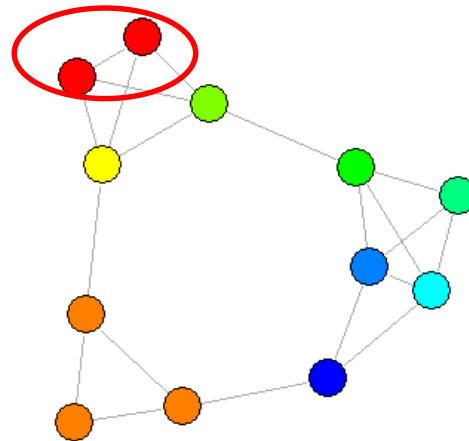
$Q = -0.044$



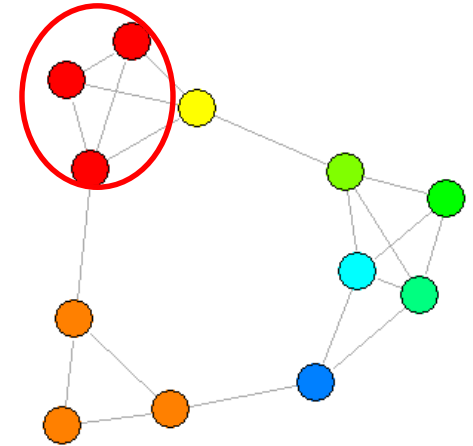
$Q = 0.037$



$Q = 0.076$

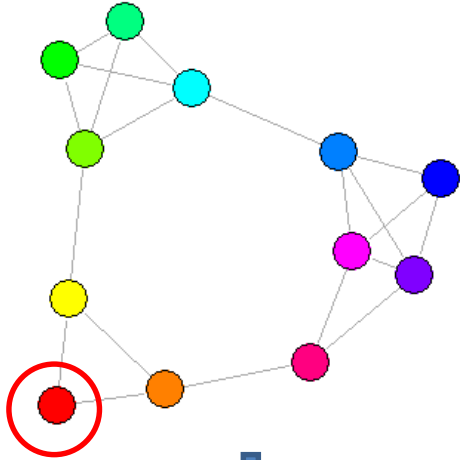


$Q = 0.146$

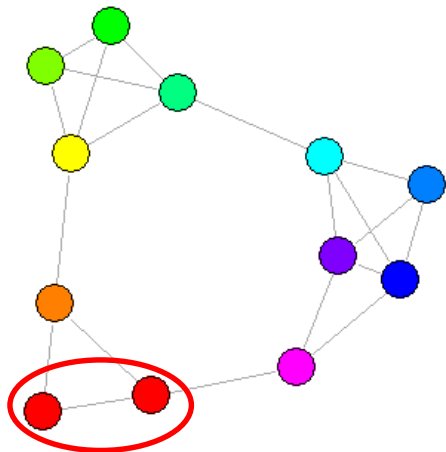


# Fastgreedy, déroulement pas à pas

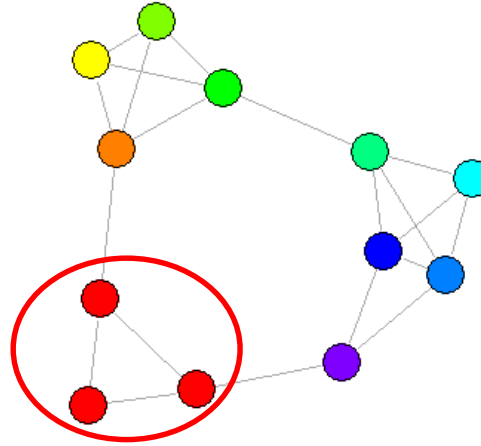
$Q = -0.086$



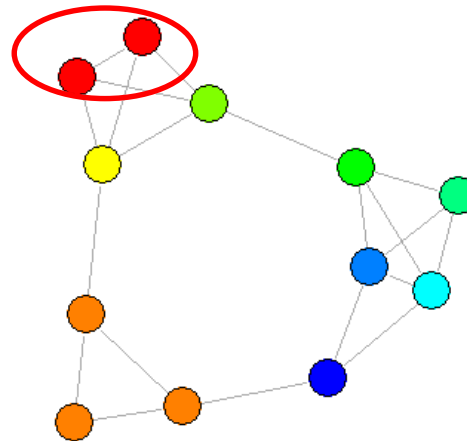
$Q = -0.044$



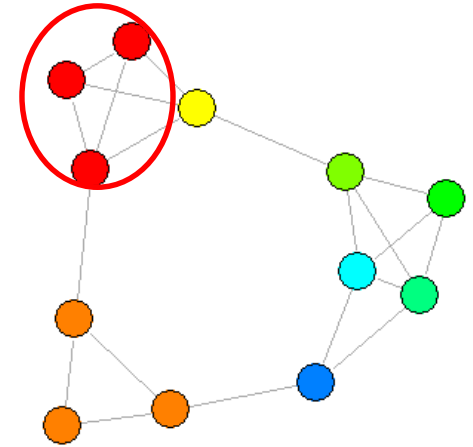
$Q = 0.037$



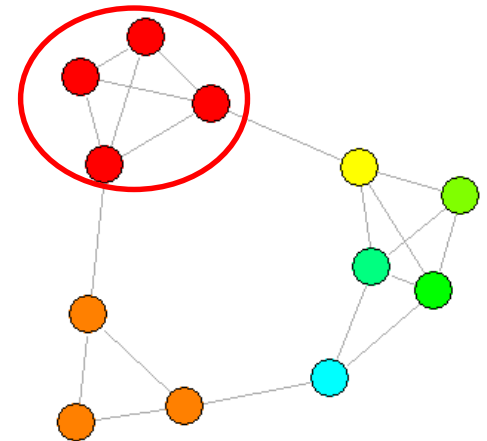
$Q = 0.076$



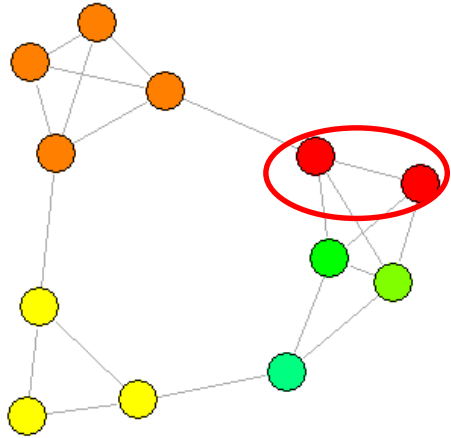
$Q = 0.146$



$Q = 0.246$



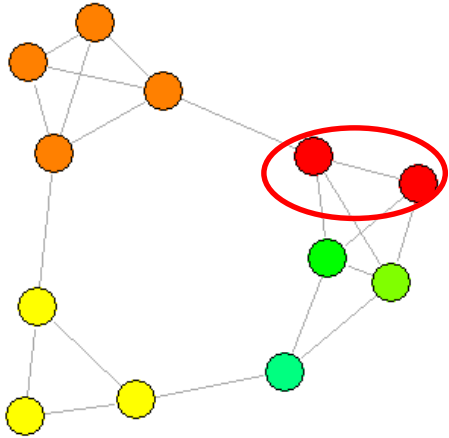
Q= 0.281



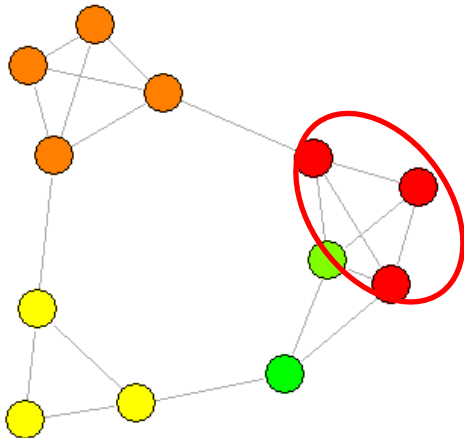


# Fastgreedy, déroulement pas à pas

$Q = 0.281$

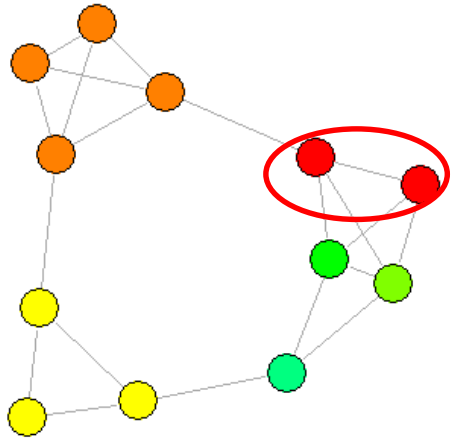


$Q = 0.346$

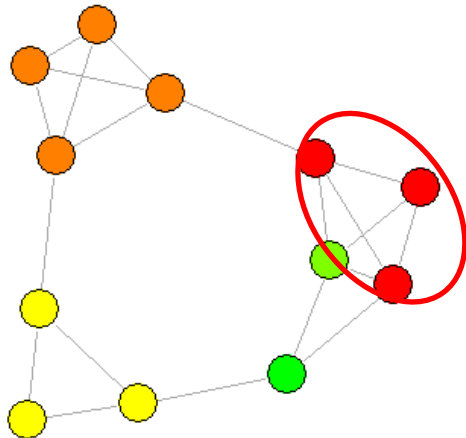


# Fastgreedy, déroulement pas à pas

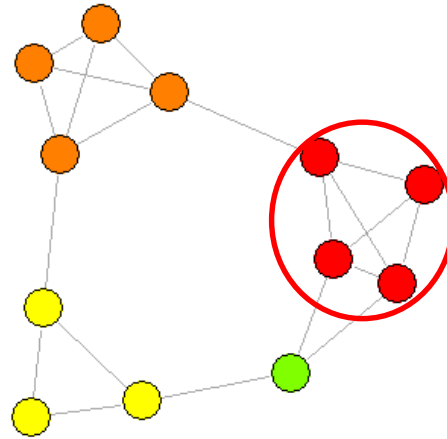
Q= 0.281



Q= 0.346

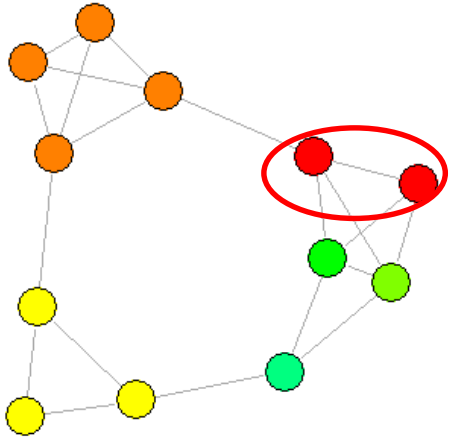


Q= 0.441

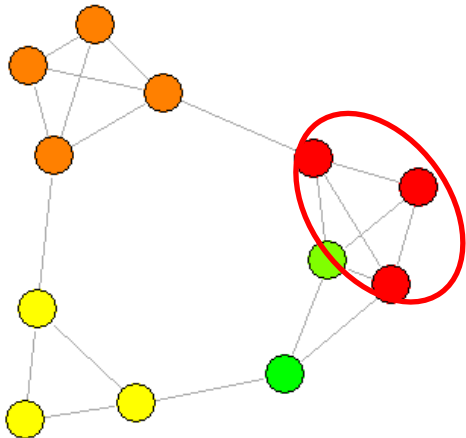


# Fastgreedy, déroulement pas à pas

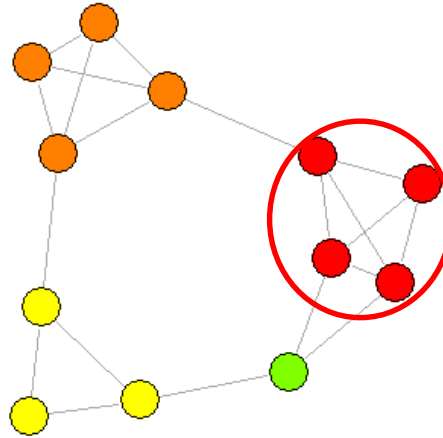
Q= 0.281



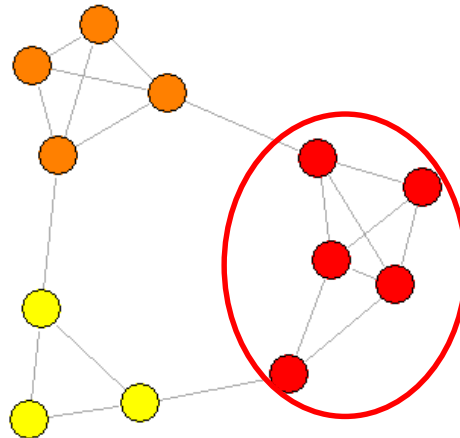
Q= 0.346



Q= 0.441

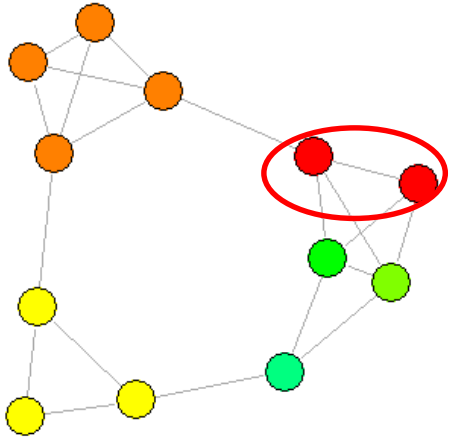


Q= 0.485

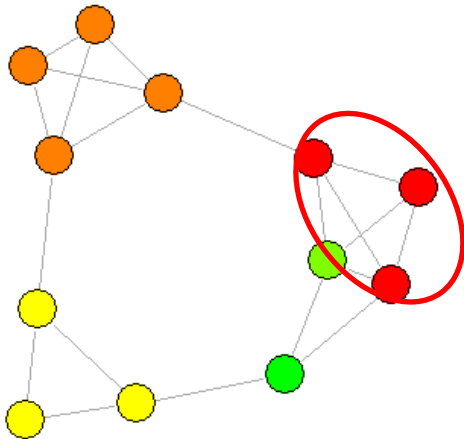


# Fastgreedy, déroulement pas à pas

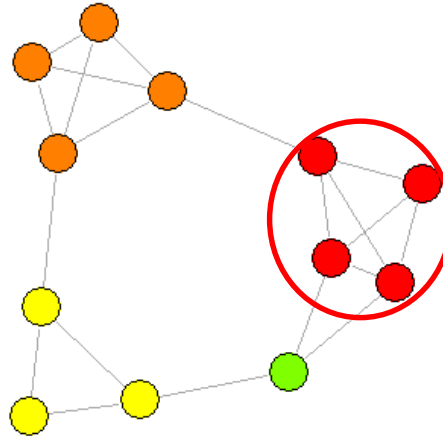
Q= 0.281



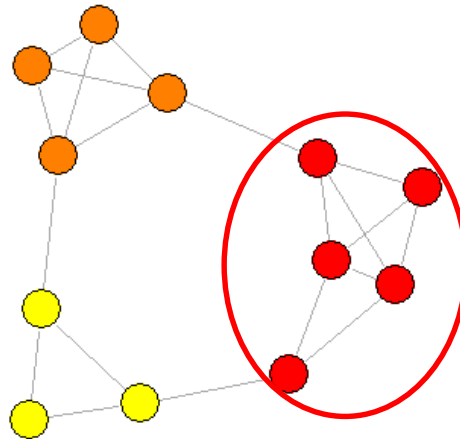
Q= 0.346



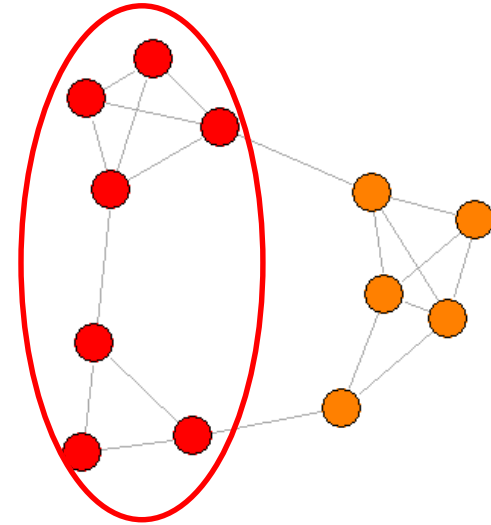
Q= 0.441



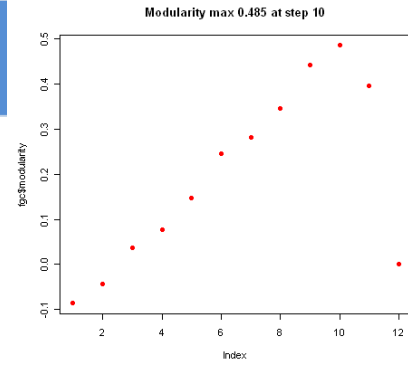
Q= 0.485



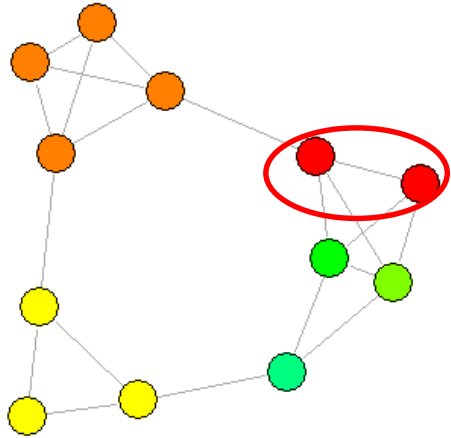
Q= 0.395



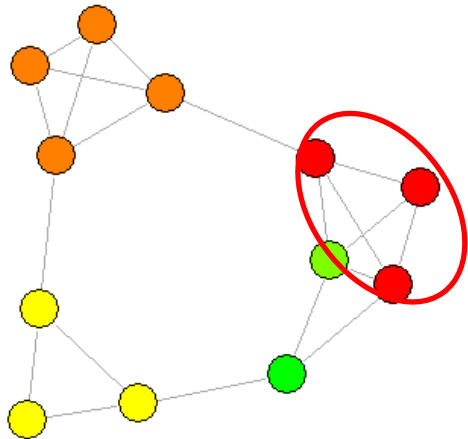
# Fastgreedy, déroulement pas à pas



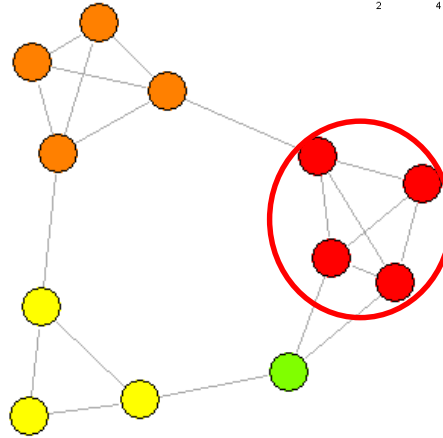
Q= 0.281



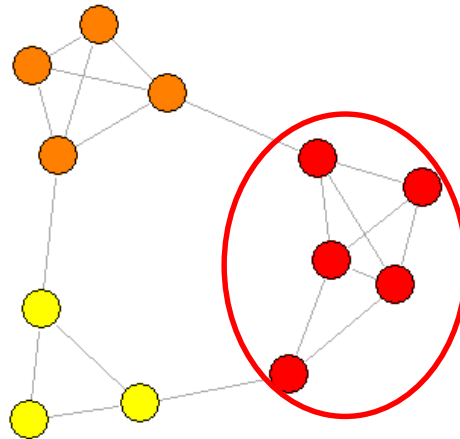
Q= 0.346



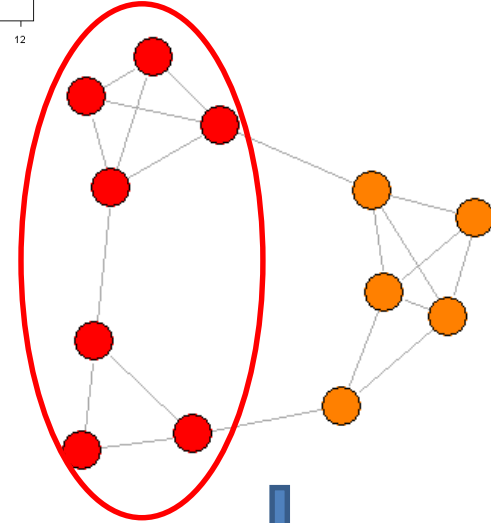
Q= 0.441



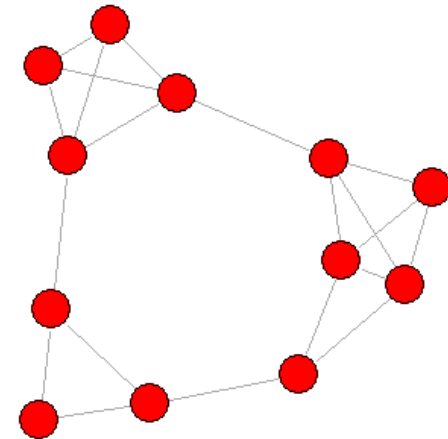
Q= 0.485



Q= 0.395



Q= 0



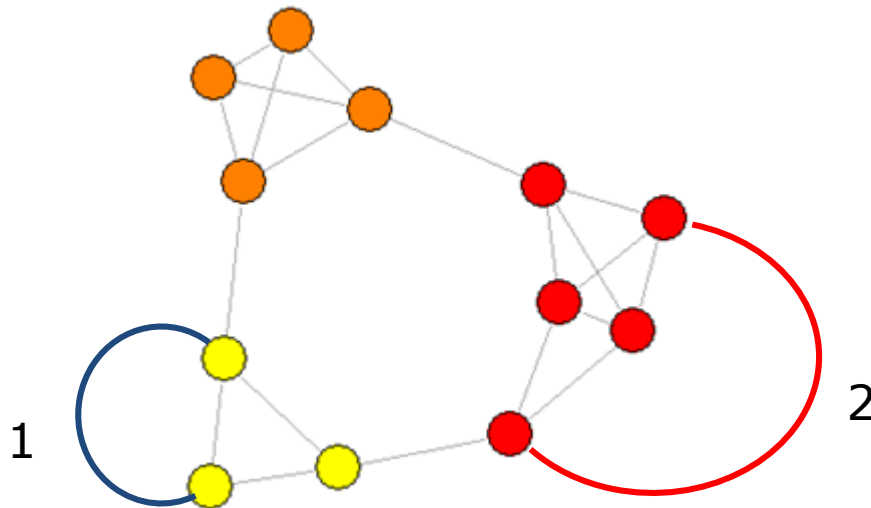


- Méthode probabiliste d'optimisation d'une fonction  $F$  (Hamiltonian par exemple).
- Deux types de mouvements sont pris en compte:
  1. mouvements locaux, où un seul sommet est déplacé d'une communauté à l'autre, communauté prise au hasard,
  2. mouvements globaux, fusion/fission de communautés.
- Donne de très bon résultats mais au dépend de temps de calculs prohibitifs pour les gros graphes.

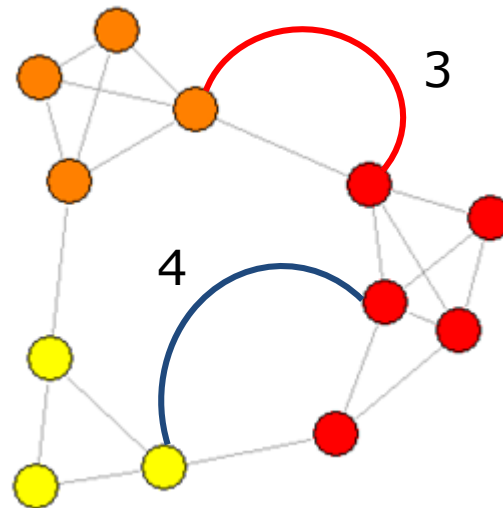
- Une **fonction de qualité** pour l'assignement des nœuds à une communauté doit suivre un principe simple:
  - ♦ grouper ensemble les sommets qui sont liés et séparer ceux qui ne le sont pas!
- Reichardt et Bornholdt proposent 4 contraintes:



