Encadrant :
MAI Xiaoyi

MEJEAN Alexandre
MAISON Jonas

CentraleSupélec Campus de Gif, Année 2016-2017

Rapport projet de conception N°SS403 :

Méthodes spectrales pour la détection de groupes d'affinités dans les grands réseaux

GitHub : https://github.com/Jonas1312/CommunityDetection

Sommaire

Introduction

Les graphes permettent de décrire une multitude de systèmes complexes issus de divers domaines : réseaux sociaux, moteurs de recherche, réseaux biologiques, circuits électroniques, jeux en réseaux et encore bien d'autres applications. On modélise en général ces graphes par un ensemble de nœuds qui représentent les entités du réseau ainsi que par des arêtes qui représentent les relations entre ces entités. L'existence de communautés dans un réseau correspond à la présence de certains groupes de nœuds qui sont très fortement connectés entre eux par rapport aux autres nœuds du graphe. On appelle partitionnement de graphe la détection de communautés non chevauchantes.

Il existe actuellement dans la littérature une multitude de travaux portant sur la détection de communautés dans les graphes et ce domaine d'étude est en pleine expansion depuis la création des premiers réseaux sociaux. La plupart de ces méthodes consistent à déterminer une partition des sommets du graphe optimisant un certain critère de qualité d'un partitionnement. Ce problème d'optimisation est NP-complet, et il est donc difficile d'obtenir des résultats en un temps raisonnable sur des graphes de plusieurs milliers de nœuds.

Dans le cadre de notre projet nous avons abordé les algorithmes spectraux, qui sont aussi très utilisés pour la détection de communauté. Le partitionnement spectral utilise les vecteurs et les valeurs propres d'une matrice d'affinité définie à partir d'un graphe et réalise ensuite de la classification non-supervisée des vecteurs propres dominants pour trouver des partitions.

Nous aborderons donc dans un premier temps la génération de graphes aléatoires à l'aide du « stochastic block model », puis nous présenterons les diverses matrices d'affinités que nous avons utilisé et plus particulièrement la matrice appelée Bethe Hessian, et qui a la particularité de donner de meilleurs résultats sur les graphes sparses. Enfin, nous étudierons les performances de nos algorithmes et comparerons les résultats des diverses matrices d'affinités.

Génération de graphes aléatoires

Le « stochastic block model » est un modèle de génération de graphes aléatoires très connu et utilisé dans la littérature. Il prend en paramètre :

- Le nombre de nœuds n du graphe à générer.
- Une matrice symétrique de probabilités $P = [P_{ab}]$ de taille $r \times r$. Dans ce rapport, on nommera « cin » les coefficients de la diagonale de cette matrice, et « cout » les coefficients en dehors de la diagonale.
- Une partition $\{C_1, C_2, \dots, C_r\}$ des nœuds du graphe (où C_i est la i-ème communauté) telle que :
    - $\bigcup C_i = ensemble\ des\ noeuds\ du\ graphe$
    - $C_i \cap C_j = \emptyset$
    - $C_i \neq \emptyset$

La probabilité pour que deux nœuds $u \in C_a$ et $v \in C_b$ soient reliés par une arête est donc définie comme étant égale à P_{ab}.

En modifiant la matrice de probabilité on peut donc obtenir des graphes avec certaines propriétés.

