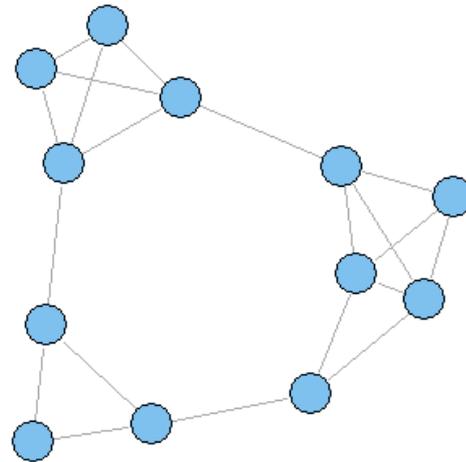
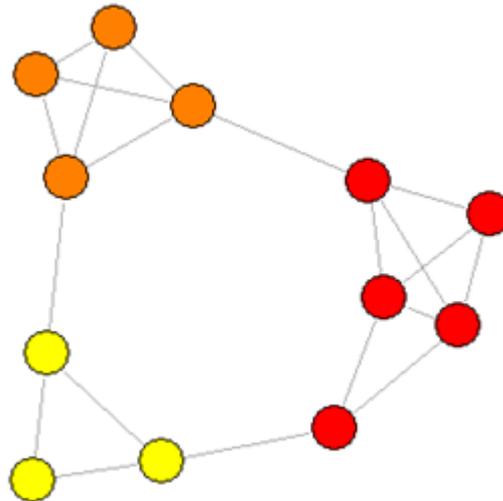


Structure en communautés dans les graphes



Combien de communautés?

- 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.
- => structure en **communautés**.
- Exemple : un graphe avec trois communautés



- **Communauté** = groupes de sommets (**modules**) qui partagent les mêmes propriétés et/ou le même rôle dans le graphe (voies métaboliques, réseau de régulation, évolution...)
- Identifier les communautés :
 - ♦ classer les sommets en fonction de leur position topologique dans le graphe:
 - Les sommets fortement connectés dans leur communauté peuvent avoir un rôle structurant pour cette communauté.
 - Les sommets à l'interface de communautés peuvent être des médiateurs (échange, lien fonctionnelle...).
- Le **but** de la détection de communautés dans les graphes est d'identifier des modules uniquement sur la base de la topologie.
- Question toujours débattue: **Quelle est le meilleur partitionnement d'un graphe en communautés?**
 - ♦ Communautés chevauchantes ou non?
 - ♦ Une partition ou une organisation hiérarchique des partitions?
 - ♦ Comparer des partitions réalisées sur le même graphe?
 - ♦ Mesurer la qualité d'une partition?

- **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:** focalisé sur les arêtes du voisinage immédiat.
 - Exemple, une communauté = sous graphe pour lequel chaque nœud a 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!

- Problème : existe un grand nombre de partitions possibles!
- La fonction de modularité Q de Newman et Girvan (2004) est la plus populaire.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta_{C_i, C_j} ,$$

- avec
 - ♦ A_{ij} : matrice adjacence,
 - ♦ k_i : degré du sommet i ($d(i)$)
 - ♦ m : le nombre de total d'arêtes dans le graphe
- La fonction δ 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 pris en compte.

- Autre formulation:

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

observé attendu

- 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 s

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

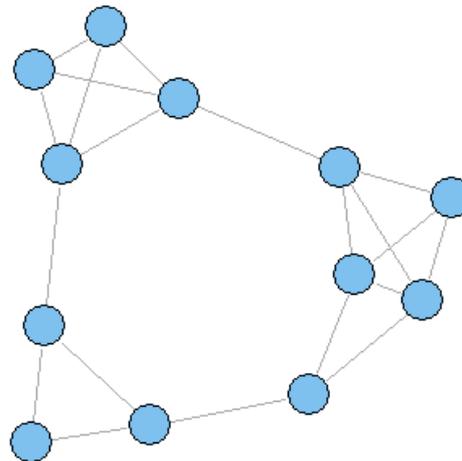
observé attendu

- 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.
 - ♦ Dans ce cas 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 ($\max(Q) = 1$).
- $Q < 0$, signale l'absence de structure.

- Inconvénients de Q :
 - ◆ A partir de quelle valeur de Q 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.
 - ◆ 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!

- 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.