- Une fonction de qualité pour l'assignement des nœuds à une communauté doit suivre un principe simple:
  - ♦ grouper ensemble les sommets qui sont liés et séparer ceux qui ne le sont pas!
- Reichardt et Bornholdt proposent 4 contraintes:
  1. gratifier les arêtes internes entre sommets de la même communauté,
  2. pénaliser les arêtes absentes entre sommets de la même communauté,



- Une fonction de qualité pour l'assignement des nœuds à une communauté doit suivre un principe simple:
  - ♦ grouper ensemble les sommets qui sont liés et séparer ceux qui ne le sont pas!
- Reichardt et Bornholdt proposent 4 contraintes:
  1. gratifier les arêtes internes entre sommets de la même communauté,
  2. pénaliser les arêtes absentes entre sommets de la même communauté,
  3. pénaliser les arêtes entre sommets de communautés différentes,
  4. gratifier les arêtes absentes entre sommets de communautés différentes.



- D'où la fonction suivante:

$$H(\{\sigma\}) = -\sum_{i \neq j} a_{ij} A_{ij} \delta(\sigma_i, \sigma_j) + \sum_{i \neq j} b_{ij} (1 - A_{ij}) \delta(\sigma_i, \sigma_j) \\ + \sum_{i \neq j} c_{ij} A_{ij} (1 - \delta(\sigma_i, \sigma_j)) - \sum_{i \neq j} d_{ij} (1 - A_{ij}) (1 - \delta(\sigma_i, \sigma_j))$$

$$H(\{\sigma\}) = - \text{internal links} + \text{internal non-links} \\ + \text{external links} - \text{external non-links}$$

- Où  $A_{ij}$  est la matrice d'adjacence de  $G$ ,
- $\sigma_i \in \{1, 2, 3, \dots, q\}$  sont les index des groupes de sommets  $i$  dans  $G$
- $a_{ij}, b_{ij}, c_{ij}, d_{ij}$  sont les poids des différentes contributions.
- La fonction  $\delta$  est égale à 1 si  $i$  et  $j$  appartiennent à la même communauté et 0 sinon.

## Simplifications:

- Si les liens intra et inter ont le même poids,  $a_{ij} = c_{ij}$  et  $b_{ij} = d_{ij}$   
alors il suffit de considérer les liens intra et les liens absents.

- Il reste à choisir  $a_{ij}$  et  $b_{ij}$ . Un choix pratique est de les exprimer en fonction de la probabilité  $p_{ij}$  qu'un lien existe entre les sommets  $i$  et  $j$ .

$$a_{ij} = 1 - \gamma p_{ij} \text{ et } b_{ij} = \gamma p_{ij}$$

- Simplification de  $H$ :

$$H(\{\sigma\}) = -2 \sum_{i \neq j} (A_{ij} - \gamma p_{ij}) \delta(\sigma_i, \sigma_j) + \sum_{i \neq j} A_{ij} - \sum_{i \neq j} \gamma p_{ij}$$

et avec

$$p_{ij} = \frac{k_i k_j}{2M}$$

$M$  : nombre total d'arêtes dans le graphe et  $\gamma = 1$

$$H(\{\sigma\}) = -2 \sum_{i \neq j} \left( A_{ij} - \frac{k_i k_j}{2M} \right) \delta(\sigma_i, \sigma_j)$$

- Relation avec la modularité  $Q$  ?

