Two modularity-based clustering algorithms and a Cytoscape plugin for PPI networks

Laurent Tichit, Philippe Gambette et Alain Guénoche
Institut de Mathématiques de Luminy - CNRS
Université d'Aix-Marseille
tichit@univmed.fr

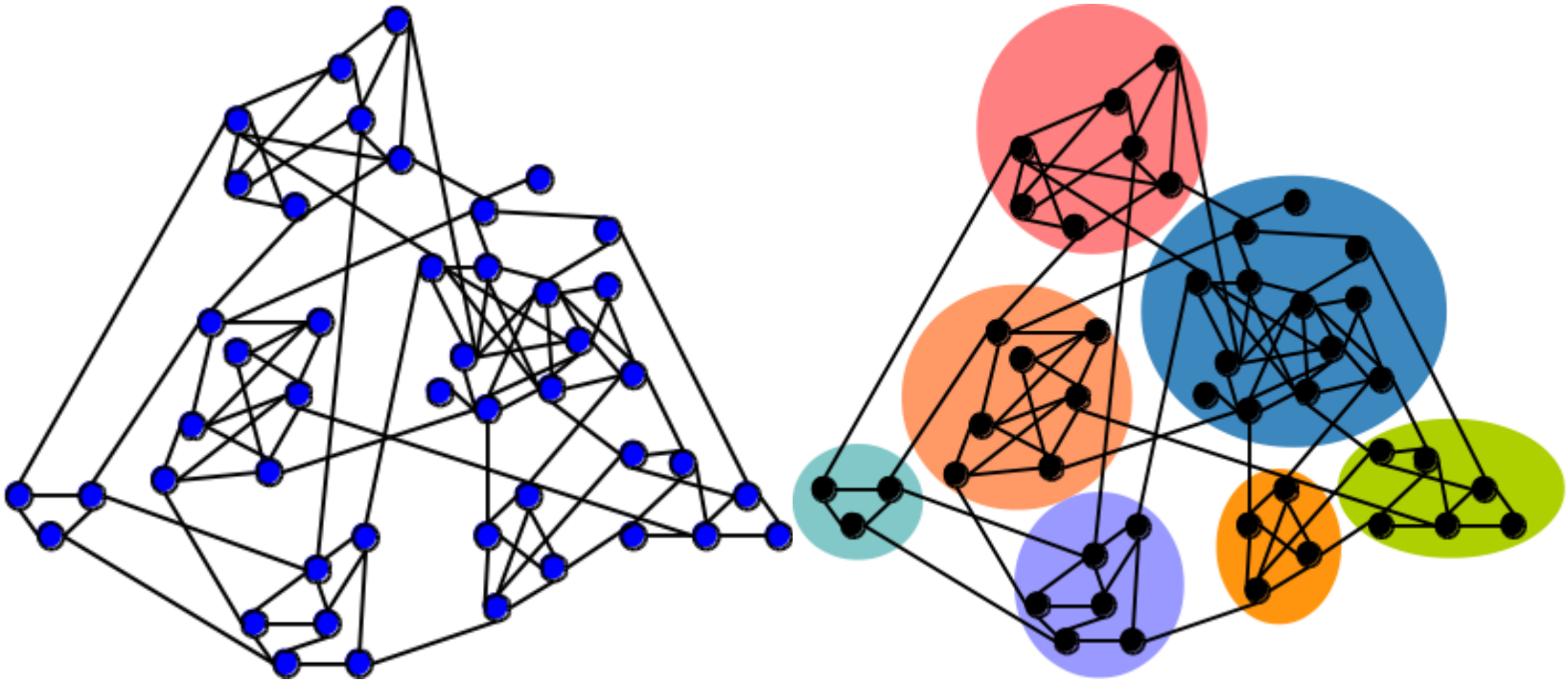
16 novembre 2011

Sommaire

1. Partitionnement de graphe
2. Modularité et modularité entière
3. Fusion-Transfert (FT)
4. Transfert-Fusion itératif (TFit)
5. Évaluation
6. Plugin pour Cytoscape

Problème : le partitionnement d'un graphe

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



Problème : le partitionnement d'un graphe

$G = (V, E)$ graphe simple, connexe, $|V| = n, |E| = m$

Déterminer une partition $P = \{C_1, ..., C_k\} \in \mathcal{P}$

- en classes connexes
- qui maximise la modularité
- à nb. de classes libre

Notations

- δ symbole de Kronecker habituel
- $P(x)$ la classe de x dans P
- $\delta_{P(x)P(y)} = 1$ si x et y sont réunis, $\delta_{P(x)P(y)} = 0$ sinon.

