

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

[Sommaire 2](#_Toc477022843)

[Introduction 3](#_Toc477022844)

[Génération de graphes aléatoires 4](#_Toc477022845)

[Algorithmes spectraux 7](#_Toc477022846)

[Analyse des résultats 7](#_Toc477022847)

[Conclusion 7](#_Toc477022848)

[Références 8](#_Toc477022849)

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, et 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 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 du graphe à générer
* Une matrice symétrique de probabilités de taille
* Une partition des nœuds du graphe (où est la i-ème communauté) telle que :

La probabilité pour que deux nœuds et soient reliés par une arête est donc définie comme étant égale à .

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

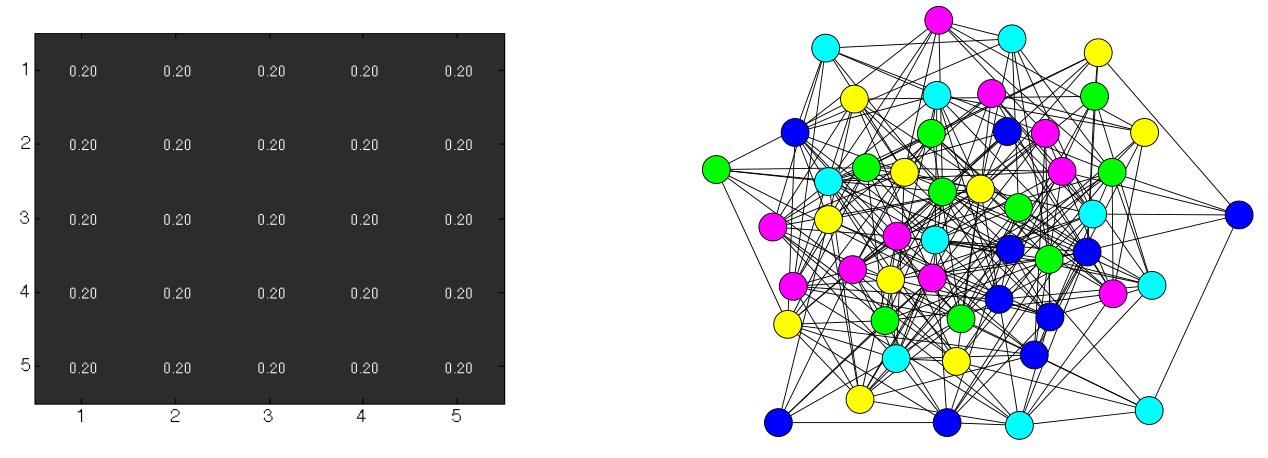


Figure : Graphe purement aléatoire, distribution uniforme

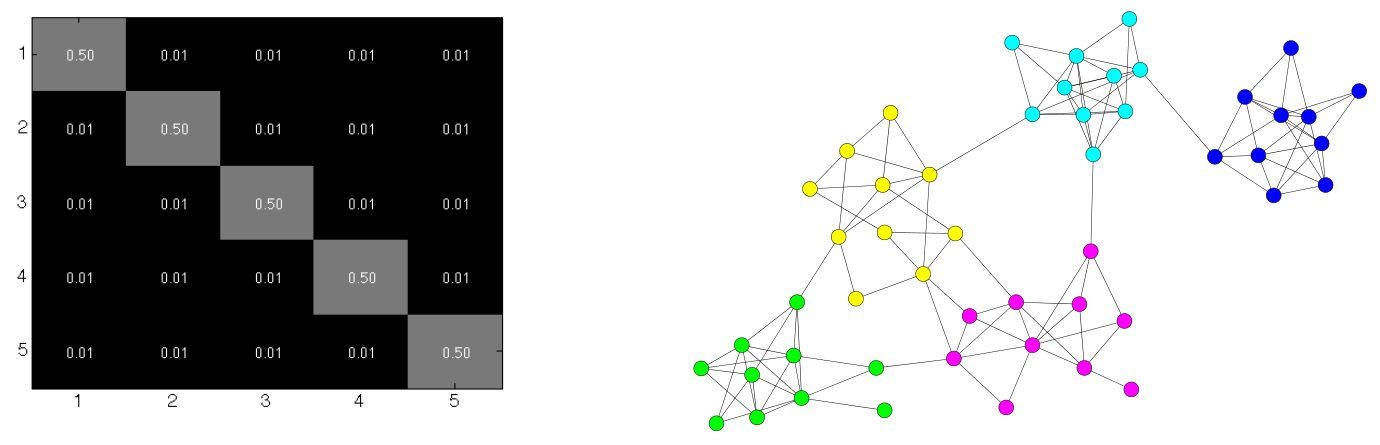


Figure : « Assortative model », on distincte bien les communautés, les nœuds intra-communautaires sont fortement connectés entre eux

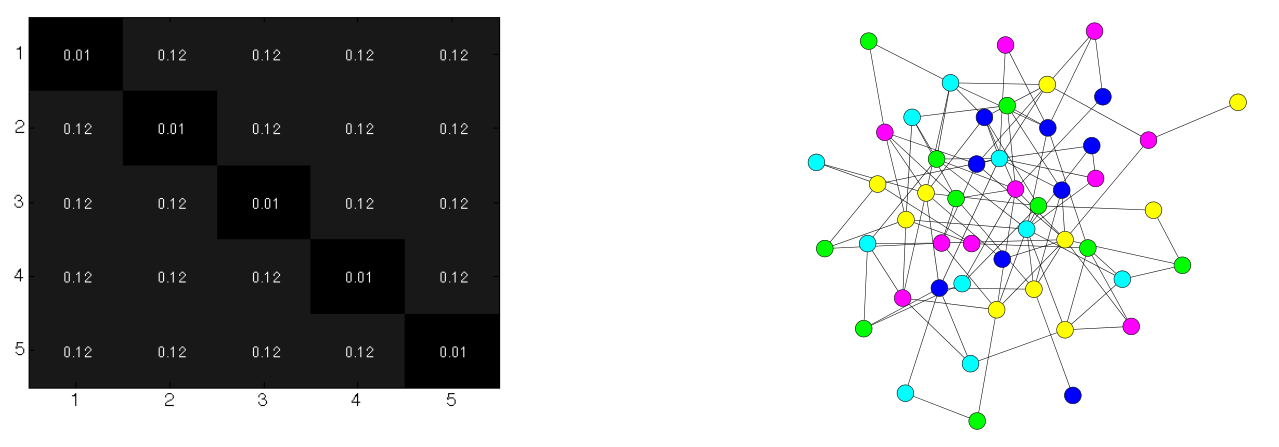


Figure : « Dissortative model », les communautés sont difficilement distinguables, les nœuds intra-communautaires sont faiblement connectés entre eux

