Détection automatique de structures blocs sur des matrices

Luce Le Gorrec, Sandrine Mouysset, Daniel Ruiz, Philip A. Knight, Iain S. Duff

Détection automatique de structures blocs sur des matrices

Luce le Gorrec[1], Sandrine Mouysset[2], Daniel Ruiz[3], Philip A. Knight[4] et Iain S. Duff[5]

[4]University of Strathclyde, Glasgow
[5]Rutherford Appleton Laboratory, Didcot
[1,2,3]Université de Toulouse - IRIT, Toulouse

Résumé

Nous présentons un algorithme capable de partitionner les noeuds d'un graphe pondéré dirigé ou non, en détectant la structure par blocs de sa matrice d'adjacence. Grâce à un équilibrage doublement stochastique de cette matrice, l'intégralité de la structure par blocs est détectée à l'aide d'un minimum de vecteurs singuliers (en théorie un couple suffit).

Nous présentons les différentes étapes de notre algorithme, avant d'évaluer ses performances sur deux tâches de classification non supervisée : la détection de communautés et la détection de formes dans des nuages de points. En comparant les résultats obtenus à ceux d'autres algorithmes spécifiquement conçus pour ces tâches, nous montrons que notre algorithme est compétitif—dans le cas de la détection de communautés—, ou meilleur—dans le cas de la détection de formes—, par rapport aux méthodes existantes.

1 Introduction

Regrouper les éléments similaires dans un jeu de données sans information a priori est le but de la classification non supervisée (ou clustering) pour lequel de nombreuses techniques ont été proposées. Un des moyens couramment utilisés consiste à trouver des blocs dans les matrices représentant le jeu de données–telles que les matrices d'adjacence ou d'affinité, les tables de contingences ou Laplacien du graphe.

Le lien entre les propriétés spectrales et structurelles des matrices est un élément-clé de nombreux algorithmes de clustering [For10, Sch07, FMSV08]. Pour détecter des classes (ou clusters), les algorithmes spectraux classiques séparent en deux l'ensemble des noeuds de façon récursive, en suivant le signe des entrées d'un vecteur singulier. Cela peut être numériquement coûteux puisqu'il est nécessaire de calculer un nouveau vecteur singulier pour chaque bipartition. De plus, il se peut que la partition trouvée à chaque itération soit éloignée de la structure par blocs réelle de la matrice (un exemple est fourni Figure 3.1(c)). Les autres méthodes spectrales existantes [NJW01, VL07, LR$^+$15] consistent à projeter les données dans le sous-espace vectoriel généré par les vecteurs propres dominants du Laplacien du graphe, en prenant autant de vecteurs qu'il y a de classes attendues, ce qui suppose que ce nombre est connu.

Dans ce papier, nous présentons une nouvelle approche spectrale pour le clustering. Notre but est de partitionner l'ensemble des noeuds d'un graphe pondéré, dirigé, et fortement connexe. Nous travaillons sur la matrice d'adjacence du graphe, notée $\mathbf{A}$. La principale nouveauté de notre méthode est de considérer les éléments singuliers de l'équilibrage doublement stochastique de la matrice $\mathbf{A}$. Trouver un équilibrage doublement stochastique pour une matrice $\mathbf{A}$ consiste à trouver deux matrices diagonales $\mathbf{D}$ et $\mathbf{F}$ telles que la somme sur chaque ligne et sur chaque colonne de $\mathbf{P} = \mathbf{DAF}$ est égale à 1. Nous montrons dans la Section 2 que l'équilibrage doublement stochastique améliore la fidélité de l'information structurelle contenue dans les vecteurs singuliers dominants de $\mathbf{P}$. En particulier, un faible nombre de vecteurs permet d'obtenir une information significative sur la structure par blocs sous-jacente de la matrice, le tout sans information complémentaire sur le nombre ou la taille des blocs.

En outre, travailler avec les vecteurs singuliers de $\mathbf{P}$ rend possible l'analyse des graphes dirigés et permet d'obtenir des clusterings précis, même lorsque la structure du graphe est très dissymétrique.

Le papier est construit de la manière suivante. En Section 2, nous exposons la connexion entre les vecteurs singuliers dominants d'une matrice doublement stochastique et sa structure par blocs. En Section 3 nous utilisons des outils du traitement du signal afin de détecter les informations sur le clustering contenues dans les vecteurs singuliers. Dans la Section 4, nous décrivons comment récupérer au fil des itérations des vec-

teurs singuliers qui continuent de fournir des informations utiles. La Section 5 est dédiée au réarrangement des blocs : nous devons compiler et analyser les clusterings venant des vecteurs singuliers gauche et droit, et éventuellement des itérations précédentes dans le cas où plus d'un couple de vecteurs est analysé. Nous présentons aussi une mesure adaptée à l'évaluation de la qualité de notre clustering et qui nous sert aussi pour notre critère d'arrêt. Enfin, des résultats empiriques sur des bancs d'essai sont présentés Section 6 afin d'indiquer l'efficacité de notre procédure.

2 Equilibrage doublement stochastique.

L'équilibrage doublement stochastique d'une matrice $\mathbf{A}$ existe à condition que $\mathbf{A}$ soit *bi-irréductible*, c'est-à-dire qu'il n'existe pas de permutations des lignes et des colonnes permettant de mettre $\mathbf{A}$ sous forme triangulaire par blocs. Comme cela est précisé Section 1, nous travaillons sur la matrice d'adjacence d'un graphe dirigé fortement connexe. $\mathbf{A}$ est donc carrée, positive, et bi-irréductible dès lors que sa diagonale ne comporte pas d'éléments nuls. Si nécessaire, nous ajoutons donc des termes non nuls dans sa diagonale, ce qui n'a pas d'effet sur la structure par blocs sous-jacentes que nous recherchons.

Le théorème suivant est une conséquence directe du théorème de Perron-Frobenius [Fro12, Per07].

