Une adaptation de l'algorithme de Müllner pour la détection de communautés dans des réseaux complexes

Brieuc Conan-Guez et Manh Cuong Nguyen

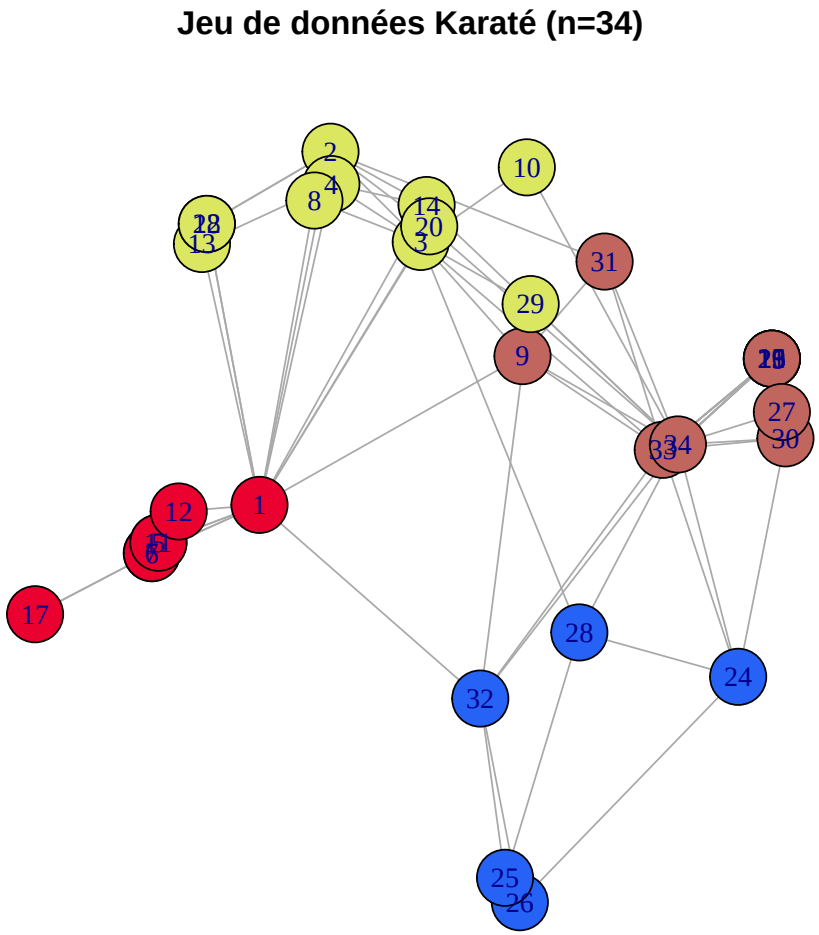
Laboratoire d'Informatique Théorique et Appliquée (LITA)

25 Juin 2013

Graphes et Modules (communautés)

- Graphe non orienté
- Module
  - Sous-ensemble de sommets du graphe
  - Connexions denses au sein du module (connexions intra)
  - Connexions faibles entre modules (connexions inter)
- Aussi appelé "communauté" pour les réseaux sociaux
- On note n le nombre de sommets et m le nombre d'arêtes

Détection des communautés : jeu de données Karaté



Le critère de modularité

- But : mesurer la qualité du partitionnement d'un graphe
- Newman (2004) propose le critère de modularité, noté Q.
- Q : différence entre la proportion d'arêtes intra et la proportion d'arêtes intra attendue pour un graphe aléatoire respectant les degrés.

$$Q = \sum_i (e_{ii} - a_i^2)$$

e_{ij} : proportion d'arêtes qui joignent C_i à C_j
a_i : proportion d'extrémités qui appartiennent à C_i

Modularité : propriétés

- Q à valeurs dans $[-1, 1]$
- Des valeurs élevées de Q indiquent une structure forte de communautés
- On cherche donc à maximiser Q

- Q permet déterminer le nombre optimal de communautés

- Seule la fusion de deux communautés **connectées** permet d'améliorer Q.

