Grands Réseaux d'Interaction
TP n° 7 : Modularité et communautés

Règles Générales

- Les règles générales en TP restent valides, voir feuille du TP 1
- Ce TP est à rendre sur Moodle pour le **23 novembre**

I) Description de la séance

a) But du TP

Le but de ce TP est d'implémenter :

1. un calcul de la modularité,
2. un algorithme naïf pour trouver des communautés en maximisant de la modularité.

b) Travail à effectuer

1. Dans un premier temps, implémenter un calcul de modularité et le tester sur les exemples fournis (voir Moodle)
2. Dans un second temps, implémenter l'algorithme de maximisation de modularité. Vous pouvez le tester sur des exemples donnés sur Moodle, et sur des exemples plus imposants que vous pourrez trouver sur : `http://konect.uni-koblenz.de/networks/` ou `https://snap.stanford.edu/data/` (attention, très peu sont au format .dot)

c) Rendu attendu

- Format de fichier graphe : `.dot` ou liste d'arêtes (1 paire de numéros de sommets par ligne, séparés pas un espace)
- Format de fichier communautés : une communauté par ligne, chaque ligne comportant les sommets de la communauté séparés par des espaces.
  Exemple pour les communautés de la Figure 1 (communautés verte et bleue) :
  ```
  1 2 4 8 12 13 18 20 22
  5 7 6 11 17
  [...]
  ```

1. Pour la modularité, un programme qui étant donné en argument un fichier graphe et un découpage en communautés, écrit sur la sortie standard la valeur de la modularité.
   Usage (exemple) :
   ```
   $ ./modularite [graphe] [communautes]
   ```
   Sortie (exemple) :
   ```
   > 0.345
   ```
2. Pour l'algorithme, un programme qui, étant en argument un fichier graphe, écrit sur la sortie standard la modularité maximale Q_{max} obtenue pour la partition de communautés C_{max}, puis cette partition C_{max} dans le même format que le fichier de communautés décrit précédemment.

Usage (exemple, les valeurs ne correspondent à rien) :
`./algo_newman [graphe]`
Sortie (exemple) :
`> 0.6498` (modularité)
`1 2 4 3 4 5 6` (partition en communautés)
`7 8 9 10`
`[...]`
Ainsi qu'une option permettant d'avoir les modularités obtenues à chaque itération de l'algorithme, usage (exemple) :
`$ ./algo -v [graphe]`
Sortie (exemple) :
`> 0.1`
`0.2`
`0.3`
`0.4`
`0.45`
`0.35`
`0.25`

II) Modularité (généralités)

La plupart (mais pas tous !) les grands réseaux d'interaction ont une structure *modulaire*, c'est-à-dire que leur graphe est composé de sous-graphes localement denses. Les réseaux sociaux, biologiques sont souvent très modulaires, alors que les réseaux de transport urbain (métro par exemple), ne le sont que peu voire pas du tout.
Ces sous-graphes localement denses sont des ensembles de sommets dont la densité interne (liens entre eux) est très supérieure à la densité externe (liens avec le reste du graphe). Dans la littérature, ces ensembles sont appelés *clusters*, ou *communautés*, et nombre d'algorithmes existent pour tenter de les retrouver, ce qui n'est pas une tâche aisée car plusieurs découpages en clusters/communautés peuvent être sémantiquement valables pour un même graphe.
Au départ, ces découpages étaient obtenus au moyen de mesures existantes, telles que les centralités (voir par exemple l'algorithme de Girvan et Newman (2002) utilisant la *betweenness*), mais ces méthodes étaient souvent lentes et parfois peu pertinentes. C'est pourquoi Newman et Girvan (2004) ont introduit un critère de qualité d'un découpage de graphe en communautés nommé *modularité*[2] .
La modularité a immédiatement suscité un fort engouement dans le domaine de l'analyse des réseaux, ainsi que dans les domaines faisant usage de l'analyse des réseaux tels que la biologie moléculaire, et de très nombreux travaux l'ont exploitée depuis une douzaine d'années. C'est une mesure incontournable pour exploiter des réseaux, bien qu'elle souffre de plusieurs inconvénients telles que la difficulté de la maximiser efficacement qui est une tâche NP-complète dans la plupart des cas (Brandes et al., 2007) et la limite de résolution (Fortunato & Barthélemy, 2007) : la modularité ne capture pas, ou mal, les plus petites communautés. Notamment, lorsqu'on maximise itérativement la modularité, ce qui permet d'obtenir une structure hiérarchique (voir le schéma Figure 1), la modularité favorise les regroupements et donc les communautés d'une certaine taille.