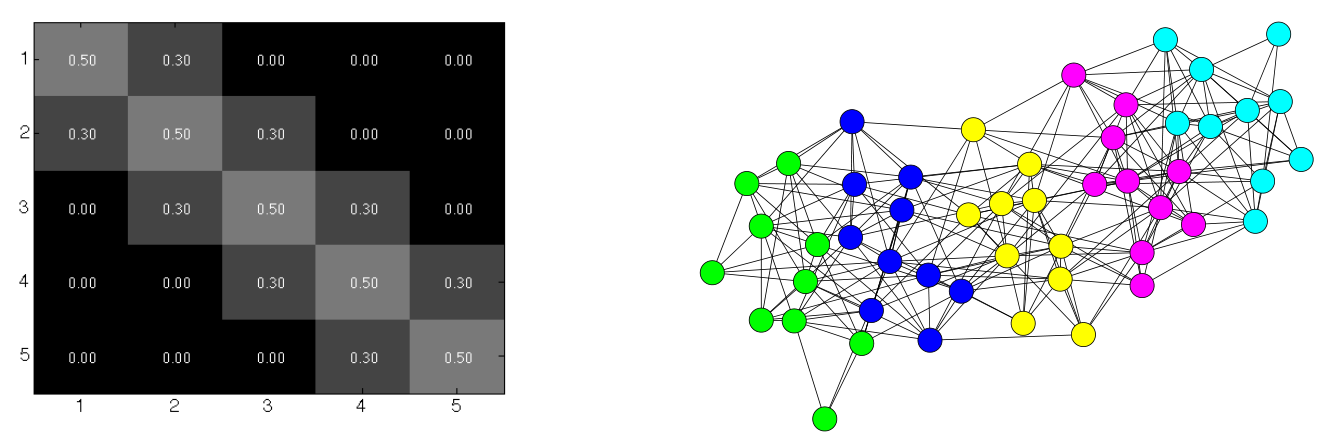


Figure : Communautés ordonnées

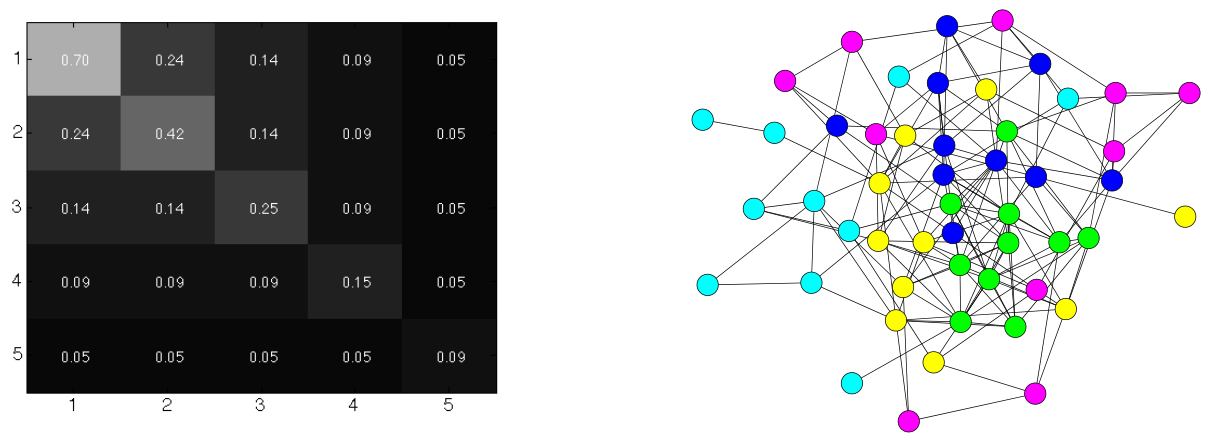


Figure : « Core-periphery structure », avec une communauté centrale (verte)

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

* La matrice d’adjacence d’un graphe de taille est définie telle que :

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 :
* Dans le cas des graphes générés par le stochastic block model, on montre que l’espérance du nœud de degré appartenant à la communauté est égale à :

Sous forme matricielle on a :

Où est le nombre de nœud dans le communauté .

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.

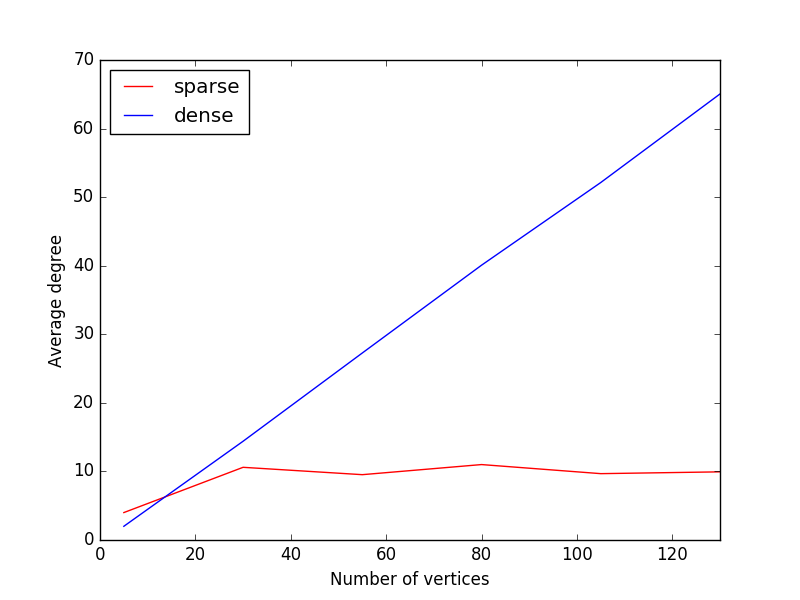


Figure : Comparaison entre graphe sparse et graphe dense

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.

