Équilibrage bi-stochastique des matrices pour la détection de structures par blocs et applications

Luce Le Gorrec

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Luce LE GORREC

Le 28 octobre 2019

Équilibrage bi-stochastique des matrices pour la détection de structures par blocs et applications

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Daniel RUIZ et Sandrine MOUYSSET

Jury

M. Mario Arioli, Rapporteur
M. Renaud Lambiotte, Rapporteur
M. Iain Duff, Examinateur
M. Ernesto Estrada, Examinateur
Mme Géraldine Morin, Examinatrice
Mme Patricia Conde-Céspedes, Examinatrice
M. Daniel RUIZ, Directeur de thèse
Mme Sandrine Mouysset, Co-directrice de thèse

Remerciements

Tout d'abord, je souhaite remercier de tout mon coeur Sandrine et Daniel, mes directeurs de thèse. Déjà, merci de m'avoir offert l'opportunité de faire une thèse. Merci pour vos conversations passionnantes et passionnées sur des sujets scientifiques divers, du début à la fin de ce doctorat. Merci d'avoir su éveiller ma curiosité, et de m'avoir ainsi donné l'envie d'aller creuser plus loin, de m'ouvrir à de nouvelles perspectives. Merci pour vos relectures attentives, vos réponses éclairantes, votre soutien sans faille, votre gentillesse et votre patience dans mes (nombreux) moments de doute sur le sens de la vie. Merci de m'avoir appris un métier. Et plus généralement, merci pour tout ce que vous m'avez apporté, scientifiquement et humainement. Très honnêtement, je doute qu'il existe meilleur duo d'encadrants au monde, et je m'estime très chanceuse d'avoir mené cette thèse sous votre supervision toujours bienveillante. Vous êtes deux personnes magnifiques.

Merci à Ms Arioli et Lambiotte de m'avoir fait l'honneur de rapporter cette thèse. Merci pour leurs commentaires constructifs et leur expertise. Cela m'a permis d'aborder certains aspects sous un nouvel angle. Je suis honorée que vous considériez mes travaux dignes d'un travail de doctorat. Merci aussi d'avoir accepté de faire parti de mon jury de soutenance.

Thanks to the jury members for having accepted our invitation for examining this defence : Ernesto Estrada, Iain Duff, Géraldine Morin, Mario Arioli, Renaud Lambiotte and Patricia Conde-Céspedes. Thanks for your time.

Merci à toutes les personnes qui ont pris une part active à ces travaux. Un grand merci à Stergos Afantenos, pour m'avoir ouvert la porte du traitement automatique du langage naturel, pour ta disponibilité et nos conversations enrichissantes, ainsi que pour ta gentille lettre de recommandation. Merci à Sukru Torun, for your help with graph partitioning and some softwares that I definitely would never have been able to use without you. Thanks also for your constant good mood at work, that has clearly helped. Merci à Nicolas Urien, pour ton excellent travail de stage. Un stage de trois mois c'est court, mais le travail que tu as abattu dans cette durée est impressionnant. J'espère que tout se passe bien à l'ENSAE, et si un jour tu as besoin d'une recommandation, ou si

tu veux revenir vers la recherche, n'hésite pas à me (nous) contacter. A huge thanks to Iain and Phil, for accepting me as a new worker on the research project you had initiated with my advisors. Thanks a lot for having given me the possibility to explore different research orientations, and for your valuable advices (in maths, and also in English). Iain, thanks a lot for hosting me at your very nice home at East Hagbourne. Phil, thanks for the opportunity you provide to me to keep working with you, and thanks a lot for putting your trust in me for this wonderful postdoctoral research project that I can't wait to start. Thanks also to you and to the researchers from the marine team and math department at Strathclyde for the exciting two months I had spent there in June-July 2017. Merci en particulier à Emma Dolmaire, pour son amitié durant ce séjour.

Merci à ceux qui m'ont aidée ou aiguillée de façon ponctuelle au cours de ces trois ans : merci à Jean-François Marcotorchino pour son retour sur mes travaux sur la modularité, thanks to Yao Zhu, for having nicely answered to my questions about his work. Merci à Philippe Leleux pour l'aide informatique du 28, matin ! :)

Merci à mon entourage au travail : à l'équipe APO dans son intégralité, pour leur accueil, les échanges réguliers et très instructifs sur des sujets variés, notamment en pause déjeuner. Un merci particulier à Ronan, Ehouarn, et Sandrine pour leur aide avec les cours, et à Ehouarn et Daniel pour les bières (je vous en dois encore un certain nombre, j'essaierai de faire un pot de thèse en conséquence). Merci à l'équipe ALGO du CERFACS, pour leur accueil, et aussi pour l'organisation du Burn's Supper. Un big up particulier à mes co-doctorants Philippe Leleux et Pierre Matalon. Merci à tout le staff administratif de l'ENSEEIHT, en particulier à Sylvie Armengaud Metche, Muriel Pernier, Muriel de Guibert et Elisabeth Rey pour leur aide, leur efficacité, leur disponibilité, leur gentillesse et leur bonne humeur. Merci au service informatique de l'ENSEEIHT pour leurs réponses patiente à mes questions (souvent idiotes). Merci à Agnès Requis et Martine Labruyère, de l'EDMITT, pour leur réactivité et leur aide sur de nombreux aspects administratifs.

Merci à toutes les personnes qui ont constitué ma vie sociale ces trois dernières années. En particulier, merci à Jim pour l'escalade et la franche camaraderie (et aussi la guitare). Merci à Philou et Camille, pour les quizz, les soirées, et aussi pour Chouchen. Merci à Quentin (et à Ollivier) pour les soirées bières, et pour avoir veillé sur Auguste à Clermont. Pour avoir gentiment assisté à la soutenance et pour avoir été d'une grande aide dans la préparation du pot. Merci à Piv et Irénée, pour notre méca flux quasi hebdomadaires, l'ambiance celtico-espagnole, le chat-sitting, pour nous avoir appris le palet breton (et l'aviron...) et nous avoir fait goûter à la cuisine typique espagnole. On espère vous revoir très vite, à Madrid et à Glasgow. Merci à vous trois pour votre aide précieuse et appréciée avec les quiches, les cookies et les gougères. Merci à Aloïse pour les sorties escalades, merci

pour tes conseils sortie culturel, et pour tout ce que tu m'as appris sur les musées et l'art. J'ai adoré discuter et aller grimper avec toi, et je regrette que nous ayons attendu aussi longtemps pour vraiment nous rencontrer, car j'aurais aimé avoir plus de temps pour apprendre de toi. Merci à Céline, pour son amitié de longue date, nos visions politiques communes et complémentaires. Merci pour nos conversations citoyennes, et pour tes interrogations politiques et philosophiques, toujours très pertinentes, et qui font réfléchir. J'ai toujours adoré ça chez toi. Merci aussi pour m'avoir gentiment accompagnée pogoter sur la Ruda ! Et aussi pour me tenir informée de ce qui se passe dans les cercles d'anciens arsonvaliens. Et enfin, merci Ivanne pour tout ce que tu m'as apporté et appris (l'escalade, le russe, et le cinéma roumain, entre autre), merci pour ta rigueur morale (qui fait du bien en ces temps troublés), merci d'être toujours et tellement toi. Tu es une personne magnifique et j'espère que tu auras beaucoup de bonheur quelque soit le continent où tu te trouves, et que tu viendras me voir à Glasgow.

Merci de tout coeur à ma famille. En particulier à mes parents et à mes frères et soeurs. Merci à ma mère, pour ton réconfort et ta vivacité d'esprit, ton enthousiasme et ton énergie, qui te permettent de mener à bien un nombre extraordinaire de projets. Merci à mon père, pour m'avoir encouragée à m'orienter vers les mathématiques, pour ta ténacité au travail qui me sert de modèle au quotidien, ta rigueur et ta droiture, les valeurs que tu m'as inculquées, ton humour et tes bons mots (les spontanés sont les meilleurs). Merci à vous deux pour votre soutien tout au long de mes études, et d'un point de vue plus court terme, merci pour votre aide et votre disponibilité dans la préparation de ma soutenance et de ce gargantuesque pot. Merci à Tangui pour ta réussite (je suis ta plus grande fan p***** b***), nos échanges et notre proximité pour la vie ! Merci à Quentin pour tes conversations enrichissantes sur de nombreux sujets, pour tes passions et ton enthousiasme. Merci énormément pour ta présence et ton accueil. Merci à vous deux (et à Lia et Solea) d'être venus partager ce moment avec nous ! Merci à Elise pour ta protection, ta combativité exemplaire, et tes gentilles attentions. Merci à vous tous pour votre amour et votre soutien inconditionnels, pour ne m'avoir jamais laissé tomber, et parce que vous avez toujours su me dire ce que j'avais besoin d'entendre. J'espère que vous êtes fiers de moi comme je suis fière de chacun d'entre vous. Je vous aime tous énormément.

Enfin, merci de tout mon coeur à Auguste, de me supporter encore (et toujours) après 6 ans (déjà !) côte-à-côte. Merci pour ton soutien, ta douceur, ton amour, et tes attentions. Merci de venir avec moi à Glasgow. On sera toujours plus forts tous les deux. Je t'aime.

Table des matières

Table des figures

Liste des tableaux

Introduction

La détection de structure par blocs dans les matrices est un point-clé dans plusieurs domaines des mathématiques appliquées et de l'informatique. Dans le domaine de la détection de communautés par exemple, étant donné un graphe, issu par exemple d'un réseau social, l'enjeu est de trouver des groupes d'individus fortement connectés [7, 8]. La détection de communautés peut se définir comme un problème de permutations symétriques des lignes et des colonnes de la matrice d'adjacence du graphe étudié afin de faire apparaître une structure par blocs diagonaux, comme le montre la Figure 1. Le petit exemple de réseau social de cette figure présente deux communautés : *Justine, Manon, Théo* d'une part et *Tangui, Quentin, Elise, Anna* d'autre part. Un unique lien existe entre ces deux communautés. En organisant la matrice d'adjacence de sorte que les lignes et les colonnes correspondant à *Manon*, *Justine* et *Théo* soient adjacentes, de même pour *Tangui*, *Quentin*, *Anna* et *Elise*, on fait apparaître une structure par blocs diagonaux dans cette matrice.

Plus généralement, la détection de structures par blocs présente un intérêt pour le domaine de l'analyse de données, plus précisément pour la classification non supervisée, dès lors que les données peuvent être représentées matriciellement.

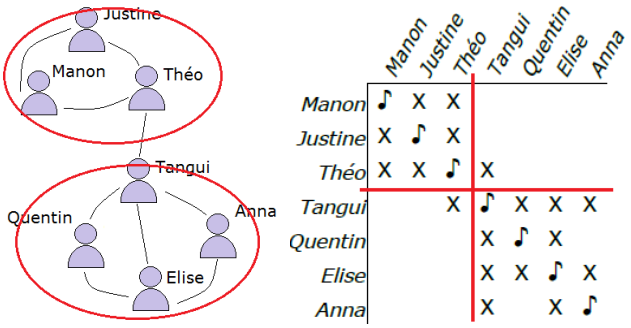


Figure 1 : *Illustration d'un petit réseau social à gauche, et de sa matrice d'adjacence à droite.*

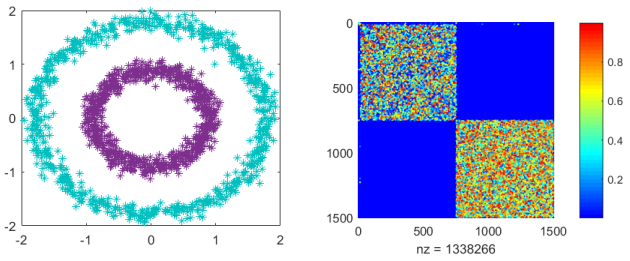


FIGURE 2 : *Un nuage de points en 2 dimensions à gauche, et sa matrice d'affinité gaussienne à droite.*

En représentant les interactions entre ces données via leur affinité ou leur similarité [9–11], on obtient une matrice carrée, dont on veut permuter symétriquement les lignes et les colonnes afin de mettre en évidence une structure diagonale par blocs. Le cas présenté Figure 1 en est un cas particulier, dans lequel les interactions entre les données sont binaires : elles existent ou n'existent pas. Le problème repose alors uniquement sur la structure creuse de la matrice, le but étant de trouver des blocs diagonaux plus denses, tandis que les blocs non diagonaux sont plus creux. Dans d'autres cas, la structure recherchée peut ne dépendre que des valeurs numériques, comme cela est présenté en Figure 2. Sur la gauche de cette figure, on montre un nuage de points en 2 dimensions formant deux cercles distincts. La matrice contenant les affinités gaussiennes entre chaque couple de points est affichée à droite – l'affinité gaussienne entre deux points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ étant définie par $exp\left(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\right)$, avec σ un paramètre à fixer, et $||.||$ la distance euclidienne. En organisant cette matrice de sorte que les lignes et les colonnes associés aux points formant le cercle intérieur correspondent aux plus petits indices, tandis que les lignes et colonnes associés aux points du cercle extérieur correspondent aux plus grands indices, on fait apparaître une structure diagonale par blocs. Les blocs diagonaux sont ici numériquement dominants, et les blocs non diagonaux ont des valeurs numériques très faibles.

Toujours dans le domaine de l'analyse de données, il existe un autre type de représentation des données sous forme matricielle qui peut conduire à la détection d'un autre type de structures par blocs. Il s'agit des tables de données, dans lesquelles un ensemble de N données $\left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N} \in \mathbb{R}^p$, caractérisées par p paramètres, est représenté par la matrice

$$\mathbf{X} = \begin{bmatrix} {\mathbf{x}^{(1)}}^T \\ \vdots \\ {\mathbf{x}^{(N)}}^T \end{bmatrix}$$

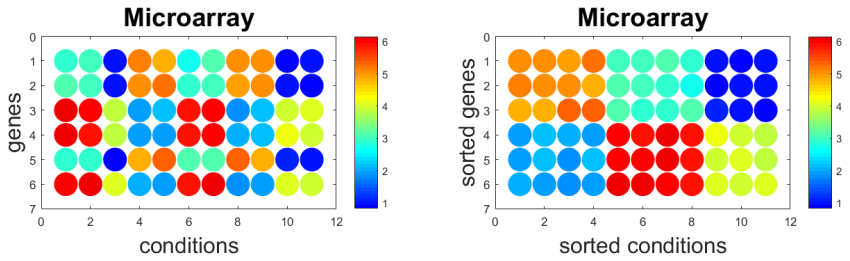


FIGURE 3 : *Détection d'une structure en échiquier sur une puce ADN. A gauche, la matrice dans sa forme initiale. A droite, la matrice après permutations de ses lignes et de ses colonnes faisant apparaître sa structure par blocs.*

dans laquelle l'entrée (i, j) correspond à la valeur du j^{eme} paramètre de la i^{eme} donnée. Pour ce type de matrices, on souhaite trouver des blocs de lignes et des blocs de colonnes qui soient fortement corrélés intra-bloc, et plus faiblement inter-blocs. En supposant, par exemple, que l'on peut modéliser les entrées de cette matrice par les réalisations d'une loi de probabilité jointe sur les données et les paramètres, le but est alors de trouver une partition des lignes et des colonnes de la matrice qui maximise l'information mutuelle entre la distribution initiale des entrées et celle des entrées regroupées en classes [12, 13]. Pour ce type de matrices, on cherche à obtenir une structure dite "en échiquier". Un exemple inspiré de [14] est donné Figure 3. La matrice représente une puce ADN : chaque ligne correspond à un gène, dont on a mesuré l'expression sous un certain nombre de conditions expérimentales – représentées par les colonnes. En réorganisant les lignes et les colonnes pour regrouper les entrées dont les valeurs numériques sont proches, on fait apparaître la structure en échiquier de la matrice. Chaque bloc correspond alors à une co-classe, dans laquelle le groupe de gènes réagit de la même façon pour le groupe de conditions expérimentales.

Un autre domaine dans lequel la détection d'une structure par blocs joue un rôle crucial est le calcul haute performance. Par exemple, on peut vouloir appliquer divers traitements à un très grand graphe [15], ou accélérer la résolution d'un système linéaire [16]. Dans tous les cas, on découpe la matrice en sous-blocs. Si l'on est intéressé par le calcul parallèle, chaque bloc pourra être traité par un ordinateur ou un processus indépendant. On veut généralement limiter le nombre de communications entre les processus, et que tous ces processus aient à réaliser un travail de complexité équivalente, pour éviter que des processus ayant terminé leur travail attendent trop longtemps la réponse d'un processus encore en traitement [17]. On peut, par exemple, rechercher une structure diagonale par

blocs dans laquelle les blocs se superposent [18] ou non [16], ou une structure par blocs de lignes [19, 20]. En général, le partitionnement en blocs de la matrice se fait sur la structure creuse de la-dite matrice – c'est-à-dire que l'on considère de façon équivalente toutes les valeurs numériques non nulles –, le but étant de trouver des blocs de taille équivalente ou possédant un nombre équivalent d'entrées non nulles, tout en minimisant les interactions entre ces blocs. Cependant, dans le domaine de la résolution de systèmes linéaires, des travaux récents mettent en évidence que le fait de ne s'intéresser qu'à la structure creuse de la matrice dans la détection de sa structure par blocs peut pénaliser l'efficacité du solveur, et qu'un compromis doit être trouvé entre le nombre de connexions entre les blocs et les valeurs numériques de ces connexions [16, 21].

L'enjeu principal de cette thèse est d'observer l'impact de l'équilibrage bi-stochastique – aussi appelé équilibrage doublement stochastique – sur la détection de structures par blocs dans les matrices. L'équilibrage bi-stochastique d'une matrice $\mathbf{A} \in \mathbb{R}_+^{n \times n}$ revient à trouver deux vecteurs $\mathbf{r}, \mathbf{c} \in {\mathbb{R}_+^*}^n$ tels que :

$$\begin{cases} \mathbf{P}\mathbf{e} = \mathcal{D}(\mathbf{r})\mathbf{A}\mathcal{D}(\mathbf{c})\mathbf{e} = \mathbf{e} \\ \mathbf{P}^T\mathbf{e} = \mathcal{D}(\mathbf{c})\mathbf{A}^T\mathcal{D}(\mathbf{r})\mathbf{e} = \mathbf{e} \end{cases} \tag{1}$$

où $\mathbf{e} \in \mathbb{R}^n$ est le vecteur composé de 1 et $\mathcal{D}(\mathbf{x})$ est une matrice diagonale dont l'entrée (i, j) vaut la i^{eme} coordonnée du vecteur $\mathbf{x}$ si $i = j$, et 0 sinon. L'équilibrage bi-stochastique a été utilisé dans de nombreux domaines pour des applications diverses, depuis le début des années 1930. Une revue détaillée sur l'équilibrage doublement stochastique, son historique en fonction des différents domaines d'application, ainsi que sa généralisation à d'autres types d'objets que les matrices, peut être trouvée dans [22]. Par exemple, il a servi à étudier la connectivité des graphes dirigés grâce à sa capacité à mettre en évidence les arcs critiques [23]. Slater s'en est servi comme première étape d'un algorithme de classification hiérarchique pour l'analyse de flux migratoires [24], à cause de sa capacité à équilibrer les valeurs numériques d'une matrice, tout en conservant les mesures d'association de ces valeurs [25]. En outre, les vecteurs $\mathbf{r}$ et $\mathbf{c}$ peuvent aussi être utilisés pour des problématiques de classement – typiquement, classer des pages web si la matrice $\mathbf{A}$ représente la matrice d'adjacence de ces pages [26]. Par ailleurs, il a été montré que les systèmes linéaires sont généralement mieux conditionnés après application d'un équilibrage doublement stochastique [27]. L'équilibrage doublement stochastique est en outre envisagé dans [25] comme un sous-problème de la découverte de sommes marginales dans les matrices non négatives, problématique bien connue en économie et en statistique.

Nous avons pensé que cet équilibrage pourrait améliorer la détection des structures par blocs pour trois raisons. Les deux premières viennent d'une observation sur la limitation

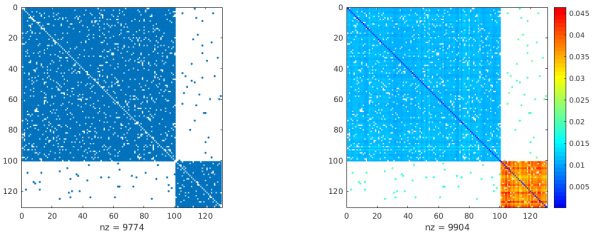


FIGURE 4 : *Impact de l'équilibrage doublement stochastique sur les petites communautés. A gauche, la matrice d'adjacence du graphe. A droite, l'équilibrage doublement stochastique de la matrice de gauche.*

des algorithmes de détection de communautés existants. D'abord, pour les algorithmes classiques de détection de communautés, il est généralement compliqué de détecter précisément les communautés dans un graphe lorsque les tailles des communautés sont hétérogènes [28]. Cela peut s'illustrer de la façon suivante : considérons le cas d'un graphe $G = (V, E)$, avec V l'ensemble des noeuds de G, et E l'ensemble des arêtes, que nous supposons non dirigées. Supposons que nous ayons deux communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ disjointes – autrement dit, deux composantes connexes – de taille respective $n_1 > n_2$, et que la densité de liens, que nous notons p_{in}, soit la même dans chaque communauté :

$$p_{in} = \frac{1}{n_1^2} \left|\{(i,j) \in E : i,j \in \mathcal{C}_1\}\right| = \frac{1}{n_2^2} \left|\{(i,j) \in E : i,j \in \mathcal{C}_2\}\right|.$$

Alors, en moyenne, un noeud dans $\mathcal{C}_1$ partagera plus de liens avec les noeuds de sa communauté qu'un noeud dans $\mathcal{C}_2$ ($n_1 \times p_{in} > n_2 \times p_{in}$). Il sera donc plus compliqué pour un algorithme de détecter la petite communauté. Or, dans le graphe associé à la matrice obtenue après équilibrage bi-stochastique de la matrice d'adjacence du graphe G, ces deux valeurs seront strictement égales et vaudront 1. En effet, l'équilibrage favorise les valeurs numériques du petit bloc par rapport à celle du grand bloc. Cette observation reste vraie dans le cas où les deux composantes sont faiblement connectées. Un exemple est donné Figure 4. Sur cette figure, on voit que les valeurs numériques dans le petit bloc sont favorisées par rapport à celle du grand bloc après équilibrage. En outre, le poids moyen des liens que partagent les noeuds à l'intérieur de leur communauté respective est harmonisé. En effet, en conservant les notations de l'équation (1), on a pour ces matrices

$$\begin{cases} \frac{1}{n_1} \sum_{i \in \mathcal{C}_1} \sum_{j \in \mathcal{C}_1} \mathbf{A}(i,j) = 89 \\ \frac{1}{n_2} \sum_{i \in \mathcal{C}_2} \sum_{j \in \mathcal{C}_2} \mathbf{A}(i,j) = 26.5 \end{cases} \quad \text{et} \quad \begin{cases} \frac{1}{n_1} \sum_{i \in \mathcal{C}_1} \sum_{j \in \mathcal{C}_1} \mathbf{P}(i,j) = 0.99 \\ \frac{1}{n_2} \sum_{i \in \mathcal{C}_2} \sum_{j \in \mathcal{C}_2} \mathbf{P}(i,j) = 0.98 \end{cases}$$

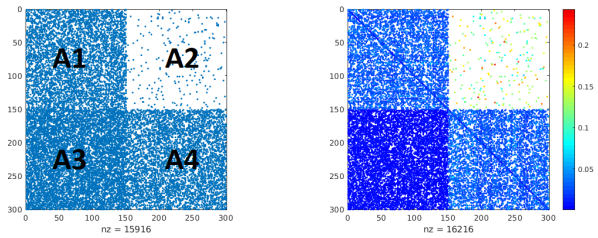


FIGURE 5 : *Impact de l'équilibrage doublement stochastique sur les communautés avec flot. A gauche, la matrice d'adjacence du graphe. A droite, l'équilibrage doublement stochastique de la matrice de gauche.*

Un second problème que l'on rencontre lors la détection de communautés intervient dans le cas les graphes dirigés. En effet, certains algorithmes [2, 29], très efficaces sur les graphes non dirigés, se généralisent naturellement aux graphes dirigés en ne considérant pas la direction des arêtes. Du point de vue de la matrice d'adjacence, cela revient à considérer $\mathbf{A} + \mathbf{A}^T$ au lieu de $\mathbf{A}$. Cependant, cette astuce ne peut fonctionner que dans le cas où les directions des arêtes sont à peu près équilibrées. Dans le cas d'un graphe avec flot – c'est-à-dire avec un fort déséquilibre entre les arêtes entrantes et sortantes relativement à une communauté –, ces algorithmes ainsi généralisés échoueront à détecter la structure par blocs dans la matrice d'adjacence. Cela est lié au fait que, dans cette configuration, bien que la matrice $\mathbf{A}$ possède une structure par blocs, cette structure est estompée, voire inexistante, dans la matrice $\mathbf{A}+\mathbf{A}^T$, comme on peut facilement l'envisager au regard de la matrice de gauche de la Figure 5. Dans ce cas, appliquer un équilibrage bi-stochastique à la matrice $\mathbf{A}$ comme préambule à l'application des algorithmes sur $\mathbf{A}+\mathbf{A}^T$ nous semble tout indiqué. En effet, l'équilibrage bi-stochastique peut avoir comme effet induit de "gommer" un flot existant. Considérons par exemple la matrice $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Pour cette matrice, il n'existe pas d'équilibrage doublement stochastique. Cependant, dans une telle configuration, les algorithmes d'équilibrage vont faire tendre les vecteurs $\mathbf{r}$ et $\mathbf{c}$ vers $(0, +\infty)^T$ et $(+\infty, 0)^T$ respectivement, de sorte que $\mathbf{P} \to \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, tendant ainsi à faire disparaître l'élément non diagonal de $\mathbf{A}$ [26, 30]. Dans le cas du graphe avec flot montré Figure 5, en reprenant les notations de l'équation (1), on a :

$$\begin{cases} \frac{1}{n} nnz(\mathbf{A}_1) = \frac{1}{n} nnz(\mathbf{A}_4) = 30 \\ \frac{1}{n} nnz(\mathbf{A}_2) = 1.5 \\ \frac{1}{n} nnz(\mathbf{A}_3) = 45 \end{cases} \quad \text{et} \quad \begin{cases} \frac{1}{n} \sum_{i,j} \mathbf{P}_1(i,j) = \frac{1}{n} \sum_{i,j} \mathbf{P}_4(i,j) = 0.8 \\ \frac{1}{n} \sum_{i,j} \mathbf{P}_2(i,j) = \frac{1}{n} \sum_{i,j} \mathbf{P}_3(i,j) = 0.2 \end{cases}$$

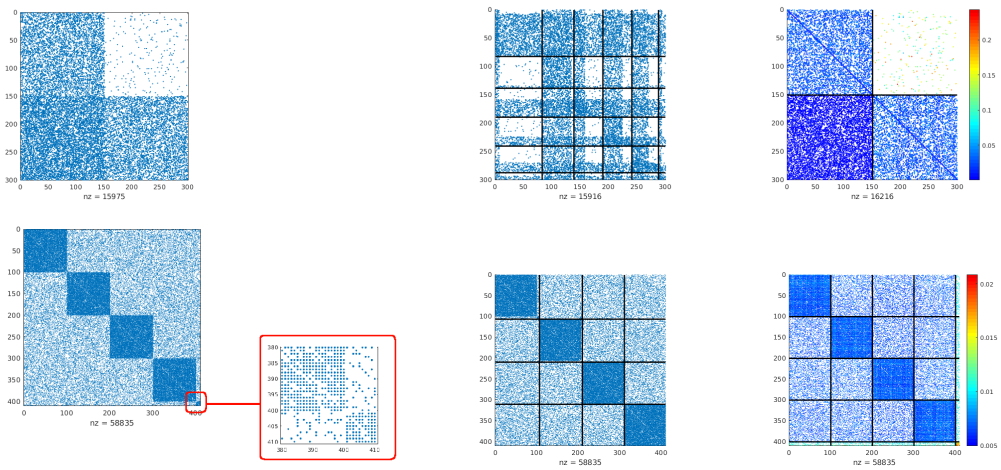
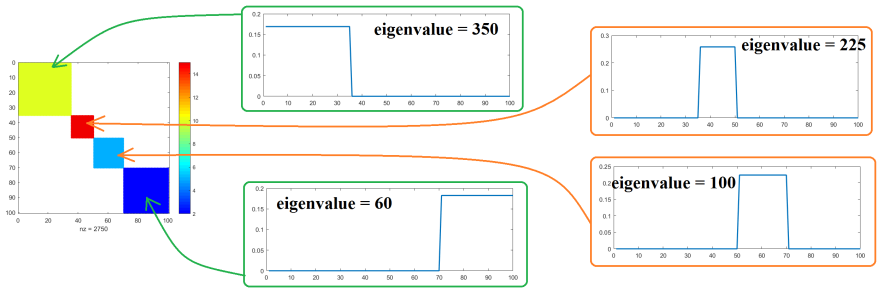


FIGURE 6 : *Application de l'algorithme de Louvain à deux matrices d'adjacence. Les matrices d'adjacence sont affichées à droite. Les images du milieu correspondent à la structure en communautés identifiée par l'algorithme appliqué à ces matrices d'adjacence. Les images de droite correspondent à la structure en communautés identifiée par l'algorithme appliqué aux équilibrages doublement stochastiques des matrices d'adjacence.*

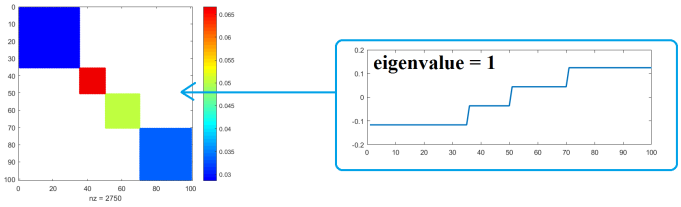
avec $nnz(\mathbf{B})$ le nombre d'éléments non nuls de la matrice $\mathbf{B}$. Ainsi, l'équilibrage bi-stochastique favorise globalement le poids des blocs diagonaux contre celui des blocs non diagonaux, ce qui devrait permettre aux algorithmes de détection de communautés développés pour les graphes non orientés d'identifier les communautés dans un graphe orienté avec flot.

La Figure 6 résume ces deux points en comparant les résultats de l'algorithme de Louvain, décrit dans [29], appliqué aux matrices d'adjacence de deux graphes et à leur équilibrage doublement stochastique. Dans cette figure, les images du haut correspondent à la détection d'une structure avec flot, les images du bas à la détection d'une structure avec une petite communauté au milieu de communautés plus grosses (il y a un zoom en rouge sur cette petite communauté dans l'image de gauche). Les figures du milieu montrent la structure en communautés identifiée par l'algorithme [29] lorsqu'on l'applique aux deux matrices d'adjacence. On observe la difficulté qu'a l'algorithme à détecter la structure en communautés du graphe avec flot. On observe aussi que la petite communauté n'est pas identifiée dans la figure du bas. Cependant, si l'on applique ce même algorithme après équilibrage doublement stochastique de ces deux matrices, il est cette fois-ci capable de détecter les structures par blocs de façon parfaite.

Le troisième intérêt que nous avons trouvé à cet équilibrage pour la détection de structures par blocs réside dans les propriétés induites après équilibrage sur les vecteurs



(a)



(b)

FIGURE 7 : *(a) : Une matrice par blocs diagonaux, avec une unique valeur numérique dans chaque bloc, et les quatre couples (valeur propre, vecteur propre) permettant de détecter sa structure. (b) : La même matrice après équilibrage doublement stochastique. Toute sa structure par blocs est visible via l'aspect constant par morceaux d'un unique vecteur propre, associé à la valeur propre* 1.

propres dominants de la matrice. En effet, de nombreux algorithmes [9, 14, 31–33], souvent appelés algorithmes spectraux, se basent sur les vecteurs propres d'une matrice pour détecter sa structure par blocs. Cela vient du fait que dans le cas d'une matrice possédant une structure par blocs, on est capable d'observer ces blocs au travers de ses vecteurs propres. Un exemple est donné Figure 7(a) : dans la matrice représentée, chaque bloc est caractérisé par un couple (valeur propre,vecteur propre). En observant quatre vecteurs propres calculés indépendamment, on est alors capable de trouver la structure par blocs de cette matrice. Mais en considérant l'équilibrage doublement stochastique de cette matrice, on peut faire apparaître toute la structure par blocs sur un unique vecteur propre. En effet, les valeurs numériques à l'intérieur des blocs ont été modifiées de sorte que chaque bloc est caractérisé par la même valeur propre, égale à 1. En choisissant un unique vecteur propre associé à 1, on est alors théoriquement capable de détecter l'intégralité de la structure de cette matrice en observant l'aspect constant par morceaux des composantes du-dit vecteur.

Plan de la thèse

Le but de cette thèse est d'investiguer le champ de la détection de structures par blocs dans les matrices, à la lumière de l'équilibrage doublement stochastique, et ceci pour divers types de matrices et d'applications. Etant donnée l'équation (1), il est clair que cette notion n'existe que pour les matrices carrées, et nous limiterons donc notre étude à ce type de matrices.

Chapitre 1 : Impact de l'équilibrage bi-stochastique sur les réseaux

Dans ce chapitre, nous nous intéressons à l'impact de l'équilibrage doublement stochastique sur la détection de communautés, dans le cas où les graphes sont non pondérés. En effet, en équilibrant les matrices de tels graphes, nous changeons fondamentalement la nature du problème en introduisant des valeurs numériques dans les matrices que l'on cherche à partitionner. Pour ce faire, nous commençons par présenter un certain nombre de mesures classiquement utilisées pour évaluer la cohérence d'une structure en communautés proposée pour un graphe. Quand cela est nécessaire, nous généralisons ces mesures aux cas des graphes pondérés, et en particulier aux cas des graphes dont la matrice d'adjacence est doublement stochastique. Nous observons que cela unifie ou simplifie un certain nombre de ces mesures. Nous donnons quelques considérations sur les structures en communautés optimisant ces mesures. Enfin, nous testons ces différentes mesures pour détecter la structure en communautés de réseaux générés artificiellement et dont on connaît la structure recherchée. Nous les appliquons aux réseaux avec et sans équilibrage. Cela nous permet de conclure que l'équilibrage doublement stochastique améliore la détection de la structure en communautés, et nous permet d'évaluer et de comparer ces mesures.

Chapitre 2 : Un Algorithme spectral

Nous proposons un algorithme pour détecter la structure par blocs de matrices carrées. Cet algorithme se base sur la structure constante par morceaux des vecteurs singuliers de la matrice après équilibrage doublement stochastique. La détection de cette structure constante par morceaux, loin d'être triviale, nécessite l'utilisation d'outils de traitement du signal, que nous présentons et adaptons afin de les rendre adéquats pour notre application. Un post-traitement basé sur une des mesures présentées au Chapitre 1 est ensuite détaillé. Enfin, nous observons les performances de notre algorithme pour le problème de classification non supervisée de la détection de forme dans un nuage de points,

où nous mettons en avant ses bons résultats par rapport à des algorithmes existants.

Chapitre 3 : Applications

Trois applications de la détection de structures par blocs dans les matrices carrées sont étudiées. D'abord nous revenons sur la détection de communautés, en comparant les performances de notre algorithme présenté, au Chapitre 2, à celles d'algorithmes spécifiquement conçus à cet effet. Nous observons que les résultats de notre méthode sont comparables à ceux de ces algorithmes. Dans la deuxième partie de ce chapitre, nous proposons d'utiliser l'algorithme du Chapitre 2 pour créer un préconditionneur de type Bloc Jacobi pour le solveur de Krylov `gmres`. Enfin, nous nous intéressons à la nécessité de découvrir des structures par blocs dans le domaine du traitement automatique du langage naturel.

Définitions et notations

Nous allons ici introduire quelques notions, définitions et notations qui seront utilisées tout au long de ce manuscrit.

Graphes

Nous appellerons **graphe simple** tout graphe G non orienté non pondéré. Nous définirons alors ce graphe par $G = (V, E)$, avec V l'ensemble fini des sommets, et $E \subset V \times V$ l'ensemble des arêtes.

Tout **graphe pondéré non orienté** – que nous appellerons parfois, par abus, simplement **graphe pondéré** – sera défini par $G = (V, E, w)$, avec V et E définis comme précédemment, et

$$\begin{array}{rccc} w : & E & \longrightarrow & \mathbb{R}_+ \\ & (i,j) & \longmapsto & w(i,j) \end{array}$$

la fonction des poids. En particulier, nous ne nous intéresserons qu'aux graphes de poids positifs, c'est pourquoi ici, w est à valeurs dans $\mathbb{R}_+$.
Nous notons que l'espace d'arrivée pourra être restreint dans certains cas $\mathbb{N}$, notamment quand nous nous intéresserons aux **graphes pondérés en nombres entiers**.

La matrice d'adjacence d'un graphe $G = (V, E, w)$ est une matrice $\mathbf{A} =$

$(a_{i,j})_{1\leq i,j\leq n}$ avec $n = |V|$ le cardinal de l'ensemble des noeuds, et telle que :

$$a(i,j) = \begin{cases} w(i,j) & \text{si } (i,j) \in E \quad \text{et G est pondéré,} \\ 1 & \text{si } (i,j) \in E \quad \text{et G est simple,} \\ 0 & \text{sinon.} \end{cases}$$

Le **degré d'un noeud** k sera noté d_k, et égal à $\sum_{i=1}^{n} a_{k,i}$.

Matrices et vecteurs

L'espace des matrices de taille p par q, à valeurs dans un ensemble Ω, sera noté $\mathcal{M}_{p,q}(\Omega)$, ou $\Omega^{p\times q}$. Si la matrice est carrée, *id est* $p = q$, nous noterons simplement $\mathcal{M}_p(\Omega)$ ou $\Omega^{p\times p}$. Les matrices sont généralement représentées par des lettre capitales en gras : $\mathbf{A} \in \mathcal{M}_{p,q}(\Omega)$ ou encore $\mathbf{M} \in \mathbb{R}^{n\times n}$. L'entrée (i,j) d'une matrice $\mathbf{A}$ sera en général notée $a_{i,j}$.

L'espace Ω^p représente **l'espace vectoriel de dimension** p, construit sur Ω. Les vecteurs seront généralement représentés par des lettres minuscules, en gras : $\mathbf{u} \in \mathbb{R}^n$, par exemple. La i^{eme} composante (ou coordonnée) d'un vecteur $\mathbf{u}$ sera en général notée u_i. On supposera toujours que les vecteurs sont des matrices colonnes, c'est-à-dire que $\mathbf{u} \in \Omega^p$ signifie $\mathbf{u} \in \mathcal{M}_{p,1}(\Omega)$.

L'opérateur $.^T$ représente la **transposition**. Autrement dit, soit $\mathbf{A} \in \mathcal{M}_{p,q}(\Omega)$, alors $\mathbf{A}^T$ est la matrice de $\mathcal{M}_{q,p}(\Omega)$ dont l'entrée (i,j) vaut $a_{j,i}$.

Soit $\mathbf{A} \in \mathcal{M}_n(\Omega)$, on notera $diag(\mathbf{A}) \in \mathcal{M}_n(\Omega)$ **la matrice diagonale** contenant la diagonale de $\mathbf{A}$. En outre, pour $\mathbf{u} \in \Omega^p$, on notera $\mathcal{D}(\mathbf{u}) \in \Omega^{p\times p}$ la matrice dont tous les coefficients valent 0, sauf sa diagonale qui est égale au vecteur $\mathbf{u}$.

Par ailleurs, on s'intéresse essentiellement à des graphes non orientés, et les matrices qui leur correspondent sont symétriques. Nous noterons alors $S_n(\Omega)$ l'ensemble des **matrices symétriques** de taille n, à coefficients dans Ω.

La notation $\mathbf{e} \in \mathbb{R}^p$ représente le vecteur $(1...1)^T$. En outre, sauf mention contraire, $\mathbf{e}_i$ représente le vecteur dont toutes les coordonnées sont nulles, sauf la i^{eme} qui est égale à 1. La matrice $\mathbf{I}$ est **la matrice identité**.

Nous aurons aussi besoin de **la matrice dont tous les éléments valent** 1, c'est-à-dire $\mathbf{e}\mathbf{e}^T$. Quelle que soit sa taille, nous noterons $\mathbf{J}$ cette matrice.

Nous nous intéresserons à des **matrices stochastiques** de $S_n(\mathbb{R}_+)$, c'est-à-dire des matrices dont la somme des éléments sur chacune des lignes vaut 1 (les matrices doublement stochastiques sont en particulier stochastique). Les matrices stochastiques sont les matrices

d'adjacence de **graphes de transition** non orientés, c'est-à-dire des graphes pour lesquels l'arête (i, j) représente la probabilité de passer de l'état i à l'état j.

Matrices bi-stochastiques

Nous énonçons quelques résultats sur la propriété de bi-stochasticité des matrices.

Définition 1. ***Matrice doublement stochastique :*** *Une matrice carrée à coefficients non négatifs* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *est dite* ***doublement stochastique*** *ou* ***bi-stochastique*** *si*

$$\begin{cases} \mathbf{A}\mathbf{e} = \mathbf{e} \\ \mathbf{A}^T\mathbf{e} = \mathbf{e} \end{cases},$$

Définition 2. ***Equilibrage doublement stochastique :*** *Soit une matrice carrée à coefficients non négatifs* $\mathbf{A} \in \mathbb{R}^{n \times n}$*, trouver un équilibrage doublement stochastique pour* $\mathbf{A}$ *signifie trouver deux vecteurs* $\mathbf{r}, \mathbf{c} \in \mathbb{R}_+^{*\,n}$ *tels que*

$$\begin{cases} \mathcal{D}(\mathbf{r})\mathbf{A}\mathcal{D}(\mathbf{c})\mathbf{e} = \mathbf{e} \\ \mathcal{D}(\mathbf{c})\mathbf{A}^T\mathcal{D}(\mathbf{r})\mathbf{e} = \mathbf{e} \end{cases},$$

où $\mathbf{r}$ *et* $\mathbf{c}$ *sont appelés* ***facteurs d'équilibrage*** *de* $\mathbf{A}$*.*

L'existence d'un équilibrage doublement stochastique est donné par le théorème de Sinkhorn-Knopp [30]. Nous l'énonçons après avoir introduit deux définitions sur la structure des matrices, auxquelles ce théorème fait appel. Ces définitions peuvent être trouvées dans [34].

Définition 3. ***Bi-irréductibilité ou entière indécomposabilité :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *est dite* ***bi-irréductible*** *s'il n'existe pas deux matrices de permutation* $\mathbf{R}, \mathbf{Q}$ *telles que :*

$$\mathbf{RAQ} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_{1,2} \\ 0 & \mathbf{A}_2 \end{bmatrix}$$

avec $\mathbf{A}_1, \mathbf{A}_2$ *deux matrices carrées non vides.*

Remarque 1. *Cela signifie qu'il n'existe pas de permutations indépendantes des lignes et des colonnes permettant de mettre* $\mathbf{A}$ *sous forme triangulaire par blocs.*

Définition 4. ***Support total :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *est dite* ***à support total*** *si tous ses éléments non nuls se trouve sur une diagonale strictement non nulle. Une*

caractérisation de cette définition est que $\mathbf{A}$ *est à support total s'il existe deux matrices de permutations* $\mathbf{R}, \mathbf{Q}$ *telles que*

$$\mathbf{RAQ} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix}$$

où $\mathbf{A}_1, ..., \mathbf{A}_k$ *sont des matrices bi-irréductibles [34].*

Nous pouvons maintenant énoncer le théorème de Sinkhorn-Knopp [30]

Théorème 1. ***Sinkhorn-Knopp :*** *Soit* $\mathbf{A} \in {\mathbb{R}_+}^{n \times n}$*, une condition nécessaire et suffisante pour qu'il existe une matrice doublement stochastique* $\mathbf{P}$ *de la forme* $\mathbf{DAF}$ *avec* $\mathbf{D}$ *et* $\mathbf{F}$ *deux matrices diagonales, et dont la diagonale principale est strictement positive, est que* $\mathbf{A}$ *soit à support total. Si* $\mathbf{P}$ *existe, alors* $\mathbf{P}$ *est unique.* $\mathbf{D}$ *et* $\mathbf{F}$ *sont aussi uniques à une multiplication scalaire près si et seulement si* $\mathbf{A}$ *est bi-irréductible.*

Nous introduisons maintenant la notion d'irréductibilité, qui véhicule une information essentielle sur les graphes d'adjacence associés à de telles matrice.

Définition 5. ***Irréductibilité :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *est dite* ***irréductible*** *s'il n'existe pas de matrice de permutation* $\mathbf{Q}$ *telle que :*

$$\mathbf{QAQ}^T = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_{1,2} \\ 0 & \mathbf{A}_2 \end{bmatrix}$$

avec $\mathbf{A}_1, \mathbf{A}_2$ *deux matrices carrées non vides. Une caractérisation des matrices irréductibles est qu'elles sont les matrices d'adjacence de graphes fortement connexes [35].*

Remarque 2. *Bien entendu, toute matrice bi-irréductible est irréductible. A l'inverse, si une matrice* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *est irréductible et que sa diagonale ne possède pas d'élément nul, alors* $\mathbf{A}$ *est bi-irréductible. Cela se montre simplement en appliquant l'algorithme de Pothen et Fan [36] à* $\mathbf{A}$*. Le but de cet algorithme est de permuter* $\mathbf{A}$ *pour la mettre sous forme triangulaire par blocs. Une fois que* $\mathbf{A}$ *est permutée de sorte à ce que sa diagonale ne possède pas d'élément nul, seules des permutations symétriques sur les lignes et les colonnes sont nécessaires pour trouver une forme triangulaire par blocs pour* $\mathbf{A}$*. Ainsi le fait que la diagonale de* $\mathbf{A}$ *soit pleine permet de forcer* $\mathbf{Q} = \mathbf{R}^T$ *dans la définition 3, ce qui nous ramène à la définition 5.*

Structures par blocs

Nous nous intéresserons aussi aux **matrices représentant des relations d'équivalence** sur l'ensemble des sommets, c'est-à-dire des matrices symétriques à valeurs dans $\{0,1\}$, qui respectent les propriétés de réflexivité et de transitivité. Pour une matrice $\mathbf{X} = (x_{i,j}) \in S_n(\{0,1\})$ représentant une relation d'équivalence, on peut écrire ces propriétés sous la forme relationnelle :

$$\text{réflexivité :} \quad \forall i \in \{1,...,n\}, \quad x_{i,i} = 1 \tag{2}$$

$$\text{transitivé :} \quad \forall i,j,k \in \{1,...,n\}, \quad x_{i,j} + x_{j,k} - x_{i,k} \leq 1 \tag{3}$$

comme cela est montré dans l'article [37].

La caractérisation de la propriété de réflexivité donnée par l'équation (2) implique qu'une matrice représentant une relation réflexive est à diagonale non nulle. Avec la caractérisation de la transitivité donnée par l'équation (3), on obtient que les matrices représentant une relation d'équivalence sur un ensemble de taille n sont les matrices symétriques $\mathbf{X} \in S_n(\{0,1\})$ pour lesquelles il existe une matrice de permutation $\mathbf{P} \in S_n(\{0,1\})$, telle que

$$\mathbf{PXP} = \begin{bmatrix} \mathbf{C}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{C}_r \end{bmatrix} \tag{4}$$

avec $\forall k \in \{1,...,r\}, \mathbf{C}_k = \mathbf{J}$ bloc de taille n_k – cf par exemple [38].

Par abus de langage, ces matrices caractérisant les relations d'équivalence seront aussi appelées **partitionnements**.

Un **partitionnement** étant à proprement parler une partition de $\{1,...,n\}$, on assimile un partitionnement à la relation d'équivalence fournissant les classes d'équivalence correspondant à la sus-dite partition. On remarque avec l'équation (4) que les classes d'équivalence correspondent aux blocs diagonaux. **L'ensemble des relations d'équivalence** sur n sommets sera noté $Eq(n)$. Les classes d'équivalences seront en général simplement appelées **classes**.

Chapitre 1

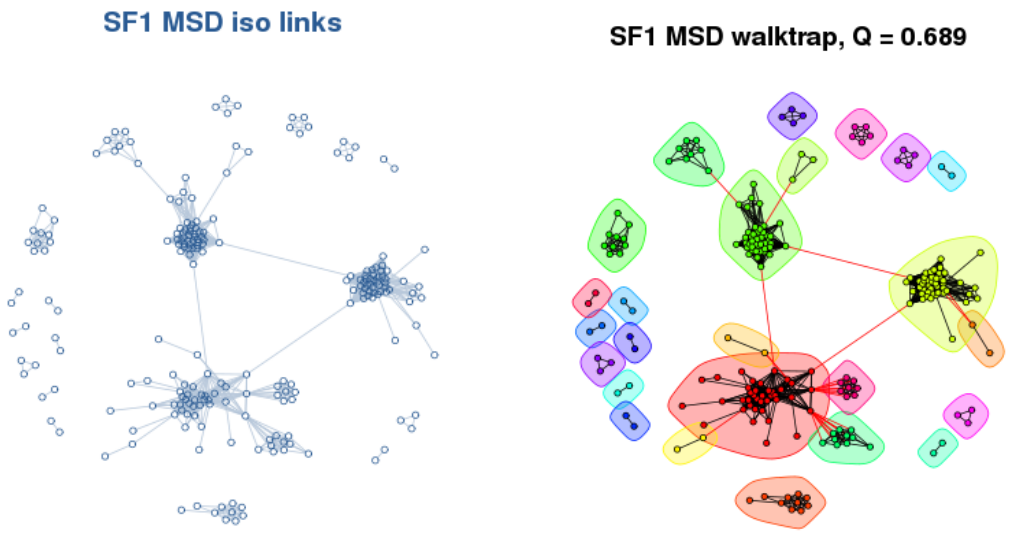
Impact de l'équilibrage bi-stochastique sur les réseaux

La détection de communautés dans les graphes est une problématique très actuelle. Les applications sont nombreuses, et les problèmes à prendre en considération très variés. Par exemple :

— Dans l'analyse de grands graphes : si un graphe est trop grand pour que l'on puisse étudier sa structure globale, le découper en communautés pour faciliter son analyse est une solution classique. C'est l'approche proposée notamment dans l'article [39].

— Suivant ce que représente le graphe, un groupe de sommets fortement liés peut signifier que ces sommets ont une propriété commune. Par exemple les auteurs de [1, 40] veulent trouver des groupes de protéines orthologues, c'est-à-dire des protéines ayant la même fonction dans un génome. Pour se faire, ils constituent un graphe dans lequel les noeuds sont les protéines. Deux protéines a et b sont liées par une arête si elles sont iso-orthologues, c'est-à-dire que la protéine la plus proche – au sens du pourcentage d'ADN – de a dans le génome auquel appartient b est la protéine b, *et vice et versa*. Deux protéines iso-orthologues ont une fort probabilité d'être orthologues. Mais cela n'est pas certain à cause d'éventuelles mutations génétiques. Pour trouver les groupes de protéines orthologues, les auteurs cherchent les communautés dans le graphe des liens d'iso-orthologie. La Figure 1.1, extraite de [1], montre un exemple de l'application de l'algorithme Walktrap [2] sur un graphe des liens d'iso-orthologie.

Une liste plus exhaustive de ces applications est donnée dans la thèse de Patricia Conde-Céspedes [41]. La détection de communautés consiste à trouver un partitionnement des sommets d'un graphe, et donc une structure diagonale par blocs sur la matrice choisie



(a) (b)

FIGURE 1.1 : *Ces deux images sont extraites de [1]. (a) : graphe des liens d'iso-orthologie d'un groupe de protéines de type Membrane Spanning Domaine – ou MSD. (b) : Résultat de l'algorithme Walktrap [2] sur ce graphe.*

pour représenter le graphe. Quand on modularise, on souhaite réunir des sommets qui ont des caractéristiques communes, qui se ressemblent, ou encore qui sont fortement liés les uns aux autres, dans une même classe. En parallèle, on cherche à ce que les sommets appartenant à des classes différentes soient différents : ne partagent pas ces mêmes caractéristiques, soient peu reliés entre eux, etc... Autrement dit, les sommets de classes différentes doivent posséder une différence structurelle.

On remarque que la définition proposée ici n'est pas une définition rigoureuse au sens mathématique du terme. En effet, il n'existe pas de définition formelle des communautés d'un graphe. Cette non formalisation du problème fait qu'il existe aujourd'hui beaucoup de critères pour évaluer la qualité du découpage d'un graphe en communautés. Ces critères dépendent du point de vue mathématique choisi pour appréhender un graphe, et du sens que l'on donne à une structure en communautés.

La façon la plus évidente est d'envisager le graphe comme un ensemble de noeuds et d'arêtes. Le partitionnement des noeuds en communautés doit alors présenter des classes dans lesquelles les arêtes entre les noeuds sont nombreuses, tandis que les arêtes sont peu nombreuses entre des noeuds appartenant à des classes différentes. La qualité d'une structure en communautés correspond donc à un décompte de cette quantité d'arêtes intra et inter classes. La mesure la plus employée pour cela est la modularité de Newman

et Girvan [42] : étant donnés un graphe et une structure en communautés, cette mesure compare le nombre des arêtes intra classes à l'espérance de cette quantité dans le modèle de configuration, c'est-à-dire la variable aléatoire dont les réalisations sont les graphes ayant la même distribution des degrés.

Une autre façon de considérer un graphe est de le voir comme la représentation d'une (ou plusieurs) relation(s) binaire(s) sur un ensemble d'objets. Ces objets représentent les noeuds du graphe, et les arêtes du graphe représentent le fait que deux objets soient liés dans la (les) relation(s) existante(s). Dans ce cas, la recherche d'une structure en communautés sur un graphe correspond à la recherche d'une relation d'équivalence sur l'ensemble des objets, qui approche au mieux la (les) relation(s) binaire(s). Dans ce contexte, il existe plusieurs mesures de la pertinence d'un découpage en communautés. Nous en donnons deux exemples.

— Un graphe peut être utilisé pour représenter une agrégation de relations binaires [43, 44]. Supposons que les objets de l'ensemble sont caractérisés par plusieurs paramètres ou modalités et ne peuvent prendre qu'une seule catégorie parmi ces modalités (en reprenant l'exemple de [41], les objets peuvent être des voitures, caractérisées par une couleur, une marque, ...). Dans ce cas, la relation d'équivalence que l'on cherche sur le graphe doit représenter une forme de consensus, c'est-à-dire qu'elle doit coïncider avec la plupart des relations d'équivalence du regroupement. Une idée classique est donc de rechercher la relation d'équivalence dont la somme des écarts à chaque relation du regroupement est minimisée. Plusieurs versions existent en fonction de la façon de mesurer l'écart entre deux relations.

— Dans une autre approche proposée par Zahn [45], le but est d'approcher une unique relation binaire symétrique, définie sur un ensemble d'objets V, par une relation d'équivalence. L'auteur propose d'évaluer la qualité de la relation d'équivalence en définissant une distance entre deux relations binaires, distance qui se base sur des considérations ensemblistes : une relation binaire étant envisagée comme un sous-ensemble du produit cartésien $V \times V$, si l'on note $\mathcal{R} \subset V \times V$ la relation binaire symétrique à approcher, on va chercher $\mathcal{X} \subset V \times V$ une relation d'équivalence qui minimise $|\overline{\mathcal{R}} \cap \mathcal{X}| + |\overline{\mathcal{X}} \cap \mathcal{R}|$, c'est-à-dire la somme des nombres d'éléments de $V \times V$ présents dans une relation et le complémentaire de l'autre.

On peut aussi appréhender un graphe comme une chaîne de Markov [46]. Dans ce cas, les noeuds du graphe sont les états pouvant être pris par une variable aléatoire, et les poids des arêtes sont proportionnels aux probabilités des changements d'état. En effet, soit un graphe $G = (V, E, w)$, le fait qu'il existe une arête du noeud $i \in V$ vers le noeud $j \in V$ de poids $w(i, j)$ se caractérise du point de vue des chaînes de Markov par le fait

que la variable aléatoire X, si elle se trouve dans l'état i à une étape n, a une probabilité $w(i,j)/\sum_{k\in V} w(i,k)$ de se trouver dans l'état j à l'étape $n+1$, autrement dit

$$p(X_{n+1} = j | X_n = i) = \frac{w(i,j)}{\sum_{k\in V} w(i,k)}.$$

Le lecteur intéressé par les chaînes de Markov et les marches aléatoires est invité à se référer à [25].

Une façon classique d'envisager les communautés dans une chaîne de Markov est de les aborder du point de vue des marches aléatoires : puisque les noeuds d'une même communauté sont fortement liés entre eux, et peu liés avec l'extérieur de cette communauté, un marcheur aléatoire se trouvant dans une communauté devrait y passer du temps. Plusieurs mesures pour évaluer la qualité d'un découpage en communautés ont été définies via cette idée de temps passé dans une communauté.

— La mesure appelée *Map equation* [47, 48] est basée sur l'idée d'encoder les marches aléatoires sur un minimum de bits. Classiquement, pour encoder une marche aléatoire efficacement, on attribue un code à chaque noeud, en faisant en sorte qu'un noeud visité fréquemment soit codé sur peu de bits (les noeuds visités rarement sont codés sur plus de bits, puisqu'ils sont peu fréquents dans la description de la marche). Les auteurs suggèrent qu'il existe une façon plus efficace d'encoder une marche aléatoire, en se basant sur la notion de partitionnement des noeuds du graphe. L'idée est d'utiliser un compromis entre deux niveaux d'encodage : un niveau pour coder l'entrée de la marche dans une partition, et l'autre pour coder les noeuds à l'intérieur de la partition, ainsi qu'un code pour désigner la sortie de la partition. En effet, cela augmente *a priori* le nombre de bits, puisqu'il est nécessaire de coder l'entrée (et la sortie) dans une partition. Cependant, cela permet aussi d'utiliser moins de bits pour coder les noeuds, puisque des noeuds dans des partitions différentes peuvent avoir le même code. Evidemment, plus un marcheur reste longtemps dans une même partition, plus cet encodage à deux niveaux fait sens. En se basant sur des outils de la théorie de l'information, les auteurs fournissent une borne minimum permettant d'encoder les marches avec ces deux niveaux. Cette borne ne dépend que des caractéristiques du graphe et du partitionnement du graphe choisi. C'est cette borne qu'ils nomment *Map equation* : sur un graphe donné, trouver le partitionnement du graphe minimisant cette borne permet de trouver un partitionnement des noeuds du graphe tel que les marcheurs aléatoires resteront longtemps dans chaque partition. Ces partitions correspondent bien à une notion de communautés.

— Une autre mesure, appelée *stabilité de Markov* [49], permet aussi d'évaluer la qualité

d'un découpage en communautés à travers les marches aléatoires. A nouveau, l'idée est qu'un marcheur aléatoire se trouvant dans une communauté doit y rester un certain temps. Ainsi, étant donné un temps t et un partitionnement en c classes, les auteurs calculent une matrice $c \times c$ dont un élément (i, j) indique la probabilité d'être dans la communauté j en étant parti de la communauté i, après t étapes. Cette mesure, basée sur la covariance entre la variable aléatoire à une étape n et à une étape $n + t$, est intéressante de par sa dépendance à t : le partitionnement optimal pour la stabilité de Markov dépend fortement de l'échelle de temps observée : plus grande est la valeur de t, plus un marcheur a de chance de s'être éloigné de son noeud de départ. Le partitionnement optimal est constitué de classes moins nombreuses et plus grandes pour les grandes valeurs de t, par rapport au partitionnement optimal pour de petites valeurs de t. En outre, les partitionnements auront en général des valeurs de stabilité plus faible pour de grandes valeurs de t. Ainsi, cette mesure permet d'évaluer la pertinence d'une structure en communautés en fonction d'une échelle de temps.

Enfin, on peut envisager un graphe comme la réalisation d'une variable aléatoire, en supposant que sa structure en communautés, ainsi que ses arêtes, ont été générées par des processus aléatoires – typiquement un modèle stochastique par blocs [50]. Dans le cas des modèles stochastiques par blocs par exemple, on suppose que les noeuds ont été affectés à des blocs par la réalisation de variables aléatoires, et que les arêtes entre les noeuds d'un bloc k ont été générées par une loi de probabilité possédant des caractéristiques qui sont propres à ce bloc, de même pour les arêtes entre les noeuds d'un bloc k et d'un bloc p. Dès lors que les espérance d'arêtes à l'intérieur des blocs sont supérieures à celles entre les blocs, les graphes générés par de tels modèles doivent avoir une structure en communautés. Si l'on envisage un graphe comme un objet généré par un tel modèle (en supposant connus les types de loi de probabilité du modèle, mais pas leurs paramètres), la qualité d'une structure en communautés proposée se mesure par maximum de vraisemblance entre le graphe à disposition et le modèle dont on a estimé statistiquement les paramètres.

Il s'agit-là d'une liste non exhaustive des mesures permettant d'évaluer la qualité d'un découpage en communautés étant donné un graphe, le but étant de montrer la prolifération de ce type de mesures. Bien entendu, il existe de nombreuses correspondances entre ces différentes mesures, et leur équivalence ou leur proximité sous certaines conditions est très régulièrement étudiée, car elle permet en partie d'unifier les différentes définitions de communautés. A titre d'exemple, il a été démontré que la modularité de Newman et Girvan est équivalente à la stabilité de Markov pour une échelle de temps égale à 1 [49]. Cette même modularité s'avère être équivalente dans certains cas au maximum de vraisemblance

avec un certain type de modèle stochastique par blocs [51]. Une étude [49] a aussi été menée sur la proximité de la stabilité de Markov (en fonction des différentes échelles de temps) avec d'autres mesures, notamment les notions de coupes et de coupes normalisées, qui sont des mesures quantifiant le nombre d'arêtes existant entre les communautés. Le récent article [52] propose aussi de rapprocher diverses mesures (modularité et coupes notamment) via l'introduction d'une nouvelle mesure paramétrée, nommée LambdaCC.

Dans ce chapitre, nous limiterons notre étude à des mesures ayant la propriété de linéarité (dont la définition peut être trouvée dans [41]). Pour un graphe à n sommets dont la matrice d'adjacence est $\mathbf{A}$, et une structure en communautés définie par une matrice $\mathbf{X}$, il s'agit des mesures pouvant se mettre sous la forme

$$F(\mathbf{A}, \mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_{i,j}) x_{i,j} = \sum_{i,j} \Phi(a_{i,j}) x_{i;j}$$

avec $\Phi : \mathbb{R} \to \mathbb{R}$. Nous imposons cette limitation afin de pouvoir mener une étude comparative sur ces mesures via l'algorithme de Louvain. En effet, il a été montré dans [53] que la propriété de linéarité est une condition suffisante pour pouvoir utiliser une mesure dans l'algorithme de Louvain. Cet algorithme étant une heuristique simple, efficace et de faible complexité pour optimiser une mesure sur un graphe, son utilisation pour comparer des mesures nous semble justifier.

Beaucoup de ces mesures linéaires ont été créées dans le cas des graphes non pondérés. Certains de ces travaux préexistent même à la notion de communautés, comme l'article de Zahn [45], dans lequel la problématique est de mesurer de façon ensembliste l'écart entre une relation binaire symétrique et une relation d'équivalence.

Comme nous l'avons vu en introduction, l'équilibrage doublement stochastique de la matrice d'adjacence d'un graphe permet de mettre en exergue sa structure en communautés. Or, travailler sur un équilibrage doublement stochastique de la matrice d'adjacence nécessite d'être en mesure de traiter les graphes pondérés, puisque ces matrices sont, par essence, pondérées.

La question qui se pose alors est la suivante : comment généraliser le critère de Zahn à une relation symétrique pondérée ? Car si l'exercice consistant à rendre équivalentes les notions de relations binaires symétriques et de graphes non orientés non pondérés est relativement simple, le cas des graphes pondérés demande à être analysé plus en profondeur, par exemple ici pour lui trouver un équivalent ensembliste.

Pour un certain nombre de critères d'évaluation du découpage d'un graphe en communautés – ces critères étant aussi appelés mesures de modularités –, une généralisation existe au moins dans le cas des graphes pondérés en nombres entiers. Cette généralisation

est, par exemple, justifiée pour la modularité de Newman et Girvan dans l'article [54], en considérant un graphe pondéré en nombre entier comme un multigraphe non pondéré, autrement dit une somme de graphes non pondérés possédant le même ensemble de sommets. Cette généralisation a l'air naturelle dans le cas du critère de Condorcet tel qu'il est défini par Patricia Conde-Céspedes et Jean François Marcotorchino dans leur article [37]. Mais lorsqu'on regarde de plus près, s'il est naturel de représenter un tableau de Condorcet – *id est* une somme de tableaux dont chacun représente une relation d'équivalence, tous ayant le même ensemble d'objets – comme un graphe pondéré, étant donné que les valeurs de ce tableau sont des entiers, le sens inverse demande à être justifié : est-il possible d'envisager n'importe quel graphe pondéré en nombres entiers comme un tableau de Condorcet ? A cet égard, la généralisation proposée par Romain Campigotto, Patricia Conde-Céspedes et Jean-Loup Guillaume dans leur article [53], et qui recouvre à la fois le critère de Condorcet et celui de Zahn, mérite une analyse en profondeur.

La seule justification permettant de passer d'un graphe pondéré en nombres entiers à un graphe ayant une pondération quelconque que nous ayons rencontrée dans la littérature est celle de l'article de Newman [54], dans lequel l'auteur propose d'envisager le poids d'une arête comme un débit. Ainsi, un graphe donné possède sa propre unité de débit – notons-la r –, qui n'est pas nécessairement 1, et l'auteur propose d'écrire le poids d'une arête $w(i,j)$ comme $r \times n_{i,j}$, avec $n_{i,j} \in \mathbb{N}$. Cependant, cette astuce ne fonctionne pas dans le cas d'un graphe ayant une pondération quelconque. On verra dans cette section comment il est possible de généraliser cela aux cas des graphes à pondération quelconque.

L'objectif de la première partie de ce chapitre est donc de proposer une généralisation pour un certain nombre de mesures de modularité dans le cas des graphes non orientés à pondérations quelconques, afin de pouvoir les comparer sur des matrices symétriques ayant comme caractéristique commune d'être bi-stochastiques. Notre travail s'appuie en grande partie sur les travaux de Patricia Conde-Céspedes. En effet, dans sa thèse ainsi que dans les articles qu'elle a co-écrits avec Jean-François Marcotorchino, Romain Campigotto et Jean-Loup Guillaume, de nombreux critères sont listés, explicités, étudiés et comparés entre eux, ce qui nous donne un point de départ théorique solide et étoffé.

Dans la Section 1.1, nous proposons une réflexion sur la généralisation de quelques mesures de modularité dans le cas des graphes pondérés. En Section 1.2, nous comparerons ces différents critères dans le cas de graphes dont la matrice d'adjacence est bi- stochastiques. Enfin nous comparerons en Section 1.3 les résultats de l'algorithme de Louvain [29] sur les graphes bruts et leur équivalent après équilibrage doublement stochastique de la matrice, pour montrer que l'équilibrage doublement stochastique permet une meilleure détection des communautés.

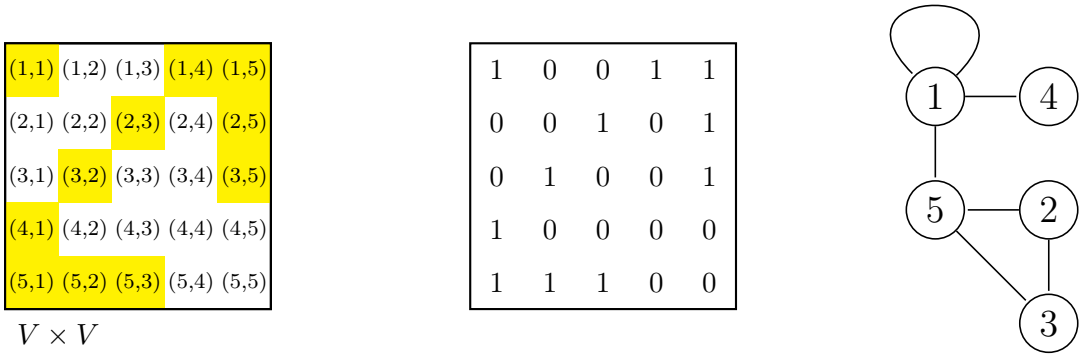
1.1 Généralisation de certains critères au cas des graphes pondérés

Dans cette section, nous allons présenter plusieurs critères d'évaluation de la qualité d'un partitionnement, et réfléchir à leur généralisation dans le cas des graphes pondérés, *id est* des matrices de $S_n(\mathbb{R}_+)$, dans le cas où ces critères ne sont pas naturellement construits sur cet ensemble.

1.1.1 Critère de Zahn

Le premier article dans lequel est formellement posé le problème du calcul de la distance entre deux relations binaires symétriques est celui de Zahn [45]. Dans cet article, Zahn cherche à approcher une relation symétrique binaire quelconque par une relation d'équivalence. Plus précisément, soit V un ensemble fini d'éléments, tel que $|V| = n$. Par commodité, on notera $V = \{1, ..., n\}$.

On note $\mathcal{R}$ la relation binaire symétrique croisant V – *id est* $\mathcal{R}$ est un opérateur qui met en relation un élément de V et un autre élément de V – que l'on cherche à approcher. En fait, on a très simplement $\mathcal{R} \subset V \times V$. Naturellement, il nous vient à l'esprit plusieurs façons d'envisager $\mathcal{R}$: sous la forme d'un ensemble, comme Zahn la définit, mais aussi sous la forme d'un graphe simple et donc sous forme matricielle, via sa matrice d'adjacence. Un exemple de ces trois représentations pour $V = \{1, ..., 5\}$ et $\mathcal{R} = \{(1,1), (1,4), (1,5), (2,3), (2,5), (3,5)\}$ est illustré en Figure 1.2.



$V \times V$

FIGURE 1.2 : *A gauche, ensemble $V \times V$, avec, surlignés en jaune, les éléments de $\mathcal{R} \subset V \times V$. $\mathcal{R}$ étant une relation symétrique, on pourrait ne la représenter qu'avec la moitié de ce dessin. Au centre, la matrice symétrique représentant $\mathcal{R}$, constituée de 0 et de 1. A droite, le graphe simple représentant $\mathcal{R}$*

Zahn cherche $\mathcal{X} \subset V \times V$ une relation d'équivalence qui "approche au mieux" $\mathcal{R}$.

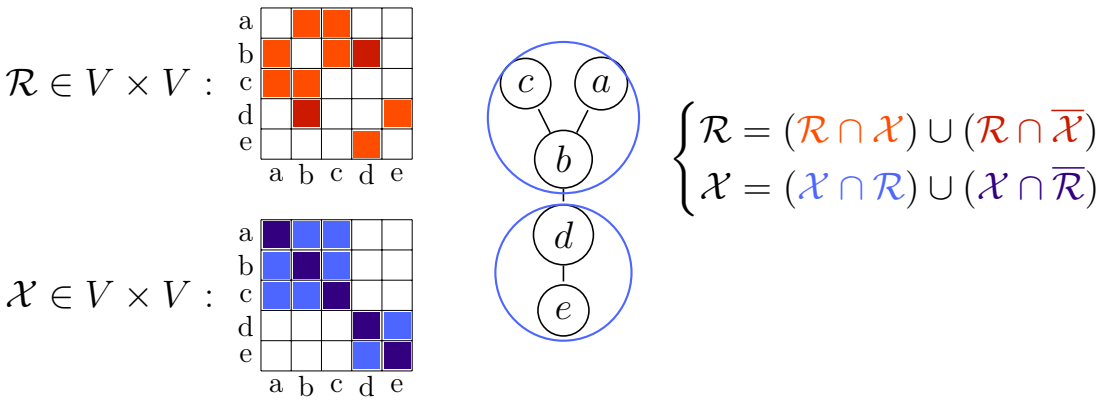


FIGURE 1.3 : *La distance de Zahn définie entre deux relations se base sur la représentation ensembliste des-dites relations.*

Pour ce faire, Zahn se place dans un contexte ensembliste pour définir une distance entre ces deux relations. Un exemple pour montrer comment Zahn construit sa distance est donné Figure 1.3. Dans cette figure sont affichées une relation binaire symétrique $\mathcal{R}$ et une relation d'équivalence $\mathcal{X}$, définies sur un ensemble $V \times V$ où $V = \{a, b, c, d, e\}$. La représentation ensembliste est à gauche de la figure, et la représentation via les graphes et les partitionnements est donnée au centre de la figure. On montre à droite de cette figure que chacune de ces relations peut être écrite comme la réunion de son intersection avec l'autre relation – carreaux affichés en clair à gauche de la figure – et de son intersection avec le complémentaire de l'autre relation – carreaux affichés en foncé. Ainsi, Zahn définit sa distance comme le nombre d'éléments qui sont dans l'intersection d'une relation et du complémentaire de l'autre, *id est* :

$$d_Z(\mathcal{R}, \mathcal{X}) = |\overline{\mathcal{X}} \cap \mathcal{R}| + |\overline{\mathcal{R}} \cap \mathcal{X}|.$$

Ainsi, Zahn cherche à résoudre :

$$(P) : \begin{cases} \min\limits_{\mathcal{X} \subset V \times V} d_Z(\mathcal{R}, \mathcal{X}) & = & |\overline{\mathcal{X}} \cap \mathcal{R}| + |\overline{\mathcal{R}} \cap \mathcal{X}| \\ \mathcal{X} \in Eq(n) \end{cases} \tag{1.1}$$

En notant $\mathbf{X}$ la représentation matricielle de $\mathcal{X}$ de terme général $(x_{i,j})$, et $\mathbf{A}$ celle de

$\mathcal{R}$ de terme général $(a_{i,j})$, on obtient les relations suivantes :

$$\begin{cases} \mathcal{R} \cap \overline{\mathcal{X}} = \{(i,j) \in V \times V : a_{i,j}(1 - x_{i,j}) \neq 0\} \\ \mathcal{X} \cap \overline{\mathcal{R}} = \{(i,j) \in V \times V : x_{i,j}(1 - a_{i,j}) \neq 0\} \end{cases}$$

On note $\overline{\mathbf{X}} = \mathbf{J} - \mathbf{X} = (\overline{x}_{i,j})_{i,j=1}^{n}$ et $\overline{\mathbf{A}} = \mathbf{J} - \mathbf{A} = (\overline{a}_{i,j})_{i,j=1}^{n}$ les matrices complémentaires de $\mathbf{X}$ et $\mathbf{A}$. On peut alors ré-écrire la distance de Zahn en utilisant ces représentations matricielles :

$$d_Z(\mathbf{A}, \mathbf{X}) = \frac{1}{2} \sum_{i,j} (a_{i,j}\overline{x}_{i,j} + x_{i,j}\overline{a}_{i,j}) \tag{1.2}$$

où le facteur 1/2 dénote simplement le fait que les deux matrices sont symétriques. Ainsi, si (i,j) est dans $\mathcal{R}$ – respectivement $\mathcal{X}$ – alors $a_{i,j} = a_{j,i} = 1$ – de même pour $x_{i,j}$.

Maintenant, comment peut-on généraliser ce raisonnement, dans le cas où nous n'avons plus une relation symétrique binaire, mais une relation symétrique pondérée, c'est-à-dire un graphe non orienté pondéré ?

Dans ce cas, on ne peut plus se contenter de compter les éléments : il faut prendre en compte les poids. En effet, il est naturel de considérer que plus la liaison entre deux éléments est grande, plus ces éléments ont une propriété en commun. Dans ce cas, le fait de ne pas comptabiliser $(i,j) \in \mathcal{R}$ doit être d'autant plus pénalisant que $a_{i,j}$ est grand.