Méthodes hierachiques utilisant le critère de modularité

- **Méthodes divisives** :

  - Extremal Optimization (Duch et Arenas - 2005)
  - Spectral Graph Bisection (Newman - 2006)

- **Méthodes agglomératives** :

  - Méthode de Louvain (Blondel et al - 2008)

  - CAH classique (plusieurs algorithmes) :
    . CNM (Clauset, Newman et Moore - 2004)
    . WT (Wakita et Tsurumi - 2007)
    . FO (Francisco et Oliveira - 2011).

Significativité : un nouveau critère d'agrégation

- ΔQ_{ij} : variation de Q après fusion de C_i et C_j
- **Significativité** : nouveau critère d'agrégation pour la CAH

$$S_{ij} = \frac{\Delta Q_{ij}}{\sqrt{d(C_i)d(C_j)}}.$$

$d(C_i)$: le degré de la communauté C_i

- Partitions avec le critère S_{ij} meilleures que celles avec ΔQ_{ij}
- Algorithmes beaucoup plus rapides

Principe de la CAH pour la modularité

Entrée : Matrice creuse d'agrégation ΔQ (ou la significativité S)

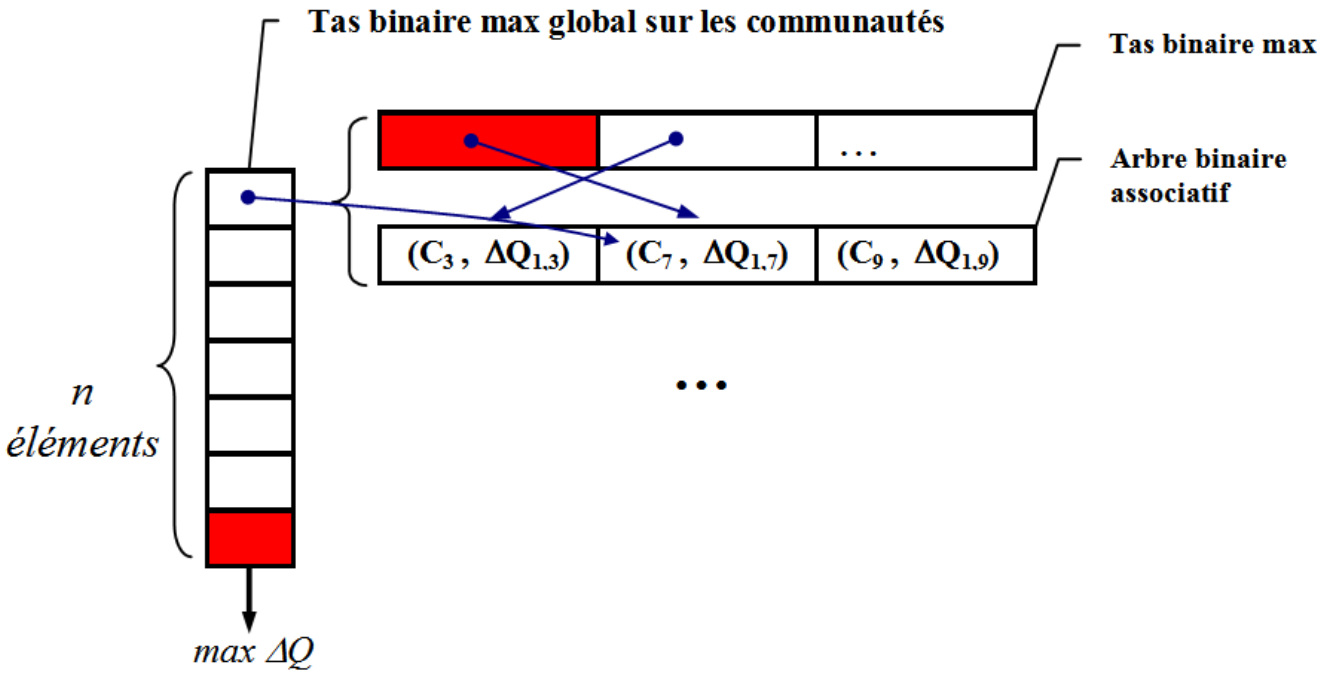
1. Sélectionner le plus grand ΔQ_{ij}

2. Fusionner les communautés correspondantes C_i et C_j

3. Mettre à jour la matrice d'agrégation

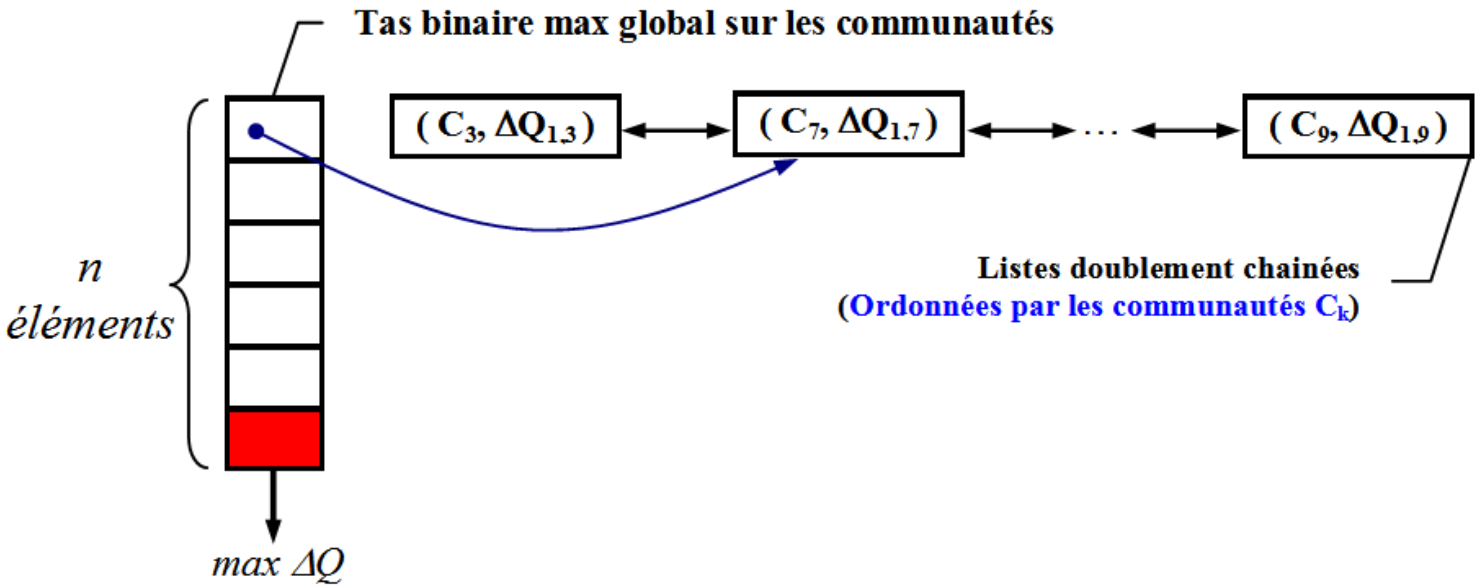
Sortie : "dendrogramme" - une liste de triplets $(C_i, C_j, \Delta Q_{ij})$.