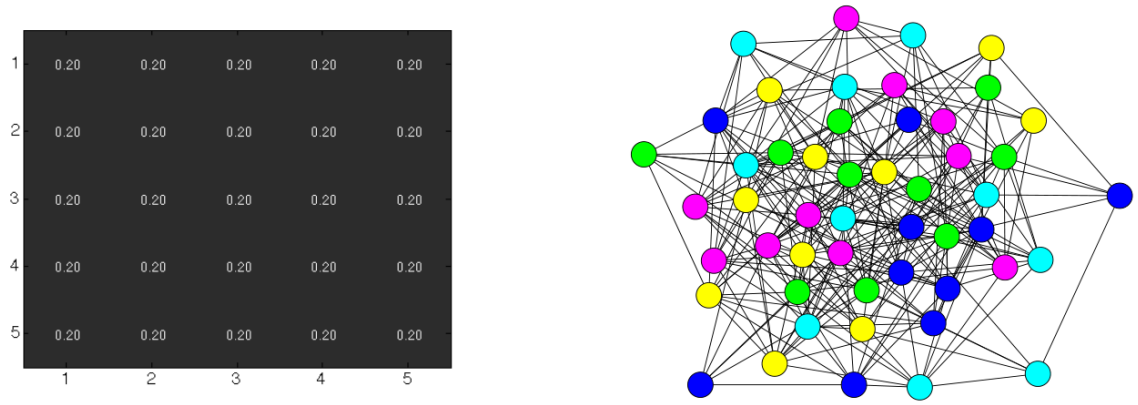

Figure 1 : Graphe purement aléatoire, distribution uniforme

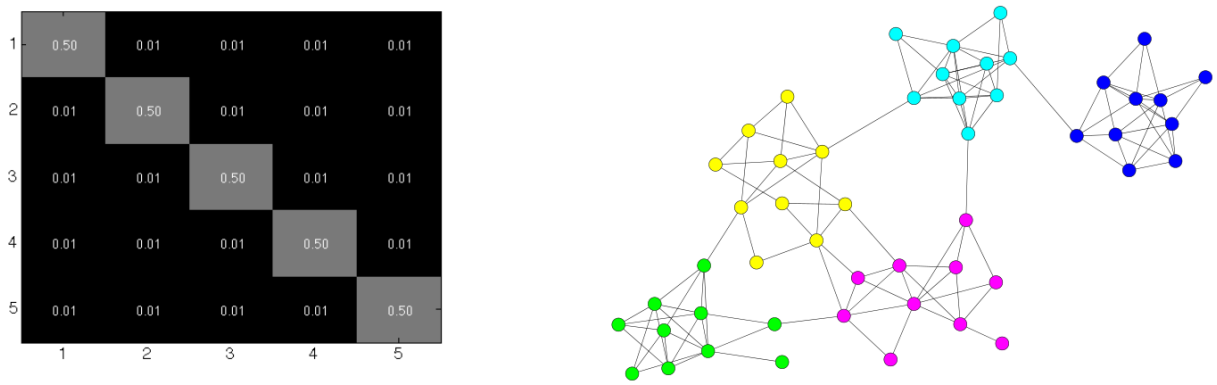

Figure 2 : « Assortative model », on distincte bien les communautés, les nœuds intra-communautaires sont fortement connectés entre eux

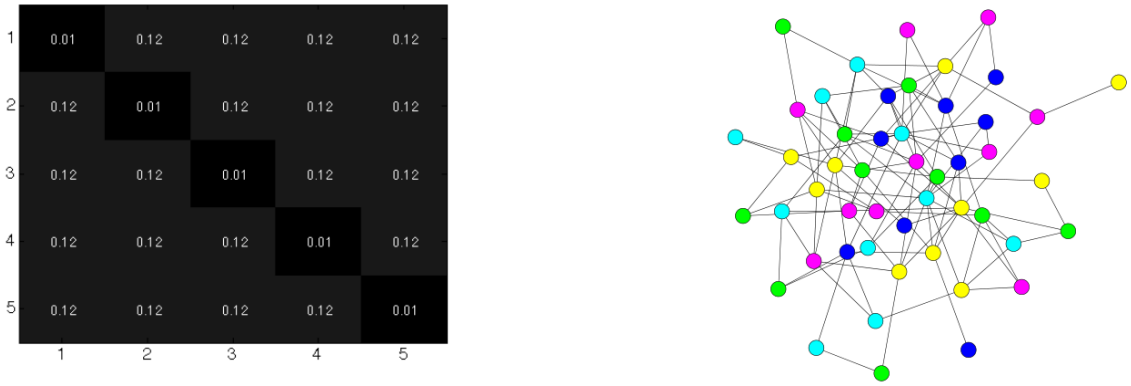

Figure 3 : « Dissortative model », les communautés sont difficilement distinguables, les nœuds intra-communautaires sont faiblement connectés entre eux