Théorème 1 *Soit* $\mathbf{S} \in \mathbb{R}^{n\times n}$ *symétrique, irréductible et doublement stochastique. Alors* 1 *est valeur propre de* $\mathbf{S}$ *de multiplicité* 1*, et le vecteur propre associé est un multiple de* $\mathbf{e} = (1...1)^T$*. Par ailleurs, si* $\mathbf{S}$ *est symétriquement permutable en une matrice bloc diagonale avec* k *blocs irréductibles, alors* 1 *est valeur propre de multiplicité* k*, et une base du sous-espace propre associé est :*

$$\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\},$$

où v_{pq} *pour* $p \in \{1, \dots, k\}$ *est tel que*

$$v_{pq} = \begin{cases} 1, & \text{si } q \text{ est dans le bloc } p \\ 0, & \text{sinon.} \end{cases}$$

Ainsi, le calcul d'un vecteur propre associé à 1 d'une telle matrice $\mathbf{S}$ sera de la forme

$$\mathbf{x} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \dots + a_k\mathbf{v}_k.$$

En forçant $\mathbf{x} \in \mathbf{e}^{\perp}$, on peut raisonnablement supposer $a_i \neq a_j, \forall i \neq j$. Puisque ces vecteurs forment une partition disjointe de $\{1, \dots, n\}$, il est possible d'identifier la contribution de chaque bloc précisément, et donc de caractériser les partitions de manière exacte, en utilisant l'ensemble $\{a_1, \dots, a_k\}$. En effet, réordonner le vecteur singulier suivant un ordre adapté à la structure par blocs de $\mathbf{S}$ le mettra sous forme constante par morceaux. Cette idée est proche de celle exploitée dans [LN15], où les auteurs se servent de l'ordre croissant des vecteurs singuliers dominants de deux équilibrages stochastiques d'une matrice $\mathbf{A}$ afin de visualiser la structure par blocs de $\mathbf{A}$.

Comme corollaire du Théorème 1, considérons une matrice doublement stochastique non symétrique $\mathbf{P}$. Alors $\mathbf{P}\mathbf{P}^T$ et $\mathbf{P}^T\mathbf{P}$ sont toutes deux symétriques et doublement stochastiques, on peut donc leur appliquer le Théorème 1. Ainsi, si $\mathbf{P}\mathbf{P}^T$ ou $\mathbf{P}^T\mathbf{P}$ possède une structure par blocs, le vecteur singulier gauche ou droit dominant de $\mathbf{P}$ aura une contribution de chaque vecteur de la base associée, et il sera possible, comme précédemment, d'identifier la structure par blocs de lignes et/ou de colonnes de $\mathbf{P}$.

Notre algorithme est conçu pour des matrices qui sont des perturbations de matrices parfaitement diagonales par blocs. Les vecteurs singuliers dominants de telles matrices devraient avoir une structure proche d'une structure constante par morceaux. Ainsi, en réordonnant dans l'ordre croissant les vecteurs calculés, on devrait faire apparaître une structure proche d'une structure par blocs, ou au moins des blocs de lignes ou de colonnes faiblement corrélés les uns avec les autres. Le calcul de l'équilibrage doublement stochastique se fait avec la méthode de Newton décrite dans [KR13]. Cette méthode est peu coûteuse car elle ne fait intervenir que des produits matrices–vecteurs.

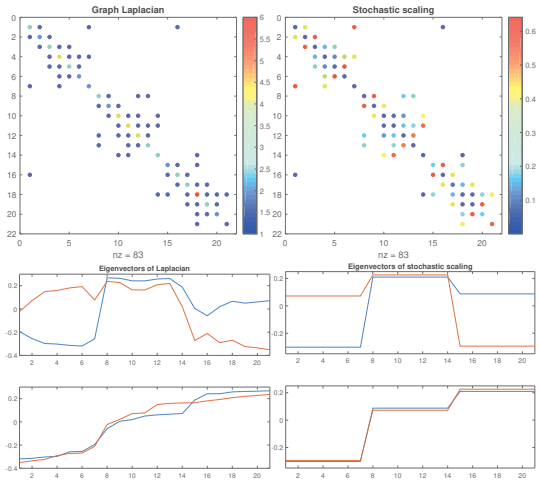

FIGURE 2.1 – Deux représentations matricielles d'un graphe et leurs vecteurs propres.

Afin de souligner l'avantage de l'équilibrage doublement stochastique par rapport à d'autres méthodes spectrales, nous utilisons le petit exemple Figure 2.1. Dans cet exemple, nous observons, pour un graphe présentant trois clusters distincts faiblement liés entre eux, l'information structurelle véhiculée par les vecteurs propres du Laplacien du graphe d'une part et ceux de sa matrice d'adjacence équilibrée d'autre part. Pour assurer la bi-irréducibilité, les termes diagonaux de la matrice d'adjacence sont fixés à 10^{-8} avant l'équilibrage. En bas de la Figure 2.1 on montre la structure numérique des vecteurs utilisés pour l'identification des clusters : les vecteurs associés aux deux plus petites valeurs propres pour le Laplacien (le vecteur de Fiedler est en bleu) ne permettent pas de détecter nettement les clusters (cela reste vrai pour le Laplacien normalisé). A contrario, les deuxième et troisième vecteurs propres dominants de la matrice stochastique permettent de caractériser les clusters parfaitement et sans ambiguïté.

3 Identification des clusters

Un point-clé de notre algorithme est d'identifier les clusters à l'aide des vecteurs singuliers de la matrice bi-irréductible doublement stochastique $\mathbf{P}$, sans connaissance préalable du nombre de clusters ni des permutations des lignes et des colonnes faisant apparaître la structure par blocs sous-jacente.

Les algorithmes de partitionnement qui travaillent sur des vecteurs analogues au vecteur de Fiedler divisent la matrice en deux blocs selon le signe des éléments du vecteur. Mais avec notre équilibrage doublement stochastique plus de deux blocs apparaissent sur un même vecteur, comme expliqué Section 2. Si $\mathbf{P}\mathbf{P}^T$ ou $\mathbf{P}^T\mathbf{P}$ a une structure sous-jacente quasi par blocs, ses vecteur propres dominants doivent avoir une structure quasi constante par morceaux après réordonnancement dans l'ordre croissant de ses entrées, comme dans la Figure 2.1. Notre problème est alors de détecter les paliers du vecteur réordonné que nous avons choisi d'envisager comme un problème de traitement du signal. Le vecteur peut en effet être considéré comme un signal 1D dont on cherche à détecter les sauts–ou les arêtes. Ainsi, nous procédons à la détection des clusters en appliquant des outils de traitement du signal, et notamment en effectuant un produit de convolution entre notre vecteur courant et un filtre spécifique. Les pics dans la convolution correspondent aux sauts dans le signal. Ces outils ont l'avantage de ne pas nécessiter la connaissance du nombre de clusters a priori.