Dans l'article de Romain Campigotto *et al.* [53], on trouve une généralisation pour le critère de Zahn, qui revient à considérer $\overline{\mathbf{A}} = a_{max}\mathbf{J} - \mathbf{A}$, avec $a_{max} = \underset{i,j}{max}(a_{i,j})$, comme la matrice complémentaire de $\mathbf{A}$. Le reste est inchangé. Ainsi, le critère de Zahn tel que généralisé dans cet article est

$$d_Z^{pond_1}(\mathbf{A}, \mathbf{X}) = \frac{1}{2} \sum_{i,j} (a_{i,j}(1 - x_{i,j}) + x_{i,j}(a_{max} - a_{i,j})) \tag{1.3}$$

Cependant, le fait de définir ainsi la matrice complémentaire de $\mathbf{A}$ dans le cas pondéré amène à l'exemple donné en Figure 1.4. Sur cette figure, deux partitionnements sont proposés pour un même graphe. Bien que les éléments de la première composante connexe soient moins liés que ceux de la deuxième, nous souhaiterions tout de même obtenir deux classes : en effet, les deux classes sont disjointes, et dans chacun des deux graphes connexes, les éléments sont tous liés de la même façon. Or le partitionnement proposant une classe par noeud dans le graphe connexe le moins lié donne une valeur plus faible au critère de Zahn tel que généralisé par Romain Campigotto *et al.*

Plus généralement, ce qui nous pose problème avec le critère $d_Z^{pond_1}$ de l'équation (1.3),

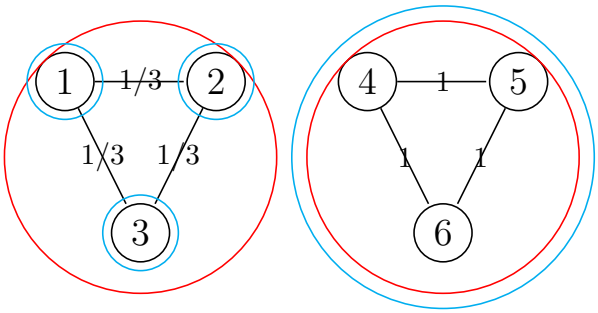


FIGURE 1.4 : *Graphe non orienté pondéré, contenant deux sous-graphes connexes. En rouge et en cyan, on a respectivement les classes de* $\mathbf{X}_r$ *et* $\mathbf{X}_c$*, deux propositions de partitionnement. On a* $d_Z^{pond_1}(\mathbf{A}, \mathbf{X}_r) = 5 > d_Z^{pond_1}(\mathbf{A}, \mathbf{X}_c) = 4$ *en prenant* $d_Z^{pond_1}$ *comme définie à l'équation (1.3). En prenant* $d_Z^{pond_2}$ *comme définie à l'équation (1.4),* $d_Z^{pond_2}(\mathbf{A}, \mathbf{X}_r) = -2$ *et* $d_Z^{pond_2}(\mathbf{A}, \mathbf{X}_c) \approx -0.7$*, d'où* $d_Z^{pond_2}(\mathbf{A}, \mathbf{X}_r) < d_Z^{pond_2}(\mathbf{A}, \mathbf{X}_c)$

c'est :

— qu'il décompte tous les éléments de $\mathcal{X} \cap \overline{\mathcal{R}}$ au poids a_{max}, ce qui rend la prise en compte d'un élément de cet ensemble aussi pénalisante que la pire prise en compte d'un élément de $\mathcal{R} \cap \overline{\mathcal{X}}$ pour toutes les configurations de $\mathcal{X}$ possibles.

— le fait que tous les éléments de l'intersection $\mathcal{X} \cap \mathcal{R}$ soient pénalisants, sauf si leur valeur dans $\mathcal{R}$ est égale à a_{max}.

Nous proposons donc une autre généralisation du critère de Zahn, s'appuyant cette fois sur la moyenne des éléments :

$$d_Z^{pond_2}(\mathbf{A}, \mathbf{X}) = \frac{1}{2} \sum_{i,j} (a_{i,j}(1 - x_{i,j}) + x_{i,j}(a_{mean} - a_{i,j})) \tag{1.4}$$

avec $a_{mean} = \dfrac{1}{n^2} \sum_{i,j} a_{i,j}$. Ce qui signifie que la matrice complémentaire de $\mathbf{A}$ devient $\overline{\mathbf{A}} = a_{mean}\mathbf{J} - \mathbf{A}$ dans le cas pondéré. On remarque qu'elle n'est alors plus non-négative.

Par rapport aux deux problèmes évoqués relativement au critère de Zahn généralisé aux graphes pondérés, tel qu'il était proposé équation (1.3), la matrice complémentaire définie ci-dessus est une meilleure alternative. En effet :

— chaque élément de $\overline{\mathcal{R}} \cap \mathcal{X}$ pénalise le critère de Zahn de la valeur de l'élément moyen a_{mean},

— les éléments de l'intersection $\mathcal{X} \cap \mathcal{R}$ pénalisent le critère dans le cas où leur valeur dans $\mathcal{R}$ est inférieure à la valeur moyenne a_{mean}. Sinon, au contraire, ils vont

apporter une contribution négative à ce critère, que l'on cherche à minimiser.

On peut par ailleurs noter que cette généralisation convient mieux que la précédente dans l'exemple donné Figure 1.4, où l'on observe que le critère de Zahn ainsi généralisé a une valeur plus faible pour le découpage du graphe en deux communautés que pour le découpage ayant une communauté par sommet pour la première composante connexe.

Remarque 3. *Il est évidemment possible d'envisager d'autres façons de généraliser le critère de Zahn en définissant autrement la matrice complémentaire de* $\mathbf{A}$*. Dans le cas des matrices creuses par exemple, on peut considérer que* $a_{i,j} = 0$ *n'est pas une valeur de pondération au même titre que toute autre, et que les indices* $\{ (i,j) : a_{i,j} = 0 \}$ *ont un statut particulier, relatif à la structure de* $\mathcal{R}$*. En ce sens, on peut définir* a_{mean} *différemment, par exemple en choisissant* $a_{mean} = \frac{1}{nnz} \sum_{i,j} a_{i,j}$*,* nnz *étant le nombre d'éléments non nuls dans* $\mathbf{A}$*. En prenant cette généralisation, dans l'exemple de la Figure 1.4, les partitionnements en rouge et en cyan deviennent équivalents.*

1.1.2 Critère de Newman et Girvan

Le critère de Newman et Girvan, aussi appelé simplement modularité, est le plus connu et le plus utilisé des critères de modularité. Newman et Girvan l'introduisent dans leur article [42] pour des graphes simples, en extrapolant la notion d'"assortative mixing" présentée par Newman dans son article [55]. A l'origine Newman et Girvan associent ce critère à des algorithmes de classification hiérarchique : quelle que soit la méthode de hiérarchisation utilisée, une fois le dendrogramme établi, il faut choisir un partitionnement parmi ceux proposés par les différents niveaux du dendrogramme. C'est le partitionnement donnant la meilleure mesure de modularité qui sera finalement conservé.

Cette mesure, étant donnée ses excellents résultats, a été largement reprise dans différents algorithmes de découpage en communautés et de classification, que ce soit pour évaluer un découpage, comme critère d'acceptation ou de rejet, ou comme une fonction que l'on cherche à maximiser [2, 29, 42, 56]. On peut noter qu'il a été montré, dans l'article de Brandes *et al* [57], que le problème de maximisation de cette fonction-objectif est "strongly NP-complete", et que les articles proposant d'approcher le partitionnement maximisant la modularité sont donc essentiellement des heuristiques qui permettent d'obtenir un résultat acceptable. Ceci étant dit, depuis la mise en lumière dans l'article [58] des problèmes de limite de résolution présentés par cette mesure, elle est plutôt utilisée comme un outil de validation que comme une fonction à maximiser dont on cherche l'optimum.

Dans cette partie, nous allons définir le critère de Newman et Girvan, expliquer sa généralisation aux graphes pondérés en nombres entiers proposée par Newman, et proposer

une généralisation aux graphes à pondération quelconque.

Graphes simples et pondérés en nombres entiers

Pour définir leur critère de modularité, Newman et Girvan partent du principe qu'un graphe aléatoire ne possède pas de structure en communautés. Pour un partitionnement donné, on compare donc la fraction des arêtes ayant leurs deux sommets dans une même classe (pour le graphe étudié) à la même quantité dans le cas d'un graphe aléatoire ayant la même distribution des degrés que le graphe étudié. La pertinence du partitionnement proposé est validée par la supériorité de cette valeur pour le graphe étudié sur celle obtenue pour le graphe aléatoire. Ainsi, pour un graphe simple, en notant $\mathbf{A} \in S_n(\{0,1\})$ sa matrice d'adjacence de terme général $(a_{i,j})$, avec $m = \frac{1}{2}\sum_{i,j} a_{i,j}$ le nombre d'arêtes, et pour un partitionnement $\mathbf{X}$ de terme général $(x_{i,j})$, possédant $\mathcal{K}$ classes notées $C_1, ...C_{\mathcal{K}}$, la modularité est donnée par la formule :

$$\forall \mathbf{A} \in S_n(\{0,1\}), \forall \mathbf{X} \in Eq(n), F_{NG}(\mathbf{A},\mathbf{X}) = \sum_i \Big(e_{i,i} - \big(\sum_j e_{i,j}\big)^2\Big)$$

avec $\mathbf{E} = (e_{p,q}) \in \mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ la matrice telle que $e_{p,q} = \frac{1}{2m}\sum_{i\in C_p}\sum_{j\in C_q} a_{i,j}$.

Cette définition de la modularité est proposée sous une formulation différente par Newman dans [54] :

$$F_{NG}(\mathbf{A},\mathbf{X}) = \frac{1}{2m}\sum_{i,j}(a_{i,j} - \frac{d_i d_j}{2m})x_{i,j} \tag{1.5}$$

avec $d_i = \sum_k a_{i,k}$ le degré du noeud i. Cette formulation correspond clairement à la différence entre le nombre d'arêtes intra-communautés du graphe initial représenté par $\mathbf{A}$ et le nombre d'arêtes intra-communautés d'un graphe aléatoire ayant la même distribution de degrés que $\mathbf{A}$, puisque le terme général de la matrice d'adjacence d'un tel graphe est $\frac{d_i d_j}{2m}$, comme cela est montré sur le petit exemple de la Figure 1.5.

Dans ce même article, Newman généralise un certain nombre de ses résultats aux graphes pondérés, notamment l'algorithme bien connu du *Edge Betweeness*, algorithme de hiérarchisation conçu à l'origine pour des graphes simples – cf [59].

Pour généraliser ses résultats, Newman commence par adapter un certain nombre de notions de base et algorithmes aux graphes pondérés en nombres entiers, en envisageant ces derniers comme des multigraphes. Autrement dit, il considère un graphe non orienté pondéré en nombres entiers comme une somme de graphes simples partageant le même

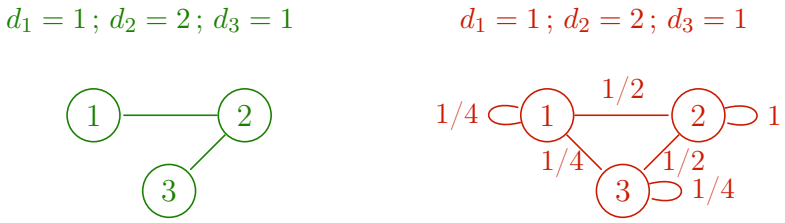


FIGURE 1.5 : *A gauche : un graphe simple. A droite : le graphe aléatoire ayant la même distribution des degrés que le graphe de gauche. La distribution des degrés est donné au dessus pour les deux graphes.*

ensemble de sommets. Le poids d'une arête devient alors le nombre de graphes non pondérés qui possèdent cette arête. Un exemple est donné Figure 1.6. Pour généraliser la modularité aux graphes pondérés, il conserve la formule de l'équation (1.5), et les définitions de m et de d_i comme pour un graphe non pondéré, *id est* $m = \frac{1}{2}\sum_{i,j} a_{i,j}$, et $d_i = \sum_k a_{i,k}$. L'unique différence avec la définition de la modularité donnée par l'équation (1.5) tient donc dans le fait que $\mathbf{A} \in S_n(\mathbb{N})$.

Newman fournit en outre une astuce censée permettre de passer aux graphes à pondération quelconque, du moins dans l'adaptation du *Edge Betweeness*. Soit $G = (V, E, w)$ un graphe pondéré avec $w : E \to \mathbb{R} \setminus \{0\}$. L'astuce proposée par Newman consiste à écrire le poids de chaque arête $(i, j) \in E$ sous la forme $rn_{i,j}$, avec $n_{i,j} \in \mathbb{N}$ et $r \in \mathbb{R}$. Ainsi, on peut écrire $w = rw'$, avec $w' : E \to \mathbb{N} \setminus \{0\}$. Newman définit r comme l'unité de débit du graphe, et le poids $rn_{i,j}$ d'une arête représente la capacité de cette arête à transporter $n_{i,j}$ unités d'un flot quelconque.

Graphes quelconques

Le problème pour la généralisation aux graphes pondérés quelconques, c'est que l'astuce énoncée par Newman n'est pas vraie dans le cas général. En effet, un tel couple (r, w') n'existe que dans le cas où le ratio entre deux poids non nuls quelconques est rationnel. Formellement, nous démontrons la propriété suivante :

Propriété 1. *Soit $G = (V, E, w)$ un graphe, avec $w : E \to \mathbb{R}_+^*$ sa fonction poids, et $\mathbf{A} \in S_n(\mathbb{R}_+)$ sa matrice d'adjacence, alors*

$$\exists (r, \mathbf{N}) \in (\mathbb{R}_+^* \times S_n(\mathbb{N})) : \mathbf{A} = r\mathbf{N} \iff \forall (i, j), (p, q) \in E, \frac{w(i, j)}{w(p, q)} \in \mathbb{Q}. \tag{1.6}$$

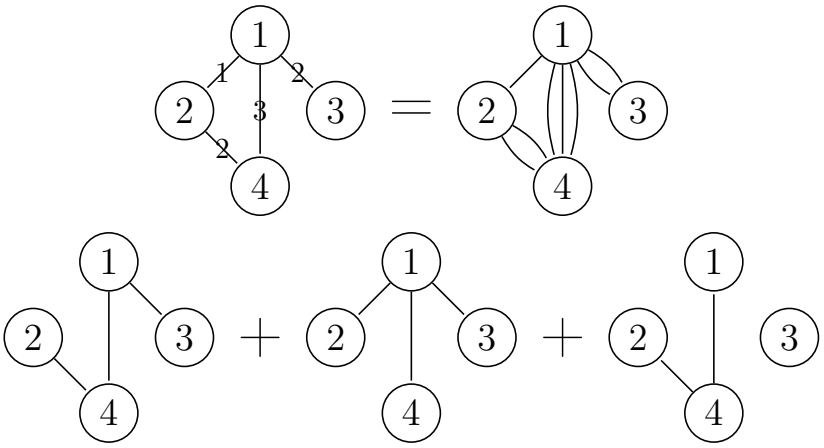


FIGURE 1.6 : *Représentation des graphes pondérés en nombres entiers : en haut à gauche, un graphe pondéré en nombre entier, à sa droite, le multigraphe correspondant, en dessous, une décomposition en graphes simples permettant d'obtenir le graphe pondéré en nombre entier. Cette somme n'est pas unique.*

Démonstration. Soit $V = \{1, ..., n\}$, alors $|E| = m \leq \frac{n(n+1)}{2}$. Pour alléger les notations et améliorer la compréhension, on écrira $E = \{1, ..., m\}$ en supposant que l'indexation des arêtes correspond à leur tri dans l'ordre lexicographique. $\forall i \in E$, on notera $w(i) = w_i$.
($\Rightarrow$) trivial : si $\exists r \in \mathbb{R}_+^* : \forall i \in E, w_i = rn_i$, avec $n_i \in \mathbb{N}\backslash\{0\}$, alors $\forall i, j \in E, \dfrac{w_i}{w_j} = \dfrac{n_i}{n_j} \in \mathbb{Q}$.
($\Leftarrow$) Par récurrence sur $|E|$:
Initialisation : Si $|E| = 1$, alors $\dfrac{w_1}{w_1} = 1 \in \mathbb{Q}$, et $w_1 = w_1 \times 1$, donc en posant $r = w_1$, on a $\forall i \in E, w_i = rn_i$, avec $n_i \in \mathbb{N} \setminus \{0\}$.

Récurrence : On suppose un ensemble $E = \{1, ..., k\}$, et un vecteur $w \in (\mathbb{R}_+^*)^k$ tel que $\forall i, j \in E, \dfrac{w_i}{w_j} \in \mathbb{Q}$. On suppose en outre que

$$\exists r \in \mathbb{R}_+^*, \exists N \in (\mathbb{N} \setminus \{0\})^k : \forall i \in E, w_i = rN_i.$$

On ajoute une coordonnée $w_{k+1} \neq 0$ au vecteur w pour former le vecteur $\tilde{w} \in (\mathbb{R}_+^*)^{k+1}$. On choisit w_{k+1} de sorte que $\forall i \leq k, \dfrac{w_{k+1}}{w_i} \in \mathbb{Q}$.

Notons $\tilde{E} = \{1, ..., k+1\}$, alors $\forall i, j \in \tilde{E}, \dfrac{w_i}{w_j} \in \mathbb{Q}$. En effet :

— soit $i, j \leq k$, d'où $\dfrac{w_i}{w_j} \in \mathbb{Q}$ par définition de w,

— soit $j \leq k$ et $i = k + 1$, d'où $\dfrac{w_i}{w_j} \in \mathbb{Q}$ par définition de w_{k+1},

— soit $i \leq k$ et $j = k+1$, d'où $\frac{w_i}{w_j} \in \mathbb{Q}$ car c'est l'inverse de $\frac{w_j}{w_i} \in \mathbb{Q}$,

— soit $i = j = k+1$, et $\frac{w_i}{w_j} = 1 \in \mathbb{Q}$.

On vérifie donc bien l'hypothèse de droite de l'équivalence donnée à l'équation (1.6).

Par ailleurs :

$$\forall i \in E, \exists p_i, q_i \in \mathbb{N} \setminus \{0\} : \frac{w_{k+1}}{w_i} = \frac{p_i}{q_i}$$

De plus, étant donnée l'hypothèse de récurrence sur $w \in (\mathbb{R}_+{}^*)^k$, on a

$$\forall i \in E, w_i = rN_i.$$

On peut donc écrire :

$$\forall i \in E, \frac{w_{k+1}}{w_i} = \frac{w_{k+1}}{rN_i}$$

Ainsi, $\forall i \in E$, on a l'égalité :

$$\frac{w_{k+1}}{rN_i} = \frac{p_i}{q_i}$$

et donc :

$$\forall i \in E, w_{k+1} = rN_i\frac{p_i}{q_i}$$

On veut maintenant trouver $\tilde{r} \in \mathbb{R}^*$ tel que :

$$\exists \tilde{N} \in (\mathbb{N} \setminus \{0\})^{k+1} : \forall i \in \tilde{E}, w_i = \tilde{r}\tilde{N}_i.$$

On choisit un indice $i_0 \in E$ et on pose $r' = \frac{r}{q_{i_0}}$. On a alors :

$$w_{k+1} = rN_{i_0}\frac{p_{i_0}}{q_{i_0}} = r'N_{i_0}p_{i_0} = r'N'_{k+1}, \text{ avec } N'_{k+1} = p_{i_0}N_{i_0} \in \mathbb{N}.$$

ainsi que :

$$\forall i \in E, w_i = rN_i = \frac{r}{q_{i_0}}q_{i_0}N_i = r'q_{i_0}N_i = r'N'_i, \text{ avec } N'_i = q_{i_0}N_i \in \mathbb{N}$$

Finalement, on obtient que :

$$\forall i \in \tilde{E}, \exists N'_i \in \mathbb{N} : \tilde{w}_i = r'N'_i$$

Et la récurrence est établie. □

Dans la suite, nous allons montrer pourquoi la formule de l'équation (1.5) peut

malgré tout être utilisée pour mesurer la modularité des graphes à pondération positive quelconque.

Pour ce faire, nous introduisons la fonction

$$\begin{array}{rccc} \Phi: & S_n(\mathbb{R}_+) \times Eq(n) & \longrightarrow & \mathbb{R} \\ & (\mathbf{A}, \mathbf{X}) & \longmapsto & \frac{1}{2m} \sum_{i,j} (a_{i,j} - \frac{d_i d_j}{2m}) x_{i,j} \end{array}$$

où l'on rappelle que $2m = \sum_{i,j} a_{i,j}$ et $\forall k \in \{1, ..., n\} d_k = \sum_i a_{k,i}$.

On va montrer que cette fonction permet de prolonger le critère de Newman à tous les graphes non orientés à pondération positive.

Déjà, on peut remarquer que Φ coïncide avec la modularité de Newman sur $S_n(\mathbb{N})$:

$$\forall \mathbf{X} \in Eq(n), \forall \mathbf{A} \in S_n(\mathbb{N}), \Phi(\mathbf{A}, \mathbf{X}) = F_{NG}(\mathbf{A}, \mathbf{X})$$

De plus, c'est une fraction rationnelle suivant sa première variable, car $2m$ et les degrés d_k dépendent linéairement des $a_{i,j}$. Elle est donc continue suivant sa première variable, partout sauf en $\mathbf{0}_{\mathbb{R}^{n \times n}}$, seule matrice positive pour laquelle $2m = 0$.

On introduit l'ensemble

$$R_n(\mathbb{Q}) = \left\{ \mathbf{A} = (a_{i,j}) \in S_n(\mathbb{R}_+) \text{ tq } \forall (k,l), (p,q) \in \{1, ..., n\}^2, \left\{ \begin{array}{l} \text{soit } a_{p,q} = 0, \\ \text{soit } \dfrac{a_{k,l}}{a_{p,q}} \in \mathbb{Q} \end{array} \right. \right\}.$$

Concrètement, il s'agit de l'ensemble des matrices vérifiant la propriété 1 ci-dessus, et donc pour lesquelles Newman a généralisé sa notion de modularité, via l'introduction de la notion de débit.

Avec cette notation, la propriété 1 se reformule

$$R_n(\mathbb{Q}) = \{ \mathbf{Q} \in S_n(\mathbb{R}_+) : \mathbf{Q} = r\mathbf{N}, \text{ avec } r \in {\mathbb{R}_+}^* \text{ et } \mathbf{N} \in S_n(\mathbb{N}) \}.$$

On remarque que la décomposition des $\mathbf{Q} \in R_n(\mathbb{Q})$ en $r\mathbf{N}$ n'est pas unique. Il faut donc montrer que la définition de Φ reste malgré tout cohérente, autrement dit, si $\mathbf{Q} \in R_n(\mathbb{Q})$ est tel que $\mathbf{Q} = r\mathbf{N} = \tilde{r}\tilde{\mathbf{N}}$, alors

$$\Phi(\mathbf{N}, \mathbf{X}) = \Phi(\tilde{\mathbf{N}}, \mathbf{X}), \forall \mathbf{X} \in Eq(n)$$

Soit $\mathbf{Q} \in R_n(\mathbb{Q}) : \mathbf{Q} = r\mathbf{N}$, avec $(r, \mathbf{N}) \in (\mathbb{R}_+^* \times S_n(\mathbb{N}))$, on a en fait :

$$\begin{aligned}
\forall \mathbf{X} \in Eq(n), \Phi(\mathbf{Q}, \mathbf{X}) &= \frac{1}{\sum\limits_{s,t} q_{s,t}} \sum_{i,j} \left(q_{i,j} - \frac{\sum\limits_k q_{i,k} \sum\limits_k q_{j,k}}{\sum\limits_{s,t} q_{s,t}} \right) x_{i,j} \\
&= \frac{1}{\sum\limits_{s,t} rn_{s,t}} \sum_{i,j} \left(rn_{i,j} - \frac{\sum\limits_k rn_{i,k} \sum\limits_k rn_{j,k}}{\sum\limits_{s,t} rn_{s,t}} \right) x_{i,j} \\
&= \frac{1}{\sum\limits_{s,t} n_{s,t}} \sum_{i,j} \left(n_{i,j} - \frac{r^2 \sum\limits_k n_{i,k} \sum\limits_k n_{j,k}}{r^2 \sum\limits_{s,t} n_{s,t}} \right) x_{i,j} \\
&= \Phi(\mathbf{N}, \mathbf{X})
\end{aligned}$$

Ainsi, si l'on reprend la proposition de Newman qui consiste à considérer r comme l'unité de débit du graphe, on remarque que Φ ne dépend pas de cette unité de débit. Cela implique notamment que pour deux matrices $\mathbf{A}_1$ et $\mathbf{A}_2$ égales à une multiplication scalaire près, le graphe de la fonction partielle $\Phi(\mathbf{A}_{1/2}, .) : Eq(n) \longrightarrow \mathbb{R}$ sera le même. Si ces matrices sont à coefficients entiers, alors cela montre qu'elles auront la même modularité, étant donné que la construction de Φ est issue de la modularité définie par Newman sur les graphes pondérés en nombre entiers. Par ailleurs, ce fait permet d'élargir la notion de modularité aux matrices de $R_n(\mathbb{Q})$ de façon cohérente. Ainsi, pour une matrice $\mathbf{Q} \in R_n(\mathbb{Q})$, on pose :

$$\begin{array}{rccc}
F_{NG}(\mathbf{Q}, .) : & Eq(n) & \longrightarrow & \mathbb{R} \\
& \mathbf{X} & \longmapsto & F_{NG}(\mathbf{Q}, \mathbf{X}) = \Phi(\mathbf{Q}, \mathbf{X})
\end{array}$$

Cette généralisation reprend et étaye la généralisation initialement proposée par Newman pour les matrices de $S_n(\mathbb{N})$.

Enfin, on a la propriété suivante :

Propriété 2.

$$\forall \mathbf{A} \in S_n(\mathbb{R}_+), \exists (\mathbf{Q}_k)_{k \in \mathbb{N}} \in (S_n(\mathbb{Q}_+))^{\mathbb{N}} \subset (R_n(\mathbb{Q}))^{\mathbb{N}} : (\mathbf{Q}_k) \underset{k \to \infty}{\longrightarrow} \mathbf{A}.$$

Démonstration. Ce résultat est obtenu en utilisant la densité de $\mathbb{Q}$ dans $\mathbb{R}$, et le fait que $\mathcal{M}_n(\mathbb{R})$ est un espace vectoriel normé de dimension finie (et donc toutes les normes sont équivalentes).

On a donc $\overline{S_n(\mathbb{Q}_+)} \supset S_n(\mathbb{R}_+)$. □

Ainsi, on peut prolonger par continuité la notion de critère de modularité à toutes les matrices dans $S_n(\mathbb{R}_+)$ par passage à la limite : soit un partitionnement $\mathbf{X} \in Eq(n)$ donné, une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ et une suite de matrices $(\mathbf{Q}_k)_k \in (R_n(\mathbb{Q}))^{\mathbb{N}} : (\mathbf{Q}_k) \underset{k\to\infty}{\longrightarrow} \mathbf{A}$:

$$F_{NG}(\mathbf{A}, \mathbf{X}) = \lim_{k\to\infty} (F_{NG}(\mathbf{Q}_k, \mathbf{X})) = \Phi(\mathbf{A}, \mathbf{X})$$

par continuité de $\Phi(., \mathbf{X})$. On a ainsi étendu (par prolongement par continuité) le critère de Newman aux graphes non orientés à pondération quelconque. On peut donc expliciter sa formule

$$\forall \mathbf{A} \in S_n(\mathbb{R}_+), \forall \mathbf{X} \in Eq(n), F_{NG}(\mathbf{A}, \mathbf{X}) = \frac{1}{2m} \sum_{i,j} (a_{i,j} - \frac{d_i d_j}{2m}) x_{i,j}$$

qui est la même que celle de l'équation (1.5), par continuité de la fonction Φ sur les $a_{i,j}$.

1.1.3 Critère de Condorcet

Ce critère sert initialement à mesurer la similarité entre une relation d'équivalence et une matrice relationnelle de Condorcet, *id est* une somme de matrices représentant chacune une relation d'équivalence. Pour bien comprendre ce critère, nous nous sommes appuyés sur l'article [44] de Pierre Michaud et Jean François Marcotorchino.

Dans cette section, après avoir défini ce critère, nous allons montrer qu'il ne peut être qu'associé à un problème de Condorcet, et qu'il n'est pas cohérent de chercher la valeur du critère de Condorcet pour une matrice quelconque.

Définition

Soit $\mathbf{C}^1, ..., \mathbf{C}^m \in Eq(n)$, de termes généraux $(c^k_{i,j}), k = 1, ...; m$, soit $\mathbf{X} \in Eq(n)$, le critère de Condorcet est défini de la façon suivante :

$$F_c(\{\mathbf{C}^k\}_k, X) = \sum_{k=1}^{m} \left(\sum_{i,j} c^k_{i,j} x_{i,j} + \overline{c}^k_{i,j} \overline{x}_{i,j} \right)$$

avec $\forall k \in \{1, ..., m\}, \overline{c}^k_{i,j} = 1 - c^k_{i,j}$, et $\overline{x}_{i,j} = 1 - x_{i,j}$.
Cette équation se réécrivant :

$$F_C(\{\mathbf{C}^k\}, \mathbf{X}) = \sum_{i,j} \left\{ \Big(\sum_{k=1}^{m} c^k_{i,j} \Big) x_{i,j} + \Big(\sum_{k=1}^{m} \overline{c}^k_{i,j} \Big) \overline{x}_{i,j} \right\} \tag{1.7}$$

on peut poser $\mathbf{C} = \sum_{k=1}^{m} \mathbf{C}^k$ et $\overline{\mathbf{C}} = \sum_{k=1}^{m} \overline{\mathbf{C}}^k = m\mathbf{J} - \mathbf{C}$.
Dans l'article [37] de Patricia Conde-Céspedes et Jean François Marcotorchino, il est écrit que le critère de Zahn est équivalent au critère de Condorcet dans le cas où $m = 1$. Il est ici évident qu'il s'agit du cas où l'on élargit le problème de Condorcet aux relations binaires symétriques – autrement, le problème de Zahn étant de trouver une relation d'équivalence qui approche au mieux une relation binaire symétrique, si cette dernière est une relation d'équivalence, le problème serait trivial. Dans ce cas, le problème de la maximisation du critère de Condorcet et celui de la minimisation du critère de Zahn sont effectivement strictement équivalents, comme le précisent Pierre Michaud et Jean François Marcotorchino dans leur article [44].

Dans notre travail, cependant, nous considérons séparément le critère de Zahn et le critère de Condorcet. En effet, la généralisation que nous proposons pour Zahn revient à pénaliser, de façon équivalente, la prise en compte d'une mise en relation entre deux éléments dans le partitionnement (alors que les deux éléments ne sont pas liés dans la relation à approcher), et la non prise en compte dans le partitionnement d'une mise en relation existant dans la relation à approcher. Cette généralisation est proposée en respectant l'esprit de la distance définie par Zahn.

Si l'on se base sur l'esprit dans lequel le critère de Condorcet a été construit, alors la généralisation naturelle ressemble plus à celle utilisée pour généraliser le critère de Newman et Girvan, c'est-à-dire en passant par les multigraphes et en approchant n'importe quel graphe par un multigraphe d'unité de débit réelle. C'est en effet le plus logique, puisque le critère de Condorcet est déjà défini pour les graphes à valeurs entières. Maintenant, si l'on choisit de considérer que le critère de Condorcet est une généralisation du critère de Zahn pour plusieurs relations symétriques pondérées, on pourra toujours se ramener à un problème de Zahn pour une seule relation symétrique pondérée. En effet, dans ce cas on considère simplement que dans l'équation (1.7), les matrices $\mathbf{C}^k$ peuvent représenter des relations symétriques pondérées, et donc être à valeurs réelles. Or, la somme de l'élément moyen de chacune de ces matrices est égale à l'élément moyen de la somme de ces matrices. Ainsi, la somme des matrices complémentaires $\overline{\mathbf{C}^k}$ sera égale à la matrice complémentaire de la somme si on utilise la généralisation proposée l'équation (1.4) – c'est-à-dire en considérant $\overline{c^k}_{i,j} = c^k_{mean} - c^k_{i,j}, \forall k$. Cela reviendra donc au même que de considérer directement que l'on ne s'intéresse qu'à une unique relation symétrique pondérée, définie par la matrice $\mathbf{C} = \sum_k \mathbf{C}^k$.

Impossibilité de généraliser

Dans le cas où l'on a une matrice $\mathbf{C} \in S_n(\mathbb{R}_+)$ et un partitionnement $\mathbf{X} \in Eq(n)$ proposé pour $\mathbf{C}$, cela fait-il sens de chercher à mesurer la similarité de ces deux matrices via le critère de Condorcet ?

Pour ce faire, il faut d'abord ramener $\mathbf{C}$ à un problème de Condorcet, c'est-à-dire trouver m matrices $\mathbf{C}_1, ..., \mathbf{C}_m \in Eq(n)$ – donc, symétriques, à valeurs dans $\{0, 1\}$, à diagonale pleine, et respectant la propriété de transitivité – telles que $\mathbf{C} = \sum_{k=1}^{m} \mathbf{C}_k$, afin de pouvoir calculer $\overline{\mathbf{C}}$.

Dans un premier temps, on va supposer $\mathbf{C} \in S_n(\mathbb{N})$. Dans ce cas, on remarque que, puisque $\mathbf{C}$ est une somme de m matrices représentant des relations d'équivalence, alors, étant donné la propriété de réflexivité imposée à une telle relation, on doit avoir $\forall i \in \{1, ..., n\}, c_{i,i} = m$. Or, cette propriété sur $\mathbf{C}$ est d'autant plus restrictive que, dans les faits, cette diagonale n'a aucun impact sur le fait qu'un partitionnement soit meilleur qu'un autre, conformément au critère de Condorcet. En effet, soit $\mathbf{C} \in S_n(\mathbb{N})$ une matrice relationnelle de Condorcet, somme de m relations d'équivalence, alors

$$\begin{aligned}\forall \mathbf{X} \in Eq(n), F_c(\mathbf{C}, \mathbf{X}) &= \sum_{i,j} \big(c_{i,j}x_{i,j} + (1 - c_{i,j})(1 - x_{i,j})\big) \\ &= \sum_{i \neq j} \big(c_{i,j}x_{i,j} + (1 - c_{i,j})(1 - x_{i,j})\big) + Tr(\mathbf{C}),\end{aligned}$$

étant donné que $\mathbf{X}$ est une relation d'équivalence, et qu'ainsi, $\forall i \in \{1, ..., n\}, x_{i,i} = 1$. Ainsi, pour deux partitionnements $\mathbf{X}^1, \mathbf{X}^2 \in Eq(n)$, on a :

$$\begin{aligned}F_C(\mathbf{C}, \mathbf{X}^1) - F_C(\mathbf{C}, \mathbf{X}^2) &= \sum_{i \neq j} \big(c_{i,j}x^1_{i,j} + \overline{c}_{i,j}\overline{x}^1_{i,j}\big) + Tr(\mathbf{C}) \\ &\quad - \Big(\sum_{i \neq j} \big(c_{i,j}x^2_{i,j} + \overline{c}_{i,j}\overline{x}^2_{i,j}\big) + Tr(\mathbf{C})\Big) \\ &= \sum_{i \neq j} \big(c_{i,j}(x^1_{i,j} - x^2_{i,j}) + \overline{c}_{i,j}(\overline{x}^1_{i,j} - \overline{x}^2_{i,j})\big)\end{aligned}$$

Cette différence ne dépend pas des élément diagonaux de $\mathbf{C}$. Ainsi les critères de Condorcet de deux matrices dans lesquelles seuls les éléments diagonaux différent auront les mêmes variations sur $Eq(n)$. Cela reste cohérent avec la remarque de l'article [44] qui précise que, par convention, on ne considère jamais la valeur des éléments diagonaux lorsque l'on définit la matrice relationnelle de Condorcet. Ainsi, pour simplifier, on supposera dans les calculs que nos éléments diagonaux sont nuls, tout en gardant à l'esprit qu'ils peuvent prendre n'importe quelle valeur fixée, sans que cela ne change le comportement du critère

de Condorcet.

Maintenant que nous avons relâché la contrainte sur la diagonale des matrices relationnelles de Condorcet, nous allons chercher à savoir si ce critère reste cohérent pour les matrices dans $S_n(\mathbb{N})$.

Pour une matrice $\mathbf{C} \in S_n(\mathbb{N})$, on cherche à décomposer cette matrice en matrices représentant des relations d'équivalence. C'est-à-dire que l'on cherche un jeu de matrices $\mathbf{C}_1, ..., \mathbf{C}_m \in Eq(n)$ telles que

$$\mathbf{C} - diag(\mathbf{C}) = \sum_{k=1}^{m} \mathbf{C}_k - m\mathbf{I}_n$$

Vérifions préalablement qu'une telle famille de matrices existe toujours :

Propriété 3. *Soit* $\mathbf{C} \in S_n(\mathbb{N})$*, alors*

$$\exists m \in \mathbb{N}, \exists (\mathbf{C}^k)_k \in (Eq(n))^m : \mathbf{C} - diag(\mathbf{C}) = \sum_{k=1}^{m} \mathbf{C}^k - m\mathbf{I}_n$$

Démonstration. On peut exhiber une famille triviale :
soit $\mathcal{J} = \{(i,j), i > j : c_{i,j} \neq 0\}$, on a $|\mathcal{J}| = nnz(\mathbf{C} - diag(\mathbf{C}))/2$, et on indexe les éléments de $\mathcal{J}$ dans l'ordre lexicographique. Ainsi, $\mathcal{J}_1$ représente le couple d'indices de l'élément non nul de la partie triangulaire inférieure de $\mathbf{C}$ ayant le plus petit indice de ligne, puis le plus petit indice de colonne correspondant possibles. On pose $m = \sum_{i>j} c_{i,j}$.

Alors on construit la famille $(\mathbf{C}^p)_{p=1...m}$ en prenant pour tout $k \in \{1, ..., |\mathcal{J}|\}$ la relation d'équivalence à $n-1$ classes, ne possédant que des classes singletons, sauf une classe qui lie les éléments i_k et j_k, avec $(i_k, j_k) = \mathcal{J}_k$, et en l'ajoutant $c_{i_k,j_k} \in \mathbb{N}$ fois à la famille. □

On remarque que cette décomposition n'est pas unique. Or, puisque le critère de Condorcet dépend du nombre de matrices d'équivalence dans la décomposition, il est nécessaire, pour pouvoir le généraliser, qu'il reste cohérent quelle que soit la décomposition choisie. Cela ne veut pas dire qu'il doit prendre les mêmes valeurs pour tous les partitionnements quelle que soit la décomposition choisie, mais que ses variations sur les partitionnements restent les mêmes. Or ce n'est pas le cas.

Voici un contre-exemple : on s'intéresse à la matrice $\mathbf{C} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}$ que l'on va décomposer de deux façons différentes :

$$\begin{array}{ccccccc} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} & = & \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} & + & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ \iff \mathbf{C} & = & \widetilde{\mathbf{C}}^1 & + & \widetilde{\mathbf{C}}^2 \end{array} \tag{1.8}$$

ou encore :

$$\begin{array}{ccccccccc} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} & = & \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & + & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} & + & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} & + & \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ \iff \mathbf{C} & = & \widehat{\mathbf{C}}^1 & + & \widehat{\mathbf{C}}^2 & + & \widehat{\mathbf{C}}^3 & + & \widehat{\mathbf{C}}^4 \end{array} \tag{1.9}$$

La matrice complémentaire de $\mathbf{C}$ associée à la décomposition de l'équation (1.8) est $\overline{\mathbf{C}} = 2\mathbf{J} - \mathbf{C}$, celle associée à la décomposition de l'équation (1.9) est $\overline{\mathbf{C}} = 4\mathbf{J} - \mathbf{C}$.

On prend deux partitionnements : $\mathbf{X}^1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ et $\mathbf{X}^2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$.

On a alors

$$F_C(\{\widetilde{\mathbf{C}}^p\}_{p=1}^2, \mathbf{X}^1) = 8 = F_C(\{\widetilde{\mathbf{C}}^p\}_{p=1}^2, \mathbf{X}^2).$$

Ainsi les deux partitionnements apparaissent donc comme équivalents, conformément à la décomposition associée à l'équation (1.8). Cependant :

$$F_C(\{\widehat{\mathbf{C}}^q\}_{q=1}^4, \mathbf{X}^1) = 8 < F_C(\{\widehat{\mathbf{C}}^q\}_{q=1}^4, \mathbf{X}^2) = 16.$$

Ainsi le partitionnement $\mathbf{X}^2$ apparaît donc comme meilleur conformément à la décomposition associée à l'équation (1.9).

En conclusion, le critère de Condorcet pour une matrice $\mathbf{C} \in S_n(\mathbb{N})$ dépend de la façon que l'on a de décomposer $\mathbf{C}$ en somme de relations d'équivalence. Cette décomposition n'étant pas unique, chercher à mesurer la similarité entre une matrice $\mathbf{C}$ et un partitionnement $\mathbf{X}$ via un critère de Condorcet ne fait sens que si cette matrice $\mathbf{C}$ provient initialement d'un problème de Condorcet. Nous ne nous intéresserons pas à ce critère dans la suite de notre étude.

1.1.4 Critère d'Owsinski et Zadrozny

Le critère qu'Owsinsky et Zadrozny décrivent dans leur article [43] est une généralisation du critère de Condorcet. A l'origine, il servait à résoudre des problèmes d'agrégation de préférence, c'est-à-dire avec des matrices relationnelles de Condorcet représentant une relation d'ordre – et donc non symétriques.

Dans la thèse de Patricia Conde-Céspedes [41], il a été démontré que, pour un graphe simple, le partitionnement maximisant le critère de Zahn est un partitionnement tel que dans chacune de ses classes, le nombre d'arêtes intra-classe est au moins égal à la moitié du nombre d'arêtes que possède le graphe complet des sommets de la classe. Cette propriété est assez restrictive, et le critère d'Oswinski et Zadrozny propose un critère semblable, mais avec un paramètre permettant de choisir le pourcentage d'arêtes intra-classes du meilleur partitionnement dans le cas d'un graphe simple. Ainsi, pour une matrice de Condorcet $\mathbf{A}$ et un partitionnement $\mathbf{X}$ donné,

$$F_{OZ}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \Big((1-\alpha) a_{i,j} x_{i,j} + \alpha \overline{a}_{i,j} \overline{x}_{i,j} \Big)$$

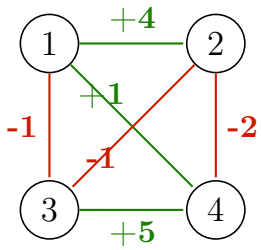
avec $\overline{\mathbf{A}}$ la matrice complémentaire de la matrice de Condorcet définie comme à la section précédente, et $\alpha \in [0, 1]$. Pour chaque classe du partitionnement maximisant ce critère, le nombre d'arêtes intra-classe est au moins $100\alpha\%$ du nombre d'arêtes du graphe complet correspondant.

Ce critère présente le même problème que le critère de Condorcet quant à sa généralisation : il est nécessaire de connaître le problème de Condorcet initial pour l'utiliser. Il est cependant possible d'envisager ce critère comme un critère de Zahn paramétré.

Ce choix du paramètre α correspond justement à un cas où l'utilisateur souhaite fixer la densité des communautés. Or, nous souhaitons ne pas supposer que nous avons une connaissance *a priori* sur l'aspect des communautés que nous souhaitons obtenir. Ce critère ne sera donc pas repris dans la suite de notre étude.

1.1.5 Critère de Demaine et Immorlica

Ce critère, introduit dans [60] et aussi appelé *Correlation Clustering*, s'applique sur des graphes pondérés à valeurs réelles d'un type différent de ceux que nous avons vus jusqu'à présent : une arête de poids positif représente une similarité entre les sommets qu'elle relie, similarité d'autant plus grande que le poids sera important. A l'inverse, un poids négatif sur une arête correspond à une dissimilarité entre les deux sommets correspondants. Naturellement, on veut mettre dans une même classe des sommets



$$\begin{aligned} E^+ &= \{(1,2),(1,4),(3,4))\} \\ &= \{(i,j) : a_{i,j} > 0\} \\ E^- &= \{(1,3),(2,3),(2,4)\} \\ &= \{(i,j) : a_{i,j} < 0\} \end{aligned}$$

$$F_{CC}(\mathbf{A},\mathbf{X}) = \sum_{(i,j)\in E^+} a_{i,j}x_{i,j} - \sum_{(i,j)\in E^-} a_{i,j}\overline{x}_{i,j}$$

FIGURE 1.7 : *Dans le graphe de gauche, les liens en vert représentent des similarités entre les noeuds qu'ils relient. L'ensemble de ces arêtes est noté E^+. Les arêtes en orange représentent des dissimilarités entre les noeuds qu'elles lient. L'ensemble de ces arêtes est noté E^-. Le critère de Correlation Clustering, donné en-dessous, compte le nombre d'arête positives intra-communautés et d'arêtes négatives inter-communautés.*

similaires : "maximizing agreement", et dans des classes différentes des sommets différents : "minimizing disagreement". Ainsi, pour une matrice $\mathbf{A} \in \mathcal{M}_n(\mathbb{R})$ et un partitionnement $\mathbf{X}$, le critère de *Correlation Clustering* F_{CC} mesure le nombre d'arêtes positives à l'intérieur des classes et le nombre d'arêtes négatives entre les classes. Un exemple est donnée Figure 1.7.

En posant

$$\forall i,j \in \{1,...,n\}, a^+_{i,j} = \begin{cases} a_{i,j} & \text{si } a_{i,j} > 0 \\ 0 & \text{sinon} \end{cases} \text{ et } a^-_{i,j} = \begin{cases} -a_{i,j} & \text{si } a_{i,j} < 0 \\ 0 & \text{sinon} \end{cases},$$

on peut ré-écrire la formule en bas de la Figure 1.7 comme :

$$\begin{aligned} F_{CC}(\mathbf{A},\mathbf{X}) &= \sum_{i,j}(a^+_{i,j}x_{i,j} + a^-_{i,j}\overline{x}_{i,j}) \\ &= \sum_{i,j}\left(a^+_{i,j}x_{i,j} + a^-_{i,j}(1-x_{i,j}\right) \\ &= \sum_{i,j}\left(a^+_{i,j} - a^-_{i,j}\right)x_{i,j} + \sum_{i,j} a^-_{i,j} \end{aligned}$$

La dernière somme de l'équation ci-dessus ne dépendant que de $\mathbf{A}$, Patricia Conde

Céspedes ré-écrit ce critère dans sa thèse comme :

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \left(a^+_{i,j} - a^-_{i,j}\right) x_{i,j},$$

ce qui permet d'arriver à la forme très simple :

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} a_{i,j} x_{i,j} \tag{1.10}$$

Le problème qui se pose alors pour l'utilisation de ce critère dans notre cas d'étude, c'est que nos matrices sont à valeurs dans $\mathbb{R}_+$. L'élément qui maximise le critère donné par l'équation (1.10) est alors trivialement le partitionnement ayant une seule classe regroupant tous les sommets. Cependant, nous envisageons trois façons de généraliser ce critère, qui dépendent essentiellement de l'interprétation que nous faisons des entrées nulles de la matrice creuse considérée. Dans la suite de cette section, on s'intéressera au critère associé à une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$.

Comme le plus fort cas de dissimilarité

On considère ici $\mathbf{A}$ comme une matrice pleine, les éléments nuls dans $\mathbf{A}$ étant donc considérés comme n'importe quelle autre entrée de $\mathbf{A}$. On peut "centrer" $\mathbf{A}$, c'est-à-dire appliquer le critère de Demaine et Immorlica comme défini dans [60], non pas sur $\mathbf{A}$ mais sur $\mathbf{A} - \frac{2m}{n^2}\mathbf{J}$, avec $2m = \sum_{i,j} a_{i,j}$. En faisant cela, on suppose que $\frac{2m}{n^2}$, l'élément moyen de $\mathbf{A}$, représente l'indifférence entre deux sommets. Les sommets de $\mathbf{A}$ liés par un poids plus grand vont être considérés comme similaires, et ceux liés par un poids plus petit, comme différents, y compris ceux liés par un poids nul – qui seront nécessairement les liens entre les sommets les plus différents. Dans ce cas, le critère devient

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \left(a_{i,j} - \frac{2m}{n^2}\right) x_{i,j} \tag{1.11}$$

ce qui correspond exactement au critère dit d'écart à l'uniformité présenté dans [61], que nous allons voir dans la suite.

Comme un manque d'information

On peut appliquer le raisonnement précédant tout en conférant à la partie creuse de $\mathbf{A}$ une caractéristique : il s'agit d'un manque d'information. C'est-à-dire que les entrées nulles de $\mathbf{A}$ sont les entrées pour lesquelles des données sont manquantes. L'élément moyen

ne doit alors pas dépendre de ces éléments nuls : il s'agira de $\frac{2m}{nnz}$, avec $2m$ défini comme précédemment, et nnz le nombre d'entrées non nulles dans $\mathbf{A}$. On note $G = (V, E, w)$ le graphe associé à $\mathbf{A}$, alors le critère de Demaine et Immorlica se généralise comme suit :

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{(i,j)\in E} \big(a_{i,j} - \frac{2m}{nnz}\big)x_{i,j} \tag{1.12}$$

Ici, les éléments de la structure creuse de $\mathbf{A}$ ne jouent aucun rôle, ni dans le calcul de l'élément moyen, ni dans la recherche du meilleur partitionnement, puisque l'on ne s'intéresse qu'aux couples d'indices dans E. Pour les autres arêtes, comme au point précédent, elles représentent une similarité quand leur poids est supérieur à l'élément moyen, et une dissimilarité sinon.

Comme le seul cas de dissimilarité

On considère maintenant que toute entrée strictement positive dans $\mathbf{A}$, aussi petite soit-elle, signifie qu'il existe une similarité entre les deux sommets concernés, et que les seuls cas de dissimilarité correspondent à la partie creuse de $\mathbf{A}$. Dans ce cas, la dissimilarité entre deux éléments n'est pas mesurée : elle existe ou non, mais on ne peut pas savoir si deux éléments sont peu ou très différents. On va donc conférer à cette dissimilarité une valeur arbitraire. Elle sera toujours la même puisqu'on ne dispose pas d'information complémentaire sur cette dissimilarité. Notons λ cette valeur. λ va dépendre de comment on veut rendre pénalisante la prise en compte d'une dissimilarité dans le partitionnement. On peut par exemple envisager $\lambda = -\max_{i,j}(a_{i,j})$, ou $\lambda = -\frac{2m}{nnz}$. Plus généralement, toute autre valeur négative peut être envisagée. En notant $G = (V, E, w)$ le graphe associé à $\mathbf{A}$, et quelle que soit la valeur choisie pour $\lambda < 0$, on aura :

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{(i,j)\in E} a_{i,j}x_{i,j} + \lambda \sum_{(i,j)\notin E} x_{i,j}. \tag{1.13}$$

En conclusion, pour ce critère, nous choisirons la généralisation proposée à l'équation (1.13) pour la suite de notre étude. En effet, choisir celle de l'équation (1.11) ne présente pas d'intérêt dans ce rapport puisque ce critère est strictement équivalent à un autre critère que nous allons étudier. Nous n'étudierons pas non plus en profondeur la généralisation proposée à l'équation (1.12), car dans le cas de matrices doublement stochastiques obtenues à partir des matrices d'adjacence de graphes simples, cette approche nous semble plus logique. En effet, dans un tel graphe, une arête entre deux sommets signifie que ces deux sommets sont en relation. Si deux sommets ne sont pas en relation – et sont donc

dissimilaires –, cela se traduit par l'absence de l'arête, et donc par un élément nul dans la matrice d'adjacence, ainsi que dans son équilibrage doublement stochastique. Il nous semble donc important de considérer les entrées nulles de nos matrices bi-stochastiques comme la caractéristique d'une dissimilarité entre des sommets.

Remarque 4. *Le récent papier de Veldt et al. [52] montre que nous ne sommes pas les seuls à nous intéresser à la généralisation de cette mesure dans le cas des graphes simples. Et nous verrons à la fin de ce chapitre que cela paraît cohérent car cette mesure est capable de produire de très bons résultats lorsqu'on l'utilise avec un choix judicieux de λ.*

1.1.6 Critère de modularité équilibrée

Ce critère a été introduit pour les graphes simples par Patricia Conde-Céspedes dans [61]. Comme son nom l'indique, il permet d'équilibrer le critère de Newman et Girvan : on rappelle qu'étant donné un graphe et un partitionnement, le critère de Newman et Girvan compare le nombre des arêtes intra-communautés du graphe avec ce même nombre d'arêtes dans un graphe aléatoire. Le critère de modularité équilibré compare en plus le nombre d'arêtes inter-communautés du graphe avec cette valeur dans le même graphe aléatoire.

Nous allons voir en détail comment est construit ce critère, puis nous proposerons une généralisation différente de celle décrite par Romain Campigotto *et al* dans [53] pour un graphe à pondération quelconque, avant d'adapter cette généralisation au cas spécifique des matrices bi-stochastiques.

Définition dans le cas d'un graphe simple

Soit $G = (V, E)$ un graphe simple, de matrice d'adjacence $\mathbf{A}$. On rappelle que le critère de Newman et Girvan associé à $\mathbf{A}$ mesure, pour un partitionnement donné, le pourcentage d'arêtes intra-communautés du graphe G par rapport au même pourcentage pour un graphe aléatoire, dont les degrés des sommets sont les mêmes que ceux du graphe G. Ainsi Newman et Girvan définissent l'expression "groupe de sommets fortement connectés" par rapport au nombre de connexions entre sommets du même groupe dans un graphe aléatoire. La définition de l'expression "groupes de sommets faiblement connectés" n'est alors qu'un corollaire, puisque le nombre total d'arêtes dans le graphe est fixé par la somme des degrés des sommets dans les deux graphes.

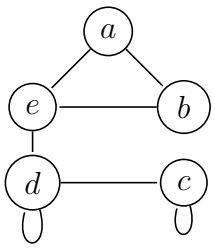
Le critère de modularité équilibrée redéfinit les deux expressions précédentes. En prenant explicitement en compte la non-existence d'un lien dans le graphe G, l'expression "sommets faiblement connectés' est accentuée par rapport au sens que lui donnent Newman

et Girvan. La formulation de ce critère est la suivante :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j}\Big(a_{i,j} - \frac{d_i d_j}{2m}\Big)x_{i,j} + \sum_{i,j}\Big(\overline{a}_{i,j} - \frac{(n-d_i)(n-d_j)}{n^2-2m}\Big)\overline{x}_{i,j}$$

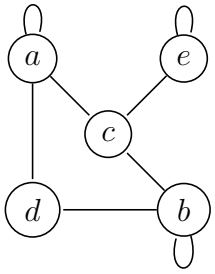
avec $\overline{\mathbf{A}} = \mathbf{J} - \mathbf{A}$, $\forall i \in \{1,...,n\}$, $d_i = \sum_{k=1}^{n} a_{i,k}$ est le degré du noeud i, et $2m = \sum_{i,j} a_{i,j}$ la somme des éléments de $\mathbf{A} \in S_n(\{0,1\})$.

Un exemple concret du complémentaire d'un graphe simple et de la matrice d'adjacence correspondante est donné Figure 1.8. Grâce à cette figure, il est très facile de remarquer que $\forall i \in \{1,...,n\}$, le degré du sommet i de la matrice complémentaire est donné par $\overline{d_i} = \sum_{k=1}^{n} \overline{a}_{i,k} = n - d_i$ et que la somme de ses éléments vaut $2\overline{m} = \sum_{i,j} \overline{a}_{i,j} = n^2 - 2m$.



$=$

0	1	0	0	1
1	0	0	0	1
0	0	1	1	0
0	0	1	1	1
1	1	0	1	0



$=$

1	0	1	1	0
0	1	1	1	0
1	1	0	0	1
1	1	0	0	0
0	0	1	0	1

FIGURE 1.8 : *En haut : un graphe simple et sa matrice d'adjacence. En bas : le complémentaire du graphe simple et sa matrice d'adjacence*

Soit la fonction Φ telle que :

$$\begin{array}{rccc} \Phi : & S_n(\{0,1\}) \times S_n(\{0,1\}) & \longrightarrow & \mathbb{R} \\ & (\mathbf{A}, \mathbf{X}) & \longmapsto & \sum\limits_{i,j} \Big(a_{i,j} - \frac{d_i d_j}{2m}\Big) x_{i,j} \end{array}$$

On remarque que, si $\mathbf{X} \in Eq(n)$, on retombe sur la modularité classique. Alors le critère de modularité équilibrée peut se mettre sous la forme suivante pour une matrice $\mathbf{A} \in S_n(\{0,1\})$, et un partitionnement $\mathbf{X} \in Eq(n)$ donné :

$$F_{BM}(\mathbf{A}, \mathbf{X}) = \Phi(\mathbf{A}, \mathbf{X}) + \Phi(\overline{\mathbf{A}}, \overline{\mathbf{X}}) \tag{1.14}$$

Cette formulation permet, dans le cas d'un graphe simple, de mettre en évidence le fait que la mesure de modularité équilibrée définisse de façon équivalente les expressions "sommets faiblement connectés" et "sommets fortement connectés".

Enfin, on remarque que, pour un graphe simple dont $\mathbf{A}$ est la matrice d'adjacence, on a $\mathbf{A} + \overline{\mathbf{A}} = \mathbf{J}$. C'est-à-dire que la superposition des deux graphes, *id est* leur somme ou la somme de leurs matrices d'adjacence, correspond à un graphe complet.

Généralisation aux graphes pondérés

Pour un graphe pondéré quelconque et sa matrice d'adjacence $\mathbf{A} \in S_n(\mathbb{R}_+)$, la généralisation proposée par R. Campigotto *et al* dans [53] consiste à prendre

$$\overline{\mathbf{A}} = \underset{i,j}{max}(a_{i,j})\mathbf{J} - \mathbf{A}, \tag{1.15}$$

ce qui revient à considérer que $\underset{i,j}{max}(a_{i,j})$ est le poids maximum pouvant être atteint par une arête.

La généralisation complète du critère de modularité équilibrée pour une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ et un partitionnement $\mathbf{X} \in Eq(n)$ proposée dans cet article est la suivante :

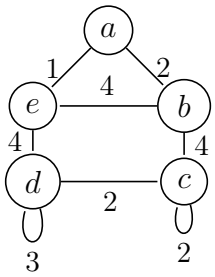
$$F_{BM}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{d_i d_j}{2m}\Big) x_{i,j} + \sum_{i,j} \Big(\overline{a}_{i,j} - \frac{(n-d_i)(n-d_j)}{n^2 - 2m}\Big) \overline{x}_{i,j} \tag{1.16}$$

avec $d_i = \sum\limits_{k=1}^{n} a_{i,k}$ et $2m = \sum\limits_{i,j} a_{i,j}$.

Nous proposons ici une autre généralisation, qui permet de rester proche de la formulation proposée à l'équation (1.14), en généralisant Φ aux matrices dans $S_n(\mathbb{R}_+)$.

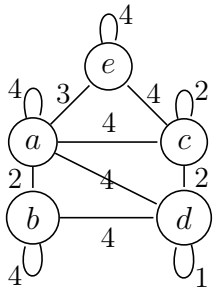
Nous définissons dans un premier temps la matrice complémentaire de $\mathbf{A}$ comme

à l'équation (1.15). Alors, pour un découpage $\mathbf{X} \in Eq(n)$ donné, pour que $\Phi(\overline{\mathbf{A}}, \overline{\mathbf{X}})$ corresponde à la différence des pourcentages d'arêtes inter-communautés de la matrice complémentaire et d'un graphe aléatoire ayant la même distribution des degrés que $\overline{\mathbf{A}}$, il faut prendre en compte le fait qu'avec la matrice complémentaire de $\mathbf{A}$ définie à l'équation (1.15), le degré du sommet i dans le graphe complémentaire est $\overline{d_i} = \sum_k \overline{a_{i,k}} = \max_{i,j}(a_{i,j})n - d_i$, et que la somme des éléments de la matrice complémentaire vaut $2\overline{m} = \sum_{i,j} \overline{a_{i,j}} = \max_{i,j}(a_{i,j})n^2 - 2m$. Pour s'en assurer, un exemple de graphe pondéré et de son complémentaire est donné Figure 1.9.



$=$

0	2	0	0	1
2	0	4	0	4
0	4	2	2	0
0	0	2	3	4
1	4	0	4	0



$=$

4	2	4	4	3
2	4	0	4	0
4	0	2	2	4
4	4	2	1	0
3	0	4	0	4

FIGURE 1.9 : *En haut : un graphe pondéré et sa matrice d'adjacence. En bas : le complémentaire du graphe pondéré comme défini à l'équation (1.15) et sa matrice d'adjacence*

Dans ce cas, on a

$$\Phi(\overline{\mathbf{A}}, \overline{\mathbf{X}}) = \sum_{i,j} \left(\overline{a}_{i,j} - \frac{(\max_{i,j}(a_{i,j})n - d_i)(\max_{i,j}(a_{i,j})n - d_j)}{\max_{i,j}(a_{i,j})n^2 - 2m} \right) \overline{x}_{i,j}$$

et notre proposition de généralisation pour le critère de modularité équilibrée devient :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j}\Big(a_{i,j} - \frac{d_i d_j}{2m}\Big)x_{i,j} + \sum_{i,j}\left(\overline{a}_{i,j} - \frac{(\underset{i,j}{max}(a_{i,j})n - d_i)(\underset{i,j}{max}(a_{i,j})n - d_j)}{\underset{i,j}{max}(a_{i,j})n^2 - 2m}\right)\overline{x}_{i,j}$$

On remarque que l'on a par ailleurs $\mathbf{A} + \overline{\mathbf{A}} = \underset{i,j}{max}(a_{i,j})J$, que l'on peut envisager comme la généralisation de la notion de graphe complet dans le cas pondéré : dans le graphe associé à cette matrice, toutes les arêtes sont présentes, et de poids maximal.

Nous souhaitons maintenant étudier ce qui se passe dans le cas où la matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ est bi-stochastique. Bien que l'on puisse utiliser la généralisation précédente, les matrices bi-stochastiques possèdent une borne supérieure naturelle pour le poids pouvant être atteint par une arête : c'est 1. Il nous semble alors naturel de plutôt choisir comme matrice complémentaire de $\mathbf{A}$ la matrice $\overline{\mathbf{A}} = \mathbf{J} - \mathbf{A}$, de manière générique, pour toute matrice bi-stochastique. En effet, si $\mathbf{A}$ est une matrice bi-stochastique, c'est cette matrice complémentaire qui, ajoutée à $\mathbf{A}$, permet de saturer toutes les arêtes, puisque l'on a $\mathbf{A} + \overline{\mathbf{A}} = \mathbf{J}$. Dans ce cas, pour un partitionnement $\mathbf{X} \in Eq(n)$ donné, notre proposition de généralisation du critère de modularité équilibrée dans le cas des matrices bi-stochastiques devient :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j}\Big(a_{i,j} - \frac{d_i d_j}{2m}\Big)x_{i,j} + \sum_{i,j}\Big(\overline{a}_{i,j} - \frac{(n-d_i)(n-d_j)}{n^2 - 2m}\Big)\overline{x}_{i,j}$$

Or puisque $\mathbf{A}$ est bi-stochastique, on a $\forall i, d_i = 1$ et $2m = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j} = \sum_{i=1}^{n} 1 = n$. Ainsi, après simplification, notre formulation de F_{BM} se réécrit :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j}\Big(a_{i,j} - \frac{1}{n}\Big)x_{i,j} + \sum_{i,j}\Big(\overline{a}_{i,j} - \frac{(n-1)}{n}\Big)\overline{x}_{i,j} \tag{1.17}$$

On remarque que, pour une matrice bi-stochastique, en choisissant la même matrice complémentaire $\overline{\mathbf{A}} = \mathbf{J} - \mathbf{A}$, les critères de modularité équilibrée généralisés par l'équation (1.16) et l'équation (1.17) sont égaux.

En conclusion, nous avons proposé une généralisation de la mesure de modularité équilibrée différente de celle trouvée dans la littérature. En outre, dans cette formulation, la définition de la matrice complémentaire, si elle convient dans un cas général, peut être modifiée lorsque l'on travaille sur des matrices spécifiques, et notamment quand on connaît le poids de saturation des arêtes, comme c'est le cas pour les matrices bi-stochastiques. Pour les matrices bi-stochastiques, on remarque aussi que si l'on applique cette définition

de la matrice complémentaire dans la généralisation de Romain Campigotto *et al.*, les deux généralisations sont alors strictement équivalentes.

1.1.7 Critère d'écart à l'indétermination

Ce critère, introduit par P. Conde-Céspedes dans [61], est largement développé dans sa thèse [41]. Il se base sur la notion de tableau de contingence croisant deux variables catégorielles. On peut définir simplement une variable catégorielle comme un ensemble de propriétés ou modalités incompatibles – c'est-à-dire qu'un objet possède une et une seule des propriétés d'une variable. Supposons que ces deux variables catégorielles – notées $\mathbf{F}$ et $\mathbf{G}$ – possèdent p, respectivement q, modalités. Supposons en outre que l'on ait un ensemble de J objets auxquels on a attribué les propriétés de $\mathbf{F}$ et $\mathbf{G}$. Alors le tableau de contingence croisant $\mathbf{F}$ et $\mathbf{G}$ est défini par la matrice $\mathbf{N} = (n_{u,v}) \in \mathcal{M}_{p,q}(\mathbb{R}_+)$ telle que $\forall u \in \{1, ..., p\}, \forall v \in \{1, ..., q\}$, $n_{u,v}$ est le nombre d'objets de l'ensemble qui vérifient à la fois la propriété u de la variable $\mathbf{F}$ et la propriété v de la variable $\mathbf{G}$. Les notations $n_{u,.}$, respectivement $n_{.,v}$, correspondent au nombre d'objets de l'ensemble vérifiant la propriété u de la variable $\mathbf{F}$, respectivement la propriété v de la variable $\mathbf{G}$. En d'autres termes, $n_{u,.} = \sum_{t=1}^{q} n_{u,t}$ et $n_{.,v} = \sum_{s=1}^{p} n_{s,v}$.

Etant donnés ces paramètres, on parle d'indétermination parfaite entre les deux variables catégorielles $\mathbf{F}$ et $\mathbf{G}$ lorsque

$$\forall (u, v), n_{u,v} = \frac{n_{u,.}}{q} + \frac{n_{.,v}}{p} - \frac{J}{pq}$$

Pour créer son critère d'écart à l'indétermination, Patricia Condé-Céspedes envisage une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ comme un tableau de contingence. Dans ce cas, les deux variables catégorielles associées à ce tableau ont nécessairement un même nombre de modalités n, et ces variables catégorielles ont été testées sur $2m = \sum_{i,j} a_{i,j}$ objets. L'indétermination parfaite entre ces deux variable s'écrit alors :

$$\begin{aligned} \forall (i, j), a_{i,j} =& \frac{a_{i,.}}{n} + \frac{a_{.,j}}{n} - \frac{2m}{n^2} \\ =& \frac{d_i}{n} + \frac{d_j}{n} - \frac{2m}{n^2} \end{aligned}$$

avec d_k le degré du noeud k dans le graphe associé à la matrice $\mathbf{A}$.

C'est à partir de cette observation que P. Conde-Céspedes dérive un critère qu'elle

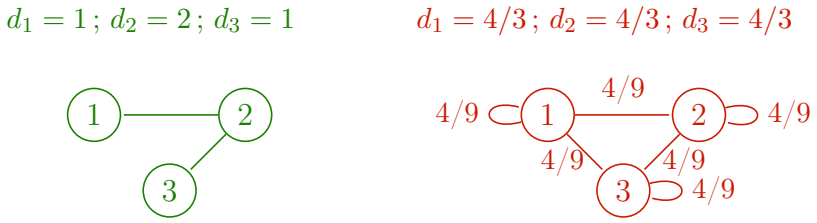


FIGURE 1.10 : *A gauche : un graphe simple. A droite : le graphe aléatoire ayant le même nombre d'arêtes que le graphe de gauche, et une distribution uniforme des degrés. La distribution des degrés est donné au dessus des deux graphes.*

appelle critère d'écart à l'indétermination, et qui est défini comme suit :

$$F_{DI}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{d_i}{n} - \frac{d_j}{n} + \frac{2m}{n^2} \Big) x_{i,j}.$$

Ce critère étant initialement défini sur les matrices symétriques à coefficient dans $\mathbb{R}_+$, il n'est pas nécessaire de le généraliser.

1.1.8 Critère d'écart à l'uniformité

Le critère d'écart à l'uniformité, introduit par P. Conde-Céspedes dans [61], mesure, pour un partitionnement donné, l'écart entre le pourcentage d'arêtes intra-communautés dans le graphe étudié et le pourcentage d'arêtes intra-communautés dans le graphe uniforme associé, ayant le même nombre d'arêtes. Si l'on note m le nombre d'arêtes et n le nombre de sommets dans le graphe étudié, le graphe uniforme associé correspond au graphe tel que chaque sommet a un degré égal à $\frac{2m}{n^2}$, comme illustré Figure 1.10.

Ainsi, pour une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ et un partitionnement $\mathbf{X} \in Eq(n)$, on a simplement

$$F_{DU}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{2m}{n^2} \Big) x_{i,j}$$

avec $2m = \sum_{i,j} a_{i,j}$.

La valeur $\frac{2m}{n^2}$ est appelée densité d'arêtes. Elle est positive, et nécessairement inférieure à 1 dans le cas d'un graphe simple. Dans le cas des graphes pondérés, cette valeur peut être supérieure à 1. Si l'on considère une matrice pleine – et donc un graphe complet pondéré, éventuellement avec certaines arêtes de poids nul – cette valeur est égale au poids moyen d'une arête.

1.2 Comparaison de critères dans le cas des matrices bi-stochastiques

Cette section est dédiée à la comparaison des différents critères présentés à la Section 1.1, lorsque le graphe étudié est un graphe de transition, c'est-à-dire que la matrice des poids est stochastique – et donc bi-stochastique, puisque l'on s'intéresse à des graphes non orientés, et donc à des matrices symétriques.

Dans toute cette section, nous étudions une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ bi-stochastique. Nous commençons par simplifier et exprimer les différents critères sous une forme canonique permettant une comparaison plus aisée. Ensuite, nous comparons les différences entre les propriétés des partitionnements favorisant ces critères.

1.2.1 Rappel et homogénéisation des critères

Nous allons ici rappeler les critères vus à la section précédente, et les mettre sous forme d'un fonction à maximiser. En d'autres termes, lorsque le meilleur partitionnement est obtenu par minimisation du critère, nous allons modifier ce critère, afin que le meilleur partitionnement soit systématiquement obtenu par la maximisation du critère. En outre, nous allons faire les simplifications qui interviennent étant donné que la matrice étudiée est bi-stochastique.

Comme cela est proposé dans le Chapitre 6 de la thèse de Patricia Conde-Céspedes [41], nous ré-écrirons ces critères sous la forme canonique

$$F(\mathbf{A}, \mathbf{X}) = \sum_{i,j} (\phi_{i,j} - \overline{\phi}_{i,j}) x_{i,j} \tag{1.18}$$

Les coefficients $\phi_{i,j}$ et $\overline{\phi}_{i,j}$ sont appelés respectivement terme d'accord positif et terme d'accord négatif. La raison de ce choix est que la comparaison des critères deux à deux est plus aisée via la comparaison de ces termes d'accord positifs et négatifs.

Critère de Zahn

On rappelle que l'on a généralisé le critère de Zahn en posant :

$$\begin{array}{rccc} d_Z(\mathbf{A}, .) : & Eq(n) & \longrightarrow & \mathbb{R} \\ & \mathbf{X} & \longmapsto & \frac{1}{2} \sum_{i,j} \left(\overline{a}_{i,j} x_{i,j} + a_{ij} \overline{x}_{i,j} \right) \end{array}$$

avec

$$\overline{a}_{i,j} = mean(a_{p,q}) - a_{i,j} = \frac{\sum_{p=1}^{n}\sum_{q=1}^{n} a_{p,q}}{n^2} - a_{i,j} = \frac{\sum_{p=1}^{n} 1}{n^2} - a_{i,j} = \frac{1}{n} - a_{i,j}$$

vu que $\mathbf{A}$ bi-stochastique.
Ainsi, on a $\forall \mathbf{A} \in S_n(\mathbb{R}_+)$ bi-stochastique et $\forall \mathbf{X} \in Eq(n)$:

$$\begin{aligned} d_Z(\mathbf{A},\mathbf{X}) &= \tfrac{1}{2}\sum_{i,j}\Big(\Big(\tfrac{1}{n} - a_{i,j}\Big)x_{i,j} + a_{i,j}(1-x_{i,j})\Big) \\ &= \tfrac{1}{2}\Big(\sum_{i,j}\Big(\tfrac{1}{n} - 2a_{i,j}\Big)x_{i,j} + \sum_{i,j} a_{i,j}\Big) \\ &= \sum_{i,j}\Big(\tfrac{1}{2n} - a_{i,j}\Big)x_{i,j} - \tfrac{n}{2} \end{aligned}$$

Or, $\dfrac{n}{2}$ ne dépend pas de $\mathbf{X}$, et, $\forall \mathbf{X}_1, \mathbf{X}_2 \in Eq(n)$, on a

$$d_Z(\mathbf{A},\mathbf{X}_1) - d_Z(\mathbf{A},\mathbf{X}_2) = \big(d_Z(\mathbf{A},\mathbf{X}_1) + \frac{n}{2}\big) - \big(d_Z(\mathbf{A},\mathbf{X}_2) + \frac{n}{2}\big)$$

Observer les variations de d_Z est donc équivalent à observer les variations de $d_Z + \frac{n}{2}$, et le critère de Zahn initial peut donc être remplacé par une version plus simple car sans constante :

$$d_Z(\mathbf{A},\mathbf{X}) = \sum_{i,j}\left(\frac{1}{2n} - a_{i,j}\right)x_{i,j}$$

En outre, pour ce critère, trouver un bon partitionnement $\mathbf{X} \in Eq(n)$ étant donnée une matrice $\mathbf{A} \in S_n(\mathbb{R}_+)$ revient à minimiser ce critère. Or, dans la forme canonique proposée à l'équation (1.18), on a besoin qu'un bon partitionnement soit obtenu par maximisation du-dit critère. On va donc s'intéresser à l'opposé de cette fonction, autrement dit à la fonction

$$\begin{array}{rccc} \Psi: & S_n(\mathbb{R}^+) \times Eq(n) & \longrightarrow & \mathbb{R} \\ & \mathbf{A},\mathbf{X} & \longmapsto & \displaystyle\sum_{i,j}\left(a_{i,j} - \tfrac{1}{2n}\right)x_{i,j} \end{array}$$

On a $(\mathbf{A},\mathbf{X}) \mapsto \Psi(\mathbf{A},\mathbf{X}) = -d_Z(\mathbf{A},\mathbf{X})$. Ainsi, observer les variations de $d_Z(\mathbf{A},.)$ sur $Eq(n)$ est strictement équivalent à observer les variations de $\Psi(\mathbf{A},.)$ sur $Eq(n)$.

Dans le reste de ce chapitre, afin d'avoir une mesure de similarité la plus simple possible pour le critère de Zahn, nous noterons F_Z la fonction Ψ. Autrement dit, dans la suite de ce chapitre, on a

$$F_Z(\mathbf{A},\mathbf{X}) = \sum_{i,j}\left(a_{i,j} - \frac{1}{2n}\right)x_{i,j}.$$

Critère de Newman et Girvan

Le critère de Newman et Girvan est déjà un critère à maximiser. Pour un partitionnement $\mathbf{X} \in Eq(n)$ donné, il a été défini à l'équation (1.5) comme :

$$F_{NG}(\mathbf{A}, \mathbf{X}) = \frac{1}{2m} \sum_{i,j} \Big(a_{i,j} - \frac{d_i d_j}{2m} \Big) x_{i,j}$$

Or, dans le cas d'une matrice bi-stochastique, on a $2m = \sum_{i,j} a_{i,j} = n$ et $\forall i, d_i = \sum_{k=1}^{n} a_{i,k} = 1$. Donc finalement, le critère de Newman se ré-écrit :

$$F_{NG}(\mathbf{A}, \mathbf{X}) = \frac{1}{n} \sum_{i,j} \Big(a_{i,j} - \frac{1}{n} \Big) x_{i,j}.$$

Puisque les variations du critère ne sont pas impactées par la multiplication du critère par un scalaire positif, on peut formuler le critère de Newman et Girvan suivant la forme de l'équation (1.18) :

$$F_{NG}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{1}{n} \Big) x_{i,j}.$$

Critère de Demaine et Immorlica

Nous choisissons une des généralisations proposées à la Section 1.1.5 pour ce critère :

$$F_{CC}(\mathbf{A}, \mathbf{X}) = \sum_{(i,j) \in E} a_{i,j} x_{i,j} - \lambda \sum_{(i,j) \notin E} x_{i,j}$$

avec $E \subset \{1, ..., n\} \times \{1, ..., n\}$ l'ensemble des entrées non nulles de $\mathbf{A}$, et $\lambda \in \mathbb{R}_+^*$ une valeur qui correspond à la pénalisation de la prise en compte dans $\mathbf{X}$ d'un élément de la structure creuse de $\mathbf{A}$. Cette équation n'admet par ailleurs aucune simplification liée à la propriété bi-stochastique de $\mathbf{A}$.