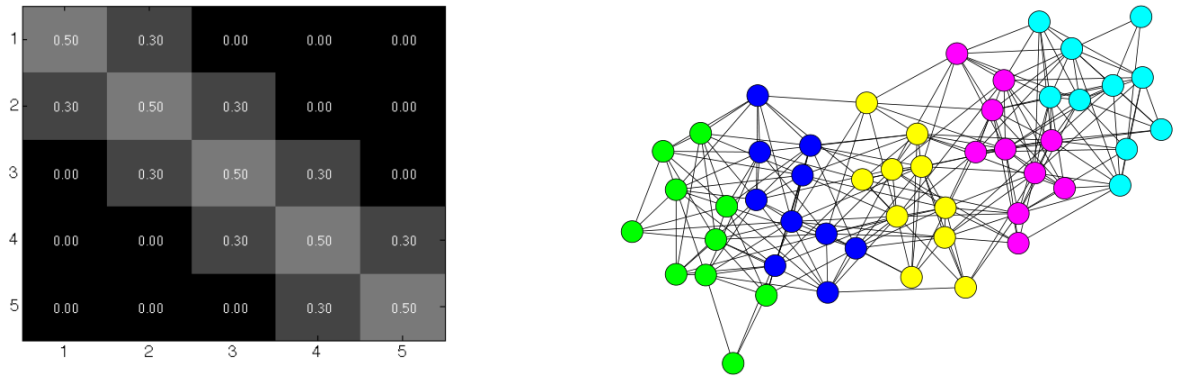

Figure 4 : Communautés ordonnées

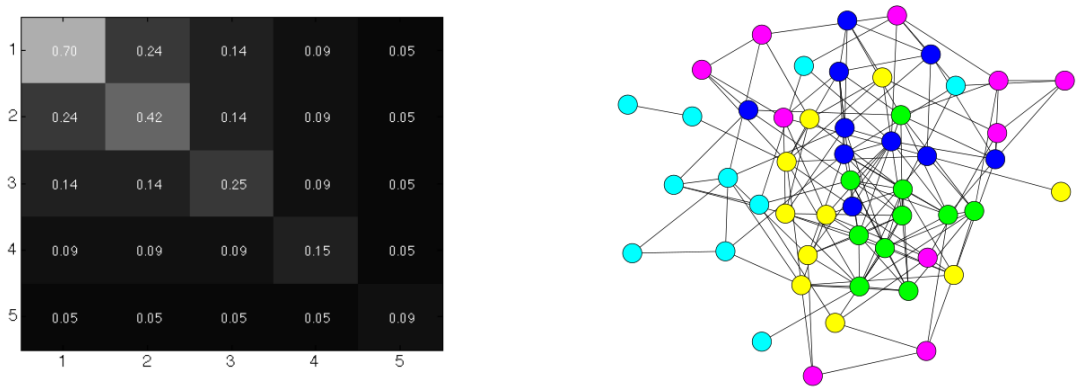

Figure 5 : « Core-periphery structure », avec une communauté centrale (verte)

Pour la suite du rapport nous allons préciser quelques définitions sur la théorie des graphes :

- La matrice d'adjacence d'un graphe $A = [A_{ij}]$ de taille $n \times n$ est définie telle que :

$$A_{ij} = \begin{cases} 1 \; si \; le \; noeud \; i \; et \; le \; noeud \; j \; sont \; liés \; par \; une \; arête \\ 0 \; sinon \end{cases}$$

  Si le graphe n'est pas orienté la matrice d'adjacence est alors symétrique. Nous aborderons dans ce rapport uniquement le cas des graphes non orientés.

- Le degré d'un nœud est défini comme étant le nombre d'arêtes qui lui sont incidents. On a alors :

$$d_i = \sum_{j=1}^{n} A_{ij}$$

- Dans le cas des graphes générés par le stochastic block model, on montre que l'espérance du nœud i de degré d_i appartenant à la communauté C_a est égale à :

$$E[d_i] = E\left[\sum_{j=1}^{n} A_{ij}\right] = \sum_{j=1}^{n} E[A_{ij}] = \sum \{P_{ab} \mid X_j \in C_b\} = \sum_{b=1}^{r} n_b P_{ab}$$

Sous forme matricielle on a :

$$\begin{pmatrix} E[d_1] \\ \vdots \\ E[d_r] \end{pmatrix} = \begin{pmatrix} P_{11} & \cdots & P_{1r} \\ \vdots & \ddots & \vdots \\ P_{r1} & \cdots & P_{rr} \end{pmatrix} \times \begin{pmatrix} n_1 \\ \vdots \\ n_r \end{pmatrix}$$

Où n_i est le nombre de nœud dans le communauté i.

On distingue en général deux types de graphe :

- Les graphes denses, pour lesquels le degré moyen des nœuds augmente lorsque le nombre de nœuds augmente
- Les graphes « sparses », pour lesquels le degré moyen des nœuds reste constant même si le nombre de nœuds augmente.

Pour générer un graphe sparse à partir du stochastic block model il suffit de diviser la matrice de probabilité par le nombre de nœud du graphe désiré.

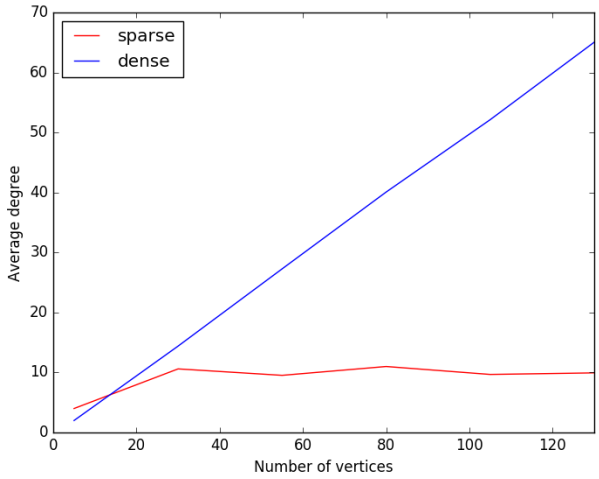

Figure 6 : Comparaison entre graphe sparse et graphe dense

Nous avons fait le choix d'implémenter nos algorithmes en langage Python, qui présente l'avantage d'être très accessible et qui possède un grand nombre de bibliothèques permettant la visualisation de graphes.

