Séminaire Graphes et structures discrètes du LIP et de l'IXXI
Lyon – 06/03/2012

Partitionnement de graphes par optimisation de modularité et consensus de partitions

Philippe Gambette

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

Plan

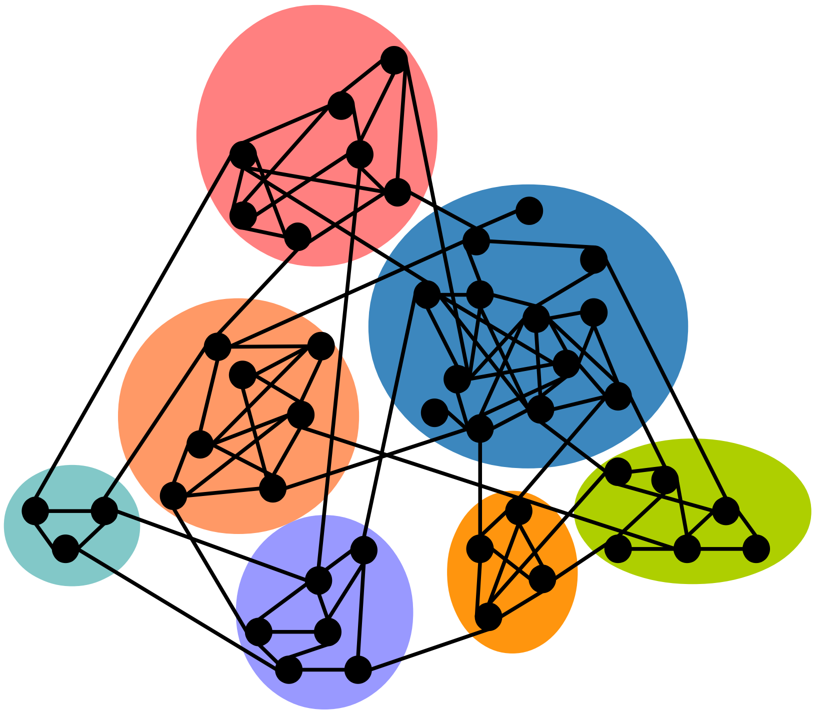
- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

La modularité

Partitionnement d'un réseau : couvrir tous les sommets

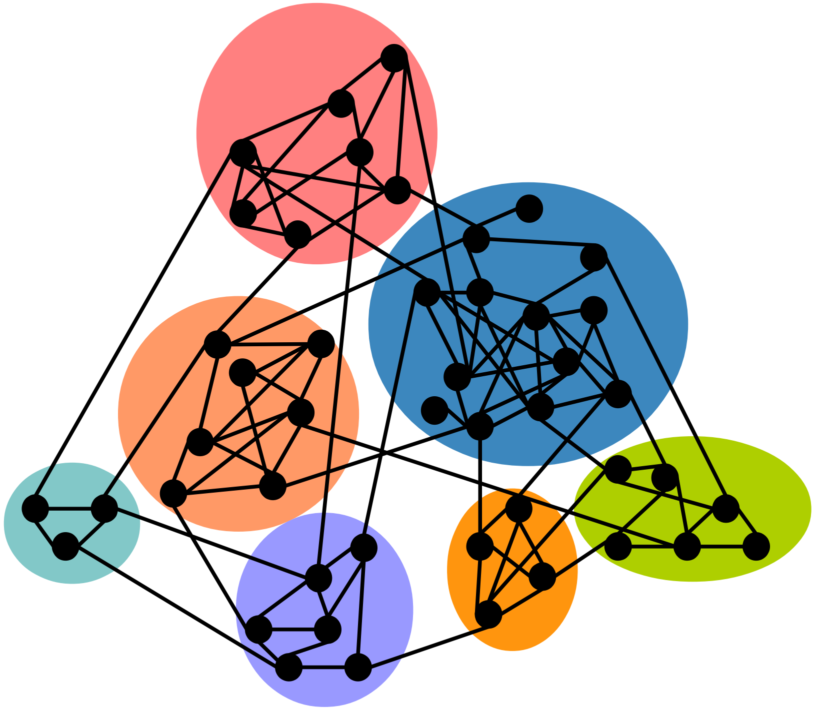
La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



La modularité

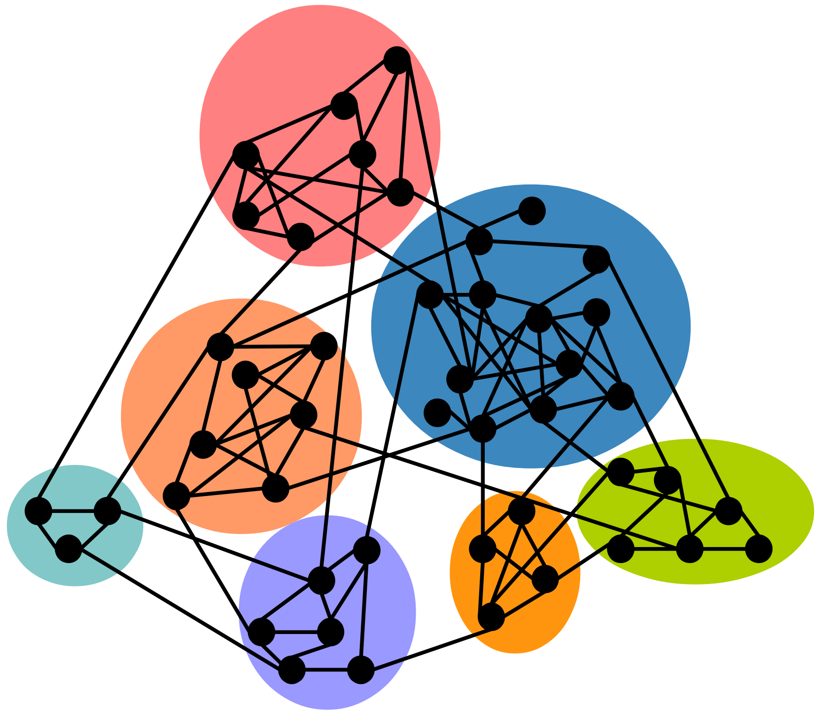
Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



sans fixer aucun paramètre ?

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : qualité du partitionnement

Girvan & Newman 2004

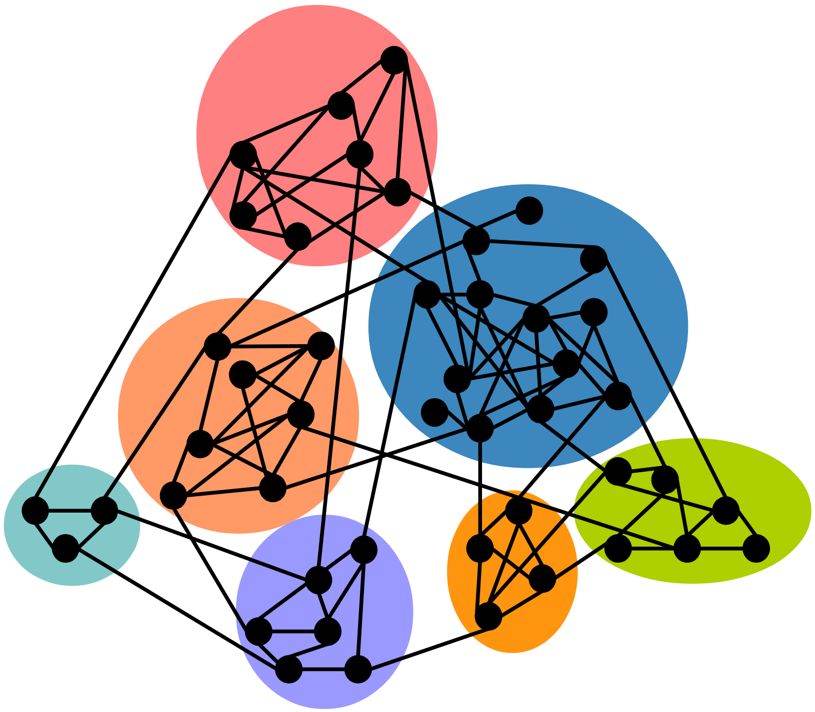
$$M(G,P) = \sum_{C_i \in P} M(C_i)$$

$$M(C_i) = e_{ii} - \left(e_{ii} + \sum_{j \neq i} e_{ij}/2\right)^2$$

e_{ij} = proportion d'arêtes avec un sommet dans C_i et l'autre dans C_j

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : qualité du partitionnement

Girvan & Newman 2004

$$M(G,P) = \sum_{C_i \in P} M(C_i)$$

$$M(C_i) = \underbrace{e_{ii}}_{} - \underbrace{\left(e_{ii} + \sum_{j \neq i} e_{ij}/2\right)^2}_{}$$

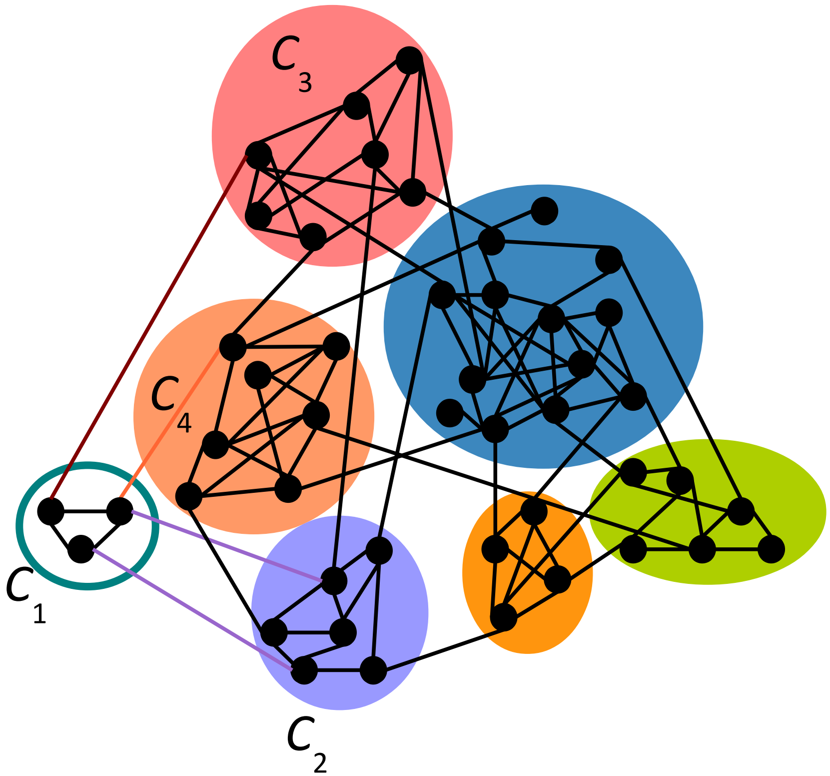
proportion d'arêtes observées dans la classe C_i

proportion d'arêtes attendues dans la classe C_i s'il n'y avait pas de communauté, et répartition au hasard en respectant les degrés

e_{ij} = proportion d'arêtes avec un sommet dans C_i et l'autre dans C_j

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : qualité du partitionnement

Girvan & Newman 2004

$$M(G,P) = \sum_{C_i \in P} M(C_i)$$

$$M(C_i) = \underbrace{e_{ii}}_{\text{proportion d'arêtes observées dans la classe } C_i} - \underbrace{\left(e_{ii} + \sum_{j \neq i} e_{ij}/2\right)^2}_{\text{proportion d'arêtes attendues dans la classe } C_i \text{ s'il n'y avait pas de communauté, et répartition au hasard en respectant les degrés}}$$

$e_{11} = 3$; $e_{12} = 2$; $e_{13} = 1$; $e_{14} = 1$

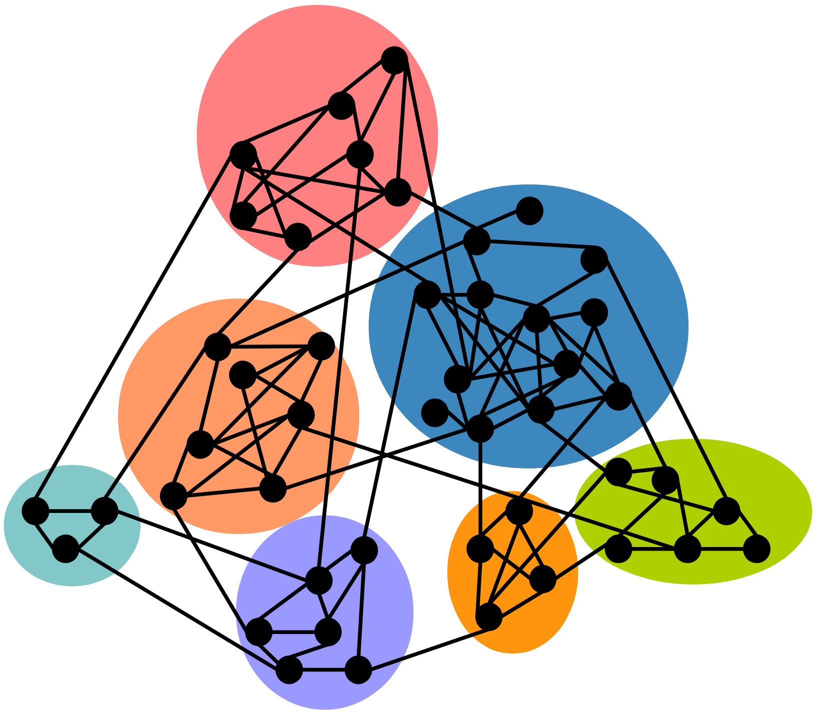
$M(C_1) = 3/100 - (7/100)^2$

$= 0.0251$

e_{ij} = proportion d'arêtes avec un sommet dans C_i et l'autre dans C_j

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : formule équivalente

Newman 2004

$$M(G,P) = \frac{1}{2m} \sum_{u,v} \left(G_{uv} - \frac{d(u)d(v)}{2m}\right)\alpha_{uv}$$