Nous avons choisi d'utiliser le filtre de Canny [Can86], largement utilisé pour détecter des arêtes en traitement du signal et de l'image. Pour optimiser la détection des arêtes ce filtre combine trois critères : une bonne détection, une bonne localisation et la contrainte qu'une arête corresponde effectivement à un pic dans une convolution appropriée. Afin de satisfaire ces critères le filtre de Canny utilise un opérateur construit à partir du produit de convolution de la sortie d'un rapport signal/bruit (SNR) et d'une fonction de localisation (le filtre). La fonction de localisation optimale est la première dérivée du noyau Gaussien [Can86].

Nous avons adapté l'implémentation de la détection des arêtes afin de la rendre optimale pour notre application. Lors de l'utilisation du filtre, un paramètre nommé taille de la fenêtre glissante doit être fixé. Il détermine la raideur du filtre et l'étroitesse de son support. Comme montré dans la Figure 3.1, une faible valeur résulte en un filtre raide à support étroit qui va générer de nombreux pics dans la convolution, tandis qu'une grande valeur donne un filtre lisse à support large qui détectera moins de sauts.

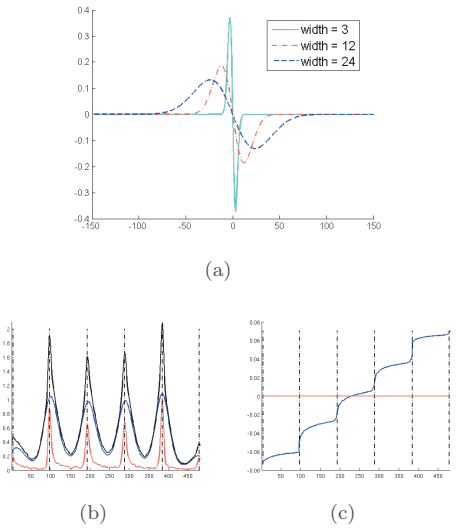

FIGURE 3.1 – (a) Le filtre pour différentes tailles de fenêtre glissante. (b) et (c) Détection des paliers.

Pour éviter les effets indésirables de la taille de la fenêtre, la détection des paliers est réalisée avec deux tailles de fenêtre différentes, une grande et une petite par rapport à la taille du vecteur. Nous travaillons ensuite sur la somme des convolutions afin de détecter les pics principaux et de s'assurer que nos sauts sont à la

fois nets et de taille suffisante.

La Figure 3.1(b) montre un exemple de l'effet de la fenêtre glissante sur la détection des sauts, via la matrice 480×480 `rbsb480` issue de la collection SuiteSparse [DH11]. Sur cette figure, il est clair que les sauts du vecteur sont mieux identifiés en utilisant la somme des convolutions. Dans la Figure 3.1(c), le vecteur singulier gauche est affiché afin d'indiquer la qualité des sauts détectés par les filtres. A contrario, la ligne rouge horizontale de la Figure 3.1(c) indique la bi-partition suggérée par le vecteur de Fiedler : en imposant d'avoir précisément deux clusters la séparation obtenue ne correspond finalement à aucun des paliers du vecteur.

4 SVD projetée

Si l'on suppose que $\mathbf{P}$ possède k blocs, alors les k vecteurs singuliers dominants devraient quasiment couvrir le sous-espace défini par les $\mathbf{v}_i$ décrits Section 2. En réalité, on peut trouver de nombreux blocs avec un unique couple de vecteurs singuliers. Mais on peut aussi itérer notre processus en utilisant une information complémentaire permettant d'affiner notre clustering. Chaque itération va nécessiter une nouvelle information spectrale, sinon nous redécouvrirons les mêmes blocs en boucle. Pour s'assurer que les nouveaux vecteurs apportent une information complémentaire nous travaillons avec les vecteurs propres dominants des équations normales de $\mathbf{P}$ projetées dans le sous-espace orthogonal à celui défini par les blocs déjà détectés. A la première itération, la projection s'effectue contre $Span(\mathbf{e})$. En effet, $\mathbf{e}$ est vecteur propre dominant de toute matrice doublement stochastique. Il sera renvoyé comme vecteur propre dominant de $\mathbf{P}\mathbf{P}^T$ et $\mathbf{P}^T\mathbf{P}$ si l'on ne prend pas garde à l'éviter, auquel cas une itération sera allouée à l'analyse d'un vecteur ne fournissant aucune information sur la structure par blocs de la matrice. Après la première itération, $\mathbf{e}$ appartient implicitement à l'espace des blocs identifiés.

La Figure 4.1 présente un exemple de ce procédé sur une matrice stochastique symétrique $\mathbf{P}$ ayant 4 blocs diagonaux. En utilisant le vecteur propre dominant de $\mathbf{P}$ orthogonal à $\mathbf{e}$ on détecte 3 blocs. Les deux blocs ayant les densités internes les plus faibles ne sont pas séparés (illustré Figure 4.1(b) avec le vecteur, dans son ordre naturel et dans l'ordre croissant de ses entrées). On projette ensuite $\mathbf{P}$ dans le complémentaire orthogonal de ces trois blocs. Le vecteur propre dominant de cette projection est affiché Figure 4.1(c). Il permet de séparer sans ambiguïté les blocs restés ensemble à l'itération précédente. Dans la Section 5 nous décrirons comment fusionner ces deux structures afin de découvrir la structure par blocs complète.