On peut par contre remarquer que, naturellement, $\forall (i,j) \notin E, a_{i,j} = 0$. Ainsi, en notant

$$\begin{array}{rccc} \mathcal{X}_{\overline{E}} : & \{1, ..., n\} \times \{1, ..., n\} & \longrightarrow & \{0, 1\} \\ & (i,j) & \longmapsto & \begin{cases} 1 \text{ si } (i,j) \in \overline{E} \\ 0 \text{ sinon} \end{cases} \end{array}$$

la fonction caractéristique de la partie creuse de $\mathbf{A}$, on peut mettre ce critère sous la

forme de l'équation (1.18) :

$$F_{CC}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \big(a_{i,j} - \lambda \mathcal{X}_{\overline{E}}(i,j)\big) x_{i,j}.$$

Critère de modularité équilibrée

Ce critère a déjà été développé dans le cas d'une matrice bi-stochastique. En complétant les simplifications de l'équation (1.17) , on obtient la forme de droite :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{1}{n}\Big) x_{i,j} + \sum_{i,j} \Big((1-a_{i,j}) - \frac{n-1}{n}\Big)(1-x_{i,j}) = 2\sum_{i,j} \big(a_{i,j} - \frac{1}{n}\big) x_{i,j}$$

et l'on peut supprimer le coefficient multiplicateur pour obtenir une formulation du critère qui coïncide avec l'équation (1.18) :

$$F_{BM}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \big(a_{i,j} - \frac{1}{n}\big) x_{i,j}.$$

Critère d'écart à l'indétermination

Ce critère a été défini de la façon suivante :

$$F_{DI}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{d_i}{n} - \frac{d_j}{n} + \frac{2m}{n^2}\Big) x_{i,j}$$

avec $d_i = \sum_{k=1}^{n} a_{i,k}$ et $2m = \sum_{i,j} a_{i,j}$. Dans le cas d'une matrice bi-stochastique, on a $\forall i, d_i = 1$ et $2m = n$, donc le critère se simplifie comme suit :

$$F_{DI}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{1}{n}\Big) x_{i,j}.$$

Critère d'écart à l'uniformité

Cette mesure de modularité correspond, pour un partitionnement $\mathbf{X}$ donné, à l'équation :

$$F_{DU}(\mathbf{A},\mathbf{X}) = \sum_{i,j} \Big(a_{i,j} - \frac{2m}{n^2}\Big) x_{i,j}.$$

Critères dans le cas stochastique	**Équations sous la forme** $\sum_{i,j}\Big(\phi_{i,j}-\overline{\phi}_{i,j}\Big)x_{i,j}$
Zahn	$F_Z(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\frac{1}{2n}\Big)x_{i,j}$
Newman et Girvan	$F_{NG}(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\frac{1}{n}\Big)x_{i,j}$
Demaine et Immorlica	$F_{CC,\lambda}(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\lambda\mathcal{X}_{\overline{E}}(i,j)\Big)x_{i,j}$, avec $\lambda>0$
Newman équilibré	$F_{BM}(\mathbf{A},\mathbf{X})=\sum_{i,j}\big(a_{i,j}-\frac{1}{n}\big)x_{i,j}$
Écart à l'indétermination	$F_{DI}(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\frac{1}{n}\Big)x_{i,j}$
Écart à l'uniformité	$F_{DU}(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\frac{1}{n}\Big)x_{i,j}$

TABLE 1.1 : *Table des différents critères de modularité.*

Comme précédemment, **A** étant bi-stochastique, on obtient :

$$F_{DU}(\mathbf{A},\mathbf{X})=\sum_{i,j}\Big(a_{i,j}-\frac{1}{n}\Big)x_{i,j}.$$

En résumé

Tous les critères, dans le cas de matrices bi-stochastiques et, mis sous la forme de l'équation (1.18), sont listés dans la Table 1.1.

1.2.2 Comparaison des critères

Grâce à la Table 1.1, on remarque que pour tous les critères, le terme d'accord positif est le même : $a_{i,j}$, ce qui est assez logique. En effet, il est cohérent que la mise en relation de deux éléments via le partitionnement améliore le critère de façon proportionnelle à la valeur du poids du lien existant entre ces deux éléments. Nous allons donc nous intéresser aux termes d'accord négatif, pour tenter de comprendre le comportement des différents critères sur l'ensemble des partitionnements.

On notera $\overline{\phi}^{NG}_{i,j}=\overline{\phi}^{NG}=\dfrac{1}{n}$ le terme d'accord négatif pour le critère de Newman et Girvan, $\overline{\phi}^{Z}_{i,j}=\overline{\phi}^{Z}=\dfrac{1}{2n}$ celui du critère de Zahn, et $\overline{\phi}^{\lambda}_{i,j}=\lambda\mathcal{X}_{\overline{E}}(i,j)$ celui du critère de Demaine et Immorlica.

Remarque 5. *Le critère d'Owsinski et Zadrozny, que nous avons présenté Section 1.1.4 et choisi de ne pas conserver pour la suite de notre étude, aurait pu être envisagé comme un critère de Zahn paramétré. Dans ce cas, avec la forme canonique de l'équation (1.18) et en considérant une matrice* $\mathbf{A} \in S_n(\mathbb{R}_+)$ *bi-stochastique, on aurait eu*

$$F_{OZ}(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \left(a_{i,j} - \frac{\alpha}{n} \right) x_{i,j},$$

soit finalement un critère comme ceux de la Table 1.1, avec un terme d'accord négatif à faire varier dans $]0, 1/n[$.

Une Première remarque

Grâce à leur forme dans la Table 1.1, on peut remarquer que

$$F_{DI} = F_{DU} = F_{BM} = F_{NG}.$$

Ces quatre critères ont donc exactement le même comportement sur les partitionnements. Cette remarque fait écho au Théorème 6.1 de la thèse de Patricia Condé-Cespedes [41], dans lequel elle affirme que ces quatre mesures sont équivalentes dans le cas de graphes réguliers, c'est-à-dire de graphes dont tous les sommets ont le même degré. Or, toute matrice bi-stochastique est en particulier un graphe 1-régulier, puisque la somme des éléments de chacune de ses lignes – ce qui correspond au degré de chacun des sommets – est égale à 1. Notons que le Théorème 6.1 de la thèse de Patricia Condé-Cespedes est démontré dans le cas des graphes non pondérés. La Table 1.1 permet une généralisation de ce théorème dans le cas des matrices doublement stochastiques, pondérées.

Partitionnements optimaux

Dans cette partie, nous discutons les différences entre les propriétés des partitionnements optimisant les différents critères. Nous observons ensuite ces propriétés sur de petites matrices. Nous commençons par présenter et rappeler quelques remarques. Soit $\mathbf{A} \in S_n(\mathbb{R})$ doublement stochastique, et $F_{CC,\lambda}(\mathbf{A}, .), F_Z(\mathbf{A}, .)$ et $F_{NG}(\mathbf{A}, .)$ les critères respectivement de Demaine et Immorlica, de Zahn et de Newman et Girvan associés à $\mathbf{A}$.

Remarque 6. *En comparant les termes d'accord positifs et négatifs, on remarque que :*

1. *Ces trois critères ont le même terme d'accord positif,*
2. $$\overline{\phi}^{NG} = \frac{1}{n} > \overline{\phi}^{Z} = \frac{1}{2n},$$

3.
$$\forall \lambda < \frac{1}{n}, \forall (i,j) \in \{1...n\}^2, \overline{\phi}^{\lambda}_{i,j} < \overline{\phi}^{NG}_{i,j} = \overline{\phi}^{NG},$$

4.
$$\forall \lambda < \frac{1}{2n}, \forall (i,j) \in \{1...n\}^2, \overline{\phi}^{\lambda}_{i,j} < \overline{\phi}^{Z}_{i,j} = \overline{\phi}^{Z},$$

5.
$$\forall \lambda_1 > \lambda_2 > 0, \forall (i,j) \in \{1,...,n\}^2, \overline{\phi}^{\lambda_1}_{i,j} \geq \overline{\phi}^{\lambda_2}_{i,j}.$$

Remarque 7. *En notant* $\mathbf{X}^{NG} = \underset{\mathbf{X} \in Eq(n)}{argmax} F_{NG}(\mathbf{A}, \mathbf{X})$, *respectivement* $\mathbf{X}^{Z} = \underset{\mathbf{X} \in Eq(n)}{argmax} F_{Z}(\mathbf{A}, \mathbf{X})$, *on a*

$$\sum_{i,j} x^{NG}_{i,j} \leq \sum_{i,j} x^{Z}_{i,j}.$$

Autrement dit, le partitionnement optimisant le critère de Newman et Girvan présente moins de relations que le partitionnement maximisant le critère de Zahn. Ce résultat est démontré sous des hypothèses plus générales dans l'Annexe A.1.

Remarque 8. *Si* $\lambda > n/2$, *alors le partitionnement optimisant* $F_{CC,\lambda}$ *:*

$$\mathbf{X}^{\lambda} = \underset{\mathbf{X} \in Eq(n)}{argmax} F_{CC,\lambda}(\mathbf{A}, \mathbf{X})$$

n'admet aucun élément dans la structure creuse de $\mathbf{A}$. *La démonstration de ce résultat est donnée en Annexe A.2. Cela donne la borne supérieure :*

$$\sum_{i,j} x^{\lambda}_{i,j} \leq nnz(\mathbf{A} + \mathbf{I}_n)$$

Après avoir énoncé ces trois remarques, nous notons qu'il est cependant relativement complexe d'obtenir plus de résultats généraux sur les partitionnements optimisant les différents critères. En effet, il est aisé d'envisager que pour deux critères F_1 et F_2 ayant un même terme d'accord positif, et tel que leurs termes d'accord négatif respectent

$$\overline{\phi}^{F_1}_{i,j} > \overline{\phi}^{F_2}_{i,j} \geq 0, \forall i,j,$$

la prise en compte d'une relation entre deux éléments i et j dans un partitionnement sera d'autant plus pénalisée dans le critère F_1 par rapport au critère F_2, que $\overline{\phi}^{F_1}_{i,j}$ est grand devant $\overline{\phi}^{F_2}_{i,j}$.

Cependant l'impact sur un critère de la prise en compte d'une relation (i,j) dans un partitionnement ne dépend pas que des termes d'accord positifs et négatifs du critère sur

(i, j), puisque le fait d'activer ce lien dans le partitionnement active en général d'autres liens. En effet, tous les éléments liés à i vont devoir aussi être liés à j, *et vice et versa*, pour que le partitionnement reste une relation d'équivalence.

En outre, il peut aussi être intuitif de penser que, si un partitionnement $\mathbf{X}^1 \in Eq(n)$ possède plus de relations qu'un autre partitionnement $\mathbf{X}^2 \in Eq(n)$, *id est* :

$$\sum_{i,j} x^1_{i,j} \geq \sum_{i,j} x^2_{i,j},$$

il possède moins de classes – qui sont alors de plus grandes tailles. Ce résultat n'est pas non plus vrai dans le cas général, comme le montre le contre-exemple de l'Annexe A.2.

Il est donc assez compliqué de comparer les comportements des critères sur tel ou tel partitionnement dans le cas général. A notre connaissance, assez peu de résultats sur l'aspect des partitionnements optimisant les différents critères ont été établis sans supposer un nombre donné de classes, ou un type particulier de graphes. Brandes *et al.* ont par contre démontré dans [57] que pour la matrice d'adjacence $\mathbf{A}$ d'un graphe G, un entier $k \geq 2$ et un réel η, le problème de décision :

"`Trouver` $\mathbf{X} \in Eq(n)$ `présentant au plus` k `classes, tel que` $F_{NG}(\mathbf{A}, \mathbf{X}) \geq \eta$"

était "strongly NP-Complete". Cela laisse envisager la difficulté à obtenir des résultats théoriques sur le comportement général de ce critère – et des autres critères qui ne se formulent pas plus simplement que celui de Newman et Girvan. En outre, l'analyse est encore compliquée par le fait que l'on n'a pas de moyen de connaître le maximum de modularité pouvant être atteint par un partitionnement étant donné un graphe.

Cependant, ces critères ne sont pas construits sur des graphes complètement quelconques, mais sur des matrices supposées présenter une structure avec des communautés. De ce fait, on attend des partitionnements optimisant les critères qu'ils coïncident avec la structure par bloc de la matrice. On part donc du principe que les différences entre les partitionnements porteront sur des éléments "litigieux" de la matrice de départ. Par exemple, s'il existe de nombreux liens entre deux communautés, on s'attend à ce que ces deux communautés fusionnent dans le partitionnement optimisant un critère qui a un terme d'accord négatif faible pour les éléments concernés par la fusion des deux communautés.

Autrement dit, si l'on s'intéresse au cas de deux critères F_1 et F_2 construits sur une même matrice présentant une structure par blocs, il n'est pas aberrant de supposer qu'un partitionnement maximisant F_1 présentera plus probablement des relations que le partitionnement maximisant F_2, aux endroits où les termes d'accord négatif du premier

critère sont inférieurs à ceux du second.

En considérant huit matrices bi-stochastiques $\mathbf{A}_1, ..., \mathbf{A}_8$ de taille 4×4, on s'intéresse aux partitionnements optimisant respectivement les critères de Zahn, de Newman et Girvan, et de Demaine et Immorlica pour les valeurs du paramètre λ égales à $\lambda_1 = 1/12, \lambda_2 = 1/8, \lambda_3 = 1/4$ et $\lambda_4 = 1/2$. Les partitionnements optimaux sont nommés respectivement $\mathbf{X}_{NG}^{\mathbf{A}_i}, \mathbf{X}_{Z}^{\mathbf{A}_i}, \mathbf{X}_{\lambda_1}^{\mathbf{A}_i}, \mathbf{X}_{\lambda_2}^{\mathbf{A}_i}, \mathbf{X}_{\lambda_3}^{\mathbf{A}_i}, \mathbf{X}_{\lambda_4}^{\mathbf{A}_i}$, avec $\mathbf{A}_i, i = 1..8$ la matrice sur laquelle on teste les critères.

Les matrices $\mathbf{A}_1, ..., \mathbf{A}_8$ ont été construites en appliquant un algorithme d'équilibrage bi-stochastique aux matrices binaires – *id est* dont les éléments sont soit 0 soit 1 – dont les structures creuses sont données ci-dessous. Les partitionnements ont été trouvés en testant tous les partitionnements possibles sur la matrice étudiée, et en récupérant le meilleur pour chacun des critères.

$$\mathbf{A}_1 = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} ; \quad \mathbf{X}_{Z}^{\mathbf{A}_1} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_1} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_1} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_1} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_1} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \end{bmatrix} ; \quad \mathbf{X}_{NG}^{\mathbf{A}_2} = \mathbf{X}_{Z}^{\mathbf{A}_2} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_2} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_2} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_2} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_2} = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix} ; \quad \mathbf{X}_{NG}^{\mathbf{A}_3} = \mathbf{X}_{Z}^{\mathbf{A}_3} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_3} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_3} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_3} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_3} = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}$$

$$\mathbf{A}_4 = \begin{bmatrix} \times & & & \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix} ; \quad \mathbf{X}_{NG}^{\mathbf{A}_4} = \mathbf{X}_{Z}^{\mathbf{A}_4} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_4} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_4} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_4} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_4} = \begin{bmatrix} \times & & & \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}$$

$$\mathbf{A}_5 = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix} ; \quad \mathbf{X}_{NG}^{\mathbf{A}_5} = \mathbf{X}_{Z}^{\mathbf{A}_5} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_5} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_5} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_5} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_5} = \begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}$$

On note ici un premier résultat : pour toutes les matrices bi-stochastiques conçues sur des matrices présentant une structure en communautés parfaite, les partitionnements optimaux de chacun des critères sont égaux, et correspondent parfaitement à la structure de communautés de la matrice. On remarque que $\mathbf{X}_{NG}^{\mathbf{A}_1}$ n'apparaît pas. La raison est que, d'après la construction des matrices, on a $\forall(i,j), \mathbf{A}_1(i,j) = 1/n$. Ainsi,

$$\forall \mathbf{X} \in Eq(n), F_{NG}(\mathbf{A}_1, \mathbf{X}) = 0.$$

Autrement dit, le critère de Newman et Girvan est constant sur l'espace des partitionnements, pour cette première matrice.

Considérons maintenant la matrice $\mathbf{A}_6$ tridiagonale :

$$\mathbf{A}_6 = \begin{bmatrix} \times & \times & & \\ \times & \times & \times & \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}$$

$$\mathbf{X}_{NG}^{\mathbf{A}_6} = \mathbf{X}_{Z}^{\mathbf{A}_6} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_6} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_6} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_6} = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix} ; \mathbf{X}_{\lambda_1}^{\mathbf{A}_6} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

Pour cette matrice, le partitionnement $\mathbf{X}_{\lambda_1}^{\mathbf{A}_6}$ – qui présente des termes d'accord négatif toujours inférieurs à ceux des autres critères – rassemble tous les éléments dans une même classe. Puisque toutes les matrices ci-dessous présentent une structure creuse plus petite que la matrice $\mathbf{A}_6$, ce devrait être le cas aussi pour $\mathbf{X}_{\lambda_1}^{\mathbf{A}_7}$ et $\mathbf{X}_{\lambda_1}^{\mathbf{A}_8}$. Les autres partitionnements représentent le découpage en deux classes, dont chacune est une clique dans la matrice $\mathbf{A}_6$. Il existe un unique lien entre ces deux classes.

Dans la matrice $\mathbf{A}_7$, seule une relation n'existe pas – celle entre les éléments 1 et 4 :

$$\mathbf{A}_7 = \begin{bmatrix} \times & \times & \times & \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}$$

$$\mathbf{X}_{NG}^{\mathbf{A}_7} = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}; \mathbf{X}_{Z}^{\mathbf{A}_7} = \mathbf{X}_{\lambda_1}^{\mathbf{A}_7} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_7} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_7} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_7} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

On observe que est tous les partitionnements, à l'exception de $\mathbf{X}_{NG}^{\mathbf{A}_7}$, rassemblent tous les éléments dans la même classe. On explique cela par le fait que $\mathbf{X}_{NG}^{\mathbf{A}_7}$ correspond à la maximisation du critère qui présente le terme d'accord négatif le plus fort pour tout couple (i, j) en dehors de la structure creuse de $\mathbf{A}_7$.

Observons enfin les résultats retournés pour la matrice $\mathbf{A}_8$:

$$\mathbf{A}_8 = \begin{bmatrix} \times & \times & & \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}$$

$$\mathbf{X}_{NG}^{\mathbf{A}_8} = \begin{bmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \times \\ & & \times & \times \end{bmatrix}; \mathbf{X}_{Z}^{\mathbf{A}_8} = \mathbf{X}_{\lambda_3}^{\mathbf{A}_8} = \mathbf{X}_{\lambda_4}^{\mathbf{A}_8} = \begin{bmatrix} \times & & & \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix};$$

$$\mathbf{X}_{\lambda_1}^{\mathbf{A}_8} = \mathbf{X}_{\lambda_2}^{\mathbf{A}_8} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix};$$

Ce dernier cas est intéressant, car, en dehors de $\mathbf{X}_{\lambda_1}^{\mathbf{A}_8}$ et $\mathbf{X}_{\lambda_2}^{\mathbf{A}_8}$ qui correspondent à la maximisation des deux critères ayant les termes d'accord négatif les plus faibles pour tout couple (i, j), deux partitionnements à deux classes différents sont retournés. Le critère de Newman et Girvan est optimisé pour un partitionnement à deux classes ayant le même nombre d'éléments. Tandis que le partitionnement maximisant les trois critères restant

présente aussi deux classes, l'une à un élément, la seconde à trois. Dans les deux cas, chacune des classes est une clique. Cependant, le découpage obtenu pour la maximisation du critère de Newman et Girvan présente deux liens inter-classes, alors que celui par maximisation du critère de Zahn n'en présente qu'un.

Dans tous ces cas très simples, les partitionnements optimaux coïncident bien avec les conjectures faites au début de cette section.

Application de ces critères sur un graphe et observations

Pour illustrer les effets décrits ci-dessus sur un exemple moins trivial, nous avons mené une étude sur le graphe simple $G = (V, E)$ et sa matrice d'adjacence $\mathbf{A}$, tous deux représentés en Figure 1.11(a) et (b). Nous avons d'abord appliqué à $\mathbf{A}$ l'algorithme présenté dans l'article [62] afin de la rendre bi-stochastique. Cette matrice est présentée en Figure 1.11(c). Notons $\tilde{G} = (V, E, w)$ le graphe de transition qui lui est associé.

Nous avons appliqué à $\tilde{G}$ l'Algorithme 1. Il s'agit d'une généralisation aux graphes de transition non orientés d'un algorithme connu sous le nom d'algorithme glouton, ou *Greedy Algorithm*, présenté notamment dans les articles [57] et [63]. Initialement, c'est une heuristique du problème de maximisation de la modularité de Newman et Girvan, dans le cas des graphes simples – article [57] – ou pondérés – article [63]. Cet algorithme fonctionne de la façon suivante :

1– On commence par proposer le partitionnement trivial, dans lequel chaque sommet du graphe est une classe.

2– On fusionne la paire de classes – *id est* on met dans une même classe les deux classes – dont la fusion augmente le plus la modularité.

3– On boucle sur 2 tant qu'il existe un couple de classes dont la fusion améliore la modularité.

Ici, nous avons appliqué cet algorithme à 6 critères : le critère de Newman et Girvan, celui de Zahn, et celui de Demaine et Immorlica avec différentes valeurs de λ : $\lambda_1 = \frac{1}{3n}$, $\lambda_2 = \frac{1}{2n}$, $\lambda_3 = \frac{1}{n}$ et $\lambda_4 = \frac{5}{n}$, $n = 50$ étant le nombre de sommets de G.

Afin de pouvoir comparer les partitionnements retournés par cet algorithme en termes simples, nous avons besoin d'introduire la définition suivante :

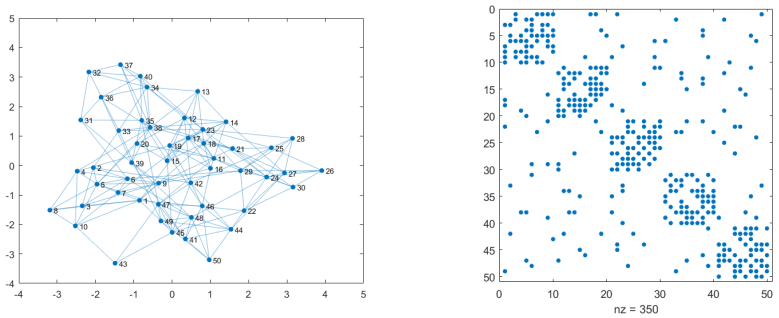
Définition 6. *Soit un ensemble* $\{1, ..., n\}$ *et deux partitionnements sur cet ensemble* $\mathcal{C} = \{C_1, ..., C_p\}$ *et* $\mathcal{K} = \{K_1, ..., K_q\}$*. On dit que* $\mathcal{K}$ ***est supérieur à*** $\mathcal{C}$ *– ou* ***plus grand que*** $\mathcal{C}$ *–, si*

$$\forall k \in \{1, ..., q\}, \exists J_k \subset \{1, ..., p\} : K_k = \bigcup_{j \in J_k} C_j$$

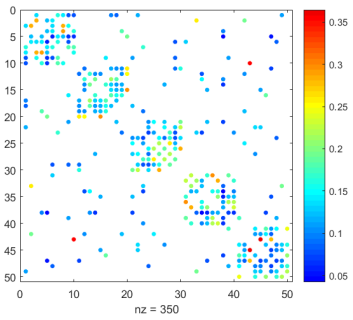
Algorithm 1: : *Algorithme glouton généralisé*

Data: $\tilde{G} = (V, E, w)$, un graphe de transition avec $|V| = n$; $F(\tilde{G}, .) : Eq(n) \to \mathbb{R}$ un critère d'évaluation de partitionnement défini sur $\tilde{G}$

```
/* Le point de départ de l'algorithme est le partitionnement possédant une classe
   par élément :                                                              */
C ← {{C_1}, {C_2}, ..., {C_n}} = {{1}, {2}, ..., {n}};
/* Il y a une classe par sommet dans le graphe, donc n classes                */
nb_Clust ← n;
continuer ← true;
F_ref ← F(G̃, C) ;                                  // Initialisation de F_ref
while continuer do
    /* A chaque tour de boucle, on cherche le couple dont la fusion augmente le
       plus le critère. On stocke cette augmentation dans Δ_max               */
    Δ_max ← 0;
    for i = 1..nb_Clust − 1 do
        for j = i+1..nb_Clust do
            /* Dans C_crt, il y a le partitionnement C dans lequel les classes
               indexés par i et j ont été fusionnés.                          */
            C_crt ← {C_1, ..., C_i ∪ C_j, C_{i+1}, ..., C_{j−1}, C_{j+1}, ...};
            Δ ← F(G̃, C_crt) − F_ref;
            /* Si le gain du critère pour cette fusion est meilleur que le
               meilleur gain pour toutes les autres fusions testées
               précédemment...                                                */
            if Δ > Δ_max then
                C_suivt ← C_crt;
                Δ_max ← Δ;
                /* ... on conserve cette meilleure configuration en mémoire.  */
        end
    end
    if Δ_max ≤ 0 then
        /* Si aucune fusion n'améliore le critère, on renvoie le partitionnement C
           */
        continuer ← false;
        return C;
    else
        /* Sinon, on met à jour les différentes variables pour répéter le
           processus sur cette meilleure configuration possible.              */
        C ← C_suivt ;
        F_ref ← F_ref + Δ_max ;
        nb_Clust ← nb_Clust − 1;
    end
end
```



(a) (b)



(c)

FIGURE 1.11 : *(a) : le graphe G - (b) : sa matrice d'adjacence – dont on voit qu'elle présente 5 blocs diagonaux, relativement bien définis. (c) : la même matrice après équilibrage stochastique. Cette matrice a évidemment la même structure creuse que la matrice d'adjacence de G.*

On note alors :

$$\mathcal{C} \prec \mathcal{K} \tag{1.19}$$

Remarque 9. *Voici quelques remarques à propos la définition précédente :*

- *"$\prec$" représente une relation d'ordre partielle sur les partitionnements. Il s'agit d'un ordre partiel classiquement utiliser pour comparer des relations d'équivalence sur un ensemble.*
- *La définition 6 ci-dessus implique qu'on a $p \geq q$, c'est-à-dire que $\mathcal{K}$ contient moins de classes que $\mathcal{C}$. Les classes de $\mathcal{K}$ sont évidemment de plus grande taille que celles de $\mathcal{C}$, puisqu'il s'agit de réunions des classes de $\mathcal{C}$.*

- *Soit* $\mathcal{F} = \big\{\{1\}, ..., \{n\}\big\}$ *le partitionnement contenant une classe par élément, et* $\mathcal{G} = \big\{\{1, ..., n\}\big\}$ *possédant une unique classe contenant tous les éléments, alors :*

$$\forall \mathcal{C} \in Eq(n), \mathcal{F} \prec \mathcal{C} \prec \mathcal{G}$$

- *Soit* $\mathbf{X}_1 = (x^1_{i,j})^n_{i,j=1}, \mathbf{X}_2 = (x^2_{i,j})^n_{i,j=1} \in Eq(n)$*, on a*

$$\big(\mathbf{X}_1 \prec \mathbf{X}_2\big) \iff \big((x^1_{i,j} = 1) \implies (x^2_{i,j} = 1)\big)$$

On note $\mathcal{C}_{NG}, \mathcal{C}_Z, \mathcal{C}_{\lambda_1}, \mathcal{C}_{\lambda_2}, \mathcal{C}_{\lambda_3}$ et $\mathcal{C}_{\lambda_4}$, les partitionnements retournés par l'Algorithme 1 pour les critères respectivement de Newman et Girvan, de Zahn, de Demaine et Immorlica avec un paramètre λ égal à λ_1, λ_2, λ_3 et λ_4. Précisons déjà que $\mathcal{C}_{\lambda_2} = \mathcal{C}_Z$ et $\mathcal{C}_{\lambda_3} = \mathcal{C}_{NG}$. Ces quatre partitionnements sont représentés Figure 1.12.

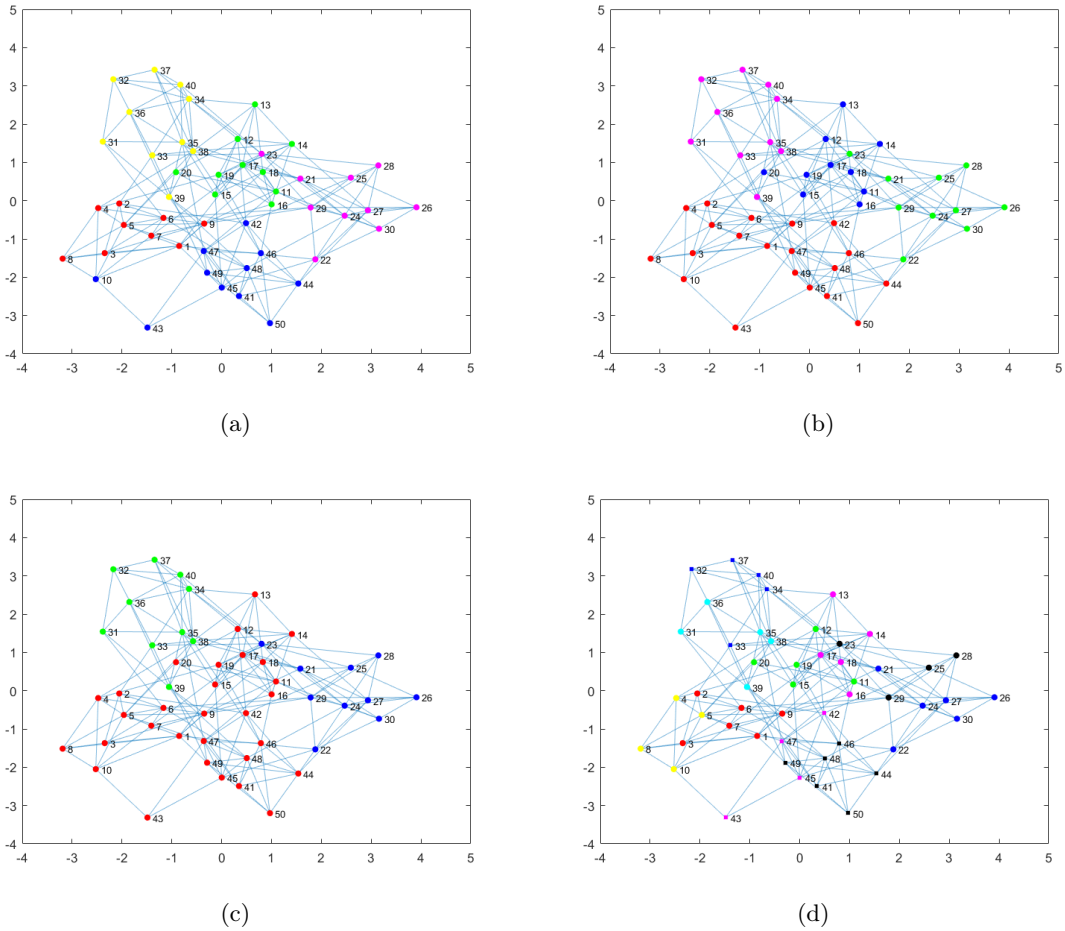
Puisque l'Algorithme 1 n'est qu'une heuristique, on n'a aucune assurance que le partitionnement retourné soit le maximum global du critère sur tous les partitionnements possibles. Il est donc important de vérifier que chaque critère est plus grand sur le partitionnement retourné par sa propre adaptation de l'algorithme glouton que sur les partitionnements correspondant aux autres critères. C'est le cas pour ce graphe $\tilde{G}$ – mais nous notons que nous avons testé plusieurs graphes avant de trouver ce graphe G dont les partitionnements résultant de l'Algorithme 1 vérifient cette condition. La Table 1.2 donne la mesure de chaque critère sur chaque partitionnement.

TABLE 1.2 : *Mesures des différents critères définis sur* $\tilde{G}$ *pour les partitionnements retournés par l'Algorithme 1. En gras : la meilleure valeur du critère sur le jeu de partitionnements proposé.*

	$\mathcal{C}_{NG} = \mathcal{C}_{\lambda_3}$	$\mathcal{C}_Z = \mathcal{C}_{\lambda_2}$	$\mathcal{C}_{\lambda_1}$	$\mathcal{C}_{\lambda_4}$
F_{NG}	**27.19**	25.39	19.98	21.65
F_Z	32.21	**32.39**	30.98	24.21
F_{CC,λ_1}	35.50	36.46	**36.51**	26.17
F_{CC,λ_2}	34.63	**34.99**	33.78	25.87
F_{CC,λ_3}	**32.03**	30.59	25.58	24.97
F_{CC,λ_4}	11.23	-4.61	-40.02	**17.77**

En reprenant les notations de la définition 6, on remarque via la Figure 1.12 les propriétés suivantes :

- On a $\mathcal{C}_Z \succ \mathcal{C}_{NG}$.
- $\forall i : \lambda_i \leq \dfrac{1}{n}, \mathcal{C}_{\lambda_i} \succ \mathcal{C}_{NG}$.



(a) (b)

(c) (d)

FIGURE 1.12 : *Différents partitionnements pour le graphe G présenté en Figure 1.11(a). Pour chaque figure, deux sommets sont dans la même classe si ils ont la même couleur, et la même forme (pour tous les partitionnements à part $\mathcal{C}_{\lambda_4}$, l'identification des classes est faite par la couleur uniquement). (a) : Le partitionnement $\mathcal{C}_{NG} = \mathcal{C}_{\lambda_3}$, possédant 5 classes ; (b) : $\mathcal{C}_Z = \mathcal{C}_{\lambda_2}$ ayant 4 classes ; (c) : $\mathcal{C}_{\lambda_1}$ possédant 3 classes ; (d) : $\mathcal{C}_{\lambda_4}$ possédant 10 classes.*

- De même ; on a $\forall i : \lambda_i \leq \dfrac{1}{2n}, \mathcal{C}_{\lambda_i} \succ \mathcal{C}_Z$.
- $\forall i : \lambda_i \geq \dfrac{1}{2n}, \mathcal{C}_Z \succ \mathcal{C}_{\lambda_i}$.
- On ne peut pas comparer $\mathcal{C}_{NG}$ et $\mathcal{C}_{\lambda_4}$ via la relation d'ordre donnée à l'équation (1.19). Même si $\mathcal{C}_{\lambda_4}$ possède plus de classes, toutes de plus petites tailles, que $\mathcal{C}_{NG}$, le sommet 10 n'est pas classé avec les mêmes sommets dans chacun des deux partitionnements.

Des deux derniers points, on conclut qu'il est plus compliqué d'avoir une idée de l'aspect des partitionnements optimisant le critère de Demaine et Immorlica pour un paramètre λ supérieur à $\frac{1}{n}$, respectivement $\frac{1}{2n}$ par rapport à ceux optimisant le critère de Newman et Girvan, respectivement de Zahn. En effet, pour ces deux derniers points, le terme d'accord négatif d'un des critères n'est pas systématiquement supérieur à celui du second critère.

Il est assez logique que, pour F_{CC,λ_4}, l'élément 10 soit mis dans la classe jaune – puisqu'il est lié à quatre éléments de cette classe –, plutôt que dans la classe magenta car il ne présente un lien qu'avec l'élément 43 de cette classe. En effet, le terme d'accord négatif de F_{CC,λ_4} pénalise très fortement la prise en compte d'une relation dans la structure creuse de la matrice – cinq fois plus que le critère de Newman et Girvan.

Un récapitulatif de ces comparaisons est donnée Table 1.3.

TABLE 1.3 : *Récapitulatif des comparaisons entre partitionnements. Le sens de lecture de la relation est le suivant : le partitionnement indiqué par la ligne est supérieur/inférieur/égal – en fonction du signe dans la case correspondante – au partitionnement indiqué par la colonne. Le symbole $\oslash$ signifie que les deux partitionnements correspondants ne peuvent pas être comparés via la relation d'ordre partielle énoncée par la Définition 6.*

	$\mathcal{C}_{\lambda_1}$	$\mathcal{C}_{\lambda_2}$	$\mathcal{C}_{zahn}$	$\mathcal{C}_{\lambda_3}$	$\mathcal{C}_{newm}$	$\mathcal{C}_{\lambda_4}$
$\mathcal{C}_{\lambda_1}$	$=$	$\succ$	$\succ$	$\succ$	$\succ$	$\succ$
$\mathcal{C}_{\lambda_2}$	$\prec$	$=$	$=$	$\succ$	$\succ$	$\succ$
$\mathcal{C}_{zahn}$	$\prec$	$=$	$=$	$\succ$	$\succ$	$\succ$
$\mathcal{C}_{\lambda_3}$	$\prec$	$\prec$	$\prec$	$=$	$=$	$\oslash$
$\mathcal{C}_{newm}$	$\prec$	$\prec$	$\prec$	$=$	$=$	$\oslash$
$\mathcal{C}_{\lambda_4}$	$\prec$	$\prec$	$\prec$	$\oslash$	$\oslash$	$=$

1.3 Impact de l'équilibrage bi-stochastique sur la détection de communautés

Nous rappelons que le but initial de cette étude est d'observer l'impact de l'équilibrage doublement stochastique de la matrice d'adjacence d'un réseau sur la mise en évidence de sa structure en communautés. Dans le cas de la détection de communautés au sens classique du terme, un tel équilibrage change fondamentalement la nature du problème. En effet, les valeurs numériques de la matrice doublement stochastique ont une grande influence dans la détection de ses blocs, tandis que la matrice d'adjacence est par essence binaire, puisqu'elle représente l'existence ou la non existence d'un lien entre deux sommets. Il est donc crucial de vérifier que l'équilibrage doublement stochastique des matrices

d'adjacence des réseaux ne compromet pas la structure de communautés du-dit réseau. C'est l'objet de cette section.

Dans les sections précédentes, nous avons présenté un certain nombre de mesures de modularité, qui ont été généralisées au cas des graphes pondérés quand cela était possible, et en particulier adaptées au cas des matrices doublement stochastiques. Ici, nous allons comparer les résultats du découpage en communautés de réseaux simples (non pondérés, non orientés, et sans self-loop) dans le cas des différentes mesures de modularité présentées précédemment, dans leur version binaire ainsi que dans leur version numérique, en les appliquant aux matrices d'adjacence des réseaux, préalablement mises sous forme doublement stochastique.

1.3.1 Démarche expérimentale

Dans cette partie, nous explicitons l'algorithme que nous utilisons pour récupérer la structure en communautés des réseaux, et comment nous adaptons cet algorithme aux différents critères développés précédemment. Nous détaillons ensuite les générateurs automatiques de réseaux que nous avons utilisés pour générer des réseaux simples possédant une structure en communautés contrôlée. Enfin, nous développons les mesures que nous avons utilisées pour mesurer la précision des structures en communautés retournées par les différentes versions de l'algorithme. En premier lieu, nous faisons une remarque sur un pré-traitement que nous avons appliqué aux matrices d'adjacence binaires pour les rendre doublement stochastique.

Pré-traitement des réseaux Comme nous l'avons vu au théorème 1, l'équilibrage doublement stochastique d'une matrice nécessite que cette dernière soit bi-irréductible. Afin de s'assurer que cette propriété est vérifiée, et en accord avec la remarque 2, l'algorithme proposé ajoute une diagonale quasi nulle (10^{-8}) à la matrice d'adjacence du réseau avant de l'équilibrer. Nous notons que cela ne change pas la structure diagonale par blocs de la matrice.

Algorithme de Louvain Pour récupérer la structure en communautés des réseaux, nous avons utilisé l'algorithme de Louvain [29]. Il s'agit à l'origine d'une heuristique pour maximiser la modularité de Newman et Girvan. Cet algorithme est à ce jour l'un des algorithmes de détection de communautés les plus précis d'après la récente étude [64]. Par ailleurs, dans l'article [53], il est montré que cet algorithme peut être généralisé à d'autres mesures de modularité que celle de Newamn et Girvan sous certaines conditions

– que vérifient les mesures que nous avons généralisées précédemment. Nous avons donc choisi d'utiliser cet algorithme pour sa précision, sa renommée, et sa générabilité.

L'algorithme de Louvain, dans les grandes lignes, fonctionne de la manière suivante :

1. On commence par considérer que chaque noeud du réseau correspond à une communauté (`INIT`).
2. Pour chaque noeud, on va mesurer le gain en modularité (`GAIN`) obtenu en supprimant le noeud de sa classe courante (`REMOVE`) et en ajoutant ce noeud à une classe adjacente (c'est-à-dire une classe dont l'intersection avec le voisinage du noeud n'est pas vide).
3. On trouve ainsi la communauté telle que l'ajout de ce noeud dans cette communauté maximise le gain, et on insère alors le noeud dans cette communauté (`INSERT`).
4. Si aucune classe ne permet d'obtenir un gain positif, ou supérieur à l'ajout du noeud dans la classe dont il est issu, on remet le noeud dans sa classe initiale.
5. On s'arrête lorsque la situation est stable, c'est-à-dire que pour chaque noeud, la communauté à laquelle il appartient est la meilleure possible en terme de modularité, et donc aucun noeud ne change de communauté.
6. On crée alors un méta-graphe : dans ce nouveau graphe, chaque communauté du graphe initial devient un méta-noeud, les arêtes à l'intérieur d'une même communauté deviennent des "self-loop" – des arêtes ayant le même sommet à ses deux extrémités – de poids égal à la somme des poids des arêtes internes à la communautés, et les arêtes inter-communauté sont sommées – c'est-à-dire que s'il existe deux arêtes de poids 1 entre la communauté C_1 et la communauté C_2, on aura dans le méta-graphe une arête de poids 2 entre les noeuds C_1 et C_2. Le degré du méta-noeud correspond donc à la somme des degrés des noeuds le composant. Le méta-noeud se voit en outre attribuer une taille, qui est la somme des tailles des noeuds le composant, en supposant que dans le graphe initial, chaque noeud est de taille 1.
7. On boucle sur 1. tant que le méta-graphe résultant du processus n'est pas égal au graphe passé en paramètre – *id est* tant que des communautés ont été fusionnées.

Le processus s'arrête naturellement lorsque le méta-graphe passé en entrée du processus est tel que la meilleure configuration est le partitionnement ayant une classe par noeud.

Dans [53], il est expliqué que pour pouvoir généraliser l'algorithme de Louvain à d'autres mesures, les fonctions `INIT`, `INSERT`, `REMOVE` et `GAIN` doivent être adaptées. Nous avons donc évalué ces fonctions pour les mesures de modularités binaires applicables à la détection de communautés sur des réseaux simples, à savoir le critère de Zahn (ZN), le critère de Newman et Girvan (NG), le critère de modularité équilibrée (BM), et les

critères d'écart à l'indétermination (DI) et d'écart à l'uniformité (DU). Nous avons de même évalué ces fonctions pour les mesures de modularité dans le cas bi-stochastique : critères de Zahn pondéré (ZNP), Newman et Girvan pondéré (NGP) – dont on rappelle qu'il est équivalent dans le cas pondéré aux critères d'écart à l'uniformité, d'écart à l'indétermination et de modularité équilibrée –, et Demaine et Immorlica (CC_λ), que nous avons testé pour différentes valeurs de λ, à savoir :$\lambda_1 = \frac{1}{3n}$; $\lambda_2 = \frac{1}{2n}$; $\lambda_3 = \frac{1}{n}$; $\lambda_4 = \frac{2}{n}$; où n est le nombre de noeuds dans le graphe initial. Les fonctions `INIT`, `INSERT` et `REMOVE` mettent à jour les éléments qui composent les communautés, ainsi que deux paramètres de ces communautés : `tot`, qui correspond au poids total des arêtes adjacentes à la communauté, et `s`, qui correspond à la taille de la communauté. La mise à jour de la valeur `tot` d'une communauté correspond naturellement à l'ajout/la suppression du degré du noeud courant, et la mise à jour de sa valeur `s` à l'ajout/la suppression de la taille du noeud courant. En fonction des différents critères, la seule différence dans ces trois fonctions `INIT`, `INSERT` et `REMOVE` est qu'elles ne vont pas toutes mettre à jour les deux valeurs `s` et `tot`. Par exemple, NG n'a besoin que de `tot`, et ZN uniquement de `s`. La fonction `GAIN`, en revanche, est différente pour chaque critère. L'évaluation de `GAIN` pour les différentes mesures est précisée en Table 1.4.

Générateurs automatiques de réseaux Pour comparer la précision des structures en communautés retournées pour les différentes versions de l'algorithme de Louvain, nous avons utilisé des programmes qui génèrent aléatoirement des graphes artificiels ayant une structure en communautés connue. Cela permet d'évaluer la qualité d'un découpage suivant des critères plus objectifs que la modularité, et avec un panel de réseaux quasi-illimité.

— Le banc d'essais de Newman et Girvan (NGBenchmark) développé dans [59] est extrêmement utilisé pour tester les algorithmes de détection de communautés. Il génère un graphe ayant 128 noeuds, et une structure à 4 communautés, chacune possédant 32 noeuds. Ce banc d'essai étant une référence, nous comparons les différents critères sur les réseaux générés via ce banc d'essai. Cependant, les contraintes qu'il fixe sur les réseaux ne lui permettent pas de générer des réseaux ayant des caractéristiques réalistes. Il ne doit donc servir que comme première évaluation, pour savoir si un critère est apte à mesurer la cohérence d'une structure en communautés, ou s'il ne doit pas être utilisé dans ce contexte.

— Le banc d'essais de Landicichinetti-Fortunato-Radicchi (LFRBenchmark), développé dans [65], génère des réseaux simples qui ont une structure en communautés ressemblant effectivement à celles que l'on peut trouver dans des réseaux provenant

du monde réel (sociologie, biologie, web, etc.). Les communautés ont des tailles qui suivent une loi puissance, de paramètre fixé par l'utilisateur. De même pour le degré des noeuds. Il est aussi possible de fixer la taille minimale et maximale des communautés, le degré moyen, le degré maximal, etc. Les paramètres que nous avons choisis pour générer nos réseaux sont donnés Table 1.5.

Ces bancs d'essais sont pratiques car ils génèrent des graphes aléatoires dont on connaît la structure en communautés. La force de cette structure est fournie par un paramètre appelé paramètre de mixage et noté μ. Ce paramètre est défini pour chaque noeud i du graphe de la façon suivante :

$$\mu(i) = \frac{d_i^{out}}{d_i}$$

avec d_i le degré du noeud i, et d_i^{out} le nombre d'arêtes liant i à un noeud extérieur à sa communauté. Un exemple est donné Figure 1.13, où le noeud 1 présente un paramètre de mixage égal à $\mu(1) = \frac{2}{5}$ si l'on considère la structure en communautés en rouge. Si tous les noeuds ont une valeur de μ inférieure à 0.5, alors le graphe a une structure

Critères	`GAIN` **lors de l'ajout du (méta-)noeud** i **à la classe** C
ZN	$2d_w(i,C) - \mathtt{s}_i\mathtt{s}_C$
NG	$d_w(i,C) - \frac{1}{2m} d_i \mathtt{tot_C}$
BM	$2(n^2-2m)d_w(i,C) + \frac{4m-n^2}{2m} d_i \mathtt{tot_C} + 2m\mathtt{s}_i\mathtt{s}_C - n(\mathtt{tot}_C\mathtt{s}_i + d_i\mathtt{tot}_C)$
DI	$d_w(i,C) - \frac{1}{n}(\mathtt{s}_C d_i + \mathtt{tot}_C\mathtt{s}_i + \frac{2m}{n^2}\mathtt{s}_i\mathtt{s}_C)$
DU	$d_w(i,C) - \frac{2m}{n^2}\mathtt{s}_c\mathtt{s}_i$
ZNP	$2d_w(i,C) - \frac{1}{n}\mathtt{s}_i\mathtt{s}_C$
NGP	$d_w(i,C) - \frac{1}{n}\mathtt{s}_i\mathtt{s}_C$
CC_λ	$d_w(i,C) - \lambda(\mathtt{s}_i\mathtt{s}_C - d_b(i,C))$

TABLE 1.4 : *Dans ces formules :* d_i *– resp.* $\mathtt{s}_i$ *– est le degré – resp. la taille – du (méta-)noeud* i, $\mathtt{tot}_C$ *– resp.* $\mathtt{s}_C$ *– est la somme des poids des arêtes adjacentes à – resp. la taille de – la classe* C, $2m$ *est la somme des degrés des noeuds,* n *est la somme des tailles des (méta-)noeuds,* $d_w(i,C)$ *est la valeur de la coupe entre la classe* C *et le (méta-)noeud* i *– id est le poids de l'arête entre* i *et* C*. En outre, pour le critère* CC_λ *qui implique la structure creuse de la matrice d'adjacence pondérée,* $d_b(i,C)$ *est valeur de la coupe entre* i *et* C *dans le cas du réseau simple (non pondéré).*

TABLE 1.5 : *Paramètres pour générer les réseaux.*

Paramètre	Valeur
nombre de noeuds	233
degré moyen	20
degré maximum	48
exp. db des degrés	-2
exp. db des tailles de communautés	-1

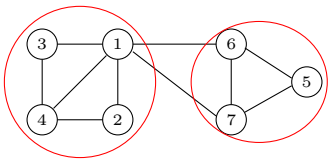


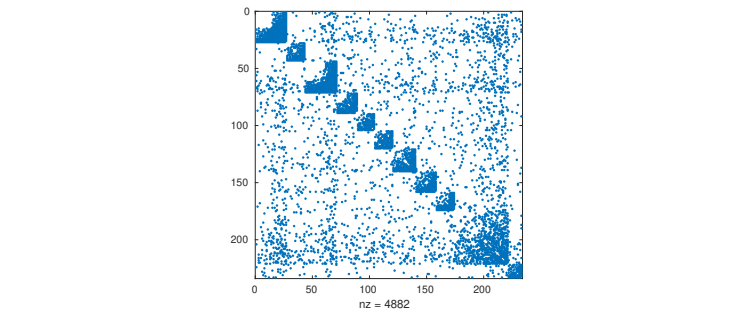
FIGURE 1.13 : *Dans ce graphe et pour la structure en communautés représentée en rouge, le paramètre de mixage du noeud* 1 *vaut* $\mu(1) = 2/5$.

en communautés au sens fort, c'est-à-dire que chaque noeud possède plus de relations avec sa communauté qu'avec l'extérieur. Normalement le graphe possède une valeur de μ par noeud. Par abus de notation, on note μ la moyenne des $\mu(i)$. Evidemment, plus μ est grand, plus la structure en communautés du graphe est mal définie. Afin d'avoir les résultats les plus justes possibles, sur chacun des bancs d'essais, nous avons généré 100 réseaux pour chaque valeur de $\mu \in \{k \times 0.03, k = 1..25\}$. Trois exemples de matrices d'adjacence des réseaux générés par ces bancs d'essais sont fournis Figure 1.14.

Evaluation Afin d'évaluer la qualité des partitionnements retournés par l'algorithme de Louvain, puisque nous connaissons la structure en communautés du réseau passé en entrée de l'algorithme, nous pouvons calculer l'information mutuelle normalisée (NMI) entre le partitionnement retourné et le partitionnement attendu, et le ratio entre le nombre de classes retourné et le nombre de classes attendu (RC), comme cela est préconisé dans l'étude [64].

La NMI est une mesure qui vient de la théorie de l'information (voir par exemple [66]). Elle permet de mesurer l'écart entre deux partitions $\mathcal{C} = \{C_1, ...C_p\}$ et $\mathcal{K} = \{K_1, ..., K_q\}$ sur un ensemble de m données, sans nécessiter que p soit égal à q. Elle est basée sur la matrice de confusion des deux partitions, à savoir la matrice $\mathbf{N} \in \mathbb{N}^{p \times q}$, telle que :

$$\forall i \in \{1...p\}, \forall j \in \{1...q\}, N(i,j) = |\{C_i \cap K_j\}|$$



(a)

(b) (c)

FIGURE 1.14 : *(a) : Matrice d'adjacence d'un réseau généré par LFRBenchmark avec les paramètres indiqués Table 1.5, pour un paramètre de mixage $\mu = 0.45$. (a) et (b) : Matrices d'adjacence de réseaux générés par NGBenchmark. (a) : Pour un paramètre de mixage $\mu = 0.3$. (b) : Pour un paramètre de mixage $\mu = 0.49$.*

avec $|.|$ la fonction cardinal. La formule de la NMI est donnée dans [64] par :

$$\mathrm{NMI}(\mathcal{C},\mathcal{K}) = \frac{-2\sum_{k=1}^{p}\sum_{l=1}^{q}\mathbf{N}(k,l)log_2\Big(m\mathbf{N}(k,l)/\big(|C_k||K_l|\big)\Big)}{\sum_{k=1}^{p}|C_k|log_2(|C_k|/m) + \sum_{l=1}^{q}|K_l|log_2(|K_l|/m)} \tag{1.20}$$

Dans le cas où $N(k,l) = 0$, on suppose que le terme correspondant dans la somme du numérateur est nul, par limite de $xlog(x)$ en 0.

Remarque 10. *La NMI peut être exprimée différemment en changeant le dénominateur de l'équation (1.20). Cependant, des tests pour différentes NMI ont été effectués par les auteurs de l'étude [64] qui précisent que toutes les versions testées fournissent des résultats cohérents par rapport à la version de l'équation (1.20). Nous utiliserons donc cette version dans nos tests.*

La NMI est à valeurs dans $[0, 1]$. Si $\mathrm{NMI}(\mathcal{C}, \mathcal{K}) = 1$, cela signifie que les deux partitions sont les mêmes. Réciproquement, plus la NMI est proche de 0, plus les partitions sont différentes.

Le RC (ratio entre les nombres de communautés) correspond simplement au ratio entre le nombre des communautés retournées par un algorithme, et le nombre de communautés attendu. En reprenant les notations utilisées pour la NMI, et en supposant que $\mathcal{C}$ est le partitionnement cible et $\mathcal{K}$ le partitionnement retourné par l'algorithme, alors $RC = q/p$.

Afin d'avoir une bonne interprétation de la qualité du partitionnement, il est important de corréler NMI et RC. En effet, la NMI seule ne prend pas explicitement en compte l'écart entre le nombre de classes dans l'un et l'autre des partitionnements.

1.3.2 Résultats expérimentaux

On observe maintenant les résultats de ces différentes mesures sur les deux bancs d'essais. D'abord, les résultats pour les mesures binaires NG, ZN, BM, DU et DI sont donnés Figure 1.15. Sur cette figure, on observe que NG, BM, DU et DI donnent des résultats extrêmement similaires en terme de NMI et de RC. Ces résultats sont même strictement identiques sur NGBenchmark, d'où le fait que seule NG soit représentée (si affichées, les courbes de BM, DI, et DU seraient confondues avec celle de NG). Les performances de ZN sont par contre sensiblement inférieures à celles des autres mesures, sur LFRBenchmark ainsi que sur NGBenchmark. On remarque en outre que ZN a tendance à sur-partitionner le graphe (*id est* retourner trop de classes) lorsque la structure en communautés devient moins nette, tandis que les autres mesures ont tendance à sous-partitionner – autrement dit, à fusionner des classes qui ne devraient pas l'être.

Si l'on observe maintenant les résultats des mesures pondérées donnés Figure 1.16, on observe de bonnes performances des critères NGP, CC_{λ_3} et CC_{λ_4}. Les critères CC_{λ_1} – en rouge –, CC_{λ_2} – en bleu –, et ZNP – en marron –, ont une tendance au sous-partitionnement qui ne leur permet pas d'être compétitifs. Cela est cohérent avec le fait que leur terme d'accord négatif est faible devant ceux de NGP, CC_{λ_3} et CC_{λ_4}. Il est donc moins pénalisant pour ces critères de mettre en relation des éléments dans la structure creuse des réseaux. On remarque en outre que les critères de Zahn binaire et de Zahn pondéré ont des comportements opposés. En effet, le critère de Zahn sur les réseaux simples ne fusionne pas suffisamment les classes, et renvoie ainsi une structure contenant trop de communautés par rapport à ce qui est attendu. *A contrario*, le critère de Zahn appliqué aux réseaux doublement stochastiques renvoie une structure avec trop peu de communautés : ce critère a donc tendance à trop fusionner les classes.

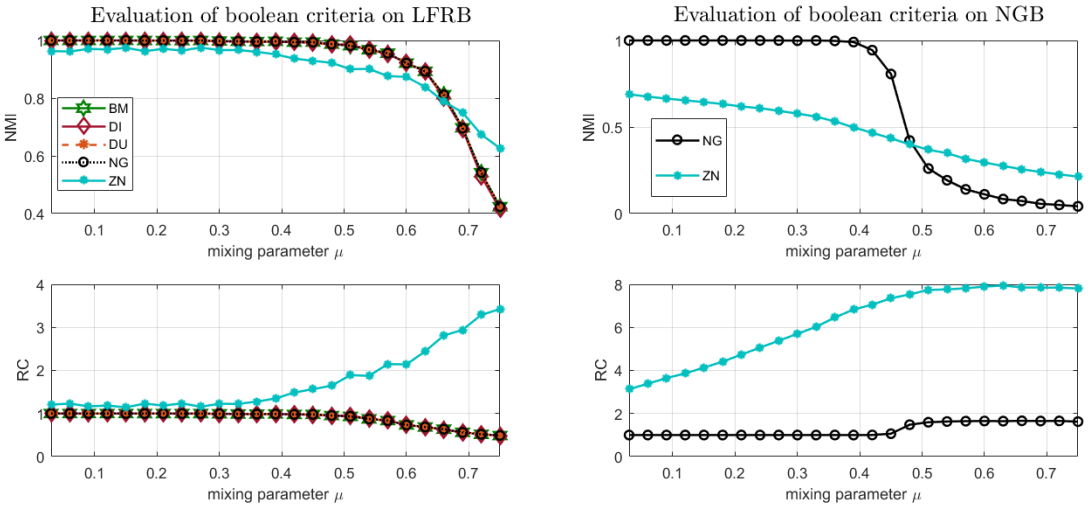


FIGURE 1.15 : *NMI et RC pour les 5 critères binaires, pour des graphes générés via LFRBenchmark à gauche, et via NGBenchmark à droite.*

Remarque 11. *Nous avons aussi testé un critère $F_{CC,1}$ qui n'est pas présenté sur les figures. Cette mesure, étant donné son fort terme d'accord négatif, ne permet pas d'approcher la structure en communautés des réseaux. Sa valeur de RC est autour de 4 sur LFRBenchmark, et entre 7 et 10 pour NGBenchmark.*

Comparons maintenant les meilleures mesures parmi les binaires (NG, BM, DI et DU) et les meilleures mesures parmi les pondérées (NGP, CC_{λ_3}, et CC_{λ_4}). Les résultats sont affichés sur la Figure 1.17.

— Sur LFRBenchmark, il est clair que CC_{λ_4} est meilleur que les autres critères pour la NMI et le RC. On peut aussi se rendre compte que la NMI de CC_{λ_3} est légèrement inférieure à celle des autres mesures, de même que son RC est moins proche de 1. Par ailleurs, au regard de ces courbes, il n'est pas possible de statuer sur une éventuelle meilleure performance de tel ou tel critère entre BM, DI, DU, NG et NGP sur LFRBenchmark : si l'on zoome sur les courbes, on voit que chaque critère peut être tantôt au dessus, tantôt en dessous des autres.

— Sur NGBenchmark, CC_{λ_4} est sensiblement inférieur aux autres critères. En outre, NG et NGP ont strictement les mêmes performances sur ce banc d'essai. Il est par contre difficile de comparer CC_{λ_3} à NG et NGP : certes, son RC est meilleur, mais sa NMI est tantôt plus élevée, tantôt plus faible.

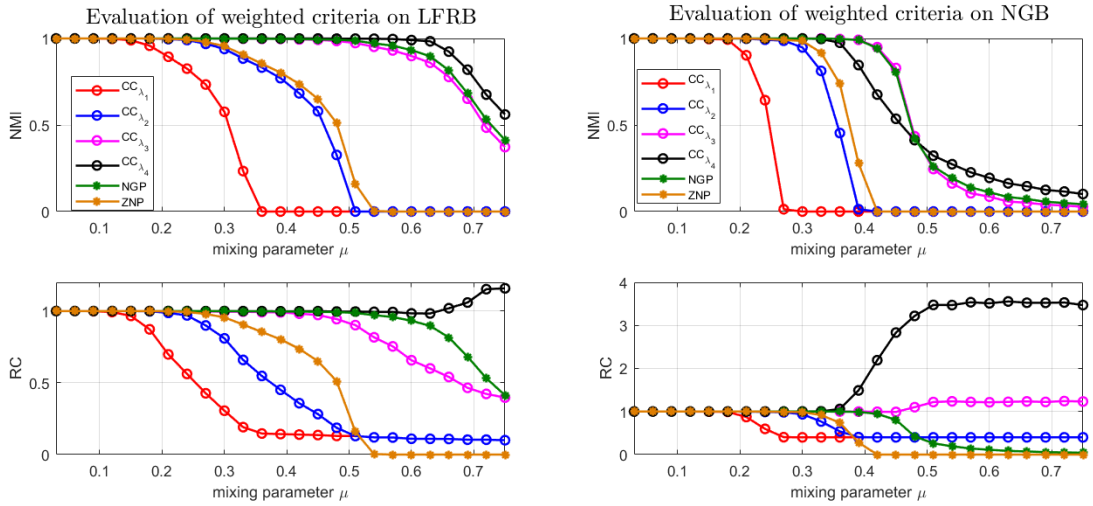


FIGURE 1.16 : *NMI et RC pour les critères pondérés, sur LFRBenchmark (à gauche) et NGBenchmark (à droite).*

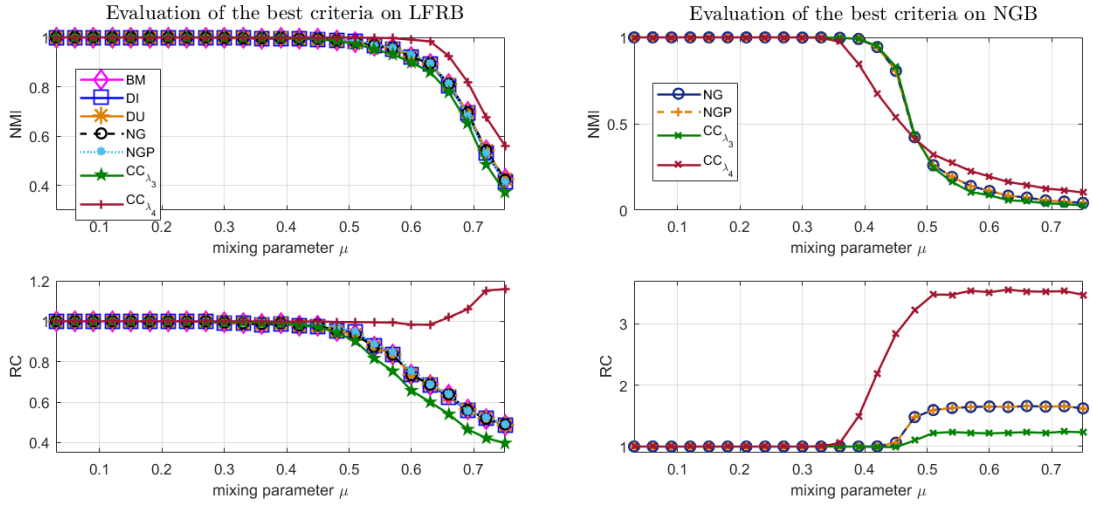


FIGURE 1.17 : *NMI et RC pour les meilleurs des critères parmi les booléens et les pondérés, sur les bancs d'essais LFRB (à gauche) et NGB (à droite).*

Afin d'affiner la comparaison des performances de ces critères, nous allons observer deux nouvelles quantités construites sur la NMI et le RC, que nous noterons $\widetilde{NMI}$ et Δ_{RC}. Nous les définissons ci-dessous.

Pour la NMI, en notant $NMI^F(\mu = p \times 0.03)$ la moyenne des NMI obtenues pour le critère F sur les 100 graphes associés au mixing parameter de valeur μ, nous construisons la quantité

$$\widetilde{NMI}^F(p \times 0.03) = \frac{1}{p}\sum_{k=1}^{p} NMI^F(k \times 0.03)$$

Pour Δ_{RC}, nous travaillons aussi sur une moyenne cumulative. Cependant, ce qui nous intéresse dans RC lors de la comparaison de performances, c'est que cette valeur soit la plus proche de 1. Ainsi, c'est cet écart que nous allons considérer. En d'autres termes, en conservant les notations ci-dessus, on construit

$$\Delta_{RC}^F(p \times 0.03) = \frac{1}{p}\sum_{k=1}^{p} |1 - RC^F(k \times 0.03)|$$

où $|.|$ représente la valeur absolue.

Puisque l'on veut comparer des critères, on choisit l'un d'entre eux – en l'occurrence, nous choisissons NG – comme critère de référence. Sur la Figure 1.18, on affiche $\widetilde{NMI}^F(\mu) - \widetilde{NMI}^{NG}(\mu)$ en haut, et $\Delta_{RC}^{NG}(\mu) - \Delta_{RC}^F(\mu)$ en bas, pour chaque valeur de μ et pour $F = BM, DU, DI, NGP$ sur LFRBenchmark, et $F = CC_{\lambda_3}$ sur NGBenchmark. Pour ces deux nouvelles quantités, une valeur inférieure à 0 signifie que F est moins précise que NG, tandis qu'une valeur supérieure à 0 indique que les découpages en communautés retournés par F sont généralement meilleurs que ceux retournés par NG. Elles permettent d'avoir une idée de la tendance des critères plutôt que de s'intéresser à une comparaison point à point.

Cette figure permet de statuer que NGP est meilleure que NG, DI, DU et BM sur LFRBenchmark. Il est en outre assez net que DI présente de moins bonnes performances que les autres critères à partir de $\mu = 0.27$. La comparaison des autres critères entre eux n'a pas beaucoup de sens, étant donné leur proximité.

Sur NGBenchmark, il est aussi compliqué de conclure. On peut cependant observer que CC_{λ_3} a de meilleures performances que les critères NG et NGP (dont on rappelle qu'ils sont strictement équivalents en terme de performances) tant que $\mu \leq 0.5$.

Ainsi, les critères pondérés CC_{λ_3} et NGP fournissent des résultats équivalents – sinon meilleurs – en terme de précision que les critères booléens sur les deux bancs d'essais que nous avons mis à l'épreuve. CC_{λ_4} est le meilleur des critères sur LFRBenchmark, mais il

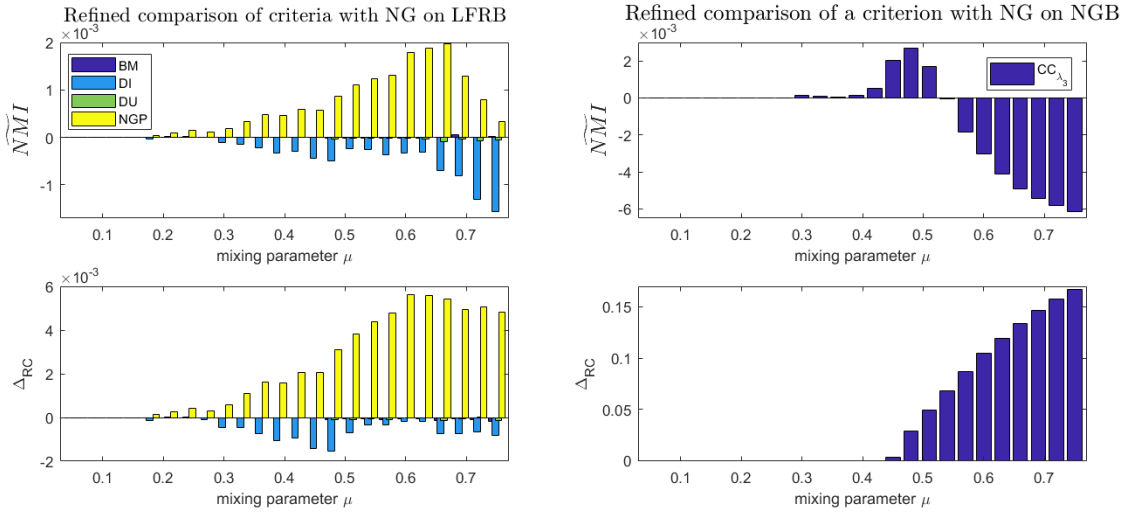


FIGURE 1.18 : *Raffinage de la comparaison des performances de critères a priori indissociables, sur LFRBenchmark (à gauche) et NGBenchmark (à droite).*

est moins précis que NG=NGP sur NGBenchmark. Quant à CC_{λ_3}, il est moins précis sur NGBenchmark, et globalement équivalent à NG=NGP sur NGBenchmark.

Il reste cependant à s'assurer que ces bons résultats ne se font pas au détriment de la convergence de l'algorithme de Louvain. Pour ce faire, la Figure 1.19 indique, la moyenne sur les 100 graphes du nombre de fois où tous les (méta-)noeuds sont parcourus par l'algorithme. Les résultats obtenus sur LFRBenchmark sont à gauche, ceux obtenus sur NGBenchmark sont à droite. Les figures du haut correspondent aux nombres des itérations pour les meilleurs critères binaires (NG,BM,DI,DU), et les figures du bas, aux nombres des itérations pour les meilleurs critères pondérés (NGP, CC_{λ_3}, CC_{λ_4}). Pour les critères pondérés, on affiche aussi le nombre d'itérations effectuées pour NG sur les réseaux binaires, à titre de comparaison.

Sur ces figures, on observe plusieurs choses :

— Sur NGBenchmark, les critères NG, BM, DI, DU et NGP ont le même nombre d'itérations, pour tous les μ. Puisqu'ils ont aussi des performances strictement équivalentes, on pense que l'algorithme se déroule de la même façon pour ces cinq critères sur ce banc d'essais.

— Sur LFRBenchmark, les critères ont besoin d'un nombre équivalent d'itérations pour que l'algorithme converge, à l'exception de CC_{λ_4} qui devient un peu plus rapide que les autres à mesure que la structure en communautés du graphes devient plus floue.

— Cette remarque reste vraie sur NGBenchmark. En revanche on remarque aussi que si la structure du réseau est nette, CC_{λ_4} a besoin de plus d'itérations que les autres critères pour atteindre la convergence. Ainsi, pour ce critère, un nombre moins important d'itérations dans le cas d'une structure en communautés mal définie se fait au détriment d'un plus grand nombre d'itérations quand la structure en communautés est forte.

— Sur LFRBenchmark, DU est systématiquement légèrement plus rapide que les autres critères booléens.

— CC_{λ_3} a un comportement opposé à celui de CC_{λ_4} en terme de nombre d'itérations, à savoir qu'il converge plus rapidement que les autres critères pour des valeurs de μ faibles, et plus lentement quand μ devient grand. Cependant, le comportement de CC_{λ_3} est moins marqué que celui de CC_{λ_4}, et se rapproche plus de celui de NG et NGP.

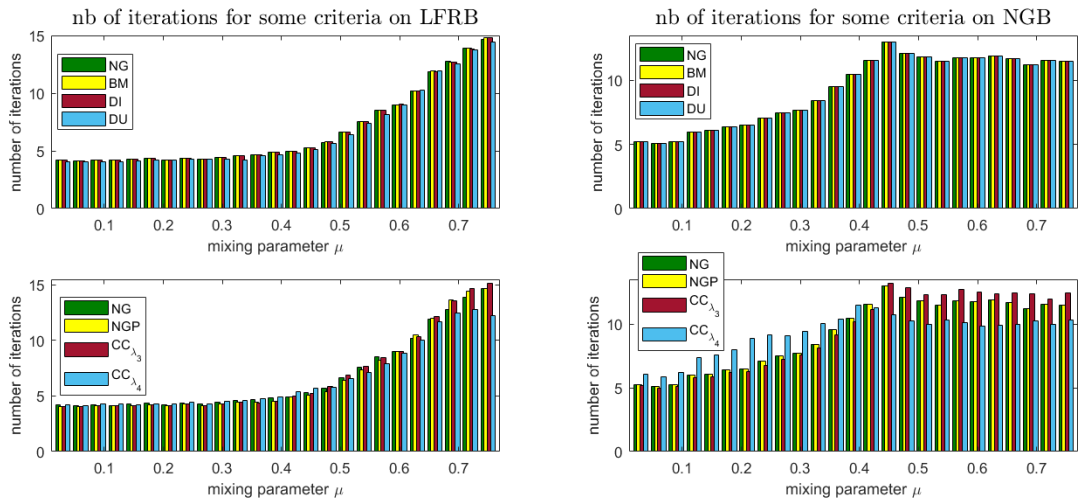


FIGURE 1.19 : *Nombre de parcours des (méta-)noeuds pour parvenir à la convergence, sur LFRBenchmark (à gauche) et NGBenchmark (à droite).*

Sur la Figure 1.19, on remarque que, dans l'ensemble, le nombre d'itérations reste du même ordre de grandeur pour tous les critères, et que le comportement global est le même, à savoir :

— Sur NGBenchmark, le nombre d'itérations croît jusqu'à une valeur de $\mu \in [0.4, 0.5]$, puis ce nombre stagne à mesure que μ croît.

— Sur LFRBenchmark, le nombre d'itération stagne jusqu'à une valeur de $\mu \in [0.4, 0.5]$, avant de croître à mesure que μ croît.

1.4 Conclusion

Dans ce chapitre, nous avons généralisé plusieurs mesures de modularité, utilisées pour détecter la structure en communautés d'un réseau, et nous avons adapté et simplifié ces mesures aux cas des graphes doublement stochastiques. Une première remarque importante est que, si le graphe est doublement stochastique, plusieurs de ces mesures deviennent strictement équivalentes.

Par ailleurs, après une étude de la performance de ces critères sur des réseaux générés aléatoirement, nous remarquons que le fait d'équilibrer la matrice d'adjacence du réseau sous forme doublement stochastique ne semble pas dénaturer la structure en communautés des réseaux. En effet, les résultats obtenus pour les critères pondérés et binaires sont équivalents lorsqu'ils sont appliqués à des réseaux générés artificiellement avec NGBenchmark. Mais surtout, les différentes versions de l'algorithme de Louvain (correspondant aux différentes mesures de modularité) qui permettent la détection la plus précise de la structure en communautés du réseau passé en paramètre, sont obtenues dans le cas des graphes issus d'équilibrage doublement stochastique préalable ; les trois meilleurs critères étant, du meilleur au pire : le critère de *Correlation Clustering* avec $\lambda = \dfrac{2}{n}$, le critère de *Correlation Clustering* avec $\lambda = \dfrac{1}{n}$, et le critère de Newman et Girvan dans sa version pondérée.

Cela confirme qu'utiliser l'équilibrage doublement stochastique comme pré-traitement à un algorithme de détection de communautés, loin de compromettre la structure en communautés initiale du réseau, permet au contraire de mettre en évidence les communautés.

Enfin, nous remarquons que les critères de *Correlation Clustering* peuvent être compliqués à mettre en place, puisqu'ils prennent en compte la matrice d'adjacence équilibrée ainsi que la structure creuse de la matrice d'adjacence du graphe binaire. Cependant, les autres critères peuvent être mis en place très simplement, et de nombreux autres critères basés sur le même principe peuvent être créés pour les graphes doublement stochastiques, en proposant d'autres valeurs pour le terme d'accord négatif. C'est-à-dire que, pour une matrice symétrique doublement stochastique $\mathbf{A} \in S_n(\mathbb{R})$, toute mesure du type

$$\begin{array}{rccc} F(\mathbf{A},.): & Eq(n) & \longrightarrow & \mathbb{R} \\ & \mathbf{X} & \longmapsto & F(\mathbf{A},\mathbf{X}) = \displaystyle\sum_{i,j} (a_{i,j} - K)\, x_{i,j} \end{array} \tag{1.21}$$

avec $K \in]0, 1]$, peut servir à détecter la structure par blocs de $\mathbf{A}$. Cette remarque est d'autant plus importante que nos tests mettent en évidence que la mesure de Newman et Girvan généralisée aux matrices doublement stochastiques possède le même défaut que dans le cas des graphes binaires, à savoir une tendance à ne pas détecter suffisamment de communautés.

Ce phénomène a été étudié en profondeur dans le cas des graphes binaires, pour lesquels il a été démontré que la modularité échoue à détecter les petites communautés. Cette propriété de la modularité est appelé limite de résolution [58]. Pour pallier cette limite, un paramètre appelé paramètre de résolution a été ajouté à la modularité de Newman et Girvan [67]. Cette mesure paramétrée s'exprime alors comme

$$F(\mathbf{A}, \mathbf{X}) = \sum_{i,j} \left(a_{i,j} - \gamma \frac{d_i d_j}{2m} \right) x_{i,j}.$$

On rappelle que la modularité classique est définie en prenant $\gamma = 1$. En posant $\gamma = n \times K$, la fonction définie Equation (1.21) correspond exactement à la modularité paramétrée, dans le cas où la matrice $\mathbf{A}$ est bi-stochastique.

Il serait donc intéressant d'analyser dans quelle mesure le phénomène observé dans nos tests, à savoir que l'optimisation du critère de Newman et Girvan détecte trop peu de communautés dans les graphes doublement stochastiques, correspond à la limite de résolution dont souffre la modularité sur les graphes binaires, ainsi que de faire varier K pour trouver des valeurs permettant de détecter les communautés de façon plus précise.

Chapitre 2

Un Algorithme spectral

Dans cette partie, nous proposons un algorithme spectral basé sur l'équilibrage doublement stochastique d'une matrice afin de détecter sa structure diagonale par blocs. Cet algorithme s'apparente aux algorithmes de classification non supervisée (ou clustering), dont le but est de regrouper les éléments similaires dans un jeu de données sans information *a priori*. Dans le même esprit que celui de notre méthode, de nombreux algorithmes de partitionnement recherchent des blocs dans les matrices représentant le jeu de données – telles que les matrices d'adjacence ou d'affinité, les tables de contingences ou le Laplacien du graphe.