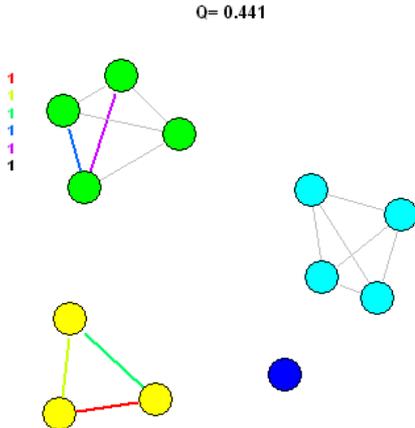
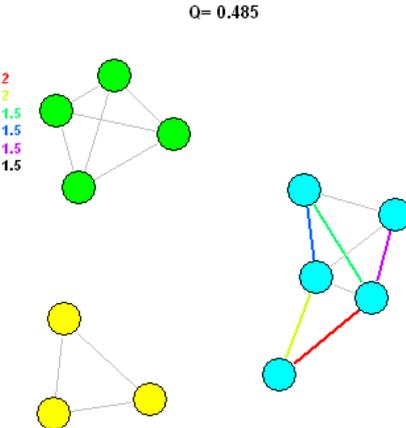
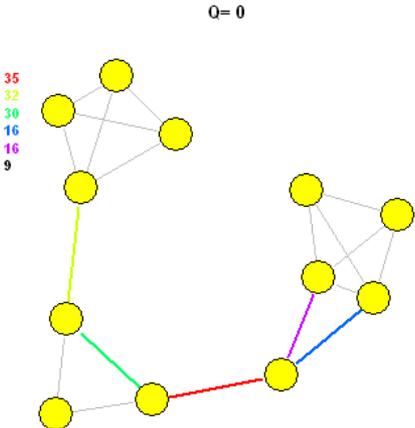
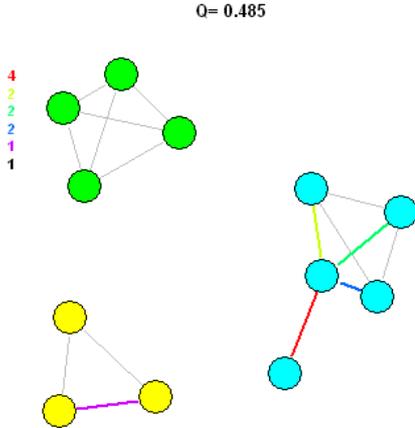
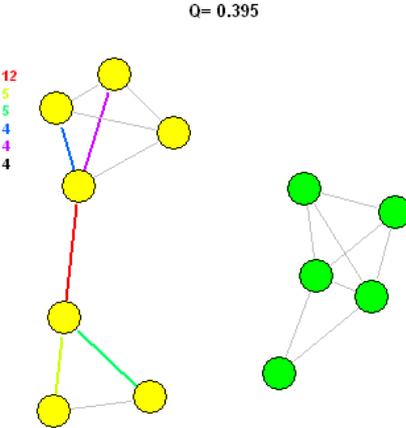
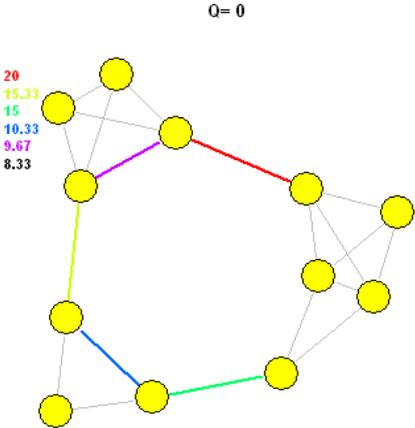
- ◆ Exemple: Edge-betweenness (Girvan et Newman 2002).
- ◆ *Betweenness* : mesure de centralité.
- ◆ *betweenness* d'une arête: nombre de chemins les plus courts entre toutes les paires de sommets qui passent par cette arête.



- ◆ La modularité Q est utilisée pour sélectionner la meilleur partition.

Edgebetweenness, déroulement pas à pas

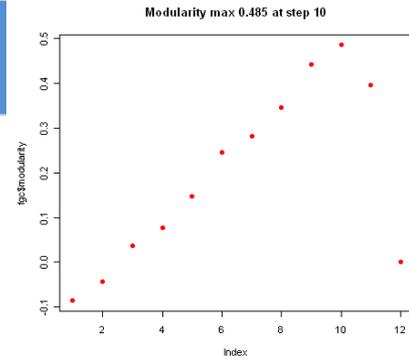
- les *betweenness* des arêtes sont triées par ordre décroissant.