5 Amélioration des clusters

Nous expliquons maintenant comment combiner les différentes partitions en blocs de lignes ou de colonnes trouvées pour chaque vecteur analysé individuellement, ainsi que la superposition de la partition en blocs de lignes et celle en blocs de colonnes une fois le processus itératif terminé afin d'obtenir un clustering des noeuds du graphe. Nous présentons ensuite une mesure adaptée à l'évaluation de la qualité de nos partitions, qui permet aussi de fusionner certains petits blocs. Grâce à cette mesure, nous développons un critère d'arrêt pour déterminer la convergence du processus itératif.

Tout au long de cette section, nous illustrons nos propos avec la matrice `rbsb480` de la collection SuiteSparse [DH11] car elle possède une intéressante structure par blocs relativement fine sur ses lignes et ses colonnes. Nous remarquons que dans les Figures 5.1, 5.2 et 5.3, nous ne présentons pas des clusterings à proprement parler, mais des superpositions de partitions indépen-

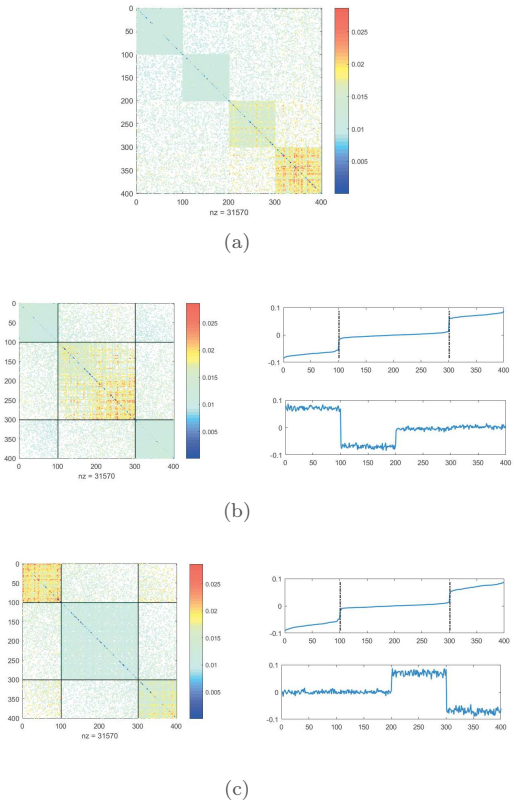

FIGURE 4.1 – Récupération d'une information additionnelle via les vecteurs propres.

dantes de lignes et de colonnes.

5.1 Superposition des partitions. Chaque fois que notre algorithme analyse un vecteur singulier droit (respectivement gauche) de la matrice, il trouve une partition des lignes (respectivement des colonnes). Evidemment, la partition peut être différente pour chacun des vecteurs analysés. Il est donc nécessaire d'incorporer la combinaison de ces partitions dans l'algorithme.

Ce processus est illustré Figure 5.1. La première étape de détection des blocs identifie 4 blocs de lignes et 5 blocs de colonnes, tandis que le deuxième couple de vecteurs analysés suggère une partition différente avec 3 blocs de lignes et 4 blocs de colonnes. Ces nouveaux blocs sont complémentaires des premiers et la combinaison des partitions résulte en 12 blocs de lignes et 20 blocs de colonnes, ces blocs étant bien séparés dans la matrice, comme le montre le schéma de gauche de la Figure 5.2.

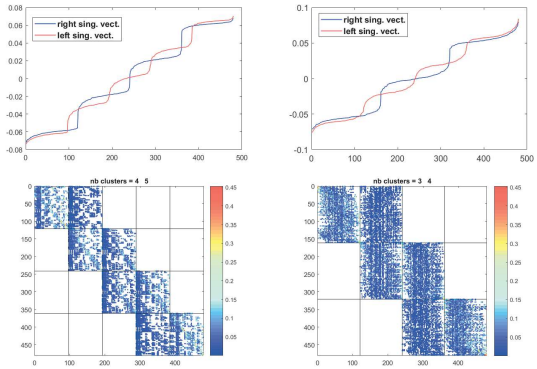

FIGURE 5.1 – Identification de blocs en utilisant différents vecteurs singuliers.

Nos tests sur `rbsb480` montrent l'intérêt d'analyser plusieurs vecteurs à la suite et de superposer les partitions en résultant. Cependant, ce processus peut aussi produire un partitionnement trop fine de la matrice initiale. Ce phénomène est illustré dans le schéma de droite de la Figure 5.2, où la fusion des partitions trouvées après trois processus d'analyse renvoie un découpage trop fin de la matrice (48 par 100 blocs). Cela motive le développement d'une méthode efficace pour fusionner de petits blocs.

5.2 Mesure de qualité. Nous basons l'analyse de nos blocs sur la mesure de modularité définie dans [New04]. La modularité de Newman peut être interprétée comme la somme sur toutes les classes de la différence entre la fraction de liens à l'intérieur de la

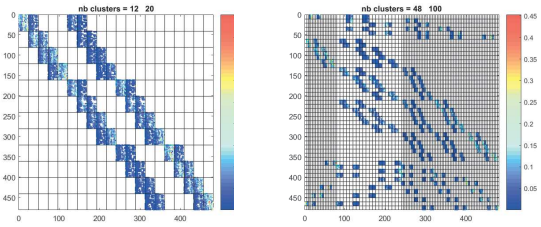

FIGURE 5.2 – Identification récursive des blocs obtenus en analysant plusieurs vecteurs à la suite.

classe et l'espérance de la fraction de ces liens dans un graphe aléatoire possédant la même distribution des degrés.

Dans le cas de notre matrice doublement stochastique $\mathbf{P}$, la partition des lignes peut être évaluée en utilisant la formule suivante :

$$\mathcal{Q}_r = \frac{1}{n} \sum_{k=1}^{r_C} \left(\mathbf{v}_k^T \mathbf{P} \mathbf{P}^T \mathbf{v}_k - \frac{1}{n} |\mathcal{J}_k|^2 \right). \qquad (5.1)$$

Cette équation est directement obtenue de la formulation de la modularité fournie dans [Bag12], en remarquant que $2m = n$ dans le cas doublement stochastique. Ici, n est le nombre de lignes de $\mathbf{P}$ et $|\mathcal{J}_k|$ est le nombre d'éléments du bloc k. De la même façon, on obtient une mesure de qualité $\mathcal{Q}_c$ pour la partition des colonnes en considérant la matrice $\mathbf{P}^T\mathbf{P}$, et en sommant sur l'ensemble des blocs de colonnes.