La modularité de Newman, 2004

- Proportion d'arêtes ayant une extrémité dans C_i et l'autre dans C_j

$$e_{ij} = \frac{|E \cap (C_i \times C_j)|}{m}$$

- Probabilité pour qu'une arête au hasard ait une extrémité dans C_i

$$a_i = e_{ii} + \frac{1}{2} \sum_{j \neq i} e_{ij}$$

- Modularité

$$M(P) = \sum_{i=1,..p} (e_{ii} - a_i^2)$$

La modularité de Newman

Modularité : qualité du partitionnement (Newman, 2004)

$$M(P) = \sum_{classes\ C_i} M(C_i)$$

$$M_{C_i} = e_{ii} - (e_{ii} + \frac{1}{2}\sum_{j \neq i} e_{ij})^2$$

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

La modularité de Newman

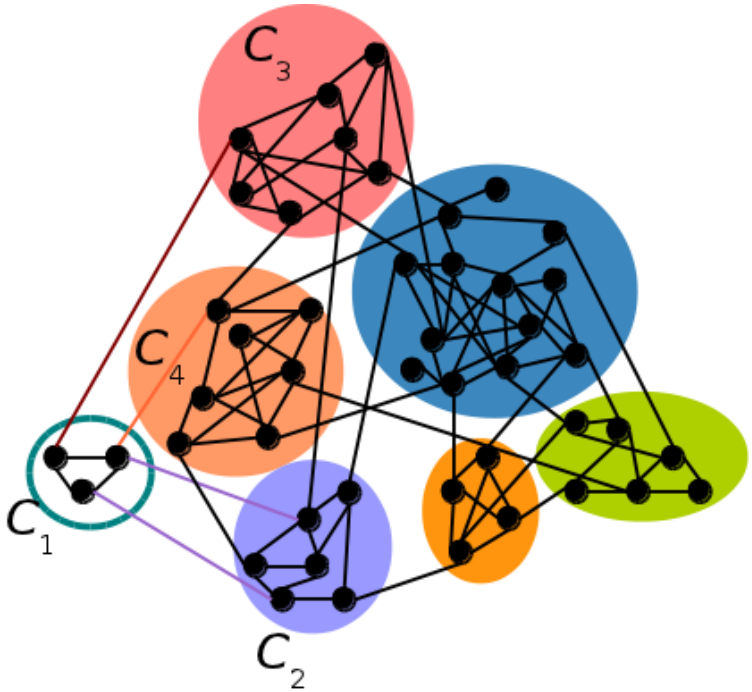
Modularité : qualité du partitionnement (Newman, 2004)

$$M_G = \sum_{classes\ C_i} M(C_i)$$

$$M_{C_i} = \underbrace{e_{ii}}_{\substack{\text{proportion d'arêtes observées dans la classe } C_i}} - \underbrace{(e_{ii} + \frac{1}{2}\sum_{j \neq i} e_{ij})^2}_{\substack{\text{proportion d'arêtes attendue dans la classe } C_i \text{ s'il n'y avait pas de communauté}}}$$

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

La modularité de Newman



Modularité : qualité du partitionnement (Newman, 2004)

$$M_G = \sum_{classes\ C_i} M(C_i)$$

$$M_{C_i} = e_{ii} - (e_{ii} + \frac{1}{2}\sum_{j \neq i} e_{ij})^2$$

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

$$M(C_1) = \frac{3}{90} - (\frac{3+\frac{1}{2}(2+1+1)}{90})^2 = 0.0302$$

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

Modularité entière (Dutch & Arenas, 2005)

- Soit d_x le degré de x dans G,
- Soit A la matrice d'incidence de G ($A_{xy} = 1$ ssi $(x, y) \in E$)

$$d_x = \sum_{(x,y)\in E} A_{xy}$$

- Soit B la matrice de terme général $B_{xy} = 2mA_{xy} - d_x d_y$

$$Q(P) = \sum_{x \neq y} B_{xy} \delta_{P(x)P(y)}$$

Proposition (Dutch & Arenas, 2005)

$$\max M(P) \equiv \max Q(P)$$

L'algorithme de Fusion-Transfert I

Partie Fusion (hiérarchie) :