- Relation entre Hamiltonian et modularité  $Q$ ?
- Si  $\gamma = 1$ :

$$H(\{\sigma\}) = -2 \sum_{i \neq j} \left( A_{ij} - \frac{k_i k_j}{2M} \right) \delta(\sigma_i, \sigma_j)$$

- On a vu que l'on peut écrire la modularité  $Q$  d'une façon légèrement différente:

$$Q = \frac{1}{2M} \sum_{i \neq j} \left( A_{ij} - \frac{k_i k_j}{2M} \right) \delta(\sigma_i, \sigma_j)$$

- Ainsi:

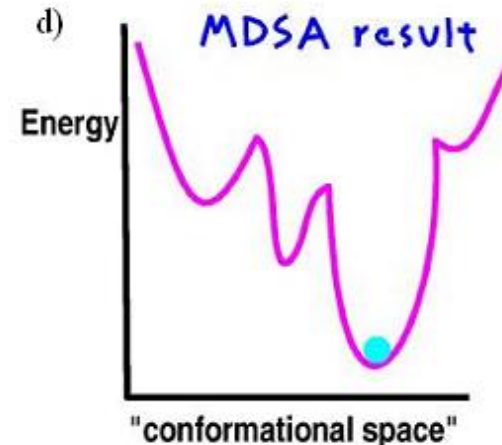
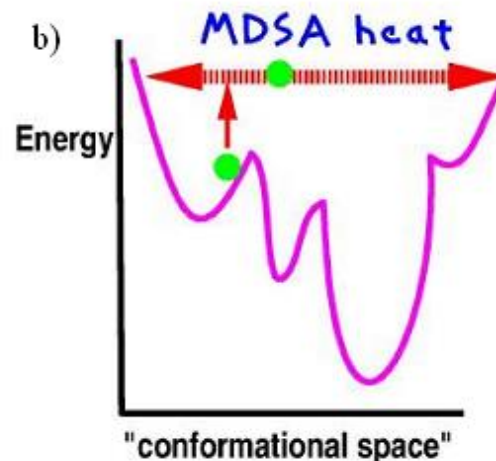
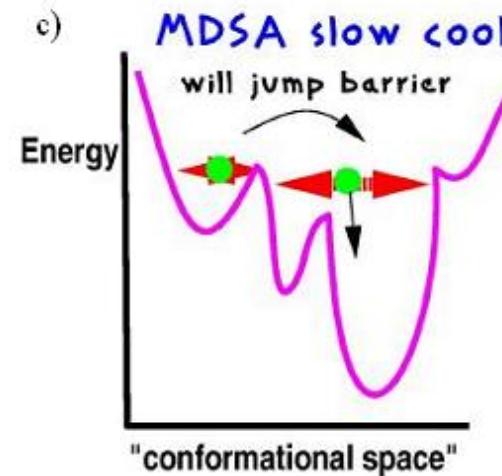
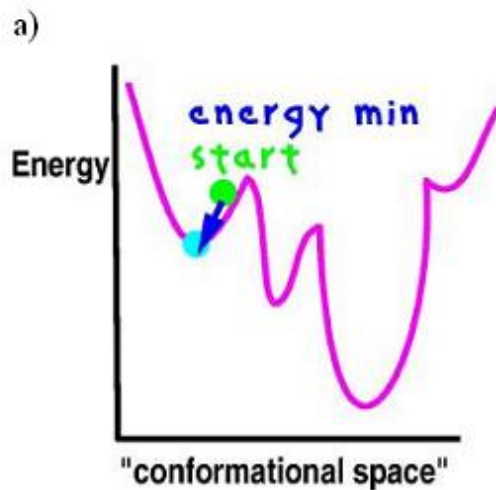
$$Q = -\frac{1}{M} H(\{\sigma\})$$

- **Maximiser  $Q$  revient à minimiser  $H$ .**
- Méthode de recuit simulé est utilisée pour optimiser l'assignement des sommets aux groupes.

# Méthode de recuit simulé

Méthode de recuit simulé = alternances de cycles de refroidissement lent et de réchauffage qui ont pour effet de minimiser l'énergie du système,

-> permet de trouver les configurations qui minimisent la fonction  $H$  (Metropolis-Hastings).



- Une température initiale permet de fixer le **nombre de sommets** qui vont pouvoir changer de communauté.
- Le système est alors refroidi pas à pas, jusqu'à un seuil fixé ou s'il n'y a plus de changement observé.
  - ♦ A chaque pas,
    - si un changement améliore la fonction, il est conservé,
    - si non, il est retenu avec une petite probabilité.

Cette procédure est non-hiérarchique et non-déterministe. Elle peut conduire à différentes solutions.

- Une température initiale permet de fixer le nombre de sommets qui vont pouvoir changer de communauté.
- Le système est alors refroidi pas à pas, jusqu'à un seuil fixé ou s'il n'y a plus de changement observé.
  - ♦ A chaque pas,
    - si un changement améliore la fonction, il est conservé,
    - si non, il est retenu avec une petite probabilité.

Cette procédure est non-hiérarchique et non-déterministe. Elle peut conduire à différentes solutions.

En répétant la procédure plusieurs fois, il est alors possible d'identifier les solutions les plus stables (fréquentes).

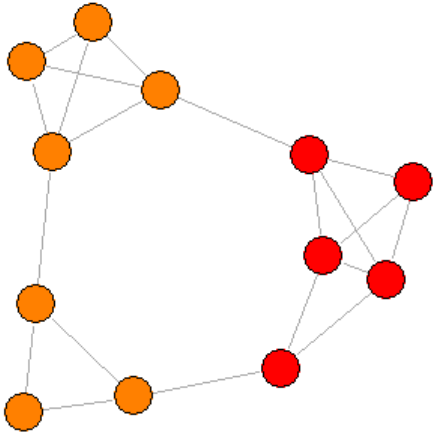
- Un critère peut être la fréquence des pairs de sommets dans les différentes solutions.



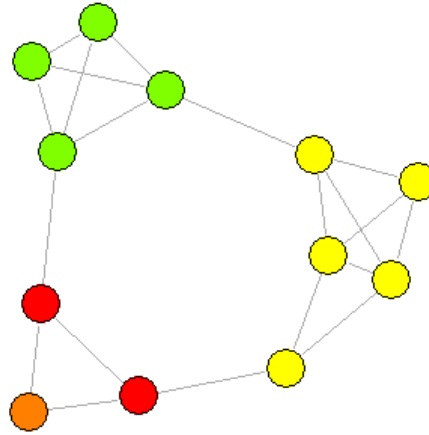
- Le paramètre **gamma** permet de pondérer l'importance de la présence/absence d'arêtes dans une communauté.
  - ♦ La valeur par défaut, 1.0 donne une importance égale aux deux types de liens.
  - ♦ Les plus petites valeurs donnent aux liens observés une plus grande importance qu'aux liens manquants.
- Le choix de ce paramètre peut être difficile *a priori*, une solution est d'expérimenter une plage de valeurs et de retenir la solution qui minimise  $H$ .