CNM (Clauset, Newman et Moore - 2004)



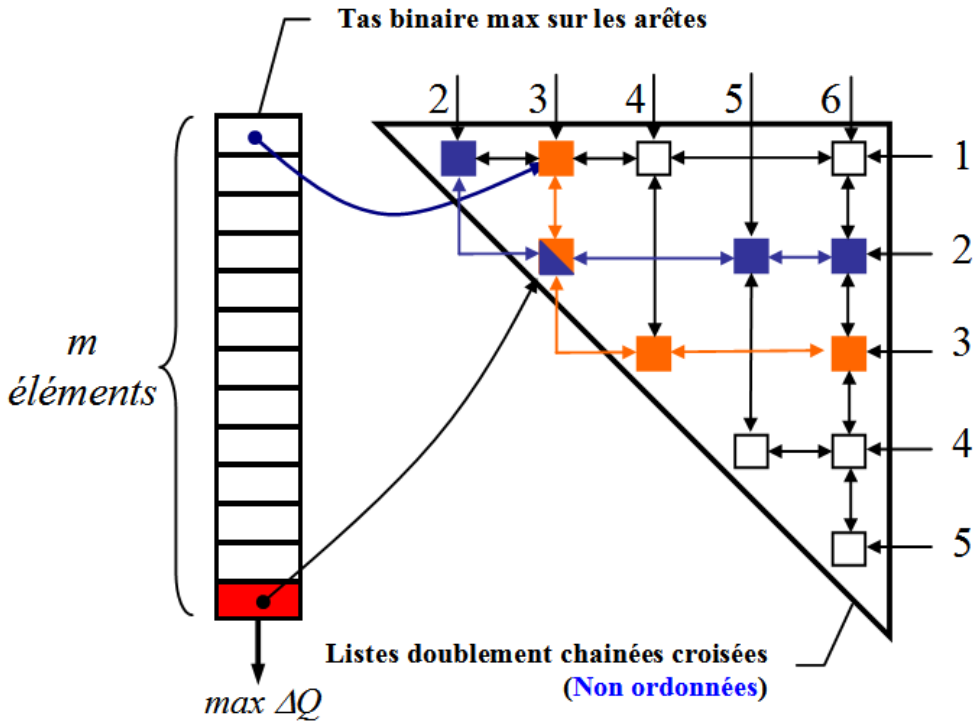
Complexité de CNM : $O(n\ ln^2(n))$.

WT (Wakita, Tsurumi - 2009)



Complexité de WT : $O(n\ ln^2(n))$.

FO (Francisco et Oliveira - 2011)



Complexité de FO : $O(n\ ln^2(n))$.

Algorithme de Müllner (2011)

Müllner : algorithme de CAH pour des matrices de dissimilarités

Notations :

- T : la triangulaire supérieure de la matrice d'agrégation (T_{ij})
- $T_{i.}$: une "demi-ligne" de T (i.e. T_{ij} avec $i < j$)

Idée principale :

- maintenir une **borne inf** sur chaque demi-ligne $T_{i.}$
- maintenir un **candidat potentiel** sur chaque demi-ligne $T_{i.}$

  - cluster qui a été le plus proche du cluster i au sens de T_{ij}
  - mais il peut avoir perdu ce statut à cause des remises à jour

- utilisation d'un tas binaire global min sur les bornes inf

Algorithme de Müllner

Pourquoi Müllner est efficace ?

- grâce à la borne inf, la recherche du meilleur cluster dans une demi-ligne donnée peut souvent être évitée

- de même, les mises à jour du tas global min sont moins fréquentes.

Complexité :

- $O(n^3)$ (Il existe un algorithme en $O(n^2\ ln(n))$)

- dans la pratique beaucoup plus rapide

- implémentation : package fastcluster du logiciel R

Algorithme de Müllner

Initialisation :

- tableau des bornes inf et des candidats potentiels
- tas binaire min des bornes inf

Répéter :

1. extraire borne inf min du tas min (clusters C_i, C_j considérés)
2. si borne inf différente du T_{ij} du candidat potentiel
   - recalculer exactement le candidat et la borne inf
   - réinsérer dans le tas et retourner à l'étape 1
3. fusionner $C_j \leftarrow C_i \cup C_j$, mettre à jour la matrice d'agrégation
4. mettre à jour les bornes inf et les candidats potentiels