La matrice d'adjacence d'un graphe pouvant être très grande et aussi très creuse, on utilise en général le format de donnée « dictionnaire » propre au langage Python, qui utilise des tableaux non pas indexés par des numéros mais par des clés pouvant être de n'importe quelle nature, par exemple un nœud d'un graphe.

```
A -> B
A -> C
B -> C
B -> D
C -> D
D -> C
E -> F
F -> C
```

```
graph = {'A': ['B', 'C'],
         'B': ['C', 'D'],
         'C': ['D'],
         'D': ['C'],
         'E': ['F'],
         'F': ['C']}
```

Figure 7 : Représentation symbolique d'un graphe à gauche et équivalent sous forme dictionnaire Python à droite

Ce type de structure de données à l'avantage d'être très rapide à générer et consomme peu de mémoire. Cependant, pour les besoins de notre projet nous n'avons pas utilisé ce type de structure car les matrices d'affinités qui nous serons utiles par la suite doivent être calculées à partir de la matrice d'adjacence, il est donc plus simple de stocker la matrice d'adjacence directement dans la mémoire sous forme « d'array » classique.

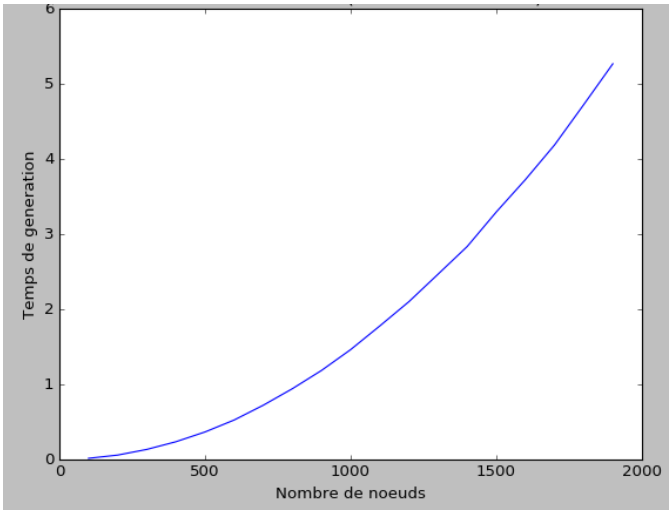

Figure 8 : Temps de génération (sec) d'un graphe par le stochastic block model en fonction du nombre de nœuds

Avec notre implémentation on obtient donc une complexité quadratique.

Algorithmes spectraux

Les algorithmes spectraux de détection de communautés permettent de créer un partitionnement d'un graphe en temps polynomial. Ces algorithmes se basent sur une méthode de classification non supervisée appliquée sur les vecteurs propres d'une matrice d'affinité définie à partir de la matrice d'adjacence d'un graphe. Dans le cadre de notre projet nous étudierons la matrice Laplacienne, la matrice de modularité, la matrice d'adjacence ainsi que la matrice de Bethe-Hess.

Lorsque l'on trace l'histogramme des valeurs propres d'une matrice d'affinité on constate que certaines valeurs propres sont isolées par rapport aux autres. En réalité, ce sont les K plus grandes ou les K plus petites valeurs propres (selon comment sont définies les matrices), K étant le nombre de communautés dans le graphe.

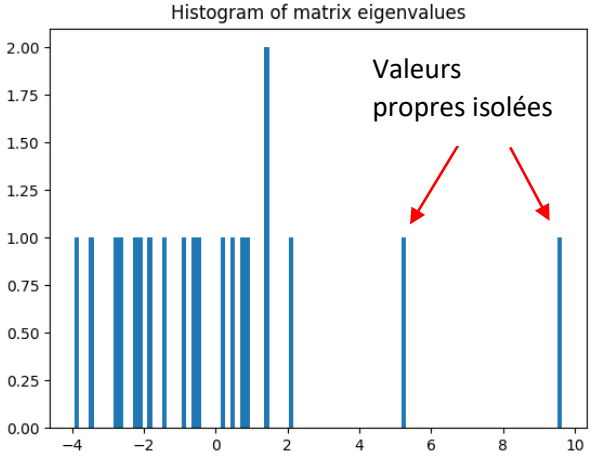

Figure 9 : Histogramme des valeurs propres d'une matrice d'affinité à deux communautés

On forme alors la matrice X avec les K vecteurs propres associés à ces K valeurs propres :

$$X = [v_1 \quad \dots \quad v_K]$$

On applique alors une méthode de classification non supervisée sur les entrées de ces vecteurs propres. Dans notre projet, nous utilisons la méthode la plus simple et la plus répandue dans la littérature : la méthode des K-moyennes.

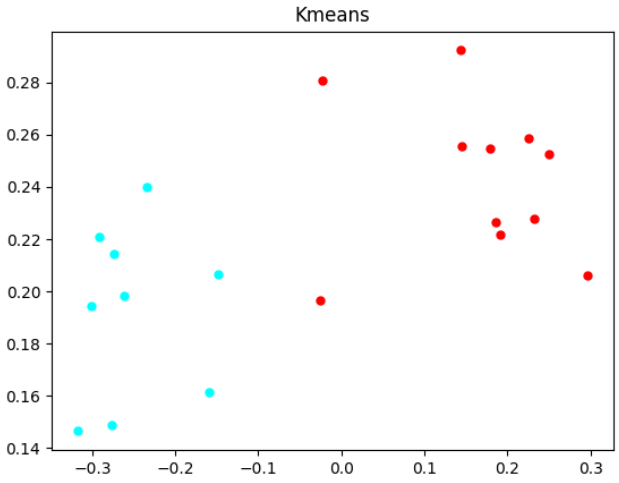

Figure 10 : Tracé des deux vecteurs propres et résultat de la classification

Les lignes de X étant maintenant classées en K communautés, on associe au nœud i du graphe la communauté correspondant à la i^{ième} ligne de X.