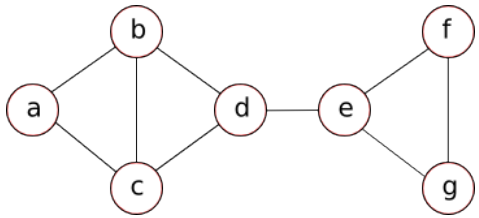
- Partir des singletons ;
  le gain de la fusion de toute paire de classes est $w(x, y)$
- Tant que W augmente
  - réunir les deux classes donnant un gain maximum
  - mettre à jour les gains de fusion de la nouvelle classe avec les autres classes

Partie Transfert (optimisation) :

- Pour chaque élément de poids non maximum dans sa classe
  - Le placer dans la classe où sa contribution est maximum, si contribution ≥ 0,
  - Sinon, en faire un singleton
  - Mettre à jour les contributions aux classes modifiées

Exemple - Fusion

Graphe G ; $m = 9$; $B_{xy} = 2mA_{xy} - d_x d_y$



Matrice A

	A	B	C	D	E	F
B	1					
C	1	1				
D	0	1	1			
E	0	0	0	1		
F	0	0	0	0	1	
G	0	0	0	0	1	1

Matrice B

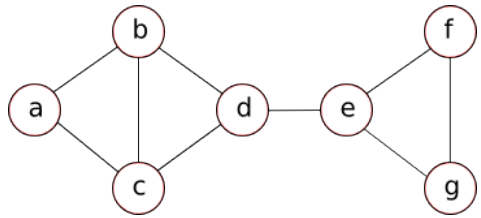
	A	B	C	D	E	F
B	12					
C	12	9				
D	-6	9	9			
E	-6	-9	-9	9		
F	-4	-6	-6	-6	12	
G	-4	-6	-6	-6	12	14

$$Q(P) = \sum_{x \neq y} B_{xy} \delta_{P(x)P(y)}$$

$\Rightarrow$ Fusion des classes $\{F\}$ et $\{G\}$ $\Rightarrow$ $Q(P) = 14$

Exemple - Fusion

Graphe G ; $m = 9$; $B_{xy} = 2mA_{xy} - d_x d_y$



Matrice A

	A	B	C	D	E	F
B	1					
C	1	1				
D	0	1	1			
E	0	0	0	1		
F	0	0	0	0	1	
G	0	0	0	0	1	1

Matrice B

	A	B	C	D	E
B	12				
C	12	9			
D	-6	9	9		
E	-6	-9	-9	9	
F,G	-8	-12	-12	-12	24

$$Q(P) = \sum_{x \neq y} B_{xy} \delta_{P(x)P(y)}$$

$$P_0 = \{A, B, C\}; P_1 = \{E, F, G\}; P_2 = \{D\}$$

$$w(P_0) = 12 + 12 + 9; w(P_1) = 14 + 12 + 12; w(P_2) = 0$$

Exemple - Fusion

Matrice A

	A	B	C	D	E	F
B	1					
C	1	1				
D	0	1	1			
E	0	0	0	1		
F	0	0	0	0	1	
G	0	0	0	0	1	1

Matrice B

	A	B	C	D	E
B	12				
C	12	9			
D	-6	9	9		
E	-6	-9	-9	9	
F,G	-8	-12	-12	-12	24

Où placer D ?

- Conserver le singleton $\Rightarrow w(\{D\}) = 0$
- Fusionner avec $P_0 \Rightarrow w(\{D\} \cup \{A, B, C\}) = -6 + 9 + 9$
- Fusionner avec $P_1 \Rightarrow w(\{D\} \cup \{E, F, G\}) = 9 - 6 - 6$

$\Rightarrow$ deux classes $\{A, B, C, D\}$ et $\{E, F, G\}$.