Mod-Müllner : adaptation de Müllner au critère Q

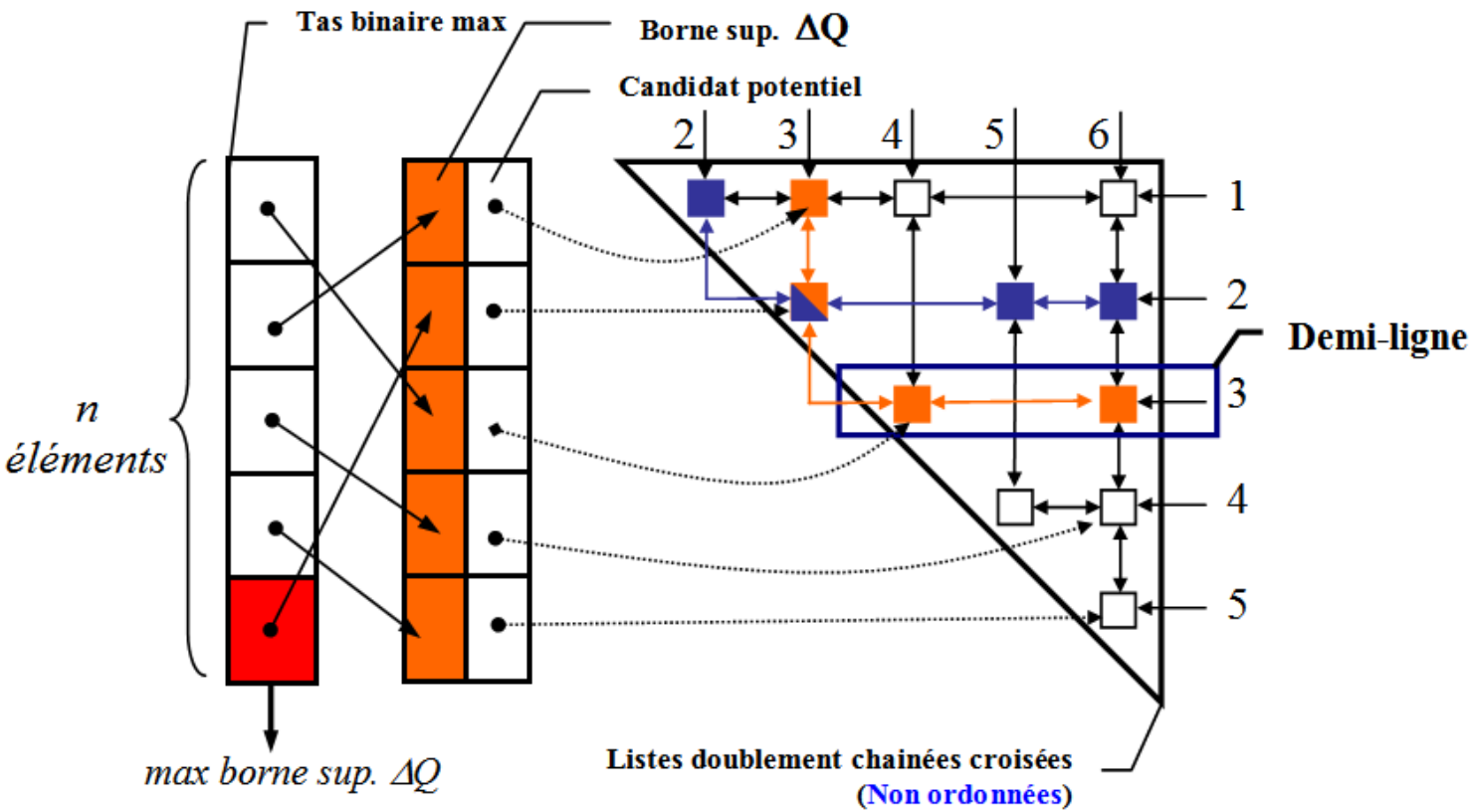
Adaptations :