Choix de la matrice d'affinité

La littérature propose plusieurs formes de matrices d'affinité pour les algorithmes spectraux. Nous étudierons ici la matrice Laplacienne, la matrice de modularité, la matrice de Bethe-Hess et la matrice d'adjacence.

Matrice Laplacienne

La matrice Laplacienne est définie par :

$$L(A) = I - D^{-\frac{1}{2}} . A . D^{-\frac{1}{2}}$$

Où :

- A est la matrice d'adjacence du graphe
- I est la matrice identité de même taille que A
- D est la matrice diagonale où le coefficient D_{ii} est égal au degré du sommet i.

Matrice de modularité

La matrice de modularité est définie par :

$$M(A) = A - \frac{d^T . d}{n_{arcs}}$$

Où :

- A est la matrice d'adjacence du graphe
- d est le vecteur qui contient les degrés des nœuds du graphe
- n_{arcs} est le nombre d'arcs du graphe.

Matrice de Bethe-Hess

La matrice de Bethe-Hess (ou Bethe-Hessian Matrix) est la matrice définie par :

$$B(r) = (r^2 - 1) - rA + D$$

Où :

- A est la matrice d'adjacence du graphe
- I est la matrice identité de même taille que A
- D est la matrice diagonale où D_{ii} est égal au degré du sommet i
- r est un paramètre à choisir.

Les histogrammes ci-dessous ont été générés à partir d'un graphe à deux communautés. On cherche donc à ce que les deux valeurs propres les plus faibles, représentées par des traits rouges, soient donc le plus isolées possible. Pour r petit, il est très difficile de distinguer la deuxième valeur propre à isoler, la détection de communautés ne peut donc pas être efficace. Lorsque r augmente, les deux plus petites valeurs propres se distinguent mieux. La publication [4] conseille de prendre r comme étant la racine carrée du degré moyen du graphe et on constate que ce choix est effectivement le meilleur sur les figures 11 et 12, où la racine carrée du degré moyen des nœuds est égale à 2.0027.

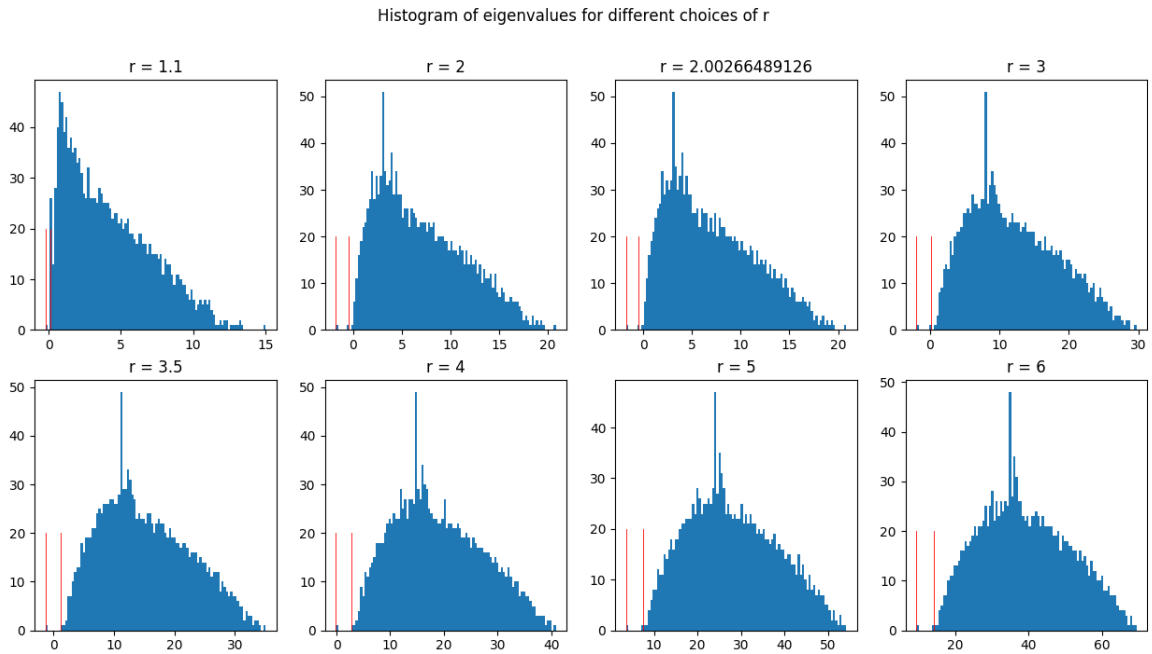

Figure 11 : Histogrammes des valeurs propres pour plusieurs valeurs de r

L'information mutuelle normalisée (NMI) est une mesure de la dépendance statistique de deux vecteurs aléatoires. Cette quantité permet donc de mesurer la qualité d'un partitionnement en communautés d'un graphe dont on connait un partitionnement de référence (par exemple si le graphe a été généré à l'aide du stochastic block model).