$G_{uv} =$ 1 si u et v adjacents

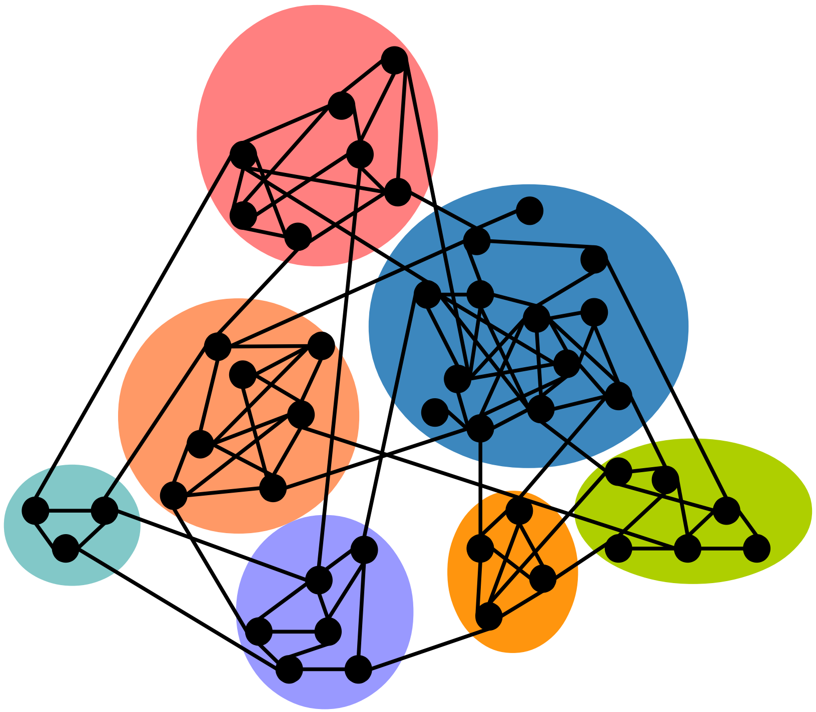
0 sinon

$\alpha_{uv} =$ 1 si u et v dans la même classe

0 sinon

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : formule équivalente

Newman 2004

$$M(G,P) = \frac{1}{2m} \sum_{u,v} \left(\boxed{G_{uv}} - \frac{\boxed{d(u)d(v)}}{2m} \right) \alpha_{uv}$$

G_{uv} = 1 si *u* et *v* adjacents

0 sinon

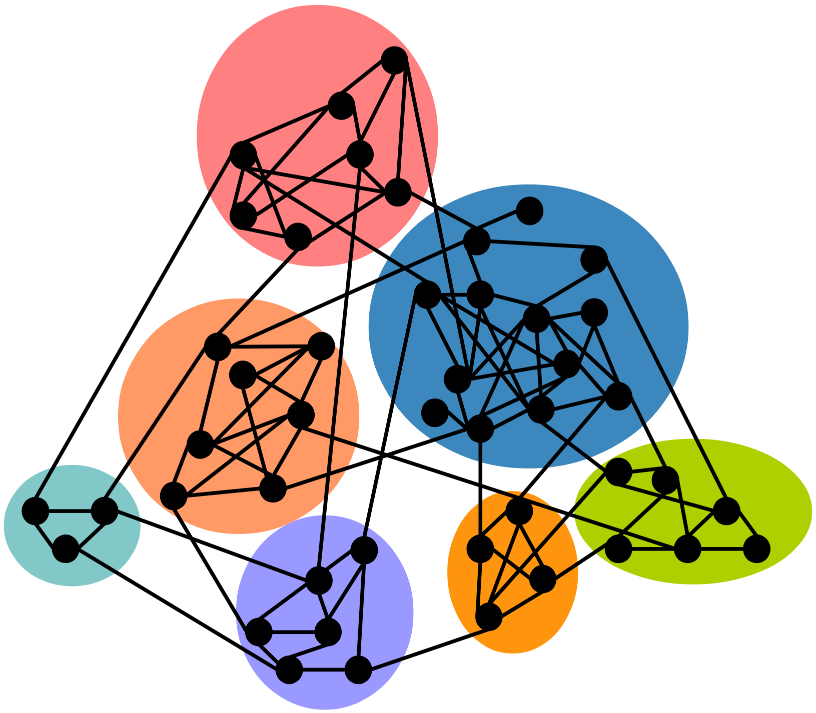
α_{uv} = 1 si *u* et *v* dans la même classe

0 sinon

Possibilité d'étendre aux graphes aux arêtes pondérées

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : formule équivalente

Newman 2004

$$M(G,P) = \frac{1}{2m} \sum_{u,v} \underbrace{\left(G_{uv} - \frac{d(u)d(v)}{2m}\right)}_{w_{uv}} \alpha_{uv}$$

G_{uv} = 1 si *u* et *v* adjacents

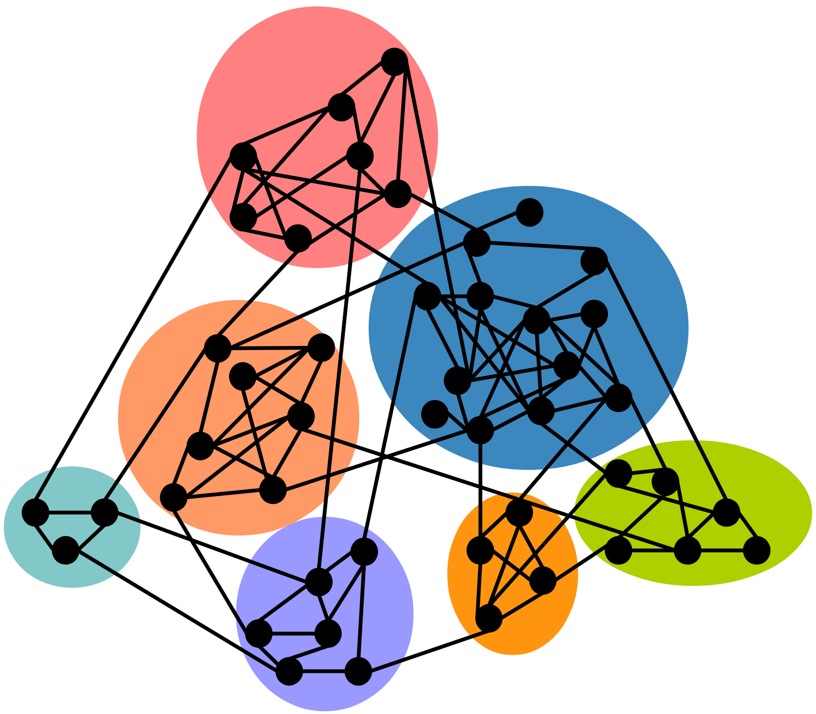
0 sinon

α_{uv} = 1 si *u* et *v* dans la même classe

0 sinon

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : formule équivalente

Newman 2004

$$M(G,P) = \frac{1}{2m} \sum_{u,v} \underbrace{\left(G_{uv} - \frac{d(u)d(v)}{2m}\right)}_{w_{uv}} \alpha_{uv}$$

Problème d'optimisation **NP-complet**

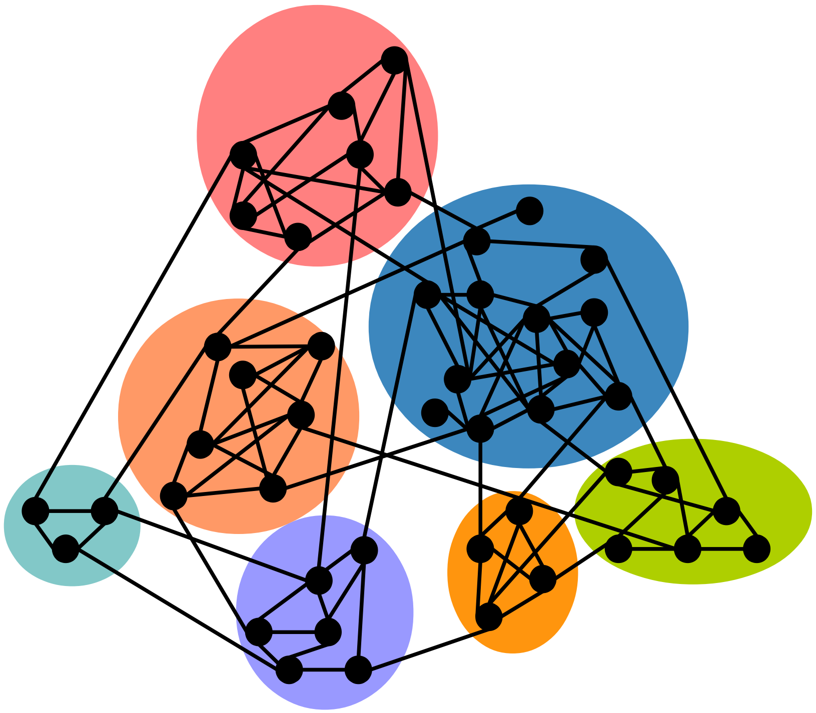
Brandes et al. 2008

- Même si la solution recherchée a 2 classes et que le graphe est peu dense
- APX-difficile si la solution recherchée a *k* classes, pour tout *k*

DasGupta & Desai, 2011

La modularité

Partitionnement d'un réseau : couvrir tous les sommets par des classes disjointes



Modularité : formule équivalente

Newman 2004

$$M(G,P) = \frac{1}{2m} \sum_{u,v} \underbrace{\left(G_{uv} - \frac{d(u)d(v)}{2m}\right)}_{w_{uv}} \alpha_{uv}$$

Problème d'optimisation **NP-complet**

Brandes et al. 2008

- Optimisation linéaire en nombres entiers

Brandes et al. 2008

- Heuristiques

Blondel et al. 2008, Rotta & Noack 2011

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

- perturber les données
- refaire tourner l'algorithme
- obtenir une nouvelle solution

fournir une solution consensus dont la robustesse est évaluée

Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

$\left.\begin{array}{l}\text{• } \boxed{\textbf{perturber}} \text{ les données} \\ \text{• refaire tourner l'algorithme} \\ \text{• obtenir une nouvelle solution}\end{array}\right\}$ fournir une solution consensus dont la robustesse est évaluée

Perturbation des données :

• modification aléatoire des poids des arêtes dans l'intervalle $[1-\varepsilon, 1+\varepsilon]$

• ajout aléatoire d'arêtes entre sommets ayant au moins un voisin commun, puis pondération selon la formule de Czekanovski-Dice :

$$1 - d_{C\text{-}D}(x,y) = 1 - \frac{|\Delta(N[x],N[y])|}{|N[x]|+|N[y]|}$$

Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

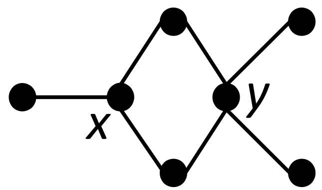
- **perturber** les données
- refaire tourner l'algorithme
- obtenir une nouvelle solution

fournir une solution consensus dont la robustesse est évaluée

Perturbation des données :

- modification aléatoire des poids des arêtes dans l'intervalle $[1-\varepsilon,1+\varepsilon]$
- ajout aléatoire d'arêtes entre sommets ayant au moins un voisin commun, puis pondération selon la formule de Czekanovski-Dice :

$$1 - d_{C\text{-}D}(x,y) = 1 - \frac{|\Delta(N[x],N[y])|}{|N[x]|+|N[y]|}$$



Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

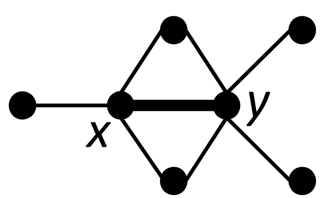
- **perturber** les données
- refaire tourner l'algorithme
- obtenir une nouvelle solution

fournir une solution consensus
dont la robustesse est évaluée

Perturbation des données :

- modification aléatoire des poids des arêtes dans l'intervalle [1-ε,1+ε]
- ajout aléatoire d'arêtes entre sommets ayant au moins un voisin commun, puis pondération selon la formule de Czekanovski-Dice :

$$1 - d_{C\text{-}D}(x,y) = 1 - \frac{|\Delta(N[x],N[y])|}{|N[x]|+|N[y]|}$$



$$1 - d_{C\text{-}D}(x,y) = 1 - 3/(5+6)$$
$$\approx 0.63$$

Partitionnement “bootstrap”

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

- **perturber** les données
- refaire tourner l'algorithme
- obtenir une nouvelle solution

fournir une **solution consensus** dont la robustesse est évaluée

Consensus de partitions :

- Problème de Régnier : par rapport à un ensemble de partitions $P_1 \ldots P_k$, trouver une partition P qui minimise la somme des nombres de paires d'éléments réunis dans P et séparés dans P_i , ou le contraire. Régnier 1965

$$\min \sum_i d(P,P_i) \text{ où } d(P,P_i) = |\Delta(\{\{x,y\}, x,y \in C_m \in P\}, \{\{x,y\}, x,y \in C'_n \in P_i\})|$$

Partitionnement "bootstrap"

Robustesse du partitionnement obtenu ?

En apprentissage automatique, en phylogénie : **bootstrap**

- **perturber** les données
- refaire tourner l'algorithme
- obtenir une nouvelle solution

fournir une $\boxed{\textbf{solution consensus}}$ dont la robustesse est évaluée

Consensus de partitions :

- Problème de Régnier : par rapport à un ensemble de partitions $P_1 ... P_k$, trouver une partition P qui minimise la somme des nombres de paires d'éléments réunis dans P et séparés dans P_i , ou le contraire. Régnier 1965

$$\min \sum_i \mathrm{d}(P,P_i) \text{ où } \mathrm{d}(P,P_i)=|\Delta(\{\{x,y\},\ x,y \in C_m \in P\},\ \{\{x,y\},\ x,y \in C'_n \in P_i\})|$$

- NP-complet si k n'est pas fixé, complexité inconnue sinon.