- maximiser Q : borne sup sur la demi-ligne $\Delta Q_{i.}$ et tas max
- ΔQ croit seulement pour les communautés connectées
  - parcourir ou mettre à jour seulement les communautés voisines
- Nécessité d'une structure de données adaptée
  - adaptation mineure de la structure de FO
  - fusion de 2 listes non ordonnées par une indexation préalable

Complexité :

- $O(n^2\ ln(n))$ (CNM, WT, FO en $O(n\ ln^2(n))$)
- dans la pratique, souvent plus rapide

Mod-Müllner : schéma



MyFO : amélioration de l'implémentation de FO

- Notre nouvelle implémentation de FO : MyFO
- MyFO utilise en partie le même code que Mod-Müllner
  - Même implémentation pour la matrice d'agrégation,
  - Implémentation semblable pour le tas binaire
- L'implémentation MyFO est plus rapide que FO
- Or FO est au moins deux fois plus rapide que CNM

Protocole expérimental

- **Environnement** : C^{++} 64 bits.
- **Ordinateur** : Core i5 2.8 Ghz, 8 Gb RAM.
- Les temps d'exécution sont calculés une fois les données chargées dans une structure de données temporaire
- La significativité est utilisée comme critère d'agrégation pour tous les algorithmes

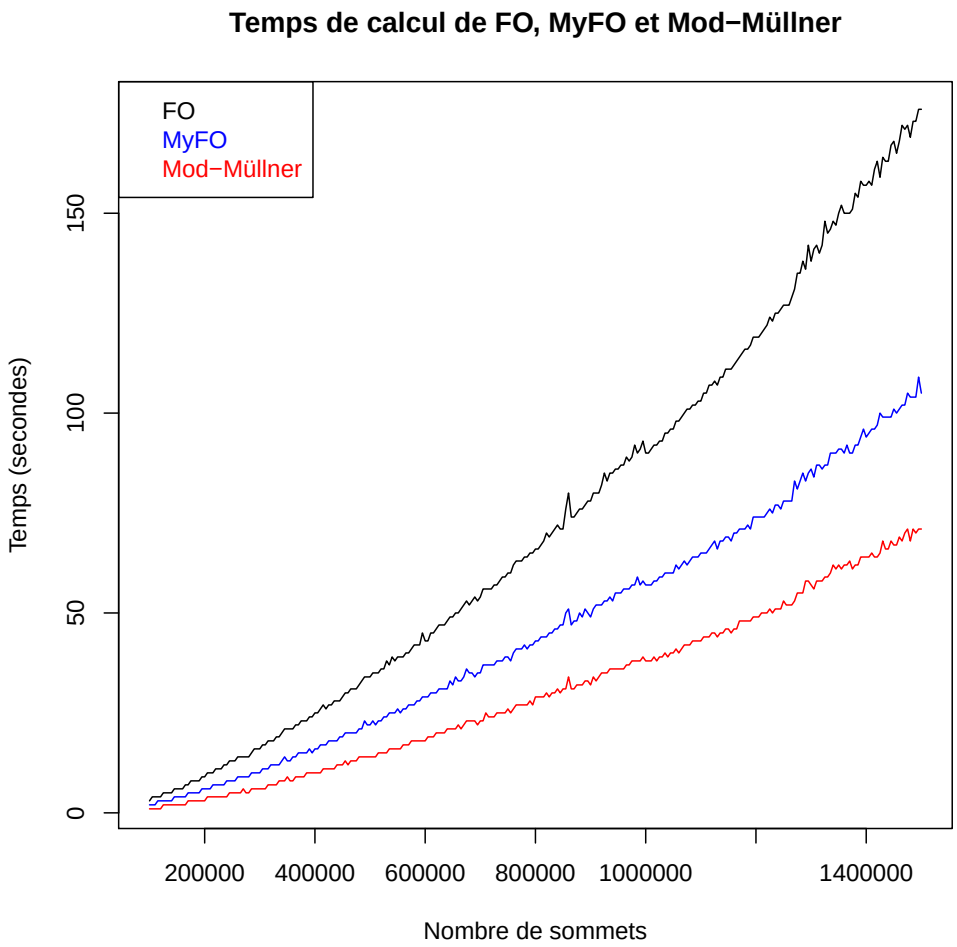
Expériences sur des données simulées

On génère plusieurs graphes selon le modèle suivant :

- nombre de communautés : 100
- probabilité d'une arête intra : p
- probabilité d'une arête inter : q
- on fixe le rapport $f = \frac{q}{p} = 0.003$
- degré moyen $= 15$
- modularité théorique $= 0.76$
- **Tailles des données** : de 100.000 à 1.500.000 sommets, jusqu'à 11 millions d'arêtes

- **Pour éviter les ex-aequo** : génération aléatoire du poids des arêtes dans l'intervalle [0.9,1.1].