# Spinglass: itérations avec valeurs du paramètre gamma différentes

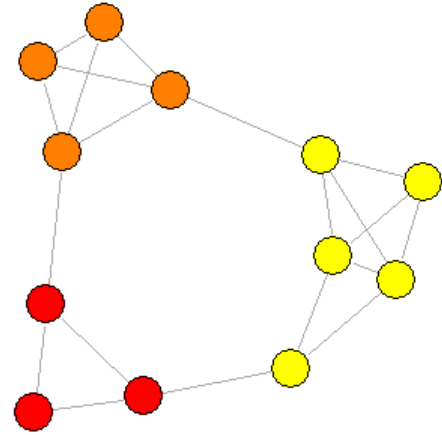
Q= 0.395 gamma= 0.1



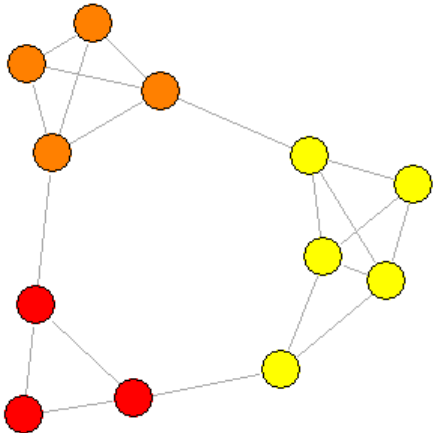
Q= 0.4 gamma= 0.4



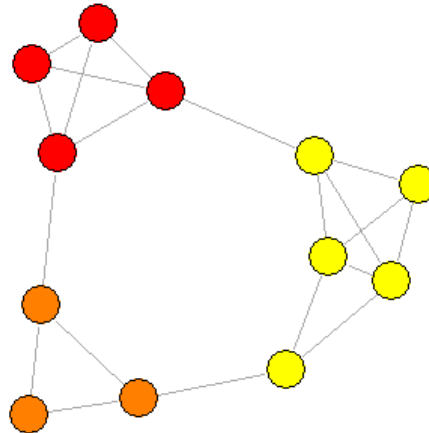
Q= 0.485 gamma= 1



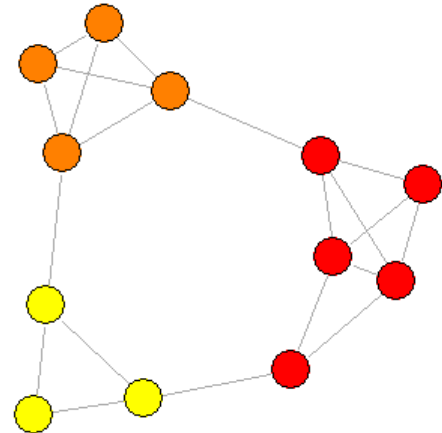
Q= 0.485 gamma= 0.2



Q= 0.485 gamma= 0.6

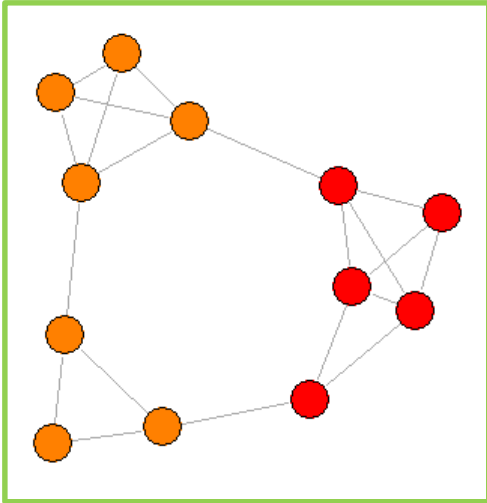


Q= 0.485 gamma= 1.2

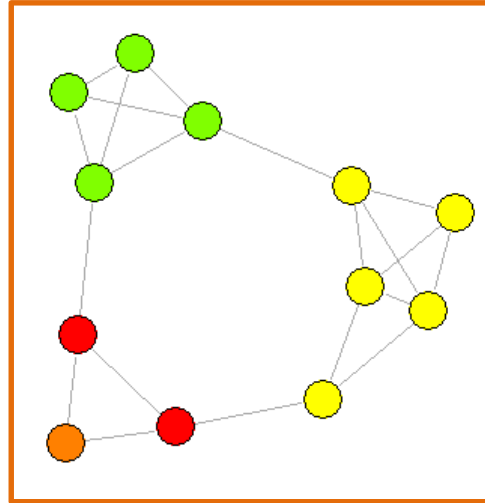


# Spinglass: itérations avec valeurs du paramètre gamma différentes

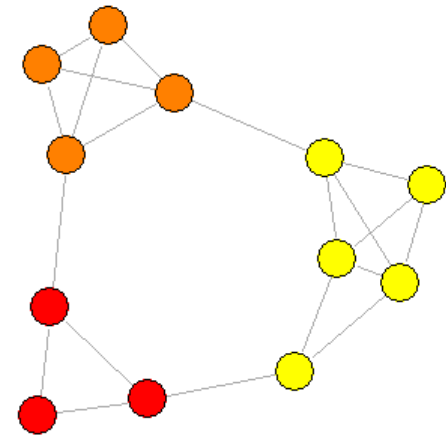
Q= 0.395 gamma= 0.1



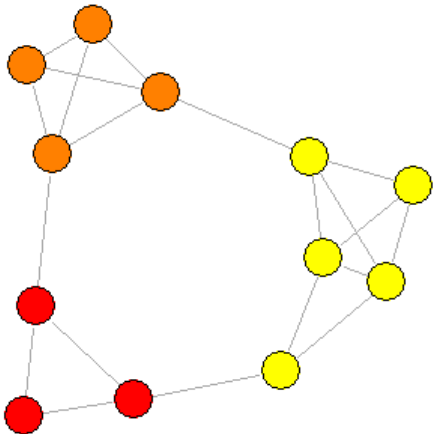
Q= 0.4 gamma= 0.4



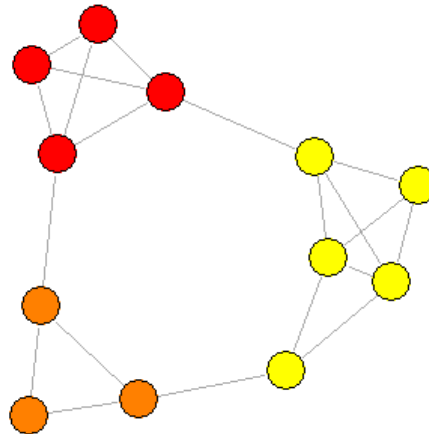
Q= 0.485 gamma= 1



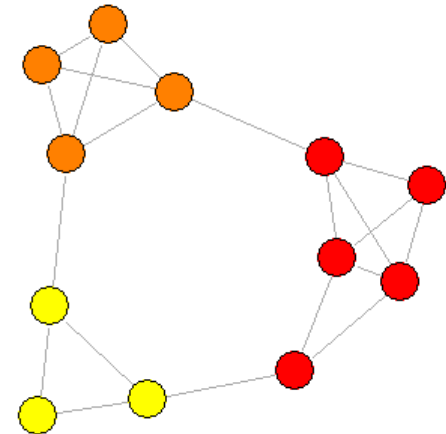
Q= 0.485 gamma= 0.2



Q= 0.485 gamma= 0.6

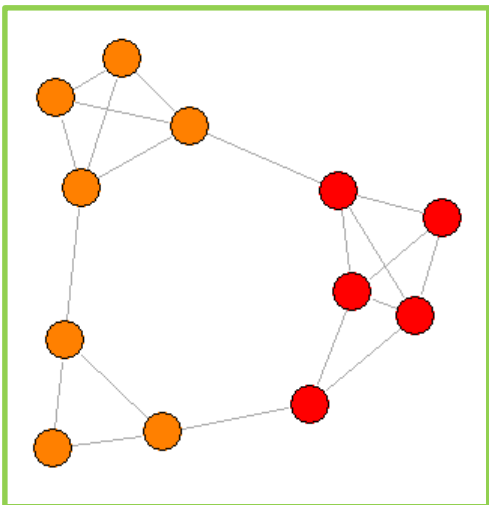


Q= 0.485 gamma= 1.2

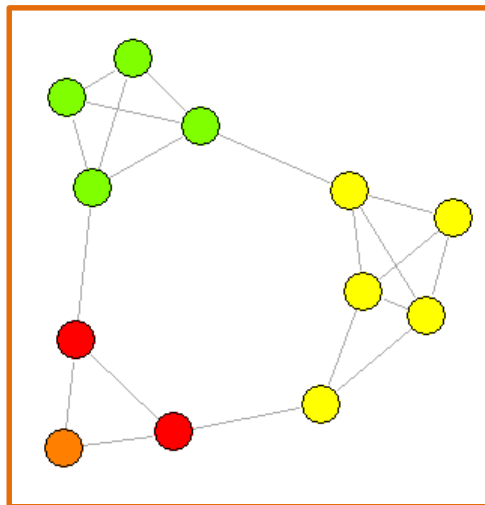


# Spinglass: itérations avec valeurs du paramètre gamma différentes

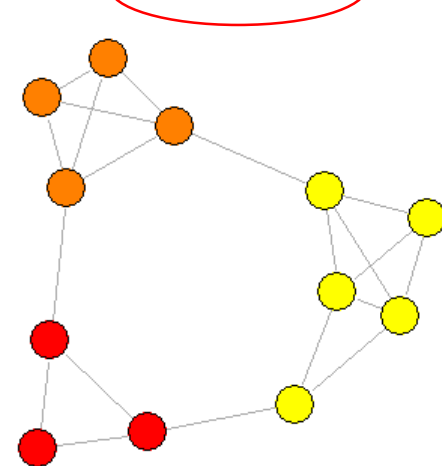
$Q=0.395$   $\gamma=0.1$



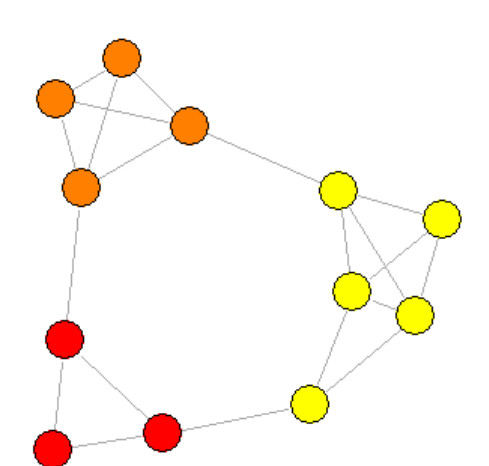
$Q=0.4$   $\gamma=0.4$



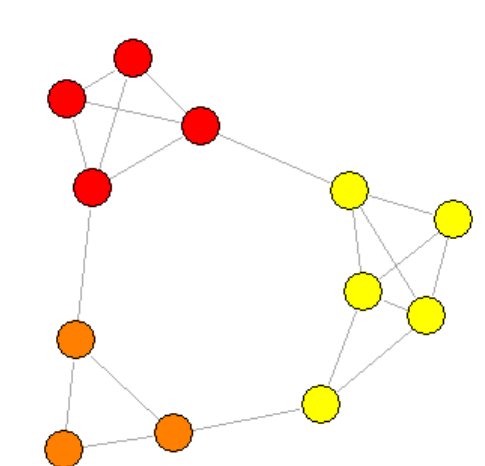
$Q=0.485$   $\gamma=1$



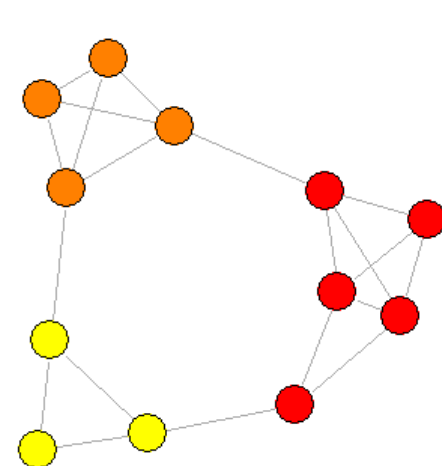
$Q=0.485$   $\gamma=0.2$



$Q=0.485$   $\gamma=0.6$



$Q=0.485$   $\gamma=1.2$





- Utiliser la **marche aléatoire** pour trouver des communautés vient de l'observation qu'un « marcheur » va passer plus de temps dans une communauté où le nombre de d'arêtes est plus grand qu'entre communautés (Pons et Latapy, 2005).
- Marche aléatoire similaire à un **processus de diffusion**.
- Sur le graphe, à chaque pas le marcheur est sur un sommet et va passer au sommet suivant choisi au hasard parmi les sommets voisins accessibles.
- La séquence des sommets visités est une **chaîne de Markov** dont les sommets sont les états de la chaîne de Markov.
- A chaque pas, le marcheur a une probabilité de  $1/d(i)$  d'aller vers l'un des voisins du sommet  $i$ .

$$P_{ij} = \frac{A_{ij}}{d(i)}$$

- où  $A_{ij}$  est la matrice d'adjacence de  $G$ ,
- $d(i)$  est le degré de  $i$ ,

# Exemple de marche aléatoire

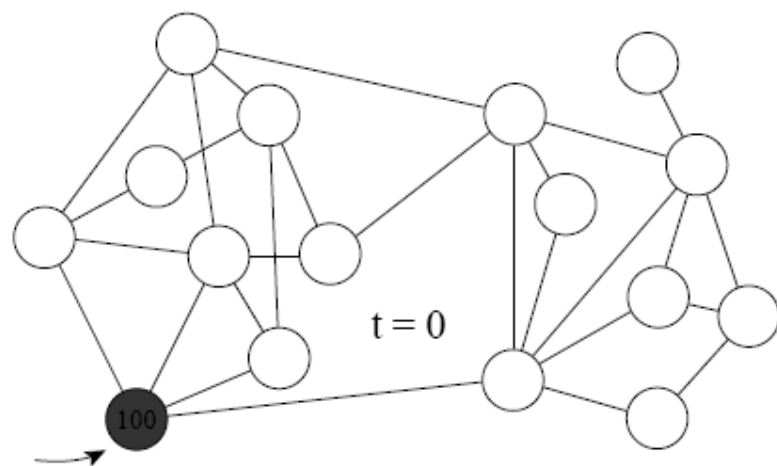


FIG. 2.1 – Exemple de dynamique d’une marche aléatoire sur un graphe non-pondéré. Le marcheur part du sommet indiqué par la flèche, les probabilités de position aux temps  $t = 0, 1, 2$  et  $3$  sont indiquées en pourcentage sur chaque sommet.

## Exemple de marche aléatoire

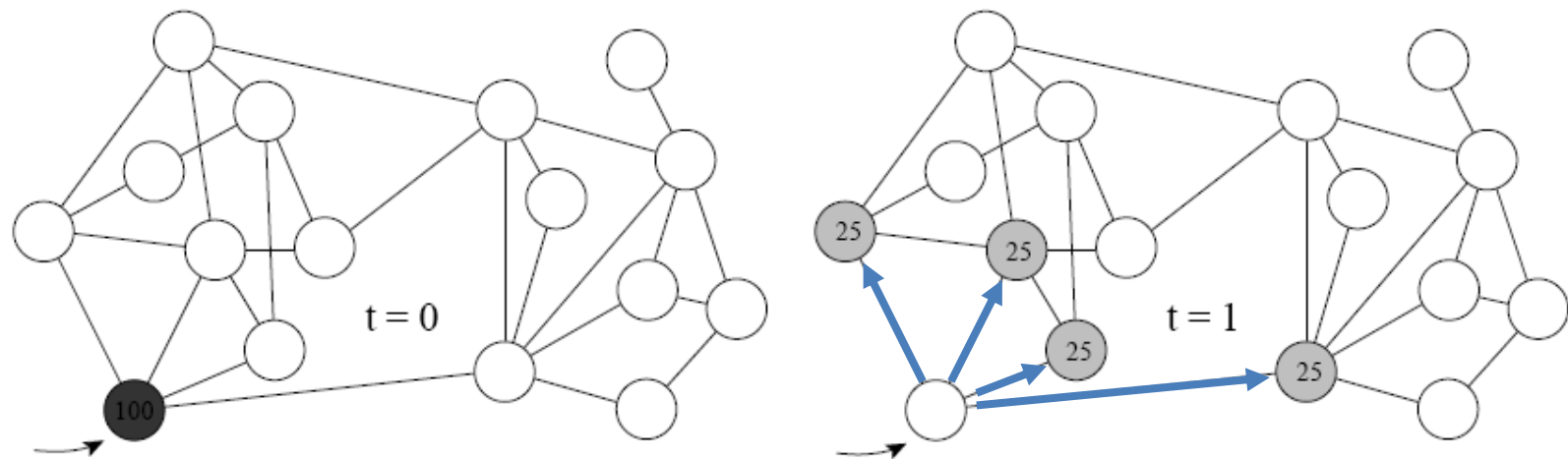


FIG. 2.1 – Exemple de dynamique d'une marche aléatoire sur un graphe non-pondéré. Le marcheur part du sommet indiqué par la flèche, les probabilités de position aux temps  $t = 0, 1, 2$  et  $3$  sont indiquées en pourcentage sur chaque sommet.



# Exemple de marche aléatoire

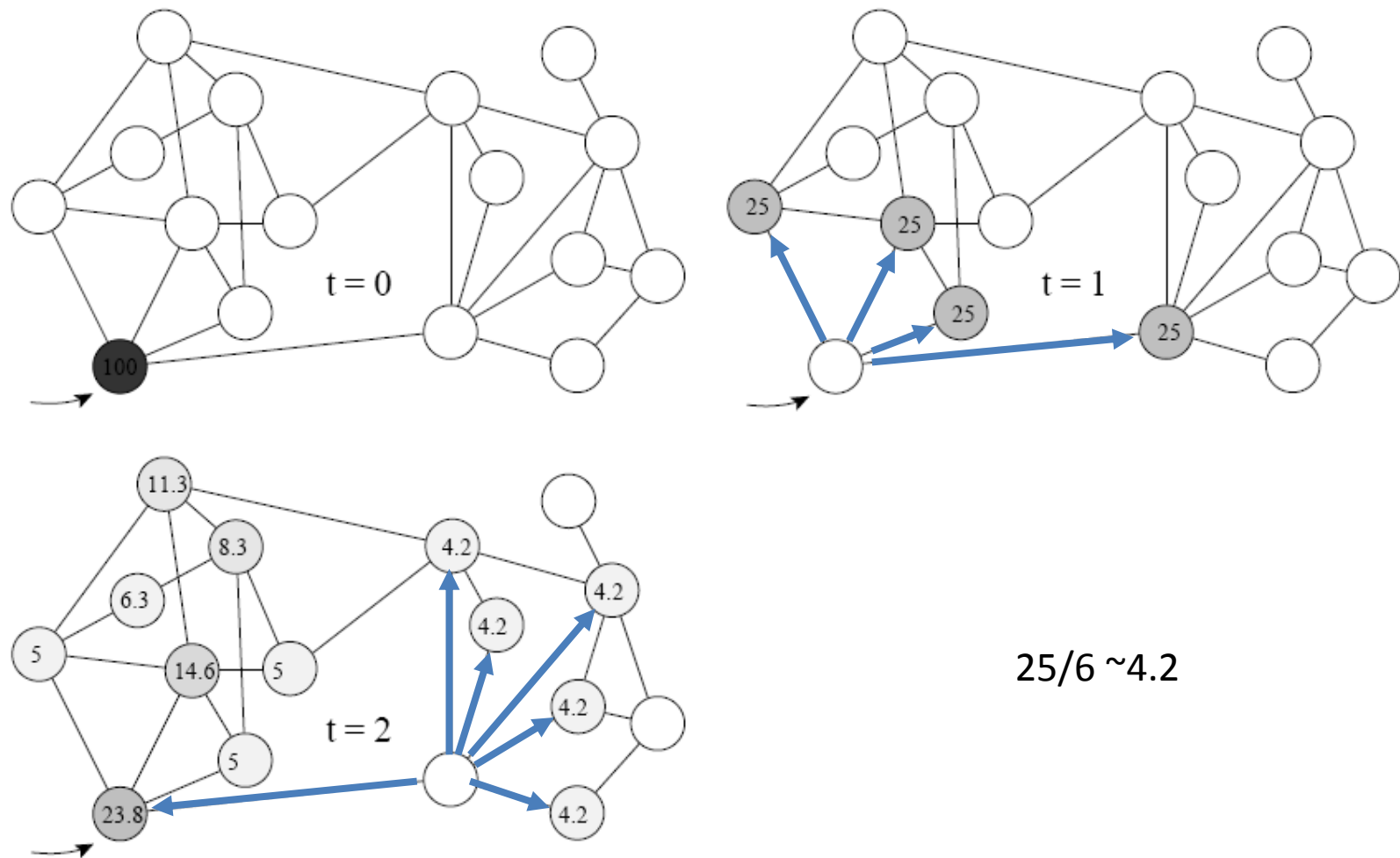


FIG. 2.1 – Exemple de dynamique d'une marche aléatoire sur un graphe non-pondéré. Le marcheur part du sommet indiqué par la flèche, les probabilités de position aux temps  $t = 0, 1, 2$  et  $3$  sont indiquées en pourcentage sur chaque sommet.