Wakabayashi 1986, Charon & Hudry 2007

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

Problème de partitionnement en cliques

Problème de **partitionnement en cliques**
Entrée : graphe *G* complet, pondéré par des arêtes de poids positif ou négatif
Sortie : partition de *G* de poids maximum

Grötschel & Wakabayashi 1989

Problème de partitionnement en cliques

Problème de **partitionnement en cliques**
Entrée : graphe G complet, pondéré par des arêtes de poids positif ou négatif
Sortie : partition de G de poids maximum

Grötschel & Wakabayashi 1989

Problème	Pondération des arêtes
Problème de Régnier (partition consensus)	$M(\{P_1 \ldots P_k\},P) = \sum_{u,v} (\lvert\{P_i \,/\, u,v \in C_j \in P_i\}\rvert - k/2)\, \alpha_{uv}$
Problème de Zahn (cluster editing)	$M(G,P) = \sum_{u,v} (-1)^{A_{uv}+1}\, \alpha_{uv}$
Modularité	$M(G,P) = \frac{1}{2m} \sum_{u,v} (A_{uv} - \frac{d(u)d(v)}{2m})\alpha_{uv}$

$\alpha_{uv} =$ 1 si u et v dans la même classe de P
0 sinon

Wakabayashi 1986, Newman 2004

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

TFit

TFit : Transferts – Fusions itérés

- Heuristique pour le partitionnement en cliques (matrice d'adjacence)

- Heuristique pour le partitionnement par modularité (listes d'adjacence)

TFit

TFit : Transferts – Fusions itérés

- Heuristique pour le partitionnement en cliques (matrice d'adjacence)

  Rapide (< 1 min) pour des graphes à 10000 sommets / partitions à 10000 éléments

- Heuristique pour le partitionnement par modularité (listes d'adjacence)

TFit

TFit : Transferts – Fusions itérés

- Heuristique pour le partitionnement en cliques (matrice d'adjacence)

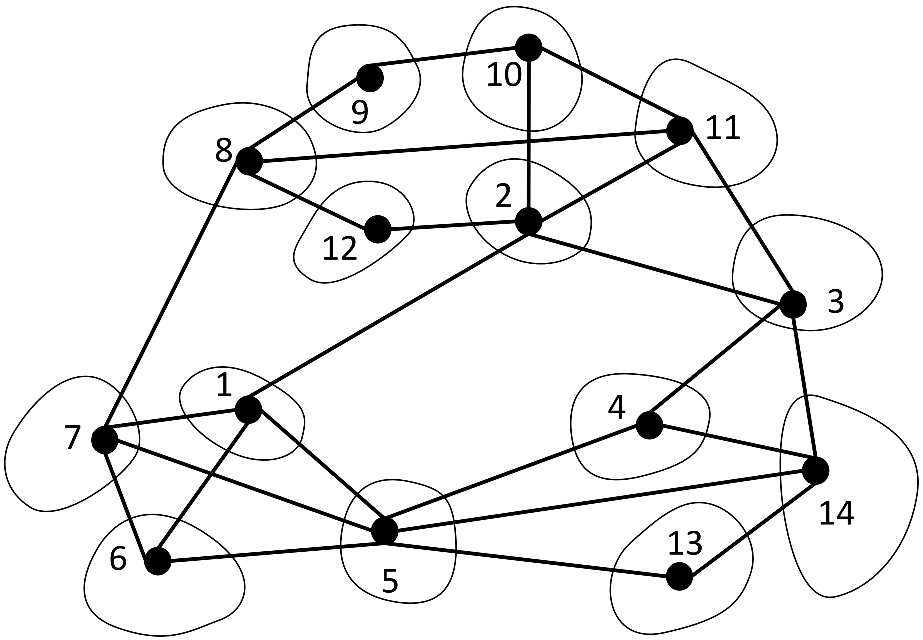
  Rapide (< 1 min) pour des graphes à 10000 sommets / partitions à 10000 éléments

- Heuristique pour le partitionnement par modularité (listes d'adjacence)

  Même principe que l'algorithme de Louvain (Blondel et al. 2008) mais on ajoute des étapes de transferts de sommets : **moins rapide, mais meilleurs résultats**.

TFit

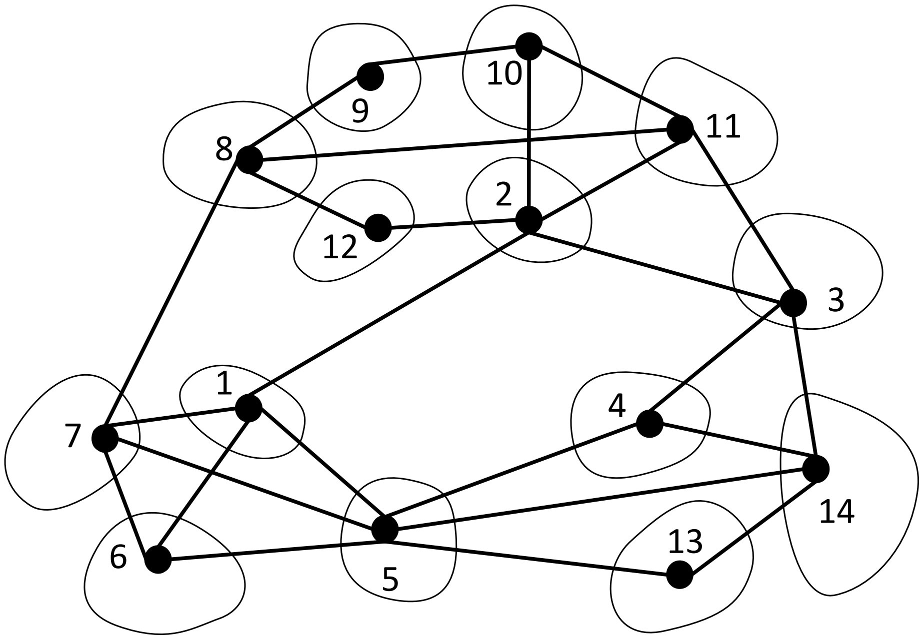
TFit : Transferts – Fusions itérés



Même principe que l'algorithme de Louvain (Blondel et al. 2008) mais on ajoute des étapes de transferts de sommets : **moins rapide, mais meilleurs résultats**.

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

T

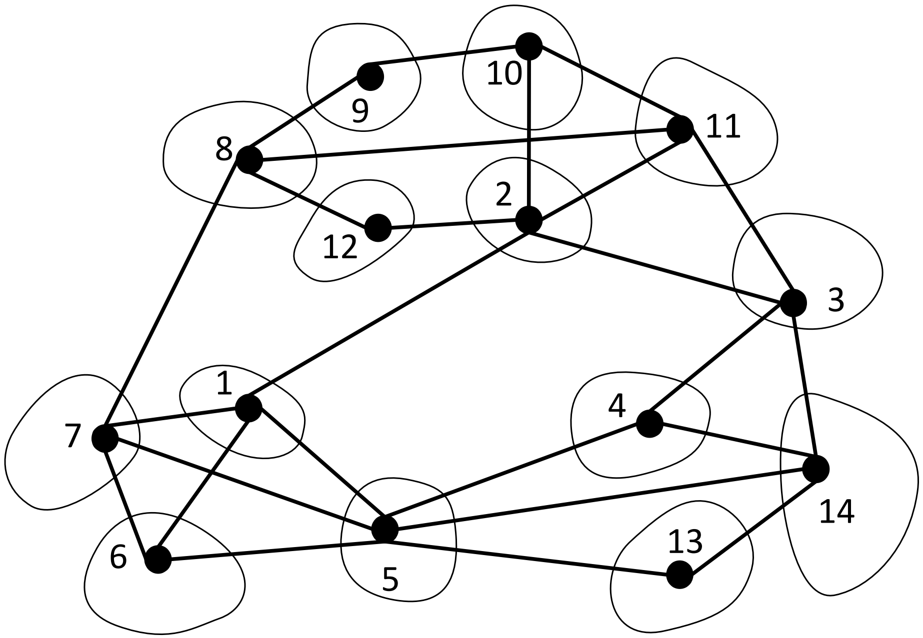
- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

F

Même principe que l'algorithme de Louvain (Blondel et al. 2008) mais on ajoute des étapes de transferts de sommets : **moins rapide, mais meilleurs résultats**.

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

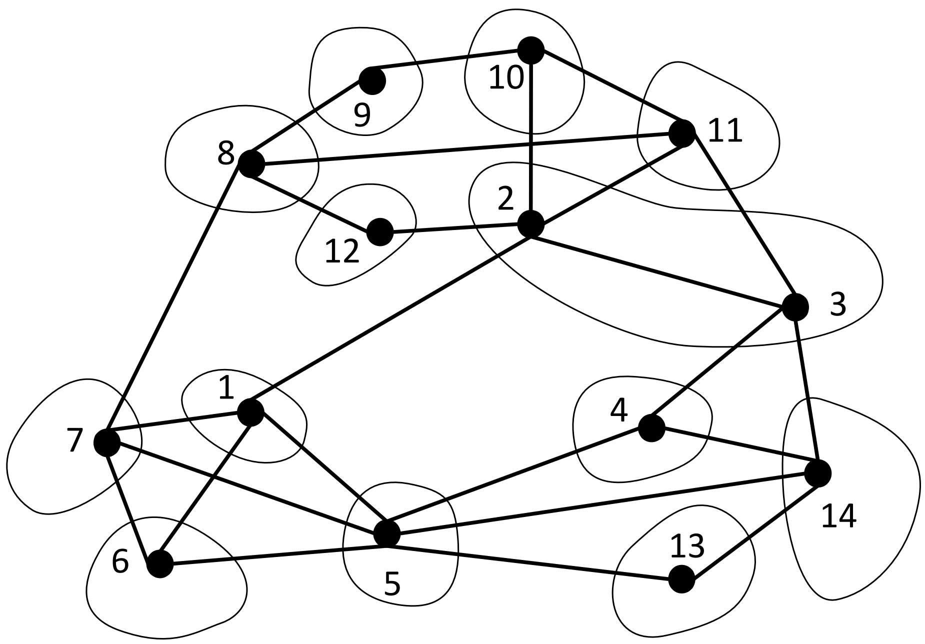
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

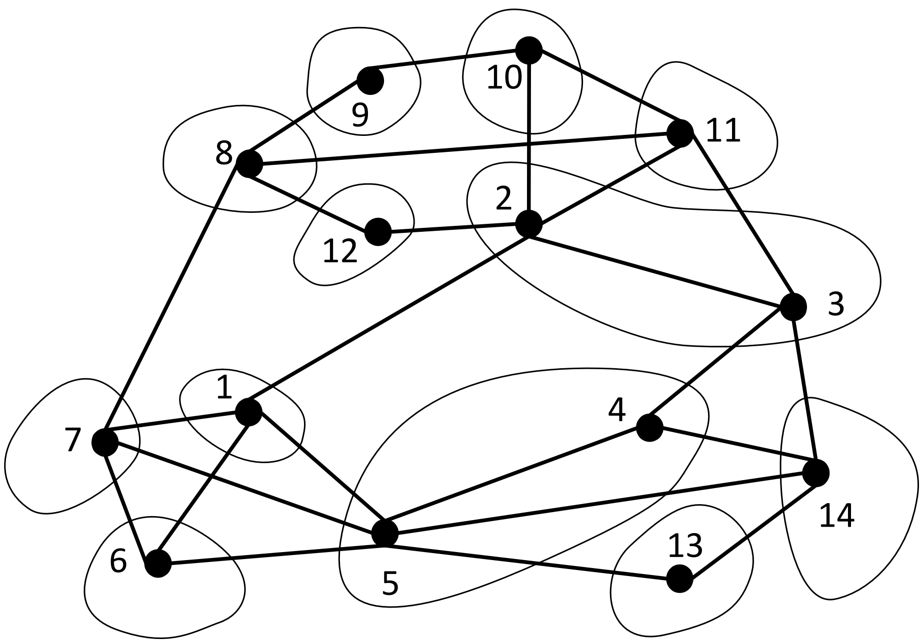
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

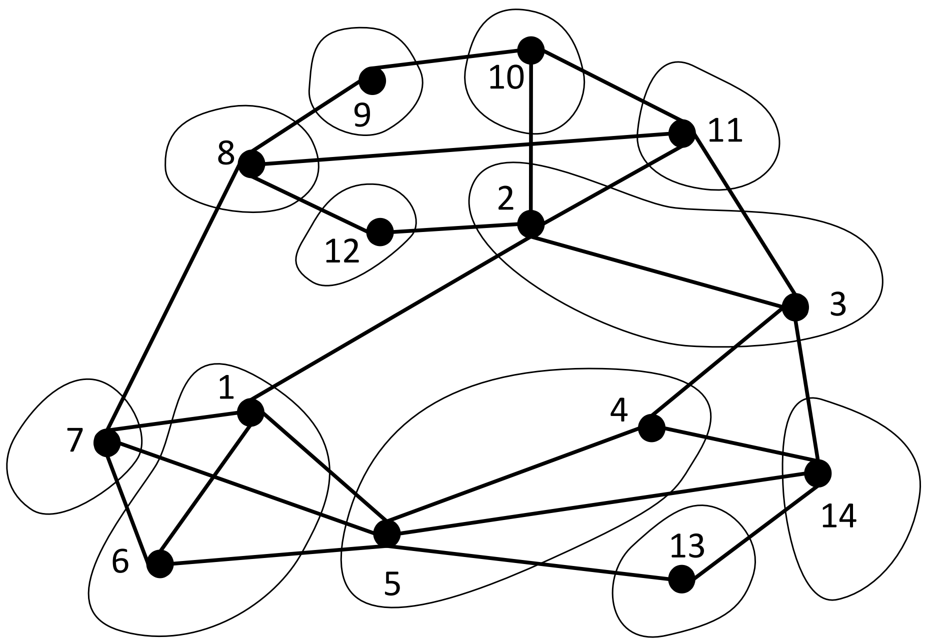
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

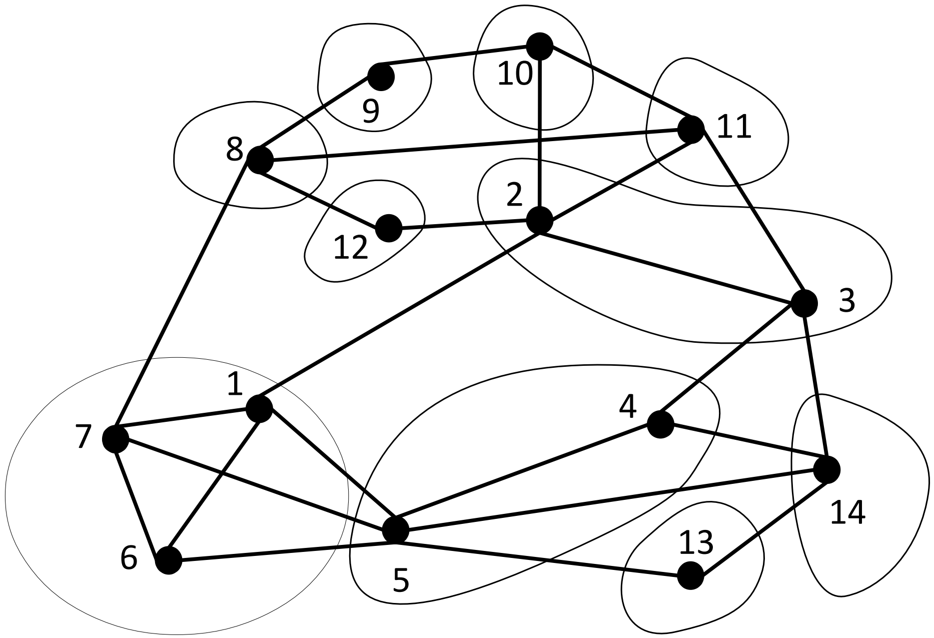
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