[1] C'est le club de karaté de Zachary, exemple très connu dans la littérature (Zachary, 1977)
[2] Ne pas confondre la propriété d'un graphe à être modulaire et le critère de Newman & Girvan, répondant tous les deux au nom de modularité. Dans ce sujet de TP, *modularité* désigne uniquement le critère.

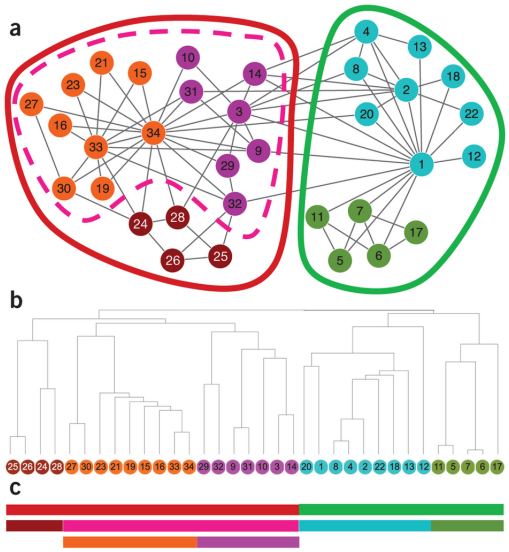

FIGURE 1 – Réseau sous forme de graphe[1] (a) et structure modulaire hiérarchique, également sous forme de dendrogramme (b) avec le code couleur des ensembles (c). Au plus haut niveau, il y a deux communautés (verte et rouge), au plus bas il y en a 5 (couleurs des sommets) Crédit : `http://www.nature.com/nmeth/journal/v10/n7/full/nmeth.2517.html`

III) Modularité (détails)

Concrètement, la modularité, calculée pour une partition en communautés C d'un graphe G, représente la fraction des arêtes de G se trouvant "à l'intérieur" de chacune des communautés $c \in C$ et "corrigée par rapport au hasard", c'est-à-dire qu'on retranche la probabilité que ces arêtes soient naturellement présentes dans un modèle de référence (*null model*[3] en anglais). Par défaut ce modèle est un graphe aléatoire comportant le même nombre d'arêtes et de sommets que le graphe G (Newman & Girvan, 2004).
En d'autres termes, on mesure à quel point la probabilité de la répartition des arêtes s'éloigne du modèle de référence pour atteindre une structure modulaire (forte densité dans les communautés, faible entre elles). Plus c'est le cas, plus la modularité a une valeur haute. Elle est bornée entre -1 et 1.
Soient un graphe non-orienté $G = (V, E)$ et un ensemble de communautés C défini comme une partition de $V, C \subset \mathcal{P}(V)$. On fait l'hypothèse que les communautés sont disjointes, c'est-à-dire : $\forall c, c' \in C, c \neq c', c \cap c' = \emptyset$, et que tout sommet appartient à une (seule) communauté : $\forall v \in V, \exists!\ c \in C\ /\ v \in c$.
L'écriture la plus simple de la modularité de la partition C du graphe G, notée $Q(C, G)$, est :

$$Q(C, G) = \sum_{c \in C} (e_{cc} - a_c^2)$$

où :

- $e_{cc'}$ est la fraction des arêtes ayant une extrémité dans la communauté c et l'autre dans $c' \in C$. Le e_{cc} utilisé dans la formule est la fraction des arêtes reliant deux sommets de la même communauté c.

$$e_{cc'} = \frac{|\{(u, v) : (u, v) \in E, u \in c, v \in c'\}|}{|E|}$$

[3] `https://en.wikipedia.org/wiki/Null_model`

– a_c est la fraction des arêtes qui ont au moins une extrémité dans la communauté c :

$$a_c = \frac{|\{(u,v) : (u,v) \in E, u \in c\}|}{|E|} = \sum_{c' \in C} e_{cc'}$$

Une définition plus "développée" mais équivalente est la suivante :

$$Q(C,G) = \frac{1}{2|E|} \sum_{u \in V} \sum_{v \in V} \left[A_{uv} - \frac{d_u d_v}{2|E|} \right] \delta(c_u, c_v)$$

où :

– A_{uv} est la valeur de la matrice d'adjacence pour (u,v), autrement dit 1 si existence de l'arête (u,v) et 0 sinon, autrement écrit : $|\{(u,v) \in E\}|$.
– d_u est le degré du sommet u, c_u est la communauté à laquelle u appartient.
– δ est le delta de Kronecker [4] , qui vaut 1 si ses deux arguments sont égaux, et 0 sinon. Dans notre cas, la valeur calculée n'entrera dans le calcul de la somme que si deux communautés identiques sont concernées : un moyen de gagner facilement du temps d'exécution !