## Exemple de marche aléatoire

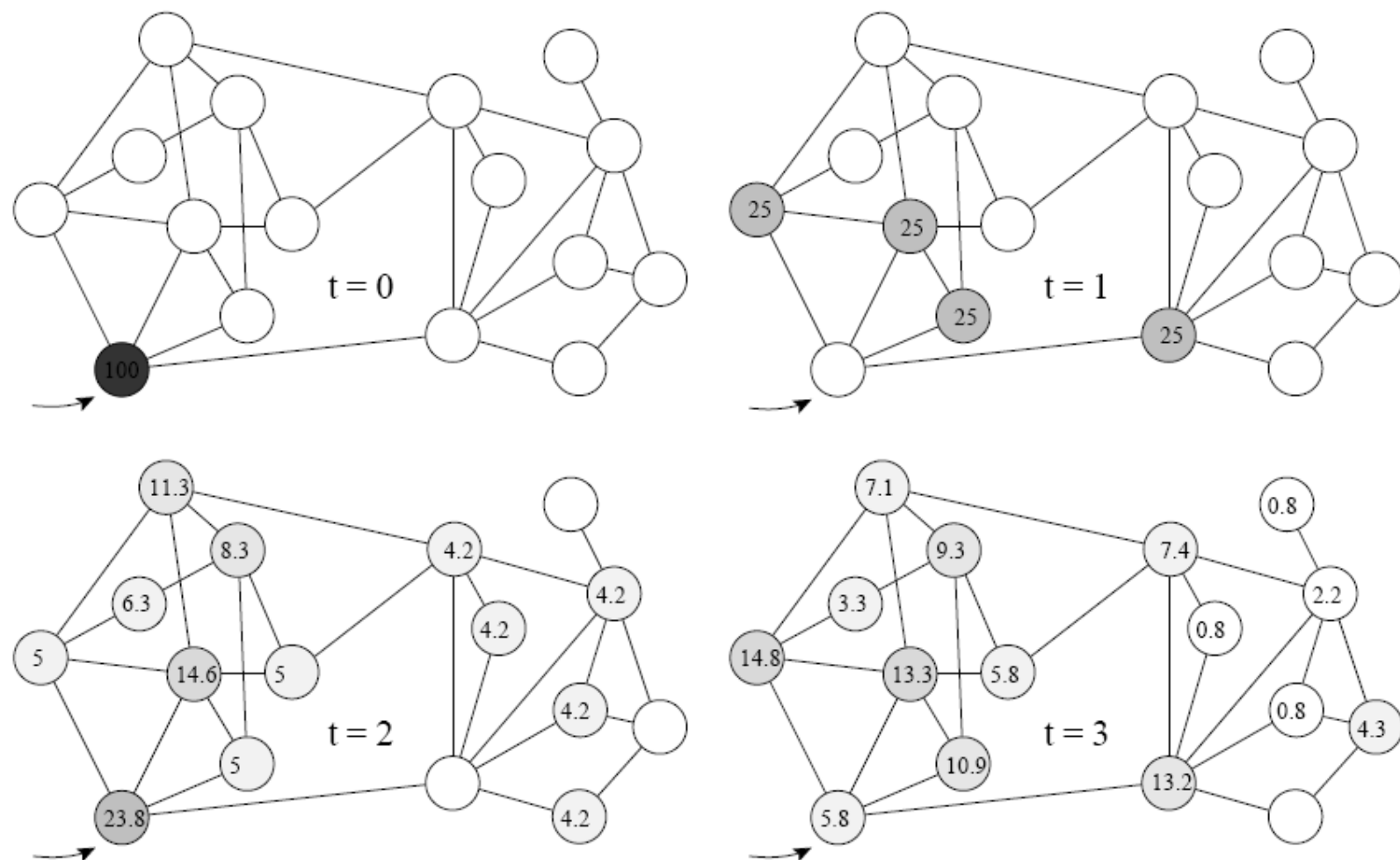


FIG. 2.1 – Exemple de dynamique d'une marche aléatoire sur un graphe non-pondéré. Le marcheur part du sommet indiqué par la flèche, les probabilités de position aux temps  $t = 0, 1, 2$  et  $3$  sont indiquées en pourcentage sur chaque sommet.

La probabilité d'aller de  $i$  à  $j$  dans une marche de longueur  $t$  est :

$$P_{ij}^t$$

La probabilité d'aller de  $i$  à  $j$  dans une marche de longueur  $t$  est

$$P_{ij}^t$$

Lien avec la structure en communautés: calculer une distance entre les sommets  $(i,j)$ .

- La distance sera
  - ♦ grande si les sommets appartiennent à des communautés différentes,
  - ♦ petite s'ils sont dans la même communauté.

La probabilité d'aller de  $i$  à  $j$  dans une marche de longueur  $t$  est

$$P_{ij}^t$$

Lien avec la structure en communautés: calculer une distance entre les sommets  $(i,j)$ .

- La distance sera
  - ♦ grande si les sommets appartiennent à des communautés différentes,
  - ♦ petite s'ils sont dans la même communauté.

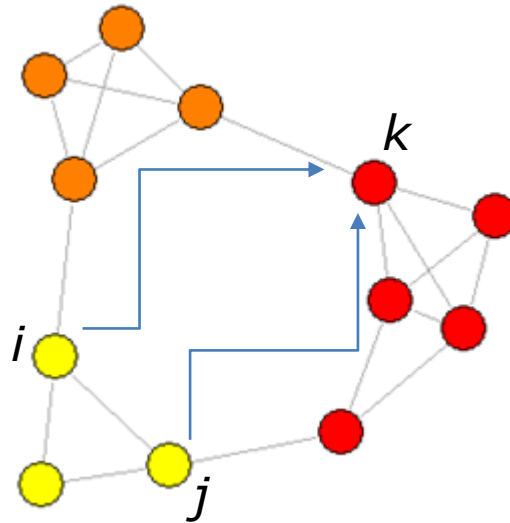
Observations:

- Si  $i$  et  $j$  sont dans la même communauté, la probabilité  $P_{ij}^t$  sera élevée.
  - ♦ Mais une probabilité élevée n'implique pas que  $i$  et  $j$  soient dans le même communauté.
- $P_{ij}^t$  dépend de  $d(j)$ , car le marcheur a plus de chance d'aller vers un sommet de fort degré.

## Idée « originale »

- Deux sommets de la même communauté ont tendance à « voire » les autres sommets de la même façon (s'ils sont éloignés).
- Ainsi, si  $i$  et  $j$  sont dans le même communauté :

$$\forall k, P_{ik}^t \approx P_{jk}^t$$



- Définition d'une distance entre sommets  $i$  et  $j$
- avec  $d(k)$  le degré de  $k$

$$r_{ij}^t = \sqrt{\sum_{k=1}^n \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}}$$

- et entre communautés  $C_1$  et  $C_2$ :

$$r_{C_1 C_2}^t = \sqrt{\sum_{k=1}^n \frac{(P_{C_1 k}^t - P_{C_2 k}^t)^2}{d(k)}}$$

- ♦ Avec

$$P_{C_1 k}^t = \frac{1}{|C|} \sum_{i \in C} P_{ik}^t$$

Le problème de trouver des communautés revient à un problème de **classification hiérarchique ascendante**.

- Initialisation : partition  $P_1$  en  $n$  communautés (nombre de sommets),
- Calculer les distances entre tous les sommets (adjacents).
- Répéter, à chaque pas  $k$ :
  - ♦ Choisir  $C_1$  et  $C_2$  dans  $P_k$  qui ont au moins une arête entre elles et qui minimise la moyenne des distances entre chaque sommets et sa communauté (méthode de Ward)

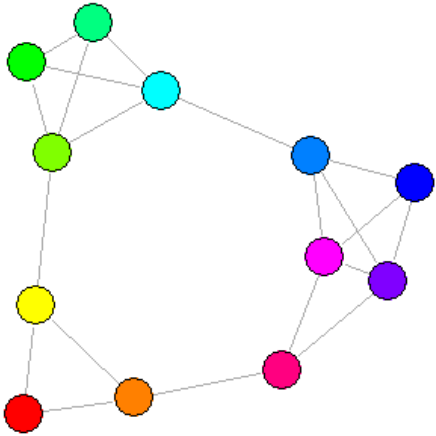
$$\sigma_k = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} (r_{iC}^t)^2$$

- ♦ Union de  $C_1$  et  $C_2$  ( $C_3 = C_1 \cup C_2$ ) et créer une nouvelle partition  $P_{k+1}$
- ♦ Mise à jour de distances entre communautés.
- ♦ L'algorithme s'arrête après  $n-1$  pas,  $P_n = \{V\}$ .



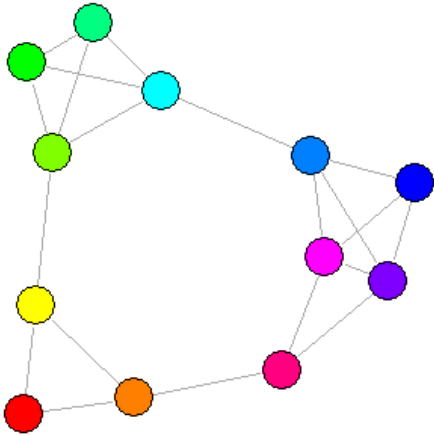
# Marche aléatoire: déroulement pas à pas