L'algorithme de Fusion-Transfert I

Partie Fusion (hiérarchie) :

- Partir des singletons ;
  le gain de la fusion de toute paire de classes est $w(x, y)$
- Tant que W augmente
  - réunir les deux classes donnant un gain maximum
  - mettre à jour les gains de fusion de la nouvelle classe avec les autres classes

Partie Transfert (optimisation) :

- Pour chaque élément de poids non maximum dans sa classe
  - Le placer dans la classe où sa contribution est maximum, si contribution ≥ 0,
  - Sinon, en faire un singleton
  - Mettre à jour les contributions aux classes modifiées

L'algorithme de Fusion-Transfert II

Partie Stochastique :

Répéter

- Partir de la meilleure partition courante π
- Construire une partition aléatoire π' en échangeant au hasard des éléments entre classes (nb. d'échanges tiré au hasard)
- Appliquer la partie Transfert $\pi' \rightarrow \pi$"
- Si $W(\pi") > W(\pi)$ alors $\pi \leftarrow \pi$"

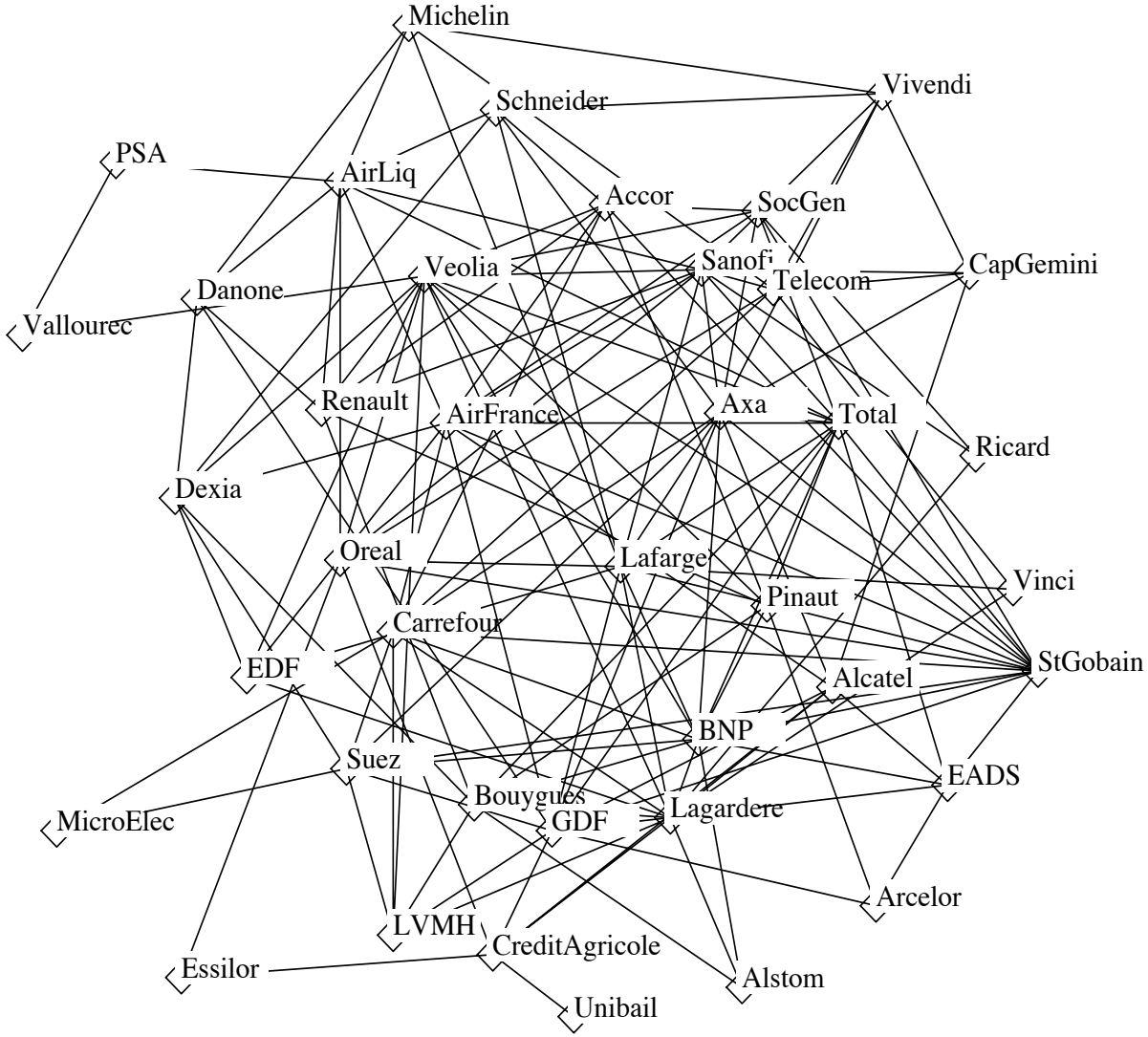
Deux paramètres

- Nb. d'essais (infructueux) consécutifs *NbEss*
- Nb. maximum d'échanges pour engendrer une partition aléatoire *SwapMax*