Par ailleurs, le lien entre les propriétés spectrales et structurelles des matrices est souvent un élément-clé de tels algorithmes [68–70]. 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 bi-partition. 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 2.4(c)). Les autres méthodes spectrales existantes [31, 71, 72] consistent à projeter les données dans le sous-espace vectoriel généré par les vecteurs propres dominants – respectivement associés aux plus petites valeurs propres – de la matrice d'adjacence du graphe – respectivement de son Laplacien –, en prenant autant de vecteurs qu'il y a de classes attendues, ce qui suppose que ce nombre est connu.

Notre algorithme étant basé sur les éléments spectraux de l'équilibrage doublement stochastique de la matrice, il faut que cet équilibrage existe, et donc que la matrice soit bi-irréductible. Nous verrons en Section 2.2 comment ramener cela aux partitionnements de graphes fortement connexes, ce qui rend notre méthode particulièrement adaptée aux

tâches de classification non supervisée.

Nous rappelons que trouver un équilibrage doublement stochastique pour une matrice $\mathbf{A} \in {\mathbb{R}_+}^{n \times n}$ 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 2.2 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 partitionnements précis, même lorsque la structure du graphe est très dissymétrique.

Le schéma global de notre algorithme est présenté Figure 2.1. Son fonctionnement sera détaillé au cours de ce chapitre. La première étape consiste en l'équilibrage doublement stochastique de la matrice. En suivant se trouve l'étape d'analyse spectrale, au cours de laquelle chaque vecteur singulier dominant est trié dans l'ordre croissant, avant analyse de sa structure constante par morceaux. En Section 2.2, nous exposons la connexion entre l'aspect constant par morceaux des vecteurs singuliers dominants d'une matrice doublement stochastique et sa structure par blocs. En Section 2.3, nous utilisons des outils du traitement du signal afin de détecter les informations sur le partitionnement contenues dans les vecteurs singuliers. Il est possible d'affiner la structure par blocs de la matrice en bouclant sur cette étape d'analyse spectrale. Dans la Section 2.4, nous décrivons comment récupérer, au fil des itérations, des vecteurs singuliers qui continuent de fournir des informations utiles. La Section 2.5 est dédiée au réarrangement des blocs : nous devons

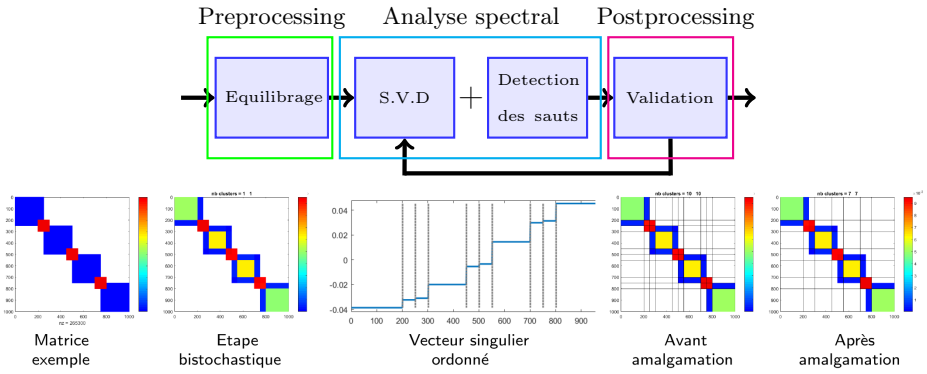


FIGURE 2.1 : *Grandes lignes de notre algorithme et un exemple d'application.*

compiler et analyser les partitionnements 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 partitionnement et qui nous sert aussi pour notre critère d'arrêt. Cette section correspond à l'étape de validation de la Figure 2.1. Enfin, quelques tests numériques sur des matrices d'affinité sont présentés Section 2.7 afin d'indiquer l'efficacité de notre procédure. Nous commençons ce chapitre par la Section 2.1 qui comporte quelques rappels et introductions de notions nécessaires à la compréhension de ce chapitre.

2.1 Quelques définitions et théorèmes

Un certain nombre de points de ce chapitre font appel à des notions méritant d'être rappelées ou introduites. C'est le but de cette section.

2.1.1 Bi-stochasticité et vecteur de Perron-Frobenius

Nous commençons par des définitions et théorèmes en lien avec la propriété de bi-stochasticité des matrices. La plupart de ces résultats sont des rappels de la section introductive *Définitions et Notations*.

Définition 7. ***Matrice doublement stochastique :*** *Une matrice carrée à coefficients non négatifs* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *est dite* ***doublement stochastique*** *ou* ***bi-stochastique*** *si*

$$\begin{cases} \mathbf{A}\mathbf{e} = \mathbf{e} \\ \mathbf{A}^T\mathbf{e} = \mathbf{e} \end{cases},$$

avec $\mathbf{e} = (1, ..., 1)^T \in \mathbb{R}^n$.

Remarque 12. *Toute matrice stochastique et symétrique est donc bi-stochastique.*

Définition 8. ***Equilibrage doublement stochastique :*** *Soit une matrice carrée à coefficients non négatifs* $\mathbf{A} \in \mathbb{R}^{n\times n}$*, trouver un équilibrage doublement stochastique pour* $\mathbf{A}$ *signifie trouver deux vecteurs* $\mathbf{r}, \mathbf{c} \in {\mathbb{R}_+^*}^n$ *tels que*

$$\begin{cases} \mathcal{D}(\mathbf{r})\mathbf{A}\mathcal{D}(\mathbf{c})\mathbf{e} = \mathbf{e} \\ \mathcal{D}(\mathbf{c})\mathbf{A}^T\mathcal{D}(\mathbf{r})\mathbf{e} = \mathbf{e} \end{cases},$$

avec $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ *l'opérateur transformant un vecteur en une matrice diagonale, et* $\mathbf{e} = (1, ..., 1)^T \in \mathbb{R}^n$*, où* $\mathbf{r}$ *et* $\mathbf{c}$ *sont appelés* ***facteurs d'équilibrage*** *de* $\mathbf{A}$.

Dans la suite, nous rappelons le théorème de Sinkhorn-Knopp [30], et les deux définitions sur la structure des matrices nécessaires à sa compréhension. Ces définitions peuvent être trouvées dans [34].

Définition 9. ***Bi-irréductibilité ou entière indécomposabilité :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *est dite* ***bi-irréductible*** *s'il n'existe pas deux matrices de permutation* $\mathbf{R}, \mathbf{Q}$ *telles que :*

$$\mathbf{RAQ} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_{1,2} \\ 0 & \mathbf{A}_2 \end{bmatrix}$$

avec $\mathbf{A}_1, \mathbf{A}_2$ *deux matrices carrées non vides.*

Remarque 13. *Cela signifie qu'il n'existe pas de permutations indépendantes des lignes et des colonnes permettant de mettre* $\mathbf{A}$ *sous forme triangulaire par blocs.*

Définition 10. ***Support total :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *est dite* ***à support total*** *si tous ses éléments non nuls se trouve sur une diagonale strictement non nulle. Une caractérisation de cette définition est que* $\mathbf{A}$ *est à support total s'il existe deux matrices de permutations* $\mathbf{R}, \mathbf{Q}$ *telles que*

$$\mathbf{RAQ} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix}$$

où $\mathbf{A}_1, ..., \mathbf{A}_k$ *sont des matrices bi-irréductibles [34].*

Nous pouvons maintenant énoncer le théorème de Sinkhorn-Knopp [30]

Théorème 2. ***Sinkhorn-Knopp :*** *Soit* $\mathbf{A} \in {\mathbb{R}_+}^{n\times n}$*, une condition nécessaire et suffisante pour qu'il existe une matrice doublement stochastique* $\mathbf{P}$ *de la forme* $\mathbf{DAF}$ *avec* $\mathbf{D}$ *et* $\mathbf{F}$ *deux matrices diagonales, et dont la diagonale principale est strictement positive, c'est que* $\mathbf{A}$ *soit à support total. Si* $\mathbf{P}$ *existe, alors* $\mathbf{P}$ *est unique.* $\mathbf{D}$ *et* $\mathbf{F}$ *sont aussi uniques à une multiplication scalaire près si et seulement si* $\mathbf{A}$ *est bi-irréductible.*

Nous rappelons maintenant la notion d'irréductibilité, qui est nécessaire à l'introduction d'un théorème important. Nous rappelons que cette définition véhicule une information essentielle sur les graphes d'adjacence associés à de telles matrices.

Définition 11. ***Irréductibilité :*** *Une matrice* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *est dite* ***irréductible*** *s'il n'existe pas de matrice de permutation* $\mathbf{Q}$ *telle que :*

$$\mathbf{QAQ}^T = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_{1,2} \\ 0 & \mathbf{A}_2 \end{bmatrix}$$

avec $\mathbf{A}_1, \mathbf{A}_2$ deux matrices carrées non vides. Une caractérisation des matrices irréductibles est qu'elles sont les matrices d'adjacence de graphes fortement connexes [35].

Remarque 14. *Bien entendu, toute matrice bi-irréductible est irréductible. A l'inverse, si une matrice $\mathbf{A} \in \mathbb{R}^{n\times n}$ est irréductible et que sa diagonale ne possède pas d'élément nul, alors $\mathbf{A}$ est bi-irréductible. Cela se montre simplement en appliquant l'algorithme de Pothen et Fan [36] à $\mathbf{A}$. Le but de cet algorithme est de permuter $\mathbf{A}$ pour la mettre sous forme triangulaire par blocs. Une fois que $\mathbf{A}$ est permutée de sorte à ce que sa diagonale ne possède pas d'élément nul, seules des permutations symétriques sur les lignes et les colonnes sont nécessaires pour trouver une forme triangulaire par blocs pour $\mathbf{A}$. Ainsi le fait que la diagonale de $\mathbf{A}$ soit pleine permet de forcer $\mathbf{Q} = \mathbf{R}^T$ dans la définition 9, ce qui nous ramène à la définition 11.*

Le théorème de Perron-Frobenius [73, 74] va nous permettre, plus tard, de montrer une caractéristique structurelle sur les vecteurs propres dominants des matrices doublement stochastiques. Ce théorème est énoncé dans [75] de la manière suivante.

Théorème 3. ***Perron-Frobenius*** *Soit $\mathbf{A} \in \mathcal{M}_n(\mathbb{R})$ une matrice à coefficients positifs et ρ son rayon spectral.*

(i) ***(Perron)*** *Supposons que $\mathbf{A}$ est primitive. Alors $\rho > 0$, ρ est une valeur propre dominante et simple de $\mathbf{A}$ et il existe un unique vecteur $\mathbf{x}^+ \in \mathbb{R}^n$ à coefficients strictement positifs tel que $\mathbf{A}\mathbf{x}^+ = \rho\mathbf{x}^+$ et $||\mathbf{x}^+||_1 = 1$.*

(ii) ***(Frobenius)*** *Supposons que $\mathbf{A}$ est irréductible. Alors $\rho > 0$, ρ est une valeur propre simple de $\mathbf{A}$ et il existe un unique vecteur $\mathbf{x}^+ \in \mathbb{R}^n$ à coefficients strictement positifs tel que $\mathbf{A}\mathbf{x}^+ = \rho\mathbf{x}^+$ et $||\mathbf{x}^+||_1 = 1$.*

Le vecteur propre $\mathbf{x}^+$ est alors appelé ***vecteur de Perron-Frobenius*** *de $\mathbf{A}$.*

Remarque 15. *Nous précisons quelques points du théorème précédent :*

- *On dit qu'une matrice $\mathbf{A} \in \mathbb{R}^{n\times n}$ est primitive s'il existe un entier $k \geq 1$ tel que la k^{ieme} puissance de $\mathbf{A}$ est à coefficients strictement positifs. Dans ce chapitre, nous nous intéresserons davantage au (ii) du théorème 3.*
- *On dit que λ est valeur propre dominante de $\mathbf{A}$ si pour toute autre valeur propre μ de $\mathbf{A}$, $|\lambda| > |\mu|$.*
- *Soit λ une valeur propre d'une matrice $\mathbf{A}$. On dit que λ est simple si l'espace propre associé à λ est une droite, id est $dim\big(Ker(\mathbf{A} - \lambda\mathbf{I})\big) = 1$. Autrement dit dans le cas du précédent théorème,* ***tous les vecteurs propres associés à ρ seront colinéaires à $\mathbf{x}^+$.***

2.1.2 Connectivité algébrique des graphes et vecteur de Fiedler

Un des buts souvent recherchés lors du partitionnement d'un graphe est de créer des composantes disjointes des noeuds de ce graphe en minimisant le nombre d'arêtes à supprimer pour y parvenir. Pour résoudre ce problème, les algorithmes spectraux de partitionnement des graphes sont souvent basés sur le vecteur dit de Fiedler.

Définition 12. ***Laplacien d'un graphe :*** *Soit $G = (V, E)$ un graphe simple dont $\mathbf{A} \in \mathbb{R}^{n\times n}$ est la matrice d'adjacence. Soit d_i le degré du noeud i – $d_i = \sum_k a_{i,k}$ – et $\mathbf{D}$ la matrice des degrés, id est $\mathbf{D} = \mathcal{D}(\mathbf{Ae})$. On appelle Laplacien du graphe G la matrice $\mathbf{L} \in \mathbb{R}^{n\times n}$ telle que*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}.$$

On peut aussi définir $\widetilde{\mathbf{L}}$ le Laplacien normalisé de G :

$$\widetilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

On a la propriété suivante sur les Laplaciens des graphes (voir par exemple [76]) :

Propriété 4. *Soit $G = (V, E)$ un graphe simple et $\mathbf{L} \in \mathbb{R}^{n\times n}$ son Laplacien. $\mathbf{L}$ est symétrique et semi définie positive. 0 est valeur propre du graphe de mutliplicité égale au nombre de composantes connexes de G. Si G est connexe, sa deuxième plus petite valeur propre – que nous noterons λ_2 – est non nulle. On l'appelle* ***connectivité algébrique du graphe*** *[77].*

Définition 13. ***Vecteur de Fiedler :*** *Soit $\mathbf{L} \in \mathbb{R}^{n\times n}$ le Laplacien d'un graphe G. On appelle vecteur(s) de Fiedler, le(s) vecteur(s) propre(s) de $\mathbf{L}$ associé(s) à la connectivité algébrique de G.*

Il existe un lien fort entre vecteur de Fiedler et partition d'un graphe. En effet, pour un graphe simple $G = (V, E)$ possédant n un nombre pair de noeuds, supposons que l'on souhaite trouver un bi-partitionnement équilibré de V – c'est-à-dire un partitionnement de V en deux ensembles V_1 et V_2 tels que $|V_1| = |V_2|$ – qui minimise la coupe – *id est* le nombre d'arêtes entre V_1 et V_2. Ce problème peut se poser de la manière suivante :

$$\text{Trouver } \mathbf{s} \in \{-1, 1\}^n \text{ tq} : \begin{cases} \mathbf{s} = \underset{\mathbf{x}\in\{-1,1\}^n}{argmin} \sum_{(i,j)\in E} (\mathbf{x}_i - \mathbf{x}_j)^2 \\ \text{sous contrainte} : \mathbf{s}^T\mathbf{e} = 0 \end{cases}$$

et affecter ensuite les noeuds aux ensembles V_1 et V_2 :

$$\begin{cases} i \in V_1 \iff \mathbf{s}_i = +1 \\ i \in V_2 \iff \mathbf{s}_i = -1 \end{cases}$$

On remarque que l'on peut ré-écrire ce problème avec $\mathbf{L}$ le Laplacien de G :

$$\text{Trouver } \mathbf{s} \in \{-1,1\}^n \text{ tq : } \begin{cases} \mathbf{s} = \underset{\mathbf{x}\in\{-1,1\}^n}{argmin}\, \mathbf{x}^T\mathbf{L}\mathbf{x} \\ \text{sous contrainte : } \mathbf{s}^T\mathbf{e} = 0 \end{cases}$$

Ce résultat est démontré dans l'Annexe B.1.

Si l'on relâche la contrainte $\mathbf{s} \in \{-1,1\}^n$ et que l'on permet $\mathbf{s} \in \mathbb{R}^n$, alors la solution de ce problème est donnée par le vecteur de Fiedler. On peut ensuite affecter les noeuds à V_1 et V_2 en choisissant :

$$\begin{cases} i \in V_1 \iff \mathbf{v}_i > 0 \\ i \in V_2 \iff \mathbf{v}_i \leq 0 \end{cases}$$

où $\mathbf{v}$ est le vecteur de Fiedler de $\mathbf{L}$.

Pour un graphe pondéré, un résultat similaire concernant le fait que le vecteur Fiedler est une relaxation du problème de la minimisation d'un certain type de coupe – appelé coupe normalisée – peut être trouvé dans [9].

De nombreuses propriétés du Laplacien pour les graphes simples et pondérés, et de ses valeurs propres peuvent être trouvées dans [78].

Remarque 16. *D'autres interprétations sur la raison pour laquelle le vecteur de Fiedler permet de trouver un bon bi-partitionnement d'un graphe existent, notamment via la physique des ondes – voir par exemple [79].*

2.1.3 Décomposition en valeurs singulières :

Dans ce chapitre, nous évoquons de nombreuses fois les vecteurs singuliers et les vecteurs propres des matrices, de même que leurs valeurs propres et singulières. Ici, nous définissons explicitement ces notions et les mettons en lien, en commençant par la décomposition en valeurs singulières.

Théorème 4. ***Décomposition en valeurs singulières :*** *Soit* $\mathbf{M} \in \mathbb{R}^{m\times n}$*, il existe une factorisation de* $\mathbf{M}$ *de la forme*

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

avec

$$\begin{cases} \mathbf{U} \in \mathbb{R}^{m \times m} & : \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}_m \\ \mathbf{V} \in \mathbb{R}^{n \times n} & : \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}_n \\ \mathbf{\Sigma} \in \mathbb{R}_+^{m \times n} & : \quad i \neq j \implies \sigma_{i,j} = 0 \end{cases}$$

Les colonnes de $\mathbf{U}$ *– respectivement de* $\mathbf{V}$ *– sont appelées* ***vecteurs singuliers*** *gauches – respectivement droits – de* $\mathbf{M}$*. Les éléments sur la diagonale de* $\mathbf{\Sigma}$ *sont appelés* ***valeurs singulières*** *de* $\mathbf{M}$*, et on a*

$$\forall i \in \{1, ..., min(m,n)\}, \begin{cases} \mathbf{M}^T\mathbf{u}_i & = \sigma_{i,i}\mathbf{v}_i \\ \mathbf{M}\mathbf{v}_i & = \sigma_{i,i}\mathbf{u}_i \end{cases}$$

avec $\mathbf{u}_i$ *– respectivement* $\mathbf{v}_i$ *– la* i^{eme} *colonne de* $\mathbf{U}$ *– respectivement* $\mathbf{V}$*.*

Nous définissons ensuite les équations normales d'une matrice, qui vont permettre de faire le lien entre vecteurs singuliers et vecteurs propres.

Définition 14. ***Equations normales :*** *Soit une matrice* $\mathbf{M} \in \mathbb{R}^{m \times n}$*, on appelle* ***équations normales*** *de* $\mathbf{M}$ *les deux matrices carrées et symétriques :* $\mathbf{M}\mathbf{M}^T \in \mathbb{R}^{m \times m}$ *et* $\mathbf{M}^T\mathbf{M} \in \mathbb{R}^{n \times n}$*.*

Propriété 5. *Soit une matrice* $\mathbf{M} \in \mathbb{R}^{m \times n}$*. Alors, la décomposition en valeurs singulières de* $\mathbf{M}$ *:*

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

correspond à la décomposition en vecteurs propres de ses équations normales, id est :

$$\begin{cases} \forall i \in \{1, ..., m\}, \mathbf{M}\mathbf{M}^T\mathbf{u}_i = \sigma_i^2\mathbf{u}_i \\ \forall j \in \{1, ..., n\}, \mathbf{M}^T\mathbf{M}\mathbf{v}_j = \sigma_j^2\mathbf{u}_j \end{cases}$$

avec

$$\sigma_k = \begin{cases} \sigma_{k,k} & si\ k \in \{1, ..., min(m,n)\} \\ 0 & sinon \end{cases}$$

Propriété 6. *Soit une matrice carrée* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *; si* $\mathbf{A}$ *est symétrique alors sa décomposition en valeurs singulières se déduit de sa décomposition en éléments propres. Plus précisément :*

- *Si* λ *est valeur propre de* $\mathbf{A}$*, alors* $|\lambda|$ *est valeur singulière de* $\mathbf{A}$*. Réciproquement, si* σ *est valeur singulière de* $\mathbf{A}$*, alors* σ *ou* $-\sigma$ *est valeur propre de* $\mathbf{A}$*.*
- *Tout vecteur propre normé de* $\mathbf{A}$ *est vecteur singulier de* $\mathbf{A}$*.*

2.2 Equilibrage doublement stochastique

Dans cette section, nous nous intéressons au partitionnement des matrices doublement stochastiques via leurs vecteurs singuliers dominants. Notamment, nous allons voir que, si la matrice a une structure par blocs, alors ces vecteurs doivent présenter un aspect constant par morceaux permettant de détecter cette structure.

L'équilibrage doublement stochastique pour du partitionnement a déjà été proposé dans [24]. Cependant, après la phase d'équilibrage, le partitionnement est effectué par un algorithme fusionnant tour à tour les noeuds les plus fortement connectés, et ne tient pas compte de la structure constante par morceaux des vecteurs singuliers dominants. En outre, dans l'article [12], les auteurs se servent de l'équilibrage doublement stochastique pour du partitionnement par détection de la structure constante par morceaux, non pas des vecteurs singuliers mais des facteurs d'équilibrage utilisés pour rendre la matrice doublement stochastique. Enfin, les auteurs de [38] utilisent aussi une matrice doublement stochastique pour de la classification non supervisée, mais dans leur cas, cette caractéristique est celle de la matrice représentant le partitionnement, qu'ils choisissent comme la solution minimisant une divergence de Bregman avec la matrice d'affinité, après relaxation de la contrainte forçant la matrice du partitionnement à être à valeurs dans $\{0, 1\}$.

Dans la suite de cette section, nous montrons en quoi les vecteurs singuliers dominants d'une matrice doublement stochastique sont particulièrement adaptés à la détection de sa structure par blocs. Ensuite, nous discutons sur la nécessité de perturber la diagonale de la matrice lorsqu'elle est vide, notamment afin de permettre son équilibrage.

2.2.1 Vecteurs singuliers dominants et structures par blocs des matrices bi-stochastiques

Le théorème suivant est une conséquence directe du théorème 3 dit de Perron-Frobenius.

Théorème 5. *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 :*

$$\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}\},$$

où $\mathbf{v}_q^{(p)}$ *pour* $p \in \{1, \ldots, k\}$ *est tel que*

$$\mathbf{v}_q^{(p)} = \begin{cases} 1, & \text{si } q \text{ est dans le bloc } p \\ 0, & \text{sinon.} \end{cases}$$

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

$$\mathbf{x} = a_1\mathbf{v}^{(1)} + a_2\mathbf{v}^{(2)} + \cdots + a_k\mathbf{v}^{(k)}.$$

En forçant $\mathbf{x} \in \mathbf{e}^{\perp}$, on peut raisonnablement supposer que $a_i \neq a_j, \forall i \neq j$. Puisque ces vecteurs forment une partition disjointe de $\{1, \ldots, n\}$, il est possible d'identifier la contribution de chaque bloc précisément, et donc de caractériser les partitions de manière exacte, en utilisant l'ensemble $\{a_1, \ldots, a_k\}$. 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 [80], 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 5, 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 5. 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 à support total – cf la définition 10. 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 [62]. Cette méthode est peu coûteuse car elle ne fait intervenir que des produits matrices–vecteurs.

Remarque 17. *Dans le cadre de cette thèse, nous ne nous intéressons pas aux problématiques du calcul performant des vecteurs singuliers dominants. Il s'agit essentiellement de justifier théoriquement et empiriquement les différentes étapes de l'algorithme, et les matrices auxquelles nous le confrontons sont de taille relativement restreinte. Pour cette raison, nous nous permettons de traiter quelques (disons d) vecteurs singuliers dominants d'une matrice* $\mathbf{P}$ *en calculant les d vecteurs propres dominants de* $\mathbf{P}$ *orthogonaux à* $\mathbf{e}$

si $\mathbf{P}$ *est symétrique, et les d vecteurs propres dominants de* $\mathbf{PP}^T$*, respectivement* $\mathbf{P}^T\mathbf{P}$*, orthogonaux au vecteur* $\mathbf{e}$*, si* $\mathbf{P}$ *n'est pas symétrique. Dans les deux cas, nous utilisons la fonction* `eigs` *de* `Matlab`*.*

Il est évident que cette problématique devra être étudiée afin que l'algorithme soit en mesure de traiter des matrices de grande taille. En effet, le calcul des éléments singuliers d'une matrice a un coût numérique certain, et la qualité de la décomposition (totale ou partielle) obtenue dépend à la fois de la méthode utilisée et des caractéristiques de la matrice (séparabilité de ses valeurs singulières notamment). Plusieurs méthodes peuvent être envisagées ou associées, parmi lesquelles :

— *Les algorithmes basés sur la bi-diagonalisation de Golub et Kahan [81], dont l'idée principale est de simplifier le problème de la décomposition en éléments singuliers de la matrice* $\mathbf{A}$ *en s'intéressant à la décomposition en éléments singuliers d'une matrice bi-diagonale* $\mathbf{B}$*, obtenue par décomposition de* $\mathbf{A}$ *sous la forme*

$$\mathbf{A} = \mathbf{XBY},$$

où $\mathbf{X}, \mathbf{Y}$ *sont des matrices unitaires. La décomposition en éléments singuliers de* $\mathbf{B}$ *peut se faire de nombreuses façons [82]. En particulier, si l'on ne souhaite qu'une décomposition partielle, on peut utiliser les algorithmes ci-dessous.*

— *Les algorithmes basés sur la méthode de la puissance itérée par blocs – cf par exemple [83] –, où un groupe de vecteurs singuliers dominants est calculé de manière itérative, en multipliant un groupe de vecteurs par une puissance de* $\mathbf{A}$*, puis en orthonormalisant ces vecteurs. Ces méthodes peuvent être couplées avec une projection de Rayleigh-Ritz pour accélérer la convergence [84].*

— *Les algorithmes itératifs basés sur les méthodes de Krylov, qui, couplés avec des filtres polynômiaux, permettent de trouver les vecteurs singuliers dominants de* $\mathbf{A}$ *de façon relativement efficace – cf par exemple [83].*

Afin de souligner l'avantage de l'équilibrage doublement stochastique par rapport à d'autres méthodes spectrales, nous utilisons le petit exemple représenté Figure 2.2. Dans cet exemple, nous observons trois représentations matricielles d'un même graphe présentant trois classes distinctes, faiblement liées entre elles. Le graphe est donné Figure 2.2(a), le Laplacien du graphe Figure 2.2(b), le Laplacien normalisé Figure 2.2(c), et l'équilibrage doublement stochastique de la matrice d'adjacence Figure 2.2 (d). On observe, en-dessous de chaque matrice, l'information structurelle véhiculée par les vecteurs propres utilisés pour la détection de la structure par blocs : les graphes en haut des Figures 2.2 (e),(f),(g) représentent ces vecteurs dans leur ordre naturel. Dans les graphes du bas, on affiche les

mêmes vecteurs dans l'ordre croissant. On remarque que les vecteurs associés aux deux plus petites valeurs propres pour le Laplacien et le Laplacien normalisé (le vecteur de

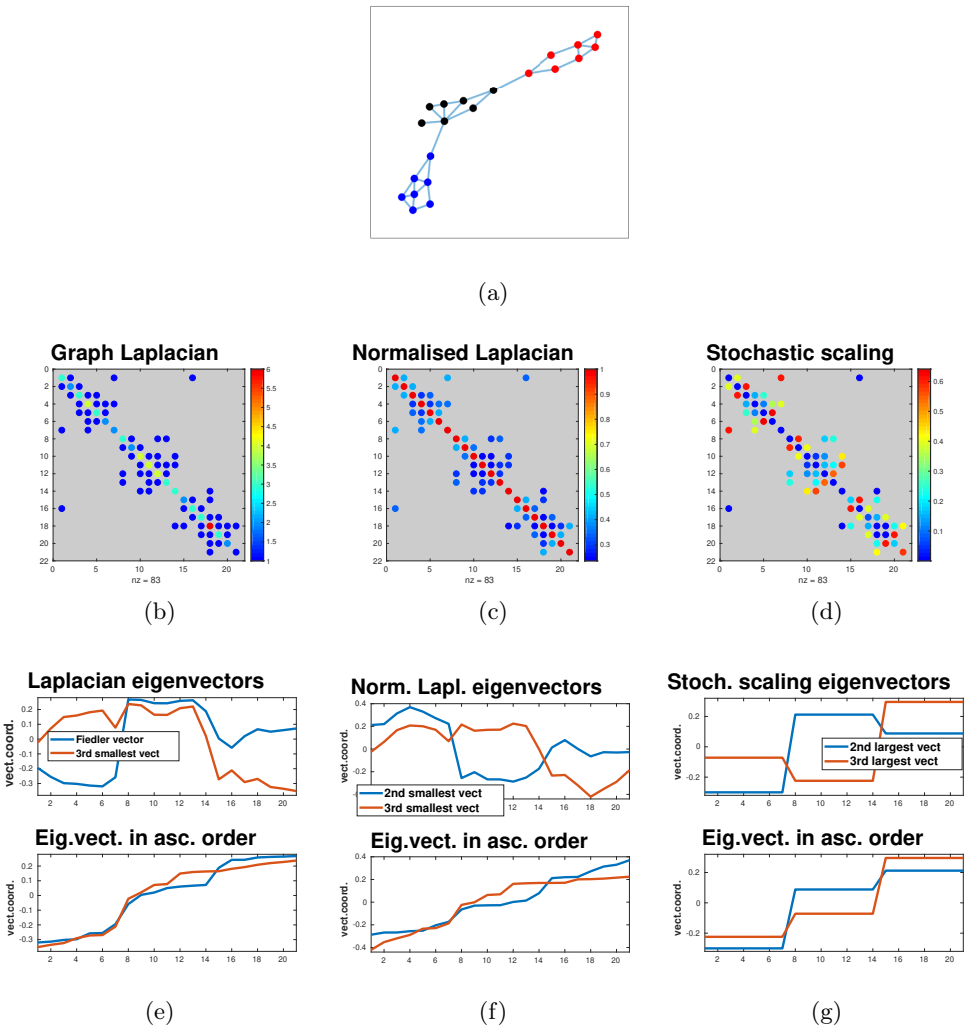


FIGURE 2.2 : *Trois représentations matricielles d'un même graphe présentant trois classes distinctes faiblement liées entre elles, et les vecteurs propres utilisés pour la détection des classes. (a) : Le graphe et ses trois composantes respectivement en bleu, noir et rouge. (b),(c),(d) : Représentations matricielles des graphes, respectivement par son Laplacien, son Laplacien normalisé et l'équilibrage doublement stochastique de sa matrice d'adjacence. (e),(f),(g) : Les vecteurs utilisés pour la détection des blocs, c'est-à-dire les vecteurs associés aux deuxième et troisième plus petites valeurs propres pour le Laplacien et le Laplacien normalisé, et ceux associés aux deuxième et troisième plus grandes valeurs propres pour la matrice doublement stochastique.*

Fiedler est en bleu) ne permettent pas de détecter nettement les classes. *A contrario*, les deuxième et troisième vecteurs propres dominants de la matrice bi-stochastique permettent de caractériser les classes, parfaitement et sans ambiguïté.

2.2.2 Perturbation de la diagonale avant équilibrage

On observe que la matrice de la Figure 2.2(d) a des éléments non nuls sur sa diagonale, alors qu'il s'agit de l'équilibrage doublement stochastique de la matrice d'adjacence d'un graphe ne présentant pas de "self-loop". En effet, comme nous l'avons précisé au théorème 2, l'existence de l'équilibrage d'une matrice sous forme doublement stochastique n'est pas trivial, une condition suffisante étant que la matrice soit bi-irréductible. Dans le cas où nous souhaitons faire du partitionnement de graphe, la matrice d'adjacence $\mathbf{A}$ est irréductible dès lors que le graphe (dirigé) représenté est (fortement) connexe. Or, nous avons vu à la remarque 14 qu'une matrice irréductible à diagonale pleine est bi-irréductible. Pour cette raison, nous ajoutons des termes non nuls dans la diagonale de la matrice traitée, si cette dernière n'est pas pleine. Cela ne change pas la structure diagonale par blocs que nous cherchons à détecter, puisque pour effectuer le partitionnement d'un graphe via sa matrice d'adjacence, nous nous limitons à des permutations symétriques des lignes et des colonnes de la-dite matrice, afin de conserver la correspondance entre un noeud du graphe et une ligne/colonne de la matrice.

Le fait que la diagonale de la matrice est pleine n'assure cependant pas la bi-irréductibilité de cette matrice. En revanche, cela permet, via une procédure de Dulmage-Mendelsohn [85], de permuter symétriquement la matrice sous une forme triangulaire par blocs [36], dans laquelle chaque bloc diagonal correspond à une composante (fortement) connexe du graphe associé. Ainsi, chaque matrice restreinte à un bloc diagonal est irréductible et à diagonale pleine, donc bi-irréductible. Nous pouvons alors appliquer notre algorithme à chaque sous-matrice restreinte à une composante connexe de façon indépendante. En effet, il est certain que deux composantes disjointes ne doivent pas être mises dans une même classe. Dans la suite, nous supposons que nous traitons une matrice correspondant à un graphe (fortement) connexe.

Ainsi, pour assurer la bi-irréductibilité des matrices que nous traitons, nous ajoutons si nécessaire des termes non nuls sur leur diagonale avant l'équilibrage. Nous fixons ces termes diagonaux à 10^{-8}. Nous justifions ce choix de façon empirique via la Figure 2.3. Sur cette figure, nous montrons que l'équilibrage de la matrice d'adjacence – affichée en (a) – du graphe donné Figure 2.2(a) ne converge pas vers une matrice doublement stochastique, comme le montre la courbe d'erreur (en (b)) de l'algorithme $\texttt{symscale}_1$ [27] appliqué cette matrice. Cet algorithme divise chaque ligne et chaque colonne de la matrice

par la racine carrée de la somme des éléments de cette ligne, respectivement de cette colonne, de manière itérative. Après chaque itération, nous mesurons l'écart entre la somme de chaque ligne de la matrice obtenue et 1, et nous remarquons qu'après quelques itérations, cet écart ne décroît plus, alors qu'il est loin d'être nul (> 0.3). Cela justifie la nécessité d'ajouter des termes non nuls à la diagonale de cette matrice avant d'effectuer son équilibrage. Nous proposons ici trois possibilités : $a_{k,k}$ vaut le degré du noeud k – cf Figure 2.3(c) –, $a_{k,k}$ vaut 1, $\forall k$ – cf Figure 2.3(d) –, ou $a_{k,k}$ vaut 10^{-8} – cf Figure 2.3(e). En dessous de chacune de ces matrices (affichées après avoir été équilibrées), on observe leurs deuxième et troisième vecteurs propres dominants – le premier est constant. On observe que l'aspect constant par morceaux des vecteurs dominants de la matrice affichée en (e) est bien plus évident que ceux des vecteurs des matrices affichées en (c) et (d). De plus, l'ajout d'éléments de faible amplitude permet de perturber au minimum la matrice d'un point de vue numérique.

En conclusion, l'exemple de la Figure 2.3 illustre la nécessité d'ajouter des éléments diagonaux dans les matrices à diagonale non pleine, afin d'assurer l'existence d'un équilibrage doublement stochastique. Cependant, le problème de savoir ce qu'il faut ajouter dans cette diagonale reste un problème ouvert. L'exemple fourni ici ne constitue qu'une justification empirique. Une étude théorique devrait être menée pour connaître et justifier la meilleure option quant au choix de ces éléments diagonaux.

2.3 Identification des classes

Le point-clé de notre algorithme est donc d'identifier les classes à l'aide des vecteurs singuliers de la matrice bi-irréductible doublement stochastique $\mathbf{P}$, et ce sans connaissance préalable du nombre de classes 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 2.2. Si $\mathbf{P}\mathbf{P}^T$ ou $\mathbf{P}^T\mathbf{P}$ a une structure sous-jacente quasi par blocs, ses vecteurs propres dominants doivent avoir une structure quasi constante par morceaux après ré-ordonnancement dans l'ordre croissant de ses entrées, comme dans la Figure 2.2(g). 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 arêtes. Nous proposons pour cela d'utiliser le filtre de Canny [86].

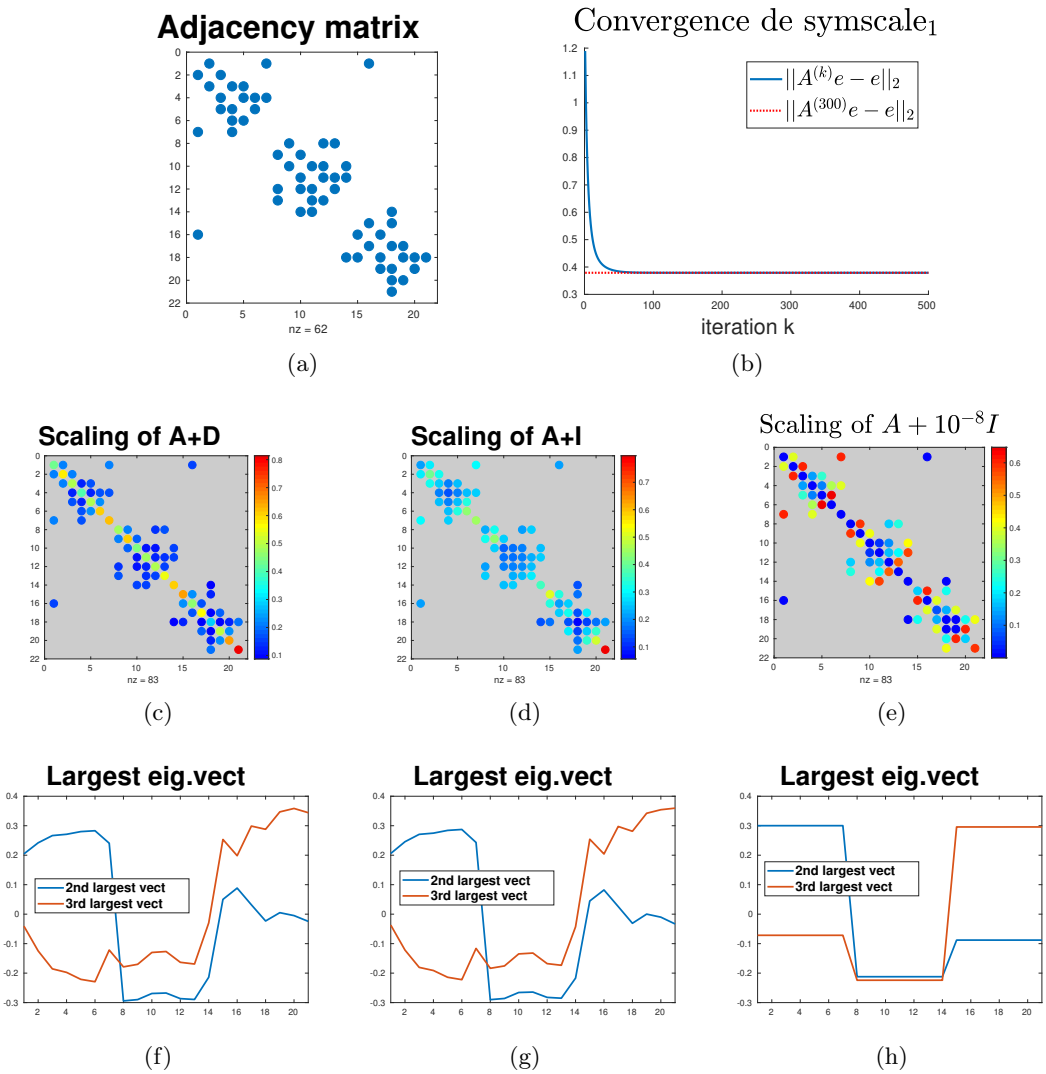


FIGURE 2.3 : *L'ajout d'éléments non nuls sur la diagonale de la matrice traitée est parfois nécessaire pour assurer l'existence d'un équilibrage doublement stochastique. Ici, la matrice affichée en (a) est à diagonale vide, et les algorithmes d'équilibrage ne permettent pas de mettre cette matrice sous forme doublement stochastique, comme le montre la courbe d'erreur en fonction des itérations de l'algorithme* `symscale`$_1$*, en (b). Nous proposons trois façons d'assurer la bi-irréductibilité de cette matrice en remplissant sa diagonale : avec le degré du noeud concerné – cf (c) –, avec* 1 *– cf (d) –, avec* 10^{-8} *– cf (e). En dessous de ces matrices sont affichés leur deuxième et troisième vecteurs propres dominants. On observe que l'aspect constant par morceaux est évident pour la matrice (e), ce qui n'est pas le cas des matrices (c) et (d).*

Le filtre de Canny consiste à effectuer 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 classes *a priori*.

Nous avons choisi d'utiliser le filtre de Canny [86], 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 [86].

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 2.4, 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.

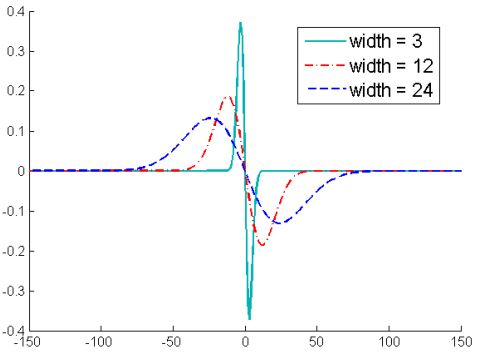
Pour éviter les effets indésirables de la taille de la fenêtre, la détection des paliers est réalisée avec deux tailles de fenêtre différentes, une grande et une petite par rapport à la taille du vecteur. Nous travaillons ensuite sur la somme des convolutions afin de détecter les pics principaux et de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

La Figure 2.4(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 [3]. Sur cette figure, il est clair que les sauts du vecteur sont mieux identifiés en utilisant la somme des convolutions. Dans la Figure 2.4(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 2.4(c) indique la bi-partition suggérée par le vecteur de Fiedler : en imposant d'avoir précisément deux classes, la séparation obtenue ne correspond finalement à aucun des paliers du vecteur.

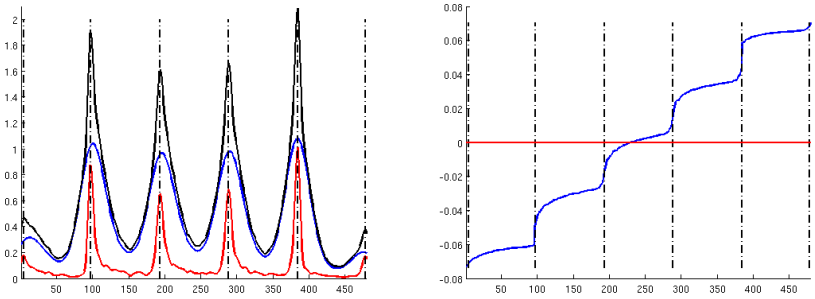
Nous avons cependant rencontré un certain nombre de difficultés lors de l'application du filtre, et nous avons mis en place des processus pour y pallier. C'est ce que nous expliquons dans la partie suivante.

2.3.1 Problèmes liés au filtre

Afin de détecter la structure par blocs d'une matrice $\mathbf{P} \in \mathbb{R}^{n \times n}$ doublement stochastique, nous appliquons le filtre de Canny sur son/ses vecteurs singuliers dominants (sauf le tout premier car il s'agit de $\mathbf{e} = (1...1)^T$), triés dans l'ordre croissant. Conformément à notre modèle, le fait de trier ces vecteurs dans l'ordre croissant doit les rendre à peu près constants par morceaux, et ainsi permettre de réorganiser les lignes et les colonnes de $\mathbf{P}$ de façon à faire apparaître sa structure par blocs. Appliquer le filtre de Canny doit détecter les sauts dans le vecteur analysé, sauts qui correspondent aux séparations entre les blocs. Un exemple est donné Figure 2.5. Sur la Figure 2.5(a) est affichée une matrice symétrique bi-stochastique présentant une structure avec 5 blocs diagonaux de taille 100 dans son ordre naturel. On observe sur le haut de la Figure 2.5(b) son vecteur dominant dans l'orthogonal de $\mathbf{e}$, trié dans l'ordre croissant. Ce vecteur présente une séparation



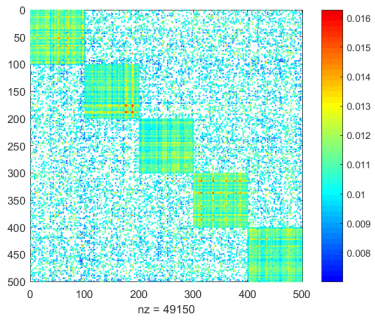
(a)



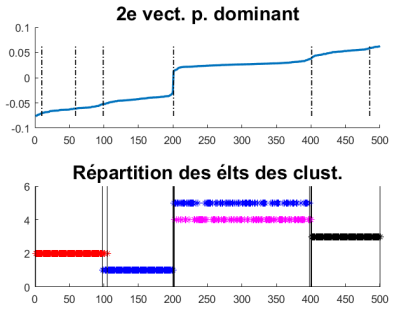
(b) (c)

FIGURE 2.4 : *(a) Le filtre pour différentes tailles de fenêtre glissante. (b) et (c) Détection des paliers.*

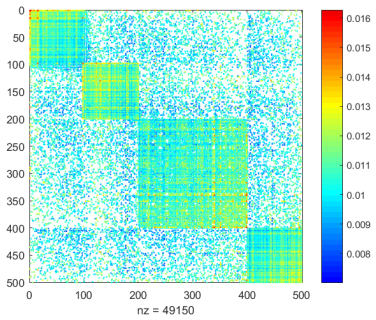
très nette à l'indice 201, et deux moins nettes autour des indices 100 et 400. Les traits en pointillés noirs correspondent aux sauts détectés par le filtre de Canny. En dessous, on affiche la répartition des éléments des classes dans ce vecteur : sur la droite $y = 1$, une astérisque dont l'abscisse vaut i signifie qu'un élément du premier bloc de la matrice (situé dans l'ordre naturel entre les indices 1 et 100) se trouve être la i^{eme} coordonnée du vecteur réordonné. Sur la droite $y = 2$, une astérisque en j signifie qu'un élément du deuxième bloc de la matrice (initialement situé entre les indices 101 et 200) est la j^{eme} cordonnée du vecteur réordonné, etc. Les traits verticaux montrent l'emplacement du premier et du dernier indice d'une classe. On peut ainsi observer que l'aspect constant par morceaux de notre vecteur propre coïncide relativement bien avec la structure par blocs de la matrice. Enfin, sur la Figure 2.5(c), on affiche la matrice de la Figure 2.5(a) après avoir réordonné ses lignes et ses colonnes conformément au tri dans l'ordre croissant



(a)



(b)



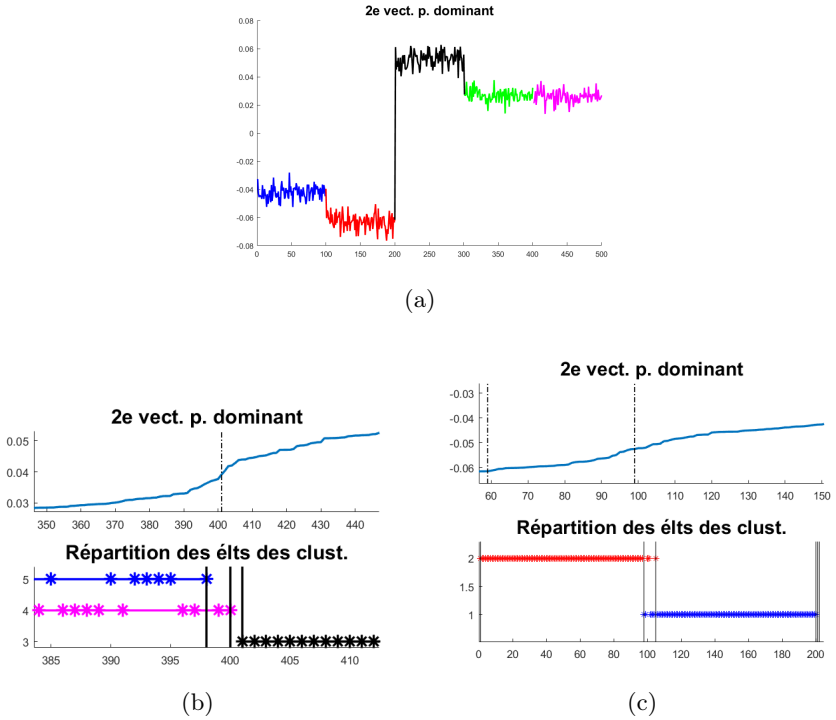
(c)

FIGURE 2.5 : *(a) : Matrice symétrique bi-stochastique avec 5 blocs diagonaux. (b)-haut : Deuxième vecteur propre dominant après tri dans l'ordre croissant. (b)-bas : Répartition des éléments des classes résultant du tri. (c) : Même matrice après permutation des lignes et des colonnes conformément au tri vecteur.*

de son vecteur propre. Deux blocs sont mélangés. Les trois autres sont relativement bien séparés.

Cependant, une observation attentive a montré qu'il était impossible de supposer que la structure par blocs proposée directement en sortie de ce processus était cohérente avec celle de la matrice. En effet, plusieurs phénomènes entrent en compte et viennent compromettre ce processus. Certains résultent du tri des vecteurs, d'autres, du produit de convolution :

— Les vecteurs singuliers dominants d'une matrice présentant une structure par blocs imparfaite dans son ordre naturel présentent ce que nous avons appelé un "effet peigne" : les coordonnées d'un vecteur qui correspondent à un même bloc sont bruitées autour d'une constante caractérisant ce bloc dans ce vecteur, constante égale à la moyenne des-dites coordonnées. Pour illustrer cet effet peigne, nous affichons sur la Figure 2.6(a) le vecteur présenté en haut de la Figure 2.5(b), cette fois dans son ordre naturel respectant la structure par blocs de la matrice. Chaque



(a)

(b) (c)

FIGURE 2.6 : *(a) : Deuxième vecteur propre de la matrice de la Figure 2.5(a) dans son ordre naturel. (b) : Zoom autour de l'indice 400 de la Figure 2.5 (b). (c) : Même chose autour de l'indice 100.*

couleur correspond à un bloc. On observe que, outre la séparation entre les deux premiers blocs et les trois derniers, on est *a priori* capable de séparer les deux premiers blocs entre eux, et de séparer le troisième bloc des quatrième et cinquième blocs. En triant le vecteur propre, les éléments d'un bloc présentant les pics les plus hauts se trouvent à côté des éléments d'un autre bloc (celui ayant la moyenne juste supérieure) présentant les pics les plus bas. Il peut être alors très difficile de faire la part des choses entre l'appartenance d'une coordonnée au bloc de droite ou de gauche, étant donné que la séparation entre les blocs se fait par une pente plus ou moins douce contenant plusieurs éléments de chacun des blocs. On dit que le tri des vecteurs propres dans l'ordre croissant a tendance à "lisser" les sauts entre les blocs. Le produit de convolution fournira alors un maximum – voire plusieurs maxima – dans cette zone, mais il est impossible d'être sûr que ce maximum séparera de façon exacte les éléments des deux blocs. C'est par exemple le cas de la séparation entre le troisième et les quatrième et cinquième blocs dans le vecteur des Figures 2.5(b) et 2.6(a) : sur la Figure 2.6(b), on affiche un zoom de la Figure 2.5(b) autour de l'indice 400. L'indice 401 permet effectivement de séparer les deux groupes de blocs. Cependant, si l'on n'a pas d'autre information que l'aspect constant par morceaux du vecteur, il n'est pas possible d'affirmer que c'est précisément cet indice qui sépare la matrice en deux groupes cohérents.

En outre, il peut y avoir des situations où l'amplitude du bruit est supérieure à la différence entre les moyennes de deux blocs successifs : dans ce cas, le tri du vecteur singulier va proposer une répartition des indices qui, quoique fausse – certains éléments du blos de droite seront à la gauche d'éléments du bloc de gauche dans le vecteur réordonné –, a une certaine cohérence : il existe un indice tel que la plupart des éléments d'un même bloc sont d'un même côté de cet indice. Alors il existe une pente au milieu du vecteur singulier réordonné, et le filtre de Canny devrait donc présenter un pic dans cette zone. Le résultat du tri et du produit de convolution va résulter en un découpage qui est incorrect. Un exemple est fourni Figure 2.6(c), en zoomant autour de l'indice 100 de la Figure 2.5(b).

— Le filtre de Canny n'a pas été conçu pour être optimal sur des données discrètes. Ainsi, même dans des cas parfaits, il arrive que des maxima soient décalés de quelques indices par rapport au saut véritable dans le vecteur, ou même qu'un maximum ne corresponde à aucun saut dans le vecteur.

— Appliquer le filtre de Canny sur le vecteur singulier, cela correspond à effectuer un produit de convolution entre le vecteur et le filtre. Comme tout produit de convolution, le vecteur qui en résulte présente des effets de bords, c'est-à-dire des

oscillations sur ses premières et dernières coordonnées. Cela vient du fait que lors de la convolution, le vecteur est implicitement prolongé à gauche et à droite par 0. Un exemple est montré sur la Figure 2.7 : l'image (a) montre un vecteur sur lequel on applique le filtre de Canny sans l'avoir préalablement prolongé. Le résultat du filtre pour deux tailles de fenêtre est donné sur l'image (b), où la partie entre les traits en pointillés correspond aux indices des produits de convolution qui seront conservés dans la somme finale – la taille du vecteur résultant de la convolution est égale à la somme des tailles du vecteur et du filtre moins 1 ; on supprime donc les extrémités pour se retrouver avec un vecteur de la taille du vecteur à analyser. Enfin, l'image (c) montre la somme des deux produits de convolution, et les traits en pointillés correspondent aux maxima, et donc aux sauts détectés dans le vecteur par le filtre. On voit qu'à ses extrémités, cette somme plonge vers des valeurs négatives, alors que le filtre de Canny sur un vecteur croissant doit normalement ne renvoyer que des valeurs positives.

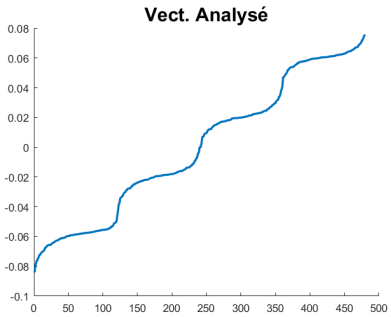
Afin d'éviter que ce phénomène d'effets de bords impacte notre résultat, nous prolongeons le vecteur singulier par une constante à chacune de ses extrémités, et dont la valeur est égale au plus petit (respectivement plus grand) élément du vecteur singulier. Cela permet de supprimer les oscillations de la convolution sur la partie qui nous intéresse. Cependant, suivant le comportement du vecteur singulier à ses bords – notamment si ses bords sont très pentus, comme c'est le cas d'un vecteur baigné dans un bruit Gaussien puis trié dans l'ordre croissant –, cela peut amener le filtre à détecter un saut au début et/ou à la fin du vecteur, saut qui n'est en réalité qu'un parasite. Ce maximum est souvent haut, et peut même être supérieur aux maxima correspondant aux véritables sauts du vecteur. Pour illustrer ce phénomène, nous reprenons le vecteur de la Figure 2.7(a) que nous prolongeons à ses extrémités : le vecteur résultant est donné Figure 2.8(a), où la prolongation est affichée en rouge. Nous observons alors, sur la Figure 2.8(c), que la somme des convolutions de la Figure 2.8(b) présente à ses extrémités des maxima, qui sont loin d'être négligeables, et dont il est clair qu'ils ne correspondent pas à de réels sauts dans le vecteur, mais aux sauts factices introduits par notre prolongation plate aux extrémités du vecteur, qui sont très pentues.

Ainsi, le filtre de Canny appliqué au vecteur singulier réordonné peut amener à la création de blocs parasites, ou au découpage de blocs cohérents. On pourrait être tenté de retourner ce découpage et laisser au processus d'amalgamation chargé de fusionner les classes – processus défini Section 2.5.2 – le soin de supprimer les blocs parasites ou de recoller les parties d'un même bloc injustement séparées. Cependant, cela ne marche

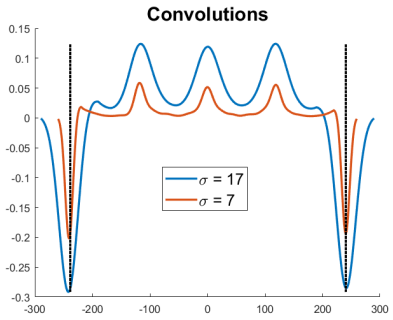
généralement pas : le processus d'amalgamation doit servir à fusionner des blocs cohérents, mais trop petits, ou ayant des valeurs numériques trop faibles, pour être considérés comme importants. S'il est utilisé pour autre chose, son comportement n'est pas prédictible. On illustre cela par la Figure 2.9. Sur les deux images de gauche, on a une matrice et une proposition de découpage par blocs représentée par les traits pleins noirs. Sur ces deux figures, une des séparations proposées n'est pas cohérente avec la structure par blocs de la matrice, et on aimerait que le processus d'amalgamation la fasse disparaître. C'est le cas pour la Figure 2.9(a), mais pas pour la Figure 2.9(b).

En conclusion, le filtre fournit trois types différents d'arêtes. Ces trois types sont illustrés Figure 2.10, où le filtre est appliqué au deuxième vecteur singulier dominant de la matrice représentée Figure 2.10(a).

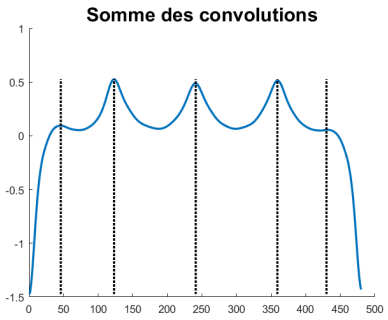
— Le premier type correspond aux arêtes correspondant à de vrais sauts dans le vecteur.



(a)



(b)



(c)

FIGURE 2.7 : *(a) : Troisième vecteur singulier gauche trié dans l'ordre croissant de la matrice* `rbsb480` *issue de [3], après équilibrage doublement stochastique. (b) : Produits de convolution avec le filtre pour deux σ différents. (c) : Somme des deux produits de convolutions. En pointillés noirs, les maxima détectés par le filtre.*

Ces arêtes peuvent cependant être placées de façon erronée par le filtre, à quelques indices de leur véritable localisation, comme c'est le cas Figure 2.10(b).

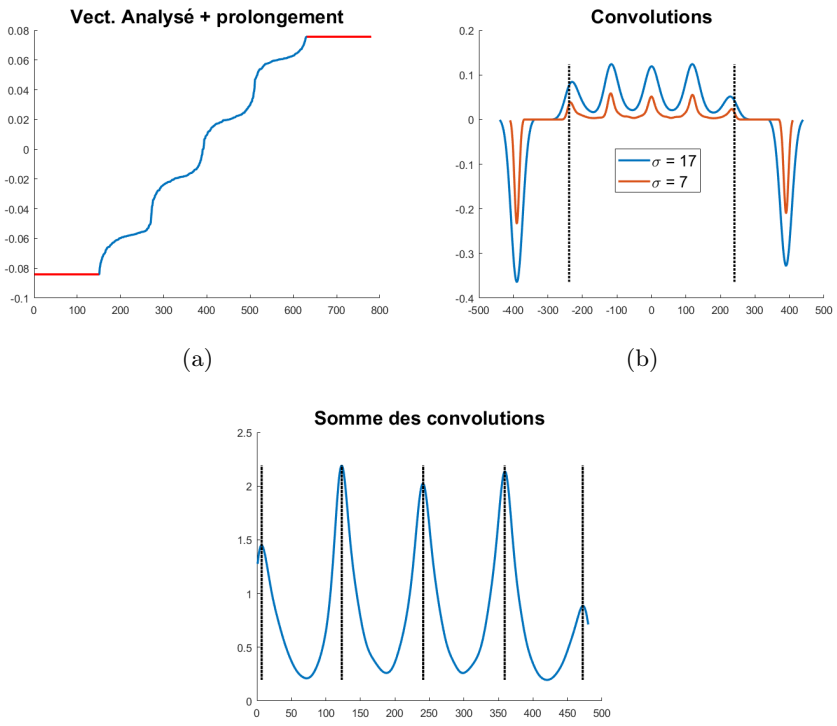
— Le deuxième type correspond aux pentes "douces" dans le vecteur et vient du fait que l'on a réordonné un vecteur bruité. Ce phénomène est montré Figure 2.10(c).

— Le troisième type d'arêtes correspond aux arêtes parasites, et peut venir du prolongement ou de phénomènes très locaux dans le vecteur analysé. Un exemple est donné Figure 2.10(d).

Pour ces raisons, il est nécessaire de développer un outil permettant *a minima* de :

— Déplacer une charnière nette qui a été mal détectée par le filtre, pour réparer les erreurs de décalages d'indices.

— Ne pas accepter des charnières en lesquelles on n'a pas suffisamment confiance, pour



(a) (b)

(c)

FIGURE 2.8 : *(a) : Troisième vecteur singulier gauche de la matrice* `rbsb480` *issue de [3] doublement stochastique avec prolongation (en rouge). (b) : Produits de convolution avec le filtre pour deux σ différents. (c) : Somme des deux produits de convolution. En pointillés noirs, les maxima détectés par le filtre.*

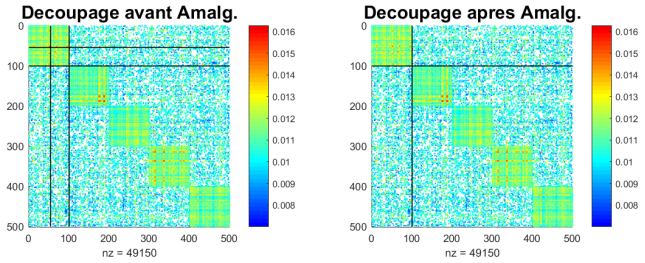
éviter de détecter des blocs correspondant à des effets de bords, ou de séparer deux blocs dont l'écart entre les valeurs moyennes n'est pas suffisante par rapport au bruit pour éviter l'entrelacement de leurs éléments lors du réordonnancement du vecteur.

— Valider avec certitude les blocs dont on est suffisamment sûrs.

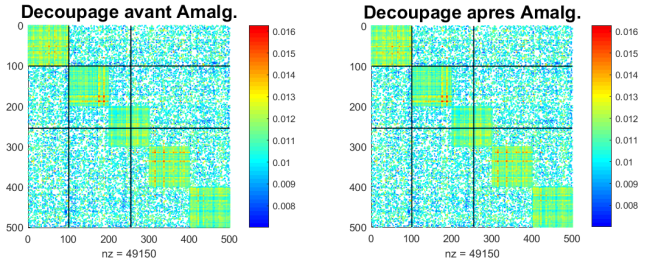
Nous allons voir dans la sous-section suivante qu'il est parfois possible d'obtenir une information complémentaire sur la structure par blocs de $\mathbf{P}$, qui peut permettre de limiter les problèmes liés à l'effet peigne.

2.3.2 Information apportée par les autres vecteurs

Si $\mathbf{P}$ est une matrice doublement stochastique présentant une structure par blocs bruitée, il est attendu que cette structure soit visible via ses vecteurs singuliers dominants

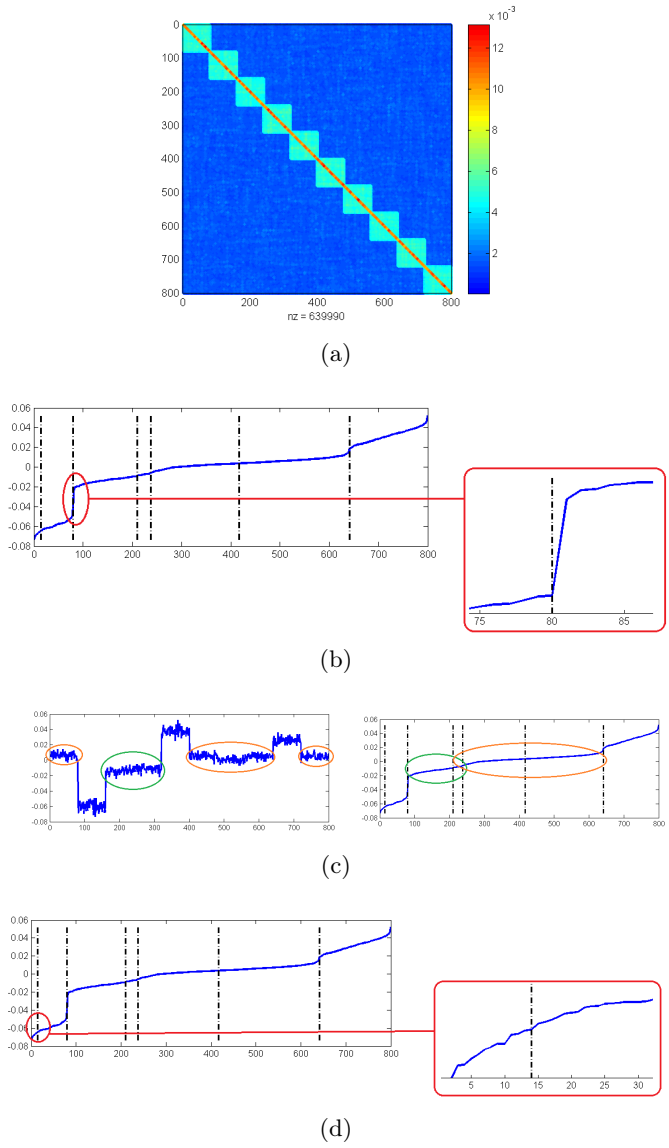


(a)



(b)

FIGURE 2.9 : *Deux propositions de découpages pour la même matrice, et le résultat du processus d'amalgamation défini Section 2.5.2 pour ce découpage sur cette matrice.*



(a)

(b)

(c)

(d)

FIGURE 2.10 : *Illustration des différents types de séparations produites par le filtre. (a) : Représentation de la matrice symétrique donc le vecteur singulier analysé est extrait. Cette matrice présente une structure ayant dix blocs diagonaux et de petites perturbations entre ces blocs. Sa structure est visible dans l'ordre naturel. (b)-gauche : en bleu, un vecteur singulier réordonné dans l'ordre croissant, et en pointillés noirs les séparations proposées par le filtre. (b)-droite : Zoom sur la séparation entourée en rouge : pour que la séparation soit correcte, elle devrait coïncider avec le premier indice du bloc de droite. Ici, il est clair qu'elle coïncide avec le dernier indice du bloc de gauche. (c)-gauche : Le vecteur singulier dans son ordre naturel. La séparation entre les blocs entourés en orange et ceux entourés en vert va résulter en une pente douce dans le vecteur réordonné. (c)-droite : La séparation entre les blocs oranges et les blocs verts est floue, certains éléments des deux ensembles peuvent être entrelacés. Le filtre a divisé cette pente douce en deux endroits. (d) : Zoom sur la première séparation retournée par le filtre : il s'agit d'un parasite introduit par la prolongation effectuée sur le vecteur pour éviter les effets de bords lors du produit de convolution.*

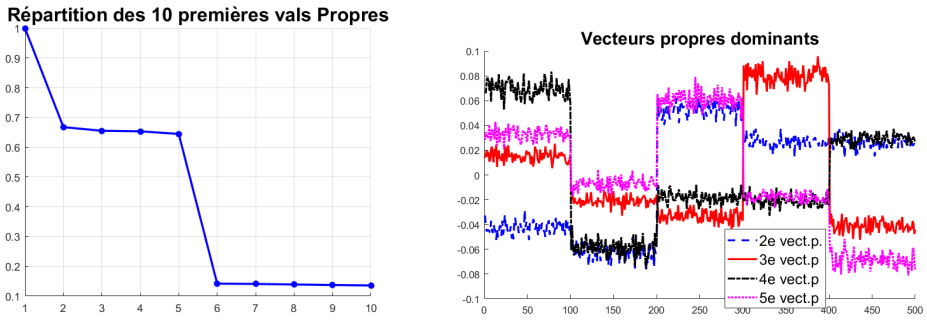
(excepté le premier) – cf [31]. Concrètement, si $\mathbf{P}$ a k blocs de lignes, on s'attend à ce que ses k premiers vecteurs singuliers droits aient une structure constante par morceaux (le premier étant simplement constant), comme on peut le voir sur la Figure 2.11. Sur cette figure, on observe les dix premières valeurs propres de la matrice symétrique de la Figure 2.5(a) – dont on rappelle qu'elle présentait 5 blocs diagonaux –, et ses deuxième, troisième, quatrième et cinquième vecteurs propres – et donc singuliers, cf la propriété 6 – dominants. Au niveau des valeurs propres, la première vaut 1, les quatre suivantes sont autour de 0.65, et on observe un net décrochage à partir de la sixième. Au niveau des vecteurs propres, on observe que chacun fournit une information sur la structure par blocs. A partir du sixième vecteur, on n'a plus d'information sur la structure par blocs. Utiliser plusieurs vecteurs permet de mettre en évidence de nombreux blocs, comme le montre la Figure 2.12 : dans ces images, chaque couleur correspond à un bloc de lignes de $\mathbf{P}$. Sur l'image (a), on voit la projection des lignes de la matrice de la Figure 2.5(a) sur son deuxième vecteur propre dominant ; sur l'image (b), sa projection sur ses deuxième et troisième vecteurs propres dominants ; et sur l'image (c), sa projection sur ses deuxième, troisième et quatrième vecteurs propres dominants. C'est-à-dire qu'en notant $\mathbf{P} \in \mathbb{R}^{n \times n}$ la matrice, $\mathbf{u}_2$, $\mathbf{u}_3$ et $\mathbf{u}_4$ respectivement ses deuxième, troisième et quatrième vecteurs propres dominants, et $\mathbf{r}_i$ le vecteur de $\mathbb{R}^n$ correspondant à la i^{eme} ligne de $\mathbf{P}$, on affiche :

— sur (a), les n points $\{X_i\}_{i=1}^n$ en 1 dimension tels que $\forall i, X_i = \mathbf{r}_i^T \mathbf{u}_2$,

— sur (b), les n points $\{Y_i\}_{i=1}^n$ en 2 dimensions tels que $\forall i, Y_i = (\mathbf{r}_i^T \mathbf{u}_2, \mathbf{r}_i^T \mathbf{u}_3)$,

— sur (c), les n points $\{Z_i\}_{i=1}^n$ en 3 dimensions tels que $\forall i, Z_i = (\mathbf{r}_i^T \mathbf{u}_2, \mathbf{r}_i^T \mathbf{u}_3, \mathbf{r}_i^T \mathbf{u}_4)$.

On peut ainsi remarquer l'apport de la prise en compte simultanée de plusieurs vecteurs singuliers dominants sur la détection des classes de lignes de la matrice $\mathbf{P}$. En effet,

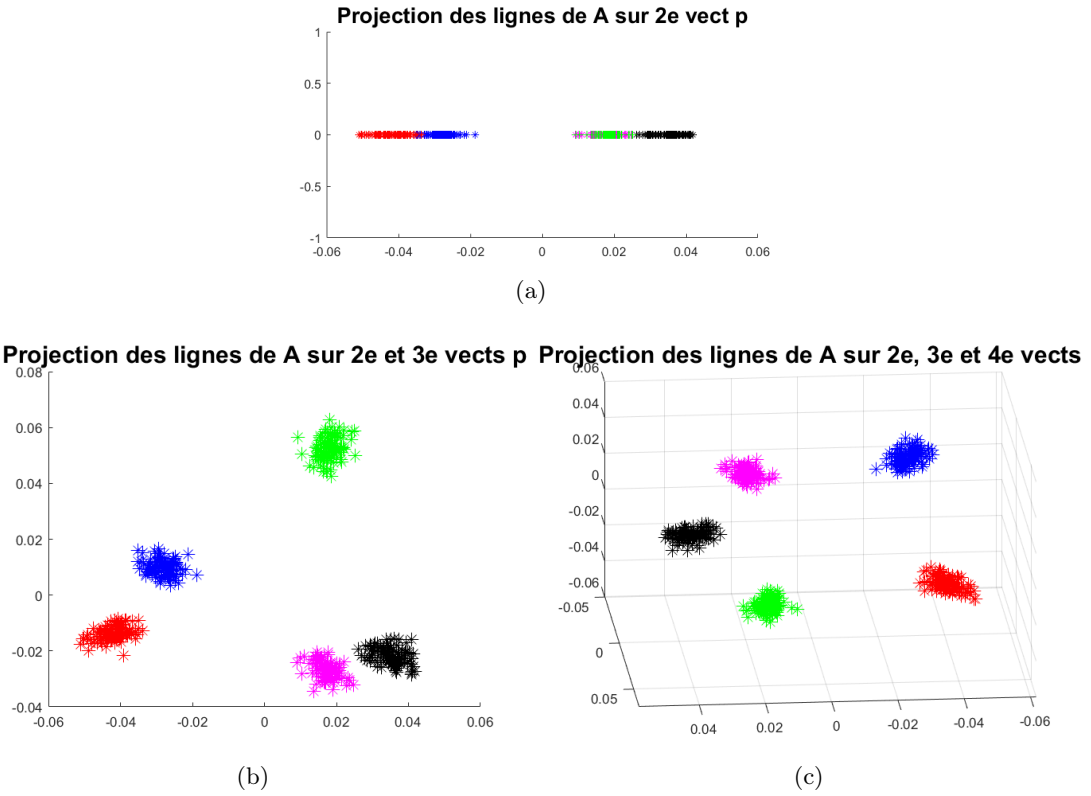


(a) (b)

FIGURE 2.11 : *(a) les dix plus grandes valeurs propres de la matrice Figure 2.5. (b) les* 2^{eme}*,* 3^{eme}*,* 4^{eme} *et* 5^{eme} *vecteurs propres dominants de la même matrice.*

la projection des lignes sur un unique vecteur permet de séparer ces lignes en deux groupes. Cette séparation est cohérente avec la structure par blocs de $\mathbf{P}$. L'intégralité de la structure est visible dès la projection sur deux vecteurs, tandis que la projection sur trois vecteurs finit d'accentuer les séparations entre les groupes rose et noir et les groupes rouge et bleu.

Cependant, à la différence des méthodes de clustering spectral classiques, le fait que $\mathbf{P}$ soit doublement stochastique impacte ses éléments spectraux de sorte que ses k premiers vecteurs peuvent potentiellement tous faire apparaître l'intégralité de la structure par blocs de lignes de $\mathbf{P}$. En pratique, il semble que toute la structure ne soit pas nettement visible sur un unique vecteur, mais que plus de deux blocs sont malgré tout détectables, comme le montre la Figure 2.11(b). Ainsi, l'information concernant un bloc peut être redondante sur plusieurs vecteurs. Le bruit appliqué aux coordonnées d'un blocs peut



(a)

(b)

(c)

FIGURE 2.12 : *Projection des lignes de* $\mathbf{P}$ *dans les espaces vectoriels définis par les* 2^{eme}, 3^{eme} *et* 4^{eme} *vecteurs propres de* $\mathbf{P}$, *soit* $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$. *Chaque couleur correspond à un bloc diagonal de* $\mathbf{P}$. *(a) : Projection sur Vect*$(\mathbf{u}_2)$. *(b) : Projection sur Vect*$(\mathbf{u}_2, \mathbf{u}_3)$. *(c) : Projection sur Vect*$(\mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4)$.

différent sur deux vecteurs différents, de même que les relations de mitoyenneté entre ces blocs.

Ainsi, combiner l'information venant de plusieurs de ces vecteurs doit permettre de mieux séparer certains blocs : en utilisant une combinaison linéaire de plusieurs vecteurs singuliers on peut obtenir un vecteur sur lequel la séparation entre deux blocs est accentuée par rapport aux vecteurs initiaux, comme cela est montré Figure 2.13. Sur cette figure, on affiche les deuxième et troisième vecteurs propres de la matrice de la Figure 2.5(a), ainsi que le vecteur égal à leur somme. Tous les trois sont triés dans l'ordre croissant, et avec pour chacun, la répartition des éléments des blocs diagonaux de $\mathbf{P}$ qui résulte de ces tris. On remarque que le deuxième bloc (astérisque en rouge) n'est pas détectable seul, ni dans le second vecteur où il est mélangé avec le premier bloc – cf Figure 2.13(a) –, ni dans le troisième où il est mélangé avec les troisième et cinquième blocs–cf Figure 2.13(b). Cependant, si l'on somme ces deux vecteurs et que l'on trie le vecteur obtenu dans son ordre croissant, on obtient le vecteur et la répartition de la Figure 2.13(c), où le deuxième bloc est cette fois-ci détectable.

Cette remarque va nous être très utile dans le développement d'un outil permettant d'affiner la détection des blocs. En effet, si nous avons plusieurs vecteurs singuliers à disposition, nous pouvons nous servir de cette information complémentaire pour peaufiner la structure par blocs retournée par le filtre. Nous pouvons notamment tenter de gommer l'effet peigne et l'incertitude qui en découle quant à l'emplacement réel de la séparation entre deux blocs, voire l'éventuel entrelacement d'indices des deux blocs.