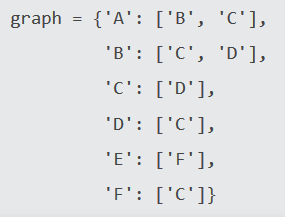
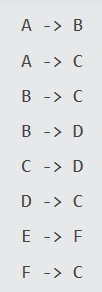


Figure : Représentation symbolique d’un graphe à gauche et équivalent sous forme dictionnaire Python à droite

Utiliser 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. Nous avons donc fait le choix de stocker la matrice d’adjacence directement dans la mémoire sous forme de tableau classique.

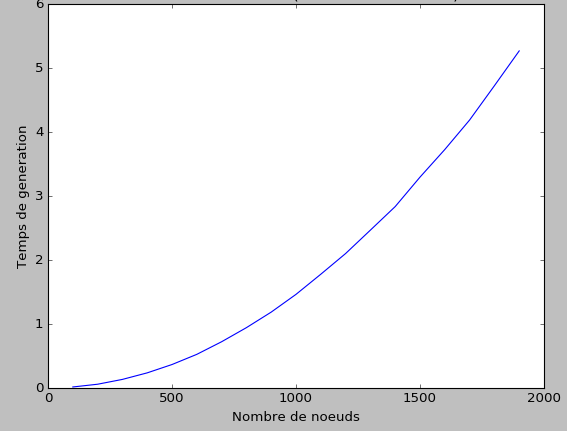


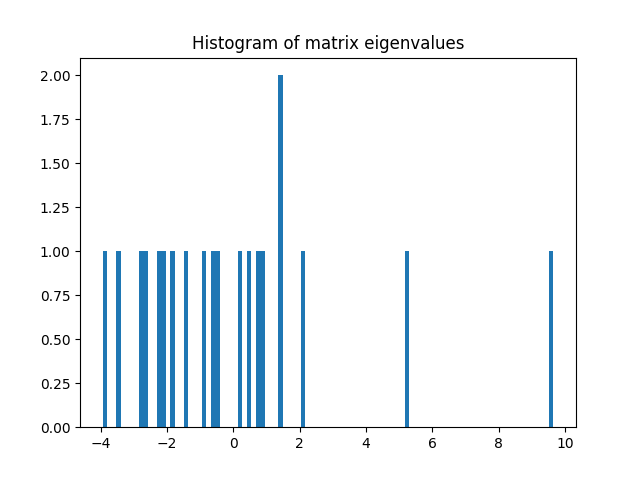
Figure : Temps de génération (sec) d’un graphe par le stochastic block model par rapport au nombre de nœuds

On obtient finalement une complexité quadratique.

Algorithmes spectraux

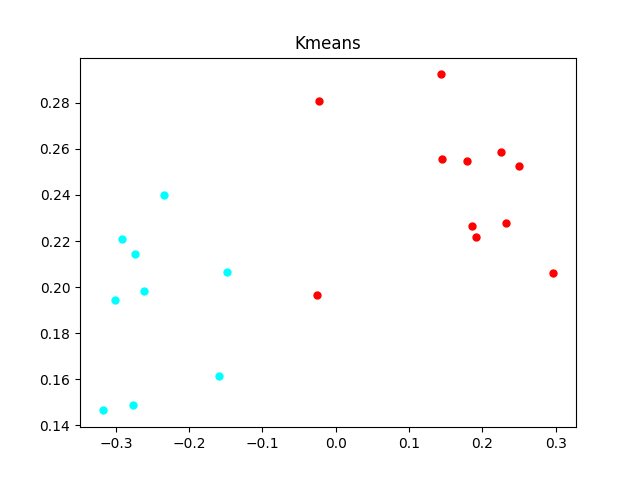
Les algorithmes spectraux de détection de communautés permettent de créer un partitionnement d’un graph en communautés en temps polynomial. Ces algorithmes se basent sur une méthode de classification non supervisée appliquée sur les vecteurs propres d’une matrice d’affinité définie à partir de la matrice d’adjacence.