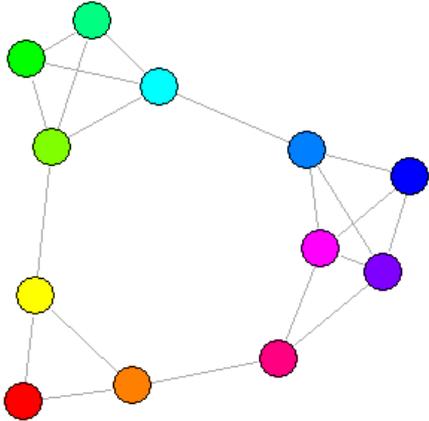
- ♦ Si la modularité Q est un bon indicateur de la qualité d'une partition, alors la partition qui maximise Q doit être la meilleur!
- ♦ Problème: maximiser Q est impossible pour la grande majorité des graphes => approximation en un temps raisonnable.
- ♦ Exemple: Fastgreedy (Newman, 2004)
- ♦ Méthode hiérarchique ascendante (agglomérative)

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 étapes 2-3 jusqu'à obtenir tous les sommets dans une seule communauté.
- Avantages: simplicité et rapidité, traitement de graphes de très grande taille.
 - Inconvénients: Q peut s'écarter de l'optimum, Q est estimée sur la globalité du graphe.