Dans la suite, nous allons présenter deux méthodes avec lesquelles nous avons tenté de peaufiner la structure proposée par le filtre de Canny. Pour chacune, nous expliquerons et illustrerons ses limites. On supposera pour cela que l'on cherche à détecter les blocs d'une matrice $\mathbf{P} \in \mathbb{R}^{n \times n}$ doublement stochastique.

2.3.3 Approche bloc à bloc

Pour résumer nos observations précédentes :

— Avec nos exemples de matrices bruitées dont nous connaissons le partitionnement idéal, nous observons que le vecteur propre dominant dans l'orthogonal de $\mathbf{e}$ permet *a priori* de mettre en évidence certains des blocs, mais que le fait de trier ce vecteur dans l'ordre croissant va amener à lisser les séparations entre les classes, à cause du fait que le bruit va intervenir dans ce tri.

— Ce que l'on remarque, en outre, c'est que ces classes apparaissent aussi sur les vecteurs propres suivants. Ces vecteurs sont certes bruités eux-aussi, mais le bruit

peut être différent d'un vecteur à l'autre, et il est possible de faire apparaître certaines classes en combinant l'information apportée par différents vecteurs.

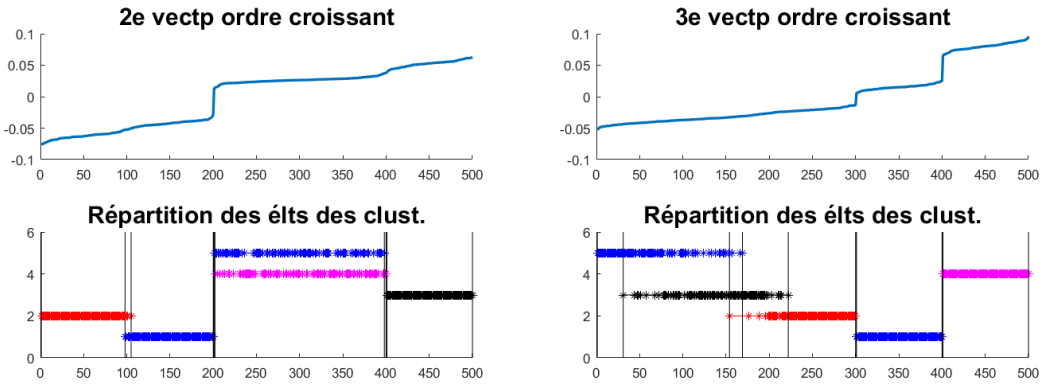
Voici alors notre premier raisonnement : si l'on soumet un bloc à un ensemble de vecteurs singuliers, on devrait être en mesure de savoir :

— si ce bloc est correct,

— si certains indices initialement mis dans ce bloc ne lui appartiennent en fait pas,

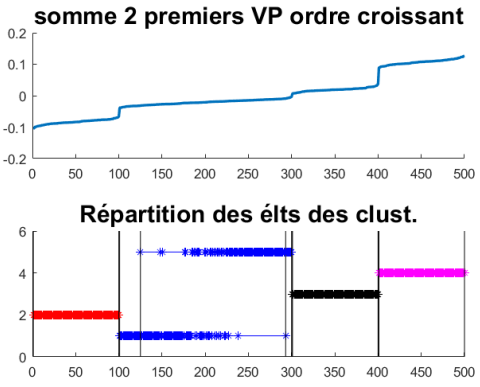
— si certains indices initialement mis hors de ce bloc lui appartiennent en fait.

La façon dont nous procédons est la suivante : nous souhaitons vérifier la cohérence d'un



(a)

(b)



(c)

FIGURE 2.13 : *(a),(b) : Deuxième et troisième vecteurs propres de* $\mathbf{P}$ *dans l'ordre croissant, et la répartition des éléments des blocs diagonaux de* $\mathbf{P}$ *qui résulte du tri. (c) : Même chose avec la somme des deux vecteurs des figures (a) et (b).*

bloc, appelons-le C_k. Pour ce faire, nous cherchons, dans l'espace vectoriel engendré par nos quelques vecteurs singuliers, le vecteur qui ressemble le plus à un vecteur constant par morceaux et qui représente le bloc en question. C'est-à-dire que nous proposons un vecteur $\mathbf{v}_{car} \in \mathbb{R}^n$ tel que

$$\mathbf{v}_{car}(i) = \begin{cases} \beta & \text{si } i \in C_k \\ \alpha & \text{sinon} \end{cases}, \text{ avec } \beta > \alpha \tag{2.1}$$

et nous allons chercher le vecteur combinaison linéaire des vecteurs singuliers qui ressemble le plus à $\mathbf{v}_{car}$. L'idée est que, parmi les vecteurs singuliers, un ou plusieurs vont avoir une structure à peu près constante pour les éléments appartenant vraiment à ce bloc. Ces vecteurs singuliers-là peuvent alors être combinés pour faire en sorte que leurs éléments dans ce bloc soient le plus proche possible de β dans la combinaison linéaire, approchant de β du même coup des éléments qui appartiennent effectivement à ce bloc mais qui possèdent la valeur α dans $\mathbf{v}_{car}$, car ils ont initialement été mal affectés. En effet, s'il y a suffisamment d'éléments bien affectés dans le bloc, la "force" qui va tirer vers β les éléments appartenant au bloc mais qui sont mal affectés devrait être assez grande pour contrer celle qui cherche à les amener vers α, étant donné l'aspect constant par morceaux des vecteurs. L'effet inverse se produit pour les éléments initialement mis dans le bloc alors qu'ils ne devraient pas l'être : ils vont être tirés vers α puisqu'ils appartiennent à des blocs dont les éléments vont être tirés vers α dans ces mêmes vecteurs singuliers.

Cela devrait permettre de mettre en évidence des éléments qui ont été mal affectés par rapport à un bloc. La Figure 2.14 présente ce procédé.

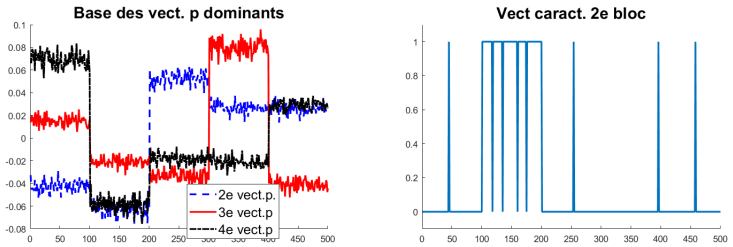
Valeurs de référence et vecteur principal

Une fois ce principe général énoncé, beaucoup de points techniques demandent à être traités. Le premier est : comment mettre à jour le partitionnement, c'est-à-dire, avec notre exemple énoncé ci-dessus, comment décider que des éléments initialement mis dans le bloc en soient retirés, *et vice versa* ?

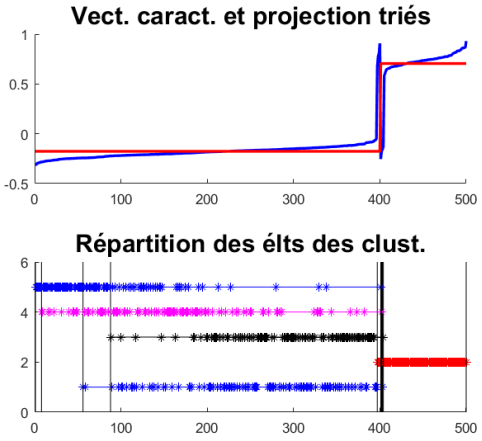
La réponse, relativement simple de prime abord, est que si la valeur d'une coordonnée du vecteur projeté est plus proche d'une valeur β_{ref}, caractérisant les éléments dans C_k, que de α_{ref}, une valeur caractérisant les éléments en dehors du bloc, alors elle doit appartenir au bloc. Elle ne doit pas y appartenir dans le cas inverse. Ainsi, notons $\mathbf{v}_p$ la projection du vecteur $\mathbf{v}_{car}$ dans la base des vecteurs singuliers à notre disposition, alors

$$\forall i \in \{1, ..., n\}, \left(i \in C_k \iff \mathbf{v}_p(i) > \frac{\alpha_{ref} + \beta_{ref}}{2} \right)$$

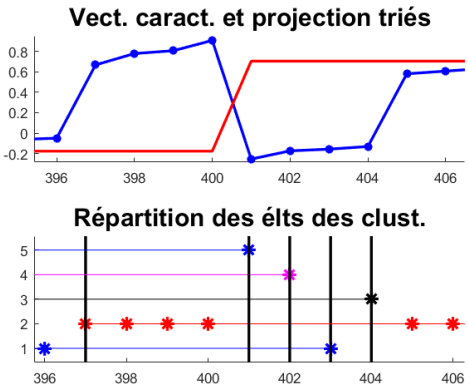
Naturellement, nous avons choisi de prendre β_{ref} et α_{ref} comme les moyennes dans le



(a)



(b)



(c)

FIGURE 2.14 : *(a)-gauche : Les trois premiers vecteurs singuliers de la matrice de la Figure 2.5(a) dans l'ordre naturel. (a)-droite : le vecteur caractéristique du bloc que l'on cherche à valider/invalider/modifier. On cherche à mettre en évidence le bloc* 2*, soit les indices compris entre* 101 *et* 200*, mais le bloc proposé ici possède quelques erreurs :* 4 *éléments du bloc sont initialement placés hors du bloc, et* 4 *éléments hors du bloc sont initialement placés dans le bloc. (b)- et (c)-haut : en rouge : vecteur caractéristique de la classe* 2*. En bleu, projection du vecteur (a)-droite dans la base de vecteurs (a)-gauche. Ces deux vecteurs bleu et rouge sont triés dans l'ordre croissant, d'abord des éléments du vecteur caractéristique – ce qui donne deux ensembles d'indices : ceux qui sont initialement dans la classe et ceux qui n'y sont pas –, puis de ceux du vecteur projeté bleu. Autour de la charnière (zoom sur la figure (c)), on voit un effet caractéristique d'une erreur dans la classe proposée : les* 4 *éléments qui devraient être dans la classe sont les quatre éléments juste avant la frontière, et qui sont "tirés vers le haut" par rapport aux autres éléments qui ne sont pas dans le bloc. A l'inverse, les* 4 *éléments mis par erreur dans le bloc sont ceux tout à fait à droite de la frontière, et qui sont, eux, "tirés vers le bas".*

vecteur $\mathbf{v}_p$ des coordonnées, respectivement du bloc C_k et de son complémentaire. Nous décrivons le processus mis en place ci-dessous.

Supposons que nous ayons d vecteurs singuliers $\mathbf{u}_1, ...\mathbf{u}_d$ à notre disposition. Nous commençons par transformer ces vecteurs de sorte qu'ils soient normés et de moyenne nulle. Cela n'impacte pas leur structure constante par morceaux, et nous supposerons donc que

$$\forall r \in \{1...d\}, \begin{cases} \sum_{i=1}^{n} \mathbf{u}_r(i) = 0 \\ \sum_{i=1}^{n} \mathbf{u}_r(i)^2 = 1 \end{cases}$$

Pour un bloc donné C_k que l'on veut vérifier/modifier, on commence par construire le vecteur $\mathbf{v}_{car}$:

$$\mathbf{v}_{car}(i) = \begin{cases} -\sqrt{\dfrac{n_k}{n(n-n_k)}} \text{ si } i \notin C_k \\ \sqrt{\dfrac{n-n_k}{nn_k}} \text{ si } i \in C_k \end{cases}$$

où $n_k = |C_k|$. Ce vecteur est l'unique vecteur normé et de moyenne nulle caractérisant le bloc C_k, et tel que $\beta > \alpha$.

On calcule ensuite $\mathbf{v}_p$ le vecteur principal de $\mathbf{v}_{car}$ dans la base définie par $\mathbf{u}_1, ...\mathbf{u}_d$, puis α_{ref} et β_{ref} tels que :

$$\begin{cases} \beta_{ref} &= \dfrac{1}{n_k} \displaystyle\sum_{i \in C_k} \mathbf{v}_p(i) \\ \alpha_{ref} &= \dfrac{1}{n-n_k} \displaystyle\sum_{j \notin C_k} \mathbf{v}_p(j) \end{cases} \tag{2.2}$$

qui seront nos valeurs de références. Nous mettons ensuite à jour le bloc C_k de la façon suivante :

$$\forall i \in \{1, ..., n\}, \left(i \in C_k \iff \mathbf{v}_p(i) > \frac{\beta_{ref} + \alpha_{ref}}{2} \right). \tag{2.3}$$

En effet, si le bloc C_k détecté par le filtre est suffisamment proche d'une vraie classe, les éléments du bloc C_k dans $\mathbf{v}_p$ devraient être proches de β_{ref}, et éloignés de α_{ref}.

Nous bouclons ensuite sur ce procédé : une fois les indices réaffectés, on a une nouvelle proposition pour le bloc C_k, et donc un nouveau vecteur caractéristique $\mathbf{v}_{car}$. On va chercher son nouveau vecteur principal, calculer les nouvelles valeurs de références β_{ref} et α_{ref}, et mettre à jour le bloc C_k, etc.

Il y a trois cas d'arrêt pour ce procédé :

1. C_k est identique à la fin de deux itérations successives : dans ce cas, $\mathbf{v}_{car}$ sera inchangé, et du même coup $\mathbf{v}_p$, etc. Il y a donc convergence.
2. il arrive une itération à laquelle $\forall i, \mathbf{v}_p(i) < \dfrac{\alpha_{ref} + \beta_{ref}}{2}$.
3. il arrive une itération à laquelle $\forall i, \mathbf{v}_p(i) > \dfrac{\alpha_{ref} + \beta_{ref}}{2}$.

Les deux derniers cas amènent nécessairement à la suppression du bloc. Pour le premier, cela dépend du vecteur principal une fois la convergence atteinte : si C_k représente effectivement un bloc de la matrice, le saut doit être bien marqué sur le vecteur principal. En revanche, il arrive aussi que l'on ait convergence du procédé alors que l'on n'observe aucun saut significatif. Il est donc nécessaire d'avoir un critère de rejet du bloc dans le cas où la pente entre le plus grand des éléments de $\mathbf{v}_p$, dont l'indice est hors de C_k, et le plus petit des éléments de $\mathbf{v}_p$, dont l'indice est dans C_k, n'est pas suffisante. Un exemple du fonctionnement du processus appliqué à un vecteur est donné aux Figures 2.15, 2.16 et 2.17. L'ébauche de l'algorithme envisagé/mis en place est donnée Figure 2.18.

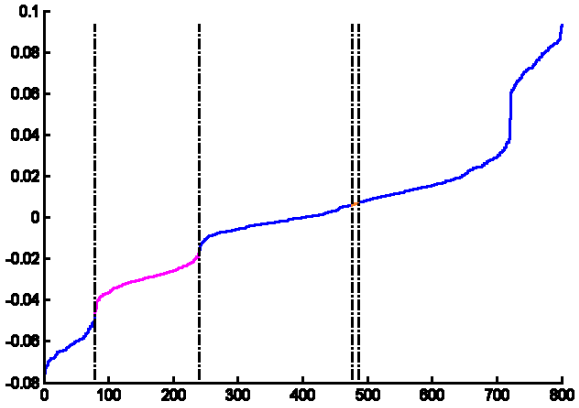
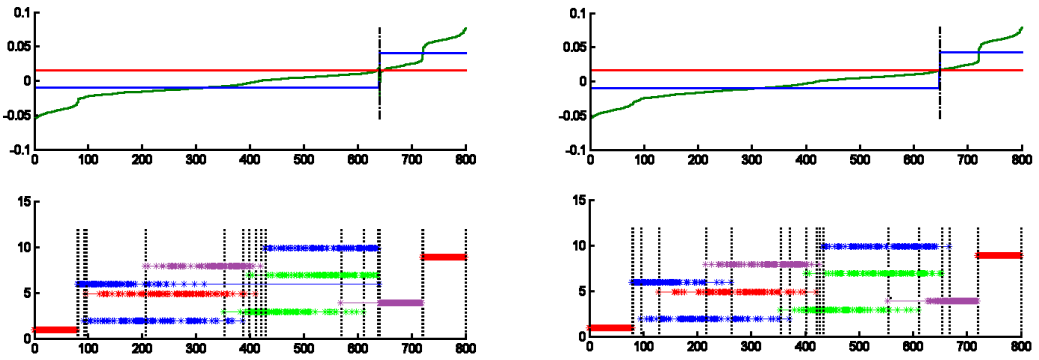


FIGURE 2.15 : *Vecteur singulier de la matrice donnée Figure 2.10(a), sur lequel l'analyse de convolution propose les classes entre les traits pointillés. Le bloc magenta sera analysé Figure 2.16, le bloc orange – séparés par les traits avant l'indice 600 – sera analysé Figure 2.17*

Problèmes liés au cas multimodal

Un problème fondamental, auquel nous avons été confronté lors de la mise en place du processus précédemment décrit, survient lorsque les vecteurs singuliers mettent en évidence plus de deux blocs. C'est ce que l'on va appeler le cas multimodal.



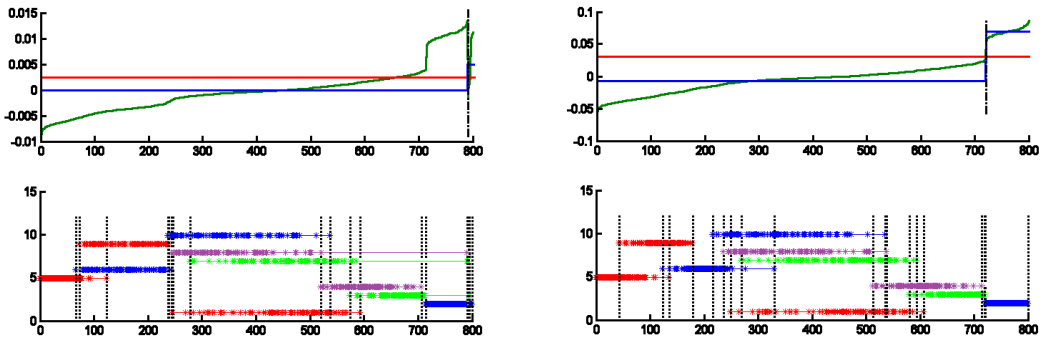
(a) (b)

FIGURE 2.16 : *Analyse du bloc magenta proposé Figure 2.15. En haut : en bleu : le vecteur caractéristique de la classe analysée, composé des valeurs de référence définies équation (2.2). En rouge, la valeur seuil. En vert, le vecteur principal, trié dans l'ordre croissant, du vecteur bleu dans l'espace des trois vecteurs singuliers dominants. En pointillés noirs, l'indice de séparation du bloc. En bas, la dispersion des classes, suivant les indices résultant du tri du vecteur principal. La convergence pour ce bloc a lieu en trois itérations. La figure (a) présente l'état initial du bloc, la figure (b), son état après convergence. Ici, il est clair que la coupure proposée va aller couper des classes qui ne devraient pas l'être. On remarque par ailleurs que le saut mis en évidence n'est pas marqué, notamment en comparaison avec ceux visibles aux indices* 81 *et* 721.

Dans ce cas, notre processus ne parvient pas à rendre constante la partie hors de C_k. Lorsque l'on trie $\mathbf{v}_p$ dans l'ordre croissant, on voit apparaître non pas un, mais plusieurs sauts, plus ou moins bien définis. Il y en a *a priori* un entre C_k et son complémentaire, mais il peut en exister d'autres. Notons ici qu'en s'intéressant aux blocs mis en évidence par l'analyse de convolution, on devrait être prémuni d'avoir plusieurs sauts dans C_k, mais pas dans le complémentaire de C_k.

Ce que l'on voit apparaître, c'est que α_{ref} représente la moyenne de plusieurs blocs. Ainsi, les éléments dont les valeurs se situent juste en dessous de celles des éléments de C_k dans $\mathbf{v}_p$ peuvent être supérieurs à la valeur seuil $\frac{\alpha_{ref} + \beta_{ref}}{2}$. En fonction de l'ensemble des éléments supérieurs à cette valeur seuil, deux cas de figures peuvent se produire :

— L'ensemble des éléments que l'on met dans C_k n'est pas très loin d'être l'union de plusieurs des blocs mis en évidence par $\mathbf{v}_p$ et le jeu de vecteurs singuliers. Par ailleurs, le nombre d'éléments mis dans C_k est à peu près égal à celui en dehors de C_k. On finit alors en général par tomber sur une séparation cohérente, dans le sens



(a) (b)

FIGURE 2.17 : *Analyse du bloc orange proposé Figure 2.15. Les légendes pour les figures haut et bas sont les mêmes que sur la figure 2.16. Ici, la convergence a lieu en* 5 *itérations. La figure (a) présente l'état initial du bloc, la figure (b), son état après convergence. La coupure finale respecte bien le découpage par blocs de la matrice, en séparant le bloc* 2 *du reste des éléments de la matrice. On remarque en outre que le saut, à l'endroit de la charnière, est extrêmement bien marqué.*

où elle découpe la matrice en deux blocs cohérents.

— L'ensemble des éléments mis dans C_k finit par tomber en plein milieu d'un bloc, c'est-à-dire dans une zone quasi constante. Cela finit par être le cas en général lorsque il n'existe pas de solution pour couper le vecteur principal en deux blocs cohérents de tailles équivalentes. On se trouve alors dans le cas où le bloc doit être rejeté dans l'algorithme présenté Figure 2.18.

De prime abord, cela peut ne pas sembler problématique. On peut en effet se dire que, peu importe que le bloc finalement trouvé corresponde bien à celui que l'on recherchait initialement, tant que la séparation finale est cohérente. Mais en fait, on va généralement retomber sur la même séparation, quel que soit le bloc que l'on essaie de mettre en évidence, tant que l'on ne change pas le jeu de vecteurs singuliers. Cela rend très coûteux un algorithme qui sera probablement inférieur, en terme de qualité des résultats, à des algorithmes de découpe bi-modale basés sur le vecteur de Fiedler. Par ailleurs, on prend aussi le risque de ne couper nulle part, alors même que des séparations nettes sur le vecteur singulier sont initialement mises en évidence par le filtre. Des exemples de ces phénomènes problématiques sont donnés Figures 2.20 et 2.19.

Le problème de cette méthode vient donc du fait que l'on considère implicitement le complémentaire de C_k comme un bloc uniforme. C'est finalement ce que l'on fait en

choisissant

$$\alpha_{ref} = \frac{1}{n - n_k} \sum_{i \notin C_k} \mathbf{v}_p(i).$$

Il faudrait donc être plus précis et baser le choix de α_{ref} sur des informations supplémentaires, notamment la connaissance d'éventuels autres blocs, qui, même s'ils ne coïncident pas parfaitement avec la structure de la matrice, devraient nous permettre d'avoir plus d'informations sur la structure constante par morceaux de $\mathbf{v}_p$.

En effet, jusqu'à présent, on approche $\mathbf{v}_p$ par un vecteur ne possédant que deux parties constantes. Or, quand plus de deux blocs sont visibles sur $\mathbf{v}_p$, cette information ne suffit pas. Il faudrait avoir des connaissances sur le bloc le plus proche de celui que l'on cherche à détecter. Or, la connaissance de l'ensemble des blocs éventuels mis en évidence par l'analyse du produit de convolution nous permet d'avoir une idée des blocs visibles. On peut donc récupérer les valeurs moyennes de chacun de ces blocs, et prendre comme valeur seuil val_{seuil} la médiane de β_{ref} et du maximum des moyennes des autres blocs. C'est-à-dire, en reprenant les notations de la Figure 2.18, qu'il faut à chaque étape

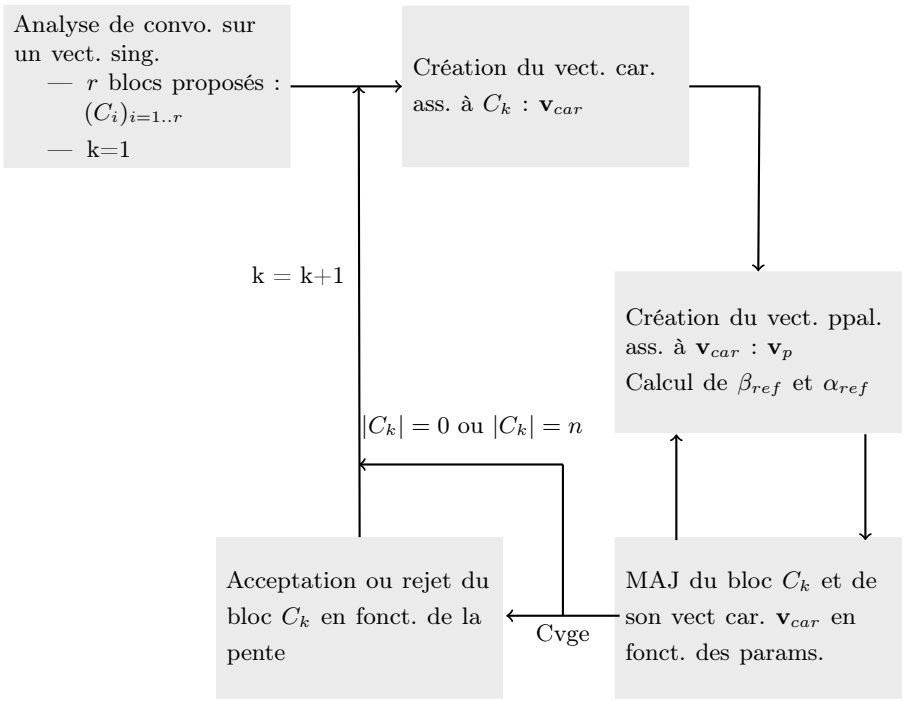


FIGURE 2.18 : *Ebauche de l'algorithme de peaufinage des blocs, bloc par bloc.*

calculer :

$$\left\{ \begin{array}{rcl} & \beta_{ref} & = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{v}_p(i) \\ \forall j \neq k, & \alpha^j_{ref} & = \frac{1}{n_j} \sum_{i \in C_j} \mathbf{v}_p(i) \end{array} \right. \tag{2.4}$$

puis

$$val_{seuil} = \frac{max_{j \neq k}\{\alpha^j_{ref}\} + \beta_{ref}}{2} \tag{2.5}$$

et enfin, mettre à jour C_k de la façon suivante :

$$\forall i \in \{1, ..., n\}, \Big(i \in C_k \iff \mathbf{v}_p(i) > val_{seuil} \Big) \tag{2.6}$$

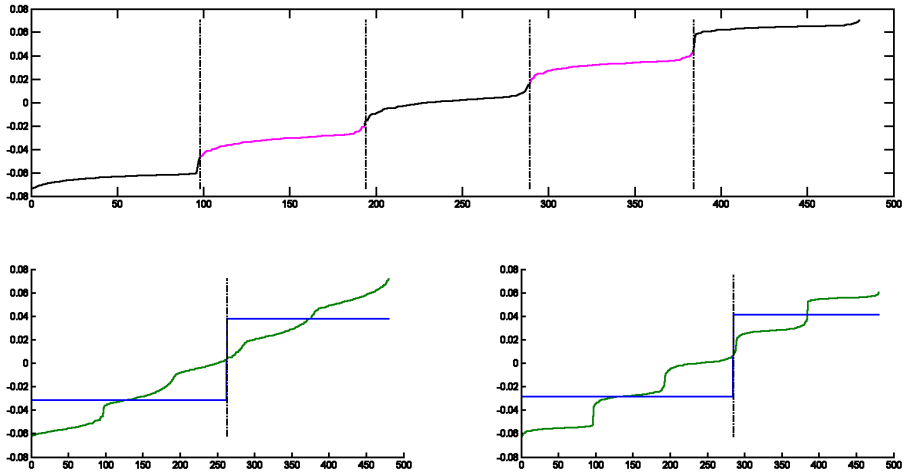


FIGURE 2.19 : *En haut : le vecteur singulier analysé. Il s'agit du vecteur colonne dominant pour la matrice* `rbsb480` *issue de [3]. Les traits en pointillés donnent les indices des séparations entre les* 5 *classes. En magenta, ce sont les deux blocs pour lesquels le résultat de l'algorithme après convergence est affiché sur les figures du bas. A gauche, il s'agit du résultat après convergence du premier bloc magenta, à droite, du second. On voit que les blocs finaux sont très différents des blocs initiaux, et aussi moins cohérents avec la vraie structure du vecteur singulier initial. Cela est lié au fait que la matrice possède un nombre impair de blocs, tous de taille à peu près équivalente.*

En fait, cela revient à tenter d'approcher $\mathbf{v}_p$ par un vecteur constant par morceaux, en supposant que les blocs détectés via l'analyse de convolution fournissent en premier lieu une approximation suffisante des zones quasi constantes.

Mais quitte à se servir de tous les blocs proposés par l'analyse de convolution, afin d'approcher un vecteur par un vecteur constant par morceaux, pour valider/invalider/modifier un bloc à la fois, autant tenter de détecter tous les blocs simultanément. C'est ce que nous allons proposer à la section suivante.

2.3.4 Version finale : approche multi blocs

Nous cherchons donc un vecteur $\mathbf{u}$, dans l'espace défini par les d vecteurs singuliers dominants $\mathbf{u}_1, ..., \mathbf{u}_d$, qui soit le plus proche possible à la fois d'un vecteur constant par morceaux et du vecteur analysé par le filtre. Pour ce faire, nous avons mis en place un processus de projections itératives dans deux sous-espaces de $\mathbb{R}^n$: le premier sous-espace est fixe, il s'agit de $\mathcal{U} = Vect(\mathbf{u}_1, ..., \mathbf{u}_d)$. Le second est le sous-espace caractérisant le

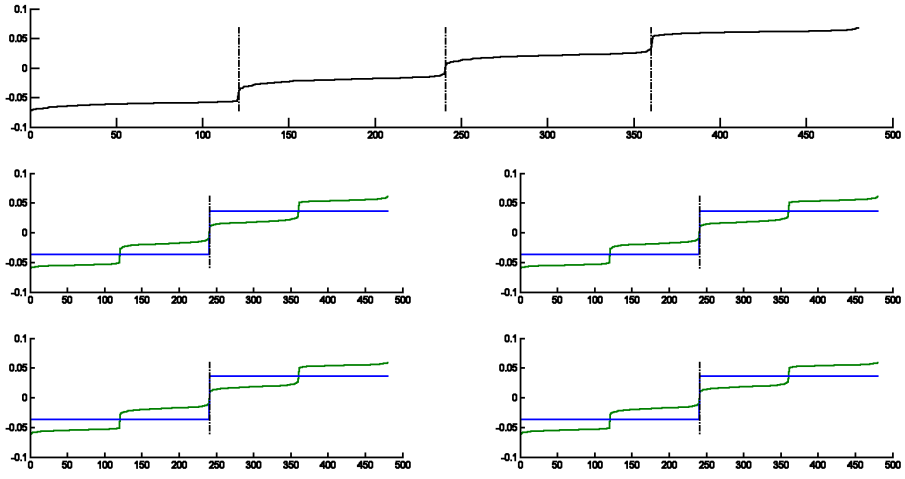


FIGURE 2.20 : *En haut : le vecteur singulier analysé. Il s'agit du vecteur ligne dominant pour la matrice* `rbsb480` *issue de [3]. Les traits en pointillés donnent les indices des séparations entre les* 4 *classes. Dessous, on a le résultat de l'algorithme après convergence pour chacun des* 4 *blocs. En fait, on va finalement trouver le même découpage pour chacun des blocs initiaux. Dans cette matrice, qui possède pourtant visiblement 4 blocs de lignes, détectés par le filtre, notre processus va, à la fin de cette itération, ne proposer que 2 blocs. Il faudra d'autres itérations pour détecter le reste de la structure via cette méthode.*

partitionnement, c'est-à-dire $\mathcal{V} = Vect(\mathbf{v}_1, ..., \mathbf{v}_r)$, avec r le nombre de classes, où un bloc C_k contenant n_k éléments est caractérisé par un vecteur $\mathbf{v}_k$ tel que

$$\forall i \in \{1...n\}, \mathbf{v}_k(i) = \begin{cases} 1/\sqrt{n_k} \text{ si } i \in C_k \\ 0 \text{ sinon} \end{cases}$$

pour $k \in \{1, .., r\}$. Ce sous-espace est modifié au cours des itérations, de même que le partitionnement.

En projetant le vecteur singulier initial $\mathbf{u}$ dans le sous-espace $\mathcal{V}$, nous obtenons un vecteur en escalier $\mathbf{v}$, dont la k^{ieme} partie constante représente la k^{ieme} classe. La valeur α_k correspondant à cette classe est utilisée comme valeur caractéristique de cette classe. Concrètement, pour un bloc C_k,

$$\alpha_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{u}(i).$$

La moyenne des valeurs caractéristiques de deux classes adjacentes est alors utilisée comme valeur seuil pour mettre à jour les classes. C'est-à-dire que

$$\forall k \in \{1...r\}, \forall i \in \{1...n\}, i \in C_k \iff \frac{\alpha_k + \alpha_{k-1}}{2} \leq \mathbf{u}(i) < \frac{\alpha_k + \alpha_{k+1}}{2},$$

en posant $\alpha_0 = -\infty$ et $\alpha_{r+1} = +\infty$. Nous obtenons alors un nouveau partitionnement, et donc un nouveau sous-espace. Nous réitérons le processus quelques fois (dans nos tests, 4 fois), dans le but d'obtenir un partitionnement stable.

A la fin de cette étape, nous calculons le vecteur $\mathbf{v}$, résultant de la projection de $\mathbf{u}$ dans le sous-espace caractérisant le partitionnement. Ce vecteur caractérise le partitionnement, et est proche du vecteur $\mathbf{u}$. Nous projetons alors $\mathbf{v}$ dans le sous-espace des vecteurs singuliers, afin que la contribution des autres vecteurs singuliers accentue les sauts dans le vecteur $\mathbf{u}$ résultant de la projection. Nous normalisons ce vecteur, et réappliquons le processus défini ci-dessus, et ainsi de suite jusqu'à ce que le partitionnement se stabilise, ce que nous reconnaissons lorsque la norme de deux vecteurs $\mathbf{u}$ consécutifs est en-dessous d'une certaine tolérance. En effet, nous montrons que si le partitionnement ne change plus, alors $\mathbf{u}$ converge vers le vecteur principal des sous-espaces $\mathcal{U}$ et $\mathcal{V}$.

Démonstration. **Convergence de la suite $\left\{\mathbf{u}^{(t)}\right\}_{t \mapsto \infty}$ vers le vecteur principal dominant associé à $\mathcal{U}$.** Dans cette preuve, nous noterons $\mathbf{U} = [\mathbf{u}_1, ...\mathbf{u}_d] \in \mathbb{R}^{n \times d}$ la matrice dont chaque colonne est un vecteur singulier, et $\mathbf{V} = [\mathbf{v}_1, ...\mathbf{v}_r] \in \mathbb{R}^{n \times r}$ la matrice dont chaque colonne est le vecteur caractéristique d'un bloc. Puisque $\{\mathbf{u}_1, ...\mathbf{u}_d\}$ et $\{\mathbf{v}_1, ..., \mathbf{v}_r\}$

sont des bases orthonormées de $\mathcal{U}$, respectivement $\mathcal{V}$, on a

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}_d$$
$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_r,$$

où $\mathbf{I}_k$ est la matrice identité de dimension k. De plus $\mathbf{U}\mathbf{U}^T$ correspond à l'opérateur de projection dans $\mathcal{U}$, respectivement $\mathbf{V}\mathbf{V}^T$ à l'opérateur de projection dans $\mathcal{V}$.

Une fois que la convergence du partitionnement est atteinte, le processus se résume très simplement à l'Algorithme 2 :

Algorithm 2: : *Edge Refinement : after convergence*

for $t = 1..\infty$ **do**
  $\mathbf{v}^{(t)} \leftarrow \mathbf{V}\mathbf{V}^T\mathbf{u}^{(t-1)}$;
  $\mathbf{u}^{(t)} \leftarrow \mathbf{U}\mathbf{U}^T\mathbf{v}^{(t)}$;
  $\mathbf{u}^{(t)} \leftarrow \mathbf{u}^{(t)}/||\mathbf{u}^{(t)}||$;
end

On va montrer que $\mathbf{u}^{(t)}$ converge vers $\tilde{\mathbf{u}}$, le vecteur principal dominant associé à $\mathcal{U}$ dans la décomposition en vecteurs principaux des sous-espaces $\mathcal{U}$ et $\mathcal{V}$.

Pour ce faire, on va commencer par expliciter $\mathbf{u}^*$ le vecteur tel que $\mathbf{u}^{(t)} \underset{t\to\infty}{\longrightarrow} \mathbf{u}^*$. Nous montrerons ensuite l'égalité entre $\mathbf{u}^*$ et $\tilde{\mathbf{u}}$.

Etape 1 : Convergence de $\mathbf{u}^{(t)}$. L'Algorithme 2 peut se ré-écrire sans intervention de $\mathbf{v}^{(t)}$:

Algorithm 3: : *Edge Refinement : after convergence*

for $t = 1..\infty$ **do**
  $\mathbf{u}^{(t)} \leftarrow \mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{u}^{(t-1)}$;
  $\mathbf{u}^{(t)} \leftarrow \mathbf{u}^{(t)}/||\mathbf{u}^{(t)}||$;
end

L'Algorithme 3 correspond à la méthode de la puissance itérée pour la matrice $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$. En outre, vu que $\mathbf{U}\mathbf{U}^T$ et $\mathbf{V}\mathbf{V}^T$ sont des projecteurs orthogonaux, alors d'après le lemme 2 de [87], on a $Spec(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T) \subset [0, 1]$. Cela signifie qu'à moins que $\mathbf{u}^{(0)}$ soit orthogonal au vecteur propre dominant de $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$ – ce qui est hautement improbable étant donnée la construction des $\mathbf{u}^{(t)}$ et de la matrice $\mathbf{V}$ –, on aura $\mathbf{u}^* = \lim_{t\to\infty} \mathbf{u}^{(t)}$ telle que :

$$\begin{cases} \mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{u}^* = \lambda\mathbf{u}^* \\ ||\mathbf{u}^*|| = 1 \end{cases} \tag{2.7}$$

avec λ la valeur propre dominante de $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$.

Etape 2 : Egalité de $\tilde{\mathbf{u}}$ et $\mathbf{u}^*$. On note

$$\mathbf{U}^T\mathbf{V} = \mathbf{Y}\mathbf{\Sigma}\mathbf{Z}^T$$

la décomposition en vecteurs principaux des espaces $\mathcal{U}$ et $\mathcal{V}$ – cf [88].

On rappelle que $\mathbf{Y}, \mathbf{Z}, \mathbf{\Sigma}$ correspond à la décomposition en valeurs singulières de $\mathbf{U}^T\mathbf{V}$ – cf le théorème 4 : $\mathbf{Y} \in \mathbb{R}^{d\times d}$ et $\mathbf{Z} \in \mathbb{R}^{r\times r}$ sont deux matrices orthonormales, et $\mathbf{\Sigma} \in \mathbb{R}^{d\times r}$ est une matrice telle que

$$\forall i, j, i \neq j \implies \sigma(i,j) = 0,$$

ordonnée ici de sorte que $|\sigma(1,1)| \geq |\sigma(2,2)| \geq ... \geq |\sigma(q,q)|$, avec $q = min(d,r)$. On notera les valeurs singulières σ_1, σ_2, etc. Le k^{ieme} vecteur principal de $\mathcal{U}$ est alors donné par $\mathbf{U}\mathbf{y}_k$, de même pour le k^{ieme} vecteur principal de $\mathcal{V}$ qui est donné par $\mathbf{V}\mathbf{z}_k$. Ainsi, $\tilde{\mathbf{u}} = \mathbf{U}\mathbf{y}_1$.

Vu que $\mathbf{Y}, \mathbf{\Sigma}, \mathbf{Z}$ est une décomposition en valeurs singulières de $\mathbf{U}^T\mathbf{V}$, et étant donnée la propriété 5, on a :

$$\begin{array}{rcl} & (\mathbf{U}^T\mathbf{V})(\mathbf{U}^T\mathbf{V})^T\mathbf{y}_1 = \sigma_1^2\mathbf{y}_1 \\ \Leftrightarrow & \mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}\mathbf{y}_1 = \sigma_1^2\mathbf{y}_1 \\ \Leftrightarrow & \mathbf{U}^T\mathbf{V}\mathbf{V}^T\tilde{\mathbf{u}} = \sigma_1^2\mathbf{y}_1 \\ \Rightarrow & \mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\tilde{\mathbf{u}} = \sigma_1^2\mathbf{U}\mathbf{y}_1 \\ \Leftrightarrow & \mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\tilde{\mathbf{u}} = \sigma_1^2\tilde{\mathbf{u}} \end{array}$$

Ainsi, $\tilde{\mathbf{u}}$ est un vecteur propre de $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$, associé à la valeur propre σ_1^2.

Montrons que $\sigma_1^2 = \lambda = \rho(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T)$:

- En reprenant l'équation (2.7), on a $\mathbf{u}^* = \frac{1}{\lambda}\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{u}^*$. Ainsi, $\mathbf{u}^* \in Im(f_{\mathbf{U}})$, avec $f_{\mathbf{U}}$ le morphisme tel que :

$$\begin{array}{rccc} f_{\mathbf{U}} : & \mathbb{R}^d & \longrightarrow & \mathbb{R}^n \\ & \mathbf{x} & \mapsto & \mathbf{U}\mathbf{x} \end{array}$$

  On a donc : $\exists \mathbf{y}^* \in \mathbb{R}^d : \mathbf{u}^* = \mathbf{U}\mathbf{y}^*$.

- Alors :

$$\begin{array}{rl} \left(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{u}^* = \lambda\mathbf{u}^*\right) & \Longleftrightarrow \left(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}\mathbf{y}^* = \lambda\mathbf{U}\mathbf{y}^*\right) \\ & \Longrightarrow \left(\mathbf{U}^T\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}\mathbf{y}^* = \lambda\mathbf{U}^T\mathbf{U}\mathbf{y}^*\right) \end{array}$$

Etant donné que les colonnes de $\mathbf{U}$ sont orthonormées, on a

$$\left(\mathbf{U}^T\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}\mathbf{y}^* = \lambda\mathbf{U}^T\mathbf{U}\mathbf{y}^*\right) \iff \left(\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}\mathbf{y}^* = \lambda\mathbf{y}^*\right)$$

Donc, si $\lambda = \rho(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T) > \sigma_1^2$, on a $\mathbf{y}^*$ vecteur propre de $\mathbf{U}^T\mathbf{V}\mathbf{V}^T\mathbf{U}$ associé à $\lambda > \sigma_1^2$, ce qui contredit la décomposition en valeurs singulières de $\mathbf{U}^T\mathbf{V}$.

Ainsi, on a $\lambda \leq \sigma_1^2$.

- De plus, σ_1^2 est une valeur propre de $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$. On a donc nécessairement $\sigma_1^2 \leq \rho(\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T) = \lambda$, d'où l'égalité.

On a donc la co-linéarité de $\mathbf{u}^*$ et $\tilde{\mathbf{u}}$, à condition que les espaces propres associés soient de dimension 1. Dans ce cas, on a $||\mathbf{u}^*|| = 1$ vu la normalisation à chaque itération dans l'Algorithme 3, et $||\mathbf{u}_1|| = 1$; en effet :

$$\tilde{\mathbf{u}}^T\tilde{\mathbf{u}} = (\mathbf{U}\mathbf{y}_1)^T(\mathbf{U}\mathbf{y}_1) = \mathbf{y}_1^T\mathbf{U}^T\mathbf{U}\mathbf{y}_1 = \mathbf{y}_1^T\mathbf{y}_1 = 1$$

avec $\mathbf{y}_1^T\mathbf{y}_1 = 1$ et $\mathbf{U}^T\mathbf{U} = \mathbf{I}_d$ vu que les colonnes de $\mathbf{Y}$, de même que celles de $\mathbf{U}$, sont orthonormées.

Ainsi, si l'espace propre associé à σ_1^2 est de dimension 1 pour $\mathbf{U}\mathbf{U}^T\mathbf{V}\mathbf{V}^T$, alors on a bien

$$\mathbf{u}^{(t)} \underset{t\to\infty}{\longrightarrow} \tilde{\mathbf{u}}$$

avec $\tilde{\mathbf{u}}$ le vecteur principal associé à $\mathcal{U}$ dans la décomposition en éléments principaux entre $\mathcal{U}$ et $\mathcal{V}$. □

A la fin de notre processus, nous devons obtenir un vecteur qui est plus proche d'un vecteur constant par morceaux que notre vecteur initial. Cependant, ce processus ne remet pas en question les arêtes proposées par le filtre. Il est donc nécessaire de mettre en place un critère d'acceptation des arêtes, afin de ne conserver que celles correspondant à des séparations dont on est sûr, afin d'éviter, notamment, de détecter de façon erronée des blocs qui n'ont rien à voir avec la vraie structure de la matrice. En effet, comme nous l'avons montré Figure 2.9, une telle erreur peut s'avérer irréparable par la suite.

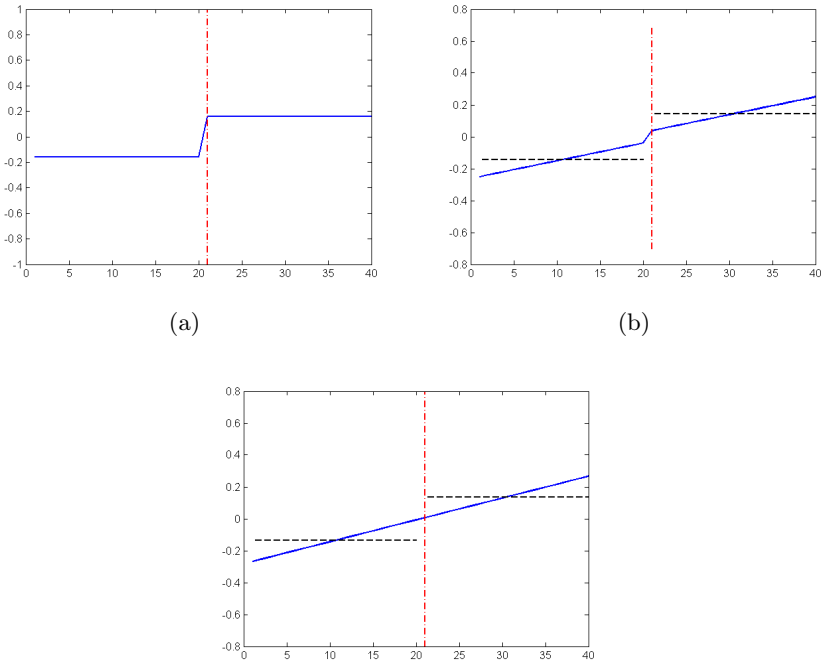
Puisque le but de notre processus itératif est d'augmenter le ratio d'aspect des véritables arêtes dans le vecteur, il est naturel, d'utiliser un rapport signal/bruit (SNR) pour évaluer si une arête doit être conservée ou non. En outre, cela est cohérent avec notre utilisation du filtre de Canny, qui consiste à envisager le problème de la détection des arêtes comme un problème de traitement du signal, en considérant notre vecteur singulier comme un signal 1D : dans un tel SNR, le signal est notre vecteur constant par morceaux

v. Ainsi, l'amplitude d'une arête est donnée par la taille du saut dans **v**. L'amplitude du bruit de fond est mesurée par l'écart entre **u** et **v** dans les blocs de chaque côté de l'arête. Le SNR d'une arête est donc donné par

$$r(e_k) = \frac{\alpha_{k+1} - \alpha_k}{\varepsilon_{k+1} + \varepsilon_k}, \tag{2.8}$$

où α_k et α_{k+1} sont les valeurs caractéristiques des blocs de chaque côté de l'arête e_k, et le bruit ε_k dans la classe C_k est donné par :

$$\varepsilon_k = \frac{1}{n_k} \sum_{i \in C_k} |\mathbf{u}(i) - \alpha_k|. \tag{2.9}$$



(a) (b)

(c)

FIGURE 2.21 : *Deux exemples de vecteurs (trait plein bleu) sur lesquels est détectée une arête (pointillés rouges). Les lignes en noir représentent les valeurs caractéristiques des blocs (omises en (a) car confondues avec le vecteur). (a) : Dans cette configuration idéale, on souhaite toujours garder l'arête, quelle que soit la taille du saut. (b) : A priori, on souhaite garder l'arête, mais cela va dépendre de la hauteur du saut. (c) : on ne veut jamais garder une telle arête, qui correspond à un artefact.*

Ce SNR est infini dans le cas d'une arête coïncidant parfaitement avec notre modèle constant par morceaux, comme cela est montré Figure 2.21(a). Une telle arête ne sera donc jamais supprimée. *A contrario*, la Figure 2.21(c) montre un exemple d'arête purement parasite, qui n'a rien à voir avec une structure constante par morceaux. Ce type de structure se rencontre lorsqu'un vecteur constant est plongé dans un bruit uniforme puis trié dans l'ordre croissant. Cette arête découpe donc un bloc uniforme en deux blocs. Nous montrons ci-dessous que le SNR d'une telle arête s'écrit

$$r(e) = 2(1 + p(c, d)), \tag{2.10}$$

où c et d sont la taille des blocs de chaque côté de l'arête e, et $p(c, d)$ est une simple fonction de c et d. Pour $c, d \geq 2$, on peut montrer que $p(c, d) \in [0, 0.125]$.

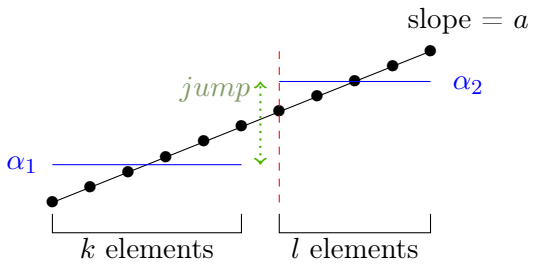


FIGURE 2.22 : *Une arête parasite (en pointillés rouges) séparant deux blocs contenant k et l éléments, et caractérisés respectivement par α_1 et par α_2*

Démonstration. Sur la configuration de la Figure 2.22

$$\begin{cases} \alpha_1 = \dfrac{1}{k}\displaystyle\sum_{i=1}^{k} a \times i = \frac{a}{2}(k+1) \\ \alpha_2 = \dfrac{1}{l}\displaystyle\sum_{i=k+1}^{k+l} a \times i = \frac{a}{l}\Big(\frac{(k+l+1)(k+l)}{2} - \frac{(k+1)k}{2}\Big) = \frac{a}{2}(l+1) + a \times k \end{cases}$$

Ainsi $jump = \alpha_2 - \alpha_1 = \dfrac{a}{2}(k + l)$.

Le bruit du bloc de droite ε_1 peut être explicité :

$$\varepsilon_1 = \frac{1}{k}\sum_{i=1}^{k} |a \times i - \alpha_1| = \frac{1}{k}\sum_{i=1}^{k} |a \times i - \frac{a}{2}(k+1)| = \frac{a}{k}\sum_{i=1}^{k} |i - \frac{k+1}{2}|$$

Si $\dfrac{k+1}{2} \in \mathbb{N}$ - *id est* k est impair -, on note $k_{1/2} = \dfrac{k+1}{2}$. Alors,

$$\begin{aligned}
\varepsilon_1 =& \frac{a}{k}\Big(\sum_{i=1}^{k_{1/2}-1} \big(k_{1/2} - i\big) + \sum_{i=k_{1/2}+1}^{k} \big(i - k_{1/2}\big)\Big) \\
=& \frac{a}{k}\Big(k_{1/2}(k_{1/2}-1) - \frac{k_{1/2}(k_{1/2}-1)}{2} + \frac{k(k+1)}{2} - \frac{k_{1/2}(k_{1/2}+1)}{2} - k_{1/2}(k-k_{1/2})\Big) \\
=& \frac{a}{k}\Big(k_{1/2}\frac{k_{1/2}-1}{2} + k_{1/2}k - k_{1/2}\frac{k_{1/2}+1}{2} - k_{1/2}\frac{k-1}{2}\Big) \\
=& \frac{ak_{1/2}}{k}\frac{k-1}{2} \\
=& \frac{a}{4k}(k^2-1)
\end{aligned}$$

Ainsi, si k est impair, alors $\varepsilon_1 = \dfrac{a(k^2-1)}{4k}$.

Si $\dfrac{k+1}{2} \notin \mathbb{N}$ - id est k est pair -, alors

$$\begin{aligned}
\varepsilon_1 =& \frac{a}{k}\Big(\sum_{i=1}^{k_{/2}} (\frac{k+1}{2} - i) + \sum_{k_{/2}+1}^{k} (i - \frac{k+1}{2})\Big) \\
=& \frac{a}{k}\Big(\frac{k+1}{2}k_{/2} - \frac{k_{/2}}{2}(k_{/2}+1) + k_{/2}(k+1) - \frac{k_{/2}(k_{/2}+1)}{2} - k_{/2}\frac{k+1}{2}\Big) \\
=& \frac{a}{2}\Big(\frac{k+1}{2} - \frac{k_{/2}+1}{2} + k + 1 - \frac{k_{/2}+1}{2} - \frac{k+1}{2}\Big) \\
=& \frac{a}{2}\Big(k + 1 - k_{/2} - 1\Big) \\
=& \frac{a}{4}k
\end{aligned}$$

Ainsi, si k est pair, alors $\varepsilon_1 = \dfrac{ak}{4}$.

On s'intéresse ensuite au bruit du bloc de gauche :

$$\varepsilon_2 = \frac{1}{l}\sum_{i=k+1}^{k+l} |a \times i - \alpha_2| = \frac{a}{l}\sum_{i=k+1}^{k+l} |\big(i - (\frac{l+1}{2} + k)\big)|$$

En substituant i par $j = i - k$, l'équation précédente devient : $\varepsilon_2 = \dfrac{a}{l}\displaystyle\sum_{j=1}^{l} |\big(j - \frac{l+1}{2}\big)|$ et l'on retombe sur la même équation que pour ε_1, avec l au lieu de k. On obtient donc la

Table 2.1.

$\varepsilon_1 + \varepsilon_2$	k impair	k pair
l impair	$\frac{a}{4}\big((k+l) - \frac{(k+l)}{kl}\big)$	$\frac{a}{4}\big(k+l-\frac{1}{l}\big)$
l pair	$\frac{a}{4}\big(k+l-\frac{1}{k}\big)$	$\frac{a}{4}(k+l)$

TABLE 2.1 : *Valeurs du bruit en fonction de la parité de l et k.*

On remarque que, quelle que soit la parité de l et k, on peut mettre le bruit sous la forme

$$\varepsilon_1 + \varepsilon_2 = \frac{a}{4}(k + l + q(k,l))$$

avec $q(k,l)$ une fonction symétrique : $q(k,l) = q(l,k)$, telle que :

$$q(k,l) = \begin{cases} 0 & \text{si } k \text{ et } l \text{ sont pairs} \\ -\frac{k+l}{kl} & \text{si } k \text{ et } l \text{ sont impairs} \\ -\frac{1}{k} & \text{si } k \text{ impair et } l \text{ pair} \end{cases}$$

On note $r(e)$ la mesure de l'arête e (en pointillés rouges sur la Figure 2.22). $r(e)$ est définie Equation (2.10) comme suit :

$$\begin{aligned} r(e) =& \frac{jump}{\epsilon_1 + \epsilon_2} \\ =& \frac{a_{/2}(k+l)}{a_{/4}(k+l+q(k,l))} \\ =& \frac{2(k+l)}{k+l+q(k,l)} \\ =& 2\Big(\frac{k+l+q(k,l)}{k+l+q(k,l)} + \frac{-q(k,l)}{k+l+q(k,l)}\Big) \\ =& 2\left(1 + \frac{-q(k,l)}{k+l+q(k,l)}\right) \end{aligned}$$

et $r(e)$ ne dépend alors pas de la pente a, mais uniquement de la taille des blocs de part et d'autre de e.

On s'intéresse au second terme de la somme ci-dessus, par rapport à la parité de k et de l :

- Si k et l sont tous les deux pairs, $q(k,l) = 0$, d'où

$$\frac{-q(k,l)}{k+l+q(k,l)} = \frac{0}{k+l} = 0$$

- Si k et l sont tous les deux impairs, $q(k,l) = -\frac{k+l}{kl}$, ainsi

$$\begin{aligned}\frac{-q(k,l)}{k+l+q(k,l)} &= \frac{\frac{k+l}{kl}}{k+l-\frac{k+l}{kl}} \\ &= \frac{k+l}{k^2l+l^2k-(k+l)} \\ &= \frac{k+l}{kl(k+l)-(k+l)} \\ &= \frac{1}{kl-1}\end{aligned}$$

- Si k impair et l pair, $q(k,l) = -\frac{1}{k}$. Alors,

$$\begin{aligned}\frac{-q(k,l)}{k+l+q(k,l)} &= \frac{1/k}{k+l-1/k} \\ &= \frac{1}{k^2+kl-1}\end{aligned}$$

- De la même façon, si k pair et l impair,

$$\frac{-q(k,l)}{k+l+q(k,l)} = \frac{1}{l^2+kl-1}$$

Ainsi, en supposant que $k, l \geq 2$, la valeur maximale pouvant être atteinte par $r(e)$ est $2(1+0.125)$, valeur obtenue pour k=3 et l=2.
La formule exacte de la mesure d'une arête dans une configuration telle que Figure 2.22 est explicitée Table 2.2.

□

Ainsi, nous avons choisi de considérer une arête comme étant parasite – et donc de ne pas la garder – si après le processus décrit précédemment, son SNR est en-dessous d'un certain seuil $\eta = 2(1+p(c,d))$. Il est aussi possible de choisir $\eta \geq 2(1+0.125)$ pour

$r(e)$	k impair	k pair
l impair	$2(1+\frac{1}{kl-1})$	$2(1+\frac{1}{l^2+kl-1})$
l pair	$2(1+\frac{1}{k^2+kl-1})$	2

TABLE 2.2 : *Valeur de la mesure dépendant de la parité de l et k.*

être plus sélectif. Dans tous les cas, le choix de ce seuil doit être un compromis entre la confiance que l'on veut avoir dans les arêtes conservées, et le nombre d'itérations globales de l'algorithme que l'on est prêt à effectuer.

Les Figures 2.23 et 2.24 montrent deux exemples des résultats obtenus via le processus de projections sur la matrice `rbsb480` issue de SuiteSparse [3], en prenant en compte 4 vecteurs singuliers. Pour ces deux figures, on affiche en haut le vecteur analysé par le filtre, les traits en pointillés représentent les séparations détectées par le filtre. En bas, on affiche le vecteur obtenu après le processus de projections itératif. Les arêtes conservées étant donné leur SNR sont affichées en rouge. Pour ces deux figures, on voit que le processus de projections permet d'accentuer les sauts existants dans le vecteur initial à des degrés différents. En outre, le SNR des arêtes est tout à fait cohérent avec l'aspect plus ou moins marqué des séparations.

Cependant, ce processus n'est toujours pas exempt d'erreur, comme le montrent les Figures 2.25 et 2.26, obtenues pour le processus de projections itératif en prenant en compte deux vecteurs singuliers, pour une matrice de l'équilibrage doublement stochastique d'un réseau artificiellement généré par le banc d'essai de Landicichinetti-Fortunato-Radicchi [65] comme présenté Section 1.3. Dans la première figure, on voit qu'il peut arriver que le processus de projections itératif échoue à détecter un saut dans le vecteur, pourtant initialement bien mis en évidence par le filtre de Canny, comme c'est le cas ici pour la deuxième arête – un peu avant l'indice 100. Dans la deuxième figure, après le processus itératif, seule la troisième arête est conservée, étant donné son SNR. Pourtant, visuellement, il n'y a pas de raison de garder celle-ci plutôt que l'avant-dernière arête, qui semble tout aussi cohérente, sinon plus. Cependant, le SNR de l'arête conservée par le processus est sensiblement supérieur au SNR de l'avant-dernière arête.

En conclusion, le processus de projections itératif, associé à l'évaluation du SNR, appliqué sur un découpage du vecteur préalablement mis en évidence par le filtre de Canny, permet en général d'améliorer la détection des arêtes faite par le filtre seul. Cependant, cet outil peut être compliqué à mettre en place, car la valeur seuil du SNR des arêtes devant être conservées peut être très différente d'une matrice à l'autre. De plus, la qualité

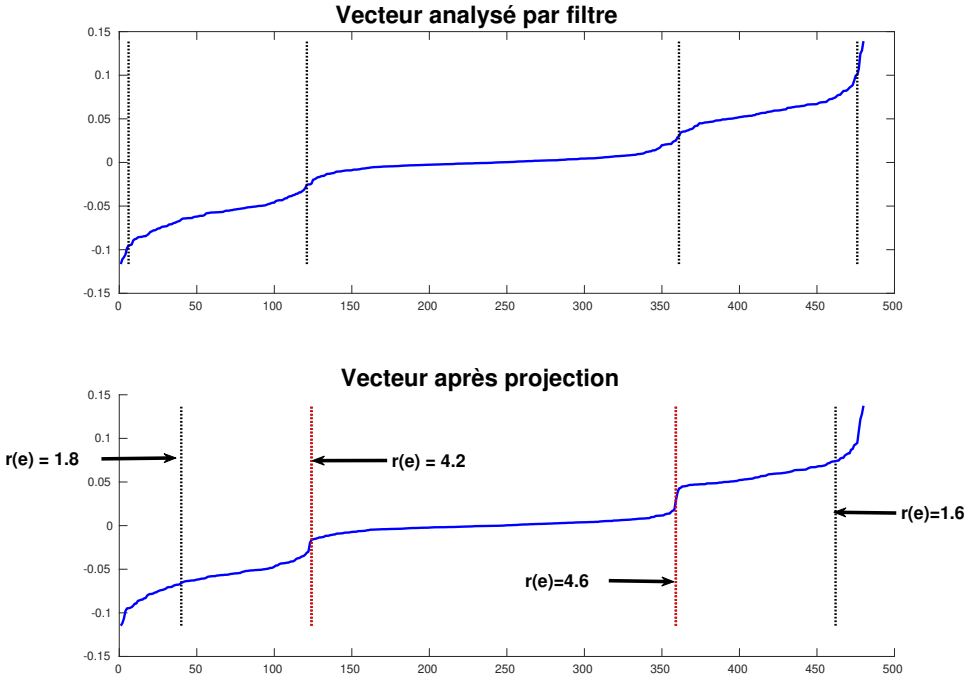


FIGURE 2.23 : *Processus de détection des arêtes par projections : en haut : un vecteur singulier gauche de la matrice* `rbsb480`*, les traits pointillés noirs représentent les arêtes retournées par le filtre de Canny appliqué à ce vecteur. En bas : le vecteur résultant du processus de projections itératif. Les arêtes conservées après analyse de leur SNR sont affichées en rouge. Le SNR de chaque arête est précisé.*

du processus des projections dépend fortement de plusieurs paramètres : la qualité du découpage initialement trouvé par le filtre, le nombre d'itérations autorisées dans le processus et le nombre des vecteurs singuliers utilisés.

2.4 SVD projetée

Notre algorithme est itératif. En effet, il est possible de boucler sur la partie "Analyse spectrale" du Schéma 2.1 afin d'affiner la structure par blocs mise en évidence par l'algorithme. La problématique traitée dans cette section concerne les vecteurs singuliers à analyser au fil des itérations. En effet, 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 2.2, comme on l'a vu à la Figure 2.11. Il est possible de trouver de

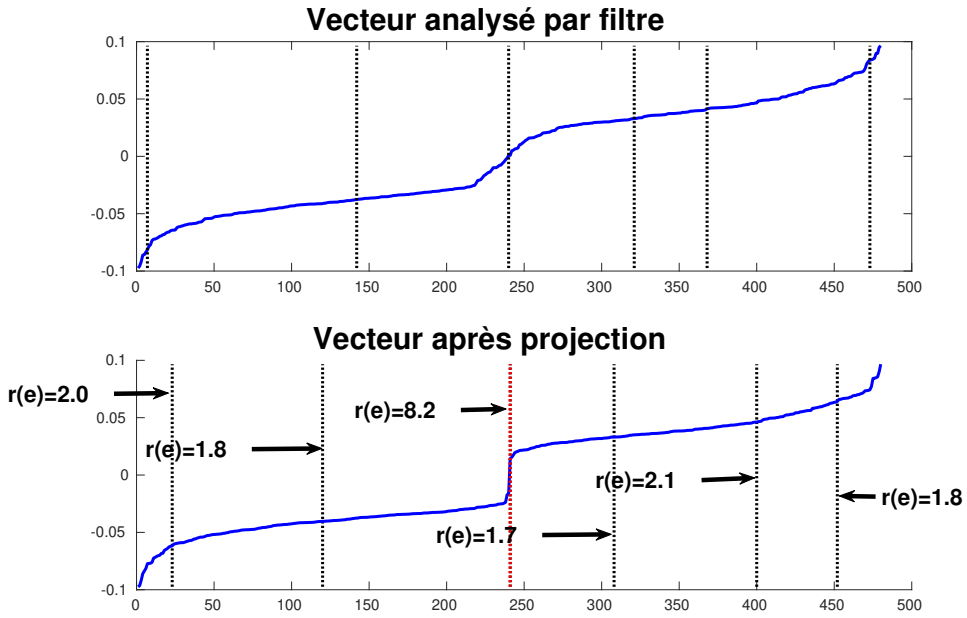


FIGURE 2.24 : *Processus de détection des arêtes par projections : en haut : un vecteur singulier droit de la matrice* `rbsb480`*, les traits pointillés noirs représentent les arêtes retournées par le filtre de Canny. En bas : le vecteur résultant du processus de projections itératif. Les arêtes conservées après analyse de leur SNR sont affichées en rouge. Le SNR de chaque arête est précisé.*

nombreux blocs avec un unique couple de vecteurs singuliers. Mais toute la structure peut ne pas être visible malgré tout, et on peut donc aussi itérer notre processus en utilisant une information complémentaire permettant d'affiner notre partitionnement. Chaque itération va nécessiter une nouvelle information spectrale, sinon nous redécouvrirons les mêmes blocs en boucle.

Lors de la première itération, la projection s'effectue généralement contre $Vect(\mathbf{e})$. 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 véhiculant aucune information sur la structure par blocs de la matrice. Pour les itérations suivantes, afin de 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. Nous notons qu'après la première itération, $\mathbf{e}$ appartient implicitement à l'espace des blocs identifiés.

Pour justifier cette approche, nous expliquons ce qui se passe dans le cas d'une matrice

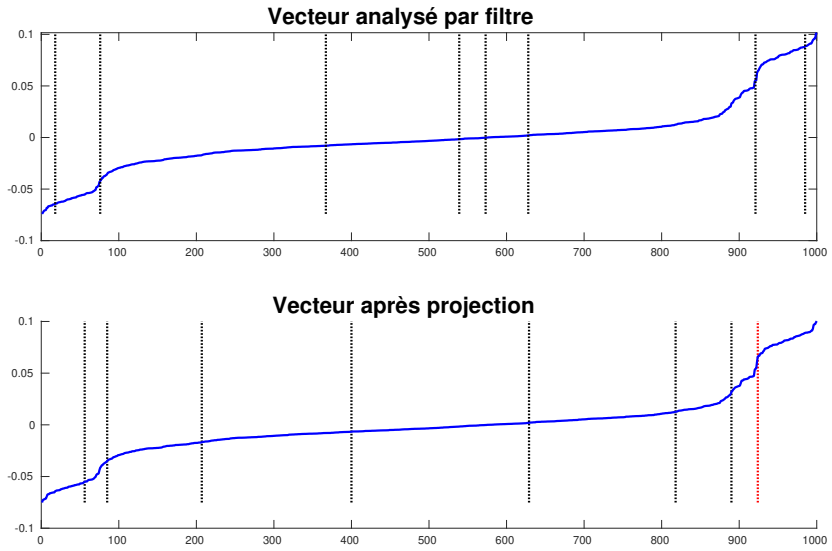


FIGURE 2.25 : *Processus de détection des arêtes par projections : en haut : un vecteur singulier dominant d'une matrice, les traits pointillés noirs représentent les arêtes retournées par le filtre de Canny. En bas : le vecteur résultant du processus de projections itératif. Les arêtes conservées après analyse de leur SNR sont affichées en rouge. On voit que la deuxième arête, initialement plutôt bien placée, a été éloignée du saut qu'elle détectait par le processus des projections.*

doublement stochastique $\mathbf{S}$ ($\mathbf{S} = \mathbf{P}\mathbf{P}^T$ ou $\mathbf{S} = \mathbf{P}^T\mathbf{P}$), possédant une structure bloc diagonale idéale :

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & & \\ & \ddots & \\ & & \mathbf{S}_p \end{bmatrix}$$

où chaque $\mathbf{S}_k$ est bi-irréductible et doublement stochastique.

Supposons que nous avons trouvé une partition de $\mathbf{S}$ en $q \leq p$ blocs $\mathbf{T}_1, ..., \mathbf{T}_q$, de tailles respectives $c_1, ..., c_q$, et que cette partition coïncide avec la vraie structure par blocs de $\mathbf{S}$, chaque bloc consistant en une réunion de blocs $\mathbf{S}_k$. Sans perte de généralité, nous supposons que notre partition coïncide avec l'ordre naturel de $\mathbf{S}$. Nous notons $\mathbf{Q}$ l'opérateur de projection dans le complément orthogonal de notre partition. La projection

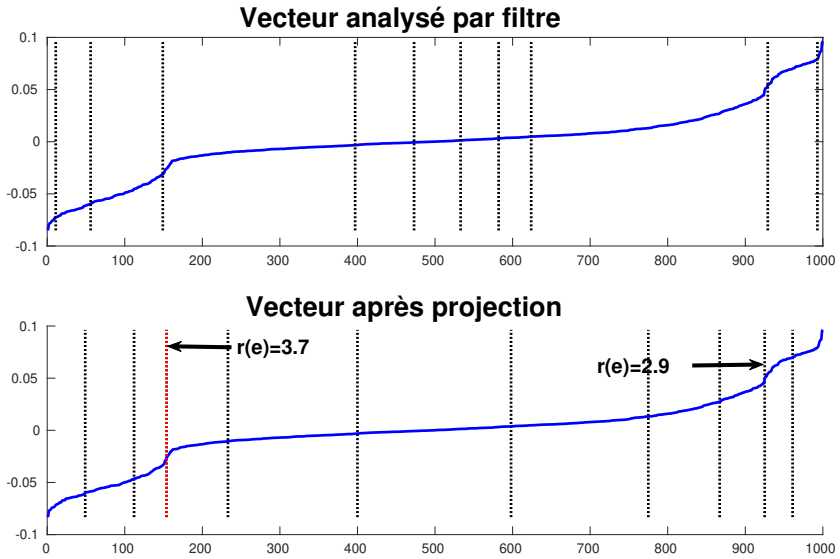


FIGURE 2.26 : *Processus de détection des arêtes par projections : en haut : un vecteur singulier dominant d'une matrice, les traits pointillés noirs représentent les arêtes retournées par le filtre de Canny. En bas : le vecteur résultant du processus de projections itératif. Les arêtes conservées après analyse de leur SNR sont affichées en rouge. Après le processus des projections, deux arêtes sont placées sur des sauts relativement bien marqués. Leur SNR est affiché, l'un est bien supérieur à l'autre, ce que leur aspect ne justifie pas.*

de $\mathbf{S}$ dans le sous-espace orthogonal à ces blocs est donné par :

$$\mathbf{QSQ} = \begin{bmatrix} \mathbf{I}_1 - \frac{1}{c_1}\mathbf{J}_1 & & \\ & \ddots & \\ & & \mathbf{I}_q - \frac{1}{c_q}\mathbf{J}_q \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 & & \\ & \ddots & \\ & & \mathbf{T}_q \end{bmatrix} \begin{bmatrix} \mathbf{I}_1 - \frac{1}{c_1}\mathbf{J}_1 & & \\ & \ddots & \\ & & \mathbf{I}_q - \frac{1}{c_q}\mathbf{J}_q \end{bmatrix}$$

où $\mathbf{I}_i$ est la matrice identité dans $\mathbb{R}^{c_i \times c_i}$, et $\mathbf{J}_i = \mathbf{e}\mathbf{e}^T$, avec $\mathbf{e} = (1...1)^T \in \mathbb{R}^{c_i}$, $\forall i \in \{1, .., q\}$.

Ainsi, pour chaque bloc $i \in \{1, .., q\}$:

$$
\begin{aligned}
(\mathbf{I}_i - \frac{1}{c_i}\mathbf{J}_i)\mathbf{T}_i(\mathbf{I}_i - \frac{1}{c_i}\mathbf{J}_i) =& (\mathbf{I}_i - \frac{1}{c_i}\mathbf{J}_i)(\mathbf{T}_i - \frac{1}{c_i}\mathbf{T}_i\mathbf{J}_i) \\
=& (\mathbf{I}_i - \frac{1}{c_i}\mathbf{J}_i)(\mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i) \qquad \text{(la somme sur les lignes de } \mathbf{T}_i \text{ vaut 1)} \\
=& \mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i - \frac{1}{c_i}\mathbf{J}_i\mathbf{T}_i + \frac{1}{{c_i}^2}{\mathbf{J}_i}^2 \\
=& \mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i - \frac{1}{c_i}\mathbf{J}_i + \frac{1}{{c_i}^2}{\mathbf{J}_i}^2 \qquad \text{(la somme sur les colonnes } \mathbf{T}_i \text{ vaut 1)} \\
=& \mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i - \frac{1}{c_i}\mathbf{J}_i + \frac{1}{c_i}\mathbf{J}_i \qquad (c_i \text{ est la dimension de } \mathbf{J}_i = \mathbf{e}\mathbf{e}^T) \\
=& \mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i
\end{aligned}
$$

En outre, pour chaque bloc $\mathbf{T}_i$, 1 est valeur propre de multiplicité égale aux nombres de blocs de $\mathbf{S}$ qui composent $\mathbf{T}_i$, d'après le théorème 5. Ainsi, il y a deux possibilités :

— Soit $\mathbf{T}_i$ est composé d'un unique bloc $\mathbf{S}_k$. Dans ce cas, le vecteur propre dominant de $\mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i$ est strictement inférieur à 1, par le théorème 3 de Perron-Frobenius. Ainsi, ce bloc ne contribuera pas au vecteur propre dominant de $\mathbf{QSQ}$. En d'autres termes, les coordonnées correspondant aux indices de ce bloc seront nulles dans le vecteur propre dominant de $\mathbf{QSQ}$.

— Soit $\mathbf{T}_i$ est une réunion de plusieurs blocs $\mathbf{S}_k$. Dans ce cas, le vecteur propre dominant de $\mathbf{T}_i - \frac{1}{c_i}\mathbf{J}_i^T$ est un vecteur propre dominant de $\mathbf{T}_i$ orthogonal à $\mathbf{e}$, qui va donc permettre de révéler les blocs individuels de $\mathbf{S}$ qui composent $\mathbf{T}_i$.

La Figure 2.27 présente un exemple de ce procédé sur une matrice bi-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 2.27(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 2.27(c). Il permet de séparer sans ambiguïté les blocs restés ensemble à l'itération précédente. Dans la Section 2.5, nous décrirons comment fusionner ces deux structures afin de découvrir la structure par blocs complète.

2.5 Amélioration des classes

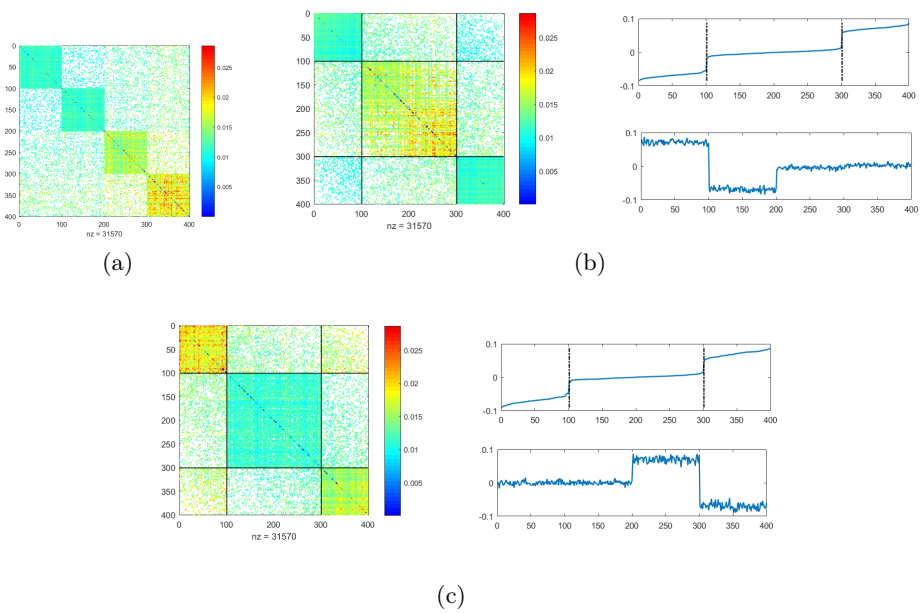
Nous expliquons maintenant comment combiner les différentes partitions en blocs de lignes ou de colonnes, trouvées pour chaque vecteur analysé individuellement, ainsi que la

superposition de la partition en blocs de lignes et celle en blocs de colonnes, obtenues une fois le processus itératif terminé, afin d'obtenir la structure diagonale par blocs résultant en un partitionnement de l'ensemble des données. 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 [3] car elle possède une intéressante structure par blocs, relativement fine, sur ses lignes et ses colonnes.