On choisit donc une matrice d’affinité définie à partir de la matrice d’adjacence du graph. Nous étudierons la matrice laplacienne, la matrice de modularité, la matrice de Bethe-Hess et la matrice de modularité. Les valeurs propres de ces matrices sont regroupées autour d’une valeur, sauf les K plus grandes (pour les matrices d’adjacence ou de modularité) ou les K plus petite (pour les matrices de Laplace ou de Bethe-Hess), K étant le nombre de communautés dans le graphe.



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

On applique alors une méthode de classification non supervisés. Dans notre projet, nous utilisons la méthode la plus répandue dans la littérature : la méthode des K-moyennes. Il est cependant possible d’utiliser un autre algorithme.



Les lignes de X étant maintenant classées en K communautés, on associe au sommet n du graphe à la communauté correspondant à la nième ligne de X.

# Choix de la matrice d’affinité

La littérature propose plusieurs formes de matrices d’affinité pour les algorithmes spectraux. Nous étudierons 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 :

Où :

* A est la matrice d’adjacence du graph
* I est la matrice identité de même taille que A
* D est la matrice diagonale où Dii est égal au degré du sommet i.

## Matrice de modularité

La matrice de modularité est définie par :

Où :

* A est la matrice d’adjacence du graph
* D est la matrice diagonale telle que Dii est égal au degré du sommet i
* narcs est le nombre d’arcs du graph.