Q=0

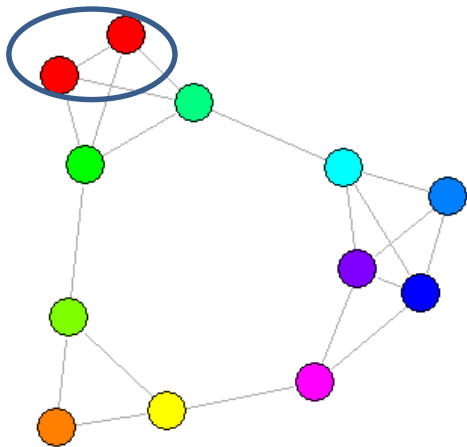


# Marche aléatoire: déroulement pas à pas

$Q=0$

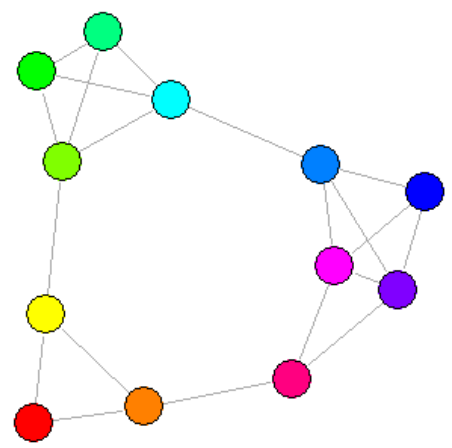


$Q= -0.048$

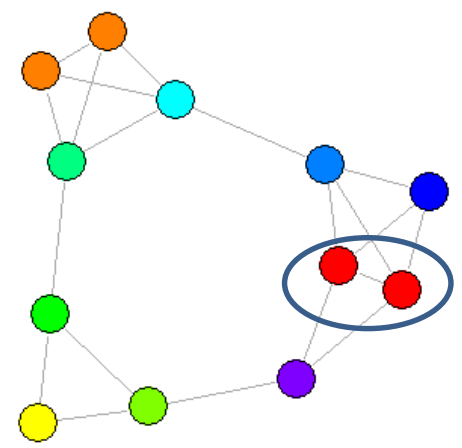


# Marche aléatoire: déroulement pas à pas

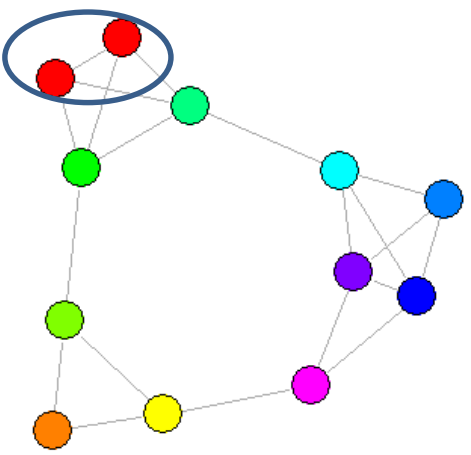
Q= 0



Q= -0.018

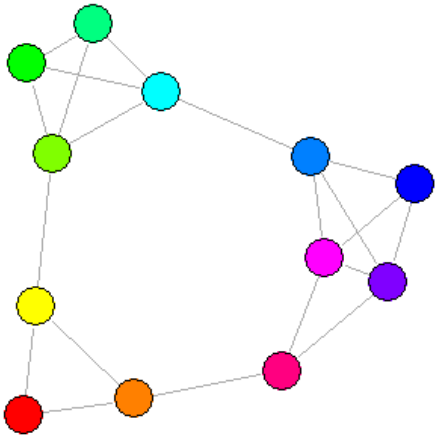


Q= -0.048

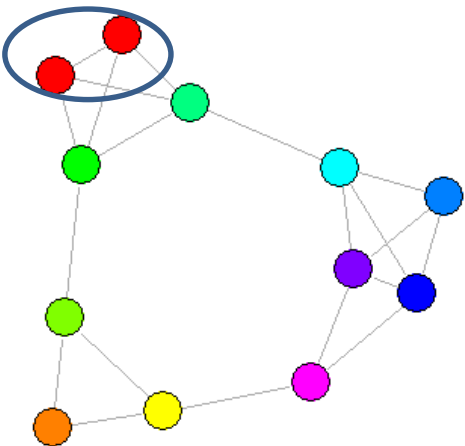


# Marche aléatoire: déroulement pas à pas

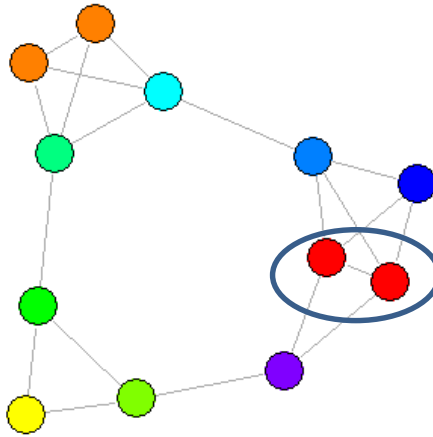
$Q=0$



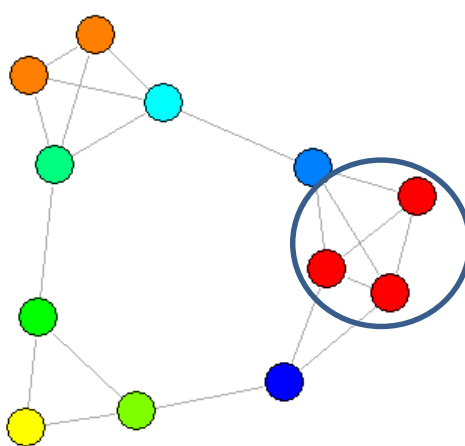
$Q=-0.048$



$Q=-0.018$

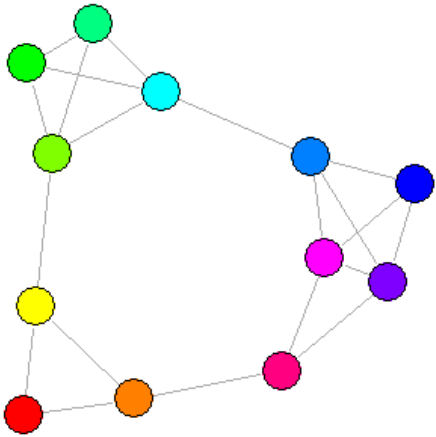


$Q=0.052$

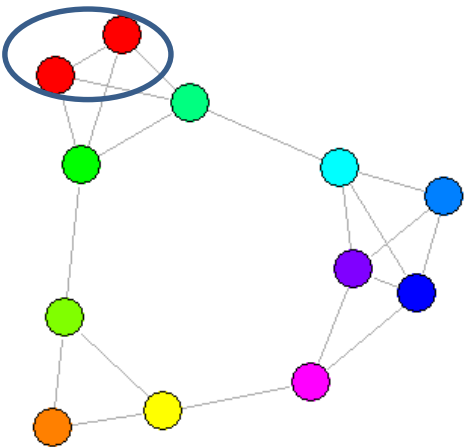


# Marche aléatoire: déroulement pas à pas

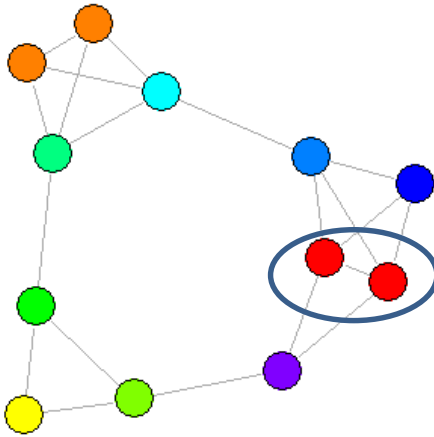
Q= 0



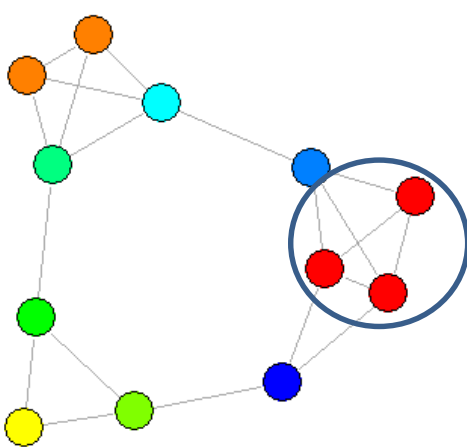
Q= -0.048



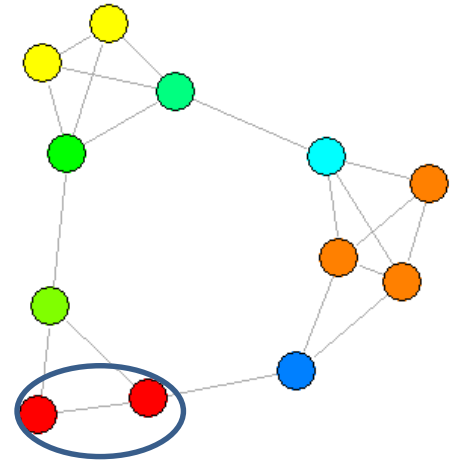
Q= -0.018



Q= 0.052

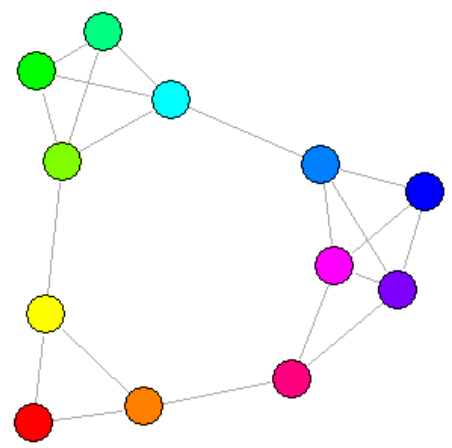


Q= 0.095

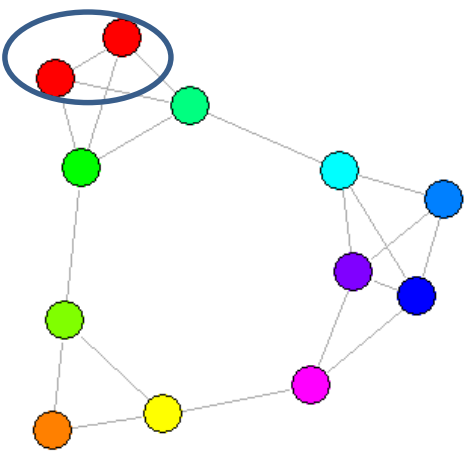


# Marche aléatoire: déroulement pas à pas

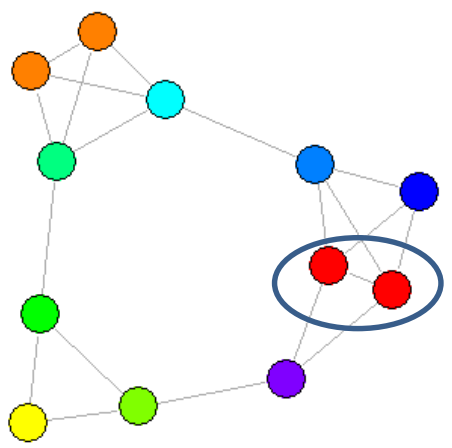
Q= 0



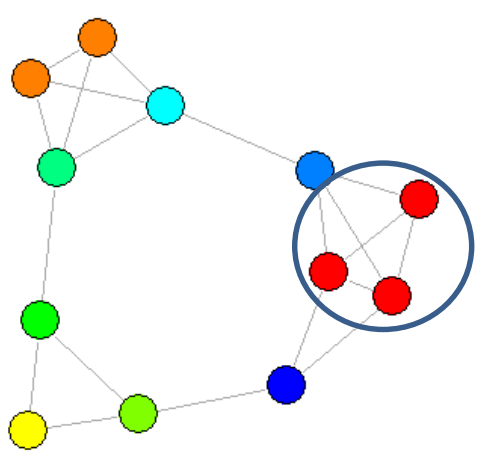
Q= -0.048



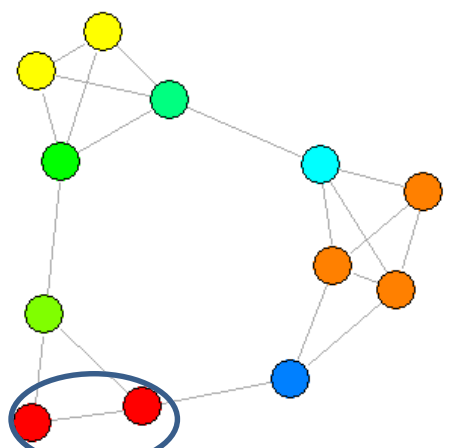
Q= -0.018



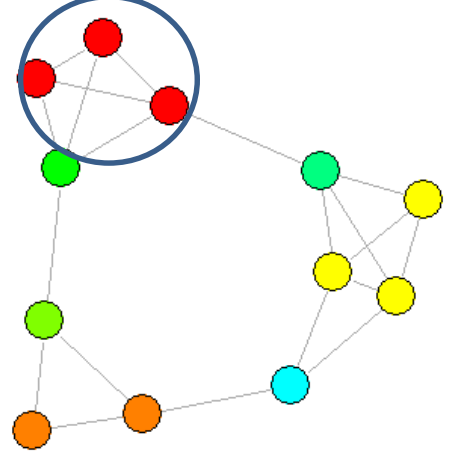
Q= 0.052



Q= 0.095

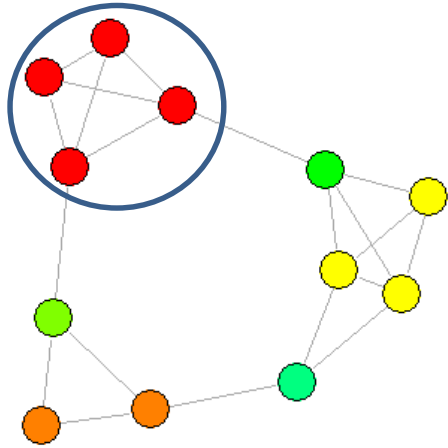


Q= 0.165



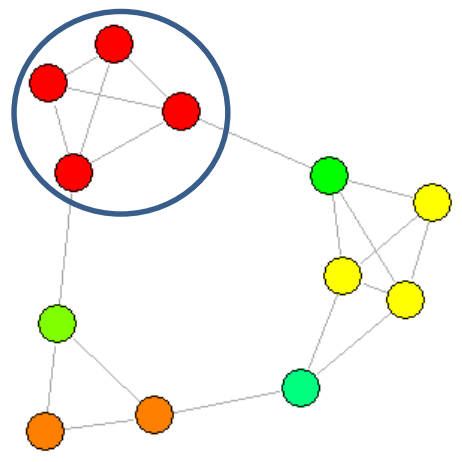
# Marche aléatoire: déroulement pas à pas