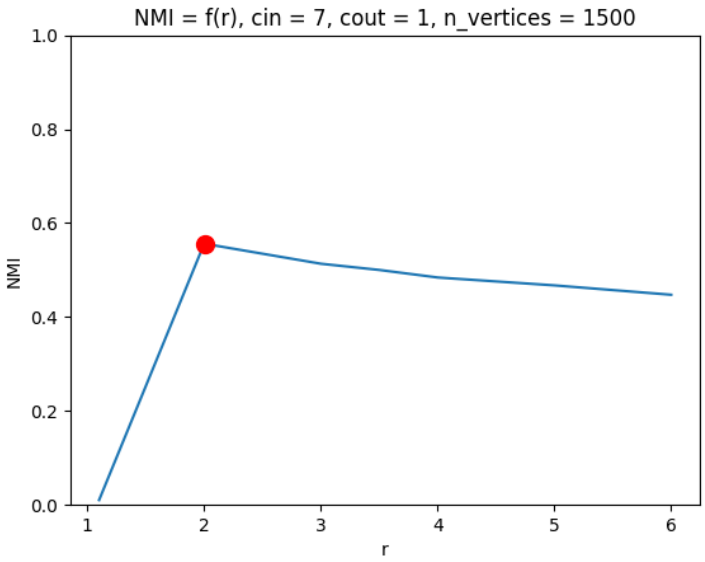

Figure 12 : Information Mutuelle Normalisée en fonction de r

La figure ci-dessus donne le tracé de l'information mutuelle normalisée en fonction du choix de r. Le point rouge correspond au r égal à la racine du degré moyen des sommets. Ce tracé confirme donc bien que ce choix de r est optimal.

Analyse des résultats

La figure ci-dessous montre la qualité de partitionnement de chaque matrice en fonction du nombre de nœud d'un graphe « sparse » à deux communautés généré par le stochastic block model. Nous avons réalisé trois mesures expérimentales puis moyenné les résultats de celles-ci.

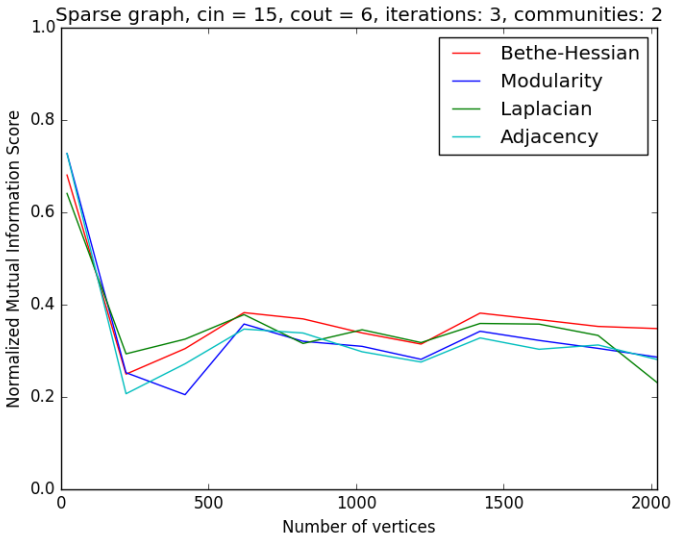

Figure 13 : NMI de chaque matrice en fonction du nombre de nœuds sur un graphe sparse

On constate globalement que la matrice Bethe-Hessian possède de meilleurs résultats que les autres. Cependant l'écart n'est pas aussi important que celui trouvé dans la publication [4].

La figure ci-dessous montre les résultats de la même expérience mais réalisée cette fois ci sur un graphe dense.

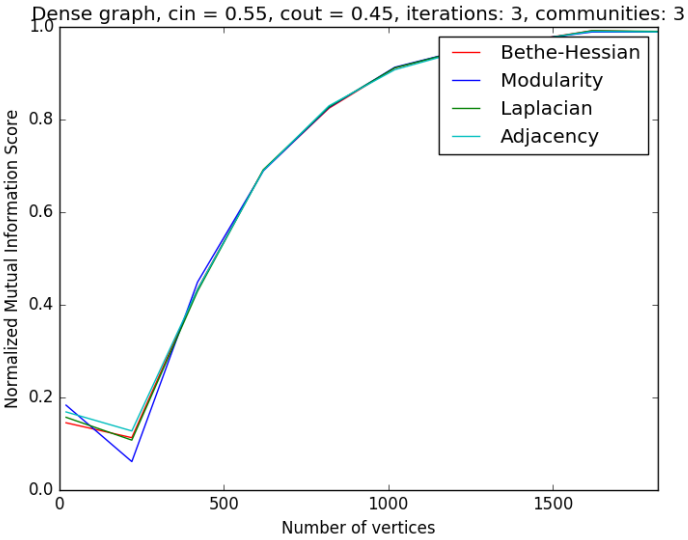

Figure 14 : NMI de chaque matrice en fonction du nombre de nœuds sur un graphe dense

Il est intéressant de remarquer que les matrices ont des performances très similaires et que plus le nombre de nœud augmente plus la qualité de partitionnement est bonne. Si l'on regarde le tracé des deux vecteurs propres isolés et le résultat des K-means sur la figure ci-dessous, on constate que plus

le nombre de nœud augmente plus les communautés sont facilement distinguables. L'algorithme des K-means est donc particulièrement efficace lorsque le nombre nœud augmente.