2.5.1 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).



(a) (b)

(c)

FIGURE 2.27 : *Récupération d'une information additionnelle via les vecteurs propres. (a) : Une matrice symétrique bi-stochastique. (b)-droite : En bas, le deuxième vecteur propre dominant de la matrice en (a) ; En haut, le même vecteur trié dans l'ordre croissant, avec les séparations détectées par le filtre. (b)-gauche : La structure par bloc de la matrice en (a) après analyse de ce premier vecteur. (c) : Même chose que (b), mais avec le vecteur propre dominant de la projection de la matrice dans l'espace des blocs déjà détectés.*

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 2.28. 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. 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 2.29.

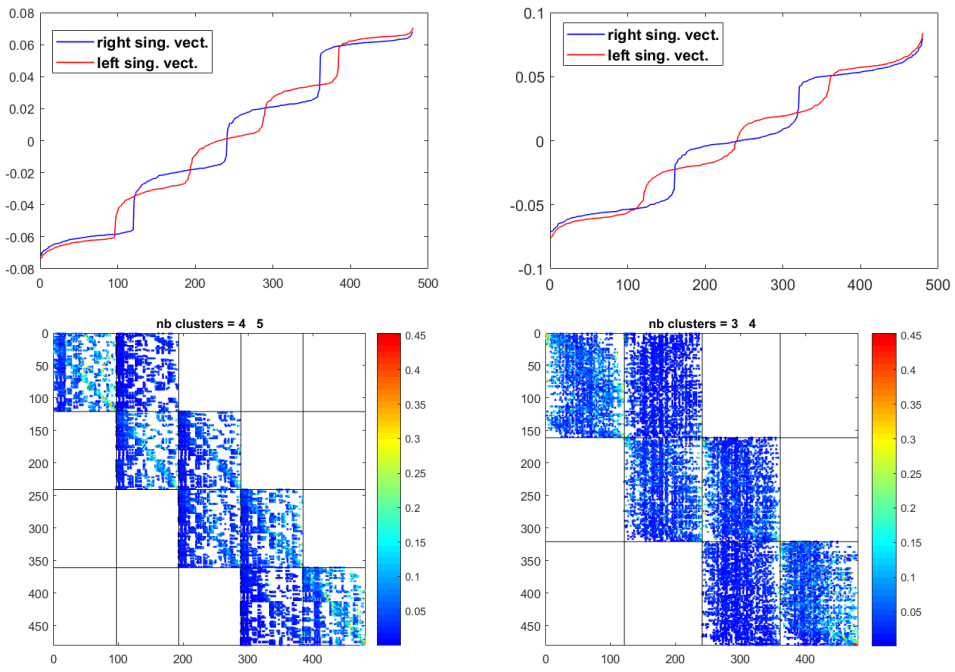


FIGURE 2.28 : *Identification de blocs en utilisant différents vecteurs singuliers.*

Nos tests sur `rbsb480` montrent l'intérêt d'analyser plusieurs vecteurs à la suite et de superposer les partitions résultantes. Cependant, ce processus peut aussi produire un partitionnement trop fin de la matrice initiale. Ce phénomène est illustré dans le schéma de droite de la Figure 2.29, où la fusion des partitions trouvées après l'analyse de trois couples de vecteurs singuliers renvoie un découpage trop fin de la matrice (48 par 100 blocs). Cela motive le développement d'une méthode efficace pour fusionner les petits blocs.

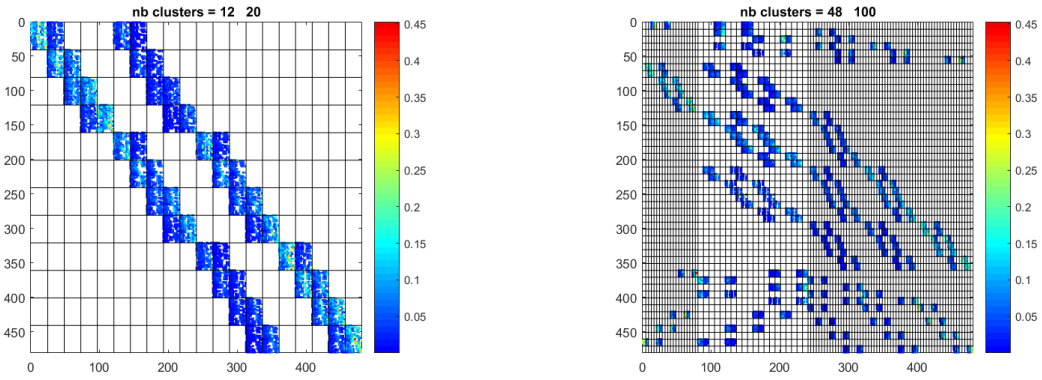


FIGURE 2.29 : *Identification récursive des blocs obtenus en analysant plusieurs vecteurs à la suite.*

2.5.2 Mesure de qualité.

Nous basons l'analyse de nos blocs sur la mesure de modularité de Newman et Girvan définie dans [42], et que nous avons étudiée Section 1.1.2. Nous rappelons que cette modularité peut être interprétée comme la somme sur toutes les classes de la différence entre la fraction de liens à l'intérieur de la classe et l'espérance de la fraction de ces liens dans un graphe aléatoire possédant la même distribution des degrés. Nous rappelons que sa formule pour une matrice symétrique $\mathbf{A} \in \mathbb{R}_+^{n \times n}$ est :

$$\mathcal{Q} = \frac{1}{2m} \sum_{i,j} (a_{i,j} - \frac{d_i d_j}{2m}) x_{i,j}$$

où d_k correspond au degré du noeud k et m est le poids total des arêtes – *id est* $d_k = \sum_{i=1}^{n} a_{i,j}$ et $2m = \sum_k d_k$. Nous avons montré en Section 1.2 que, pour une matrice $\mathbf{P}$ doublement stochastique, cette formule se simplifie en :

$$\mathcal{Q} = \sum_{i,j} (p_{i,j} - \frac{1}{n}) x_{i,j}$$

avec $\forall i, j \in \{1, ..., n\}, x_{i,j} \in \{0, 1\}$ et $x_{i,j} = 1$ si et seulement si i et j appartiennent à la même classe.

Dans le cas de notre algorithme, nous nous intéressons indépendamment aux blocs de

lignes et de colonnes. Pour notre matrice doublement stochastique $\mathbf{P}$, la partition des lignes peut être évaluée en utilisant la formule suivante :

$$\mathcal{Q}_r = \frac{1}{n}\sum_{k=1}^{r_C}\left(\mathbf{v}_k^T\mathbf{P}\mathbf{P}^T\mathbf{v}_k - \frac{1}{n}|C_k|^2\right). \tag{2.11}$$

Cette équation est directement obtenue de la formulation de la modularité fournie dans [89], en remarquant que $2m = n$ dans le cas doublement stochastique. Ici, n est le nombre de lignes de $\mathbf{P}$, r_C le nombre de blocs de lignes, $|C_k|$ est le nombre d'éléments du bloc k, et les vecteurs $\mathbf{v}_k$ sont de la forme

$$\forall i \in \{1, ..., n\}, \mathbf{v}_k(i) = \begin{cases} 1 \text{ si } i \in C_k \\ 0 \text{ sinon} \end{cases} \tag{2.12}$$

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 blocs diagonaux de $\mathbf{P}\mathbf{P}^T$ sont disjoints et de même taille.

Démonstration. On commence par montrer que $\mathcal{Q}_r \geq 0$. L'équation (2.11) se réécrit

$$\mathcal{Q}_r = \frac{1}{n}\sum_{k=1}^{r_C}\left(\mathbf{v}_k^T(\mathbf{P}\mathbf{P}^T - \frac{1}{n}\mathbf{e}\mathbf{e}^T)\mathbf{v}_k\right). \tag{2.13}$$

$\mathbf{P}\mathbf{P}^T$ est semi définie-positive. En outre, $\mathbf{e}/\sqrt{n}$ est vecteur propre normé de $\mathbf{P}\mathbf{P}^T$ associé à la valeur propre 1. Ainsi $\mathbf{P}\mathbf{P}^T - \frac{1}{n}\mathbf{e}\mathbf{e}^T$ est aussi semi définie-positive. Il en résulte que :

$$\forall k \in \{1, ..., r_c\}, \mathbf{v}_k^T(\mathbf{P}\mathbf{P}^T - \frac{1}{n}\mathbf{e}\mathbf{e}^T)\mathbf{v}_k \geq 0$$

Ainsi, $\mathcal{Q}_r \geq 0$.

On montre maintenant que $\mathcal{Q}_r \leq 1 - \frac{1}{r_C}$. Pour ce faire, on réécrit l'équation (2.11) sous la forme :

$$\mathcal{Q}_r = \frac{1}{n}\sum_{k=1}^{r_C}\mathbf{v}_k^T\mathbf{P}\mathbf{P}^T\mathbf{v}_k - \frac{1}{n^2}\sum_{k=1}^{r_C}|C_k|^2.$$

Déjà, on a

$$\frac{1}{n}\sum_{k=1}^{r_C}\mathbf{v}_k^T\mathbf{P}\mathbf{P}^T\mathbf{v}_k \leq 1.$$

En effet,

$$\sum_{k=1}^{r_C}\mathbf{v}_k^T\mathbf{P}\mathbf{P}^T\mathbf{v}_k \leq \rho(\mathbf{P}\mathbf{P}^T)\sum_{k=1}^{r_C}\mathbf{v}_k^T\mathbf{v}_k = 1 \times \sum_{k=1}^{r_C}\mathbf{v}_k^T\mathbf{v}_k = \sum_{k=1}^{r_C}|C_k| = n$$

et l'égalité est atteinte quand les $\mathbf{v}_k$, définis à l'équation (2.12), sont des vecteurs propres de $\mathbf{P}\mathbf{P}^T$ associés à 1, c'est-à-dire quand $\mathbf{P}\mathbf{P}^T$ possède r_C blocs diagonaux bi-irréductibles, comme précisé dans le théorème 5.

De plus,

$$\frac{1}{n^2}\sum_{k=1}^{r_C}|C_k|^2 \geq \frac{1}{r_C}.$$

En effet, notons $\mathbf{r} \in \mathbb{R}^{r_C}$ le vecteur tel que $\mathbf{r}(k) = |C_k|,\ \forall k \in \{1,..,r_C\}$.

Alors, par l'inégalité de Cauchy-Schwarz, on a

$$\begin{aligned}
&\sum_{k=1}^{r_C}|C_k| = \mathbf{r}^T\mathbf{e} \leq ||\mathbf{r}|| \times ||\mathbf{e}|| = \sqrt{\sum_{k=1}^{r_C}|C_k|^2} \times \sqrt{r_C} \\
\iff &\sqrt{\sum_{k=1}^{r_C}|C_k|^2} \geq \frac{1}{\sqrt{r_C}}\sum_{k=1}^{r_C}|C_k| = \frac{n}{\sqrt{r_C}} \\
\iff &\sum_{k=1}^{r_C}|C_k|^2 \geq \frac{n^2}{r_C} \\
\iff &\frac{1}{n^2}\sum_{k=1}^{r_C}|C_k|^2 \geq \frac{1}{r_C}
\end{aligned}$$

On remarque en outre qu'on a l'égalité si et seulement si $\mathbf{r}$ est colinéaire à $\mathbf{e}$, c'est-à-dire que les r_C blocs sont de même taille.

Ainsi, on a bien $\mathcal{Q}_r \leq 1 - \frac{1}{r_C}$. □

En dépit des limites bien connues de résolution de la modularité [58], cette mesure reste très employées dans le domaine de la détection de communautés, et elle est adéquate pour répondre à notre problématique. En outre, comme nous l'avons vu au chapitre précédent, dans le cas spécifique des graphes doublement stochastiques, plusieurs mesures incluant la modularité de Newman sont équivalentes.

Processus de fusion des blocs

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, décrit en Section 1.2.2 par l'Algorithme 1, est très utilisé en détection de communautés, domaine dans lequel il est connu sous le nom d'algorithme glouton – voir par exemple [63] –, mais il est généralement initié avec le partitionnement trivial dans lequel chaque classe contient un unique élément.

Cette méthode est peu coûteuse numériquement, et on a l'assurance de s'arrêter sur un maximum local, comme cela peut être démontré en adaptant la démonstration de [63]. Le processus de fusion empêche ainsi une explosion du nombre de blocs. Il est appliqué après chaque superposition de partitions.

Le processus de fusion est illustré sur `rbsb480` à la Figure 2.30. 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 2.29. La mesure de qualité est alors $\mathcal{Q}_r = 0.5574$ pour les lignes et $\mathcal{Q}_c = 0.4932$ 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 2.29. Ici, les mesures de qualité valent $\mathcal{Q}_r = 0.5574$ pour les lignes et $\mathcal{Q}_c = 0.502$ pour les colonnes, et sont largement supérieures à celles obtenues pour la partition initiale.

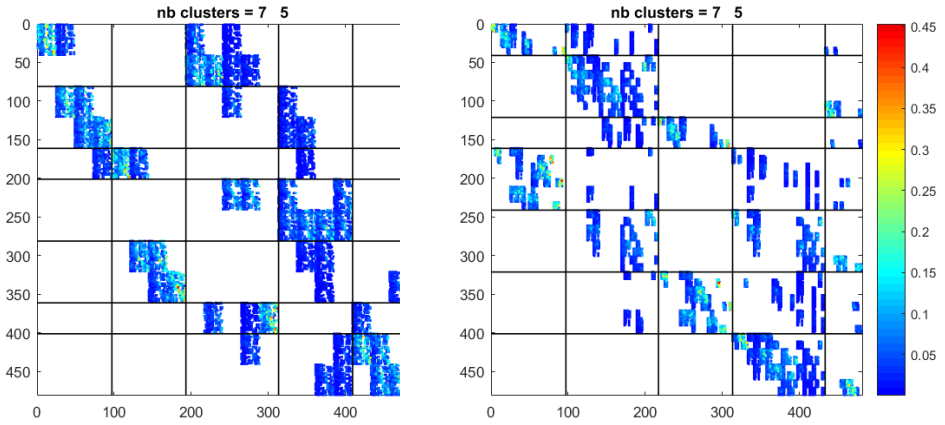


FIGURE 2.30 : *Fusion des blocs paire à paire.*

Par ailleurs, nous remarquons que le processus de fusion basé sur notre modularité

adaptée aux matrices doublement stochastiques est fortement impacté par les valeurs numériques des-dites matrices, comme le montre la Figure 2.31. Dans cette figure, on voit ce que donne le processus de fusion appliqué à deux matrices doublement stochastiques ayant la même structure creuse, mais des valeurs numériques différentes. Ces matrices sont construites avec de petits blocs diagonaux numériquement dominant chevauchant des blocs diagonaux plus larges. Les vecteurs propres dominants sont affichés, et ils font apparaître la même structure par blocs pour les deux matrices. Le processus de fusion choisit de recoller les petits blocs lorsque ceux-ci sont très dominants. *A contrario*, il "casse" ces petits blocs en les fusionnant avec les blocs plus larges lorsque les valeurs numériques de ces petits blocs ne sont pas suffisamment dominantes.

Critère d'arrêt de l'algorithme spectral

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

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

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.

2.6 Synthèse de l'algorithme spectral

Dans les sections précédentes, nous avons décrit et justifié les différentes étapes de notre algorithme spectral. La Figure 2.32 reprend le Schéma 2.1, en y apportant plus de détails. Nous rappelons brièvement ces différentes étapes en nous appuyant sur la Figure 2.32 :

1. La boîte *Equil. 2×stoch.* correspond à l'équilibrage bi-stochastique de la matrice bi-irréductible à coefficients non négatifs, passée en paramètre de l'algorithme. Cette étape est détaillée en Section 2.2.

2. La boîte *S.V.D dans orthogonal des blocs* correspond à la récupération de quelques vecteurs propres dominants des projections, dans l'espace orthogonal aux blocs déjà détectés, des équations normales de la matrice bi-stochastique. Nous avons décrit ce processus en Section 2.4.

3. La boîte *Détect. sauts via filtre* correspond à l'application du filtre de Canny sur un

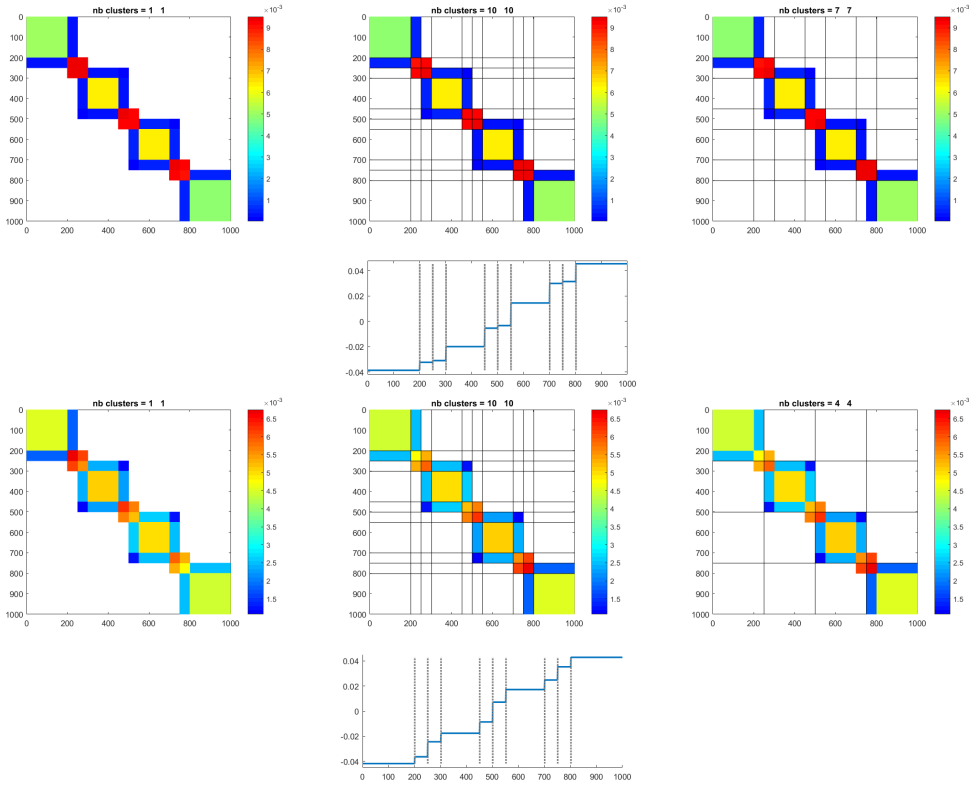


FIGURE 2.31 : *Figures de gauche : Les équilibrages doublement stochastiques de deux matrices. Ces matrices ont la même structure creuse. Avant équilibrage, les valeurs numériques des entrées dans les grands blocs sont en* $O(1)$ *pour les deux matrices ; et les valeurs numériques des petits blocs sont en* $O(10^3)$ *pour la matrice du haut, respectivement* $O(10)$ *pour celle du bas. Les figures du milieu montrent le découpage de la matrice après analyse de leur vecteur propre dominant dans l'orthogonal de* $\mathbf{e}$ *(le vecteur est affiché en dessous). Enfin, les figures de droite montrent le partitionnement obtenu après le processus de fusion des blocs.*

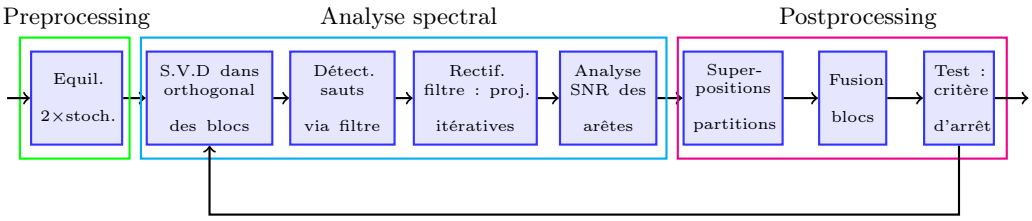


FIGURE 2.32 : *Fonctionnement global de notre algorithme spectral.*

des vecteurs dominants à notre disposition. Cette étape est décrite en introduction de la Section 2.3.

4. La boîte *Rectif. filtre : proj. itératives* correspond au processus de projections itératif que nous avons développé pour améliorer la détectabilité des arêtes retournées par le filtre de Canny. Ce processus est développé en Section 2.3.4.
5. La boîte *Analyse SNR des arêtes* correspond au filtrage des arêtes obtenues à la fin du processus des projections itératif. Ce filtrage se fait par seuillage de la mesure de SNR des arêtes. Cette mesure est développée à la fin de la Section 2.3.4.
6. La boîte *Superposition partitions* évoque la superposition des partitions par blocs de lignes ou de colonnes, trouvées aux différentes itérations, à l'aide des différents vecteurs singuliers, ainsi que la superposition de la partition par blocs de lignes et de celle par blocs de colonnes à la fin du processus. Cet aspect de l'algorithme est décrit en Section 2.5.1.
7. La boîte *Fusion blocs* correspond au processus de fusion basée sur la modularité de Newman, que nous avons décrit au début de la Section 2.5.2.
8. La boîte *Test : critère d'arrêt* correspond au critère de convergence global de notre méthode, basé sur une borne supérieure de la modularité. Nous avons détaillé cette étape à la fin de la Section 2.5.2. Si le critère d'arrêt n'est pas vérifié, on peut boucler sur la phase d'analyse spectrale. Quand on arrive à cette étape, on a une nouvelle proposition de partitionnement par blocs de lignes et de colonnes de la matrice. On va donc récupérer les vecteurs propres dominants des projections des équations normales dans l'orthogonal de ces nouveaux blocs. Evidemment, il est aussi possible de s'arrêter après un nombre fixé d'itération(s).

2.7 Application à la détection de formes

Afin de montrer les capacités de notre algorithme sur une tâche de classification non supervisée, nous l'avons testé sur les jeux de données de la librairie *scikit-learn* [90]. Il

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 [31] d'un ensemble de points $\{\mathbf{x}_i \in \mathbb{R}^p, i = 1...n\}$ est la matrice symétrique définie par

$$\mathbf{A}_{i,j} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), & \text{pour } i \neq j, \\ 0, & \text{sinon.} \end{cases} \tag{2.14}$$

La précision de notre méthode va fortement dépendre du choix du paramètre Gaussien σ. Nous le fixons en suivant l'heuristique de [91] 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-irréductibilité en fixant les éléments diagonaux à 10^{-8}, comme préconisé en Section 2.2.2. En guise de pré-traitement, nous supprimons les entrées dominantes de la matrice. C'est-à-dire que les entrées de l'équilibrage doublement stochastique supérieures à 0.55 seront considérées comme des blocs déjà définis dans l'étape de récupération des vecteurs dominants, présentée Section 2.4. En outre, le processus de fusion interdit de fusionner ces blocs avec d'autres blocs. Nous écartons ces éléments, car sinon, ils vont résulter en de tout petits blocs, qu'il va être difficile de détecter avec le filtre de Canny. Comme post-traitement, chacune de ces entrées est ensuite ajoutée à la classe la plus proche, c'est-à-dire celle 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 2.33. Les-dites méthodes sont :

— MiniBatch Kmeans : Le but de l'algorithme des K-moyennes, ou Kmeans, est de partitionner un ensemble de données en un nombre de classes passé en paramètre, en minimisant itérativement la distance entre chaque point de l'ensemble des données et le centroïde de la classe à laquelle il appartient, c'est-à-dire la moyenne des éléments constituant la classe. Le MiniBatch Kmeans est une version de Kmeans dans laquelle chaque itération est effectuée sur un sous-ensemble de l'ensemble des données – cf [92].

— Spectral Clustering : Cet algorithme sélectionne les vecteurs propres dominants d'une matrice d'affinité Gaussienne, comme définie équation (2.14), afin de construire un espace de représentation spectrale des données qui soit faible dimension – égale au

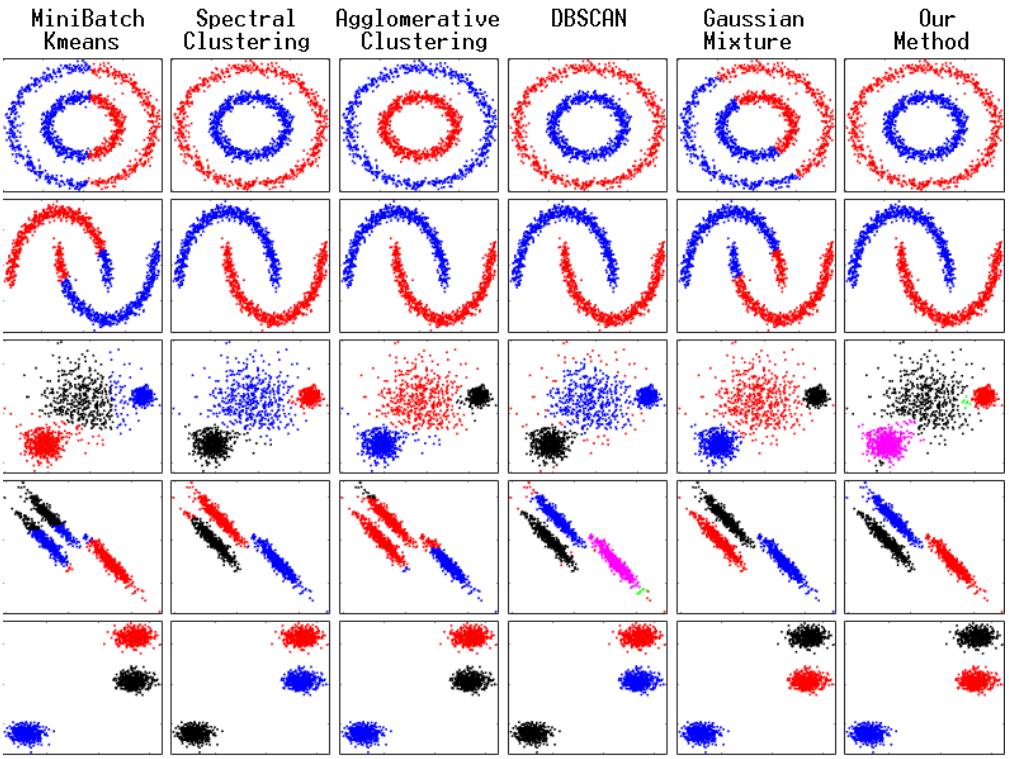


FIGURE 2.33 : *Comparaison d'algorithmes de partitionnement pour la détection de formes.*

nombre de classes voulues. Dans cet espace, le partitionnement est ensuite effectué via Kmeans [31].

— Agglomerative Clustering : Il s'agit d'un algorithme de hiérarchisation ascendant : on part d'une configuration dans laquelle chaque point est une classe, puis on fusionne les classes afin d'atteindre un nombre de classes fixé au préalable. A chaque itération, les deux classes fusionnées sont celles dont la variance de la classe résultante est la plus faible – approche de Ward [93]. Une contrainte est ajoutée concernant le fait que deux classes ne peuvent être fusionnées que si elles sont adjacentes.

— DBSCAN (Density-based spatial clustering of applications with noise) : Cet algorithme, présenté dans [94], sépare l'espace en zones où il y a une forte densité de points, séparées par des zones de faible densité de points. Les points appartenant à une même zone de forte densité appartiennent à la même classe. La notion de densité forte ou faible est paramétré par un réel ε : une zone de faible densité est un volume de l'espace dans lequel la densité de points est inférieure à ε.

— Gaussian Mixture : Les modèles "Gaussian Mixture" supposent que les données

TABLE 2.3 : *NMI des partitionnements Figure 2.33.*

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

ont été générées par un nombre fini – égal au nombre de classes recherchées – de lois Gaussiennes ayant des paramètres différents. Ces modèles trouvent une approximation des-dits paramètres. Une classe est alors l'ensemble des points ayant été générés par une même loi.

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 à la même classe. La colonne de droite montre les résultats de notre méthode, que nous avons arrêtée après l'analyse d'un unique vecteur propre. Etant donnés ces résultats, il est clair que ce vecteur fournit une information suffisante pour séparer grossièrement les nuages en classes cohérentes pour tous les jeux de données, quelles que soient leurs formes. Pour comparer formellement les partitionnements obtenus, nous avons calculé leur information mutuelle normalisée (NMI). Cette mesure, que nous avons définie Section 1.3, permet de mesurer l'écart entre deux partitionnements sans nécessiter qu'ils aient le même nombre de classes, en se basant sur leur matrice de confusion.

La NMI est donnée Table 2.3, en suivant le même ordre ligne/colonne que la Figure 2.33. 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.

2.8 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. Nous avons illustré ses performances pour détecter les formes dans un nuage

de points en deux dimensions, tâche pour laquelle notre algorithme s'est révélé avoir des résultats très corrects, le tout en n'utilisant qu'un vecteur propre. Nous remarquons néanmoins que cette application fait intervenir des matrices denses, qui ne souffrent donc pas des problèmes rencontrés lors de l'application des algorithmes spectraux aux graphes très creux – cf par exemple [95]. Le cas des graphes creux sera traité en Section 3.1.

Il serait intéressant de généraliser notre méthode pour effectuer des tâches de co-clustering. En effet, notre méthode est capable de détecter des structures rectangulaires dans les matrices carrées, comme le montre la Figure 2.34. Ici, deux matrices présentant une structure par blocs rectangulaires ont été générées avec la fonction `make_biclusters` de scikit-learn [90]. 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

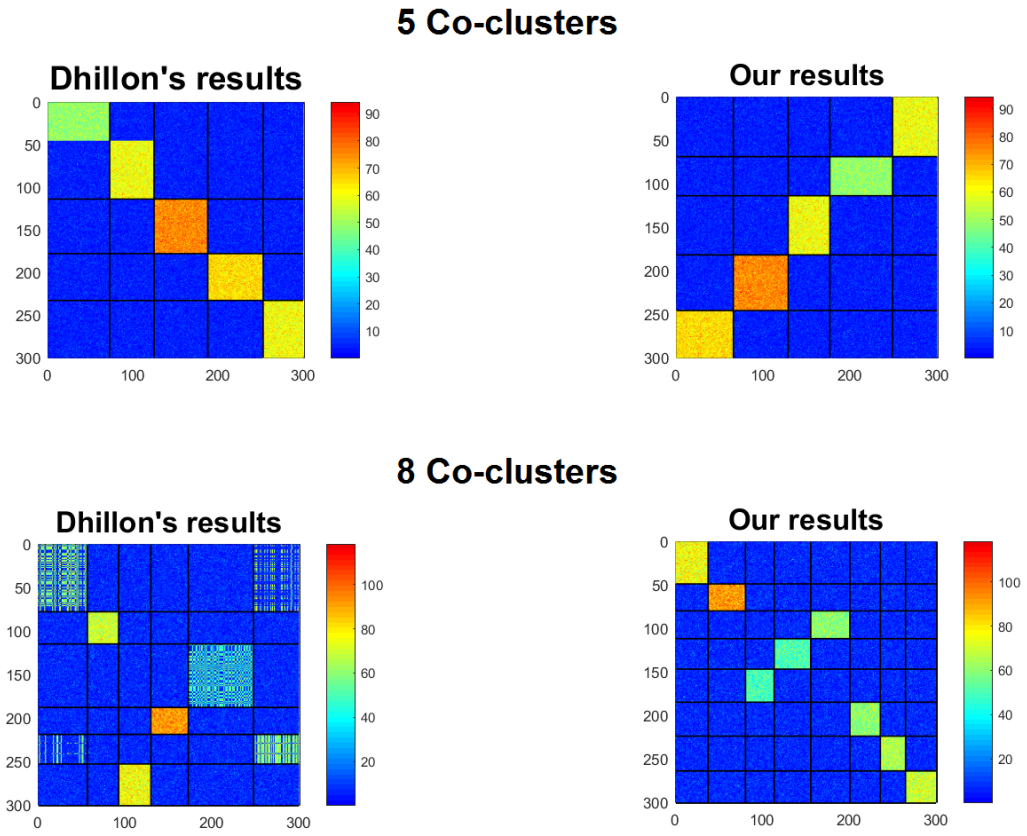


FIGURE 2.34 : *Détection de co-clusters.*

Dhillon [33] – 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 2.34 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 ([80, 96]) que l'imbrication entre les classes de lignes et de colonnes est un paramètre important de la qualité d'un co-clustering.

Des algorithmes de co-clustering basés sur l'équilibrage doublement stochastique existent [12]. Cependant, la détection des co-clusters se fait uniquement via les facteurs de scaling, ce qui impacte leur performance, car ces facteurs véhiculent souvent d'autres types d'information sur les noeuds – par exemple le fait que les noeuds sont des hubs ou des noeuds autoritaires au sens de l'algorithme HITS [97], comme montré dans [26].

Chapitre 3

Applications

Dans ce chapitre, nous explorons trois applications à la détection de structures par blocs dans les matrices. Dans la première partie, nous nous intéressons à l'utilisation de l'algorithme présenté dans la partie précédente afin de faire de la détection de communautés sur des graphes simples – non orientés et non pondérés –, générés artificiellement. Dans la deuxième partie, nous explorons à nouveau les performances de notre méthode, cette fois en tant que préconditionneur de type Jacobi par blocs. Enfin, en dernière partie, nous utilisons d'autres types d'algorithmes de classification non supervisée afin de détecter les actes de dialogues dans un chat.

3.1 Application à la détection de communautés

Dans cette partie, nous illustrons les performances de notre algorithme sur la détection de communautés, en le comparant à des algorithmes existants sur des réseaux générés artificiellement.

3.1.1 Prétraitement des données

L'objet sur lequel nous appliquons notre algorithme est un réseau, *id est* un graphe non orienté, non pondéré, et sans "self-loop". Les matrices d'adjacences de tels graphes seront des matrices symétriques, dont les entrées sont des 0 et des 1, et avec une diagonale vide.

Comme nous l'avons vu, notre algorithme fonctionne sur les matrices d'adjacence de graphes connexes, à condition que leur diagonale soit positive. On a aussi vu qu'ajouter une diagonale à la matrice d'adjacence ne modifie pas sa structure diagonale par blocs.

La première étape est donc d'ajouter quelque chose sur la diagonale, pour s'assurer que la matrice d'adjacence est à support total. On propose d'ajouter $10^{-8}\mathbf{I}$ à la matrice d'adjacence, comme préconisé en Section 2.2.2.

Ensuite, on applique l'équilibrage doublement stochastique, puis on met de côté les éléments très dominants (> 0.55). En effet, ces éléments vont résulter en de tout petits blocs qu'il va être difficile de récupérer avec l'outil présenté Section 2.3, alors qu'il est relativement simple algorithmiquement de récupérer ces éléments après l'étape d'équilibrage. Avec l'analyse de la matrice équilibrée, on propose un premier partitionnement tel que :

— Un élément très fort sur la diagonale va aboutir en une classe singleton.

— Un élément très fort hors diagonale (en (i, j)) va aboutir en une classe contenant la paire $\{i, j\}$, puisqu'alors il s'agit d'un lien privilégié entre ces deux noeuds.

— Les éléments non dominants vont être mis dans la même classe, qu'on va découper ensuite par l'analyse spectrale.

Dès la première étape d'analyse spectrale, on va s'intéresser à la projection orthogonale de la matrice doublement stochastique, comme cela est expliqué Section 2.4, dans le complémentaire des blocs déjà définis – classes singleton et paires.

3.1.2 Post-traitement des données

Les mini-blocs identifiés pendant la phase de pré-traitement sont extraits pour des raisons numériques (ils auraient résulté en des vecteurs propres de type dirac ou quasi-dirac, difficiles à détecter avec le filtre). Ceci étant, ces mini blocs pourraient aussi être fusionnés, soit entre eux, soit au sein de plus gros blocs identifiés par l'analyse spectrale. On applique donc une fusion partielle après l'analyse spectrale, en utilisant la modularité de Newman et Girvan non numérique, c'est-à-dire sur la matrice d'adjacence binaire. Nous disons que cette fusion est partielle parce que l'on ne souhaite pas appliquer un algorithme glouton sur l'ensemble des blocs : le but de ce processus est de fusionner les mini-blocs, soit entre eux, soit avec les blocs découverts par l'analyse spectrale. On force donc le processus à ne pas fusionner entre eux des blocs découverts avec l'analyse spectrale.

3.1.3 Comparaison entre deux structure en communautés

Pour évaluer la qualité de notre algorithme, nous l'avons testé sur deux bancs d'essais : le banc d'essai de Newman et Girvan, cf Section 3.1.4, et le banc d'essai de Landicichinette-Fortunato-Radicchi, cf Section 3.1.4. Ces bancs d'essais sont pratiques car ils génèrent des graphes aléatoires dont on connaît la structure en communautés. Il s'agit donc de voir à quel point la structure en communautés découverte par un algorithme est proche de la

structure en communautés effective du réseau. Pour ce faire, nous utilisons à nouveau l'information mutuelle normalisée (NMI) et le ratio entre les nombres de communautés (RC), comme nous l'avons fait Section 1.3. Nous rappelons que la NMI permet de mesurer l'écart entre deux partitionnements sans nécessiter qu'ils aient le même nombre de classes, en se basant sur leur matrice de confusion. Le RC correspond simplement au ratio entre le nombre de communautés retourné par l'algorithme – noté $\overline{C}$ – et le nombre de communautés attendu – noté C.

3.1.4 Bancs d'essais

Comme cela a été dit à la section précédnte, pour comparer notre algorithme aux autres algorithmes de détection de communautés, nous avons utilisé des programmes qui génèrent aléatoirement des graphes artificiels ayant une structure en communautés connue. Cela permet d'évaluer la qualité d'un découpage suivant des critères plus objectifs que la modularité, et avec un panel de réseaux quasi-illimité. Ces bancs d'essais, introduits Section 1.3, sont rappelés ici.

Bancs d'essais de Newman et Girvan (NGBenchmark)

Le banc d'essais de Newman et Girvan, développé dans [59], génère un graphe ayant 128 noeuds, et une structure à 4 communautés, chacune possédant 32 noeuds. Ce banc d'essai a longtemps fait référence dans le domaine de la détection de communautés. Cependant, les contraintes qu'il impose au réseau généré ne lui permette pas d'avoir des caractéristiques réalistes – tailles des communautés, distribution des degrés, ... Il ne doit donc servir que comme une première évaluation, pour savoir si un algorithme va dans la bonne direction, ou s'il est inapte à la détection de communautés.

Bancs d'essais de Landicichinetti-Fortunato-Radicchi (LFRBenchmark)

Le banc d'essais LFRBenchmark développé dans [65] génère des graphes qui ont une structure en communautés semblable à celles que l'on peut trouver dans des réseaux provenant du monde réel (sociologie, biologie, web, etc.). Les communautés ont des tailles qui suivent une loi puissance, de paramètre fixé par l'utilisateur. De même pour le degré des noeuds. Il est aussi possible de fixer la taille minimale et maximale des communautés, le degré moyen du réseau, le degré maximal des noeuds du réseau, etc.

Pour ces deux bancs d'essai, il faut fixer ce que l'on appelle le paramètre de mixage μ lors de la génération d'un réseau. Ce paramètre a été détaillé Section 1.3. Nous rappelons qu'il permet de définir la force de la structure en communautés du réseau. μ est à valeur dans $[0, 1]$. Concrètement, plus μ est grand, plus la structure en communautés du graphe est mal définie. Dans un réseau pour lequel $\mu = 0$, les communautés sont des composantes disjointes. Nous avons généré des réseaux pour les valeurs de μ dans $\{k \times 0.03, k = 1..25\}$

3.1.5 Les Algorithmes

Nous confrontons notre méthode à 5 autres algorithmes. Ces autres algorithmes sont disponibles dans la librairie *igraph* [98], disponibles en python, R et C. Cette bibliothèque numérique a une bonne visibilité, et est largement utilisée par les acteurs de la détection de communautés, d'où ce choix. Dans la suite, nous détaillons succinctement les algorithmes auxquels nous nous sommes confrontés.

Leading Eigenvector (LE)

Il s'agit d'un algorithme proposé par Newman dans [32], qui cherche à optimiser la mesure de modularité d'un graphe via le vecteur propre dominant d'une matrice qui dépend à la fois du graphe et de la modularité.

Dans les faits, pour un réseau $G = (V, E)$ donné, $|V| = n$, $|E| = m$, on crée une matrice $\mathbf{B}$, telle que :

$$\mathbf{B} = \mathbf{A} - \mathbf{P}$$

avec $\mathbf{A}$ la matrice d'adjacence du graphe, et $\mathbf{P}$ telle que :

$$\mathbf{P}_{i,j} = \frac{d_i d_j}{2m},$$

d_i étant le degré du noeud i. Cette matrice $\mathbf{P}$ est la matrice de probabilité qu'une arête existe entre i et j dans le graphe s'il ne possède aucune structure en communautés.

L'intérêt de la matrice $\mathbf{B}$ est qu'elle permet d'exprimer sous forme matricielle la modularité de ce graphe dans le cas d'une bipartition : si l'on note $\mathbf{s}$ le vecteur caractérisant une bipartition, c'est-à-dire un partitionnement à deux classes – appelons-les C_1 et C_2é–, en ayant $\mathbf{s}$ sous la forme :

$$\mathbf{s}(i) = \begin{cases} 1 & \text{si } i \in C_1 \\ -1 & \text{si } i \in C_2 \end{cases}$$

alors la mesure de modularité de cette bipartition s'écrit simplement en fonction de $\mathbf{B}$:

$$\mathcal{Q} = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}.$$

Le but de Newman est donc de résoudre

$$\begin{cases} \underset{\mathbf{s}\in\mathbb{R}^n}{argmax}\frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s} \\ \text{s.t. } \forall i \in \{1,...,n\}, \mathbf{s}(i) \in \{-1,1\} \end{cases}$$

Si l'on enlève la contrainte sur les coordonnées de $\mathbf{s}$, le vecteur qui maximise cette équation est le vecteur propre associé à la plus grande valeur propre de $\mathbf{B}$. Et on obtient le vecteur ne possédant que des 1 et des -1 qui approche au plus ce vecteur propre en appliquant la fonction "signe" au-dit vecteur propre. L'algorithme qui en découle est donné par l'Algorithme 4. La définition de la fonction `bipart` est précisée dans l'Algorithme 5.

Algorithm 4: : *Leading Eigenvector Algorithm*

Data: $\mathbf{A} \in \mathbb{R}^{n\times n}$ une matrice d'adjacence.
Result: $\mathcal{C}$ une partition de $\{1...n\}$.
begin
  Calculer $\mathbf{B}$;
  return $\mathcal{C}$ = `bipart`($\mathbf{B}$);
end

Les matrices $\mathbf{B}^{C_1}$ et $\mathbf{B}^{C_2}$ de l'Algorithme 5 sont définies de la manière suivante : Soit $C \subset V$ une classe de cardinal n_C, alors la matrice $\mathbf{B}^C \in \mathbb{R}^{n_C\times n_C}$ est

$$\forall i,j \in C, \mathbf{B}^C_{i,j} = \mathbf{B}_{i,j} - \delta(i,j)\sum_{t\in C} B_{i,t}$$

avec $\delta(i,j)$ le symbole de Kröenecker (=1 si $i = j$, 0 sinon).
Cette matrice permet de calculer le gain de modularité obtenu pour une bipartition de C dans le graphe total.
Pour résumer en quelques mots, il s'agit d'un algorithme récursif qui cherche à optimiser la modularité d'une bipartition via une approche spectrale.

Louvain (LV)

Cet algorithme, que nous avons déjà rencontré Section 1.3, est expliqué en détails dans l'article [29]. Pour rappel, étant donné un graphe $G = (V, E)$ l'algorithme de Louvain

Algorithm 5: : *Fonction récursive* `bipart`

Data: $\mathbf{B} \in \mathbb{R}^{n \times n}$ une matrice symétrique.
Result: $\mathcal{C}$ une partition de $\{1...n\}$.
begin
  $\mathbf{v} \leftarrow$ vecteur propre dominant de $\mathbf{B}$;
  $C_1 \leftarrow \{i : \mathbf{v}(i) \geq 0\}$;
  $C_2 \leftarrow \{i : \mathbf{v}(i) < 0\}$;
  $\mathbf{s} \leftarrow \mathbf{e} \in \mathbb{R}^n$;
  $\mathbf{s}(C_2) \leftarrow -1$;
  $\Delta \leftarrow \mathbf{s}^T\mathbf{B}\mathbf{s}$;
  if $\Delta \leq 0$ **then**
    return $\mathcal{C} = \{C_1 \cup C_2\}$;
  else
    return $\mathcal{C} = \{\texttt{bipart}(\mathbf{B}^{C_1}), \texttt{bipart}(\mathbf{B}^{C_2})\}$;
  end
end

démarre avec comme point de départ la structure triviale ayant une communauté par noeud. Ensuite, l'algorithme fonctionne comme suit :

— Pour chaque noeud, on trouve la communauté telle que l'ajout de ce noeud dans cette communauté va augmenter le plus la mesure de modularité. On s'arrête lorsque la situation est stable, c'est-à-dire que pour chaque noeud, la communauté à laquelle il appartient est la meilleure possible en terme de modularité.

— On crée alors un méta-graphe : dans ce nouveau graphe, chaque communauté du graphe initial devient un méta-noeud, les arêtes à l'intérieur d'une même communauté deviennent des "self-loop" – *id est* des arêtes ayant le même sommet à ses deux extrémités – de poids égal à la somme des poids des arêtes internes à la communautés, et les arêtes inter-communauté sont sommées – c'est-à-dire que s'il existe deux arêtes de poids 1 entre la communauté C_1 et la communauté C_2, on aura dans le nouveau graphe une arête de poids 2 entre les noeuds C_1 et C_2.

— On réapplique le processus précédent à ce méta-graphe, etc.

Le processus s'arrête naturellement lorsque le méta-graphe passé en entrée du processus est tel que la meilleure configuration est la structure en communautés ayant une communauté par méta-noeud.

Fast&Greedy (FG)

Il s'agit de l'Algorithme 1 introduit Section 1.2.2. Pour rappel, il s'agit d'un processus qui fusionne les communautés en cherchant à maximiser la mesure de modularité de Newman et Girvan. Cet algorithme part du point de départ où chaque élément est une communauté. Il est dit "fast" car il s'agit d'une version optimisée (les détails concernant ces optimisations sont développés dans [63]) permettant un calcul très efficace de l'algorithme initial "greedy" – glouton.

Edge Betweenness (EB)

Il s'agit d'un algorithme de hiérarchisation ascendant développé dans [42], basé sur une mesure de centralité des arêtes. Cette centralité est calculée comme suit :
Pour tous les couples de sommets dans le graphe, on trouve le(s) plus court(s) chemin(s) de l'un à l'autre. Si k est le nombre de plus courts chemins entre les deux sommets, on attribue le poids de $\frac{1}{k}$ à toutes les arêtes d'un même chemin. Ainsi, chaque arête possède un poids par plus court chemin. La mesure de centralité d'une arête est alors la somme de ces poids. Un exemple est donné à la Figure 3.1.
L'algorithme du Edge Betweenness permet de construire un dendrogramme de la façon suivante :

— On crée le graphe réduit correspondant au graphe de base dans lequel on a supprimé l'arête ayant la plus forte centralité.

— On calcule la centralité des arêtes dans ce graphe réduit.

— On répète jusqu'à ce qu'il n'y ai plus d'arêtes

On construit le dendrogramme en partant de la situation où chaque noeud est une communauté, puis en mettant d'abord en relation les deux sommets liés par la dernière arête supprimée, puis les deux sommets ou groupes de sommets liés par l'avant dernière arête supprimée, etc. et ce jusqu'à mettre en relation les groupes de sommets liés par la première arête supprimée.
Pour choisir le niveau du dendrogramme qui va définir la structure en communautés, on calcule la modularité de Newman des structures correspondant à chaque niveau, et on renvoie celle qui a la plus forte modularité.

Walktrap (WT)

L'idée directrice de cet algorithme, présenté dans [2], est d'utiliser les marches aléatoires pour détecter les communautés. Si au cours d'une marche aléatoire on se retrouve sur

l'un ou l'autre des sommets d'une même communauté, cela ne doit pas fondamentalement changer la suite de la marche, car des sommets d'une même communauté doivent avoir à peu près la même "vision" du reste du graphe. Cette "vision" est explicitée par une matrice de transition. Plus concrètement, en notant $\mathbf{A}$ la matrice d'adjacence, $\mathbf{D}$ la matrice des degrés, et $\mathbf{P}$ la matrice de transition du graphe telle que $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, les auteurs s'intéressent à la matrice de transition après un temps t suffisamment long pour contenir suffisamment d'information sur le graphe, mais suffisamment court pour que $\mathbf{P}^t$ ne soit pas impactée par le fait qu'à l'infini, la probabilité de se trouver sur un sommet i ne dépend que du degré de i. Les auteurs choisissent par défaut $t = 4$.

A partir de cette matrice $\mathbf{P}^t$, les auteurs définissent une distance entre les sommets. Ils traduisent le postulat stipulant que "si deux sommets appartiennent à la même communauté, ils ont la même vision du reste du graphe" par le fait que les deux lignes correspondant à ces sommets dans $\mathbf{P}^t$ doivent être proches. Ils définissent donc la distance

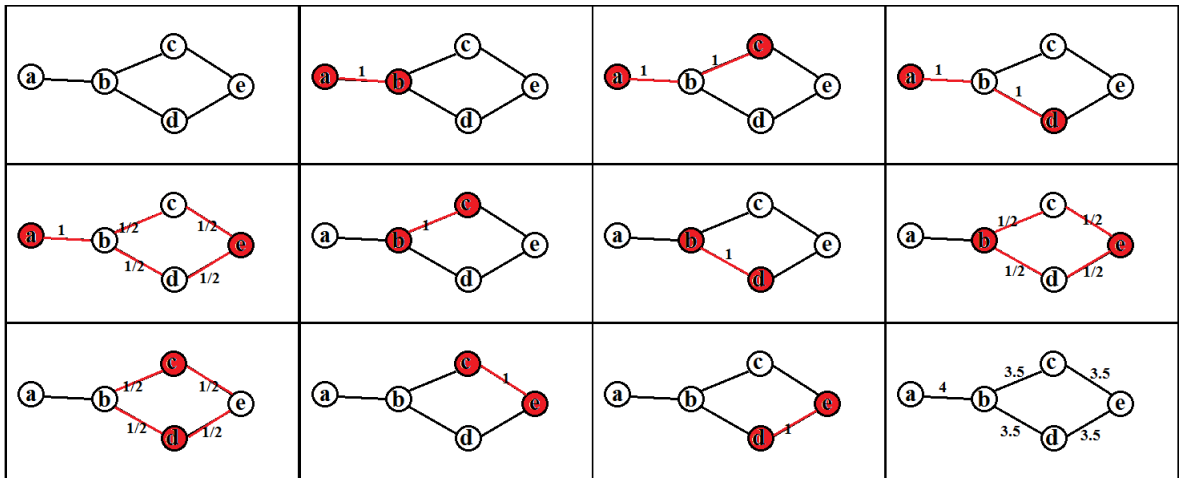


FIGURE 3.1 : *Calcul des mesures de centralité des 5 arêtes du graphe. Dans les dix figures ayant deux sommets en rouge, on affiche le poids des arêtes lié au(x) plus court(s) chemin(s) existant entre les deux sommets en rouge. par exemple, dans la figure colonne 1 ligne 2, on calcule les poids liés aux plus courts chemins entre a et e. Il existe deux plus courts chemins entre ces sommets, on va donc attribuer un poids de 1/2 aux arêtes dans chacun de ces chemins. L'arête (a,b) apparaissant dans les deux chemins, elle a un poids de* $2 \times 1/2 = 1$. *Dans la figure colonne 4 ligne 3, on affiche les centralité des arêtes, obtenues en sommant tous les poids.*

entre deux sommets i et j de la manière suivante :

$$r(i,j) = \sqrt{\sum_{k=1}^{n} \frac{(\mathbf{P}_{i,k}^t - \mathbf{P}_{j,k}^t)^2}{\mathbf{D}(k,k)}}.$$

Cette distance est, selon eux, d'autant plus faible que les sommets appartiennent à la même communauté. Ils étendent ensuite cette distance aux groupes de sommets de manière directe, en définissant pour un groupe de sommets C la valeur $\mathbf{P}_{C,j}^t = \frac{1}{|C|} \sum_{i \in C} \mathbf{P}_{i,j}^t$. Cette distance leur permet de définir une mesure de qualité d'un partitionnement $\mathcal{C} = \{C_1, ..., C_p\}$ d'un ensemble $V = \{1, ..., n\}$:

$$\sigma = \frac{1}{n} \sum_{C \in \mathcal{C}} \sum_{i \in C} r(i, C)^2$$

Une fois tous ces objets définis, il ne reste plus qu'à appliquer l'algorithme à proprement parler. Il s'agit d'un algorithme de hiérarchisation ascendante. On part du partionnement possédant une classe par noeud, puis on fusionne itérativement les classes dont la fusion diminue le plus la mesure σ, jusqu'à ce que tous les éléments soient dans la même classe. Cette technique va aboutir en un dendrogramme, et comme pour l'algorithme du Edge Betweenness, on va retourner le partitionnement correspondant au niveau du dendrogramme qui maximise la modularité de Newman.

3.1.6 Tests Numériques

Dans cette section, nous comparons notre méthode (US) développée au Chapitre 2 avec les algorithmes expliqués ci-dessus, sur NGBenchmark et LFRBenchmark. Afin d'affiner nos observations, nous menons ensuite une étude plus poussée en fonction du degré moyen des réseaux en deuxième partie de cette section. Pour évaluer la précision des méthodes, nous utilisons la NMI et le RC que nous avons rappelés Section 3.1.3.

Premiers tests

Nous avons confronté notre méthode à tous les algorithmes présentés Section 3.1.5, d'abord sur NGBenchmark. Pour chaque algorithme, chaque point sur la courbe est la valeur moyenne de 100 réseaux générés pour un même μ (moyenne des NMI, et moyenne des RC). Les performances comparées des algorithmes sont présentées Figure 3.2. Comme le ratio du nombre de communautés retournées par EB sur le nombre de communautés attendu est trop grand par rapport aux autres algorithmes, l'échelle d'affichage n'est pas

la même (se référer à l'axe des ordonnées de droite pour EB et à l'axe des ordonnées de gauche pour les autres algorithmes). Nous remarquons que lorsque $\mu < 0.5$, les algorithmes ont une précision à peu près équivalente. On remarque que les résultats de WT sont sensiblement meilleurs pour $\mu \in [0.42, 0.54]$, et que EB sur-découpe de façon importante les communautés à partir de $\mu > 0.39$ – il y a pratiquement un facteur 10 entre son ratio et celui des autres algorithmes.

Nous comparons ensuite les performances des différents algorithmes sur LFRBenchmark, paramétré comme précisé Table 3.1. Dans cette table, 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 des communautés. Les courbes de performances des différents algorithmes sont affichées Figure 3.3. A nouveau, chaque point sur la courbe est la valeur moyenne de 100 réseaux générés pour un même μ. Le ratio des nombres de communautés de EB étant toujours très supérieur à celui des autres algorithmes, nous utilisons la même astuce d'affichage que précédemment. Nous constatons que les algorithmes se divisent en deux familles : WT et LV, dont les performances restent bonnes, même pour de grandes valeurs de μ, et LE, EB et FG, dont les performances décroissent rapidement. Notre

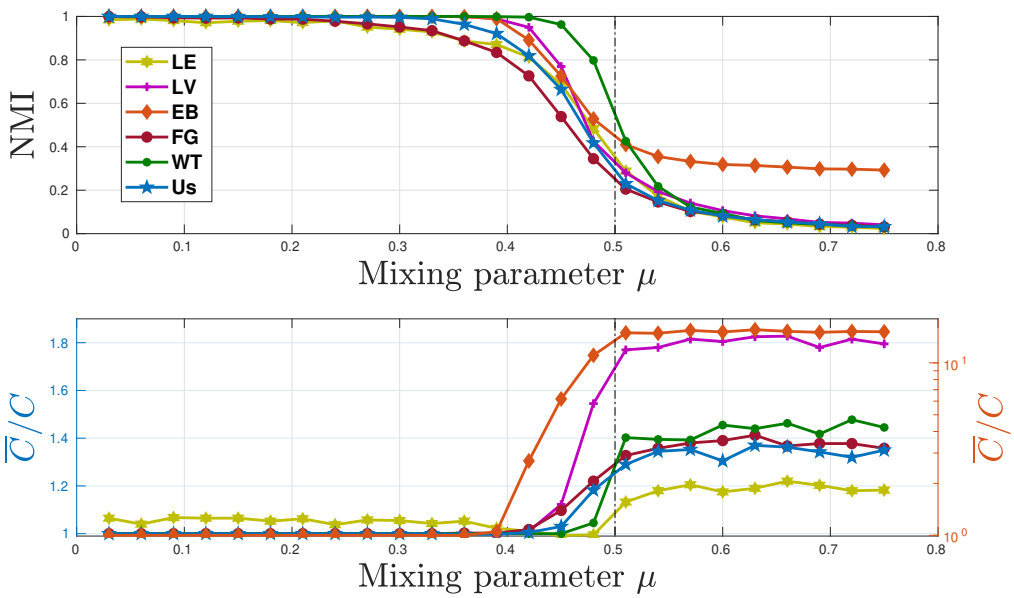


FIGURE 3.2 : *Résultats des algorithmes sur NGBenchmark. Dans la figure du bas, les résultats de EB sont à observer sur l'échelle des ordonnées à droite.*

TABLE 3.1 : *Paramètres pour générer les réseaux.*

Paramètre	Valeur
nombre de noeuds	482
degré moyen	20
degré maximum	48
exp. db des degrés	-2
exp. db des tailles de communautés	-1

algorithme se situe entre ces deux familles : il n'est pas compétitif avec les meilleurs algorithmes sur ce banc d'essai, mais cependant sensiblement meilleur que des algorithmes encore largement utilisées dans ce domaine, et qui utilisent le même type de processus (maximisation gloutonne de la modularité dans FG, utilisation d'éléments spectraux dans LE).

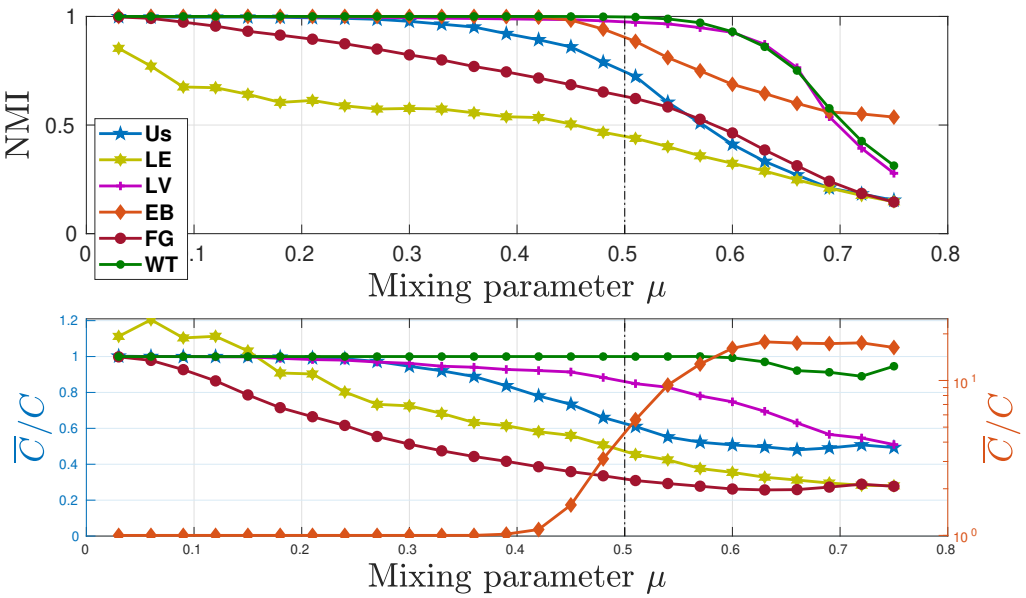


FIGURE 3.3 : *Résultats des algorithmes sur LFRBenchmark. Dans la figure du bas, les résultats de EB sont à observer sur l'échelle des ordonnées à droite.*

TABLE 3.2 : *Paramètres pour générer les réseaux.*

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

En faisant varier le degré moyen

Afin d'analyser plus précisément le comportement de notre méthode pour des tâches de détection de communautés, nous avons observé l'impact du degré moyen du réseau sur différents algorithmes : LV, WT, LE, et FG.

Nous avons appliqué ces algorithmes sur quatre ensembles de réseaux de 500 noeuds, générés avec LFRBenchmark. Chaque ensemble de réseaux correspond à un degré moyen différent $\overline{d}$. Les détails sur la génération des graphes sont donnés Table 3.2.

Les résultats sont donnés Figure 3.4. Chaque schéma correspond à la valeur de $\overline{d}$ précisée au-dessus. Pour l'obtention de ces courbes, nous avons généré 50 réseaux pour chaque valeur de μ puis calculé la NMI moyenne. A nouveau, nous pouvons diviser les algorithmes existants en deux groupes : LV 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 $\overline{d}$ augmente. Ceci peut être expliqué par les contraintes imposées lors de la génération des réseaux qui vont fournir des communautés plus grosses quand $\overline{d}$ est grand, comme montré Figure 3.5. 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, $\overline{d} = 10$ avec 40 communautés ; dans celui de droite $\overline{d} = 75$ 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 positivement impactée par l'accroissement de $\overline{d}$ que celles de FG et LE.

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 [7]. Nous avons donc comparé notre algorithme initial (USD) à notre algorithme appliqué sur le réseau perturbé (USW), c'est-à-dire sur $\mathbf{A} + \varepsilon \mathbf{e}\mathbf{e}^T$ avec

A la matrice d'adjacence du réseau. Nous avons fixé ε à 0.15 comme cela est suggéré pour l'algorithme Pagerank [99]. Ces deux versions sont comparées Figure 3.6 pour deux valeurs de $\overline{d}$ différentes. On observe que les deux versions ont un comportement similaire quand $\overline{d} = 75$, tandis que USW (courbe bleue) est bien plus précis que USD (courbe rose) pour $\overline{d} = 20$.

Pour compléter cette étude, nous avons observé le comportement de WT, LV, USD et USW pour des valeurs croissantes de $\overline{d}$, comme montré Figure 3.7. LV, WT et USW ont le même comportement lorsque l'on fait varier le degré : ils commencent par accroître leur précision, semblent atteindre un seuil, puis voient leur performance décroître à mesure que $\overline{d}$ augmente. USD augmente d'abord aussi sa précision, mais ses performances ne décroissent pas pour la valeur maximale de $\overline{d}$.

3.1.7 Conclusion

Ainsi, bien que sur des réseaux très creux, notre algorithme ne soit pas nécessairement meilleur que tous les algorithmes de détection communautés existants, il est capable de mieux détecter les communautés que des algorithmes conçus à cet effet et largement utilisés à ce jour. 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 rappelons en outre que sur la détection de formes présentée Section 2.7, il obtient

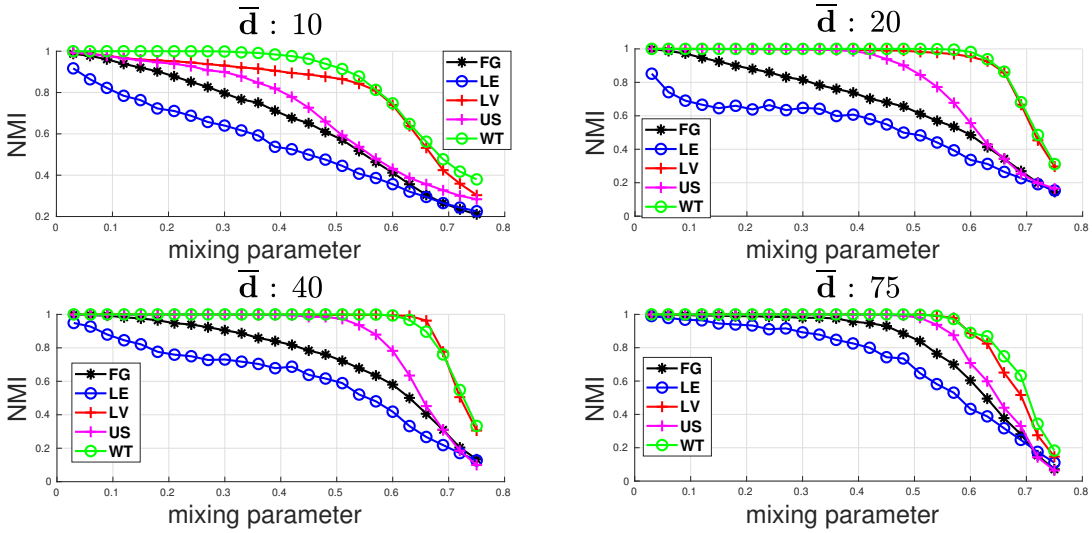


FIGURE 3.4 : *Courbes de NMI pour FG, LE, LV, WT, US.*

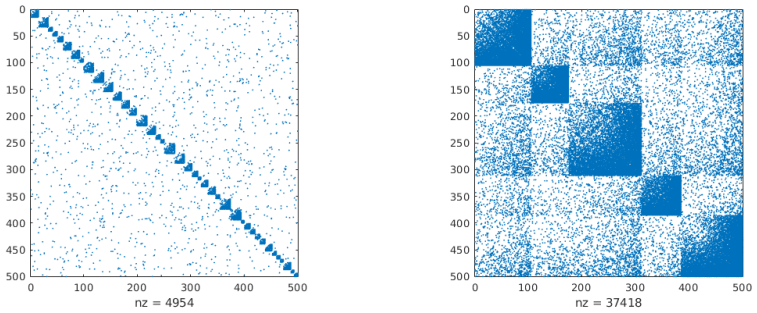


FIGURE 3.5 : *Matrices d'adjacence de réseaux.*

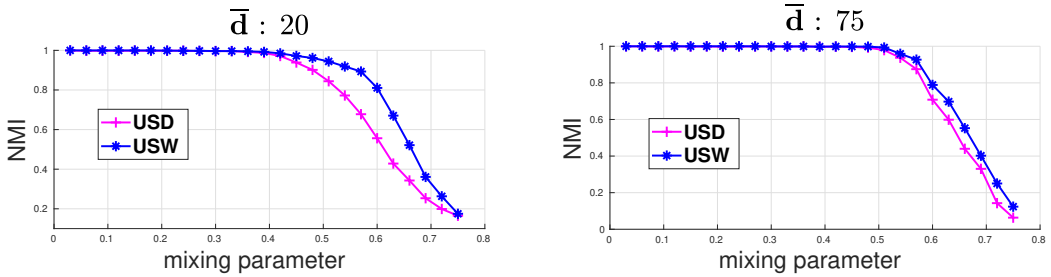


FIGURE 3.6 : *Courbes de NMI pour USD et USW.*

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*id est. un déséquilibre entre les arêtes entrantes et sortantes d'une communauté –, alors que WT et LV, tous deux contraints de travailler sur $\mathbf{A} + \mathbf{A}^T$, renvoient des résultats erronés dans ce cas. Un exemple est montré Figure 3.8, où deux communautés sont fortement connectées de façon dissymétrique. Les résultats de LV, WT et USD sont fournis, les lignes noires représentent les séparations entre les communautés. Seul notre algorithme renvoie ici un résultat cohérent.

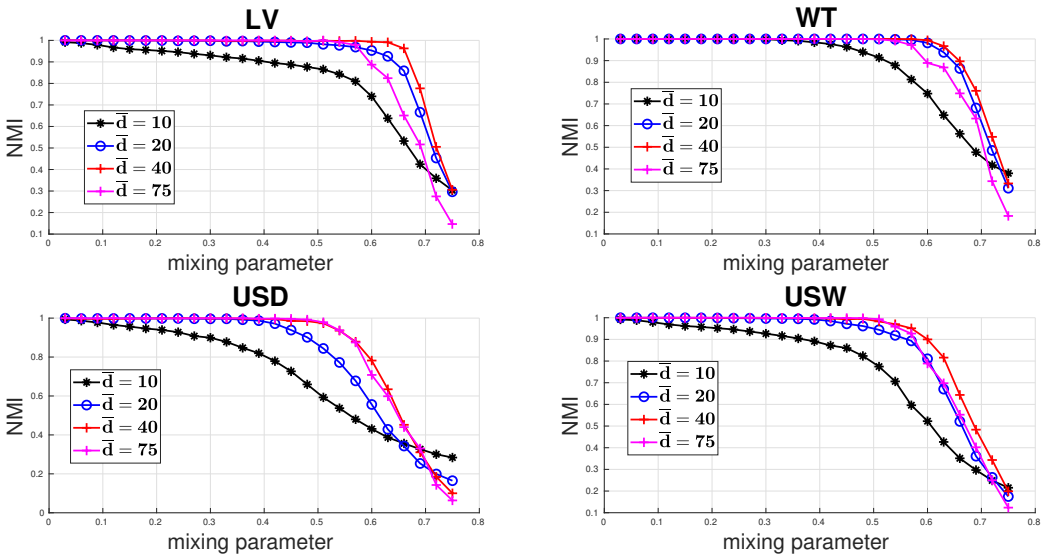


FIGURE 3.7 : *Courbes de NMI de ML, WT, USd, USw suivant des valeurs de* $\overline{d}$.

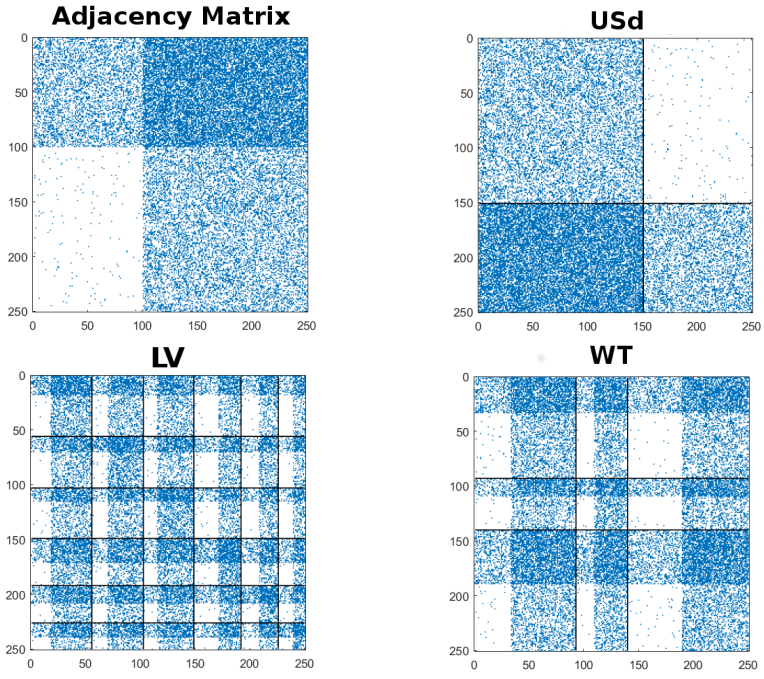


FIGURE 3.8 : *Détection de communautés dissymétriques.*

3.2 Préconditionnement de systèmes linéaires

Identifier une structure par blocs dans une matrice $\mathbf{A}$ peut offrir des opportunités pour résoudre efficacement un système linéaire avec cette matrice, c'est-à-dire pour

$$\text{Trouver } \mathbf{x} \in \mathbb{R}^n \text{ tel que : } \mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.1}$$

par exemple via la construction d'un préconditionneur adapté.

Préconditionner le système linéaire de l'équation (3.1) signifie que l'on va plutôt chercher à résoudre un problème connexe, par exemple :

$$\text{Trouver } \mathbf{x} \in \mathbb{R}^n \text{ tel que : } \mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{3.2}$$

où le système linéaire de l'équation (3.2) est numériquement plus facile à résoudre que celui de l'équation (3.1) – par exemple, le nombre de conditionnement de $\mathbf{M}^{-1}\mathbf{A}$ est plus faible que celui de $\mathbf{A}$ ou ses valeurs propres sont mieux réparties. La matrice $\mathbf{M}$ est appelée préconditionneur.

Le type de préconditionneur montré à l'équation (3.2) est appelé "préconditionneur à gauche". Il existe aussi des préconditionneurs à droite :

$$\text{Trouver } \mathbf{x} \in \mathbb{R}^n \text{ tel que : } \begin{cases} \mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b} \\ \mathbf{x} = \mathbf{M}^{-1}\mathbf{y} \end{cases} \tag{3.3}$$

et des préconditionneurs mixtes :

$$\text{Trouver } \mathbf{x} \in \mathbb{R}^n \text{ tel que : } \begin{cases} \mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\mathbf{y} = \mathbf{M}_L^{-1}\mathbf{b} \\ \mathbf{x} = \mathbf{M}_R^{-1}\mathbf{y} \end{cases} \tag{3.4}$$

Notre algorithme présenté au Chapitre 2 identifie des blocs dans des matrices, ce qui peut servir à construire des préconditionneurs par blocs, par exemple des préconditionneurs de type "bloc Jacobi amélioré" au sens introduit par Zhu et Sameh dans [16]. Un préconditionneur de type bloc Jacobi consiste à choisir $\mathbf{M}$ dans l'équation (3.2) ou (3.3) tel que

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_p \end{bmatrix}$$

où les $\mathbf{A}_k$ sont des blocs diagonaux de $\mathbf{A}$.

Dans [16], les auteurs montrent que l'efficacité d'un préconditionneur de type bloc Jacobi est fortement améliorée si les blocs diagonaux sont dominants, et que les connexions entre les blocs sont moindres. En outre, dans [21], il est démontré que pour les systèmes linéaires symétriques définis positifs, la création d'un préconditionneur de type bloc Jacobi permettant de réduire le nombre de conditionnement du système linéaire doit surtout prendre en compte l'amplitude des connexions entre les blocs, et non uniquement le nombre de ces connexions. Cela peut être envisagé comme le complémentaire de [16]. En effet, dans [16], le but est de maximiser les valeurs numériques à l'intérieur des blocs, tandis que dans [21], l'objectif est de minimiser les valeurs numériques à l'extérieur des blocs. L'intérêt de mettre les valeurs dominantes de la matrice au plus près de sa diagonale pour créer un préconditionneur efficace est aussi évoqué par Manguoglu *et al.* dans [100], leur but étant de trouver un préconditionneur à bande étroite.

Dans cette section, nous créons deux types de préconditionneurs bloc Jacobi, en post-traitant la structure par blocs de lignes et de colonnes retournée par notre algorithme spectral. Nous comparons ensuite les performances de ces préconditionneurs avec ceux obtenus par la méthode de Zhu et de Sameh en terme de nombre d'itérations du solveur itératif `gmres` sur treize matrices issues de SuiteSparse Matrix Collection [3], dont les caractéristiques sont données Table 3.3.

En bref, le solveur itératif `gmres`, proposé par Saad et Schultz dans [101], permet de résoudre des systèmes linéaires non symétriques en se basant sur les espaces de Krylov. Pour ce faire, à l'itération k, le solveur approche la solution numérique du système par le vecteur $\mathbf{x}_k$ du sous-espace de Krylov de dimension k ($\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = Vect(\mathbf{b}, \mathbf{Ab}, ...\mathbf{A}^k\mathbf{b})$) tel que la norme du résidu du système, définie par $||\mathbf{b} - \mathbf{Ax}_k||_2$, est minimisée. L'espace de Krylov gagne une dimension à chaque itération, ce qui implique que le solveur doit en théorie converger au bout de n itérations, avec n la dimension de la matrice $\mathbf{A}$.

3.2.1 Préconditionneur de Zhu et Sameh (ZS)

Dans leur papier [16], Zhu et Sameh génèrent des préconditionneurs de type bloc Jacobi possédant des blocs diagonaux dominants, et un minimum d'entrées non nulles à l'extérieur de ces blocs. Ils utilisent ces préconditionneurs pour faire décroître le nombre d'itérations d'un solveur itératif basé sur les méthodes de Krylov. Afin de créer un tel préconditionneur étant donnée une matrice $\mathbf{A}$, ils commencent par permuter la matrice de sorte à obtenir la matrice $\tilde{\mathbf{A}}$ dont le produit des entrées diagonales est maximisé. Ils partitionnent ensuite la matrice $\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^T$ en 8 blocs, de sorte que la taille des blocs est équilibrée en même temps que la coupe pondérée entre ces blocs est minimisée.