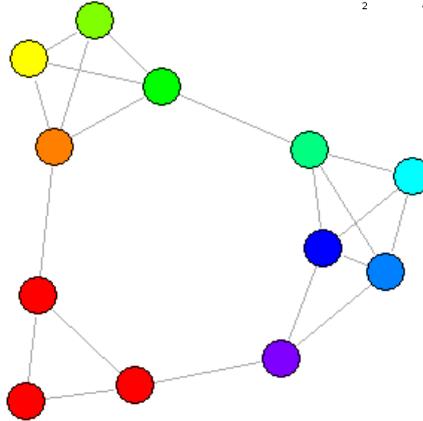
Fastgreedy, déroulement pas à pas



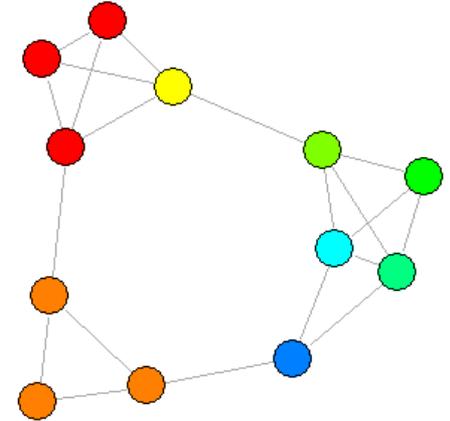
Q= -0.086



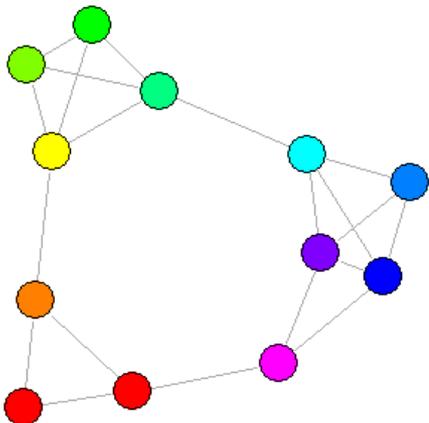
Q= 0.037



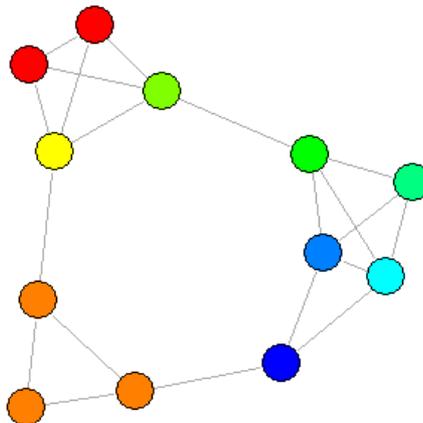
Q= 0.146



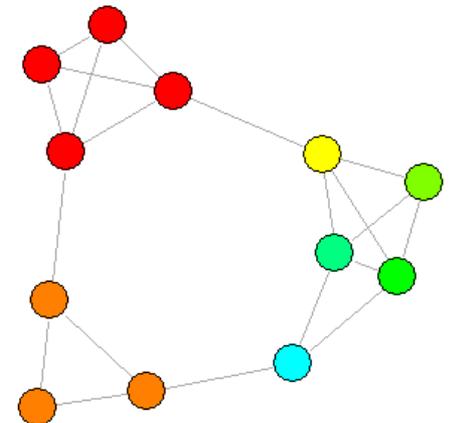
Q= -0.044



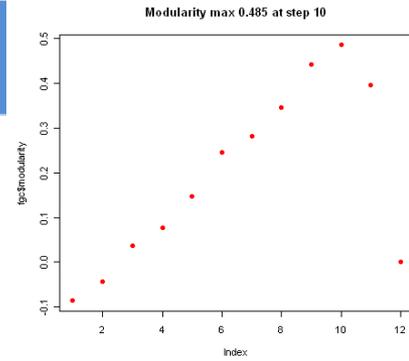
Q= 0.076



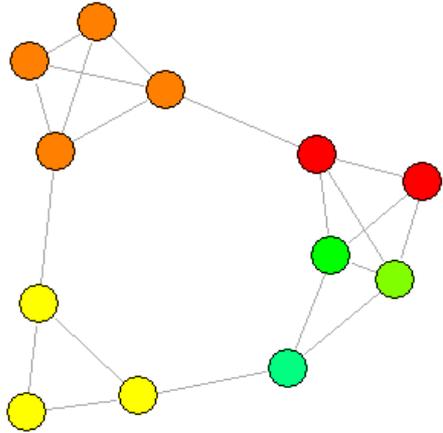
Q= 0.246



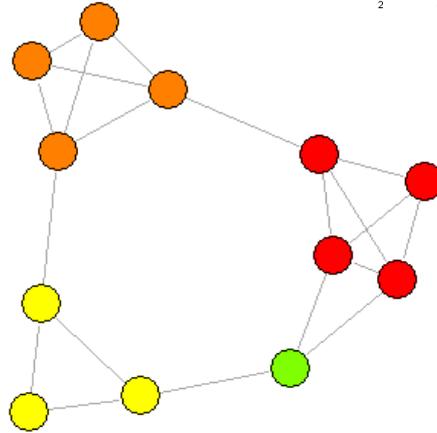
Fastgreedy, déroulement pas à pas



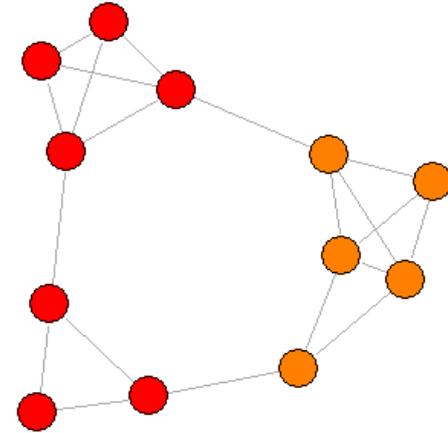
Q= 0.281



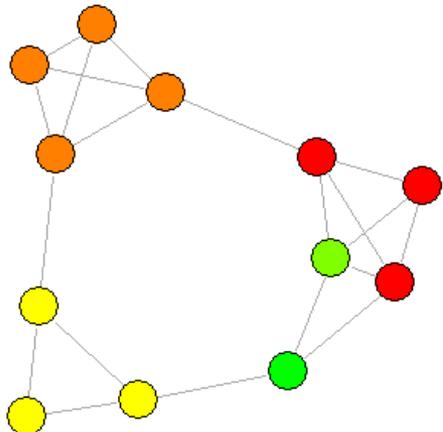
Q= 0.441



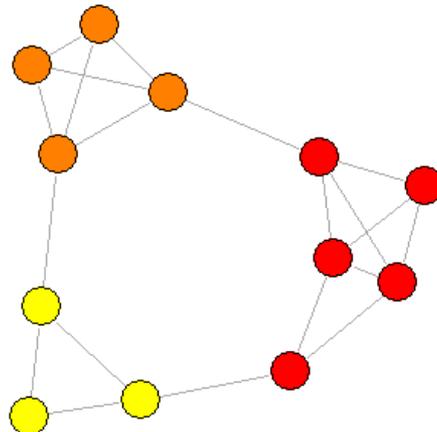
Q= 0.395



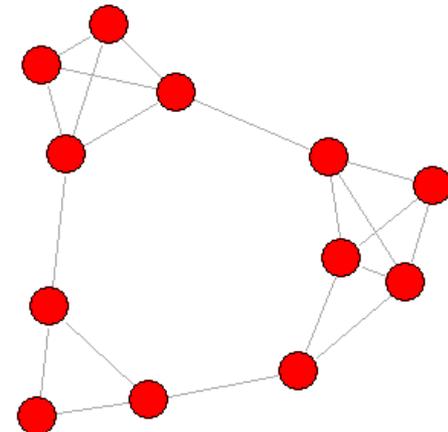
Q= 0.346



Q= 0.485



Q= 0



Recuit simulé

- 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, prise au hasard,
 2. mouvements globaux, fusion/fission de communautés.
- Donne de très bon résultats mais au dépend de temps de calcul prohibitif 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:
 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:

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

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

- 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 δ 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 non-liens.
- 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 $p_{ij} = \frac{k_i k_j}{2M}$

$$H(\sigma) = -2 \sum_{i \neq j} \left(A_{ij} - \frac{k_i k_j}{2M} \right) \delta(\sigma_i, \sigma_j) \text{ if } \gamma = 1;$$

- Relation avec Q!