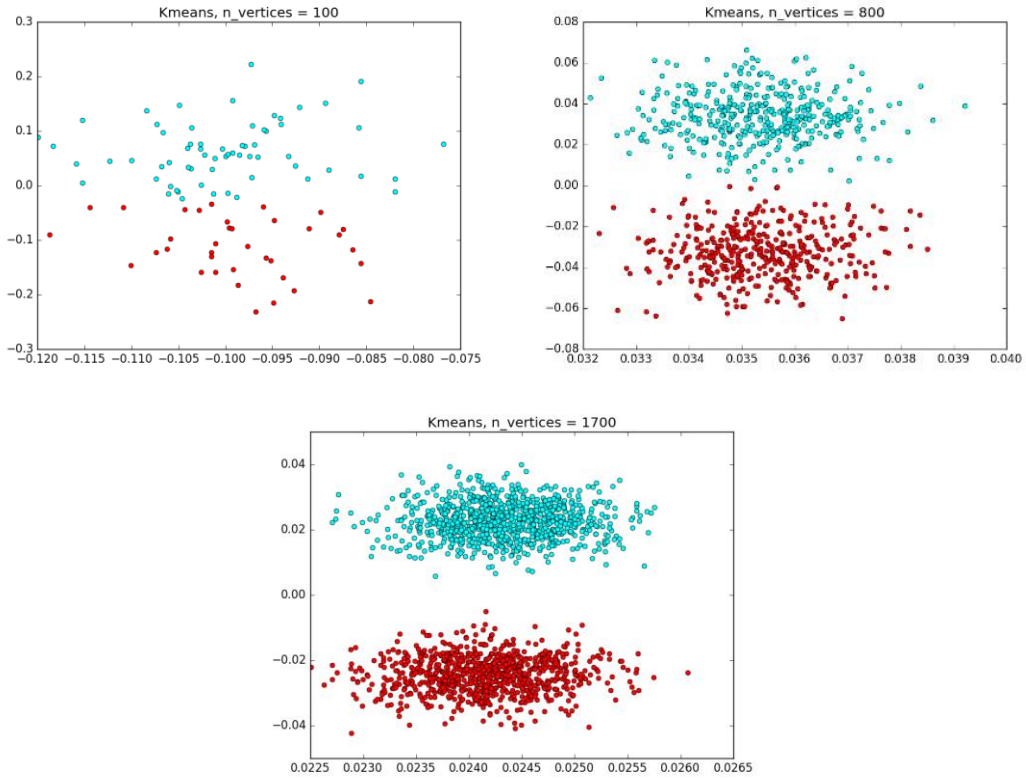

Figure 15 : Résultat des K-means pour des graphes de différentes tailles

Sur la figure ci-dessous on trace la NMI en fonction de la différence des coefficients de la matrice de probabilité du stochastic block model. Nous avons réalisé quatre mesures expérimentales puis moyenné les résultats de celles-ci.

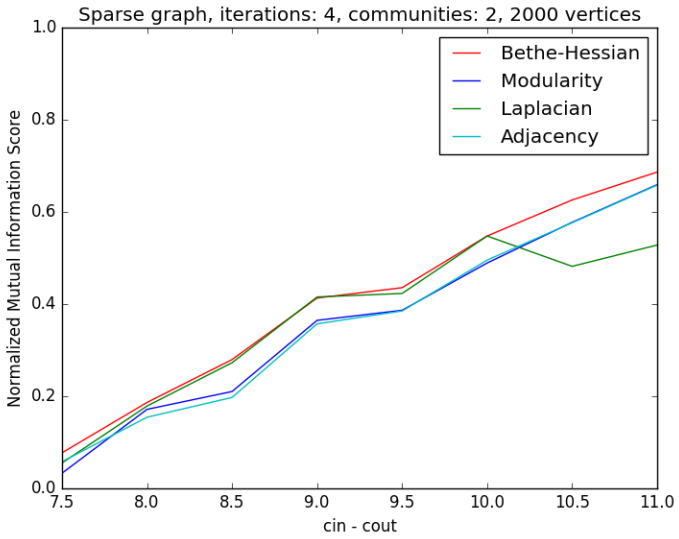

Figure 16 : NMI de chaque matrice en fonction de la différence de cin et cout

On remarque globalement que la matrice Bethe-Hessian est la meilleure.