T

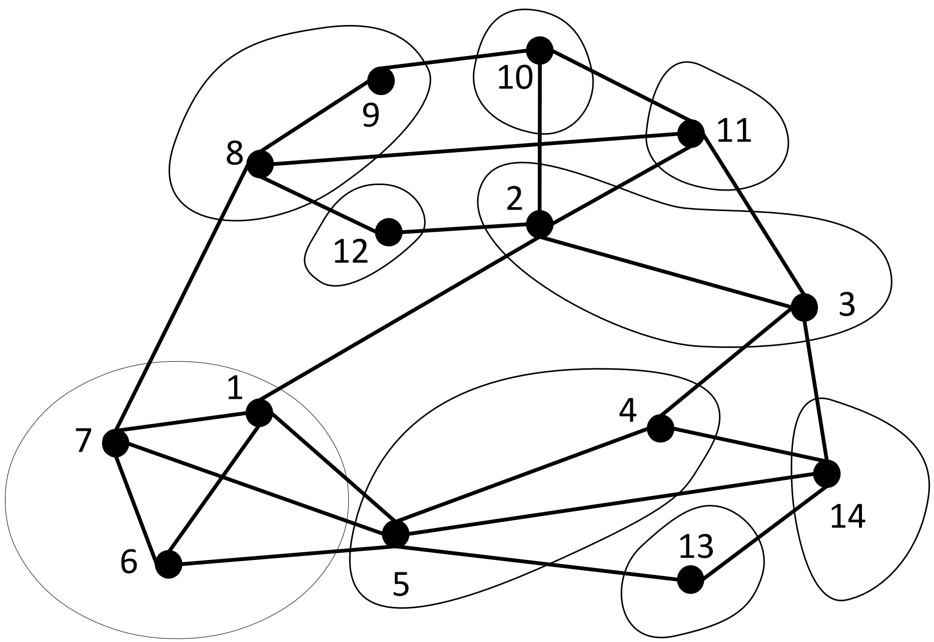
- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

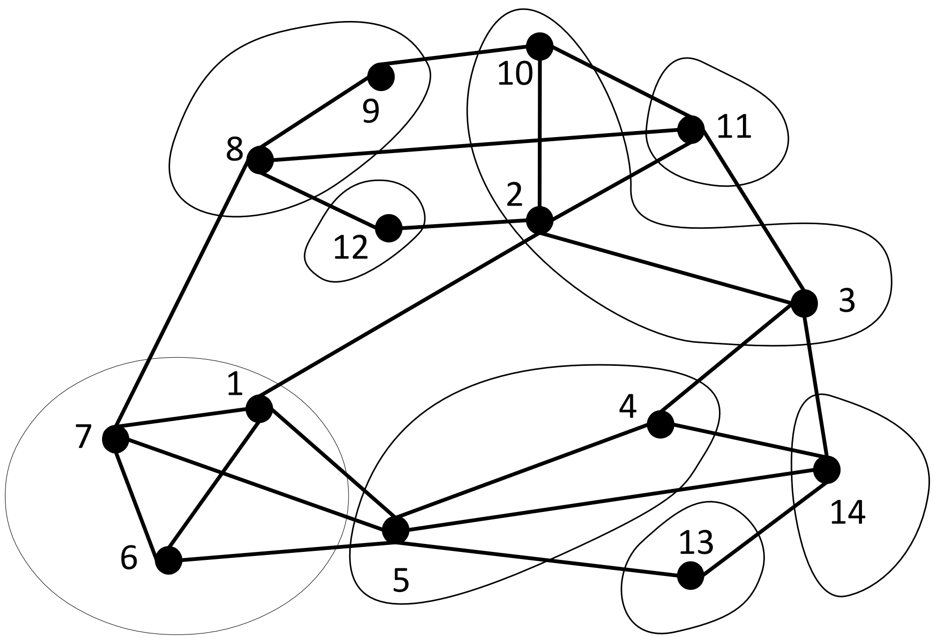
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

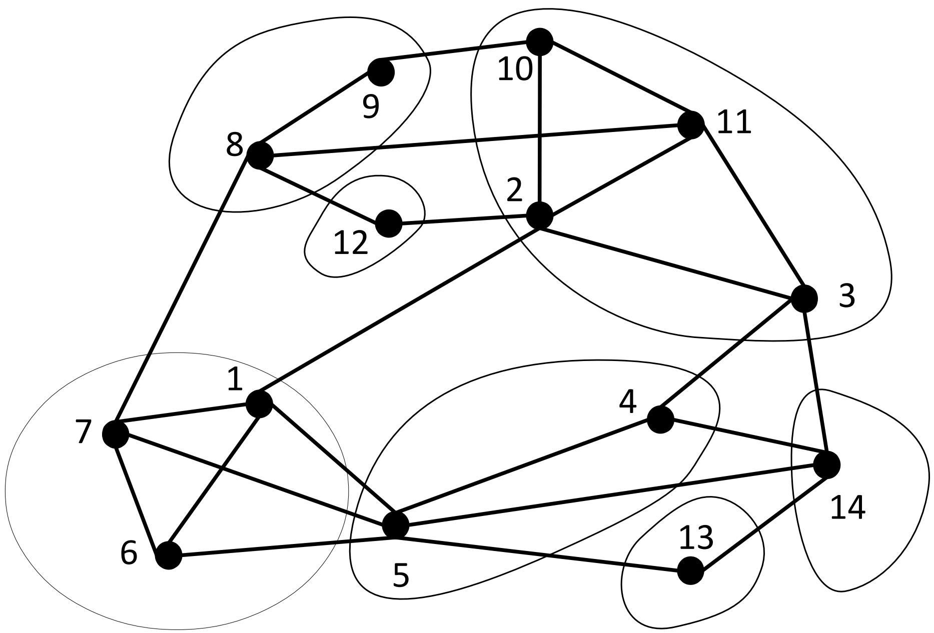
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

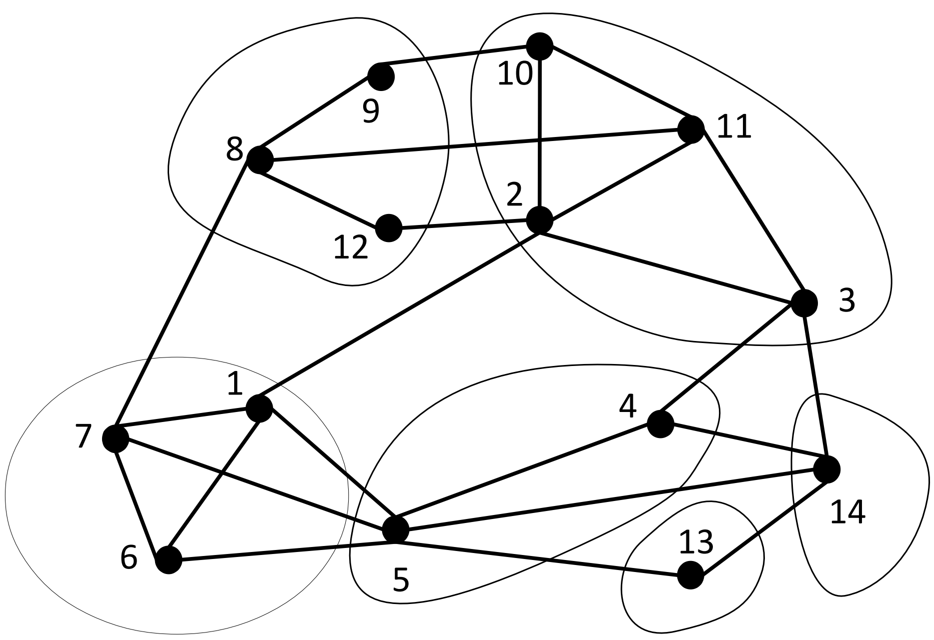
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

T

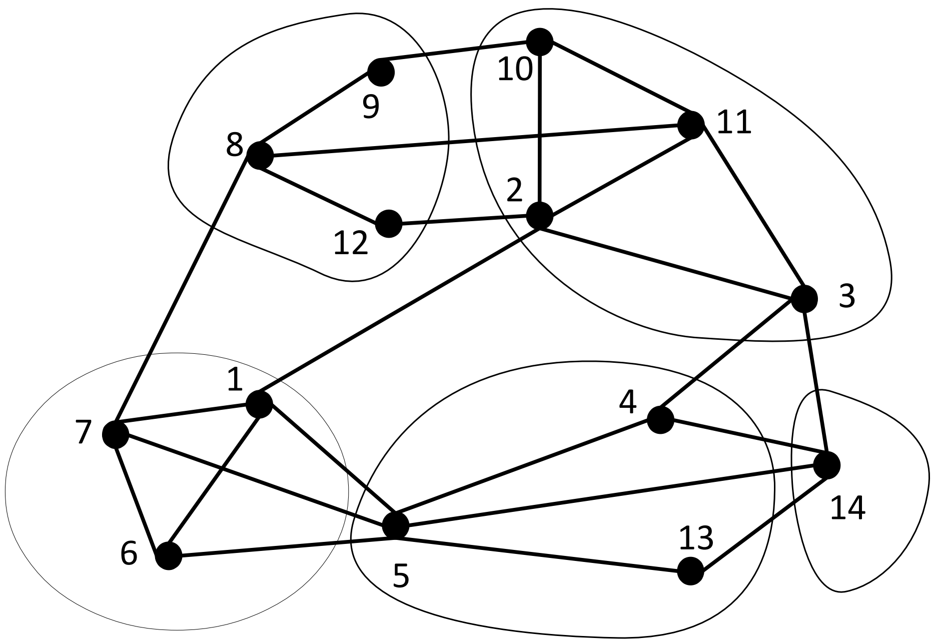
- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus.

F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

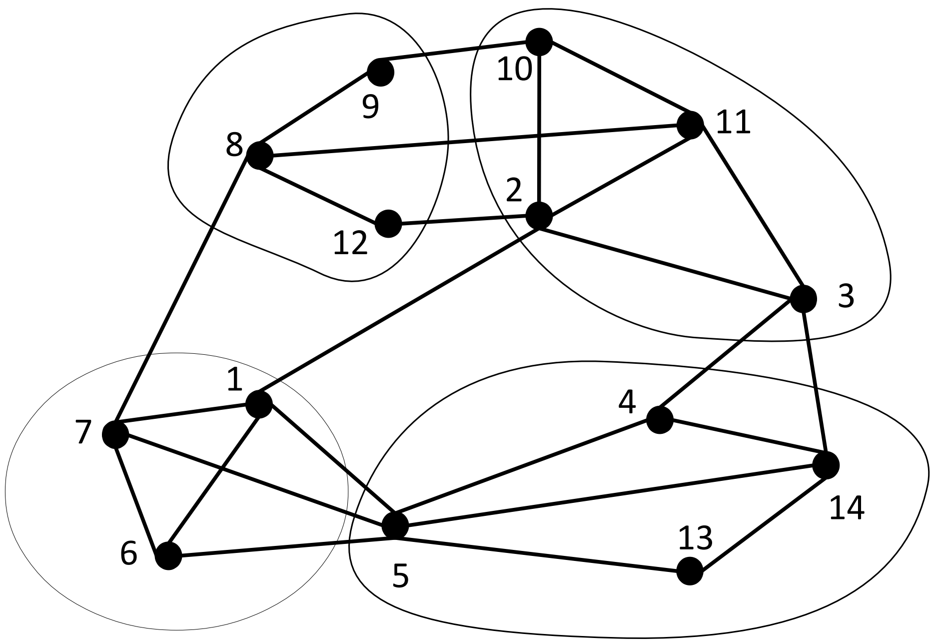
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



modularité : 0.33

Initialement :
classes = singletons

Tant qu'on améliore la modularité :

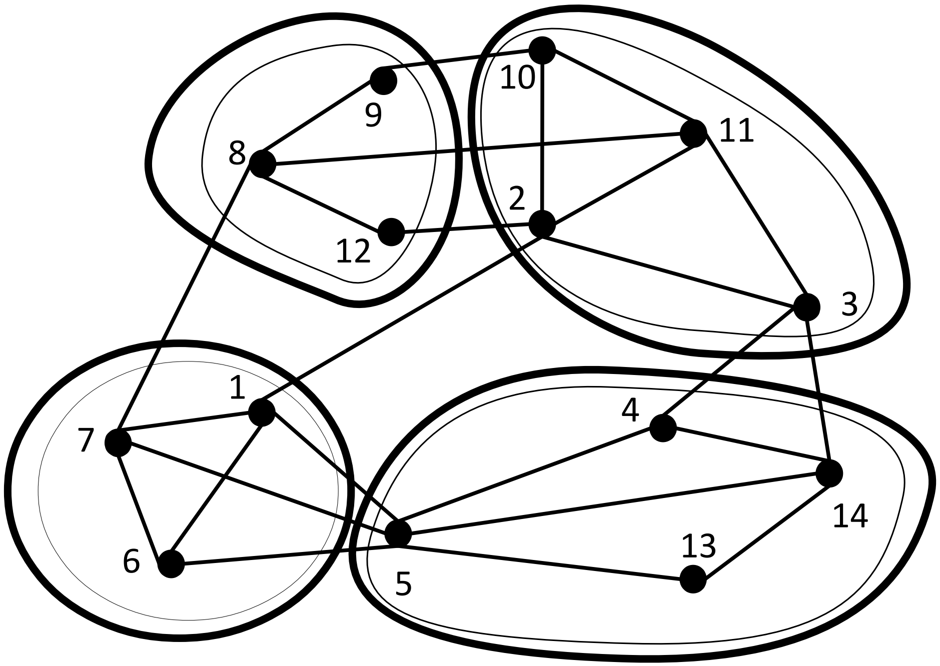
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