Complexité polynomiale : $O(n^3) + O(n^2) + \sim NbEss \times O(n^2)$

1. Partie Fusion :
   - Tant que : $O(n)$ itérations
     - Trouver les 2 classes à fusionner : $O(n^2)$
     - Mettre à jour les gains : $O(n^2)$
2. Partie Transfert :
   - Calculer les poids des éléments : $O(n^2)$
   - Tant qu'il existe ... : $O(n)$ itérations
     - Élément de poids non maximum : $O(n)$
     - L'affecter au mieux : $O(n)$
     - Mettre à jour les poids : $O(n)$
3. Partie stochastique :
   - Appliquer au moins $NbEss$ fois la partie Transfert en $O(n^2)$

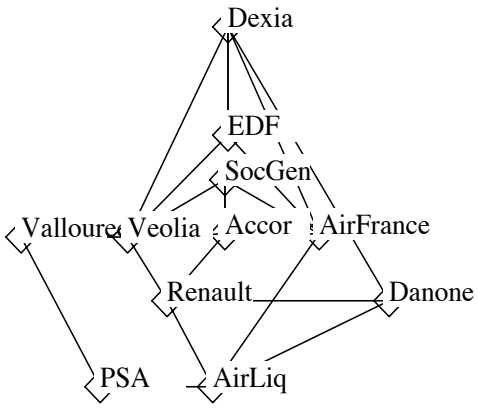
Consanguinité des Conseils d'Administration du CAC40

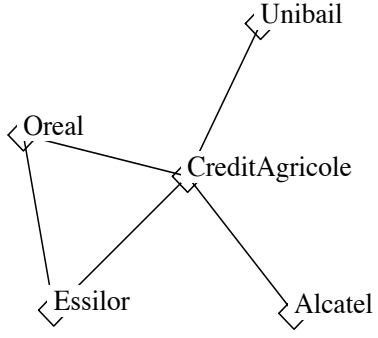


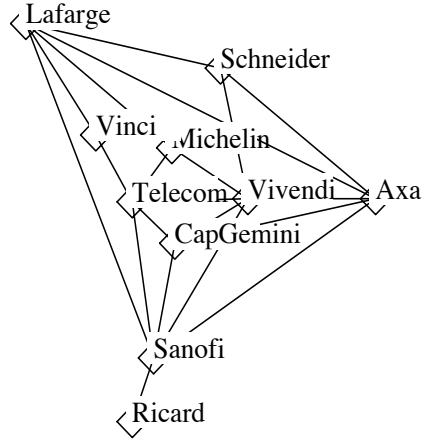
Les classes initiales du CAC 40

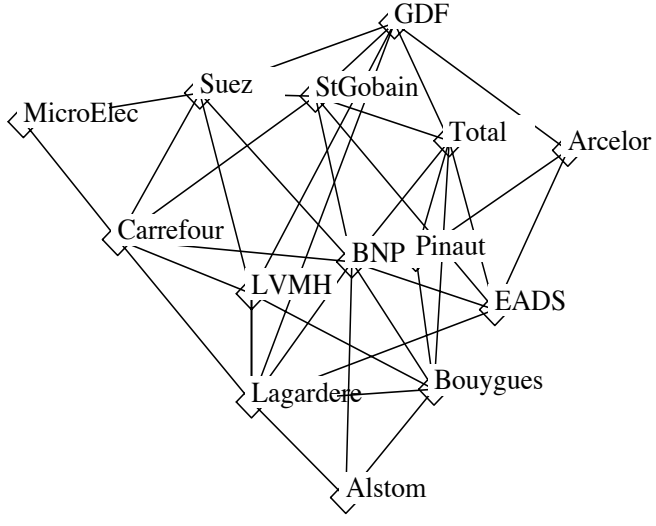
- Class 1 : Accor AirFrance AirLiq Danone Dexia EDF PSA Renault SocGen Vallourec Veolia : 0.593
- Class 2 : Axa CapGemini Lafarge Michelin Ricard Sanofi Schneider Telecom Vinci Vivendi : 0.739
- Class 3 : Alstom Arcelor BNP Bouygues Carrefour EADS GDF LVMH Lagardere MicroElec Pinaut StGobain Suez Total : 0.527
- Class 4 : Alcatel CreditAgricole Essilor Oreal Unibail : 0.717