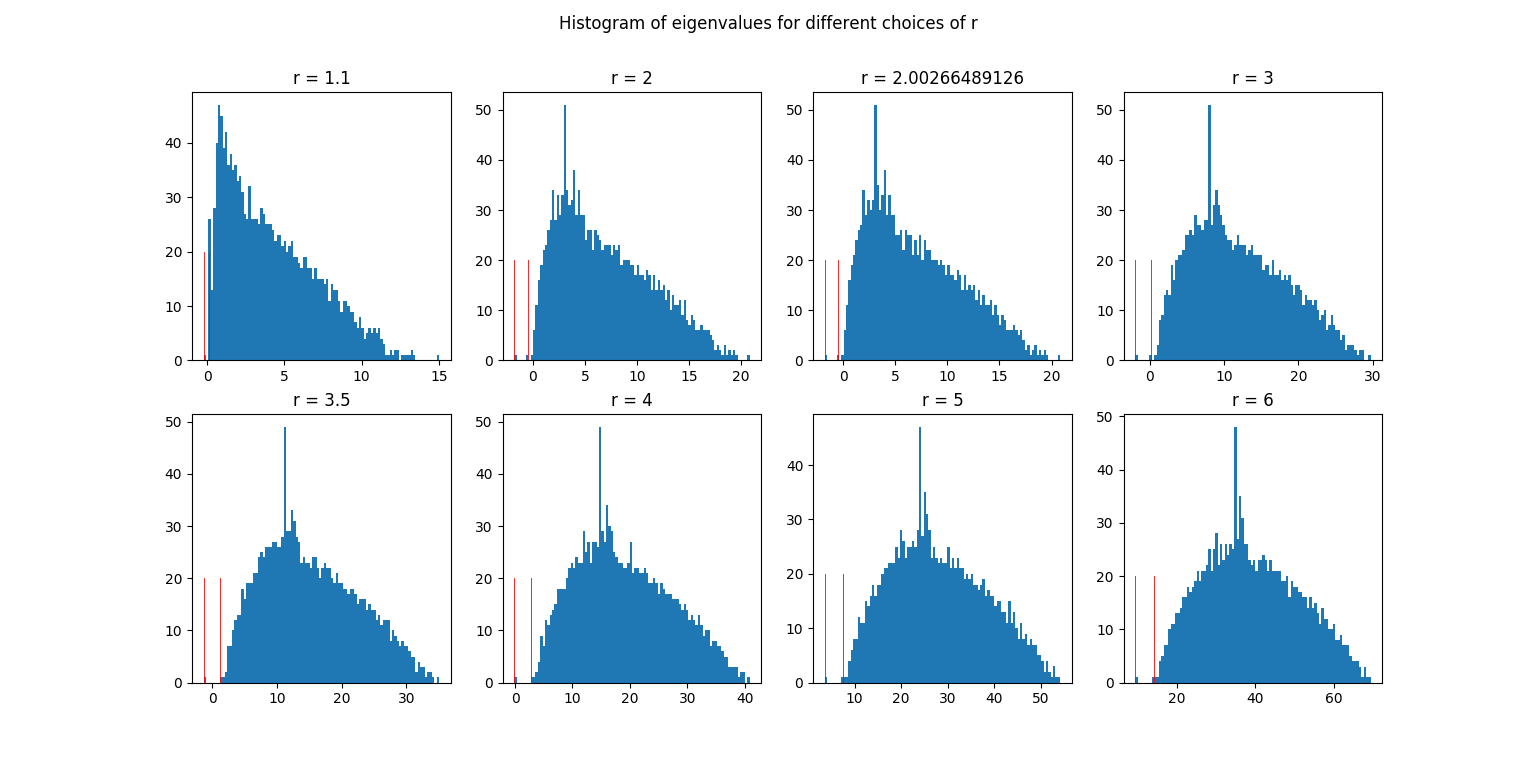
## Matrice de Bethe-Hess

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

Où :

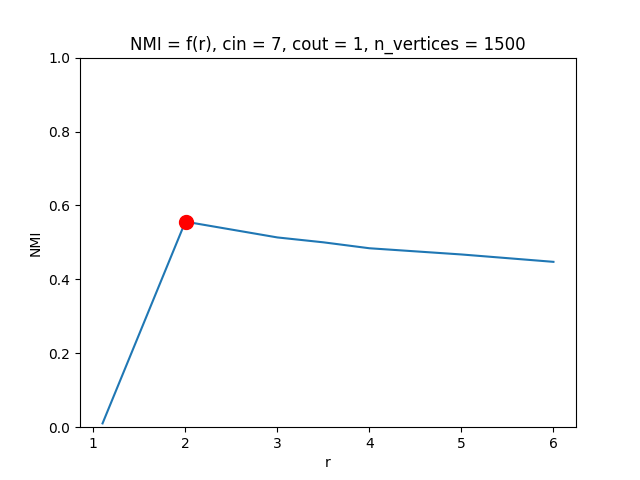
* A est la matrice d’adjacence du graph
* I est la matrice identité de même taille que A
* D est la matrice diagonale où Dii est égal au degré du sommet i
* r est un paramètre à choisir.

Nous devons donc choisir une valeur pour r.



Les histogrammes ci-dessus ont été générés à partir d’un graph à deux communautés. On cherche donc à ce que les deux valeurs propres les plus faibles, représentées par des trais rouges, soient donc le plus isolées possible. Pour r petit, la deuxième valeur propre n’est pas isolée, la détection de communautés ne peut donc pas être efficace. Lorsque r augmente, les deux plus petites valeurs propres se distinguent mieux. Pour r = 2,0027, correspondant à la racine du degré moyen des sommets du graph, les deux plus petites valeurs propres se distinguent au mieux. Si r augmente encore, la deuxième valeur propre se rapproche encore des autres valeurs propres. On choisit donc r égal à la racine du degré moyen des sommets.

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 graph a été généré à l’aide du stochastic block model).



La figure ci-dessus est le tracé de l’information mutuelle normalisée en fonction du choix de r pour l’application de l’algorithme spectral utilisant la matrice de Bethe-Hess sur une matrice générée avec le stochastic bloc model. Le point correspond au r idéal égal à la racine du degré moyen des sommets. Ce tracé confirme donc le choir de r basé sur l’histogramme des valeurs propres. Dans la suite nous n’utiliserons que ce r.

Analyse des résultats

Générateur de permutation, mutual information, tests sur graphs denses et sparses, influence de cin-cout

Conclusion

Comment connaitre n\_cluster et n\_communities

Références

AARON CLAUSET. Network Analysis and Modelling course.

SANTO FORTUNATO. Community detection in graphs. Physics Reports.

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

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

ROMAIN COUILLET. A random matrix approach to machine learning.

LAURENT MASSOULIE. Community detection with spectral methods.

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

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

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

MARC LELARGE. Algorithme des réseaux sociaux.

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

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

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

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

KEEGAN GO, KENJI HATA. Statistical Physics of community detection.