Il peut être démontré que notre mesure de qualité est comprise entre les bornes suivantes :

$$0 \leq \mathcal{Q}_r \leq 1 - \frac{1}{r_C},$$

et que la borne supérieure ne peut être atteinte que si les r_C blocs ont la même taille.

En dépit des limites bien connues de résolution de la modularité [FB07], cette mesure reste l'une des plus employées dans le domaine de la détection de communautés, et elle est adéquate pour répondre à notre problématique. En outre, dans le cas spécifique des graphes k-réguliers, plusieurs mesures incluant la modularité de Newman se comportent de la même façon [CC13]. Puisque les matrices doublement stochastiques peuvent être considérées comme les matrices d'adjacence de graphes pondérés 1-réguliers, les mesures de qualité communes donnent des résultats strictement équivalents.

Pour éviter l'apparition de mini blocs, nous fusionnons récursivement la paire de blocs dont la fusion augmente le plus la mesure de modularité, ce processus

s'arrêtant naturellement quand quelle que soit la paire de blocs, sa fusion fait décroître la valeur de la mesure.

Ce processus est très utilisé en détection de communautés, domaine dans lequel il est connu sous le nom d'algorithme glouton—voir par exemple [CNM04]—, mais il est généralement initié avec le clustering trivial dans lequel chaque classe contient un unique élément. Cette méthode est peu coûteuse numériquement et s'arrête sur un maximum local, ce qui peut être montré en adaptant la démonstration de [CNM04]. Le processus de fusion empêche ainsi une explosion du nombre de blocs, et il est appliqué après chaque superposition de partitions.

Le processus de fusion est illustré Figure 5.3. Le schéma de gauche montre la partition en 7 par 5 blocs obtenue après fusion de la partition en 12 par 20 blocs du schéma de gauche de la Figure 5.2. La mesure de qualité est alors de 0.5574 pour les lignes et 0.4932 pour les colonnes. Le schéma de droite montre la partition en 7 par 5 blocs obtenue après fusion de la partition en 48 par 100 blocs du schéma de droite de la Figure 5.2. Ici, les mesures de qualité valent 0.5574 pour les lignes et 0.502 pour les colonnes, et sont largement supérieures à celles obtenues pour la partition initiale.

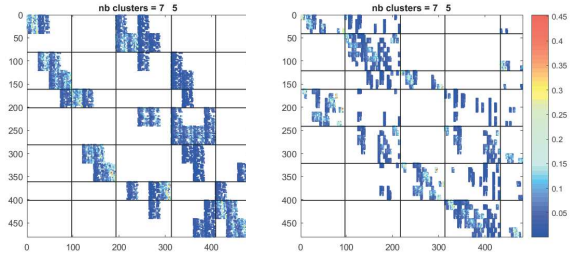

FIGURE 5.3 – Fusion des blocs paire à paire.

Pour arrêter le processus itératif, nous comparons la qualité de la partition mise à jour par le processus de fusion $\mathcal{Q}_r^{upd}$ avec celle de l'itération précédente $\mathcal{Q}_r^{ref}$. Quatre cas peuvent être rencontrés :

- Si $\mathcal{Q}_r^{upd} < \mathcal{Q}_r^{ref}$, on rejette le nouvel itéré.
- Si $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ et le nombre de blocs est supérieur, on accepte l'itéré si le gain de modularité est significatif. Etant donnée la borne supérieure $\mathcal{Q}_r \leq 1 - \frac{1}{r_C}$, nous comparons le gain réel de modularité sur le gain théorique en utilisant le ratio

$$\rho = \frac{\mathcal{Q}_r^{upd} - \mathcal{Q}_r^{ref}}{\frac{1}{nbClust^{ref}} - \frac{1}{nbClust^{upd}}}.$$

  Le seuil d'acceptation peut être ajusté pour contrôler le nombre d'itérations et la finesse de la partition. En pratique, une valeur du seuil entre 0.01 et 0.1 semble efficace.
- Si $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ et le nombre de blocs n'est pas supérieur, on accepte l'itéré.
- Si les partitions sont identiques à une permutation près, la convergence est atteinte.

6 Applications

Nous illustrons les performances de notre algorithme sur la détection de communautés, et le testons ensuite sur des jeux de données pour le clustering issus de la librairie scikit-learn [PVG+11].

6.1 Détection de communautés. Nous testons d'abord notre algorithme sur des réseaux simples. Pour assurer la bi-irréducibilité avec un effet minimal sur la structure, nous forçons les valeurs diagonales initialement nulles à 10^{-8}.

Comme prétraitement, nous supprimons les entrées dominantes (> 0.55) de la matrice après équilibrage, en considérant qu'il s'agit de blocs préalablement détectés pendant le processus décrit dans la Section 4. Sinon, ces entrées impliquent des vecteurs propres dominants quasiment canoniques dont les sauts seront difficiles à détecter. Puisque ces éléments ne sont pas réellement des communautés—il s'agit de hubs ou de noeuds pendants—, nous réincorporons ces noeuds aux communautés en utilisant notre processus de fusion des blocs une fois la convergence de l'algorithme atteinte.

Nous avons comparé notre algorithme (US) avec quatre algorithmes de détection de communautés reconnus, à savoir Walktrap (WT) [PL05], Louvain (ML) [BGLL08], Fast and Greedy (FG) [CNM04], et Leading Eigenvector (LE) [New06], en utilisant l'implémentation de la librairie `igraph` [CN06]. WT et ML sont conçus pour travailler directement sur la structure du graphe, tandis que LE se concentre sur l'exploitation de l'information spectrale. FG est l'algorithme glouton mentionné dans la Section 5.2.