Algorithm 1 Structure communautaire de modularité max, algorithme glouton (Newman, 2004)

Require: $G = (V, E)$, non orienté
Ensure: $C \subset \mathcal{P}(V)$
1: $C \leftarrow \{\{v_1\}, \{v_2\}, ..., \{v_{|V|}\}\}$
2: $Q_{max} \leftarrow -1$
3: $C_{max} \leftarrow C$
4: **for** $i = 0; i < |V| - 1; i{+}{+}$ **do**
5: Trouver $\hat{c}, \hat{c}' \in C, \hat{c} \neq \hat{c}'$ telles que :
$(\hat{c}, \hat{c}') = \text{argmax}_{c,c' \in C} \Delta Q(c, c', G)$
(voir section IV.a)
6: Fusionner $\hat{c}$ et $\hat{c}'$, la structure résultante est appelée C_i
7: Calculer la valeur de modularité $Q_i = Q(C_i, G)$
8: **if** $Q_i > Q_{max}$ **then**
9: $Q_{max} = Q_i$
10: $C_{max} = C_i$
11: **end if**
12: **end for**
13: **return** C_{max}
(la structure dont la modularité est maximale)

IV) Détection naïve de communautés par maximisation de modularité

En même temps que son introduction de la modularité, Newman (2004) a proposé un algorithme glouton pour trouver la structure communautaire atteignant la modularité maximum dans un graphe. Un pseudo-code est donné dans le listing ci-dessus.
Le principe est de regrouper petit à petit les sommets du graphe en maximisant les gains de modularités. La partition obtenue ayant la modularité maximum sera le résultat de l'algorithme. Au départ, chaque sommet forme sa propre communauté, puis on fusionne itérativement chaque paire de communauté qui maximise le gain de modularité.

[4] `https://fr.wikipedia.org/wiki/Delta_de_Kronecker`

a) Gain de modularité

Le gain de modularité $\Delta Q(c, c', C, G)$ est une valeur (qui peut être négative), décrivant la variation de la modularité de la partition C du graphe G après fusion des communautés $c, c' \in C$. Elle doit bien évidemment être calculée *avant* la fusion effective de c et c', pour évaluer si cette fusion est la plus intéressante ou non.
Voici comment l'implémenter pour l'algorithme. Si le gain est identique pour deux paires distinctes de communautés (ou plus) on en garde une seule "au hasard".

1. Tout d'abord, nous vous proposons une implémentation simple où l'on recalcule entièrement la modularité :

$$\Delta Q(c, c', C_i, G) = Q(C_i, G) - Q(C_{i-1}, G)$$

2. *(Optionnel)* Newman (2004) propose une expression du gain de modularité ΔQ évitant trop de re-calcul. Nous vous proposons de l'implémenter afin d'augmenter les performances de votre algorithm. Pour deux communautés c, c' dans C il est écrit comme :

$$\Delta Q(c, c', C_i, G) = e_{cc'} + e_{c'c} - 2a_c a_{c'} = 2(e_{cc'} - a_c a_{c'})$$

b) Union-Find *(optionnel)*

Utilisez Union-Find (vu en M1) pour construire la partition de structure communautaire C_i après fusion.

Références

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z. & Wagner, D. (2007). On finding graph clusterings with maximum modularity. *Proc. Intl. Workshop on Graph-Theoretic Concepts in Computer Science* (pp. 121–132). Springer.

Fortunato, S. & Barthélemy, M. (2007). Resolution limit in community detection. *Proc. of the National Academy of Sciences*, *104*, 36–41.

Girvan, M. & Newman, M. E. J. (2002). Community structure in social and biological networks. *PNAS*, *99*, 7821–7826.

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, *69*, 066133.

Newman, M. E. J. & Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E*, *69*, 026113.

Zachary, W. (1977). An information flow model for conflict and fission in small groups. *J. Anthropol. Res.*, *33*, 452–473.