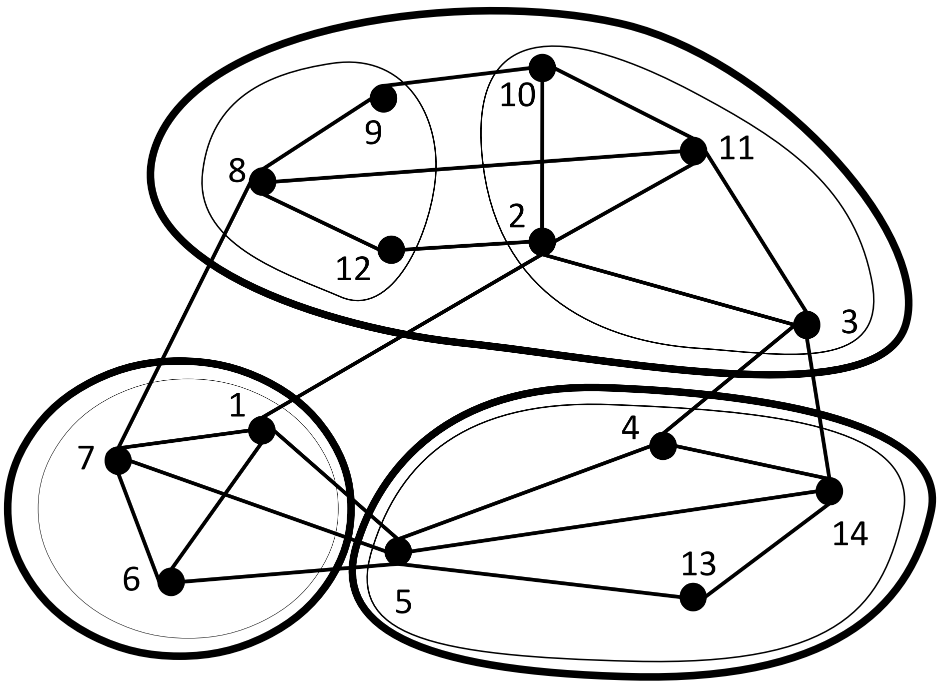
Tant qu'on améliore la modularité :

- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

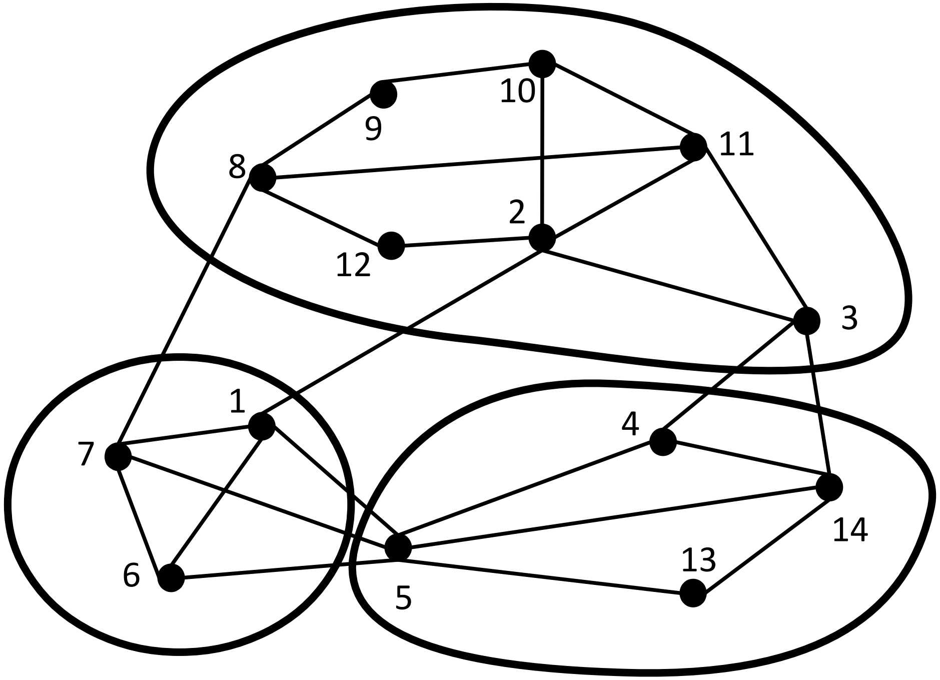
Tant qu'on améliore la modularité :

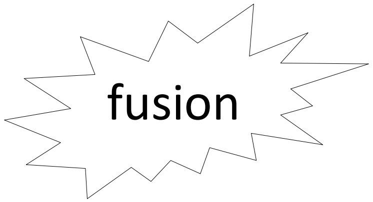
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

TFit

TFit : Transferts – Fusions itérés





Initialement :
classes = singletons

Tant qu'on améliore la modularité :

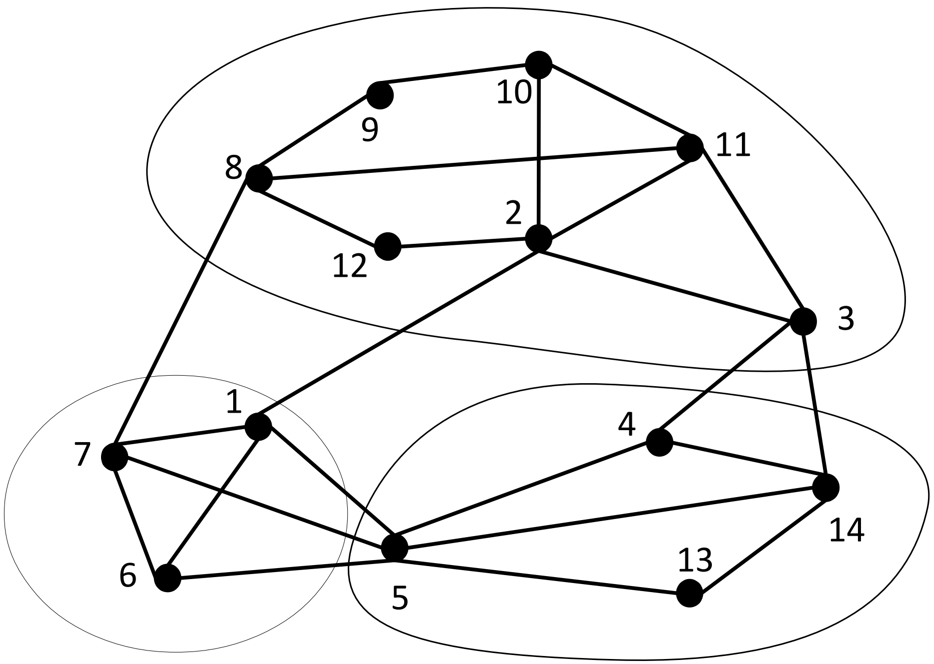
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



~~**modularité : 0.33**~~
modularité : 0.35

Initialement :
classes = singletons

Tant qu'on améliore la modularité :

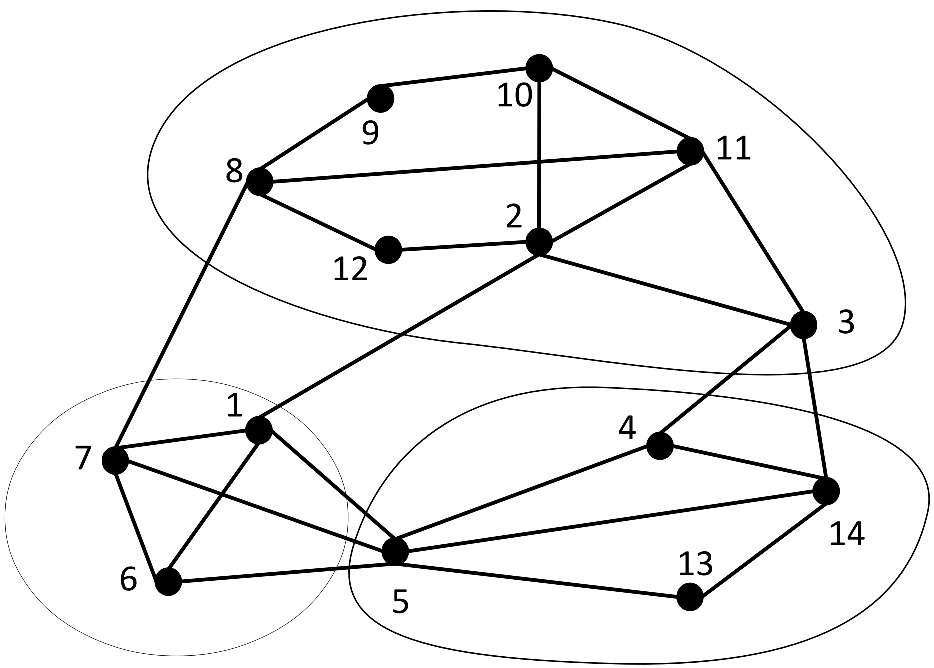
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

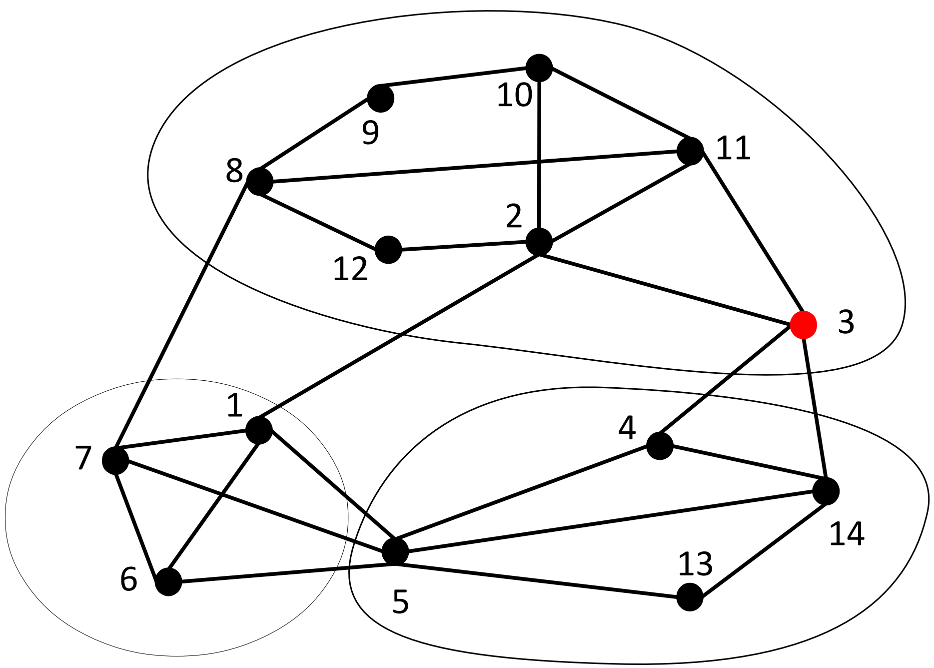
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

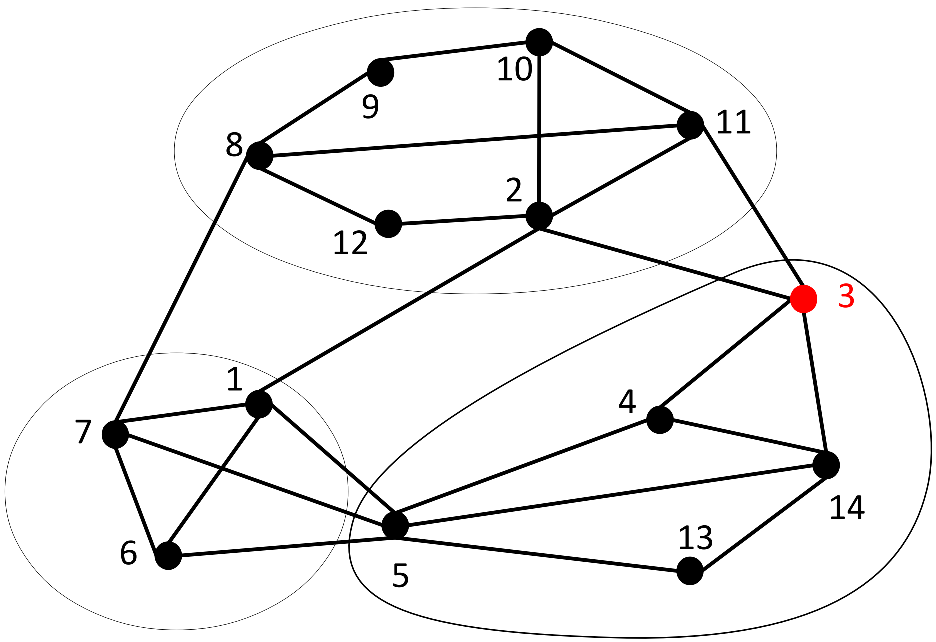
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

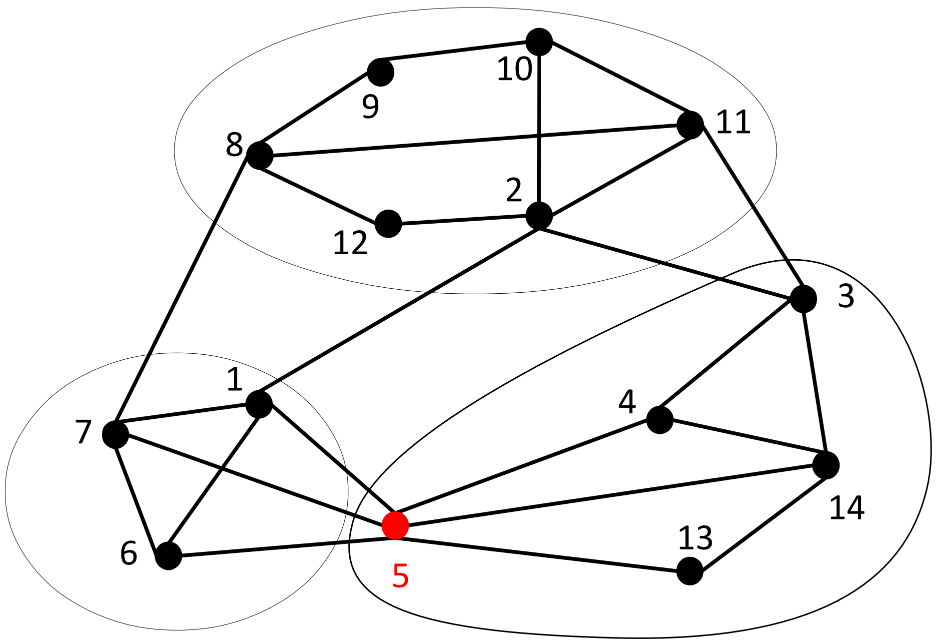
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