- Relation entre Hamiltonian et Q

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

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

- Ce qui ressemble donc à H si $p_{ij} = \frac{k_i k_j}{2M}$ and $\gamma = 1$;

- Ainsi:
$$Q = -\frac{1}{M} H \quad \sigma$$

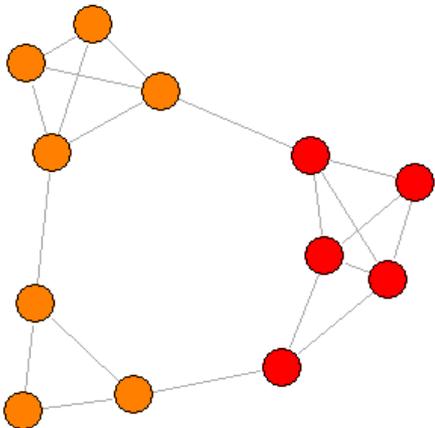
- Maximiser Q revient à minimiser H
- Méthode de recuit simulé est utilisée pour optimiser l'assignement des sommets aux groupes.

- Méthode de recuit simulé = 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 seul 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 paires 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 .

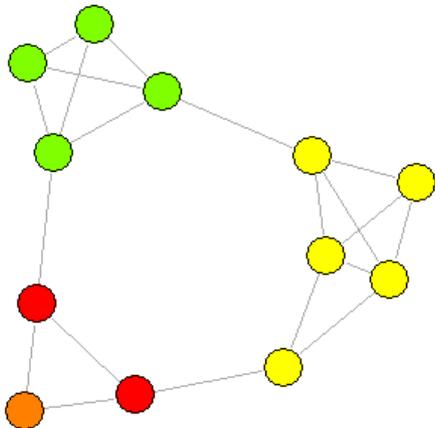
Spinglass: itérations avec valeurs du paramètre gamma différentes

- Gamma : ce paramètre 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 type de liens. Les plus petites valeurs donnent aux liens observés une plus grande importance qu'aux liens manquants.

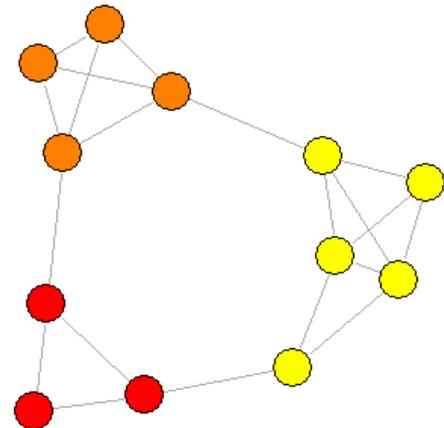
Q= 0.395 gamma= 0.1



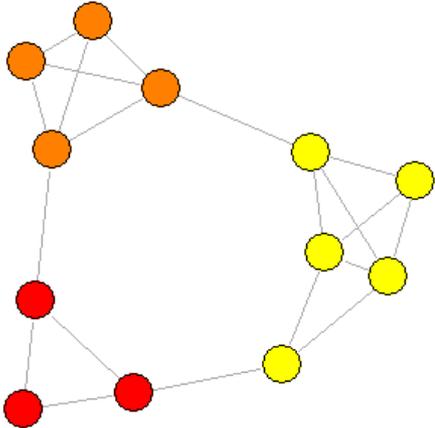
Q= 0.4 gamma= 0.4



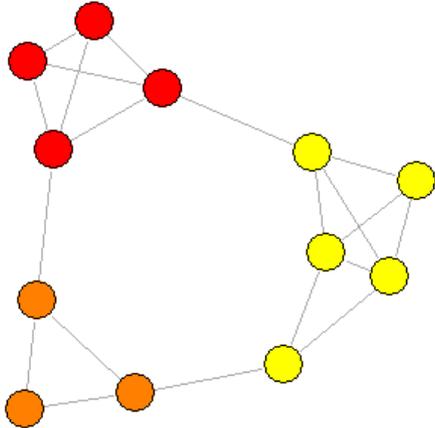
Q= 0.485 gamma= 1



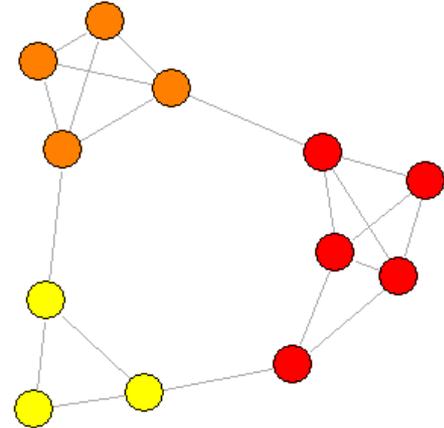
Q= 0.485 gamma= 0.2



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.
- 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.
- La séquence des sommets visités est une **chaîne de Markov** dont les sommets sont les états.
- A chaque pas, la probabilité de transition du sommet i au sommet j est