Les classes du CAC 40 sont denses



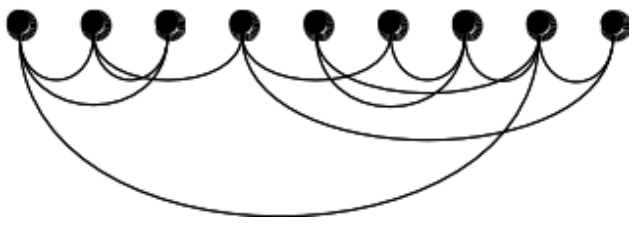






TFit - Transferts - Fusions itérés

Classes :
Sommets :

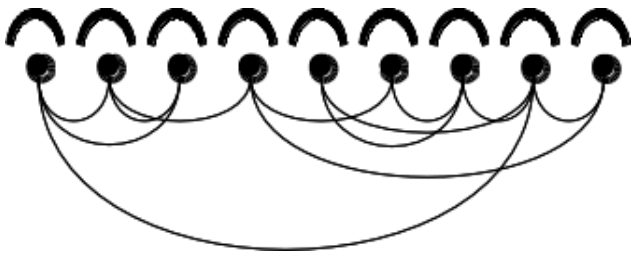


Tant qu'on améliore la modularité :

- Transfert de chaque élément vers la classe qui améliore le plus la modularité globale
- Fusions de classes qui améliorent la modularité globale

TFit - Transferts - Fusions itérés

Classes :
Sommets :