La publication [16] montre que le partitionnement d'un graphe n'est possible que si la règle suivante est respectée :

$$abs(cin - cout) > nombre\ de\ communauté\ \times \sqrt{degré\ moyen\ des\ noeuds}$$

Dans la figure 16 on a deux communautés et un degré moyen de 10, la borne est donc environ située à 7, ce qui correspond bien aux résultats obtenus.

Sur la figure ci-dessous on trace la NMI en fonction du nombre de communautés. Nous avons réalisé trois mesures expérimentales puis moyenné les résultats de celles-ci.

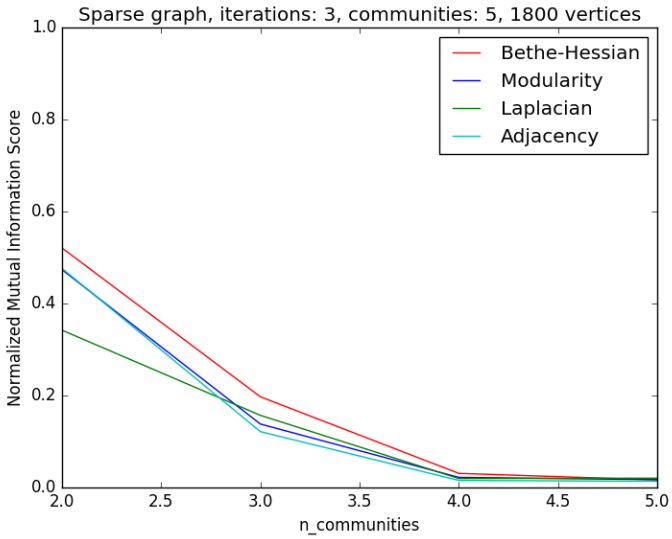

Figure 17 : NMI de chaque matrice en fonction de la différence de cin et cout

On constate encore une fois que la Bethe-Hessian possède les meilleures performances. Au-dessus de trois communautés les différentes matrices ne parviennent plus à partitionner le graphe correctement.

Conclusion

Au cours de ce projet, nous avons conçu un programme complet permettent la génération de graphes aléatoires par le stochastic block model ainsi que la recherche de communautés à l'aide de méthodes spectrales basées sur différentes matrices d'affinités. Nous avons donc pu étudier les diverses matrices d'affinités ainsi que leurs performances sous différents critères.

On constate globalement que la matrice Bethe-Hessian obtient les meilleures performances. Les matrices d'adjacence et de modularité montrent des résultats similaires. Quant à la matrice Laplacienne, elle possède des résultats assez variables, et il faut remarquer que celle-ci est assez difficile à obtenir numériquement à cause du calcul de $D^{-\frac{1}{2}}$.

En ce qui concerne la méthode des K-means, nous avons supposés que nous connaissions le nombre de communautés, ce qui n'est pas forcément toujours le cas. Dans le cas où le nombre de communautés est inconnu, on peut alors appliquer l'algorithme spectral utilisant la matrice de Bethe-Hess et choisir un nombre de communautés égal au nombre de valeurs propres négatives [15]. On peut également choisir un algorithme de classification différent de K-means ne nécessitant pas de connaitre à l'avance le nombre de communautés, tel que l'algorithme glouton-simple.

Références

[1] AARON CLAUSET. Network Analysis and Modelling course.

[2] SANTO FORTUNATO. Community detection in graphs. Physics Reports.

[3] RAJ RAO NADAKUDITI and MARK EJ NEWMAN. Graph spectra and the detectability of community structure in networks. Physical review letters.

[4] ALAA SAADE, FLORENT KRZAKALA, and LENKA ZDEBOROVA. Spectral clustering of graphs with the Bethe hessian. In Advances in Neural Information Processing Systems.

[5] ROMAIN COUILLET. A random matrix approach to machine learning.

[6] LAURENT MASSOULIE. Community detection with spectral methods.

[7] LUCA DONETTI, MIGUEL A. MUNOZ. Detecting network communities: a new systematic and efficient algorithm.

[8] CAN M. LE. Estimating community structure in networks by spectral methods.

[9] XIAO ZHANG, M. E. J. NEWMAN. Multiway spectral community detection in networks.

[10] MARC LELARGE. Algorithme des réseaux sociaux.

[11] HAFIZ TIOMOKO ALI, ROMAIN COUILLET. Random matrix improved community detection in heterogeneous networks.

[12] EMMANUEL ABBE, COLIN SANDON. Recovering communities in the general stochastic block model without knowing the parameters.

[13] PAN ZHANG. Robust spectral detection of global structures in the data by learning a regularization.

[14] M. E. J. NEWMAN. Spectral methods for network community detection and graph partitioning.

[15] KEEGAN GO, KENJI HATA. Statistical Physics of community detection.

[16] A. DECELLE, F. KRZAKALA, C. MOORE, and L. ZDEBOROVÁ. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications.

[17] F. DE MONGOLFIER, M. SOTO, L. VIENNOT. Clustering de métrique et clustering de graphe.