Nous avons appliqué ces algorithmes sur quatre ensembles de réseaux de 500 noeuds, générés avec le banc d'essai de Lancichinetti–Fortunato–Radicchi (LFR) [LFR08], qui sont largement utilisés dans le domaine de la détection de communautés pour ses caractéristiques proches de celles des réseaux issus du monde réel. Chaque ensemble de réseaux correspond à un degré moyen différent $\overline{k}$. Les détails sur la génération des graphes sont donnés Table 6.1. Nous appelons "exp. db des degrés" l'exposant de la loi de puissance utilisée pour générer la distribution des degrés, et "exp. db des tailles de communautés" l'exposant de la loi de puissance utilisée pour générer la distribution des tailles

TABLE 6.1 – Paramètres pour générer les réseaux.

Paramètre	Valeur
nombre de noeuds	500
degré moyen $\overline{k}$	{10,20,40,75}
degré maximum	$2 \times \overline{k}$
exp. db des degrés	-2
exp. db des tailles de communautés	-1

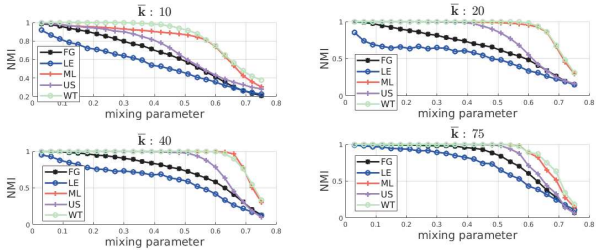

FIGURE 6.1 – Courbes de NMI pour FG, LE, ML, WT, US.

des communautés.

Nous testons le comportement des algorithmes en utilisant le paramètre de mélange qui quantifie la force d'une structure en communautés. A l'origine, ce paramètre mesure la force d'appartenance d'un noeud à une communauté via le ratio entre ses liens à l'extérieur de la communauté et son degré. Plus ce paramètre est grand pour chaque noeud, plus la structure est faible. Le paramètre de mélange du réseau μ est la moyenne des paramètres de mélanges nodaux.

Nous mesurons la précision de la structure trouvée par les algorithmes en utilisant l'information mutuelle normalisée (NMI). Cette mesure vient de la théorie de l'information (voir par exemple [FJ03]). Elle mesure l'écart entre deux partitions candidates qui n'ont pas nécessairement le même nombre de classes. La NMI est très utilisée en détection de communautés [YAT16]. Elle est à valeurs dans $[0, 1]$, valant 1 si les deux partitions sont identiques.

Les résultats sont donnés Figure 6.1. Chaque schéma correspond à la valeur de $\overline{k}$ précisée au-dessus. Pour générer ces courbes, nous avons généré 50 réseaux pour chaque valeur de μ puis calculé la NMI moyenne. Un point d'une courbe correspond donc à un couple (μ,NMI moyenne). Nous pouvons diviser les algorithmes existants en deux groupes : ML et WT, dont la NMI est proche de 1 même pour de grandes valeurs de μ, et FG et LE dont la NMI décroît rapidement. Notre algorithme est entre ces deux groupes.

Nous notons que les comportements de ces deux groupes se rapprochent quand $\overline{k}$ augmente. Ceci peut être expliqué par les contraintes imposées lors de la génération des réseaux qui vont fournir des communautés plus grosses quand $\overline{k}$ est grand, comme montré Figure 6.2. Ici, les matrices d'adjacence de deux réseaux de nos jeux de données sont affichées. Dans ces réseaux, $\mu = 0.3$. Dans le réseau de gauche, $\overline{k} = 10$ avec 40 communautés ; dans celui de droite $\overline{k} = 75$ avec 5 communautés. Les grosses communautés sont généralement mieux détectées par les algorithmes. Cependant, ce facteur n'est pas le seul déterminant pour notre algorithme car sa courbe de NMI est plus sensible à l'accroissement de $\overline{k}$ que FG et LE.

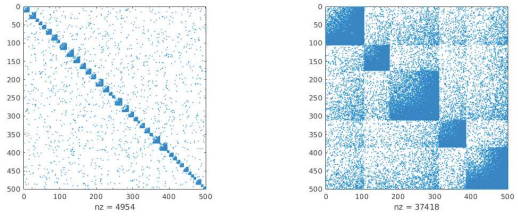

FIGURE 6.2 – Matrices d'adjacence de réseaux.

Nous avons souhaité savoir si ce comportement pouvait être lié au fait que les vecteurs propres dominants de réseaux très creux véhiculent parfois d'autres informations que la structure en communautés [FH16]. Nous avons donc comparé notre algorithme initial (USd) à notre algorithme appliqué sur le réseau perturbé (USw), c'est-à-dire sur $\mathbf{A} + \epsilon \mathbf{e}\mathbf{e}^T$ avec $\mathbf{A}$ la matrice d'adjacence du réseau. Nous avons fixé ϵ à 0.15 comme cela est suggéré pour l'algorithme Pagerank de Google [BP98]. Ces deux versions sont comparées Figure 6.3 pour deux valeurs de $\overline{k}$ différentes. On observe que les deux versions ont un comportement similaire quand $\overline{k} = 0.75$, tandis que USw (courbe bleue) est bien plus précis que USd (courbe rose) pour $\overline{k} = 20$.

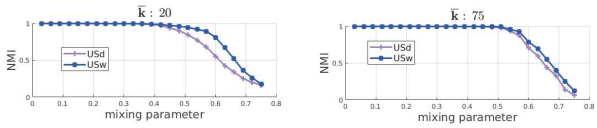

FIGURE 6.3 – Courbes de NMI pour USd et USw.

Pour compléter cette étude, nous avons observé le

comportement de WT, ML, USd et USw pour des valeurs croissantes de $\overline{k}$, comme montré Figure 6.4. ML, WT et USw commencent par accroître leur précision, semblent atteindre un seuil, puis voient leur performance décroître à mesure que $\overline{k}$ augmente. USd augmente d'abord aussi sa précision, mais ses performances ne décroissent pas pour la valeur maximale de $\overline{k}$.