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

- Où A_{ij} est la matrice d'adjacence de G ,
- $d(i)$ est le degré de i ,
- et la probabilité d'aller de i à j dans une marche de longueur t est

$$P_{ij}^t \text{ or } P_{ij}^t$$

- Calculer une distance entre les sommets i et 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 forte. Mais une probabilité forte 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.
- Deux sommets de la même communauté ont tendance à « voire » les autres sommets de la même façon.
- 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

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

- et entre communautés C_1 et C_2 :

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

- ♦ avec $P_{C_j}^t = \frac{1}{|C_j|} \sum_{i \in C_j} P_{ij}^t$

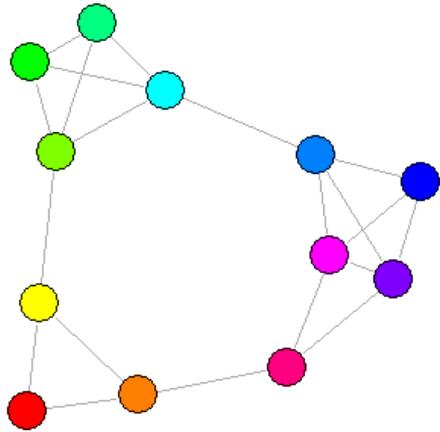
- Le problème de trouver des communautés revient à un problème de classification hiérarchique ascendante.
- Initialisation : partition P_1 en n communautés,
- Calculer les distances entre tous les sommets adjacents.
- Répéter, à chaque pas k :
 - ♦ Choisir C_1 et C_2 dans P_k qui ont au moins une arête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

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

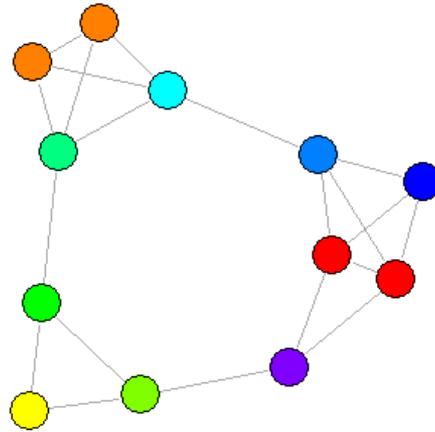
- ♦ Union de C_1 et C_2 ($C_3 = C_1 \cup C_2$) et créer une nouvelle partition P_{k+1}
- ♦ Mise à jour de distances entre communautés.
- ♦ L'algorithme s'arrête après $n-1$ pas, $P_n = \{V\}$.