$Q = 0.265$

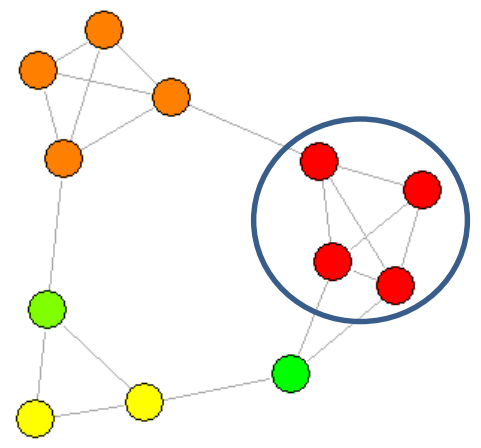


# Marche aléatoire: déroulement pas à pas

$Q = 0.265$



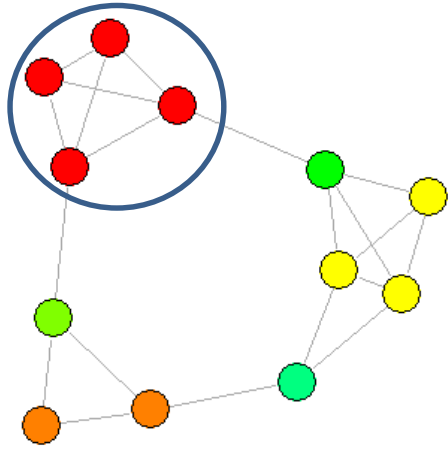
$Q = 0.36$



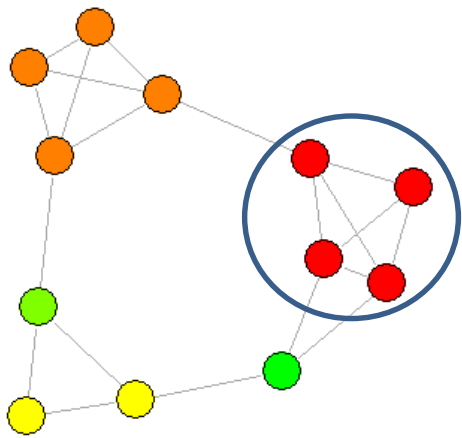


# Marche aléatoire: déroulement pas à pas

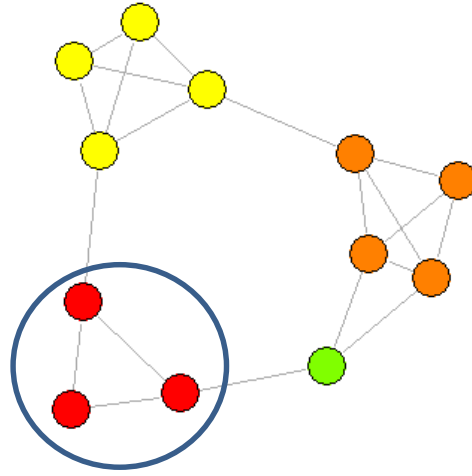
$Q = 0.265$



$Q = 0.36$

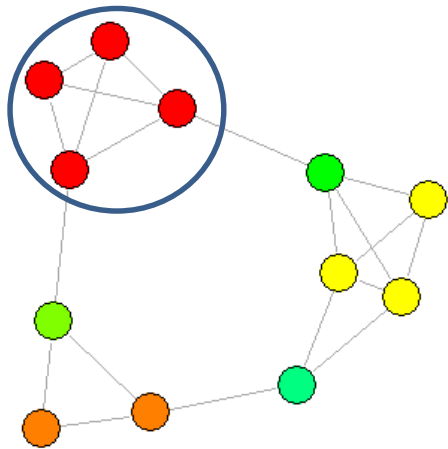


$Q = 0.441$

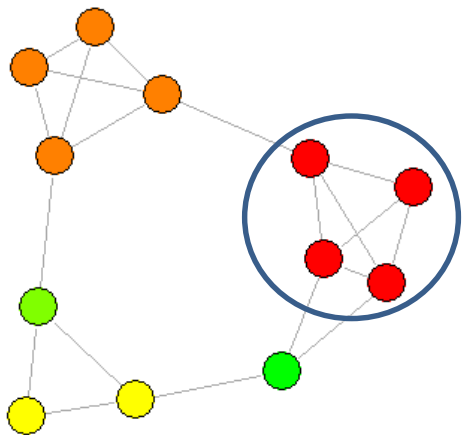


# Marche aléatoire: déroulement pas à pas

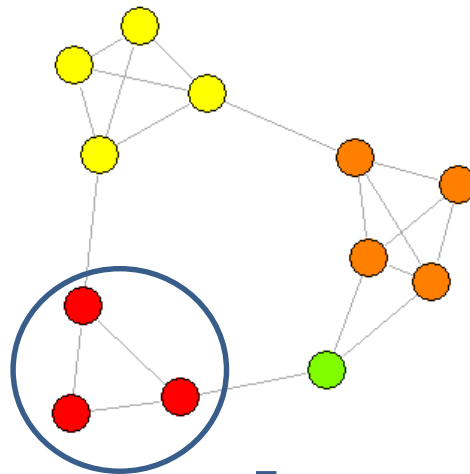
Q= 0.265



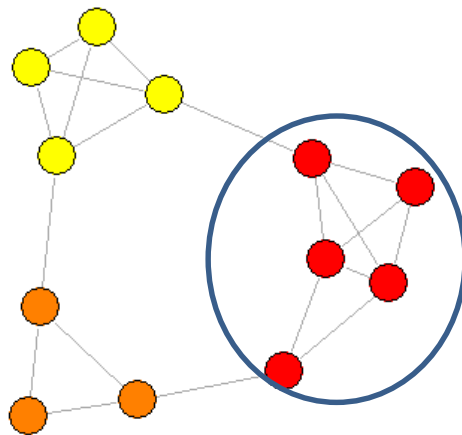
Q= 0.36



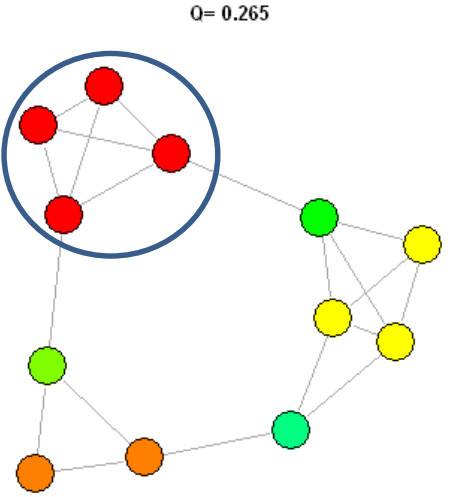
Q= 0.441



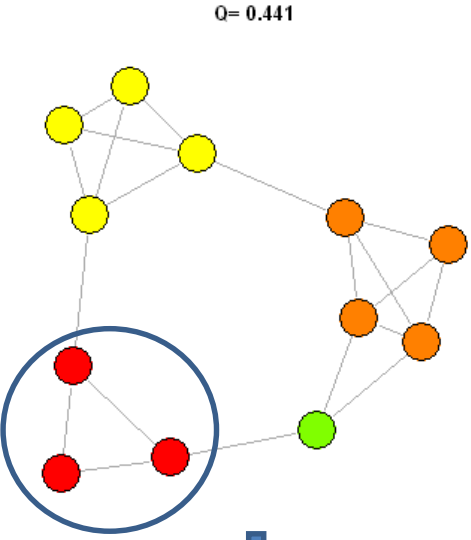
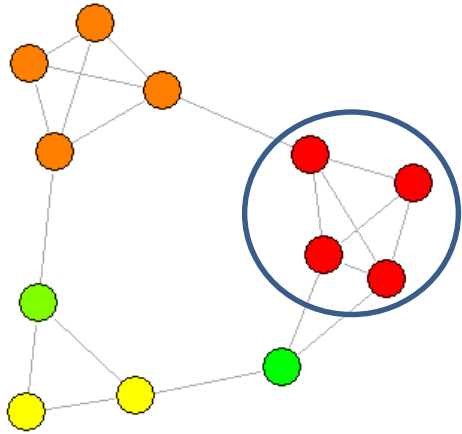
Q= 0.485



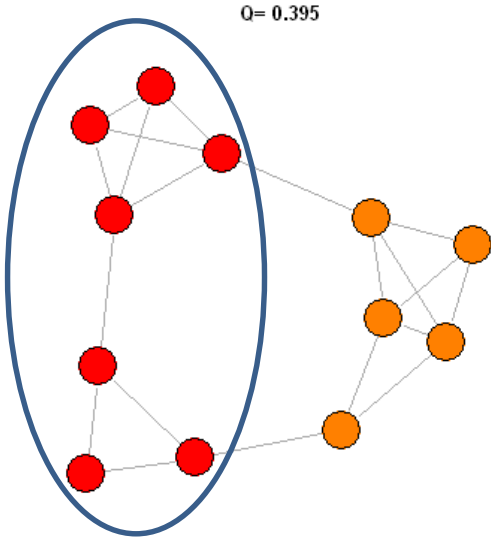
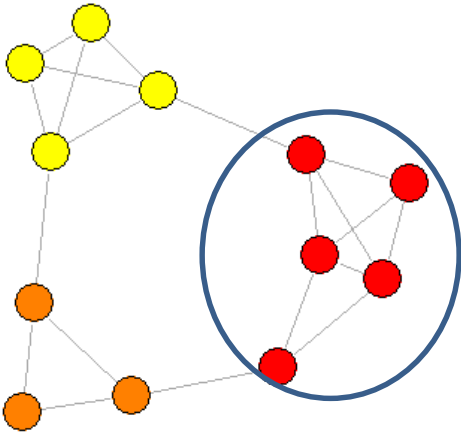
# Marche aléatoire: déroulement pas à pas



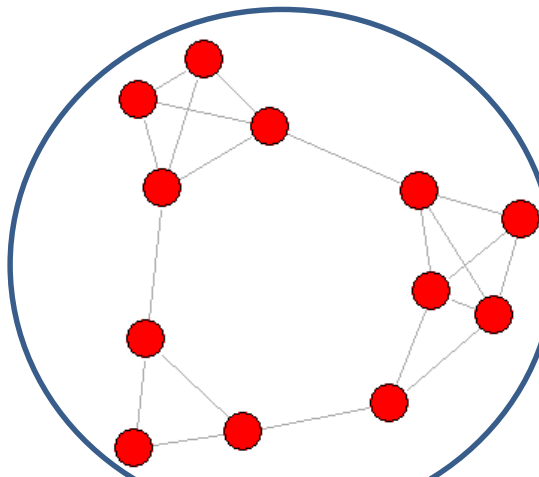
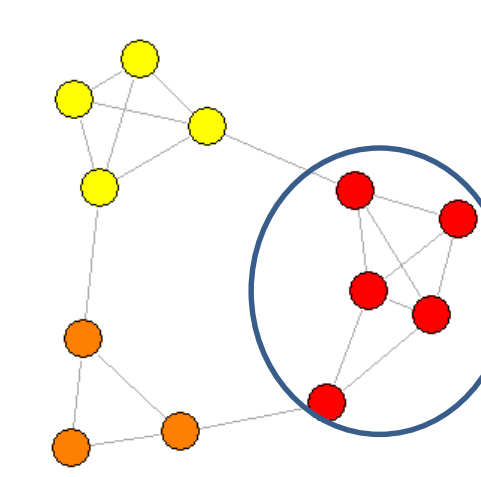
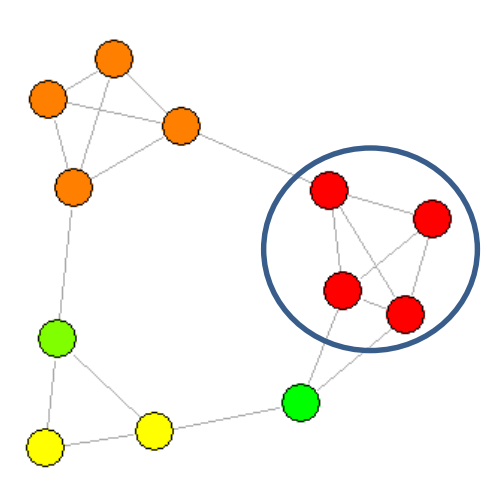
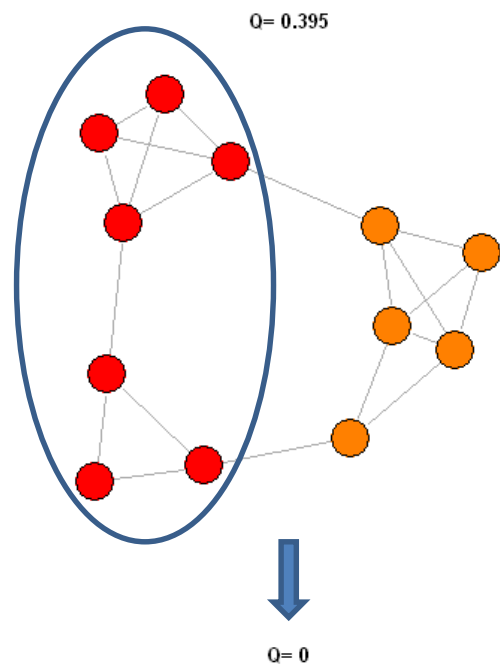
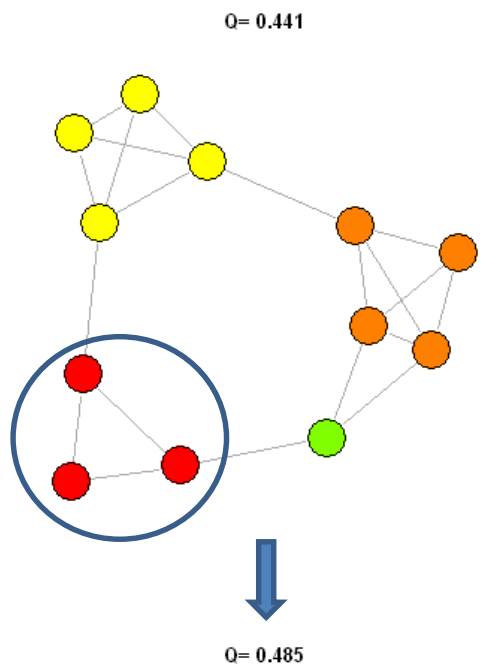
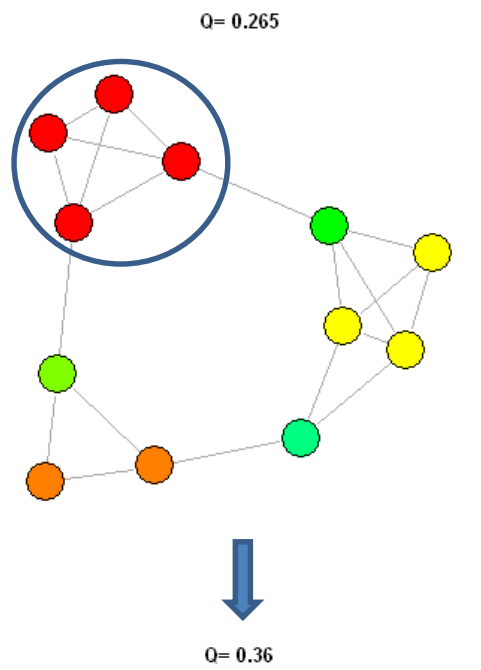
Q= 0.36