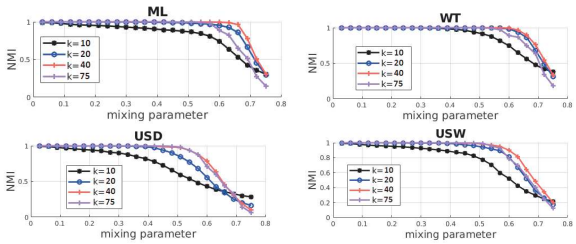

FIGURE 6.4 – Courbes de NMI de ML, WT, USd, USw suivant des valeurs de $\overline{k}$.

Ainsi, bien que notre algorithme ne soit pas nécessairement meilleur que tous les algorithmes de détection communautés existant sur des réseaux très creux, il est capable de mieux détecter les communautés que des algorithmes conçus à cet effet et largement utilisés. En outre, il semble être une alternative très encourageante quand les réseaux deviennent plus denses, puisque ses performances ne décroissent pas dans ces circonstances. Nous verrons d'ailleurs Section 6.2 qu'il obtient d'excellents résultats pour détecter les structures par blocs de matrices d'affinités, qui peuvent être envisagées comme les matrices d'adjacence de graphes pondérés complets.

Enfin, notre algorithme n'est pas contraint à travailler avec des matrices symétriques et fournit donc une méthode polyvalente pour la détection de communautés dans les graphes dirigés, même lorsque ces graphes présentent des flots déséquilibrés—i.e. un déséquilibre entre les arêtes entrantes et sortantes d'une communauté—, alors que WT et ML—tout deux contraints de travailler sur $\mathbf{A} + \mathbf{A}^T$—renvoient des résultats erronés dans ce cas. Un exemple est montré Figure 6.5, où deux communautés sont fortement connectées de façon dissymétrique. Les résultats de ML, WT et USd sont fournis, les lignes noires représentant les séparations entre les communautés. Seul notre algorithme renvoie ici un résultat cohérent.

6.2 Jeux de données pour clustering. Afin de montrer les capacités de notre algorithme sur d'autres tâches de clustering, nous l'avons testé sur les jeux de données de la librairie scikit-learn [PVG+11]. Il

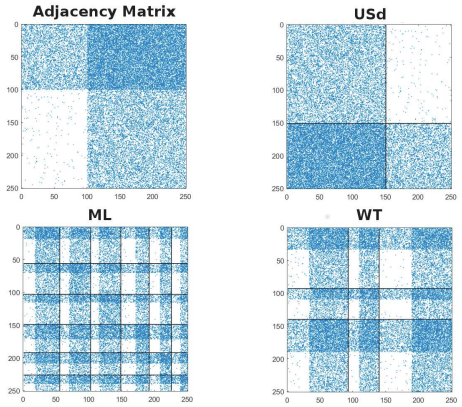

FIGURE 6.5 – Détection de classes dissymétriques.

s'agit pour l'algorithme de détecter des formes cohérentes dans des nuages de 1500 points en 2D à partir de la matrice d'affinité du nuage. Afin de montrer la modulabilité de notre méthode, nous avons mis l'accent sur les différents types de formes à détecter : certains de ces nuages peuvent être séparés linéairement, d'autres non et l'un d'eux est fortement anisotrope. La matrice d'affinité [NJW01] d'un ensemble de points $\{x_i \in \mathbb{R}^p, i = 1...n\}$ est la matrice symétrique définie par

$$\mathbf{A}_{i,j} = \begin{cases} \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}), & \text{pour } i \neq j, \\ 0, & \text{sinon.} \end{cases} \quad (6.1)$$

La précision de notre méthode va fortement dépendre du choix du paramètre Gaussien σ. Nous le fixons en suivant l'heuristique de [MNR08] pour prendre en compte à la fois la densité et les dimensions du nuage.

Puisque la matrice d'affinité est à diagonale nulle, nous forçons la bi-irréducibilité en fixant les éléments diagonaux à 10^{-8}. Comme en Section 6.1 nous supprimons les entrées dominantes de la matrice en guise de pré-traitement. Comme post-traitement, chacune de ces entrées est ensuite ajoutée à son plus proche cluster, c'est-à-dire celui contenant le point qui lui est le plus proche au sens de la distance Euclidienne.

Nous avons comparé notre méthode avec celles proposées dans la librairie scikit-learn. Les résultats pour cinq de ces méthodes sont fournies Figure 6.6. Chaque ligne correspond à un jeu de données spécifique, chaque colonne à une méthode précisée ci-dessus. Deux points d'une même couleur ont été assignés au même cluster. La colonne de droite montre les résultats de notre méthode, que nous avons arrêtée après l'analyse d'un

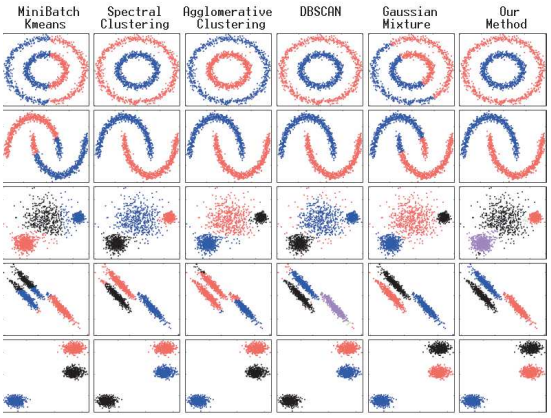

FIGURE 6.6 – Comparaison d'algorithmes de clustering.

TABLE 6.2 – NMI des clusterings Figure 6.6.

M.B. Kmeans	Spect. Clust.	Agglo. Clust.	DB SCAN	Gauss. Mixt.	Our method
$2.9.e^{-4}$	1	0.993	1	$1.3.e^{-6}$	1
0.39	1	1	1	0.401	1
0.813	0.915	0.898	0.664	0.942	0.902
0.608	0.942	0.534	0.955	1	0.996
1	1	1	1	1	1

unique vecteur propre. Etant donnés ces résultats, il est clair que ce vecteur fournit une information suffisante pour séparer grossièrement les nuages en clusters cohérents pour tous les jeux de données quelles que soient leurs formes. La NMI est donnée Table 6.2, en suivant le même ordre ligne/colonne que la Figure 6.6. On voit qu'excepté pour deux jeux de données—le quatrième où un élément a été mal assigné pendant le post-traitement, et le troisième pour lequel la NMI est supérieure à 0.9— notre méthode a toujours le meilleur score. Nous remarquons aussi qu'à part `DBSCAN` (4^{eme} colonne), toutes ces méthodes nécessitent de connaître à l'avance le nombre de classes.