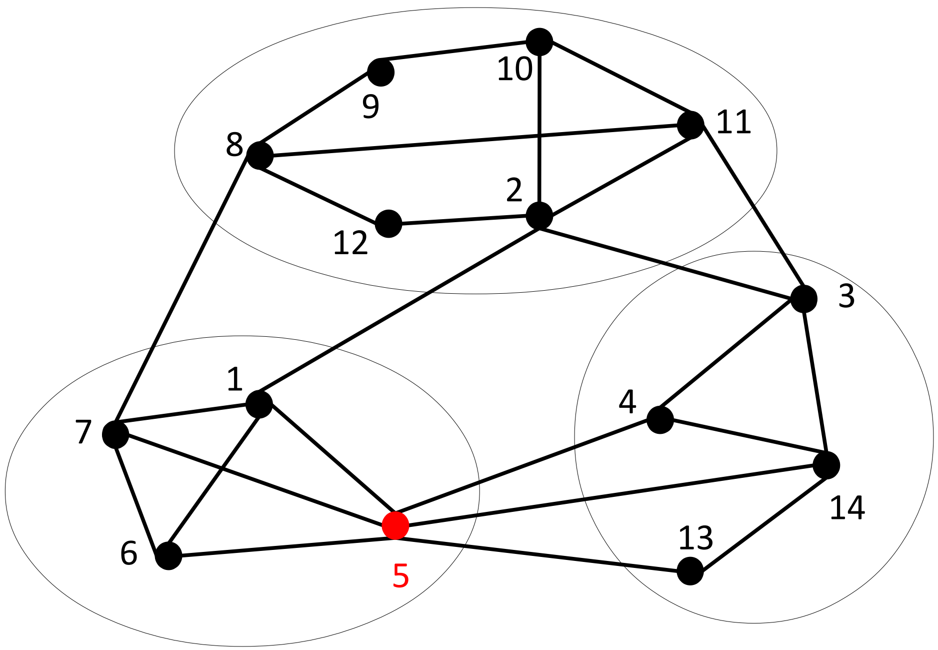
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. **T**

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. **F**

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

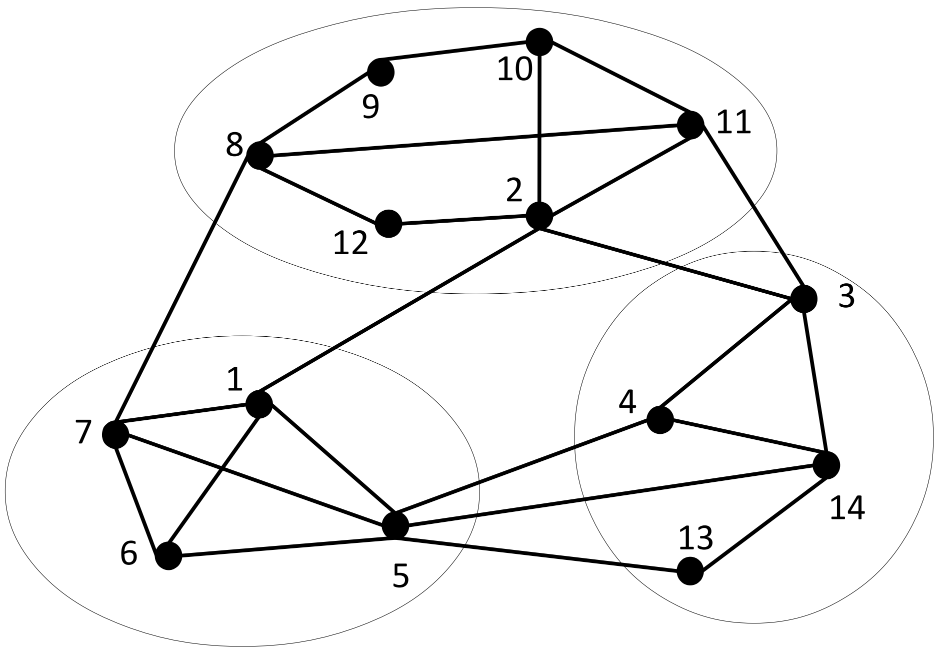
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



~~**modularité : 0.35**~~
modularité : 0.38

Initialement :
classes = singletons

Tant qu'on améliore la modularité :

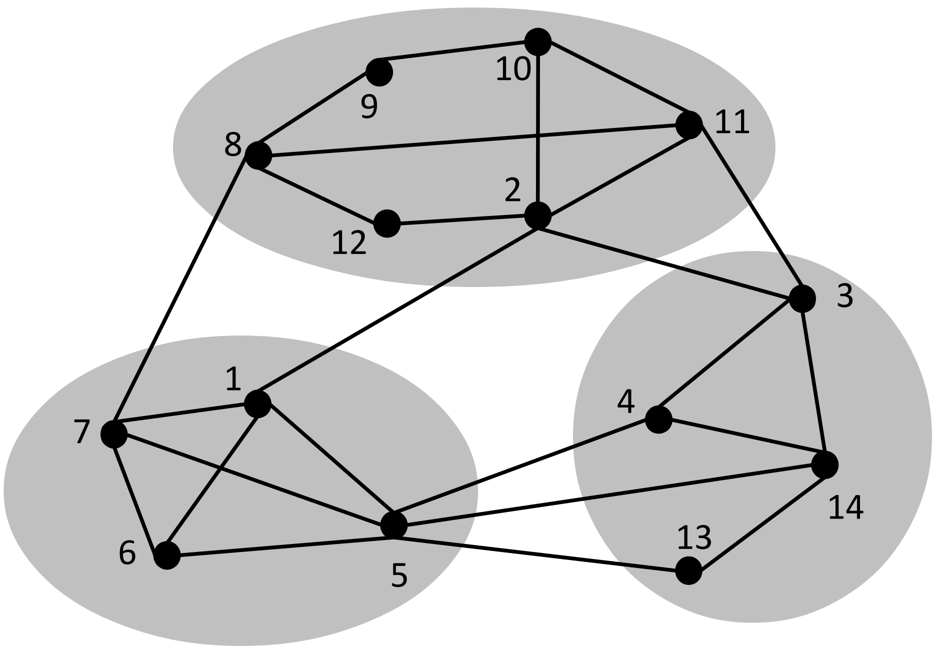
- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

Gambette & Guénoche, 2012

TFit

TFit : Transferts – Fusions itérés



Initialement :
classes = singletons

Tant qu'on améliore la modularité :

- Pour chaque sommet, si une classe améliore la modularité, transfert vers celle qui l'améliore le plus. T

- Pour chaque classe, si la fusion avec une classe améliore la modularité, transfert vers celle qui l'améliore le plus. F

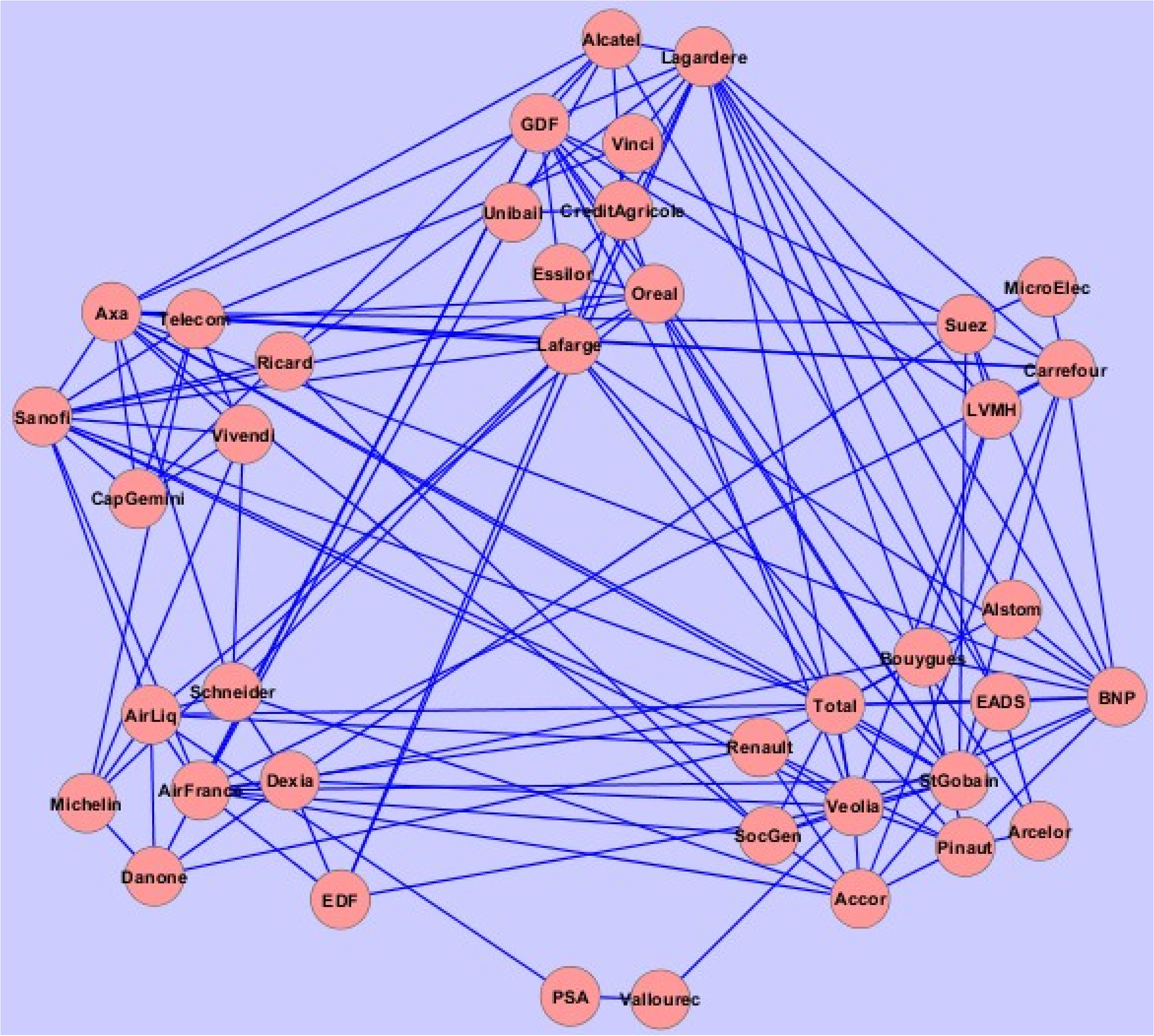
Gambette & Guénoche, 2012

Résultats de TFit

graph	n	m	Opt	Louvain	N-R	TFit
Dolphins	62	159	.5285	.5185	**.5276**	.5268
polBooks	105	441	.5272	.5266	**.5272**	.5268
afootball	115	613	.6046	**.6046**	.6045	**.6046**
A01	249	635	.6329	.6145	**.6293**	.6284
USAir97	332	2126	.3682	.3541	**.3678**	.3595
netscience	379	914	.8486	**.8475**	.8474	**.8475**
s388	512	819	.8194	.7962	.8143	**.8148**
emails	1133	5452		.5438	**.5816**	.5722

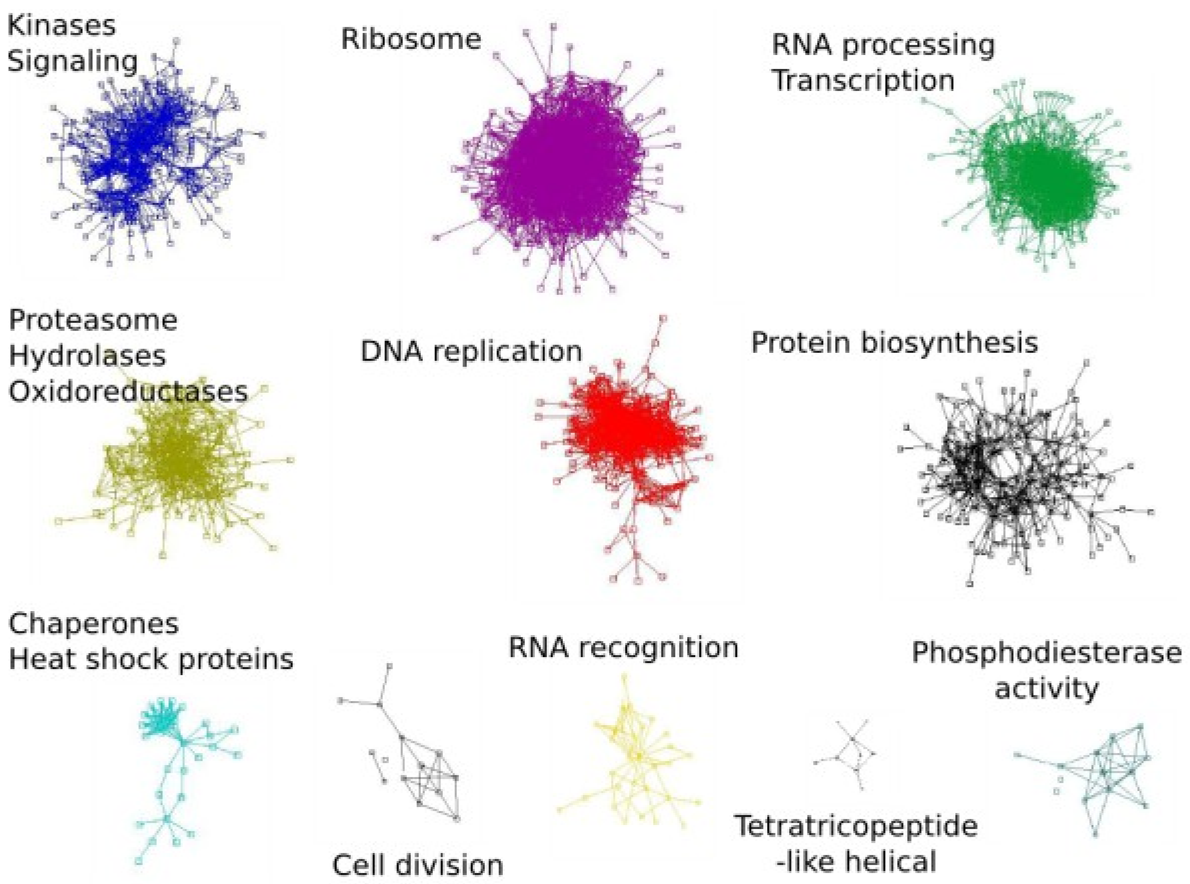
Opt (programmation linéaire) : Aloïse et al., 2010
Louvain (1 heuristique) : Blondel et al, 2008
N-R (10 heuristiques) : Noack & Rotta, 2009
Tfit (1 heuristique) : Gambette & Guénoche, 2012