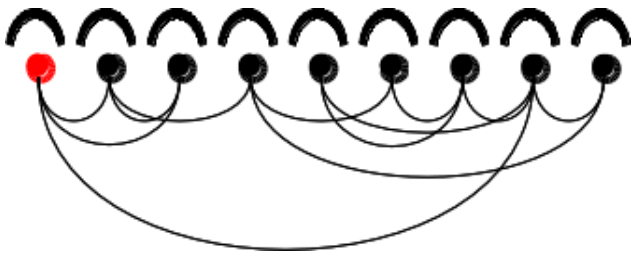


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

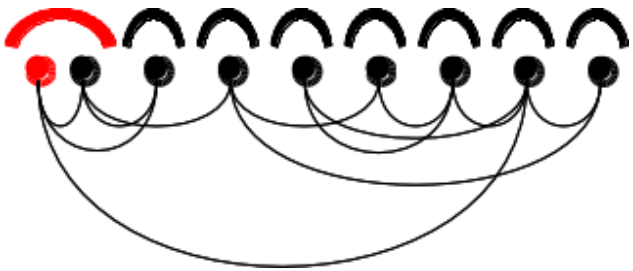


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

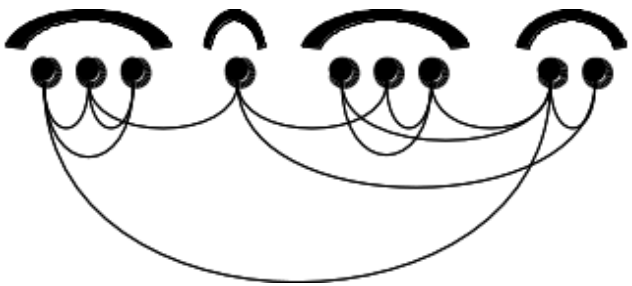


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

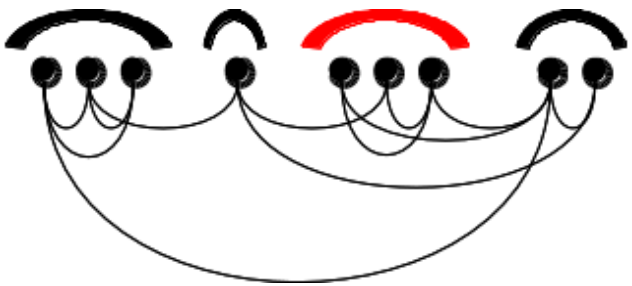


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

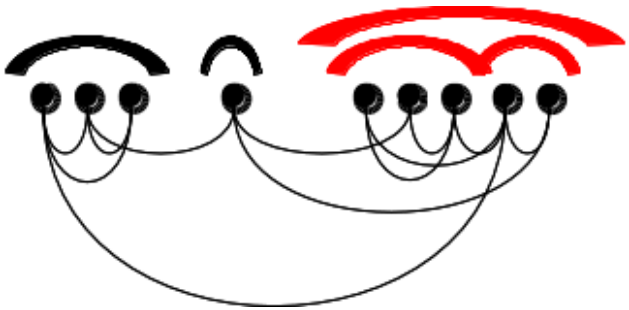


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

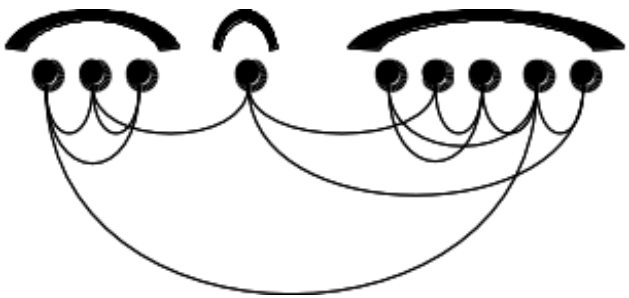


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

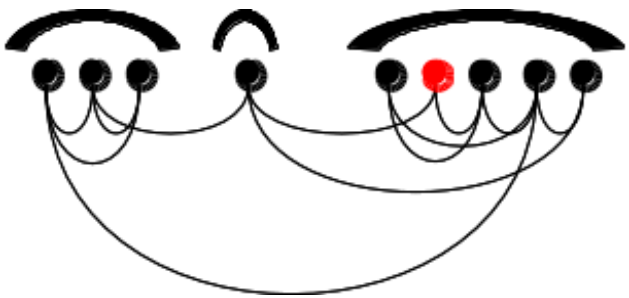


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

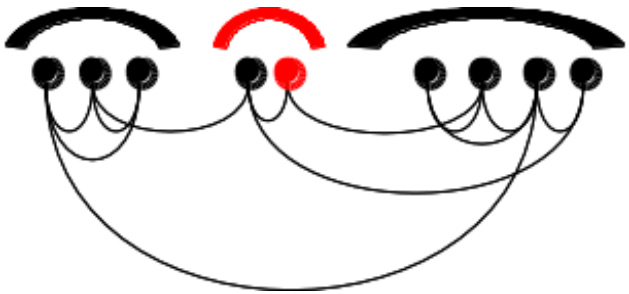


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :

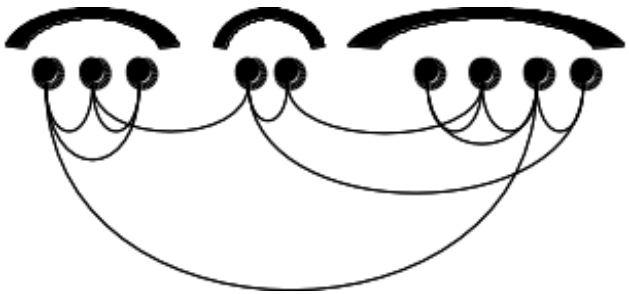


Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

TFit - Transferts - Fusions itérés

Classes :
Sommets :



Tant qu'on améliore la modularité :

- Pour chaque sommet, transfert vers la classe qui améliore le plus la modularité (T)
- Pour chaque classe, transfert vers celle qui améliore le plus la modularité (F)