Marche aléatoire: déroulement pas à pas

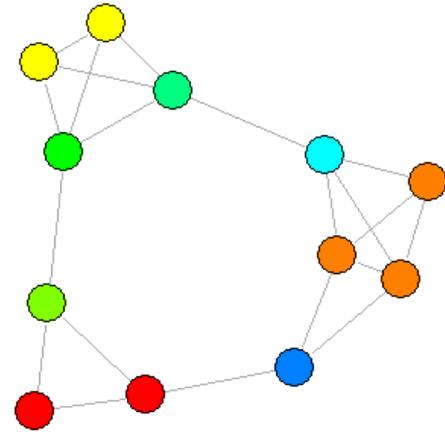
Q= 0



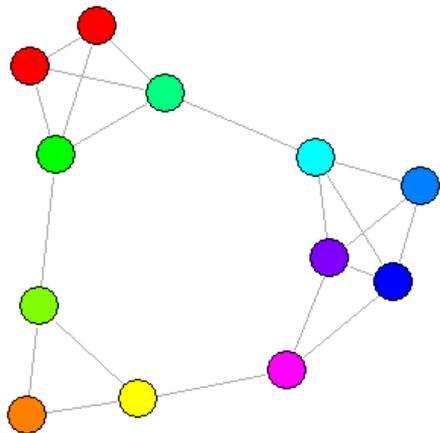
Q= -0.018



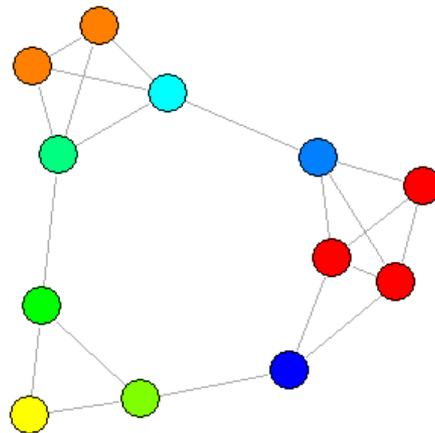
Q= 0.095



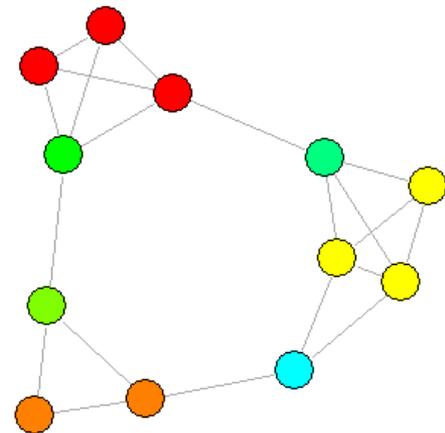
Q= -0.048



Q= 0.052

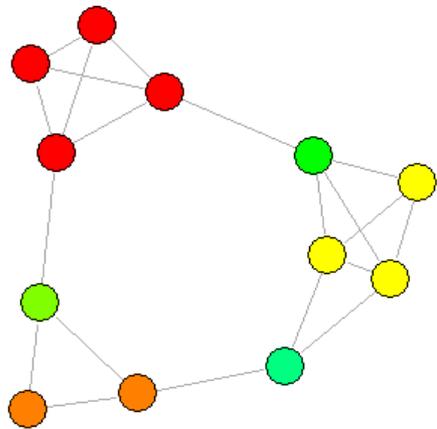


Q= 0.165

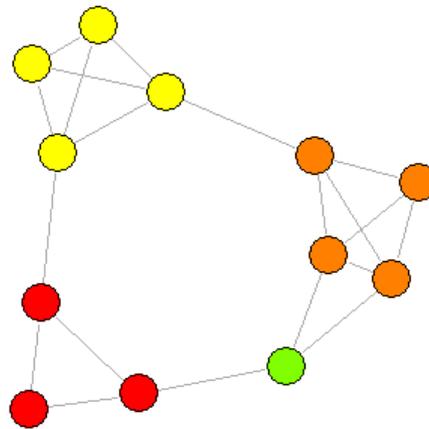


Marche aléatoire: déroulement pas à pas

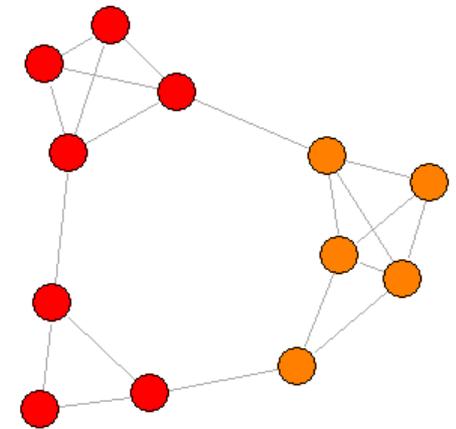
$Q = 0.265$



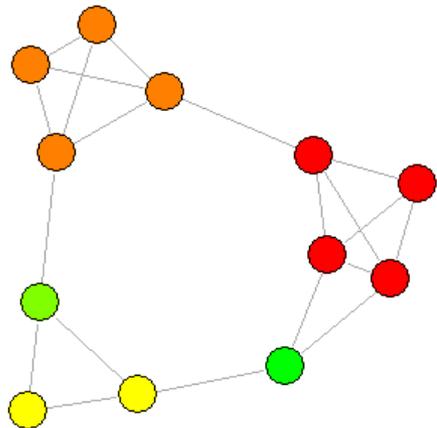
$Q = 0.441$



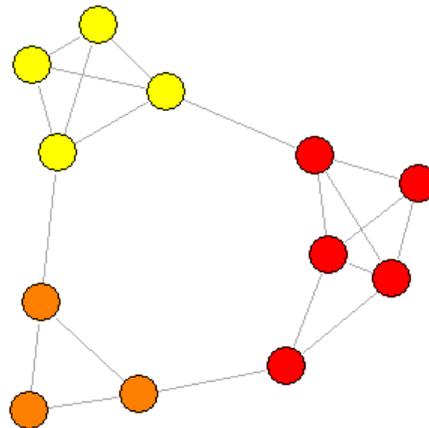
$Q = 0.395$



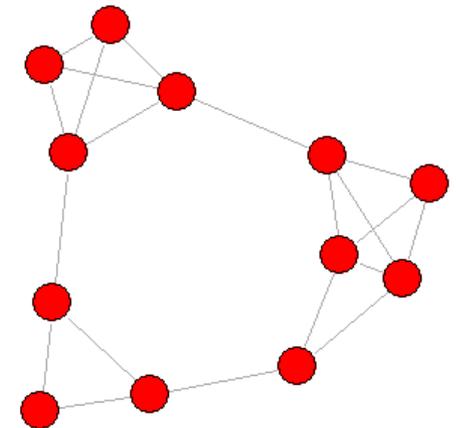
$Q = 0.36$



$Q = 0.485$



$Q = 0$



- Markov Cluster Algorithm (van Dongen, 2000)
- Méthode la plus populaire en bioinformatique!
- Simulation d'un processus de diffusion de flux dans un graphe.