Résultats de TFit



Plugin Modclust (TFit) pour Cytoscape

Résultats du partitionnement “bootstrap”



Fonctions biologiques sur-représentées dans les classes identifiées

Anaïs Baudot, Alain Guénoche (IML)

Résultats du partitionnement “bootstrap”

Étude de simulation :

• construction de graphes de 200 sommets à partir d'une partition source en 5 classes de 40 sommets

• 3 graphes de plus en plus difficiles (plus d'arêtes inter-classes, moins d'intra-classes)

• comparaison entre partition originale et finale par l'indice de Rand corrigé

• évaluation de la robustesse des partitions

Gambette & Guénoche, 2012

Résultats du partitionnement “bootstrap”

Étude de simulation :

• construction de graphes de 200 sommets à partir d'une partition source en 5 classes de 40 sommets

• 3 graphes de plus en plus difficiles (plus d'arêtes inter-classes, moins d'intra-classes)

• comparaison entre partition originale et finale par l'indice de Rand corrigé

• évaluation de la robustesse des partitions

• Perturbation 1 (élongation d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.01	.02	.03	.01		.02		.03	
G1	.30	.10	.825	.881	.880	.880	.888	.939	.886	.938	.886	.938
G2	.20	.05	.689	.793	.798	.797	.737	.842	.734	.841	.731	.842
G3	.10	.01	.615	.682	.683	.678	.719	.835	.713	.829	.708	.828

• Perturbation 2 (ajout et pondération d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.30	.50	.70	.30		.50		.70	
G1	.30	.10	.825	.833	.828	.815	.767	.845	.777	.847	.805	.868
G2	.20	.05	.689	.768	.775	.772	.700	.827	.732	.852	.758	.880
G3	.10	.01	.615	.695	.723	.728	.694	.816	.744	.852	.772	.885

Gambette & Guénoche, 2012

Résultats du partitionnement “bootstrap”

Étude de simulation :

• construction de graphes de 200 sommets à partir d'une partition source en 5 classes de 40 sommets

• 3 graphes **de plus en plus difficiles** (plus d'arêtes inter-classes, moins d'intra-classes)

• comparaison entre partition originale et finale par l'indice de Rand corrigé

• évaluation de la robustesse des partitions

• Perturbation 1 (élongation d'arêtes)

	d_i	d_e	Rand P_{ini}	Rand P_{cons} .01	Rand P_{cons} .02	Rand P_{cons} .03	Robustness .01 P_{ini}	Robustness .01 P_{cons}	Robustness .02 P_{ini}	Robustness .02 P_{cons}	Robustness .03 P_{ini}	Robustness .03 P_{cons}
G1	.30	.10	.825	.881	.880	.880	.888	.939	.886	.938	.886	.938
G2	.20	.05	.689	.793	.798	.797	.737	.842	.734	.841	.731	.842
G3	.10	.01	.615	.682	.683	.678	.719	.835	.713	.829	.708	.828

• Perturbation 2 (ajout et pondération d'arêtes)

	d_i	d_e	Rand P_{ini}	Rand P_{cons} .30	Rand P_{cons} .50	Rand P_{cons} .70	Robustness .30 P_{ini}	Robustness .30 P_{cons}	Robustness .50 P_{ini}	Robustness .50 P_{cons}	Robustness .70 P_{ini}	Robustness .70 P_{cons}
G1	.30	.10	.825	.833	.828	.815	.767	.845	.777	.847	.805	.868
G2	.20	.05	.689	.768	.775	.772	.700	.827	.732	.852	.758	.880
G3	.10	.01	.615	.695	.723	.728	.694	.816	.744	.852	.772	.885

Gambette & Guénoche, 2012

Résultats du partitionnement "bootstrap"

Étude de simulation :

• construction de graphes de 200 sommets à partir d'une partition source en 5 classes de 40 sommets

• 3 graphes de plus en plus difficiles (plus d'arêtes inter-classes, moins d'intra-classes)

• **comparaison entre partition originale et finale par l'indice de Rand corrigé**

• évaluation de la robustesse des partitions

• Perturbation 1 (élongation d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.01	.02	.03	.01		.02		.03	
G1	.30	.10	.825	.881	.880	.880	.888	.939	.886	.938	.886	.938
G2	.20	.05	.689	.793	.798	.797	.737	.842	.734	.841	.731	.842
G3	.10	.01	.615	.682	.683	.678	.719	.835	.713	.829	.708	.828

• Perturbation 2 (ajout et pondération d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.30	.50	.70	.30		.50		.70	
G1	.30	.10	.825	.833	.828	.815	.767	.845	.777	.847	.805	.868
G2	.20	.05	.689	.768	.775	.772	.700	.827	.732	.852	.758	.880
G3	.10	.01	.615	.695	.723	.728	.694	.816	.744	.852	.772	.885

Gambette & Guénoche, 2012

Résultats du partitionnement “bootstrap”

Étude de simulation :

• construction de graphes de 200 sommets à partir d'une partition source en 5 classes de 40 sommets

• 3 graphes de plus en plus difficiles (plus d'arêtes inter-classes, moins d'intra-classes)

• comparaison entre partition originale et finale par l'indice de Rand corrigé

• **évaluation de la robustesse des partitions**

• Perturbation 1 (élongation d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.01	.02	.03	.01		.02		.03	
G1	.30	.10	.825	.881	.880	.880	.888	.939	.886	.938	.886	.938
G2	.20	.05	.689	.793	.798	.797	.737	.842	.734	.841	.731	.842
G3	.10	.01	.615	.682	.683	.678	.719	.835	.713	.829	.708	.828

• Perturbation 2 (ajout et pondération d'arêtes)

			Rand				Robustness					
			P_{ini}	P_{cons}			P_{ini}	P_{cons}	P_{ini}	P_{cons}	P_{ini}	P_{cons}
	d_i	d_e		.30	.50	.70	.30		.50		.70	
G1	.30	.10	.825	.833	.828	.815	.767	.845	.777	.847	.805	.868
G2	.20	.05	.689	.768	.775	.772	.700	.827	.732	.852	.758	.880
G3	.10	.01	.615	.695	.723	.728	.694	.816	.744	.852	.772	.885

Gambette & Guénoche, 2012

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

Comparaison de résultats de partitionnements

Plusieurs **distances possibles entre partitions** :

- distance de Rand
- distance de Rand corrigée
- distance des transferts
- …

Guénoche & Denoeud, 2006

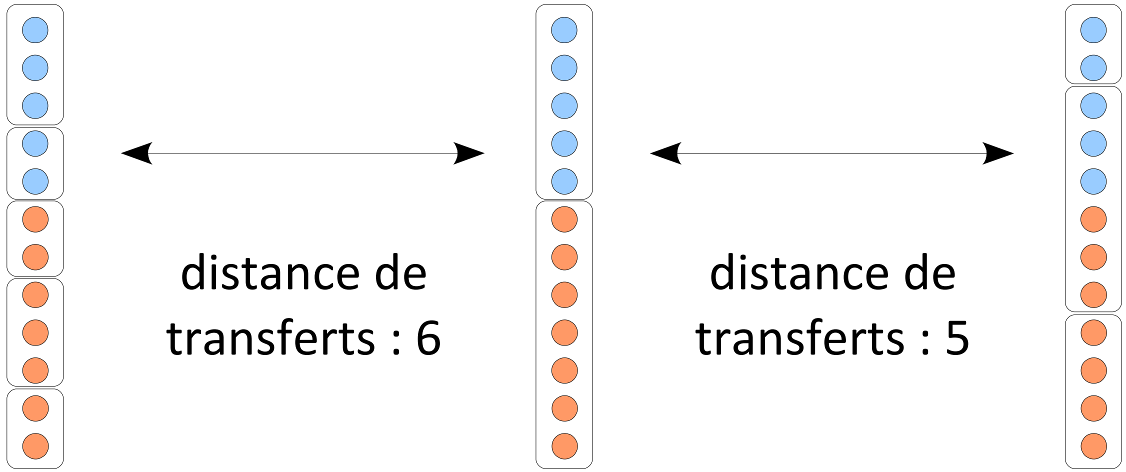
Comparaison de résultats de partitionnements

Plusieurs **distances possibles entre partitions** :

- distance de Rand
- distance de Rand corrigée
- distance des transferts
- …

Guénoche & Denoeud, 2006

Si le nombre de classes est très différent :



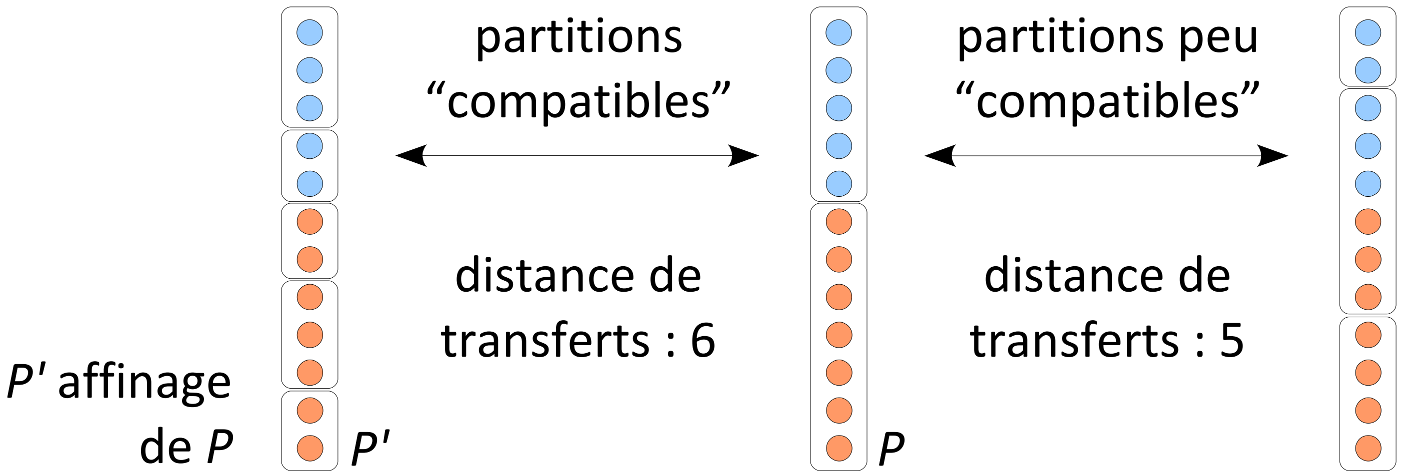
Comparaison de résultats de partitionnements

Plusieurs **distances possibles entre partitions** :

- distance de Rand
- distance de Rand corrigée
- distance des transferts
- …

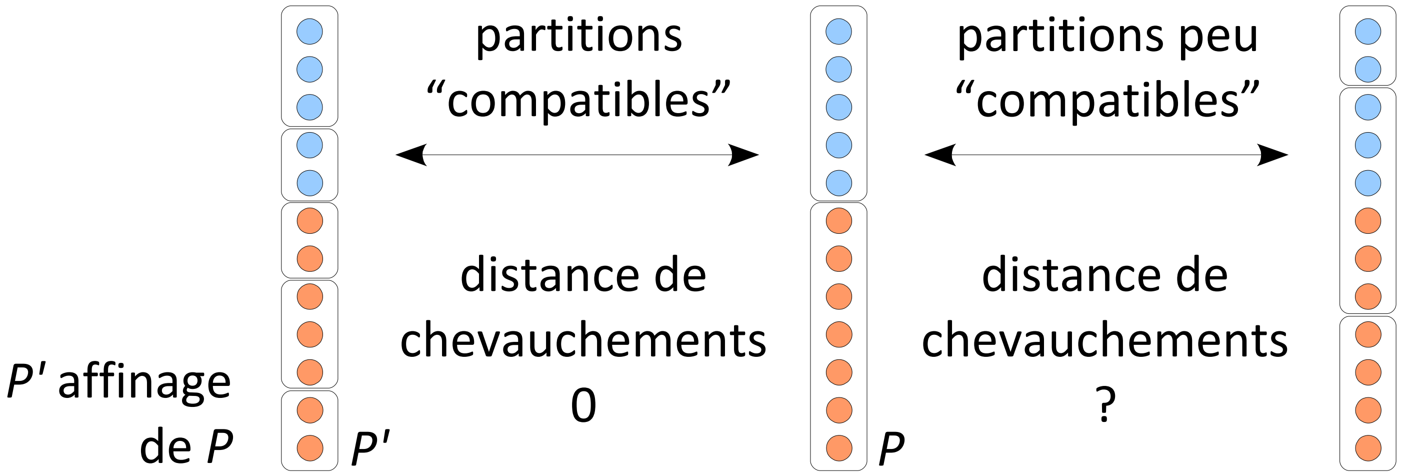
Guénoche & Denoeud, 2006

Si le nombre de classes est très différent :



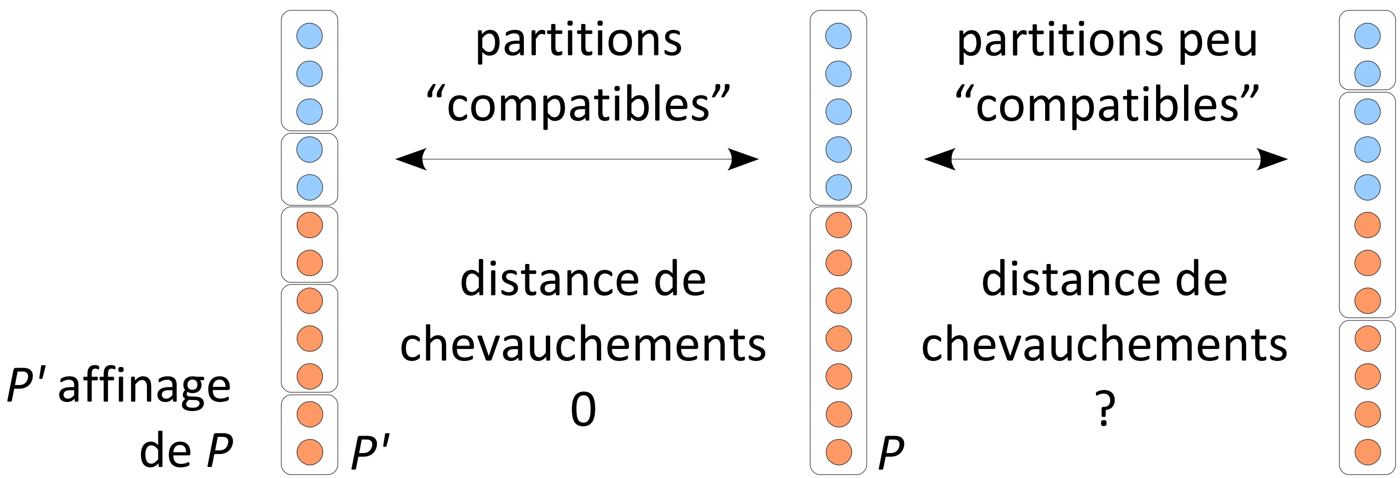
Une nouvelle distance entre partitions

d(P,P') = min |{éléments à retirer pour supprimer les chevauchements entre P et P'}|