7 Conclusions

Nous avons développé un algorithme spectral capable de partitionner une matrice avec peu de vecteurs singuliers, principalement grâce aux propriétés de l'équilibrage doublement stochastique. Notre méthode fonctionne en outre sans connaissance a priori du nombre de classes, et ne nécessite pas de "symétriser" la matrice artificiellement.

Nous l'avons ensuite utilisée sur des problèmes standards d'analyse de données, où nous l'avons confronté avec des méthodes spécifiquement conçues pour ces problèmes. Notre algorithme s'est avéré compétitif, mais il est en plus polyvalent, car conçu pour être appliqué à des matrices quel que soit ce qu'elles représentent. Pour changer d'application, il suffit donc d'adapter le pré- et le post-traitement.

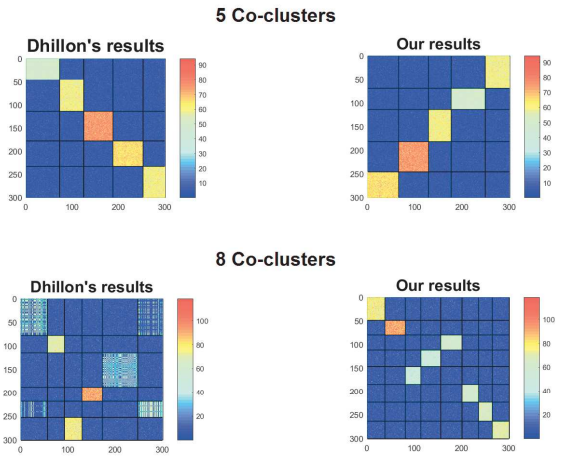

FIGURE 7.1 – Détection de co-clusters.

Le but de nos travaux futurs sera de généraliser notre méthode pour effectuer des tâches de co-clustering. En effet, elle est capable de détecter des structures rectangulaires dans les matrices carrées, comme le montre la Figure 7.1. Ici, deux matrices présentant une structure à blocs rectangulaires ont été générées avec la fonction `make_biclusters` de scikit-learn [PVG+11]. A droite, nous voyons que notre algorithme est capable de détecter sans erreur ces blocs rectangulaires dans les deux matrices, tandis que l'algorithme de Dhillon [Dhi01]–à gauche–échoue à détecter certains des blocs de la matrice du bas.

Ce court exemple fournit des perspectives encourageantes pour le co-clustering. Cependant, les résultats de la Figure 7.1 ne montrent que la superposition de partitions indépendantes sur les lignes et les colonnes, sans exploiter les liens entre ces partitions, alors qu'il est connu ([DMM03, LN15]) que l'imbrication entre les clusters lignes et colonnes est un paramètre important de la qualité d'un co-clustering.

Une étude sur la complexité est en cours, notamment au niveau du passage à l'échelle de l'algorithme. En attendant, une implémentation simple de l'algorithme en Matlab peut être téléchargée sur : *http ://cloud.irit.fr/index.php/s/2S6hZzO16R4JPJI.*

Références

[Bag12] James P Bagrow. Communities and bottlenecks : Trees and treelike networks have high modularity. *Physical Review E*, 85(6) :066118, 2012.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7) :107–117, April 1998.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8 :679–698, 1986.

[CC13] Patricia Conde-Cespedes. Modelisation et extension du formalisme de l'analyse relationnelle mathematique a la modularisation des grands graphes. *PhD Thesis, Universite Pierre et Marie Curie*, 2013.

[CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems :1695, 2006.

[CNM04] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6) :066111, December 2004.

[DH11] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1) :1–14, 2011.

[Dhi01] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA, 2001. ACM.

[DMM03] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic coclustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 89–98, New York, NY, USA, 2003. ACM.

[FB07] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1) :36–41, 2007.

[FH16] Santo Fortunato and Darko Hric. Community detection in networks : A user guide. *CoRR*, abs/1608.00163, 2016.

[FJ03] Ana L. N. Fred and Anil K. Jain. Robust data clustering. In *CVPR (2)*, pages 128–136. IEEE Computer Society, 2003.

[FMSV08] David Fritzsche, Volker Mehrmann, Daniel B. Szyld, and Elena Virnik. An SVD approach to identifying metastable states of Markov chains. *Electronic Transactions on Numerical Analysis*, 29 :46–69, 2008.

[For10] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3) :75–174, 2010.

[Fro12] Georg Frobenius. Ueber matrizen aus nicht negativen elementen. *Sitzungsber. Königl. Preuss. Akad. Wiss*, page 456–477, 1912.

[KR13] Philip A. Knight and Daniel Ruiz. A fast algorithm for matrix balancing. *IMA Journal of Numerical Analysis*, 33(3) :1029–1047, 2013.

[LFR08] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4) :046110, October 2008.

[LN15] L. Labiod and M. Nadif. A unified framework for data visualization and coclustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9) :2194–2199, Sep. 2015.

[LR+15] Jing Lei, Alessandro Rinaldo, et al. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1) :215–237, 2015.

[MNR08] Sandrine Mouysset, Joseph Noailles, and Daniel Ruiz. Using a global parameter for gaussian affinity matrices in spectral clustering. In *VECPAR*, volume 5336 of *Lecture Notes in Computer Science*, pages 378–390. Springer, 2008.

[New04] M.E.J. Newman. Analysis of weighted networks. *Physical Review E*, 70 :056131, 2004.

[New06] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3) :036104, 2006.

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering : Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems : Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA, 2001. MIT Press.

[Per07] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2) :248–263, 1907.

[PL05] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *ArXiv Physics e-prints*, December 2005.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416, 2007.

[YAT16] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6 :30750 EP –, Aug 2016. Article.