Nous avons implémenté l'algorithme présenté par Zhu et Sameh pour générer de

TABLE 3.3 : *Caractéristiques des matrices tests et nombre de blocs pour les préconditionneurs V0 et V1. On remarque que pour V1, la cible de 8 blocs n'est pas toujours atteignable, étant donné que la matrice condensée est généralement de petite taille.*

Matrices	Taille	Nombre de nonzeros	Nombre de Cond.	V0 : nb de blocs	V1 : nb de blocs
`685_bus`	685	3249	4.23×10^5	25	8
`af23560`	23560	460598	1.99×10^4	27	8
`cage8`	1015	1103	1.14×10^1	19	8
`cage9`	3534	41594	1.26×10^1	25	8
`e40r0100`	17281	553562	1.51×10^8	27	8
`epb1`	14734	95053	5.94×10^3	35	8
`gre_1107`	1107	5664	3.19×10^7	10	6
`Hamrle2`	5952	22162	6.69×10^4	617	8
`Ill_Stokes`	20896	191368	2.25×10^9	18	8
`ns3Da`	20414	1679599	7.07×10^2	46	8
`nemeth26`	9506	1511760	1.00×10^0	13	7
`rbsb480`	480	17088	1.20×10^4	7	3
`sme3Da`	12504	874887	5.12×10^7	15	5

tels préconditionneurs, en suivant les directives de leur article. Nous noterons ZS cette méthode. Supposons que nous souhaitons préconditionner le système de l'équation (3.1), avec $\mathbf{A}$ carrée. Dans un premier temps, nous cherchons la permutation des colonnes de $\mathbf{A}$ de sorte que le produit de ses éléments diagonaux soit maximisé. On cherche donc à résoudre

$$\text{Trouver } \pi : \{1...n\} \to \{1...n\} \begin{cases} \pi = \underset{\sigma}{argmax} \left| \prod_{i=1}^{n} a_{i,\sigma(i)} \right| \\ s.t.\ \forall i \neq j, \pi(i) \neq \pi(j) \end{cases} \tag{3.5}$$

Pour ce faire, nous appliquons la méthode `MC64` de la librairie `HSL` [1], décrite dans [102], sur $\mathbf{A}$, ce qui nous renvoie une matrice de permutations $\mathbf{Q}$ telle que

$$\forall i, j \in \{1...n\}, q_{i,j} = \begin{cases} 1 \text{ si } \pi(i) == j \\ 0 \text{ sinon} \end{cases}$$

avec π la permutation solution de l'équation (3.5). `MC64` retourne aussi deux facteurs d'équilibrage $\mathbf{u}, \mathbf{v} \in {\mathbb{R}_+^*}^n$ de sorte que la matrice $\tilde{\mathbf{A}} = \mathcal{D}(\mathbf{u})\mathbf{A}\mathcal{D}(\mathbf{v})\mathbf{Q}$ soit telle que ses

1. HSL, a collection of Fortran codes for large-scale scientific computation. See `http://www.hsl.rl.ac.uk/`

éléments diagonaux sont tous égaux à 1 en valeur absolue, alors que ses éléments hors-diagonaux sont tous inférieurs ou égaux à 1 en valeur absolue. $\mathcal{D}$ est l'opérateur diagonal. Nous notons que dans l'algorithme initial de Zhu et Sameh, les facteurs d'équilibrage ne sont pas pris en compte, bien que les expériences numériques de [102] illustrent que la prise en compte dans `gmres` de l'équilibrage lié à cette permutation contribue à faire décroître le nombre d'itérations. Dans nos versions, nous nous servons de notre matrice doublement stochastique en prenant en compte les facteurs d'équilibrage, étant donné que la matrice équilibrée est généralement mieux conditionnée que la matrice initiale [27]. Pour une comparaison plus juste, nous avons donc aussi pris en compte les facteurs d'équilibrage retournés par `MC64` dans ZS.

Une fois la matrice $\tilde{\mathbf{A}}$ obtenue, nous appliquons la méthode `kway` de la librairie `Metis` [103] sur $\frac{1}{2}\left(|\tilde{\mathbf{A}}| + |\tilde{\mathbf{A}}|^T\right)$, où

$$\forall i, j \in \{1...n\}, |\tilde{a}|_{i,j} = |\tilde{a}_{i,j}|.$$

Cette méthode permet de partitionner un graphe en un nombre de blocs donné. On obtient donc une matrice de permutations $\mathbf{P}$ permettant de réorganiser (symétriquement) la matrice $\left(\tilde{\mathbf{A}} + \tilde{\mathbf{A}}^T\right)$ de sorte à faire apparaître 8 blocs carrés diagonaux qui soient à la fois équilibrés en terme de dimension, et tels que la coupe résultant du partitionnement soit minimisée. On a en outre les indices de séparations entre les blocs. En notant $\hat{\mathbf{A}} = \mathbf{P}\mathcal{D}(\mathbf{u})\mathbf{A}\mathcal{D}(\mathbf{v})\mathbf{Q}\mathbf{P}^T$, on peut créer

$$\mathbf{M} = \begin{bmatrix} \hat{\mathbf{A}}_1 & & \\ & \ddots & \\ & & \hat{\mathbf{A}}_8 \end{bmatrix}$$

le préconditionneur, conformément à la partition en 8 blocs retournée par `kway`.

Ainsi, `gmres` préconditionné par ZS va résoudre le système linéaire suivant

$$\begin{cases} \mathbf{M}^{-1}\mathbf{P}\mathcal{D}(\mathbf{u})\mathbf{A}\mathcal{D}(\mathbf{v})\mathbf{Q}\mathbf{P}^T\mathbf{z} = \mathbf{f} \\ \mathbf{x} = \mathcal{D}(\mathbf{v})\mathbf{Q}\mathbf{P}^T\mathbf{z} \\ \mathbf{f} = \mathbf{M}^{-1}\mathbf{P}\mathcal{D}(\mathbf{u})\mathbf{b} \end{cases} \tag{3.6}$$

au lieu de celui de l'équation (3.1).

3.2.2 Notre premier préconditionneur (V0)

Dans notre algorithme présenté au Chapitre 2, après analyse des vecteurs singuliers gauches et droites séparément, nous obtenons des blocs rectangulaires de tailles variables. Afin d'obtenir un préconditionneur de type bloc Jacobi efficace, nous voudrions transformer ce découpage afin d'obtenir une configuration où les blocs diagonaux sont carrés et de poids maximal. Pour ce faire, nous commençons par trouver la permutation des colonnes de notre matrice $\tilde{\mathbf{A}}$ doublement stochastique maximisant le produit des éléments diagonaux, conformément à l'équation (3.5). Comme pour ZS, après application de `MC64`, on obtient une matrice de permutations $\mathbf{Q}$, telle que les éléments dominants de $\tilde{\mathbf{A}}\mathbf{Q}$ soient sur la diagonale. Puisqu'un partitionnement en lignes et colonnes de la matrice $\tilde{\mathbf{A}}$ a déjà été trouvé, on re-découpe $\tilde{\mathbf{A}}\mathbf{Q}$ conformément à ce partitionnement tout en forçant les blocs diagonaux à être carrés. Ces étapes sont présentées sur les graphes (a) et (b) de

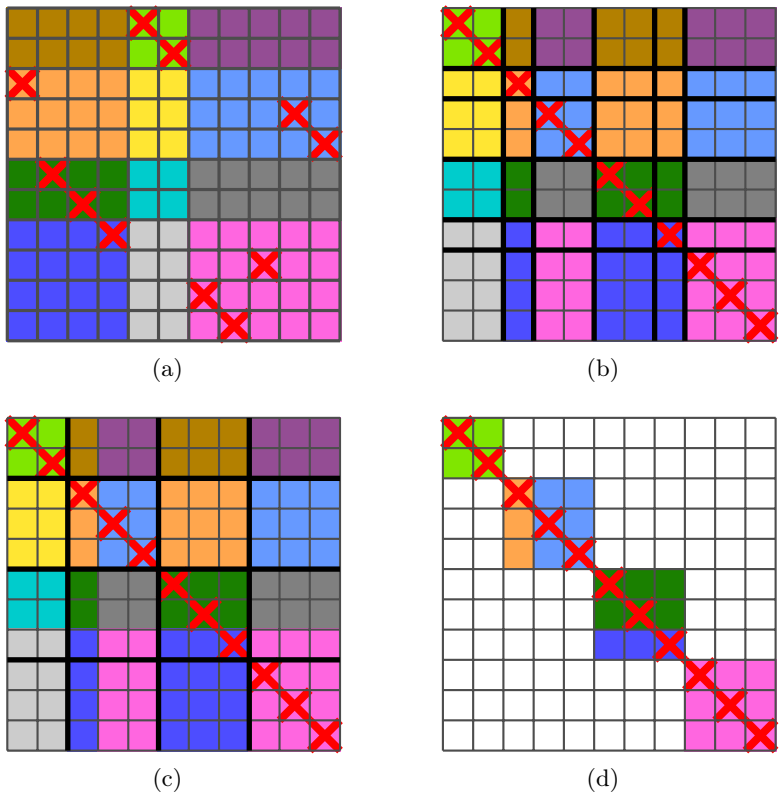


FIGURE 3.9 : *Illustration du post-traitement appliqué à la structure par blocs rectangulaires issue de l'analyse des vecteurs singuliers gauches et droits séparément, en vu d'obtenir une structure par blocs diagonaux plus dominants et carrés.*

la Figure 3.9. Dans ces figures, chaque couleur représente un bloc rectangulaire trouvé par notre algorithme spectral, et les croix rouges correspondent aux éléments que l'on va permuter sur la diagonale. Sur la Figure 3.9(b), on fait apparaître en traits noirs le découpage en blocs conforme à la partition obtenue via l'analyse spectrale et au fait d'avoir des blocs diagonaux dominants et carrés.

Cependant, ce processus peut entraîner la création de tous petits blocs, comme illustré aux Figures 3.10(b) et 3.11(a). Pour éviter cet inconvénient, nous effectuons une version symétrique du processus de fusion décrit Section 2.5.2. Cette fusion symétrique est basée sur la somme des modularités des blocs de lignes et des blocs de colonnes : $\mathcal{Q}_r + \mathcal{Q}_c$. Chaque fois qu'une paire maximisante $(k, \tilde{k})$ est trouvée, on fusionne à la fois les lignes

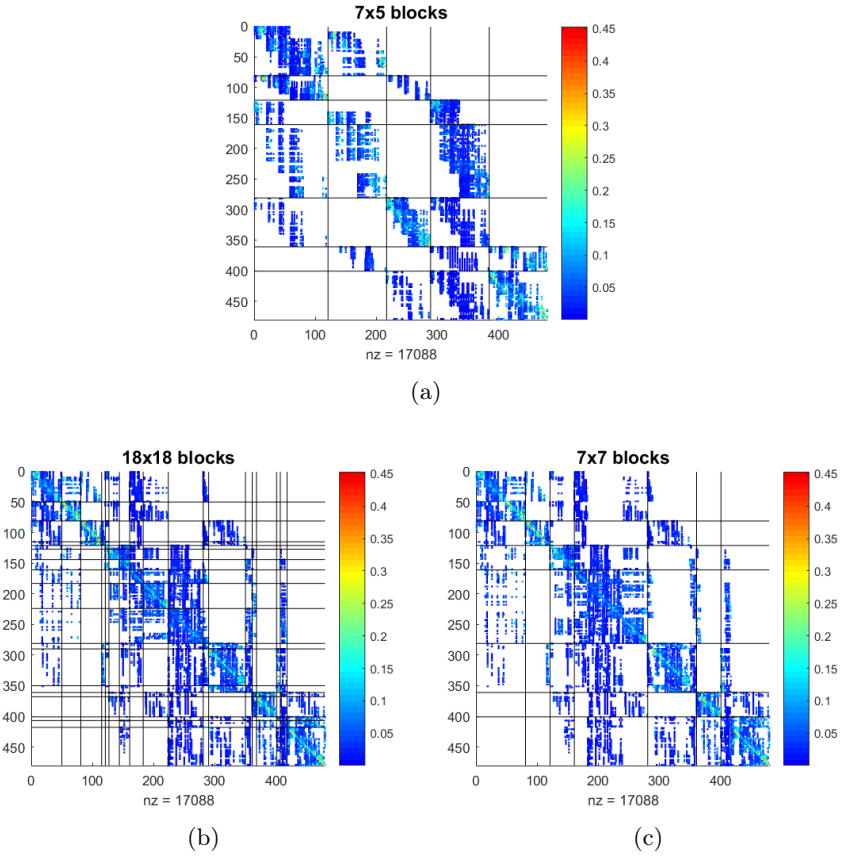


FIGURE 3.10 : *Poids maximal sur la diagonale : illustration sur* `rbsb480`. *(a) : La structure rectangulaire retournée par l'analyse des vecteurs singuliers lignes et colonnes, séparément. (b) : Découpage obtenu après avoir permuté les éléments dominants sur la diagonale. (c) : Structure par blocs diagonaux obtenue après le processus de fusion symétrique.*

et les colonnes correspondantes. Ce processus est illustré Figure 3.9(c), à partir du partitionnement de la Figure 3.9(b). On remarque que le partitionnement obtenu après fusion est symétrique. Les Figures 3.10(c) et 3.11(b) illustrent l'avantage de cette étape de fusion sur des matrices réelles issues de la collection SuiteSparse [3].

Ce processus de fusion symétrique nous fournit un nouveau partitionnement symétrique de la matrice $\tilde{\mathbf{A}}\mathbf{Q}$, caractérisé par une matrice de permutations $\mathbf{P}$ et un partitionnement de $\{1...n\}$ en un nombre quelconque de blocs. Le préconditionneur $\mathbf{M}$ en résultant est constitué des blocs carrés diagonaux de la matrice $\mathbf{P}\tilde{\mathbf{A}}\mathbf{Q}\mathbf{P}^T$, comme illustré Figure 3.9(d).

Ainsi, `gmres` préconditionné par V0 va résoudre le système linéaire suivant

$$\begin{cases} \mathbf{M}^{-1}\mathbf{P}\mathcal{D}(\mathbf{r})\mathbf{A}\mathcal{D}(\mathbf{s})\mathbf{Q}\mathbf{P}^T\mathbf{z} = \mathbf{f} \\ \mathbf{x} = \mathcal{D}(\mathbf{s})\mathbf{Q}\mathbf{P}^T\mathbf{z} \\ \mathbf{f} = \mathbf{M}^{-1}\mathbf{P}\mathcal{D}(\mathbf{v})\mathbf{b} \end{cases} \tag{3.7}$$

au lieu de celui de l'équation (3.1). Dans cette équation, $\mathbf{r}, \mathbf{s} \in {\mathbb{R}_+^*}^n$ sont les facteurs de l'équilibrage doublement stochastique de la matrice $\mathbf{A}$, c'est-à-dire tels que $\tilde{\mathbf{A}} = \mathcal{D}(\mathbf{r})|\mathbf{A}|\mathcal{D}(\mathbf{v})$ est doublement stochastique, avec $|\mathbf{A}|$ la matrice telle que

$$\forall i, j \in \{1...n\}, |a|_{i,j} = |a_{i,j}|.$$

En comparant les barres en bleu et en vert de la Figure 3.12 affichant le nombre d'itérations nécessaire à la convergence de `gmres` pour les différents préconditionneurs, nous remarquons que notre processus ainsi implémenté ne permet pas de créer un préconditionneur de type Jacobi par blocs efficace en comparaison à notre implémentation de ZS.

3.2.3 Notre second préconditionneur (V1)

Au regard de la Table 3.3 et de la Figure 3.12, il nous semble que la différence dans le nombre de blocs joue un rôle important dans l'efficacité comparée des deux préconditionneurs : la matrice `rbsb480`, qui est la seule pour laquelle le nombre de blocs retourné par notre méthode est plus faible que le nombre de blocs retourné par ZS, est aussi la seule matrice pour laquelle notre méthode permet un nombre d'itérations plus faible. *A contrario*, notre algorithme échoue à permettre la convergence de `gmres` pour la matrice `Hamrle2`, qui est celle pour laquelle V0 retourne le plus grand nombre de blocs – soit 617. La méthode ZS, quant à elle, fixe le nombre de blocs *a priori* – dans nos tests, nous avons fixé ce nombre à 8.

Nous avons donc testé une autre approche permettant de contraindre le nombre

maximal de blocs. Pour ce faire, nous commençons toujours par trouver le découpage en blocs diagonaux qui soit conforme à la partition obtenue par l'analyse spectrale et au fait d'avoir des blocs diagonaux dominants et carrés – cf Figure 3.9(b). En reprenant les notations de la section précédente, à partir de ce découpage et de notre matrice $\hat{\mathbf{A}} = \tilde{\mathbf{A}}\mathbf{Q}$, nous créons une matrice condensée, que nous noterons $\hat{\mathbf{A}}_c$. Dans cette nouvelle matrice, le poids d'une entrée (i, j) est égal à la somme des éléments dans le bloc (i, j) de la matrice $\hat{\mathbf{A}}$. Une illustration de la matrice condensée obtenue sur `Ill_Stokes` est donnée Figure 3.11(c)

Nous appliquons sur $\hat{\mathbf{A}}_c$ le processus de partitionnement des graphes `kway` issu de

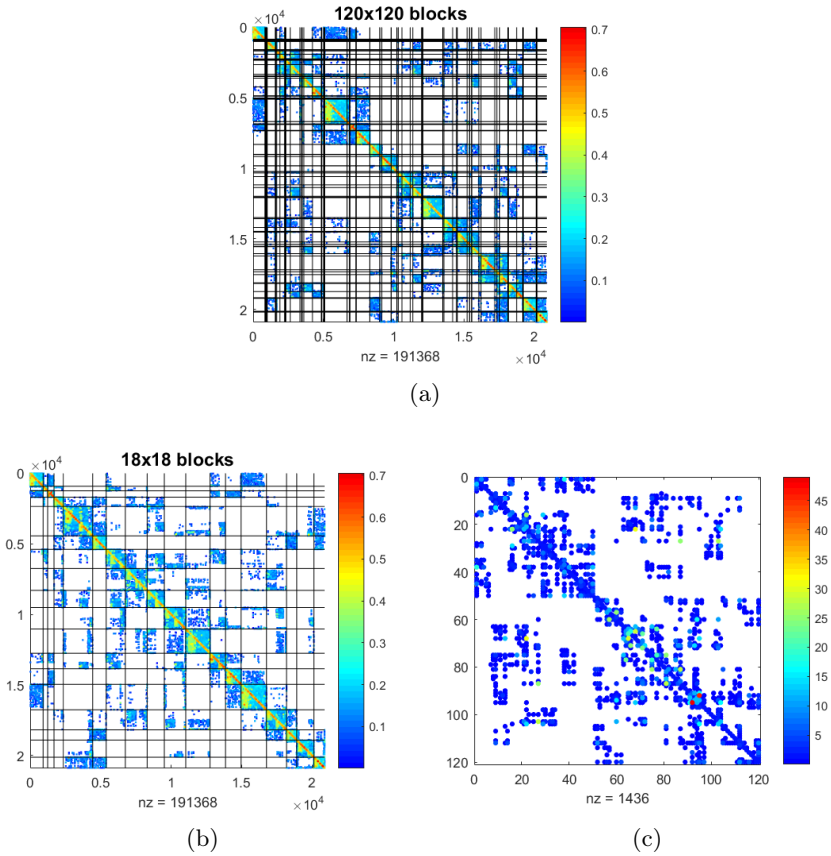


FIGURE 3.11 : *Matrice* `Ill_Stokes`. *(a) : Le partitionnement de la matrice après avoir ramené les éléments dominants sur sa diagonale. La matrice possède alors 120 blocs diagonaux. (b) : Le partitionnement de la matrice obtenu après application du processus de fusion symétrique sur le partitionnement présenté en (a). (c) : Matrice condensée* 120×120 *construite à partir de la matrice et de son partitionnement présenté en (a).*

`Metis` [103], afin de partitionner $\hat{\mathbf{A}}_c + \hat{\mathbf{A}}_c^T$ en 8 blocs, de la même façon que pour ZS. Le préconditionneur obtenu via cette méthode sera noté V1 dans la suite.

Le préconditionneur obtenu via V1 amènera à la résolution d'un système équivalent à celui de l'équation (3.7). On observe sur la Figure 3.12 que les résultats obtenus via V1 sont comparables à ceux obtenus par ZS en terme de nombre d'itérations dans `gmres`.

3.2.4 Conclusion et perspectives

Les résultats retournés par notre version V1 du préconditionneur sont équivalents à ceux retournés par ZS en terme de nombre d'itérations pour `gmres`. Il serait intéressant de voir dans quelle mesure ces bons résultats se confirment sur une architecture parallèle.

Cependant, ces résultats ne représentent qu'une première étape dans l'adaptation de notre algorithme aux problèmes de préconditionnement. Pour pouvoir garantir une utilisation plus générale de notre méthode pour ce type d'application, une étude plus approfondie doit encore être menée. En effet, les treize matrices sur lesquelles nous avons mené notre étude ont toutes la particularité d'être bi-irréductible. Nous rappelons que

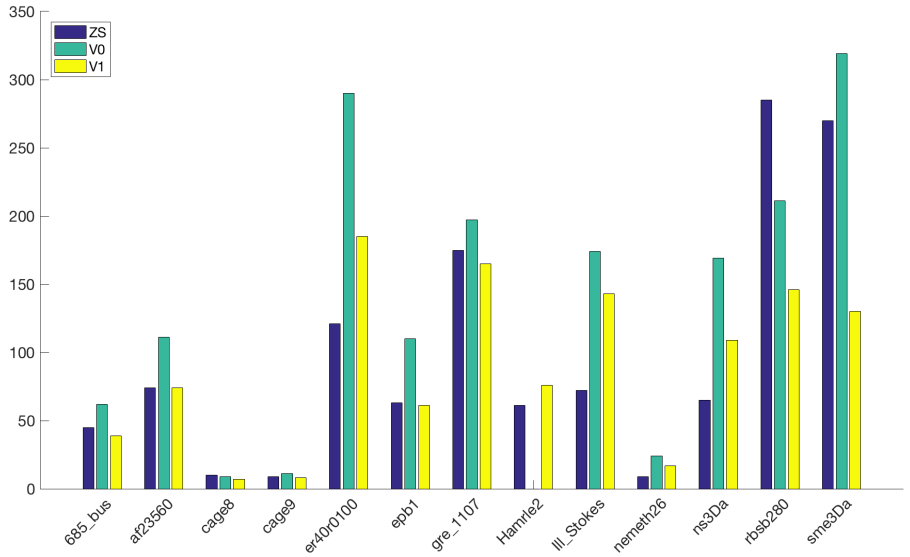


FIGURE 3.12 : *Comparaison entre le nombre d'itérations nécessaire à la convergence du solveur* `gmres` *pour une tolérance du résidu en* 10^{-6}*, avec les trois types de préconditionnement ZS, V0 et V1. C'est le résidu non préconditionné qui est calculé. Pour la matrice* `Hamrle2`*,* `gmres` *préconditionné par V0 échoue à converger.*

cette caractéristique est essentielle pour la mise sous forme doublement stochastique, point crucial de notre algorithme. Pour une matrice qui n'est pas bi-irréductible, nous pouvons la permuter pour la mettre sous forme bloc triangulaire, puis traiter chaque bloc diagonal bi-irréductible de façon indépendante. Cependant, il faut réfléchir à un traitement des différentes composantes : comment regrouper les partitions obtenues pour chaque bloc bi-irréductible indépendamment, et comment prendre en compte dans le préconditionneur les blocs hors-diagonaux de la forme triangulaire par blocs – qui ne sont *a priori* ni bi-irréductibles ni même carrés et peuvent contenir n'importe quelle valeur numérique de la matrice. Il y a aussi un grand nombre de matrices pour lesquelles une mise sous forme bloc triangulaire engendre l'apparition de très nombreux blocs diagonaux ne contenant qu'un élément. La réorganisation de ces mini blocs est essentielle pour une telle application. En bref, une analyse approfondie du type de structure des matrices que l'on souhaite préconditionner devrait être menée, le traitement de la structure creuse n'étant pas aussi simple que pour les applications de type classification.

Il reste néanmoins que cette première étude de notre méthode pour générer des préconditionneurs bloc Jacobi efficaces est concluante au regard des résultats de la Figure 3.12. En outre, d'autres types de préconditionnement par blocs peuvent être envisagés pour cette méthode, par exemple un préconditionnement pour un solveur bloc Cimmino, en mettant l'accent sur l'obtention de blocs de lignes décorrélés – voir par exemple [104]. En effet, notre algorithme retourne des partitionnements par blocs de lignes et de colonnes de façon indépendantes, partitionnements obtenus sans contrainte sur le nombre ou la tailles des blocs et prenant en compte aussi bien la structure creuse que les valeurs numériques de la matrice passée en paramètre. Cela permet une grande malléabilité des applications via des pré- et des post-traitements adéquats.

3.3 Traitement automatique du langage naturel

La détection de structures par blocs dans les matrices a des applications variées. En effet, nous avons vu aux deux sections précédentes son intérêt pour la détection de communautés ou le préconditionnement de systèmes linéaires. Cependant, pour ces deux applications, la représentation matricielle du problème est directe. Il existe en revanche bien d'autres applications *a priori* moins directes dans lesquelles les données peuvent être représentées, soit sous forme de matrices, soit sous forme de vecteurs – ce qui permet une représentation matricielle via les matrices d'affinité ou les tables de données. Nous allons nous intéresser dans cette section au traitement automatique du langage naturel, et utiliser la détection de structures dans les matrices pour répondre à des problématiques

de ce domaine.

3.3.1 Représentation vectorielle du langage

Le traitement automatique du langage naturel, comme son nom l'indique, vise à traiter des informations issues du langage naturel de façon automatique. Les applications sont diverses (générateur de réponses automatiques dans des chats [105], traitement de requêtes fournies en langage naturel à un moteur de recherche [106, 107], traducteur automatique [108], générateur de textes – le recueil de poèmes *Il pleure dans mon processeur multi-coeur*[2] auto-généré via [109] par exemple, etc.). Un des points clés du domaine est de représenter de façon formelle le-dit langage naturel de sorte à véhiculer de l'information ou de la sémantique. La pièce élémentaire de la sémantique étant le mot, de nombreuses façons de représenter les mots dans des espaces vectoriels ont été créés depuis la fin des années 50. Cette représentation permet de mesurer l'écart sémantique entre deux mots : on est capable de connaître la proximité sémantique ou syntaxique entre deux mots en calculant la distance ou l'angle entre les deux vecteurs qui les représentent [110]. Dans une première approche de 1957 décrite dans [111] par exemple, les chercheurs prédéfinissaient une base de mots et demandaient à des personnes de placer un certain nombre de mots dans leur espace, représentant ainsi la proximité ou l'éloignement de sens entre les mots testés. Avec l'arrivée de corpus de textes, des façons plus automatiques de représenter les mots sont apparues. On peut les diviser en deux grandes familles. **Les méthodes de factorisation matricielle** d'une part, et les **méthodes à fenêtre courte** d'autre part. Les premières sont non supervisées et se basent essentiellement sur les statistiques d'un corpus de textes. Les deuxièmes sont supervisées et généralement associées à une tâche, par exemple détecter le prochain mot dans une suite de mots.

Méthodes de factorisation

L'idée directrice de ce type de méthodes est que deux mots "co-occurrant" régulièrement dans des contextes similaires devraient avoir des sens proches [112]. Dans ce cas on veut que les vecteurs représentant ces deux mots soient proches. Pour ce faire, on représente matriciellement les co-occurrences des mots dans les documents d'un corpus. Ensuite, avec une méthode de réduction de dimensions, on représente tous les mots du corpus dans un espace de faible dimension. De nombreuses variantes existent pour la représentation matricielle, la réduction des dimensions, et la notion de distance utilisée pour comparer des vecteurs. Deux exemples sont donnés ci-dessous.

2. `http://www.timvandecruys.be/media/charles2018.pdf`

L'article [113], présentant la *Latent Semantic Analysis*, est un des papiers pionniers de ce type de méthodes. Le but initial des auteurs est de fournir un document d'un corpus étant donné les mots d'une requête utilisateur. Si l'utilisateur demande de la documentation sur les voitures, il est très probable qu'un document dans lequel le terme "automobile" apparaît de nombreuses fois intéressera cet utilisateur, même si le terme "voiture" n'est pas employé explicitement. Cela justifie la volonté des auteurs de trouver une représentation vectorielle des mots dans laquelle des mots sémantiquement proches seront proches, par exemple où les mots "voiture" et "automobile" sont à peu près indissociables. Les auteurs ont à leur disposition un corpus de texte, c'est-à-dire N documents $d_1, ..., d_N$, chaque document d_i consistant en une suite de mots. Nous notons $\mathcal{V} = \{w_1, ...w_n\}$ le vocabulaire, c'est-à-dire l'ensemble de tous les mots du corpus. Ils déduisent de ce corpus une matrice $\mathbf{F} \in \mathbb{N}^{n \times N}$ telle que $\mathbf{F}_{i,j}$ est le nombre d'occurrence du mot w_i dans le document d_j. Pour obtenir une représentation véhiculant de la sémantique, ils font une approximation de rang faible de leur matrice : ils calculent la SVD de $\mathbf{F}$

$$\begin{cases} \mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ \mathbf{U} \in \mathbb{R}^{n \times n} : \mathbf{U}^T\mathbf{U} = \mathbf{I}_n \\ \mathbf{V} \in \mathbb{R}^{N \times N} : \mathbf{V}^T\mathbf{V} = \mathbf{I}_N \\ \mathbf{\Sigma} \in \mathbb{R}^{n \times N} \\ \sigma(1,1) \geq \sigma(2,2) \geq ... \geq \sigma(min(n,N), min(n,N)) \end{cases}$$

et construisent $\tilde{\mathbf{F}}$ l'approximation de rang k de $\mathbf{F}$:

$$\begin{cases} \tilde{\mathbf{F}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T \\ \tilde{\mathbf{U}} \in \mathbb{R}^{n \times k} : \tilde{\mathbf{U}}^T\tilde{\mathbf{U}} = \mathbf{I}_k \\ \tilde{\mathbf{V}} \in \mathbb{R}^{N \times k} : \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \mathbf{I}_k \\ \tilde{\mathbf{\Sigma}} \in \mathbb{R}^{k \times k} \end{cases}$$

Le vecteur représentant le mot w_i est alors un vecteur de $\mathbb{R}^k$, soit donné par la i^{eme} ligne de $\tilde{\mathbf{U}}$, soit par la i^{eme} ligne de $\tilde{\mathbf{U}}\tilde{\Sigma}$. Pour comparer deux termes, les auteurs utilisent le produit scalaire ou similarité cosinus. Ils considèrent donc deux mots comme sémantiquement équivalents s'ils sont colinéaires et de même direction.

La deuxième approche est celle de HAL [114]. La problématique des auteurs de cet article est de créer un espace vectoriel de représentation des mots qui soit un espace de représentation sémantique, de façon automatique. Pour ce faire, les auteurs partent

du principe que si des mots se retrouvent dans le même contexte – c'est-à-dire sont régulièrement précédés ou suivis par les mêmes mots –, alors leur sens doit être proche. Leur représentation de cette notion passe par ce qu'ils appellent la matrice de co-occurrence, qui est une matrice carrée dans laquelle chaque ligne et chaque colonne correspond à un mot du vocabulaire, et l'entrée (i, j) de cette matrice est le nombre de fois que le mot i est précédé par le mot j, pondéré par une certaine distance paramétrée par une fenêtre, qui correspond au contexte pris en compte (c'est-à-dire au nombre de mots précédant le mot i auquel on se limite). Comme cette matrice est difficile à expliciter formellement, nous illustrons sa création par un fameux petit exemple issu de la pièce *Cyrano de Bergerac* [115] :

C'est un roc !... C'est un pic !... C'est un cap !...

Supposons que nous nous intéressons à un contexte de taille 3. Nous "scannons" chaque terme i de cette phrase, et attribuons un poids égal à 3 au mot précédant directement le mot i, égale à 2 au mot précédant le mot précédant le mot i, etc. Cela nous donne la Table 3.4.

	c	est	un	roc	c	est	un	pic	c	est	un	cap
c												
est	3											
un	2	3										
roc	1	2	3									
c		1	2	3								
est			1	2	3							
un				1	2	3						
pic					1	2	3					
c						1	2	3				
est							1	2	3			
un								1	2	3		
cap									1	2	3	

TABLE 3.4 : *Création de la matrice des co-occurrences, étape 1 : scan du texte.*

En regroupant les mots apparaissant plusieurs fois, on obtient la matrice de co-occurrences finale montrée Table 3.5, qui correspond bien à ce que l'on souhaite. Par exemple le mot "un" précède deux fois le mot "c" avec un poids de deux ; dans la matrice, l'entrée correspondant à ("c","un") est bien égale à 4. On remarque en outre que cette matrice contient dans ses colonnes l'information sur les occurrences des mots suivant un mot donné : prenons la colonne de "c" : ce mot est directement suivi par le mot "est"

trois fois, et l'entrée ("c","est") vaut trois (nombre d'occurrences) fois trois (poids des occurrences).

De cette matrice, les auteurs extraient des vecteurs de représentation des mots de dimension $2n$, n étant la taille du vocabulaire, en concaténant l'information de la ligne et de la colonne d'un mot. Dans notre exemple, les mots "cap", "pic" et "roc" seront représentés par un même vecteur : $(1,2,3,0,0,0,3,2,1,0,0,0)^T$, ce qui signifie qu'ils sont égaux dans l'espace de représentation des mots. Et il est vrai qu'au regard uniquement de l'extrait de texte étudié, ces trois mots jouent strictement le même rôle, à la fois sémantique et syntaxique. La dimension de ces vecteurs peut être réduite par une analyse en composantes principales de la matrice de co-occurrences.

Les auteurs évaluent la distance sémantique entre deux mots en mesurant leur distance de Minkowsky, c'est-à-dire que si $\mathbf{w}, \mathbf{v} \in \mathbb{R}^{2k}$ sont deux mots, ils mesurent

$$d_p(\mathbf{w}, \mathbf{v}) = \left(\sum_{i=1}^{2k} |\mathbf{w}(i) - \mathbf{v}(i)|^p \right)^{1/p},$$

avec p un paramètre à fixer. Pour éviter que la norme du mot entre en compte – la norme d'un mot ne traduisant que son nombre d'occurrence dans le corpus – ils normalisent préalablement les vecteurs de mots.

Pour ces méthodes, de nombreuses modifications et améliorations sont possibles. Par exemple, après création des matrices, on peut normaliser un mot par son nombre d'occurrence dans le corpus, pour éviter de donner un poids prépondérant à des mots très fréquents et véhiculant peu de sens ("le", "et", "du"...). On peut aussi normaliser les documents par leur taille, pour éviter que le document contenant le plus de mots se voit accorder un poids trop fort. L'article [116] présente et compare les différences existant entre les différents types de méthodes de factorisation et leurs améliorations. Une étude

	c	est	un	roc	pic	cap
c		2	4	3	3	3
est	9	2		2	2	2
un	6	9		1	1	1
roc	1	2	3			
pic	1	2	3			
cap	1	2	3			

TABLE 3.5 : *Création de la matrice de co-occurrences, étape 2 : regroupement des termes apparaissant plusieurs fois par somme des lignes et des colonnes correspondantes.*

générale et historique de ce type de méthodes est présentée dans [112].

Méthodes à fenêtre courte

Les méthodes de factorisation sont uniquement basées sur l'analyse statistique d'un corpus. Elles sont donc non supervisées. Le deuxième type de méthodes, en revanche, construit une représentation vectorielle des mots étant donnée une cible, un problème à résoudre. C'est-à-dire que dans ce type de méthodes, on apprend la représentation des mots qui donne les meilleurs résultats pour une tâche, typiquement trouver le mot suivant dans une suite de mots. C'est d'ailleurs l'application choisie dans l'article [4]. Comme souvent, les auteurs disposent d'un corpus dont ils ont extrait un vocabulaire $\mathcal{V}$ de dimension n. Les auteurs implémentent alors un réseau de neurones pour trouver une fonction modèle f qui représente la probabilité conditionnelle du corpus d'avoir comme mot suivant w_{t+1} étant donnée une suite de mots $w_t, ..., w_1 = w_1^t$; t est la fenêtre du contexte pris en compte, de la même façon que dans l'article [114]. Le choix du mot suivant à partir de la fonction f se fait par maximum de vraisemblance, c'est-à-dire que l'on va choisir

$$w_{t+1} = \underset{w \in \mathcal{V}}{argmax}\ f(w, w_t, ..., w_1)$$

Initialement, un mot est simplement représenté par son indice dans le vocabulaire. Le réseau de neurones prend alors comme entrées l'indice du mot w_t, l'indice du mot w_{t-1},..., l'indice du mot w_1 et renvoie un vecteur $P \in [0,1]^n$ tel que $P(i)$ est la probabilité que le mot w_{t+1} suivant la séquence w_1^t soit le i^{eme} mot du vocabulaire, c'est-à-dire $P(i) = f(i, w_t, ..., w_1)$. Le réseau de neurones est divisé en deux couches – ou mappings. La première se charge d'associer un vecteur de dimension d fixé à un mot. Le réseau de neurones va donc apprendre les entrées d'une matrice $\mathbf{C} \in \mathbb{R}^{d \times n}$, dans laquelle la i^{eme} colonne correspond à la représentation vectorielle du i^{eme} mot de $\mathcal{V}$. Cette couche est partagée par les t entrées du réseau. La deuxième couche associe un élément de $[0,1]$ à l'indice i sachant les vecteurs $\mathbf{C}(w_t), ..., \mathbf{C}(w_1)$, pour tous les $i \in \{1, ..., n\}$. Cette couche apprend donc une loi de probabilité conditionnelle à partir des représentations vectorielles des mots : $g(w_{t+1} = i | \mathbf{C}(w_t), ..., \mathbf{C}(w_1))$. A la fin de la phase d'apprentissage, la matrice $\mathbf{C}$ issue du premier mapping permet donc de représenter les mots du vocabulaire, de sorte que la fonction f donnée par le réseau, et définie par :

$$f(i, w_t, ..., w_1) = g(w_{t+1} = i | \mathbf{C}(w_t), ..., \mathbf{C}(w_1))$$

réalise au mieux la tâche de prédiction du mot suivant, au sens du maximum de vraisem-

blance. Une représentation simplifiée du réseau proposé dans [4] est donnée Figure 3.13. Pour d la dimension de l'espace vectoriel représentant les mots, les auteurs de [4] font des

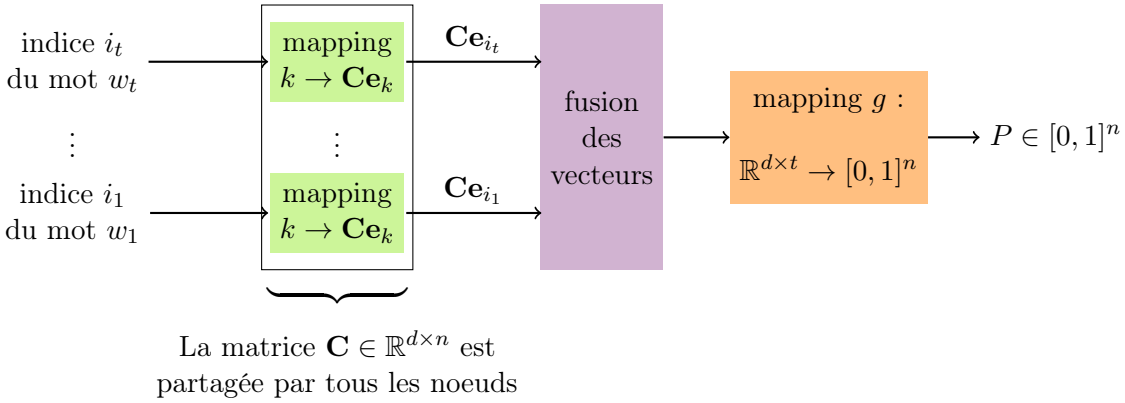


FIGURE 3.13 : *Architecture simplifiée du réseau proposé dans [4].* $\mathbf{e}_k$ *est le vecteur ayant un 1 à l'indice* k *et des 0 partout ailleurs. A la fin du processus d'apprentissage, la* k^{ieme} *colonne de la matrice matrice* $\mathbf{C}$ *produite par ce réseau contient le vecteur de* $\mathbb{R}^d$ *représentant le* k^{ieme} *mot de* $\mathcal{V}$.

tests pour $d = 30, 60$ et 100, et obtiennent leurs meilleurs résultats pour $d = 100$.

De nombreuses autres méthodes existent avec différentes tâches d'apprentissage. Parmi elles, les méthodes CBOW et skip-gram [117] sont très utilisées. La première a pour but de trouver un mot étant donné son contexte à droite et à gauche (en reprenant les notations précédentes, il s'agit de prédire w_{t+1} sachant $w_1, ..., w_t, w_{t+2}, ..., w_{2t}$). Pour la seconde, le but est de déterminer le contexte d'un mot – *id est* définir $w_1, ..., w_t, w_{t+2}, ..., w_{2t}$ sachant w_{t+1}.

Méthode utilisée dans la suite : GloVe

Nous avons vu qu'il existe de nombreuses façons de représenter les mots dans un espace vectoriel. Dans la suite de cette section, nous avons sélectionné le modèle GloVe [110]. Il s'agit d'une méthode non supervisée. L'idée principale de cette méthode est que ce qui qui véhicule l'information sémantique, c'est le ratio des probabilités de co-occurrence des mots. Afin d'expliciter cela, nous reprenons l'exemple proposé dans l'article [110]. Supposons que l'on s'intéresse à la relation sémantique des termes w_i ="glace" et w_j="vapeur" dans le contexte de la thermodynamique. En notant $X_{r,s}$ le nombre de fois que le mot w_s est rencontré dans le contexte du mot w_r, et $P_{r,s} = P(s|r) = X_{r,s}/\sum_t X_{r,t}$, la probabilité que le mot w_s apparaisse dans le contexte du mot w_r, il devrait apparaître que si un mot

w_k est en lien avec “glace” mais pas avec “vapeur”, alors $P_{i,k}/P_{j,k} >> 1$. Réciproquement, si w_k est en lien avec “vapeur” mais pas avec “glace”, alors $P_{i,k}/P_{j,k} << 1$. Les auteurs donnent des valeurs numériques issues d'un corpus réel. En prenant w_k = “solide”, on obtient $P_{i,k}/P_{j,k} = 8.9$. En prenant w_k=“gaz”, on a $P_{i,k}/P_{j,k} = 8.5 \times 10^{-2}$. Et en prenant w_k = “eau”, on a $P_{i,k}/P_{j,k} = 1.36$, ce qui correspond bien au comportement attendu. Les auteurs partent donc du principe que le modèle à apprendre est la fonction

$$F(w_i, w_j, w_k) = \frac{P_{i,k}}{P_{j,k}} \tag{3.8}$$

Après de nombreuses simplifications, ils réécrivent l'équation (3.8) par :

$$\mathbf{w}_i^T \mathbf{w}_k + b_i + c_k = log(X_{i,k}) \tag{3.9}$$

avec $\mathbf{w}_i$ les vecteurs de représentation des mots à trouver et les variables b_i et c_k des biais permettant de conserver la symétrie du modèle initial. Leur but est alors de trouver les vecteurs de représentation permettant la minimisation du problème aux moindres carrés pondérés suivant :

$$J = \sum_{i=1}^{n} \sum_{j=1}^{n} f(X_{i,j}) \left(\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j - log(X_{i,j}) \right)^2$$

avec f une fonction poids permettant de limiter l'impact de la rareté ou de la surabondance d'un terme sur le problème à résoudre.

De nombreuses versions de GloVe sont disponibles [3], et dépendent notamment du corpus sur lequel le modèle a été construit. Dans nos tests, nous prenons les représentations vectorielles issues du modèle entraîné sur le corpus Common Crawl [4] de 42 milliards d'occurrences de mots, ces vecteurs étant de dimension 300.

3.3.2 Base de données STAC

Dans la suite de cette section, nous allons nous intéresser à la base de données STAC (Strategic Conversation). STAC est un projet interdisciplinaire subventionné par le Conseil Européen de la Recherche. Le corpus est issu d'une version en ligne du jeu de plateau *Les Colons de Catane*. Il s'agit d'un jeu multi-joueur dans lequel le but est de construire des colonies, des villes et des routes, grâce à des ressources de plusieurs types (bois, laine,

3. `https://nlp.stanford.edu/projects/glove/`
4. `http://commoncrawl.org/`

blé, argile, minerai). A tour de rôle, chaque joueur lance les dés. Le lancer de dés peut résulter en la distribution de certaines ressources à certains joueurs. S'ensuit une phase de commerce : le joueur dont c'est le tour peut faire des échanges de ressources avec les autres joueurs. En général, il propose à un ou plusieurs joueur(s) d'échanger une ou plusieurs de ses ressources contre une ou plusieurs de leurs ressources. Les autres joueurs peuvent accepter, refuser, ou proposer un autre type d'échange. Dans STAC, les discussions autour de ces phases de commerces ont lieu via un chat. C'est ce chat qui constitue notre base de données.

Dans le chat, les joueurs parlent donc des échanges à effectuer, mais ils peuvent aussi échanger à propos d'autres sujets, qui ont ou non un lien avec le jeu. Le fait qu'il s'agisse d'un chat implique que le langage utilisé est proche d'un langage parlé, tout en ayant l'avantage d'être écrit et donc de ne pas nécessiter de retranscription. Par ailleurs, il peut y avoir des conversations croisées, c'est-à-dire des sous-conversations. Un exemple est donné Figure 3.14, où trois conversations ont lieu de façon croisée. Une de ces conversations porte sur les échanges. Une autre (police en gras) porte sur comment les joueurs ont eu connaissance du jeu. Une troisième (en italique) porte sur la scolarité des joueurs. On voit ici que les sujets évoqués par les joueurs sont divers, et ne se limitent pas strictement à la partie en cours.

Deux corpus

Dans le chat de STAC, on trouve plusieurs types d'interventions. Les interventions des joueurs, que nous avons présentées précédemment, constituent le premier type. Il peut aussi y avoir des spectateurs, c'est-à-dire des gens qui assistent à la partie sans y jouer. Ces spectateurs peuvent commenter la partie, interagir avec les joueurs, etc. Un exemple est donné Figure 3.15(a), où l'intervenant **mk** n'est pas un joueur. Ici il s'agit d'un superviseur donnant des consignes pour l'utilisation du site. Le dernier type d'interventions que l'on trouve dans ce chat correspond aux messages du serveur. Il s'agit de messages envoyés par le serveur dans un langage formaté et donnant des indications sur la partie (résultats des lancers de dés, état des ressources des joueurs, etc.). Un exemple est donné Figure 3.15(b) : on ajoute les messages envoyés par le système pendant la conversation affichée Figure 3.15(a). Ces messages sont ici labellisés **UI** et **SV**.

Dans la suite, nous nous intéresserons donc à deux sous corpus de STAC. Le premier, appelé corpus *Spectateur*, correspond au corpus dans lequel on prend en compte les messages des joueurs et des spectateurs, mais pas ceux du serveur. Dans le corpus appelé corpus *Situé*, on prend en compte les messages des joueurs, des spectateurs et du serveur.

165	lj	anyone want sheep for clay ?
166	gwfs	got none, sorry :(
167	gwfs	**so how do people know about the league ?**
168	wm	no
170	lj	**I did the trials**
174	tk	**I know about it from my gf**
175	gwfs	[**yeah me too,**]$_a$ [*are you an Informatics student then, lj ?*]$_b$
176	tk	**did not do the trials**
177	wm	has anyone got wood for me ?
178	gwfs	[**I did them**]$_a$ [**because a friend did**]$_b$
179	gwfs	lol wm, you cad
180	gwfs	afraid not :(
181	lj	[*no, I'm about to start math.*]$_a$ [*I just hang around appleton a lot*]$_b$
182	tk	sry no
183	gwfs	my single wood is precious
184	wm	what's a cad ?

FIGURE 3.14 : *Conversations croisées dans le corpus STAC. Dans cet exemple, il y a trois sous-dialogues concurrents, chacun représenté par une police différente. Exemple extrait de [5]*

11.1	**mk**	Yes, it's supposed to be three of you.
11.2	**mk**	Hm, how are you two for time ? Can we wait a bit longer ?
12	**tm**	I'm good for a little bit longer.
13	**rc1**	I don't mind waiting for a while.
13.1	**mk**	Great. I guess if person number 3 doesn't show up in 5 or 10 minutes it'll be you only you two anywa
13.2	**mk**	y.
14	**dv**	hi
14.1	**mk**	Ah, great. We can start.
14.2	**mk**	Sombody has to click START GAME.
14.3	**mk**	Amd and off you go !

(a)

13	**rc1**	I don't mind waiting for a while.
13.1	**mk**	Great. I guess if person number 3 doesn't show up in 5 or 10 minutes it'll be you only you two anywa
13.2	**mk**	y.
13.2.1	**UI**	dv joined the game.
13.2.2	**UI**	dv sat down at seat 1.
14	**dv**	hi
14.1	**mk**	Ah, great. We can start.
14.2	**mk**	Sombody has to click START GAME.
14.2.1	**UI**	Game started.
14.2.2	**UI**	Board layout set.
14.2.3	**SV**	Randomly picking a starting player...
14.2.4	**SV**	It's rc1's turn to build a settlement.

(b)

FIGURE 3.15 : *(a) : Un exemple d'échanges dans le chat, avec les messages spectateur, sans les messages serveur. (b) : Le même exemple avec les messages spectateur et serveur*

Découpage et annotations des corpus

Les deux corpus ont été annotés par les chercheurs de l'équipe MELODI de l'IRIT. Les deux corpus sont composés de 34 parties. Chaque partie a été découpée en plusieurs tours

de joueurs, qui correspondent aux messages échangés dans le chat entre chaque lancer de dés jusqu'à ce que la partie soit gagnée. Chaque intervention d'un joueur consiste en ce que l'on appelle une Unité de Discours Complexe (CDU). Ces CDUs peuvent être composées d'une ou plusieurs Unité(s) Elementaire(s) de Discours (EDUs). Par exemple, dans la Figure 3.15(a), le texte correspondant à la sortie 14.1 :"*Ah, great. We can start.*" est une CDU composée de deux EDUs "*Ah, great.*" et "*We can start.*", tandis que la CDU sortie 12 :"*I'm good for a little bit longer.*" est composée d'une unique EDU.

Les messages du serveur consistent en une ou plusieurs Unité(s) Elementaire(s) d'Evènement (EEUs). Dans la suite, nous parlerons indistinctement d'EDUs pour désigner les EDUs ou les EEUs.

A chaque EDU est attribué un Acte de Dialogue (DA). Le DA associé à une EDU représente la signification de cette EDU. Dans le cas particulier de STAC où le chat sert essentiellement à ce que les joueurs marchandent, les EDUs sont annotées suivant 5 classes.

Une EDU peut être

(Of) : du type Offre,

(C-Of) : du type Contre-Offre,

(Ref) : du type Refus,

(Acc) : du type Acceptation,

(Aut) : du type Autre, qui est la classe qui contient les EDUs n'étant pas directement en lien avec la phase de commerce.

Dans l'exemple de la Figure 3.14, l'EDU 165 : "*anyone want sheep for clay ?*" est une Offre, et les EDUs 166 :"*got none*" et 168 :"*no*" sont deux Refus. Les EDUs des conversations concurrentes sont du type Autre. Les statistiques de la répartition des DAs dans les deux corpus sont données Table 3.6.

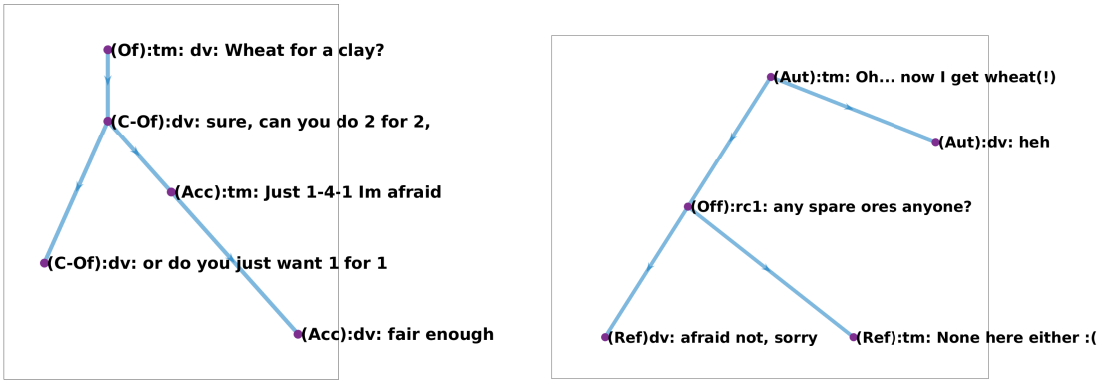
Une autre information disponible dans les annotations est la structure de graphe des tours des joueurs, c'est-à-dire comment les EDUs sont structurées entre elles. Deux exemples de la structure de graphes sur les EDUs d'un tour sont fournis Figure 3.16. Sur cette figure, pour chaque EDU correspondant à un noeud du graphe, le DA est précisé entre parenthèses.

3.3.3 Détection des actes de dialogues (DA)

La reconnaissance des DAs est une tâche fondamentale pour le traitement automatique du langage. En effet, un modèle capable de capturer la structure des séquences de DAs peut, par exemple, être utilisé afin de construire de meilleurs systèmes de génération

	label des DAs	Nombre	%
Spect.	*Autre*	7380	60.88
	Offre	1689	14.01
	Refus	1632	13.46
	C.-Offre	729	6.01
	Accept.	683	5.63
Situ.	*Other*	35331	84.02
	Offre	2791	6.64
	Refus.	1691	4.02
	C.-Offre	729	1.73
	Accept.	1508	3.59

TABLE 3.6 : *Statistiques des actes de dialogues pour les deux corpus.*



(a) (b)

FIGURE 3.16 : *Deux graphes de tours de joueurs et les DAs associés aux EDUs, obtenus via les annotations.*

automatique de dialogues [118, 119], puisque connaître l'acte de dialogue de l'EDU précédente guide la production de l'EDU suivante. En effet, si la dernière EDU est du type QUESTION, il est plus probable que l'EDU suivante soit du type RÉPONSE ou CLARIFICATION-QUESTION que du type SALUTATION.

Dans cette section, nous présentons un réseau de neurones que nous avons mis en place afin de détecter le DA d'une EDU de STAC. Par ailleurs, nous utilisons des méthodes classiques de clustering en sortie de chacune des couches du réseau, afin de valider la structure du-dit réseau en vérifiant que chaque couche permet de mieux séparer les données.

Nous verrons aussi que l'information sur les structures par blocs permet de mettre en

exergue des informations qui ne sont pas visibles si l'on ne s'intéresse qu'à la sortie du réseau. Nous conclurons sur le fait qu'une approche totalement non supervisée, comme l'algorithme spectral présenté au chapitre précédent, permet d'obtenir une information supplémentaire pouvant s'avérer précieuse.

Structure du réseau et traitement des données

La structure du réseau de neurones que nous avons mise en place pour la détection des DAs est donnée Figure 3.17. Ce réseau est composé de cinq couches :

Emb : Cette première couche correspond à la représentation vectorielle via GloVe des mots composant le texte en entrée du réseau. Dans cette première couche, le réseau n'a pas de paramètre à apprendre. Il s'agit simplement de trouver, pour chaque mot du texte en entrée, le vecteur correspondant dans GloVe.

BiLSTM : Le Long Short Term Memory (LSTM) [120] est un type particulier de réseau de neurones récurrent (RNN). En effet, les RNNs classiques permettent en théorie de traiter des données en séquence temporelle. Mais étant donnée la rétropropagation des gradients faite dans la phase d'apprentissage, le poids associé au traitement des premières données de la série devient quasiment nul. Il est donc pratiquement impossible de faire apparaître des dépendances à long terme avec ce type de réseaux. Les LSTMs sont spécifiquement conçus pour gérer ces problèmes de dépendance à long terme, en choisissant, pour chaque nouveau terme de la séquence, à quel point il doit être "oublié" ou non. Le LSTM Bidirectionnel (ou BiLSTM) traite la séquence du début vers la fin puis de la fin vers le début, et concatène les résultats pour les deux directions.

MP : Il s'agit d'une couche de "Maximum Pooling" : dans l'espace de représentation des mots, on conserve sur chaque coordonnée la valeur du mot ayant le poids le plus fort pour cette coordonnée. Cette couche n'a pas de paramètre à apprendre.

FC1,FC2 : Il s'agit de deux couches denses composées de respectivement 150 et 5 neurones, c'est-à-dire 150 – respectivement 5 – formes affines prenant en entrée le vecteur de taille 600 – respectivement 150. Les sorties de chaque neurone sont concaténées et on obtient donc des vecteurs de dimension 150 – respectivement 5. *FC1* est couplée avec un fonction d'activation $ReLU$ définie pour tout x réel par $ReLU(x) = max(0, x)$. *FC2* est la couche de prédiction des classes (5 correspond aux nombres des différents types de DA dans STAC). Elle est couplée avec une fonction classique *softmax*, définie pour tout vecteur $\mathbf{x}$ réel par $f_i(\mathbf{x}) = e^{x_i} / \sum_k e^{x_k}$, et retourne donc la probabilité d'appartenance à chaque classe pour l'EDU en entrée.

Afin de prédire le DA d'une EDU, nous prenons en compte non seulement l'EDU à strictement parler, mais aussi son 4-contexte, c'est-à-dire les 4 EDUs qui la précèdent immédiatement dans le tour. Afin de pouvoir traiter toutes les EDUs avec le même réseau, nous appliquons un padding à toutes nos entrées. Concrètement, nous ajoutons des mots dont le vecteur de représentation sera le vecteur nul au début et à la fin du texte passé en entrée du réseau, afin que toutes les entrées aient la même dimension. Un schéma du prétraitement des EDUs est donné Figure 3.18, où 63 est la taille maximale d'une EDU avec son 4-contexte pour le corpus *Spectateur*. Pour le corpus *Situé*, la taille des EDUs après padding est de 83.

Nous avons utilisé une validation croisée à 10 plis pour observer nos résultats. C'est-à-dire que nous avons divisé chacun de nos corpus en 10 groupes possédant un même nombre d'EDUs. Puis nous avons utilisé chaque groupe comme ensemble de test pour notre réseau, après avoir entraîné ce dernier sur les 9 groupes restants.

Pour entraîner notre réseau, nous avons utilisé l'optimiseur Adam avec les paramètres conseillés dans [121]. Comme fonction perte et comme métrique, nous avons choisi respectivement l'entropie croisée catégorielle et la précision catégorielle. L'entropie croisée catégorielle se définit par

$$-\frac{1}{N}\sum_{i=1}^{N} log(x_{t_i}^{(i)})$$

où N est le nombre de données dans l'ensemble d'entraînement, $\mathbf{x}^{(i)}$ est la sortie du réseau correspondant à la i^{eme} donnée de l'ensemble d'entraînement – ici, $\mathbf{x}^{(i)}$ est donc un vecteur de 5 entrées tel que $\sum_{k=1}^{5} x_k^{(i)} = 1$. Quant à $\mathbf{t} \in \mathbb{N}^N$, il s'agit du vecteur des labels que le réseau cherche à approcher. Autrement dit, $t_i = k$ signifie que la i^{eme} donnée de l'ensemble d'entraînement appartient en réalité à la k^{ieme} classe.

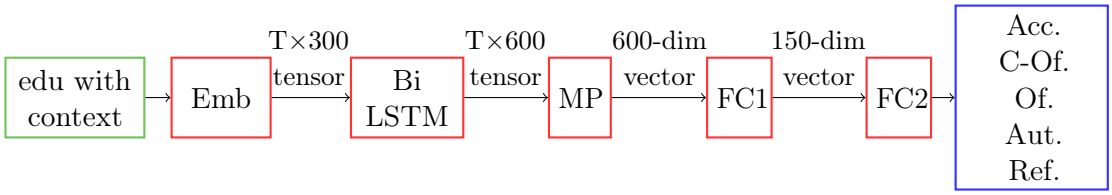


FIGURE 3.17 : *Architecture du réseau de neurones.* $T = 63$ *pour le corpus* Spectateur, et $T = 83$ pour le corpus Situé.

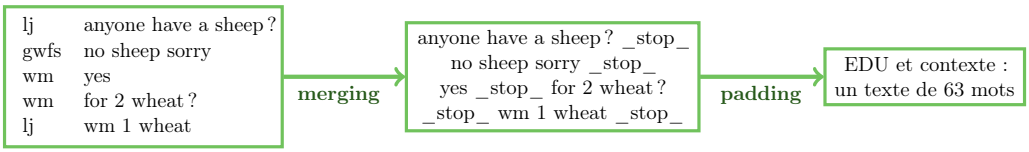


FIGURE 3.18 : *Exemple de prétraitement des EDUs.*

En reprenant les mêmes notations, la précision catégorielle est égale à

$$\frac{1}{N}\sum_{i=1}^{N}\delta(indmax(\mathbf{x}^{(i)}),t_i)$$

où $\forall \mathbf{y} \in \mathbb{R}^p, indmax(\mathbf{y}) = k$ si, $\forall i \in \{1,...,p\} \setminus \{k\}, y_i < y_k$ et $\delta(y,z) = 1$ si $y = z$, 0 sinon.

Nous avons lancé les tests pour 20 époques au maximum, avec des batchs de taille 400. Un batch consiste en un sous-ensemble des données d'apprentissage – ici chaque batch contient environ 400 données. L'ensemble des batchs est une partition de la base d'apprentissage. Lors de l'évaluation des sorties du réseau sur un batch, la perte et la métrique entre le label attendu et le label retourné par le réseau,est calculée pour chaque donnée. Une fois le batch parcouru, le réseau met à jour ses paramètres dans le but de minimiser cette erreur. Une époque consiste à parcourir tous les batchs une fois. Autrement dit, après une époque, l'ensemble des données de la base d'apprentissage ont été testées – chaque donnée ayant été testée une unique fois.

Evaluation

Afin de mesurer les performances de notre réseau, nous avons utilisé deux mesures – à savoir la F-mesure et le pourcentage d'erreur de classification.

F-mesure. Pour mesurer la précision de la classification retournée par rapport à celle attendue, nous utilisons la Précision, le Rappel et la F-mesure – qui est la moyenne harmonique de la précision et du rappel. Ces mesures sont normalement définies pour une classification binaire des données. Mais il est possible de les étendre à plusieurs classes en calculant les scores par classe, puis en faisant la moyenne des scores obtenus.

La Précision d'une classe, notée P, est la fraction du nombre de données correctement affectées à cette classe sur le nombre des données affectées à cette classe. Le Rappel d'une classe, noté R, est la fraction du nombre de données correctement affectées à cette classe sur le cardinal réel de cette classe. La F-mesure, notée F, combine précision et rappel en

calculant leur moyenne harmonique. Ainsi, pour une classe donnée, on a

$$P = \frac{tp}{tp + fp},\ R = \frac{tp}{tp + fn},\ F = 2\frac{P.R}{P + R}$$

où tp, fp et fn représentent respectivement le nombre de vrais positifs, faux positifs et faux négatifs de cette classe. Si tous les éléments de la classe ont été correctement affectés à cette classe, alors $fn = 0$ et $R = 1$. Sinon, $R < 1$. Si aucun élément hors de la classe n'a été affecté par erreur à cette classe, alors $fp = 0$ et $P = 1$. Sinon, $P < 1$. Ainsi, $F = 1$ si la classe est parfaitement bien identifiée et F est d'autant plus proche de 0 que la classe est mal identifiée.

Matrice de confusion. Nous introduisons une évaluation de l'erreur de classification au sens du nombre de données mal affectées dans une classe. En notant m le nombre de données, k le nombre de classes recherchées et $C \in \mathbb{N}^{k,k}$ la-dite *matrice de confusion*, on a $C_{i,j}$ le nombre des données de la classe i affectées à la classe j. Ainsi, $C_{i,i}$ est le nombre des données correctement affectées à la classe i.

Alors, le *pourcentage de données mal classées*, noté p, est défini par :

$$p = \frac{\sum_{i \neq j}^{k} C_{i,j}}{m}$$

Evidemment, on cherche à obtenir la plus petite valeur de p, correspondant au minimum de données mal classées.

Remarque 18. *Nous avons déjà rencontré la matrice de confusion à la Section 1.3, grâce à laquelle nous avons calculé la NMI qui mesure l'écart entre deux partitionnements. Cependant, à la différence du problème de la détection de communautés traité Section 1.3, nous nous intéressons dans cette section à un problème de classification supervisée, c'est-à-dire que nous fixons à l'avance le nombre de classes à obtenir, et que ces classes sont labellisées. La matrice de confusion est donc carrée. L'ordonnancement de ses lignes et de ses colonnes est important, car sa diagonale comporte les entrées bien classées.*

Les Algorithmes de classification non supervisée

Pour étudier l'efficacité de l'architecture proposée pour identifier le DA d'une EDU, nous avons considéré deux techniques de classification non supervisée parmi les plus utilisées, soit l'algorithme des K-moyenne (ou K-means) et l'algorithme de clustering spectral, respectivement basés sur des séparations linéaires et non-linéaires. Cette technique d'utilisation d'algorithmes de classification non supervisée pour valider l'architecture

réseau vient de [122]. Cela permet de vérifier que chaque couche permet une meilleure séparabilité des données par rapport à la couche précédente. En outre, cela peut permettre d'observer l'action du réseau sur les données au fur et à mesure des couches.

Nous rappelons succinctement le principe de ces deux algorithmes, que nous avons déjà rencontrés en Section 2.7. Nous rappelons que pour chacun, le nombre de classes à trouver est à fixer à l'avance.

K-means. Il s'agit d'un des algorithmes de classification non supervisée les plus simples et les plus populaires. Son but est de partitionner un ensemble de données en un petit nombre de classes, en minimisant la distance entre chaque point de l'ensemble des données et le centroïde de la classe à laquelle il appartient, c'est-à-dire la moyenne des éléments constituant la classe. Ces moyennes sont calculées avec les distances de Manhattan [123]. Comme nos ensembles de données tests contiennent environ 1212 EDUs pour le corpus *Spectateur*, et 4205 EDUs pour le corpus *Situé*, l'initialisation des centroïdes est effectuée sur un partitionnement préliminaire de 10% des données.

Clustering Spectral (SC). Ce que nous appelons ici Clustering Spectral est l'algorithme, présenté dans [31], qui sélectionne les vecteurs propres dominants d'une matrice d'affinité Gaussienne afin de construire un espace de représentation spectrale des données de dimension plus faible. Dans cet espace, le partitionnement est effectué via K-means. Cette méthode permet de détecter des classes quelle que soit leur forme avec un choix approprié de la fonction d'affinité du noyau Gaussien. Cette méthode est équivalente à d'autres problèmes d'optimisation tels que le kernel K-means ou les coupes normalisées de graphes [124]. Le paramètre σ de la mesure d'affinité Gaussienne, dont le rôle est de contrôler l'affinité entre les données, est basé à la fois sur la densité et la dimension de l'ensemble des données, suivant ainsi les prescriptions de [125].

Comme ces algorithmes sont non supervisés, nous choisissons de labelliser les classes retournées de sorte à minimiser p le pourcentage de données mal classées, c'est-à-dire que nous permutons les lignes de la matrice de confusion pour maximiser la somme de ses éléments diagonaux.

3.3.4 Procédure de réduction des dimensions

Etant donnée la grande dimension des données – notamment en sortie de la couche *Emb* –, nous avons appliqué l'analyse en composantes principales (ACP) pour réduire

la dimension des variables, en tentant d'extraire une information sur la distribution des données qui soit discriminante pour l'identification des DAs.

L'ACP est une méthode basée sur la décomposition spectrale de la matrice de variance-covariance issue des données. Le but est de trouver une base de l'espace dans lequel les données sont représentées par des variables qui soient décorrélées. Une telle base est obtenue en considérant les vecteurs singuliers de la matrice de représentation des données. De là, on réduit la dimension en ne conservant qu'un certain nombre d'axes principaux de la base correspondant aux valeurs singulières dominantes de la matrice. L'évaluation du pourcentage de l'information qui est conservée est obtenue par le calcul du pourcentage du spectre conservé dans l'espace réduit. Dans nos tests, nous conservons un nombre de composantes principales suffisant pour que 98% de l'information – ou contraste – soit préservé. Pour le corpus *Spectateur* – respectivement *Situé*, cela revient à conserver en moyenne 553 composantes – respectivement 836 – en sortie de *Emb*, 284 – respectivement 314 – en sortie de *BiLSTM*, 25 – respectivement 18 – en sortie de *FC1*, et 4 en sortie de *FC2* pour les deux corpus.

Analyse des résultats

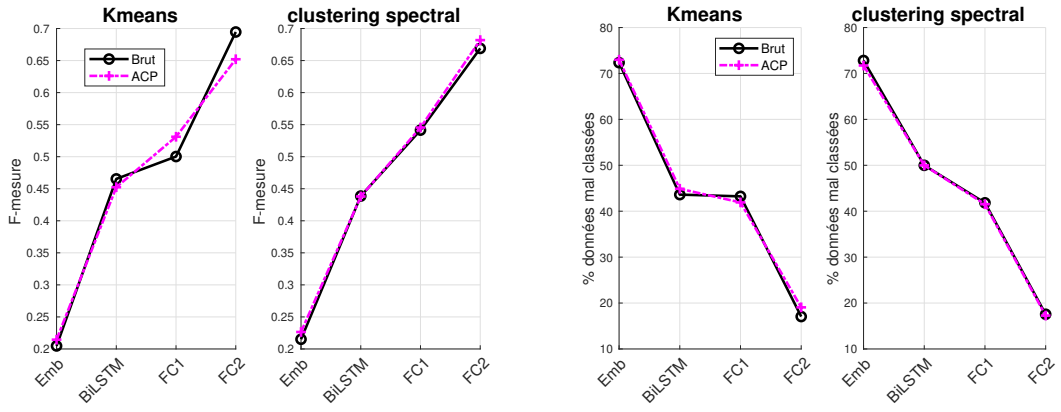
Dans un premier temps, afin de s'assurer que la profondeur de notre réseau est suffisante, nous avons réalisé des tests avec des variantes du réseau en ajoutant des couches denses. La Table 3.7 résume les différents scores de F-mesure pour les différentes versions du réseau. Cette approche montre que deux couches denses suffisent pour une classification précise des DAs, tout en évitant d'avoir une complexité trop grande pour le réseau. En effet, chaque couche dense de taille k dont l'entrée est de taille p nécessite l'apprentissage de $p \times k + k$ paramètres, puisque chacun des k neurones de la couche est une forme affine. Ainsi, plus les réseaux sont profonds avec des couches denses de grande taille, plus grande est la complexité du réseau. Pour autant, les résultats sont très similaires, voire légèrement meilleurs, dans le cas du réseau le moins profond. Ainsi, la profondeur de notre réseau est satisfaisante.

De plus, pour valider l'architecture de notre réseau, nous utilisons les algorithmes SC et Kmeans à la sortie de chaque couche. Les résultats sont fournis Figure 3.19(a) pour le corpus *Spectateur*, et Figure 3.19(b) pour le corpus *Situé*. Il est clair via ces figures que chaque couche du réseau apporte une information non négligeable sur le DA des EDUs. Sur ces figures, on remarque aussi que le comportement des algorithmes de classification non supervisée sur les données après ACP est très proche de celui sur les données brutes. On remarque en outre que pour le corpus *Situé*, les performances décroissent ou sont moins nettes pour la couche *BiLSTM* et *FC1* avec Kmeans, mais pas avec SC, ce qui

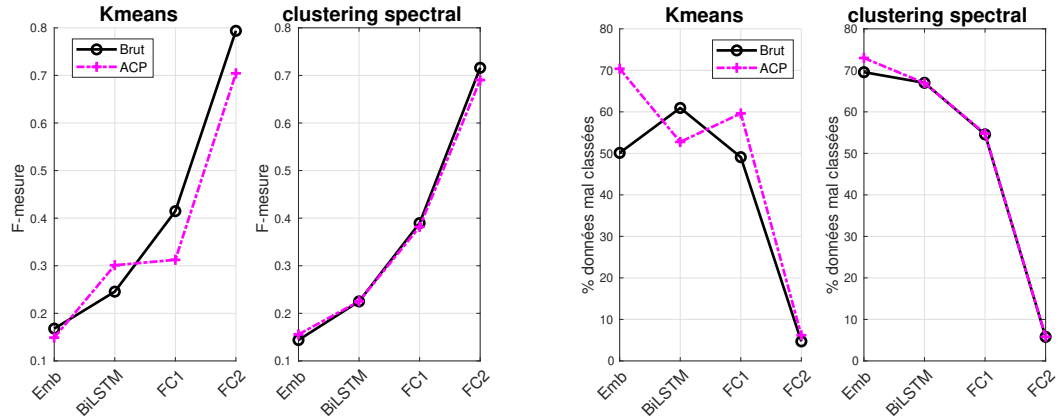
Architecture	Autre	Offer	Contre-offre	Accept.	Refus	**Sur l'ensemble**
150 - 5	**0.92**	**0.76**	0.43	**0.62**	**0.82**	**0.71**
300 - 150 - 5	**0.92**	0.74	**0.44**	**0.62**	**0.82**	0.708
300 - 150 - 50 - 5	**0.92**	**0.76**	0.40	0.61	0.81	0.706
300 - 150 - 75- 30 - 5	**0.92**	**0.76**	**0.44**	0.61	0.80	0.706

TABLE 3.7 : *Scores de F-mesure de l'estimateur du maximum de vraisemblance appliqué à la sortie du réseau pour différents nombres de couches denses.*

signifie que les données des différentes classes sont effectivement mieux séparées, mais pas de façon linéaire.



(a)



(b)

FIGURE 3.19 : *F-mesure (à gauche) et pourcentage des EDUs mal affectées (à droite) des algorithmes de classification non supervisée appliqués en sortie de chaque couche. (a) : Pour le corpus Spectateur. (b) : Pour le corpus Situé.*

Dans la Table 3.8, on montre la valeur moyenne des scores de F-mesure sur les 10 groupes de données, obtenue en assignant à chaque EDU son DA le plus probable étant donnée la sortie du réseau (ligne "Max. de Vrais." de la table). Nous avons comparé ces résultats avec la valeur moyenne des F-mesures obtenues avec Kmeans et SC appliqués respectivement sur les données brutes et les données après ACP pour le corpus *Spectateur*, et appliqués tous les deux sur les données brutes pour le corpus *Situé*, puisque ce sont les situations qui fournissent leur partitionnement le plus précis. Comme test de référence, nous avons appliqué une régression logistique sur la sortie *Emb* du réseau. Pour ce faire, nous avons utilisé la classe `LogisticRegression` de la bibliothèque `scikit-learn` de python [90], avec un solveur saga et une perte multinomiale, en laissant les autres paramètres avec leur valeur par défaut. On observe que les résultats de cette méthode sont sensiblement moins bons que ceux obtenus avec le réseau, par maximum de vraisemblance ou via les algorithmes de classification non supervisée. En effet, si l'on regarde la F-mesure moyenne, on observe pour le corpus *Spectateur*, on a un écart d'au plus 2% entre la classification obtenue par maximum de vraisemblance – qui est la meilleure configuration – et la classification obtenue via les algorithme classification non supervisée, tandis que cet écart est de 26% avec la classification obtenue par régression logistique. Pour le corpus *Situated*, la différence est encore plus flagrante, avec un écart de 39% entre la classification obtenue par maximum de vraisemblance et celle retournée par la régression logistique. Nous notons que sur ce corpus, l'écart se creuse entre les deux méthodes de classification non supervisée : la méthode Kmeans est au même niveau que le maximum de vraisemblance, et l'écart de performance avec SC est de 7%, alors qu'il n'est que de 1% pour le corpus *Spectateur*.

Conclusions et perspectives

En conclusion, notre réseau permet effectivement de détecter avec précision le DA d'une EDU. Cependant, si l'on regarde la Table 3.6, on voit que la classe "Autre" est sur-représentée. On pourrait souhaiter obtenir plus de DAs en subdivisant cette classe, notamment pour le corpus *Situé* dans lequel de nombreux messages formatés sont retournés par le serveur, et peuvent concerner l'attribution des ressources, le résultat des lancers de dés, les constructions de colonies, routes, ou villes faites par les joueurs... On observe en outre que ce sont des groupes d'EDUs qui sont visibles dans la structure par blocs des matrices d'affinités, et ce jusqu'à la sortie *FC1* du réseau. Un exemple est donné Figure 3.20, où la matrice d'affinité en sortie de la couche *FC1* pour l'un des dix groupes tests du corpus *Situé* est affichée. On a demandé à l'algorithme SC de trouver non pas 5 mais 10 classes. Sur cette figure, il est clair que la structure de la matrice contient plus

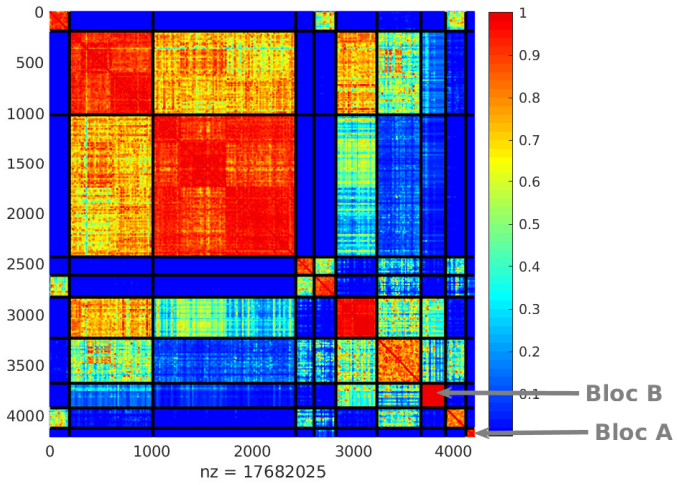
	Méthodes	Autre	Offre	C.-offre	Accept.	Refus	**Moyenne**
corpus *Spectateur*	Max. de Vrais.	**0.92**	**0.76**	0.43	**0.62**	**0.82**	**0.71**
	Kmeans données brutes	**0.92**	**0.76**	0.39	0.59	**0.82**	0.69
	SC après ACP	**0.92**	0.75	**0.44**	0.57	0.73	0.68
	Régression Log.	0.81	0.53	0.30	0.10	0.51	0.45
corpus *Situé*	Max. de Vrais.	**0.98**	**0.84**	0.45	0.83	**0.82**	**0.79**
	Kmeans données brutes	**0.98**	**0.84**	**0.48**	**0.84**	**0.82**	**0.79**
	SC données brutes	**0.98**	**0.84**	0.43	**0.84**	0.51	0.72
	Régression Log.	0.84	0.49	0.02	0.40	0.27	0.40

TABLE 3.8 : *Score de F-mesure sur K-means, SC et un estimateur du maximum de vraisemblance sur la couche en sortie du réseau, et d'un classifieur de régression logistique sur la sortie Emb.*

que cinq blocs. En observant la constitution de ces classes, certains blocs – comme c'est le cas pour les blocs A et B de cette matrice, cf les images (b) et (c) – se limitent à ces messages formatés, et on pourrait souhaiter conserver ces classes telles quelles.