Q= 0.485



# Marche aléatoire: déroulement pas à pas

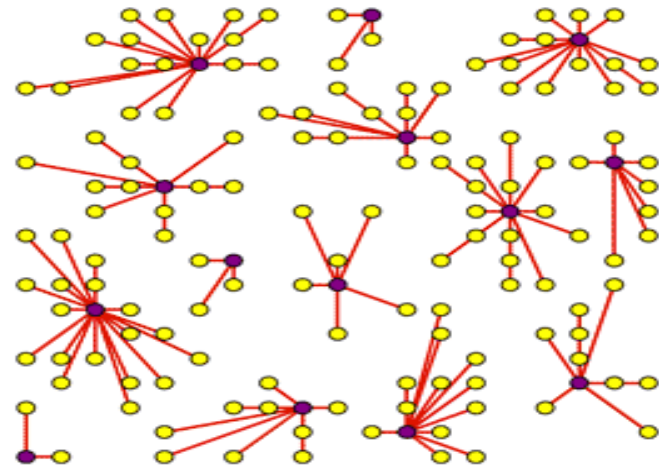
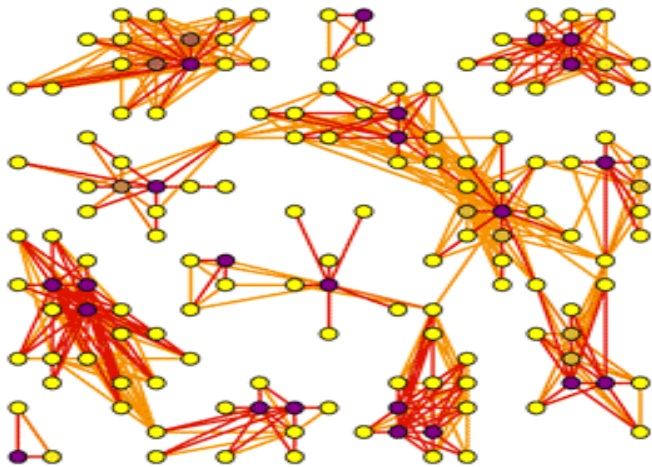
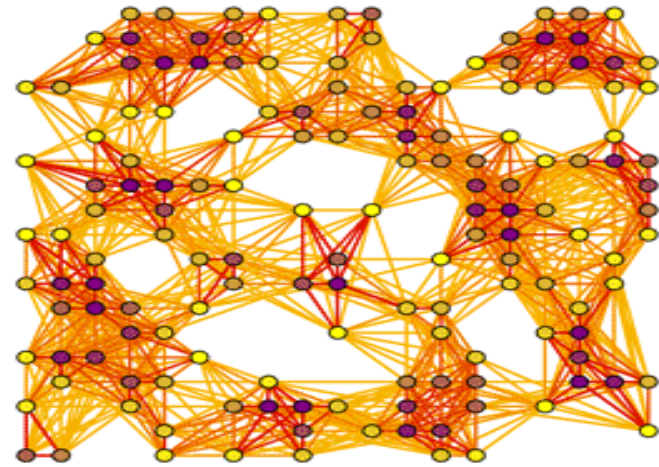
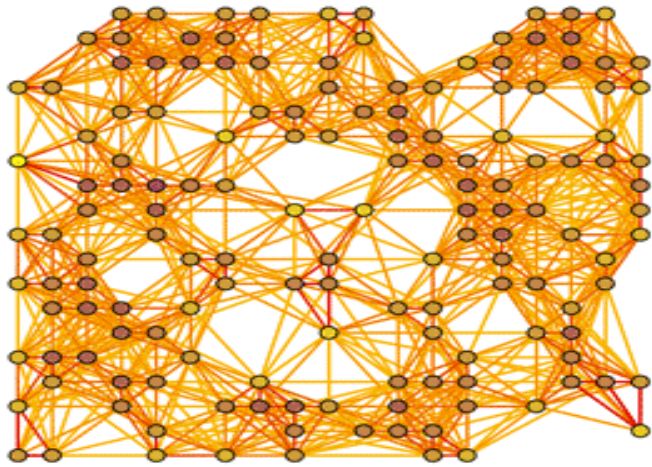


- Markov Cluster Algorithm (van Dongen, 2000)
- Méthode la plus populaire en bioinformatique!
- Simulation d'un processus de diffusion de flux dans un graphe.
  
- Initialisation: matrice stochastique  $S$  du graphe = matrice d'adjacence  $A_{ij}$  où chaque élément  $i$  est divisé par son degré  $d(i)$ .
- L'élément  $s_{ij}$  de la matrice est la probabilité d'une marche aléatoire de  $i$  à  $j$ .

$$s_{ij} = \frac{A_{ij}}{d(i)}$$

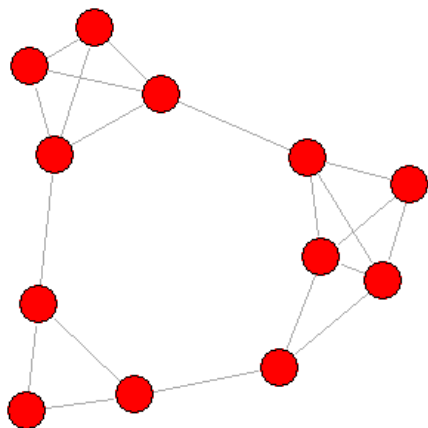
- ♦ => La somme des éléments de chaque colonne de  $S = 1$  (normalisation).

- A chaque itération, l'algorithme réalise deux étapes:
  1. **Expansion** (flux de diffusion):  $S$  est élevée à la puissance  $p$  (un entier) = matrice  $M$ . La valeur  $m_{ij}$  est alors la probabilité d'une marche aléatoire de  $i$  à  $j$  en  $p$  pas.
  2. **Inflation** (gonflement): élever chaque entrée  $m_{ij}$  de  $M$  à la puissance  $\alpha$  (un réel).
    - ♦ Le but est d'augmenter le poids des paires de sommets qui ont des  $m_{ij}$  élevées et qui ont donc de fortes chances d'appartenir à la même communauté.
    - ♦ Les éléments de chaque ligne sont divisés par leur somme marginale (normalisation). Une nouvelle matrice  $S$  est reconstituée.
- Après plusieurs itérations, le processus converge vers une matrice possédant des propriétés remarquables.
  - ♦ Ses éléments sont 0 ou 1 et le graphe correspondant est déconnecté.
  - ♦ Ses **composantes connexes** sont les communautés du graphe original.
- Avantage : simple à implémenter et très rapide.
- Inconvénient: la partition dépend du choix de  $\alpha$ , une valeur faible de  $\alpha$  donnera peu de communautés alors qu'un  $\alpha$  élevé donnera un grand nombre de communautés (granularité élevée).

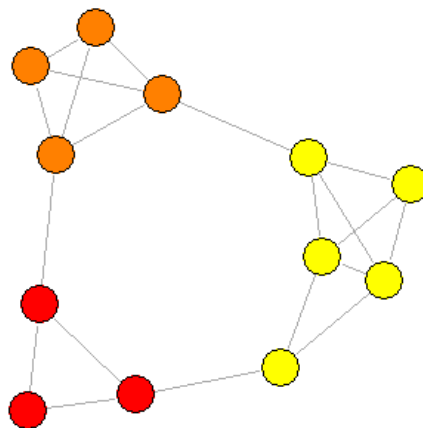


# MCL pour différentes valeurs de $\alpha$ (inflate factor)

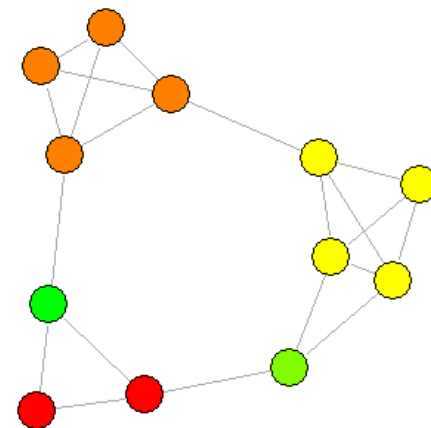
$Q=0$   $I=1.1..1.3$



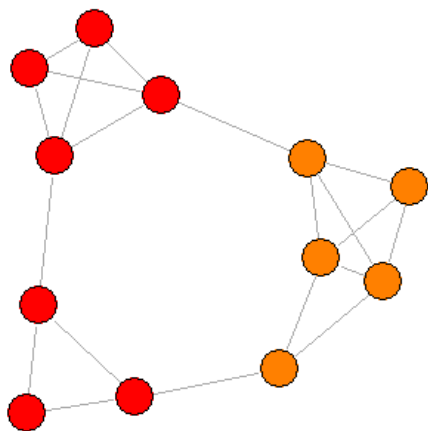
$Q=0.485$   $I=1.5..3.5$



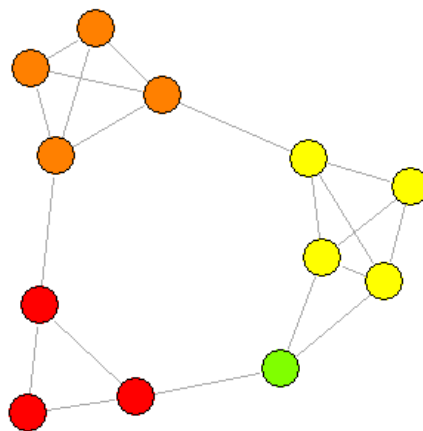
$Q=0.36$   $I=4.2..4.7$



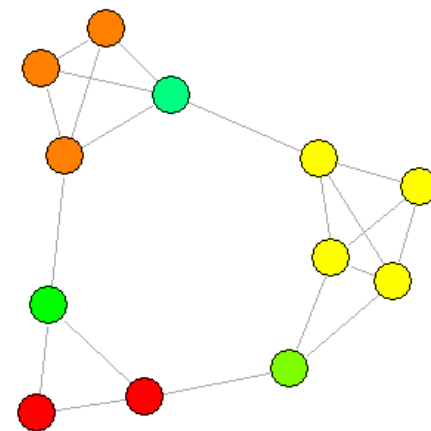
$Q=0.395$   $I=1.4$



$Q=0.441$   $I=3.6..4.2$



$Q=0.26$   $I=4.8$





# Structure hiérarchique des communautés

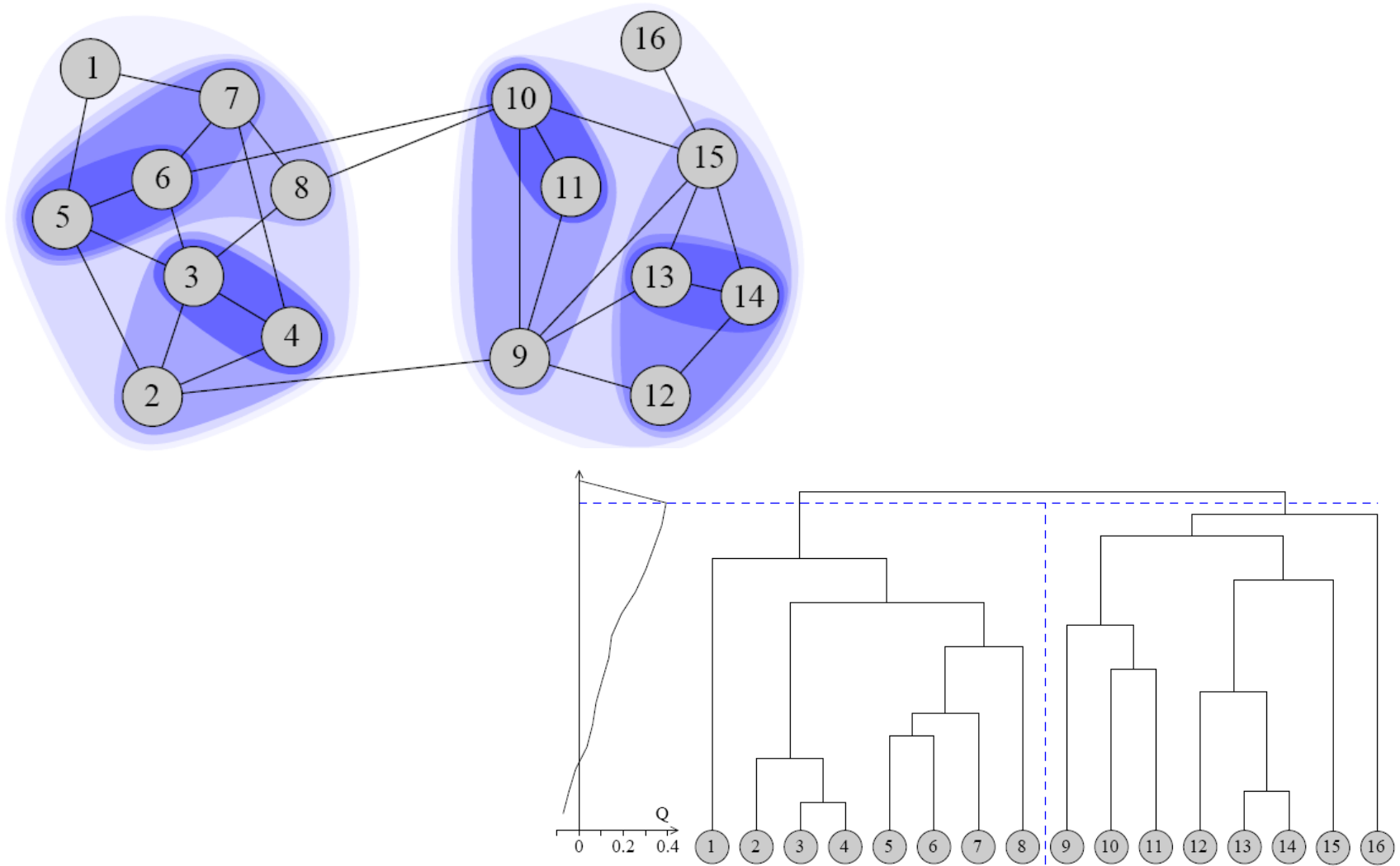


FIG. 3.2 – Dendrogramme associé aux communautés de la Figure 3.1. L'évolution de la fonction de qualité "modularité"  $Q$  (Sec. 4.2.1) donne la partition  $\mathcal{P}^k$  maximisant  $Q$  comme une coupe horizontale du dendrogramme.



# Structure hiérarchique des communautés