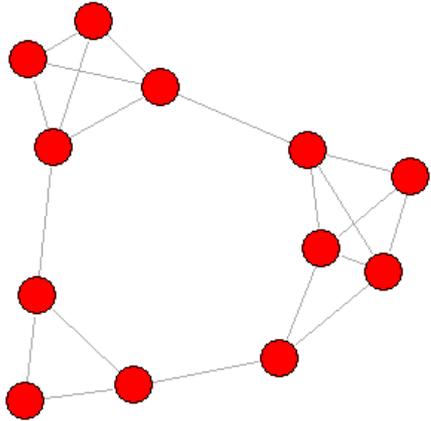
- Initialisation: matrice stochastique S du graphe = matrice d'adjacence A_{ij} où chaque élément i est divisé par son degré $d(i)$.
- L'élément s_{ij} de la matrice est la probabilité d'une marche aléatoire de i à j .

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

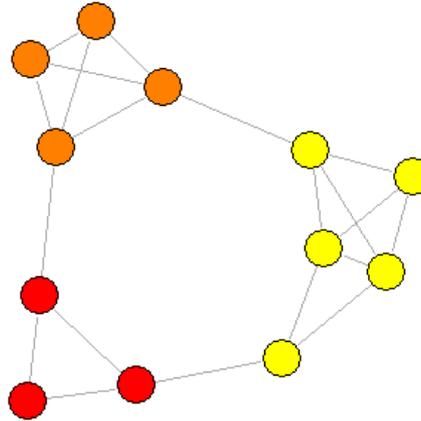
- ♦ => La somme des éléments de chaque colonne de $S = 1$.

- 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 α (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. 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 α , une valeur faible de α donnera peu de communautés alors qu'un α élevé donnera un grand nombre de communautés.

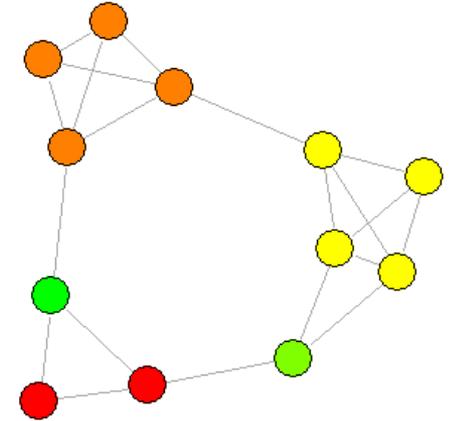
Q= 0 I= 1.1..1.3



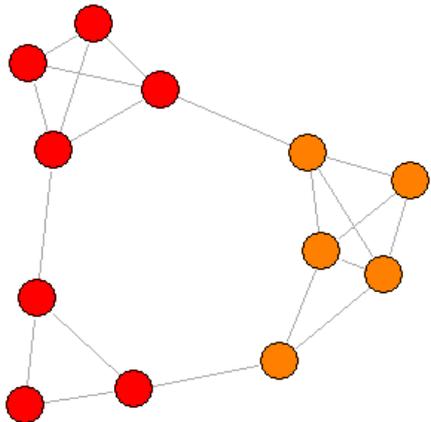
Q= 0.485 I= 1.5..3.5



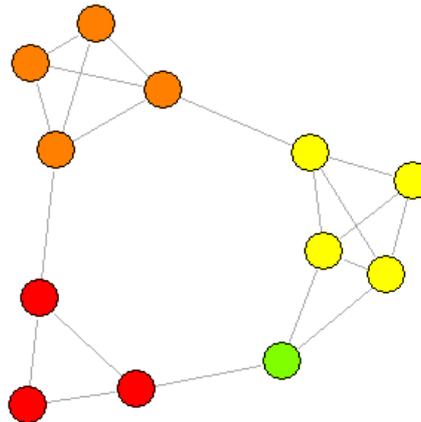
Q= 0.36 I= 4.2..4.7



Q= 0.395 I= 1.4



Q= 0.441 I= 3.6..4.2



Q= 0.26 I= 4.8