Expériences sur des données simulées



Expériences sur des données réelles

Nom	Sommet	Arête	Mod-Müllner	MyFO	$\frac{MyFO}{M-M}$	FO	$\frac{FO}{M-M}$
AMA	410 236	2 439 437	1.8 (0.854)	3.2 (0.854)	1.7	6.0 (0.855)	3.2
DEL	1 048 576	3 145 686	2.5 (0.977)	4.0 (0.977)	1.6	5.5 (0.977)	2.2
WBS	685 230	6 649 470	11.1 (0.932)	10.2 (0.932)	0.9	19.6 (0.933)	1.7
HUG	4 588 484	6 879 133	7.7 (0.989)	10.0 (0.989)	1.3	14.3 (0.988)	1.8
M6	3 501 776	10 501 936	13.3 (0.982)	19.0 (0.983)	1.4	27.0 (0.983)	2.0
3SP	3 712 815	11 108 633	12.4 (0.987)	18.6 (0.986)	1.5	25.1 (0.987)	2.0
ADA	6 815 744	13 624 320	11.1 (0.987)	16.9 (0.987)	1.5	24.5 (0.987)	2.2
KRO	731 706	15 888 007	997.0 (0.051)	405.3 (0.051)	0.4	865.8 (0.051)	0.9
RCE	14 081 816	16 933 413	42.3 (0.997)	46.6 (0.997)	1.1	54.4 (0.997)	1.3
SPR	1 632 803	22 301 964	113.8 (0.603)	139.6 (0.603)	1.2	254.2 (0.602)	2.2
AFS	1 508 065	25 582 130	4.6 (0.965)	10.1 (0.966)	2.2	10.0 (0.965)	2.2
RUS	23 947 347	28 854 312	65.8 (0.998)	73.1 (0.998)	1.1	85.4 (0.998)	1.3
HOK	1 498 023	29 709 711	8.6 (0.919)	21.5 (0.920)	2.5	45.6 (0.920)	5.3
WIK	3 515 067	42 375 912	172.6 (0.489)	227.1 (0.488)	1.3	459.1 (0.491)	2.7
D24	16 777 216	50 331 601	56.5 (0.990)	99.3 (0.990)	1.8	135.1 (0.990)	2.4

- temps en secondes (modularité)
- KRO n'a pas de structure de communautés (modularité nulle)
- gain : près d'une minute pour WIK et 30 secondes pour D24

Conclusions

- **Mod-Müllner fournit les mêmes résultats que CNM (à ex aequo prés)**
- **Implémentation relativement simple**
- **Bonnes performances pour les temps de calcul pour des graphes de plusieurs millions d'arêtes**

Merci pour votre attention.