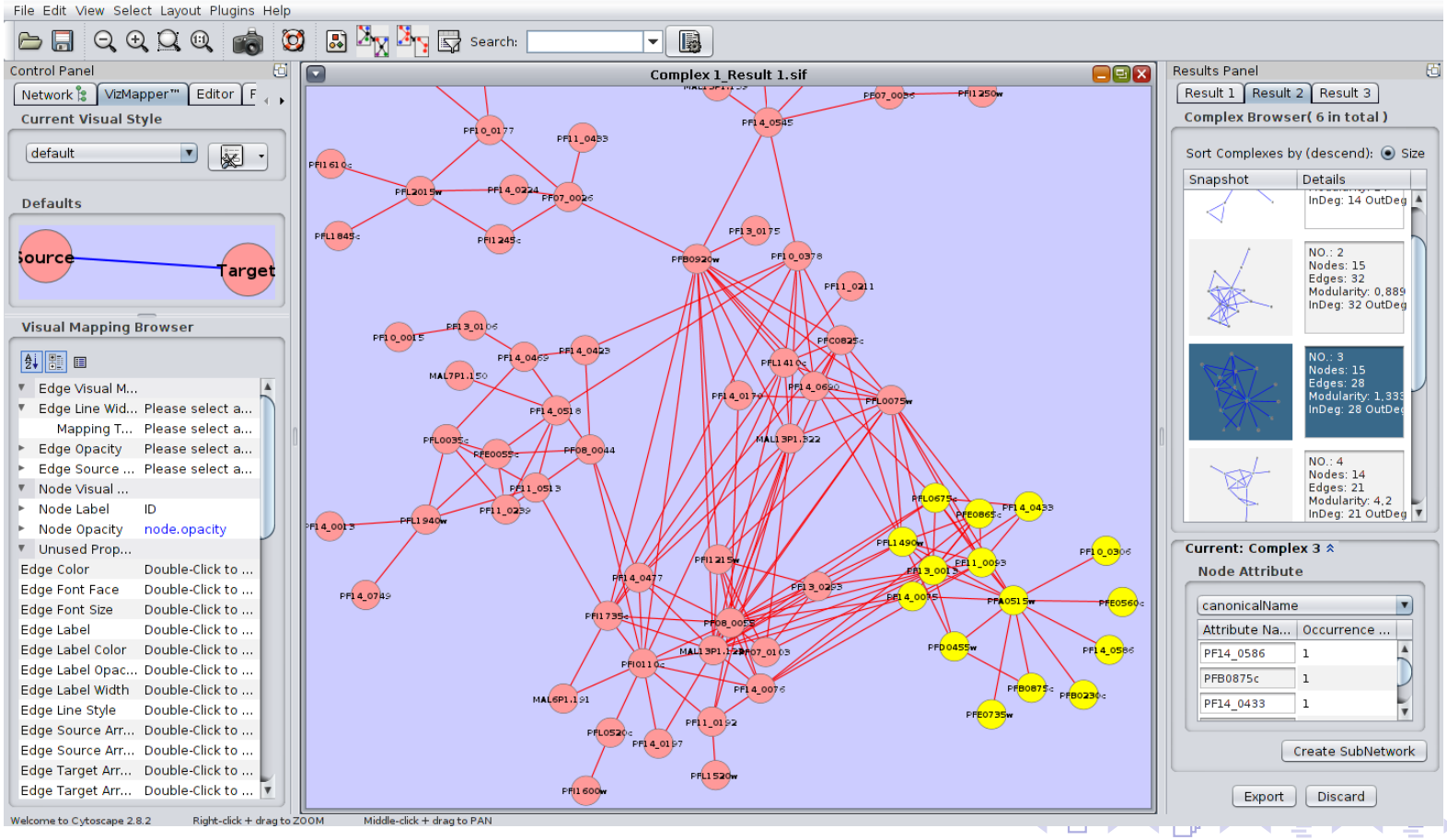
Benchmarks

graphe	n	m	Opt	Louvain	N-R	FT	TFit
Dolphins	62	159	.5285	.5185	.5276	**.5285**	.5268
polBooks	105	441	.5272	.5266	**.5272**	.5221	.5269
afootball	115	613	.6046	**.6046**	.6045	.6032	**.6046**
A01	249	635	.6329	.6145	.6293	**.6310**	.6294
USAir97	332	2126	.3682	.3541	.3678	**.3682**	.3612
netscience	379	914	.8486	.8475	.8474	.8474	**.8477**
s388	512	819	.8194	.7962	.8143	.8122	**.8154**
emails	1133	5452		.5438	**.5816**	.5556	.5747

Comparaison de *FT* et *TFit* avec d'autres méthodes (Louvain 2008, Noack & Rotta 2009) sur une suite de graphes. Partitions optimales (Aloise *et al.* 2010).

Application : ModClust plugin

- Implémentation des deux algorithmes dans un plugin Java pour Cytoscape.
- Application à l'analyse de réseaux d'interactions protéines-protéines.



Application : ModClust plugin

Fonctionalités :

- Coloration de chaque classe
- Mise en évidence des noeuds connectés à d'autres classes
- Mise en évidence des arêtes inter-classes
- Sélection d'un ensemble de classes pour construire un nouveau graphe
- Partitionnement itératif

Application

- Code C et plugin Java téléchargeables sur `http://bioinformatics.lif.univ-mrs.fr`
- Remerciements à Anaïs Baudot (IML) et Christine Brun (TAGC, INSERM).
- Travaux financés par l'ANR Moonlight (prédiction de protéines multifonctionnelles).

Merci