Une nouvelle distance entre partitions

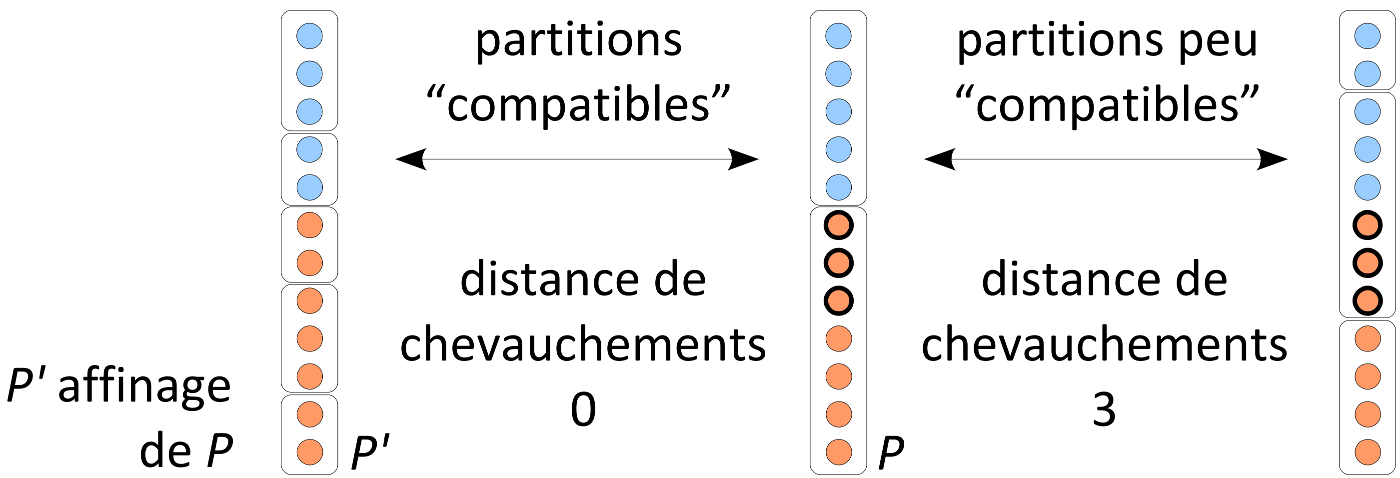
$d(P,P') = \min |\{$éléments à retirer pour supprimer les chevauchements entre P et $P'\}|$



Pas une **distance** (pas de propriété de séparation)
mais un **indice de dissimilarité**

Une nouvelle distance entre partitions

$d(P,P') = \min |\{$éléments à retirer pour supprimer les chevauchements entre P et $P'\}|$



Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Calcul de la distance de chevauchements : Maximum Compatible Subset, où *C* est l'ensemble des classes de deux partitions de *X*.

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X-R* ne contient pas de chevauchements.

Problème de décision associé : NP-complet

Steel & Hamel 1996

Algorithme de complexité paramétrée en $O^*(3^{|R|})$

Huson, Rupp, Berry, Gambette & Paul 2009

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

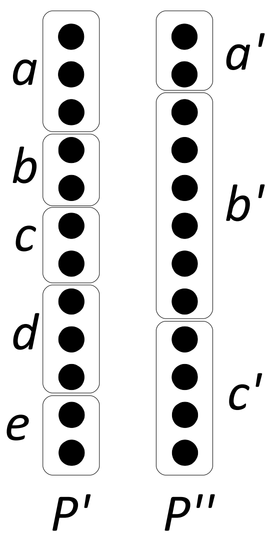
Problème de décision associé : NP-complet

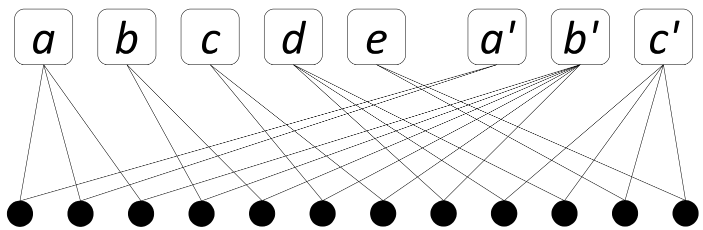
Steel & Hamel 1996

Algorithme de complexité paramétrée en O*($3^{|R|}$)

Huson, Rupp, Berry, Gambette & Paul 2009

Graphe des caractères :





Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X-R* ne contient pas de chevauchements.

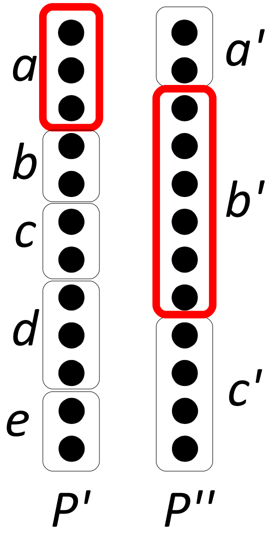
Problème de décision associé : NP-complet

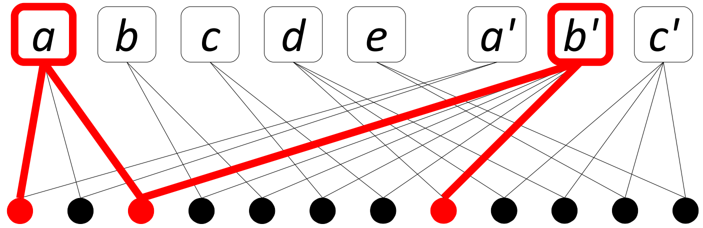
Steel & Hamel 1996

Algorithme de complexité paramétrée en $O^*(3^{|R|})$

Huson, Rupp, Berry, Gambette & Paul 2009

Graphe des caractères :





Deux ensembles se chevauchent si et seulement si le graphe des caractères **contient un "M"** comme sous-graphe induit

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble C d'ensembles d'éléments d'un ensemble X
Sortie : le plus petit sous-ensemble R de X tel que C restreint à X-R ne contient pas de chevauchements.

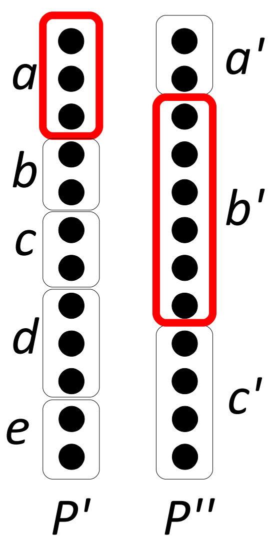
Problème de décision associé : NP-complet

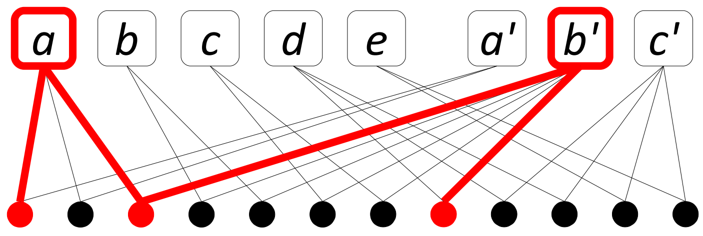
Steel & Hamel 1996

Algorithme de complexité paramétrée en $O^*(3^{|R|})$

Huson, Rupp, Berry, Gambette & Paul 2009

Graphe des caractères :





Deux ensembles se chevauchent si et seulement si le graphe des caractères **contient un "M"** comme sous-graphe induit

➡ retirer le nombre minimum de sommets du bas pour "casser les M" :

➡ 3-Hitting Set : $O^*(2.076^{|R|})$

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?

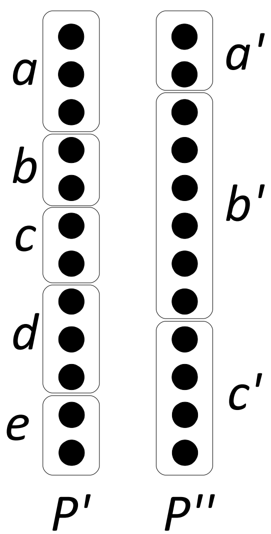
Distance de chevauchements et Max. Compatible Subset

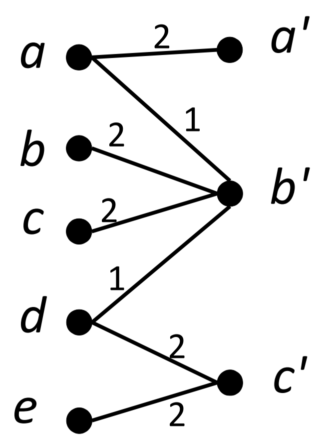
Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?
NP-complet

Gambette & Kim 2012

Graphe d'intersection des classes :





Problème : forêt couvrante d'étoiles de poids maximum dans un graphe biparti aux arêtes pondérées.

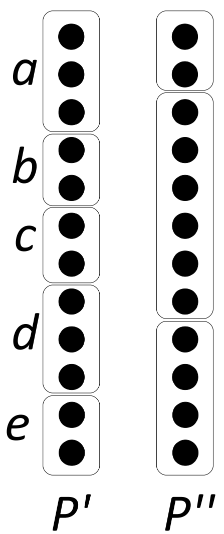
Distance de chevauchements et Max. Compatible Subset

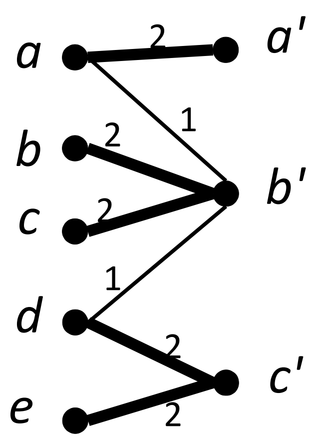
Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?
NP-complet

Gambette & Kim 2012

Graphe d'intersection des classes :





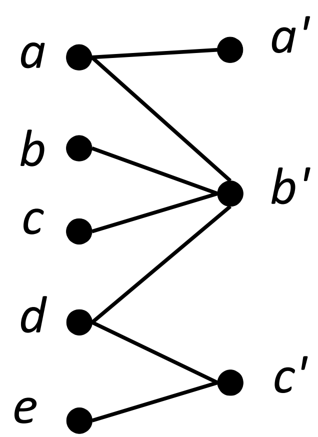
Problème : forêt couvrante d'étoiles de poids maximum dans un graphe biparti aux arêtes pondérées.

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?
NP-complet

Gambette & Kim 2012



Problème : forêt couvrante d'étoiles de poids maximum dans un graphe biparti aux arêtes pondérées.

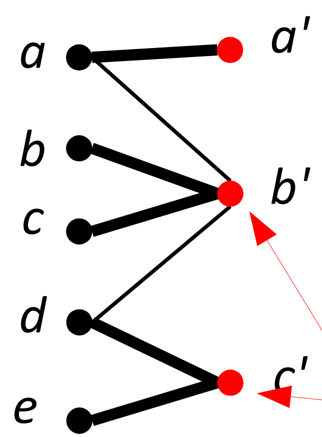
NP-complet dans un graphe où les arêtes sont pondérées à 1.

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X-R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?
NP-complet

Gambette & Kim 2012



Problème : forêt couvrante d'étoiles de poids maximum dans un graphe biparti aux arêtes pondérées.

NP-complet dans un graphe où les arêtes sont pondérées à 1.

centres des étoiles : ensemble dominant min.

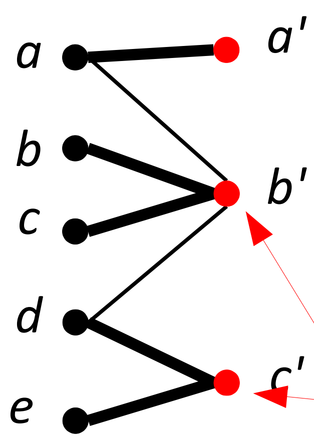
Chen, Engelberg et al 2007

Distance de chevauchements et Max. Compatible Subset

Problème **Maximum Compatible Subset** :
Entrée : ensemble *C* d'ensembles d'éléments d'un ensemble *X*
Sortie : le plus petit sous-ensemble *R* de *X* tel que *C* restreint à *X*-*R* ne contient pas de chevauchements.

Restriction aux cas où *C* est l'ensemble des classes de deux partitions de *X* ?
NP-complet

Gambette & Kim 2012



Problème : forêt couvrante d'étoiles de poids maximum dans un graphe biparti aux arêtes pondérées.

NP-complet dans un graphe où les arêtes sont pondérées à 1.

centres des étoiles : ensemble dominant min.
NP-complet pour graphes bipartis

Dewdney 1981

Plan

- Partitionnement par modularité
- Consensus de partitions et partitionnement “bootstrap”
- CPP, un problème de partitionnement en clique
- L'heuristique TFit
- Une nouvelle distance entre partitions
- Perspectives

Perspectives

- Optimisation du calcul de la **distance de chevauchement**

- Application de la méthodologie **partitionnement/comparaison des partitions** à la **comparaison de réseaux avec même ensemble de sommets**
    - réseaux biologiques provenant de sources différentes
    - réseaux sémantiques provenant de données textuelles ou cognitives
    - réseaux construits à partir de graphes bipartis (motifs d'expressions composées, noms et prénoms)

- Conception de **nouvelles méthodologies** de comparaison de réseaux avec même ensemble de sommets :
    - analyse simultanée de la structure des deux réseaux
    - nouveaux problèmes combinatoires ?

Remerciements

Merci pour votre attention !

Travail réalisé avec :

- Alain Guénoche, Laurent Tichit (IML, Marseille)
- Vincent Berry, Christophe Paul (LIRMM, Montpellier)
- Daniel Huson, Regula Rupp (ZBIT, Tübingen)
- Eunjung Kim (LAMSADE, Paris-Dauphine)