Ainsi, il serait intéressant d'étudier en profondeur les classes retournées par un algorithme totalement non supervisé (auquel on ne fixe pas un nombre prédéfini de classes) comme c'est le cas pour notre algorithme spectral présenté au chapitre précédent, afin de prendre en compte d'autres types de DAs dans STAC. Un exemple de l'application de cet algorithme à une version sparsifiée de la matrice d'affinité de la Figure 3.20(a), est donnée Figure 3.21. Concrètement, nous n'avons gardé de la matrice de la Figure 3.21 que les entrées supérieures à 0.8. Encore une fois, la matrice possède clairement une structure par blocs, qui fait apparaître plus que cinq classes. En observant la composition des blocs en Figure 3.22, on retrouve à nouveau des blocs contenant essentiellement un type de message formaté, comme les blocs A, B et F. Certains blocs peuvent contenir plusieurs types de messages formatés – c'est le cas du bloc D. Mais on observe aussi que certains blocs sont un mélange de messages formatés et d'échanges entre les utilisateurs, comme dans le bloc C : ici, toutes les EDUs sont du type Offre, et le réseau a donc fait en sorte que ces EDUs soient regroupées, bien qu'elles soient structurellement très différentes. En outre, on remarque via le bloc E que les EDUs de la classe Contre-Offre – labellisée par 2 – sont indissociées de celles d'un sous-groupe de la classe Offre – labellisée par 1 – dans la structure par bloc de cette matrice.

Et en effet, le réseau, en ayant pour objectif 5 types de DAs différents, cherche à fusionner des EDUs qui devraient structurellement appartenir à des classes différentes.



(a)

```
ariachiba_traded_1_sheep_for_1_ore_from_Kittles.
ariachiba_traded_4_sheep_for_1_wood_from_the_bank.
inca_traded_1_wheat_for_1_wood_from_Rainbow.
ariachiba_traded_1_wheat_for_1_wood_from_Rainbow.
Kittles_traded_1_ore_for_1_wheat_from_ariachiba.
Rainbow_traded_1_wheat_for_1_ore_from_Kittles.
Rainbow_traded_1_wood_for_1_sheep_from_ariachiba.
inca_traded_1_wood_for_1_sheep_from_ariachiba.
Rainbow_traded_1_wheat_for_1_sheep_from_ariachiba.
Rainbow_traded_1_clay_for_1_wood_from_Kittles.
Charlotte_traded_1_sheep_for_1_ore_from_Josephine.
Amanda_traded_1_sheep_for_1_ore_from_Josephine.
Amanda_traded_1_sheep_for_1_clay_from_Josephine.
Amanda_traded_1_sheep_for_1_wood_from_Charlotte.
Amanda_traded_4_sheep_for_1_ore_from_the_bank.
Katherine_traded_4_wood_for_1_clay_from_the_bank.
Josephine_traded_1_wood_for_1_sheep_from_Katherine.
Charlotte_traded_1_wood_for_1_sheep_from_Katherine.
Amanda_traded_4_wheat_for_1_clay_from_the_bank.
Charlotte_traded_1_clay_for_1_ore_from_Josephine.
```

(b)

```
It's_william's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_william's_turn_to_roll_the_dice.
It's_tomas.kostan's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
It's_ljaybrad123's_turn_to_roll_the_dice.
```

(c)

FIGURE 3.20 : *(a) : Matrice affinité en sortie de FC1 et sa structure par blocs retournée par l'algorithme SC pour 10 classes. (b) : Quelques exemples des EDUs du bloc A de la Figure (a). (c) : Quelques exemples des EDUs du bloc B de la Figure (a).*

Cela peut se faire au détriment d'autres classes qui sont *a priori* plus dures à séparer– comme par exemple la classe Offre et la classe Contre-Offre–et sous-représentées – comme c'est le cas pour la classe Contre-Offre, qui ne constitue que 1.75% du corpus *Situé*. En effet, en observant dans la Figure 3.23 la projection des représentations des EDUs en sortie du réseau sur leur trois composantes principales, on observe que, sur les angles deux vues en 3D de cette figure, les EDUs du type Offre – en bleu – et du type Contre-Offre – en vert – se situent exactement le long de la même arête de la pyramide. Cela permet en

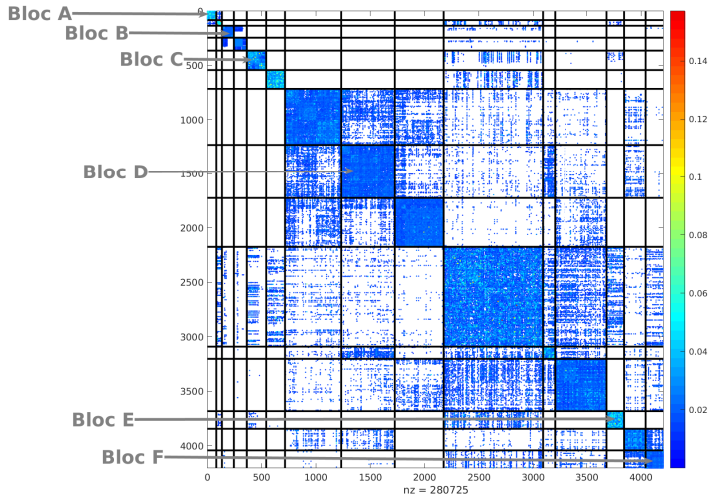


FIGURE 3.21 : *Matrice affinité en sortie de FC1 après sparsification et équilibrage doublement stochastique, et sa structure par blocs retournée par l'algorithme spectral décrit au Chapitre 2.*

outre de comprendre pourquoi la classe Contre-Offre a le plus faible score de F-mesure en Table 3.8.

Pour cette raison, nous souhaiterions investiguer le potentiel d'une approche semi-supervisée sur ce modèle. Il serait en effet très intéressant d'observer les résultats d'un réseau possédant plus de 5 classes en sortie, et de choisir quels sont les types de label à favoriser. Par exemple, on pourrait fournir les labels de toutes les EDUs de type Offre et Contre-Offre lorsqu'il s'agit d'interventions de joueurs – et non de messages du serveur –, pour améliorer la séparation entre ces deux classes, et ne pas fournir de label pour les messages formatés, laissant ainsi plus de liberté au réseau pour partitionner ces messages en plusieurs classes.

3.4 Conclusion : d'autres types de structures

Dans ce chapitre, nous nous sommes intéressés à quelques applications de la découverte de structures par blocs dans les matrices d'adjacence, les systèmes linéaires et les matrices d'affinités. Cependant, d'autres types de structures peuvent être recherchées. Nous donnons ici un exemple concret avec une expérience que nous avons menée, à nouveau pour une application dans le domaine du traitement automatique du langage naturel, sur la base de données STAC.

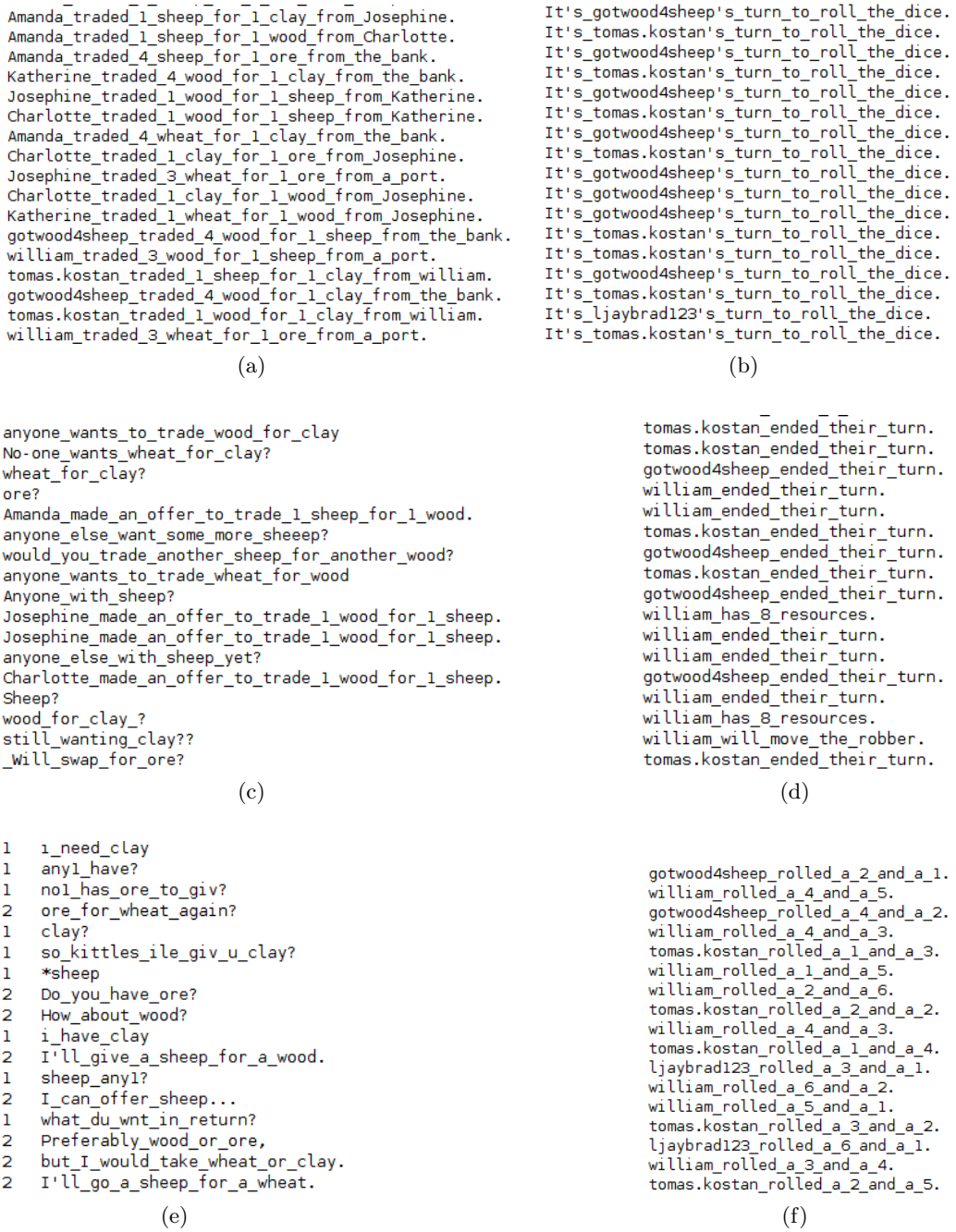


FIGURE 3.22 : *Quelques exemples des EDUs dans les blocs indiqués sur la Figure 3.21. (a) : Bloc A. Contient les messages formatés concernant les échanges. (b) : Bloc B. Contient les messages formatés indiquant le joueur dont c'est le tour. (c) : Bloc C. Contient essentiellement des EDUs du type "Offre". Mélange de messages formatés ("Josephine made an offer to trade 1 wood for 1 sheep") et d'échanges écrits par les utilisateurs. (d) : Bloc D. Contient plusieurs types de messages formatés : indiquant la fin du tour d'un joueur, indiquant les ressources d'un joueur, indiquant le changement de case du pion "robber". (e) : Bloc E. Contient les EDUs du type "Contre-Offre" (indiquées par le label 2) et un certain nombre d'EDUs du type "Offre" (label 1). (f) : Bloc F. Contient les messages formatés indiquant le résultat d'un lancer de dés.*

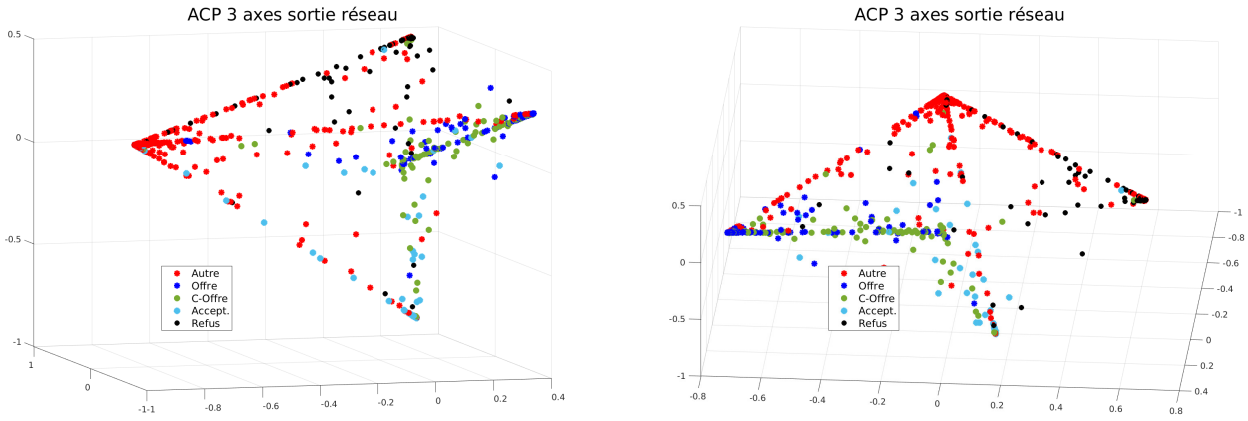


FIGURE 3.23 : *Deux angles de vues différents de la projection sur les trois axes principaux de la sortie du réseau de neurones pour un groupe test du corpus* Situé.

En effet, un autre type de problématique que souhaitent résoudre les chercheurs travaillant sur STAC [126, 127] est la détection des structures de graphe des discours, comme celles présentées à la Figure 3.16. Nous avons tenté d'investiguer cette voie en utilisant le réseau de neurones proposé dans [6], implémenté spécifiquement pour cette problématique sur cette base de données, et dont la structure est présentée Figure 3.24, en reprenant les notations de la Figure 3.17.

Comme pour la détection des DAs, nous avons utilisé ce réseau dans le but de ramener cette problématique à la détection d'une structure par blocs d'une représentation matricielle de ces couples d'EDUs.

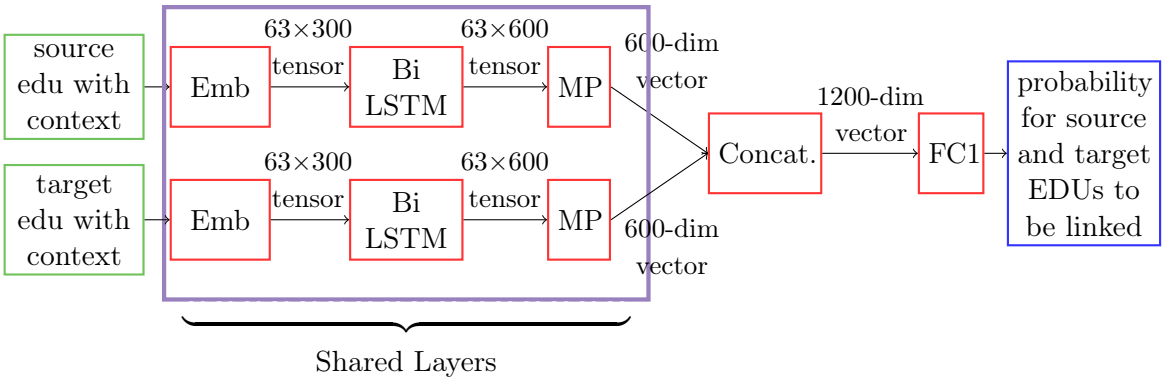


FIGURE 3.24 : *Architecture du réseau de neurones proposé dans [6].*

Nous avons donc considéré les couples d'EDUs comme nos données. Nous avons créé la matrice d'affinité gaussienne de ces données, sur laquelle nous avons appliqué les deux algorithmes spectraux évoqués précédemment – SC et l'algorithme développé au Chapitre 2. Mais les structures par blocs obtenues ne véhiculaient pas l'information sur la liaison éventuelle entre deux EDUs. Nous avons tenté cette approche en ajoutant des couches denses entre la concaténation et la sortie du réseau, mais à nouveau, la structure par blocs de la matrice ne véhiculait pas l'information sur les liaisons potentielles entre les EDUs. Afin d'ajouter un degré de liberté, nous avons travaillé directement sur la matrice des données, construite sur la sortie des diverses couches du réseau, en utilisant l'algorithme de co-clustering détaillé dans [12], qui se base sur un équilibrage doublement stochastique de la matrice – plus précisément de ses équations normales. A nouveau, les résultats obtenus ne permettaient pas d'approcher la structure que l'on souhaitait trouver.

Nous avons fini par comprendre que notre façon d'envisager le problème n'était pas correcte. En effet, la détection de la structure de graphe d'un discours est un sujet particulièrement complexe en ce qui concerne les contraintes à fixer quant à la structure recherchée. Certains modèles – ceux basés sur la théorie de la structure rhétorique notamment, cf [128] – considèrent que deux unités de discours ne peuvent être liées que si elles sont adjacentes – c'est-à-dire que l'une précède l'autre. La contrainte de la frontière à droite – cf par exemple [129] – contraint fortement les relations entre les unités de discours : une relation (dirigée) dans le discours doit pointer d'une unité de discours vers une unité qui la suit, ou vers une unité qui contient une unité qui la suit. Ces contraintes seront fondamentalement différentes en fonction du problème étudié (s'il s'agit d'un monologue ou d'un dialogue, s'il s'agit d'un dialogue à deux participants ou plus de deux participants, etc.), et en fonction du modèle choisi.

Parmi les deux articles [126, 127] dans lesquels les auteurs cherchent les structures de discours dans STAC, les auteurs commencent par calculer une probabilité locale d'attachement entre deux EDUs – c'est aussi ce que nous donne la sortie du réseau de neurones de la Figure 3.24. Une fois ces probabilités locales obtenues pour tous les couples d'EDUs, plutôt que d'utiliser un classifieur de type maximum de vraisemblance ou algorithme de classification non supervisée, comme nous l'avons fait Section 3.3.3, les auteurs font appel à un décodeur imposant des contraintes globales sur les attachements entre EDUs. Deux types de décodeurs – ou post-traitement – différents sont utilisés. Dans [126], les auteurs imposent à la structure de discours dont les noeuds sont les CDUs d'être un arbre. Le graphe restreint à chaque CDU contenant plus d'une EDU doit aussi être un arbre. Si plusieurs CDUs/EDUs sont exprimées à la suite par un même joueur, les attachements peuvent pointer d'une EDU vers une EDU la précédant

– c'est-à-dire que l'on suppose que le joueur peut évoquer quelque chose qu'il n'a pas encore dit, mais qu'il a l'intention d'exprimer plus tard. Ce type de dépendance est interdit entre les EDUs de différents joueurs ou entre les EDUs d'un même joueur séparées par l'intervention d'un autre joueur. Les auteurs appellent ces contraintes "contraintes inter-tour" et "contraintes intra-tour'. Un tour correspond en effet aux interventions d'un même joueur, non séparées par l'intervention d'un autre joueur. Ils obtiennent plusieurs sous-graphes pour un même discours, sur lesquels ils recherchent des arbres couvrants de poids maximum comme cela se fait souvent pour détecter la structure de dépendance d'une phrase – cf par exemple [130] –, arbres dans lesquels ils interdisent donc certaines arêtes. Plusieurs algorithmes de détection de d'arbres couvrants de poids maximum existent. Les auteurs justifient le choix de l'algorithme de Chu-Liu-Edmonds [131, 132] pour trouver leur arbre par le fait que la structure de discours dans STAC n'est pas nécessairement projective. Pour faire simple, deux CDUs peuvent être liées même si elles ne sont pas adjacentes, impliquant ainsi que les arêtes du graphe de la structure de discours peuvent se croiser. L'algorithme de Chu-Liu-Edmonds permet de trouver ce type de structure, ce qui n'est pas le cas de tous les algorithmes [130].

Dans [127], les auteurs conservent ces contraintes inter- et intra-tour, mais recherchent des graphes dirigés acycliques au lieu d'arbres. Un graphe dirigé est dit acyclique s'il ne possède pas de circuit, c'est-à-dire que l'on ne peut pas partir et revenir sur un même noeud si l'on suit les arêtes en respectant leur orientation. On autorise donc plus d'arêtes que dans un arbre, un arbre étant un graphe ne possédant pas de chaîne, c'est-à-dire que l'on ne peut pas partir et revenir sur un même noeud si l'on suit les arêtes quelle que soit leur orientation. Pour ce faire, les auteurs utilisent des outils venant de la programmation linéaire en nombre entier, bien plus complexes que la découverte d'un arbre couvrant de poids maximum. Ils justifient leur approche par le fait que des structures "non-arbres" existent dans STAC, comme le montre la Figure 3.25 extraite de [126], dans laquelle la structure représentée correspond effectivement à un graphe dirigé acyclique, mais pas à un arbre. Ainsi, avec l'approche par arbre, 9% des liens du corpus seront de toute façon non détectés.

Toutes les considérations que nous avons énoncées ci-dessus ont pour but de montrer que la structure recherchée par un algorithme de classification, supervisée ou non, peut être non triviale, être complexe à évaluer et à mettre en place, et fortement impacter les résultats. Dans notre approche, nous avons cherché une structure par blocs pour résoudre le problème de la structure de discours. Cette méthodologie est proche de la recherche du maximum de vraisemblance, et de l'approche appelée LOCAL dans [126, 127], dont il est montré dans chacun de ces articles qu'elle est sensiblement moins efficace que la

234	gw4s	anyone got wheat for a sheep ?
235	inca	sorry, not me
236	ccg	nope, you seem to have lots of sheep !
237	gw4s	yup baaa
238	dmm	I think i'd rather hang on to my wheat i'm afraid
239	gw4s	kk I'll take my chances then...

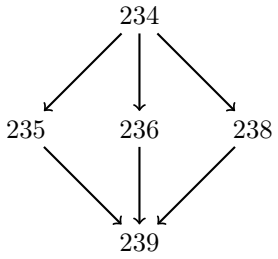


FIGURE 3.25 : *Exemple d'un dialogue dans STAC qui n'est pas un arbre. L'allocution 234 correspond à une offre – sous forme de question – de gw4s à tous les participants. Les joueurs inca, ccg et dmm répondent à sa question par un refus via les allocutions 235, 236 et 238. gw4s accuse ensuite réception de ces trois refus via l'allocution 239.*

recherche de structures prenant en considération des contraintes réelles.

En recherchant une structure par blocs, nous nous ramenons à rechercher une structure en plusieurs graphes complets et deux à deux disjoints – pour le co-clustering, on recherche des graphes bipartites complets et deux à deux disjoints – qui soit une bonne approximation de nos données initiales. Ce qui est fondamentalement différent, voire antinomique, avec la recherche d'un arbre ou d'un graphe dirigé acyclique.

En conclusion de ce manuscrit, nous allons illustré des cas dans lesquels l'équilibrage doublement stochastique ne doit pas être appliqué sans une réflexion préalable sur la structure des données que l'on souhaite mettre en évidence. Nous aurons en outre une discussion préliminaire sur la robustesse de cet équilibrage, en fonction des structures.

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à la détection de structures par blocs dans les matrices carrées, et à plusieurs de ses applications : la détection de communautés dans les réseaux, où la structure à récupérer ne concerne que la structure creuse de la matrice d'adjacence du réseau ; les préconditionneurs de type Bloc Jacobi, où la structure à détecter concerne à la fois la structure creuse et les valeurs numériques du système linéaire ; et la détection des actes de dialogues dans un discours, où nous avons travaillé sur la matrice d'affinité des représentations vectorielles des allocutions, qui est une matrice dense.

Nous avons observé que l'équilibrage doublement stochastique des matrices permet en général d'améliorer la détectabilité de la-dite structure. En particulier, le Chapitre 1 nous a permis de conclure que, même dans le cas où la structure par blocs à détecter n'est initialement liée qu'à la structure creuse de la matrice, l'équilibrage doublement stochastique permet d'accentuer cette structure et d'améliorer sa détection. En outre, le fait de travailler sur les matrices après équilibrage permet d'unifier un certain nombre de mesures conçues pour évaluer la cohérence de cette structure.

Au Chapitre 2, nous avons développé un algorithme spectral de détection de structures par blocs, travaillant sur l'équilibrage doublement stochastique de la matrice en paramètre. Cet algorithme se base sur la propriété qu'ont les vecteurs singuliers d'une matrice doublement stochastique d'être constants par morceaux si cette matrice présente une structure par blocs de lignes et/ou de colonnes. Nous avons adapté des outils de traitement du signal à notre problématique, afin de récupérer l'intégralité de la structure par blocs disponible sur un même vecteur. Cela nous a permis de développer un algorithme fonctionnant sans aucune information *a priori* sur le nombre de blocs, leur taille ou leur composition.

Cependant, étant donnés les résultats de cet algorithme, et notamment sa capacité à détecter indépendamment les blocs de lignes et de colonnes, la restriction de notre algorithme aux matrices carrées nous semble une limitation importante. En effet, comme

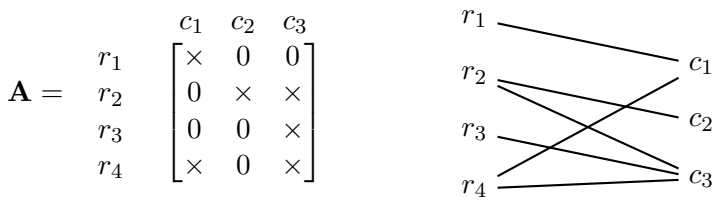


FIGURE 3.26 : *A gauche : une matrice rectangulaire. A droite : le graphe bipartite associé.*

cela est précisé en conclusion du Chapitre 2, cet algorithme paraît *a priori* tout indiqué pour une application de co-clustering. Mais cette application nécessite de pouvoir traiter les matrices rectangulaires. En effet, les matrices sur lesquelles est appliqué le co-clustering sont généralement des tables de données ou des tables de contingences [13, 14, 133, 134], comme nous l'avons illustré dans l'introduction de cette thèse.

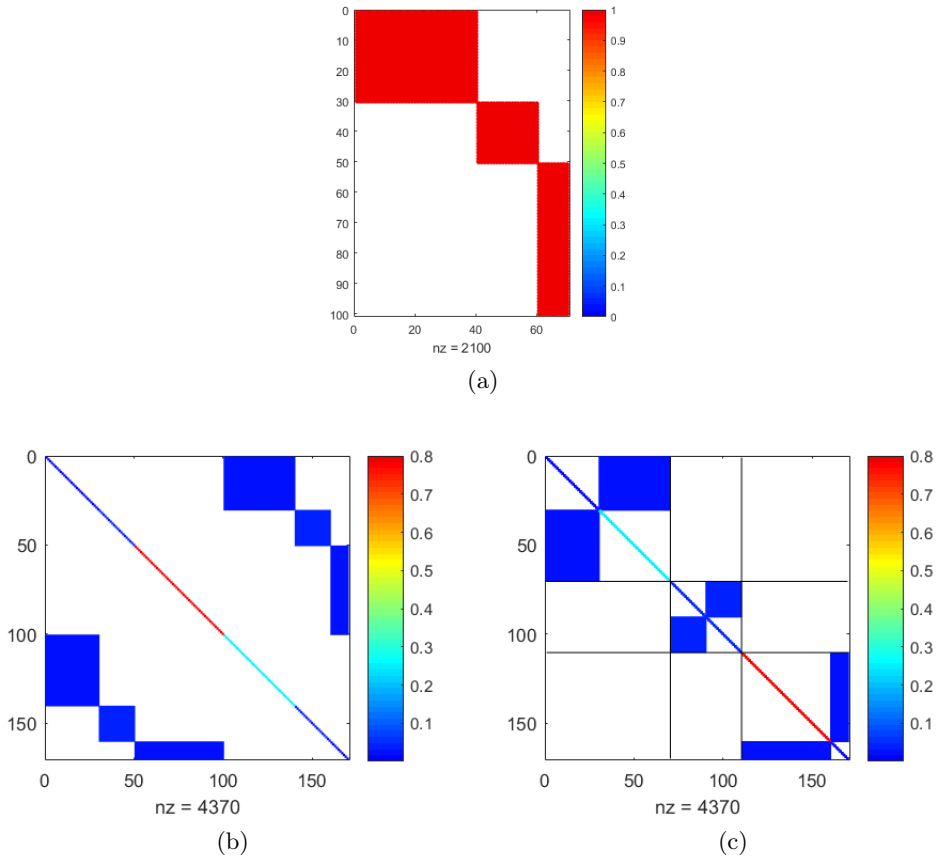
Les perspectives de cette thèse concernent donc la généralisation de nos travaux au cas des matrices rectangulaires. Dans des travaux préliminaires, nous avons investigué cette voie en considérant que les matrices rectangulaires peuvent être représentées par des graphes bipartites, comme le montre l'exemple de la Figure 3.26. Dans ce graphe bipartite, chaque ligne de la matrice correspond à un noeud, de même pour ses colonnes. Les arêtes entre deux noeuds lignes (respectivement deux noeuds colonnes) sont interdites, et on peut donc condenser la matrice d'adjacence en une matrice rectangulaire.

Il existe de nombreux dérivés des graphes bipartites. On peut penser par exemple aux structures coeur-périphérie [135], classiques dans les réseaux sociaux [136], et qui peuvent être envisagées comme des cas particuliers de graphes bipartites. Plus généralement, de nombreuses classes de réseaux sociaux ont une structure quasi bipartite, comme les réseaux de relations sexuelles, où la plupart des arêtes (mais pas toutes) lient des hommes et des femmes, les réseaux de marchés, où l'ensemble des acheteurs et des vendeurs sont à peu près dissociés, mais pas exactement, etc [137].

L'idée de considérer le co-clustering comme le partitionnement d'un graphe bipartite est relativement classique [33, 138]. Notre idée a donc été de travailler sur la matrice d'adjacence du graphe bipartite associé à la matrice rectangulaire initiale, car la matrice d'adjacence d'un graphe est toujours carrée. Soit $\mathbf{A} \in \mathbb{R}^{m \times n}$ une matrice rectangulaire. La matrice d'adjacence du graphe bipartite associé à $\mathbf{A}$ est la matrice

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}.$$

Notre but serait donc de trouver une structure diagonale par blocs sur cette matrice $\widetilde{\mathbf{A}}$



(a)

(b) (c)

FIGURE 3.27 : *(a) : Une matrice* $\mathbf{A}$ *présentant une structure à trois blocs parfaits. (b) : Une matrice augmentée de* $\mathbf{A}$*. (c) : Même matrice après permutations symétriques des lignes et des colonnes de sorte à mettre en évidence ses blocs bi-irréductibles.*

par permutations symétriques des lignes et des colonnes. Afin de pouvoir appliquer un équilibrage doublement stochastique à cette matrice tout en conservant sa symétrie, nous avons en réalité travaillé sur $\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \mathbf{I}$, qui est à support total par nature (matrice symétrique à diagonale pleine [30]). Cette propriété est illustrée Figure 3.27, où l'on voit comment réorganiser $\widetilde{\mathbf{A}}$ de sorte à la mettre sous forme d'une matrice à blocs diagonaux bi-irréductibles, qui est donc à support total.

Nous nous sommes d'abord intéressés au cas où la matrice $\mathbf{A}$ a une structure par blocs idéale. C'est-à-dire que l'on peut trouver des permutations indépendantes des lignes

et des colonnes de $\mathbf{A}$, permettant d'obtenir une matrice de la forme :

$$\begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix}$$

avec $\forall i \in \{1, ..., k\}, \mathbf{A}_i = \mu_i \mathbf{e}_{m_i} \mathbf{e}_{n_i}^T$, *id est* les blocs $\mathbf{A}_i$ sont des blocs rectangulaires composés d'une unique valeur numérique μ_i. De telles matrices sont illustrées dans les graphes (a) des Figures 3.27 et 3.28. Dans ce cas, nous avons démontré que l'équilibrage doublement stochastique $\overline{\mathbf{A}}$ de la matrice augmentée $\widetilde{\mathbf{A}}$ de $\mathbf{A}$ permet d'observer sa structure par blocs via ses vecteurs propres dominants. En Annexe C.1, nous établissons notamment que dans ce cas précis, les vecteurs propres de $\overline{\mathbf{A}}$ associés à 1 présentent une structure constante par morceaux permettant de mettre en évidence la structure de $\mathbf{A}$. Ces résultats sont illustrés sur la Figure 3.28.

En pratique, cette structure est toujours parfaitement détectable si l'on applique à la matrice $\mathbf{A}$ une perturbation qui n'impacte que ses valeurs numériques, comme le montre la Figure 3.29.

Cependant, nous avons observé qu'un faible bruitage de la structure creuse de la matrice idéale impacte fortement les vecteurs propres dominants de l'équilibrage doublement stochastique de sa matrice augmentée, ne permettant plus de détecter la structure par blocs à l'aide de ces vecteurs. Cela est illustré Figure 3.30. En effet, parmi les vecteurs propres associés aux plus grandes valeurs propres, il n'y a plus qu'un unique vecteur qui transmet de l'information sur la structure par blocs de $\mathbf{A}$: il s'agit du second vecteur. Le premier vecteur est constant, et le troisième ne transmet pas cette information, ce qui peut se deviner au regard du zoom sur les valeurs propres dominantes de $\overline{\mathbf{A}}$, sur le graphe (d), où l'on voit qu'il y a un fort décrochage entre la deuxième et la troisième valeurs propres. Par ailleurs, on voit sur le graphe (e) que ce second vecteur propre est très impacté par la perturbation, qui est pourtant très faible, comme on peut le remarquer sur le graphe (a).

Avec ces observations préliminaires, il semble clair que l'équilibrage doublement stochastique ne permet pas d'accentuer la détectabilité de tous les types de structures. En effet, nous avons déjà vu en conclusion du Chapitre 3 qu'il ne permettait pas de détecter des structures de type DAG. Nous précisons que la représentation matricielle d'un DAG n'est pas une matrice irréductible puisque qu'un DAG n'est pas fortement connexe par définition (cf la définition 5), ainsi notre astuce consistant à rendre la matrice équilibrable en remplissant sa diagonale ne fonctionne pas pour ce type de structure. Nous venons

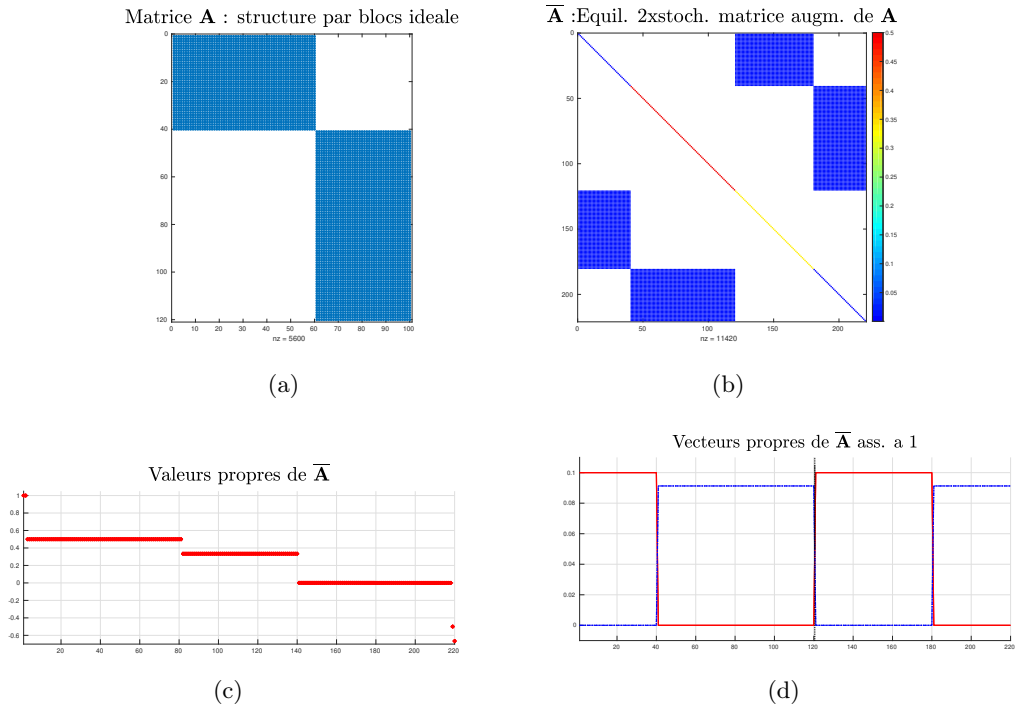


FIGURE 3.28 : *(a) : Une matrice rectangulaire* $\mathbf{A}$ *ayant une structure par blocs idéale. (b) :* $\overline{\mathbf{A}}$ *l'équilibrage doublement stochastique de la matrice augmentée construite sur* $\mathbf{A}$*. (c) : Répartition des valeurs propres de* $\overline{\mathbf{A}}$ *; les deux premières valent 1. (d) : Les deux vecteurs propres de* $\overline{\mathbf{A}}$ *associés à 1 permettent de faire apparaître la structure par blocs de lignes et de colonnes de* $\mathbf{A}$*.*

en outre d'observer que, sur les structures bipartites, le comportement de l'équilibrage doublement stochastique est imprévisible.

Ainsi, trois points nous sembleraient très intéressants à poursuivre :

— des résultats préliminaires concernant l'impact d'une petite perturbation d'une matrice sur son équilibrage doublement stochastique sont fournis au Théorème 3 dans [139]. Cette étude de la sensibilité de l'équilibrage en fonction de la structure de la matrice devrait être approfondie. Notamment, l'impact d'une petite perturbation d'une matrice par blocs sur l'aspect constant par morceaux des vecteurs singuliers dominants de son équilibrage doublement stochastique, en fonction du type de structure, permettrait de donner un cadre formel à l'utilisation de cette approche.

— Dans l'algorithme présenté au Chapitre 2, seule la phase d'équilibrage nécessite que la matrice soit carrée. Toutes les autres étapes (décomposition en valeurs singulières, outils de traitement du signal, projection dans l'orthogonal de la structure par

blocs, et processus de fusion) peuvent être aussi bien appliquées à des matrices rectangulaires. Il serait intéressant d'explorer des équilibrages opérationnels pour les matrices rectangulaires, et d'observer dans quelle mesure ils permettraient de récupérer une structure constante par morceaux sur les vecteurs singuliers dominants de la matrice après équilibrage. *A priori*, les équilibrages à somme déterminée sur les lignes et les colonnes nous semblent tout indiqué – cf [140] par exemple.

— Enfin, que ce soit en considérant une matrice rectangulaire représentant un graphe bipartite, ou une matrice carrée et le graphe adjacent associé, nous pensons qu'une analyse de la centralité des noeuds – cf par exemple [141] – devrait être menée préalablement ou en complément de l'étape d'équilibrage, afin d'améliorer la qualité du partitionnement. Dans le cas de la détection de communautés par exemple, cela pourrait permettre de porter une attention particulière à la classification correcte des

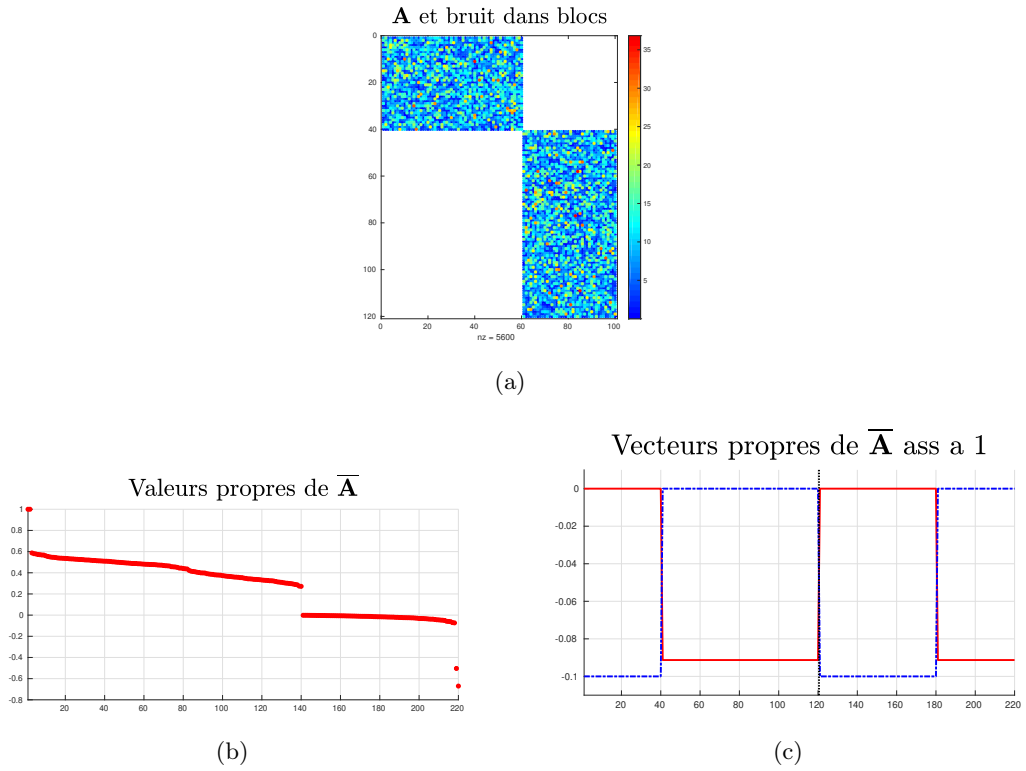


FIGURE 3.29 : *(a) : Une matrice rectangulaire* **A** *ayant une structure par blocs. La structure creuse de la matrice est idéale, mais les valeurs numériques dans ces blocs ont été perturbées par une loi normale. (b) : Répartition des valeurs propres de* $\overline{\mathbf{A}}$ *; les deux premières valent 1. (d) : Les deux vecteurs propres de* $\overline{\mathbf{A}}$ *associés à 1 permettent de faire apparaître la structure par blocs de lignes et de colonnes de* **A**.

noeuds possédant une forte centralité. En effet, le fait de travailler sur l'équilibrage doublement stochastique de la matrice d'adjacence fait *a priori* perdre l'information sur la mesure de centralité des noeuds la plus accessible, à savoir le degré (après l'équilibrage, tous les noeuds ont le même degré). Or, dans les applications réelles, on veut généralement privilégier la classification correcte des noeuds possédant la plus forte centralité, étant donné le fort impact de ces noeuds sur le reste du réseau. En réalité, il semble que cette information sur la centralité des noeuds subsiste dans les facteurs d'équilibrage : au regard de la littérature, ces facteurs semblent véhiculer une information importante, que ce soit sur la centralité des noeuds [26, 97] ou même sur la structure par blocs de la matrice sur laquelle est appliquée l'équilibrage [12]. Nous pensons qu'il est important de tirer profit de cette information complémentaire pour améliorer la qualité du partitionnement.

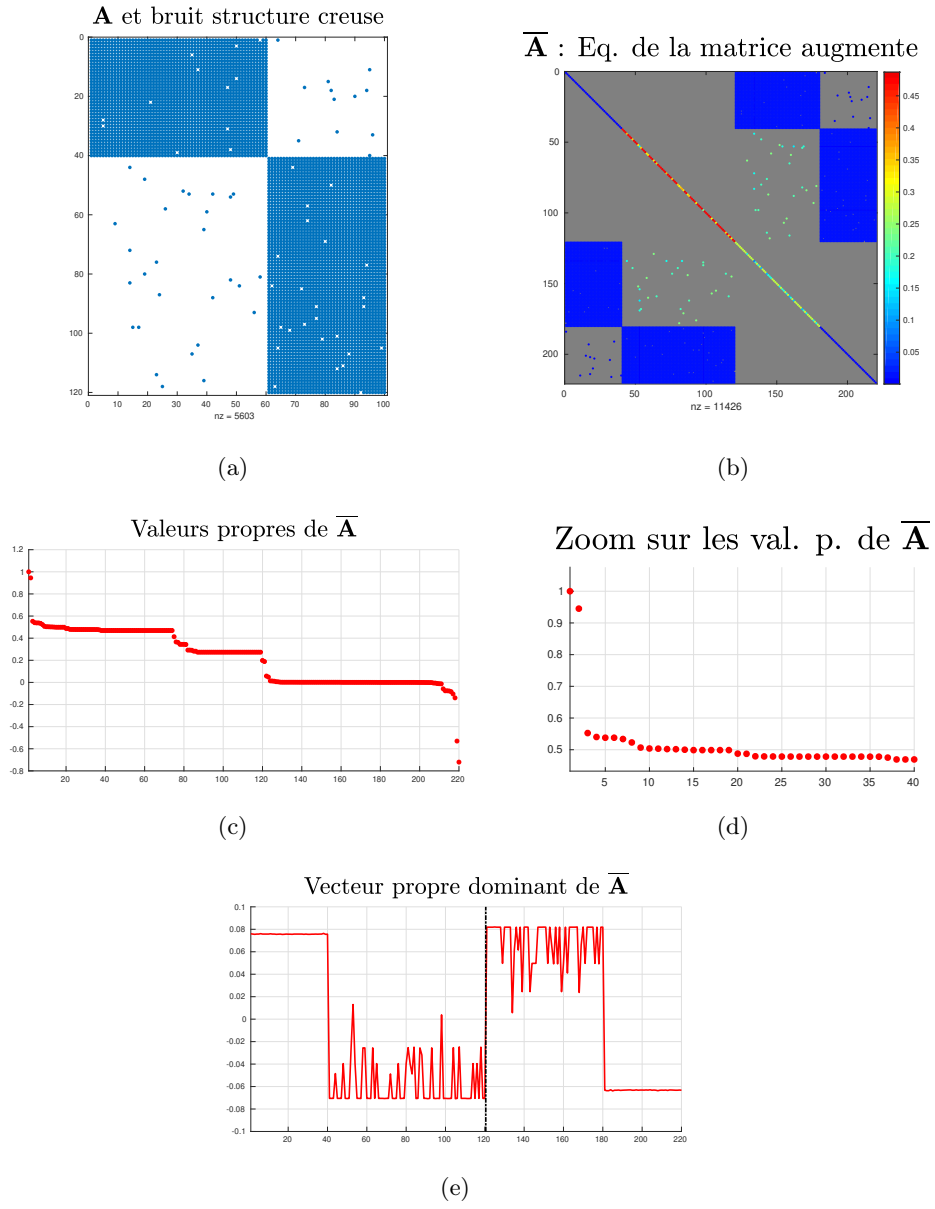


FIGURE 3.30 : *(a) : Une matrice rectangulaire* $\mathbf{A}$ *ayant une structure par blocs. La structure creuse de la matrice a été légèrement perturbée. (b) : Equilibrage doublement stochastique de* $\mathbf{A}$ *: on voit que les valeurs numériques les plus fortes sont les valeurs en dehors des blocs qui sont les plus proches de la diagonale. (c) : Répartition des valeurs propres de* $\overline{\mathbf{A}}$*. (d) : Zoom sur les premières valeurs propres de* $\overline{\mathbf{A}}$ *: il y a un net décrochage entre la deuxième et la troisième. (d) : Le vecteur propre de* $\overline{\mathbf{A}}$ *associé à sa seconde valeur propre permet plus ou moins de faire apparaître la structure par blocs de* $\mathbf{A}$*, mais il est très impacté par la perturbation, pourtant très faible, appliquée à la structure creuse de* $\mathbf{A}$*.*

Annexe A

Impact de l'équilibrage bi-stochastique sur les réseaux

A.1 Annexe 1

Propriété 7. *Soit* $\mathbf{A} \in \mathbb{R}_+^{n\times n}$*, soit* $F_1(\mathbf{A},.) : Eq(n) \to \mathbb{R}$*,* $F_2(\mathbf{A},.) : Eq(n) \to \mathbb{R}$*, deux mesures de modularités définies sur* $\mathbf{A}$ *par :*

$$\forall \mathbf{X} \in Eq(n), \begin{cases} F_1(\mathbf{A},\mathbf{X}) &= \sum\limits_{i,j} \big(\phi_{i,j} - \overline{\phi}^1_{i,j}\big)x_{i,j} \\ F_2(\mathbf{A},\mathbf{X}) &= \sum\limits_{i,j} \big(\phi_{i,j} - \overline{\phi}^2_{i,j}\big)x_{i,j} \end{cases}$$

$$\textit{Soit } \mathbf{X}_1 = \{x^1_{i,j}\}^n_{i,j=1} = \underset{\mathbf{X}\in Eq(n)}{argmax}(F_1(\mathbf{A},\mathbf{X})) \textit{ et } \mathbf{X}_2 = \{x^2_{i,j}\}^n_{i,j=1} = \underset{\mathbf{X}\in Eq(n)}{argmax}(F_2(\mathbf{A},\mathbf{X})),$$

$$\textit{alors : } \Big(\exists \delta > 0 : \forall (i,j) \in \{1...n\}^2, \overline{\phi}^1_{i,j} = \overline{\phi}^2_{i,j} + \delta, \Big) \implies \Big(\sum_{i,j} x^1_{i,j} \leq \sum_{i,j} x^2_{i,j}\Big)$$

Démonstration. On suppose que $\exists \delta > 0 : \forall (i,j) \in \{1...n\}^2, \overline{\phi}^1_{i,j} = \overline{\phi}^2_{i,j} + \delta$. Par définition de $\mathbf{X}_1$ et $\mathbf{X}_2$, on a :

$$\begin{cases} F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2) &\geq 0 \\ F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1) &\geq 0 \end{cases} \iff \begin{cases} \sum\limits_{i,j} \big(\phi_{i,j} - \overline{\phi}^1_{i,j}\big)\big(x^1_{i,j} - x^2_{i,j}\big) &\geq 0 \\ \sum\limits_{i,j} \big(\phi_{i,j} - \overline{\phi}^2_{i,j}\big)\big(x^2_{i,j} - x^1_{i,j}\big) &\geq 0 \end{cases}$$

Ainsi :

$$\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) \geq 0$$

Or :

$$\begin{aligned}
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= \sum_{i,j} \big(\phi_{i,j} - \overline{\phi}^1_{i,j}\big)\big(x^1_{i,j} - x^2_{i,j}\big) \\
&+ \sum_{i,j} \big(\phi_{i,j} - \overline{\phi}^2_{i,j}\big)\big(x^2_{i,j} - x^1_{i,j}\big) \\
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= -\sum_{i,j} \big(\phi_{i,j} - \overline{\phi}^1_{i,j}\big)\big(x^2_{i,j} - x^1_{i,j}\big) \\
&+ \sum_{i,j} \big(\phi_{i,j} - \overline{\phi}^2_{i,j}\big)\big(x^2_{i,j} - x^1_{i,j}\big) \\
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= \sum_{i,j} \overline{\phi}^1_{i,j}\big(x^2_{i,j} - x^1_{i,j}\big) \\
&- \sum_{i,j} \overline{\phi}^2_{i,j}\big(x^2_{i,j} - x^1_{i,j}\big) \\
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= \sum_{i,j} \big(\overline{\phi}^1_{i,j} - \overline{\phi}^2_{i,j}\big)\big(x^2_{i,j} - x^1_{i,j}\big) \\
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= \sum_{i,j} \delta\big(x^2_{i,j} - x^1_{i,j}\big) \\
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) &= \delta\big(\sum_{i,j} x^2_{i,j} - \sum_{i,j} x^1_{i,j}\big)
\end{aligned}$$

Donc

$$\begin{aligned}
\big(F_1(\mathbf{A},\mathbf{X}_1) - F_1(\mathbf{A},\mathbf{X}_2)\big) + \big(F_2(\mathbf{A},\mathbf{X}_2) - F_2(\mathbf{A},\mathbf{X}_1)\big) \geq 0 &\iff \delta\big(\sum_{i,j} x^2_{i,j} - \sum_{i,j} x^1_{i,j}\big) \geq 0 \\
&\iff \sum_{i,j} x^2_{i,j} \geq \sum_{i,j} x^1_{i,j}
\end{aligned}$$

□

A.2 Annexe 2

Propriété 8. *Soit $\lambda > n/2$, alors la solution du problème d'optimisation*

$$\mathcal{P} : \max_{\mathbf{X} \in Eq(n)} \big(F_{CC,\lambda}(\mathbf{A},\mathbf{X})\big)$$

défini sur la matrice stochastique $\mathbf{A}$ – associée au graphe de transition $G = (V,E)$ – n'admettra aucun élément dans la structure creuse de $\mathbf{A}$, en dehors des éléments diagonaux.

Démonstration. Le fait que les éléments diagonaux sont pris en compte vient de la définition même du partitionnement : si une matrice a un élément diagonal nul, elle ne représente pas un partitionnement, étant donné la propriété de réflexivité de toute relation d'équivalence.

Soit $\mathbf{X} = (x_{i,j})_{i,j=1}^{n} \in Eq(n)$ un partitionnement tel que $\exists (i^*, j^*) \in \overline{E} : i^* < j^*$ et $x_{i^*,j^*} = 1$. Alors, en notant $\mathbf{I}_n$ la matrice identité de taille $n \times n$, on a :

$$\begin{aligned}
F_{CC,\lambda}(\mathbf{A},\mathbf{X}) - F_{CC,\lambda}(\mathbf{A},\mathbf{I}_n) &= \sum_{i,j} \big(a_{i,j} - \lambda \mathcal{X}_{\overline{E}}(i,j)\big) x_{i,j} - \sum_{i=1}^{n} \big(a_{i,i} - \lambda \mathcal{X}_{\overline{E}}(i,i)\big) \\
&= \sum_{i,j} a_{i,j} x_{i,j} - \lambda \sum_{i \neq j} \mathcal{X}_{\overline{E}}(i,j) x_{i,j} - Tr(\mathbf{A}) \\
\text{(par symétrie de } \mathbf{X}) &= \sum_{i,j} a_{i,j} x_{i,j} - 2\lambda \sum_{\substack{i<j \\ (i,j)\in\overline{E}}} x_{i,j} - Tr(\mathbf{A}) \\
\big(\forall (i,j), a_{i,j} \geq 0 \text{ et } x_{i,j} = 0 \text{ ou } 1\big) &\leq \sum_{i,j} a_{i,j} - 2\lambda \sum_{\substack{i<j \\ (i,j)\in\overline{E}}} x_{i,j} - Tr(\mathbf{A}) \\
\big(\mathbf{A} \text{ est bi-stochastique}\big) &= n - Tr(\mathbf{A}) - 2\lambda \sum_{\substack{i<j \\ (i,j)\in\overline{E}}} x_{i,j} \\
\big(\text{vu } \lambda > 0 \text{ et } x_{i^*,j^*} = 1\big) &\leq n - Tr(\mathbf{A}) - 2\lambda \\
\big(\text{ étant donné } \lambda > n/2\big) &< -Tr(\mathbf{A}) \\
\big(\text{car } \mathbf{A} \text{ est à valeurs positives}\big) &< 0
\end{aligned}$$

Ainsi, si $\lambda > n/2$, on a $F_{CC,\lambda}(\mathbf{A},\mathbf{X}) < F_{CC,\lambda}(\mathbf{A},\mathbf{I}_n)$ pour tout partitionnement $\mathbf{X}$ ayant des éléments non nuls dans la structure creuse hors diagonale de $\mathbf{A}$. Dans ce cas, un tel partitionnement ne pourra en aucun cas être le partitionnement maximisant ce critère, et la propriété est donc démontrée. □

Remarque 19. *On note qu'on a démontré la propriété pour* $\lambda > \dfrac{n - Tr(\mathbf{A})}{2}$*, ce qui donne une borne plus fine pour* λ*.*

Contre-Exemple 1. *Soit* $\mathbf{X}_1, \mathbf{X}_2 \in Eq(n)$*, le fait d'avoir plus de relations dans* $\mathbf{X}_1$ *que dans* $\mathbf{X}_2$ *ne signifie pas que* $\mathbf{X}_1$ *a moins de classes que* $\mathbf{X}_2$*. C'est-à-dire, en notant* $\mathcal{K}_1$ *le nombre de classes de* $\mathbf{X}_1$*, et* $\mathcal{K}_2$ *le nombre de classes de* $\mathbf{X}_2$ *:*

$$\big(nnz(\mathbf{X}_1) > nnz(\mathbf{X}_2)\big) \not\Longrightarrow \big(\mathcal{K}_1 < \mathcal{K}_2\big)$$

Par exemple, en prenant $n = 7$*, et*

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & 1 & & & \\ & & & & 1 & 1 & 1 \\ & & & & 1 & 1 & 1 \\ & & & & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{X}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & & \\ 1 & 1 & 1 & 1 & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

on a $nnz(\mathbf{X}_1) = 4^2 + 3^2 = 25$*,* $nnz(\mathbf{X}_2) = 5^2 + 2 \times 1 = 27$*, avec* $\mathcal{K}_1 = 2$*,* $\mathcal{K}_2 = 3$*.*

Annexe B

Un Algorithme spectral

B.1 Annexe 1

Soit un graphe $G = (V, E)$, et $\mathbf{A}$ sa matrice d'adjacence. Nous rappelons que le Laplacien de ce graphe (introduit à la est $\mathbf{L} = \mathbf{D} - \mathbf{A}$, où $\mathbf{D}$ est la matrice des degrés de G, *id est* $d_{i,j}$ vaut le degré du noeud i si $i = j$, 0 sinon.

Propriété 9. *Soit $G = (V, E)$ un graphe simple et $\mathbf{L} \in \mathbb{R}^{n \times n}$ son Laplacien. Alors,*

$$\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{(i,j) \in E} (x_i - x_j)^2$$

Démonstration. On montre que les deux termes de l'égalité sont tous deux égaux à

$$2 \times \sum_{(i,j) \in E} \left(x_i^2 - x_i x_j \right)$$

Terme de droite :

$$\begin{aligned}
\sum_{(i,j) \in E} (x_i - x_j)^2 \quad &= \sum_{(i,j) \in E} \left(x_i^2 + x_j^2 - 2 x_i x_j \right) \\
&= \sum_{(i,j) \in E} x_i^2 + \sum_{(i,j) \in E} x_j^2 - 2 \sum_{(i,j) \in E} x_i x_j \\
(G \text{ n'est pas dirigé}) \quad &= 2 \sum_{(i,j) \in E} x_i^2 - 2 \sum_{(i,j) \in E} x_i x_j \\
&= 2 \times \sum_{(i,j) \in E} \left(x_i^2 - x_i x_j \right)
\end{aligned}$$

Terme de gauche : Nous introduisons d'abord une notation :

Si $(i, j) \in E$ on note $i \sim j$–qui est équivalent à $j \sim i$ puisque G n'est pas dirigé.

Notons $\mathbf{y} = \mathbf{L}\mathbf{x}$, on a

$$\forall k \in \{1, ..., n\}, y_k = x_k d_k - \sum_{i \sim k} x_i,$$

avec d_k le degré du noeud k.

$$\begin{aligned}
\mathbf{x}^T\mathbf{L}\mathbf{x} = \mathbf{x}^T\mathbf{y} \quad & = \sum_{k=1}^{n} \left(x_k^2 d_k - \sum_{i \sim k} x_k x_i \right) \\
& = \sum_k x_k^2 d_k - \sum_k \sum_{i \sim k} x_k x_i \\
(\textstyle\sum_k \sum_{i \sim k} \text{ parcours } E \text{ deux fois}) \quad & = \sum_k x_k^2 d_k - 2 \sum_{(i,j) \in E} x_i x_j \\
(d_k = \textstyle\sum_{i \sim k} 1 \text{ car } G \text{ n'est pas pondéré }) \quad & = \sum_k x_k^2 \sum_{i \sim k} 1 - 2 \sum_{(i,j) \in E} x_i x_j \\
& = \sum_k \sum_{i \sim k} x_k^2 - 2 \sum_{(i,j) \in E} x_i x_j \\
& = 2 \sum_{(i,j) \in E} x_i^2 - 2 \sum_{(i,j) \in E} x_i x_j \\
& = 2 \times \sum_{(i,j) \in E} \left(x_i^2 - x_i x_j \right)
\end{aligned}$$

et on a donc bien l'égalité entre les deux termes. □

Annexe C

Equilibrage bi-stochastique et matrices rectangulaires

C.1 Annexe 1

Le but de cette annexe est de démontrer qu'il existe une façon d'augmenter les matrices rectangulaires ayant une structure par blocs idéale de sorte à avoir une matrice carrée bi-irréductible dont les vecteurs propres dominants permettent d'observer la structure par blocs de la matrice rectangulaire.

Dans nos démonstrations, nous allons faire appel à l'algorithme $\texttt{symscale}_1$. Cet algorithme, présenté dans [27], est un algorithme permettant de trouver l'équilibrage doublement stochastique d'une matrice. Il s'agit d'un algorithme itératif, qui à chaque itération, divise de façon simultanée chaque ligne et chaque colonne de la matrice par la somme des éléments de cette ligne, respectivement de cette colonne. L'algorithme 6 explicite cette méthode.

Dans cette annexe, nous nous intéressons à une matrice rectangulaire ayant une structure par blocs idéale, ce que nous définissons formellement par une matrice rectangulaire $\mathbf{A} \in \mathbb{R}^{m \times n}$ telle que

$$\begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \end{bmatrix} \tag{C.1}$$

avec $\forall i \in \{1, ..., k\}, \mathbf{A}_i = \mu_i \mathbf{e}_{m_i} \mathbf{e}_{n_i}^T$, *id est* les blocs $\mathbf{A}_i$ sont des blocs rectangulaires composés d'une unique valeur numérique μ_i.

Nous notons $\widetilde{\mathbf{A}}$ la matrice augmentée de $\mathbf{A}$, définie par

$$\widetilde{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_m & \mathbf{A} \\ \mathbf{A}^T & \mathbf{I}_n \end{bmatrix} \tag{C.2}$$

Algorithm 6: : *Algorithme* `symscale`$_1$ *présenté dans [27]*

Data: $\mathbf{M} \in \mathbb{R}_+^{n\times n}$ une matrice.
Result: $\widehat{\mathbf{M}} \in \mathbb{R}_+^{n\times n}$ l'équilibrage doublement stochastique de $\mathbf{M}$.
begin
  $\mathbf{M}^{(0)} \leftarrow \mathbf{M}$;
  for k = *0,1,2,.. until convergence* **do**
    $\mathbf{R} \leftarrow \mathcal{D}\left(\left(\sqrt{||\mathbf{r}_i^{(k)}||_1}\right)_{i=1..n}\right)$; /* $\mathbf{r}_i^{(k)}$ est la i^{eme} ligne de $\mathbf{M}$ */
    $\mathbf{C} \leftarrow \mathcal{D}\left(\left(\sqrt{||\mathbf{c}_j^{(k)}||_1}\right)_{j=1..n}\right)$; /* $\mathbf{c}_j^{(k)}$ est la j^{eme} col. de $\mathbf{M}$ */
    $\mathbf{M}^{(k+1)} \leftarrow \mathbf{R}^{-1}\mathbf{M}^{(k)}\mathbf{C}^{-1}$;
  end
  $\widehat{\mathbf{M}} \leftarrow \mathbf{M}^{(k)}$;
end

Vu que $\widetilde{\mathbf{A}}$ est à diagonale pleine, elle a un support – cf par exemple [30]. De plus, $\widetilde{\mathbf{A}}$ est symétrique. Donc, d'après le Lemme 3.3 de [26], on peut permuter symétriquement $\widetilde{\mathbf{A}}$ pour qu'elle soit diagonale par blocs, et que chacun de ces blocs diagonaux soit irréductible. Puisque les permutations appliquées à $\widetilde{\mathbf{A}}$ sont symétriques, la diagonale de la matrice permutée est inchangée – et donc pleine. Ainsi, d'après la remarque 2, les blocs diagonaux sont bi-irréductibles. Cette matrice permutée est donc une somme directe de blocs bi-irréductibles, et la matrice initiale est donc à support total d'après [34].

La matrice $\widetilde{\mathbf{A}}$ est donc à support total. Ainsi, d'après le Théorème 3.1 de [27], l'algorithme 6 appliqué à $\widetilde{\mathbf{A}}$ converge vers un équilibrage doublement stochastique de cette matrice, que nous allons noter $\overline{\mathbf{A}}$. Nous allons commencer par expliciter l'équilibrage doublement stochastique de $\widetilde{\mathbf{A}}$. Cette matrice $\overline{\mathbf{A}}$ aura la même structure creuse que $\widetilde{\mathbf{A}}$. On montrera qu'on peut l'écrire sous la forme

$$\overline{\mathbf{A}} = \left[\begin{array}{ccc|ccc} \alpha_1\mathbf{I}_{m_1} & & & \widehat{\mathbf{A}}_1 & & \\ & \ddots & & & \ddots & \\ & & \alpha_k\mathbf{I}_{m_k} & & & \widehat{\mathbf{A}}_k \\ \hline \widehat{\mathbf{A}}_1^T & & & \beta_1\mathbf{I}_{n_1} & & \\ & \ddots & & & \ddots & \\ & & \widehat{\mathbf{A}}_k^T & & & \beta_k\mathbf{I}_{n_k} \end{array}\right] \tag{C.3}$$

avec $\forall i \in \{1, ..., k\}, \widehat{\mathbf{A}}_i = \lambda_i \mathbf{e}_{m_i} \times \mathbf{e}_{n_i}^T$, et $\lambda_i, \alpha_i, \beta_i > 0$.

Ensuite, nous démontrerons que les vecteurs propres dominants de $\overline{\mathbf{A}}$–ceux associés à 1–permettent d'observer la structure par blocs de la matrice initiale $\mathbf{A}$.

C.1.1 Convergence de `symscale`$_1$

Nous explicitons l'équilibrage doublement stochastique de la matrice $\widetilde{\mathbf{A}}$ grâce à la propriété 10.

Propriété 10. *L'algorithme 6* `symscale`$_1$ *appliqué à la matrice* $\widetilde{\mathbf{A}}$ *définie à l'équation (C.2) converge vers* $\overline{\mathbf{A}}$ *la matrice de l'équation C.3, avec*

$$\forall i \in \{1, ..., k\}, \begin{cases} \lambda_i = \frac{\mu_i}{\sqrt{1+a_i u_i}\sqrt{1+b_i/u_i}} \\ \alpha_i = \frac{1}{1+a_i u_i} \\ \beta_i = \frac{1}{1+b_i/u_i} \end{cases} \tag{C.4}$$

où $a_i = n_i\mu_i$, $b_i = m_i\mu_i$, *et* $u_i = \frac{b_i - a_i + \sqrt{(a_i-b_i)^2+4}}{2}$.

Pour démontrer cette propriété, on va avoir besoin d'une suite $(u_i^{(p)})_{p\geq 0}$ définie par récurrence de la manière suivante :

$$u_i^{(p)} = \begin{cases} 1 & \text{si } p = 1 \\ \sqrt{\frac{1+a_i u_i^{(p-1)}}{1+b_i/u_i^{(p-1)}}} & \text{si } p > 1 \end{cases}$$

On commence par montrer le lemme suivant :

Lemme 1. *Soit*

$$\widetilde{\mathbf{A}}^{(p)} = \left[\begin{array}{ccc|ccc} \alpha_1^{(p)}\mathbf{I}_{m_1} & & & \mathbf{A}_1^{(p)} & & \\ & \ddots & & & \ddots & \\ & & \alpha_k^{(p)}\mathbf{I}_{m_k} & & & \mathbf{A}_k^{(p)} \\ \hline {\mathbf{A}_1^{(p)}}^T & & & \beta_1^{(p)}\mathbf{I}_{n_1} & & \\ & \ddots & & & \ddots & \\ & & {\mathbf{A}_k^{(p)}}^T & & & \beta_k^{(p)}\mathbf{I}_{n_k} \end{array}\right]$$

la matrice obtenue après la p-ième itération de `symscale`$_1$ *appliquée à*

$$\widetilde{\mathbf{A}} = \widetilde{\mathbf{A}}^{(0)} = \begin{bmatrix} \mathbf{I}_m & \mathbf{A} \\ \mathbf{A}^T & \mathbf{I}_n \end{bmatrix}$$

où l'on précise que $\forall i \in \{1..k\}, \mathbf{A}_i^{(p)} = \lambda_i^{(p)} \mathbf{e}_{m_i} \mathbf{e}_{n_i}^T$. *Alors*

$$\forall i \in \{1..k\}, \forall p \geq 1, \begin{cases} \alpha_i^{(p)} = \frac{1}{1+a_i u_i^{(p)}} \\ \beta_i^{(p)} = \frac{1}{1+b_i/u_i^{(p)}} \\ \lambda_i^{(p)} = \frac{\mu_i}{\sqrt{1+a_i u_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} \end{cases}$$

Démonstration. On va démontrer ce lemme par récurrence.

Initialisation : $\forall t \in \{1..m+n\}$, on note $\widetilde{\mathbf{a}}_{t,\cdot}^{(0)}$ la t^{ieme} ligne de $\widetilde{\mathbf{A}}^{(0)}$. Alors

$$||\widetilde{\mathbf{a}}_{t,\cdot}^{(0)}||_1 = \begin{cases} 1 + n_i\mu_i = 1 + a_i = 1 + a_i u_i^{(1)} & \text{si } t \leq m \\ 1 + m_i\mu_i = 1 + b_i = 1 + b_i/u_i^{(1)} & \text{si } t > m \end{cases}$$

Alors, il est aisé de remarquer que

$$\begin{cases} \alpha_i^{(1)} & = \frac{1}{1+a_i u_i^{(1)}} \\ \beta_i^{(1)} & = \frac{1}{1+b_i/u_i^{(1)}} \\ \lambda_i^{(1)} & = \frac{\mu_i}{\sqrt{1+a_i u_i^{(1)}}\sqrt{1+b_i/u_i^{(1)}}} \end{cases}$$

Récurrence : Soit $p > 1$ tel que

$$\forall i \in \{1..k\}, \begin{cases} \alpha_i^{(p)} = \frac{1}{1+a_i u_i^{(p)}} \\ \beta_i^{(p)} = \frac{1}{1+b_i/u_i^{(p)}} \\ \lambda_i^{(p)} = \frac{\mu_i}{\sqrt{1+a_i u_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} \end{cases}$$

Alors, $\forall t \in \{1..m+n\}$, on a

$$||\widetilde{\mathbf{a}}_{t,\cdot}^{(p)}||_1 = \begin{cases} \alpha_i^{(p)} + n_i\lambda_i^{(p)} & \text{si } t \leq m, \\ \beta_i^{(p)} + m_i\lambda_i^{(p)} & \text{si } t > m. \end{cases}$$

Or

$$\begin{aligned}\alpha_i^{(p)} + n_i\lambda_i^{(p)} &= \frac{1}{1+a_iu_i^{(p)}} + \frac{n_i\mu_i}{\sqrt{1+a_iu_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} = \frac{1}{1+a_iu_i^{(p)}} + \frac{a_i}{\sqrt{1+a_iu_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} \\ &= \frac{\sqrt{1+b_i/u_i^{(p)}}}{\sqrt{1+b_i/u_i^{(p)}}(1+a_iu_i^{(p)})} + \frac{a_i\sqrt{1+a_iu_i^{(p)}}}{(1+a_iu_i^{(p)})\sqrt{1+b_i/u_i^{(p)}}} \\ &= \frac{\sqrt{1+b_i/u_i^{(p)}}+a_i\sqrt{1+a_iu_i^{(p)}}}{\sqrt{1+b_i/u_i^{(p)}}(1+a_iu_i^{(p)})} = \frac{1+a_i\sqrt{\frac{1+a_iu_i^{(p)}}{1+b_i/u_i^{(p)}}}}{1+a_iu_i^{(p)}} = \frac{1+a_iu_i^{(p+1)}}{1+a_iu_i^{(p)}}.\end{aligned}$$

De même

$$\begin{aligned}\beta_i^{(p)} + m_i\lambda_i^{(p)} &= \frac{1}{1+b_i/u_i^{(p)}} + \frac{m_i\mu_i}{\sqrt{1+a_iu_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} = \frac{1}{1+b_i/u_i^{(p)}} + \frac{b_i}{\sqrt{1+a_iu_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} \\ &= \frac{\sqrt{1+a_iu_i^{(p)}}}{\sqrt{1+a_iu_i^{(p)}}(1+b_i/u_i^{(p)})} + \frac{b_i\sqrt{1+b_i/u_i^{(p)}}}{(1+b_i/u_i^{(p)})\sqrt{1+a_iu_i^{(p)}}} \\ &= \frac{\sqrt{1+a_iu_i^{(p)}}+b_i\sqrt{1+b_i/u_i^{(p)}}}{\sqrt{1+a_iu_i^{(p)}}(1+b_i/u_i^{(p)})} = \frac{1+b_i\sqrt{\frac{1+b_i/u_i^{(p)}}{1+a_iu_i^{(p)}}}}{1+b_i/u_i^{(p)}} = \frac{1+b_i/u_i^{(p+1)}}{1+b_i/u_i^{(p)}}.\end{aligned}$$

Ainsi

$$\alpha_i^{(p+1)} = \frac{\alpha_i^{(p)}}{\frac{1+a_iu_i^{(p+1)}}{1+a_iu_i^{(p)}}} = \frac{1}{1+a_iu_i^{(p)}} \times \frac{1+a_iu_i^{(p)}}{1+a_iu_i^{(p+1)}} = \frac{1}{1+a_iu_i^{(p+1)}}.$$

De même

$$\beta_i^{(p+1)} = \frac{\beta_i^{(p)}}{\frac{1+b_i/u_i^{(p+1)}}{1+b_i/u_i^{(p)}}} = \frac{1}{1+b_i/u_i^{(p)}} \times \frac{1+b_i/u_i^{(p)}}{1+b_i/u_i^{(p+1)}} = \frac{1}{1+b_i/u_i^{(p+1)}}.$$

Aussi

$$\begin{aligned}\lambda_i^{(p+1)} &= \frac{\lambda_i^{(p)}}{\sqrt{\frac{1+b_i/u_i^{(p+1)}}{1+b_i/u_i^{(p)}}}\sqrt{\frac{1+a_i u_i^{(p+1)}}{1+a_i u_i^{(p)}}}} \\ &= \frac{\mu_i}{\sqrt{1+a_i u_i^{(p)}}\sqrt{1+b_i/u_i^{(p)}}} \times \frac{\sqrt{1+b_i/u_i^{(p)}}}{\sqrt{1+b_i/u_i^{(p+1)}}}\frac{\sqrt{1+a_i u_i^{(p)}}}{\sqrt{1+a_i u_i^{(p+1)}}} \\ &= \frac{\mu_i}{\sqrt{1+a_i u_i^{(p+1)}}\sqrt{1+b_i/u_i^{(p+1)}}}.\end{aligned}$$

Finalement, $\forall i \in \{1..k\}$, on a bien

$$\begin{cases}\alpha_i^{(p+1)} = \frac{1}{1+a_i u_i^{(p+1)}} \\ \beta_i^{(p+1)} = \frac{1}{1+b_i/u_i^{(p+1)}} \\ \lambda_i^{(p+1)} = \frac{\mu_i}{\sqrt{1+a_i u_i^{(p+1)}}\sqrt{1+b_i/u_i^{(p+1)}}}\end{cases},$$

et la récurrence est établie. □

Ainsi, on connaît l'expression de $\widetilde{\mathbf{A}}^{(p)}$ en fonction des suites $\left(u_i^{(p)}\right)_{p\geq 1}$. Il reste donc à étudier ces suites, et notamment à s'assurer qu'elles convergent vers une limite non nulle qu'il faut identifier. On le démontre en démontrant le lemme suivant.

Lemme 2. *La suite $\left(u^{(p)}\right)_{p\geq 1}$ définie par récurrence de la manière suivante :*

$$u^{(p)} = \begin{cases}1 & si\ p = 1 \\ \sqrt{\frac{1+au^{(p-1)}}{1+b/u^{(p-1)}}} & si\ p > 1\end{cases}$$

avec $a, b > 0$, converge vers la limite $\frac{(b-a)+\sqrt{(b-a)^2+4}}{2} > 0$.

Démonstration. Soit

$$f : \begin{array}{ccc}\mathbb{R}_+^* & \longrightarrow & \mathbb{R}_+^* \\ x & \mapsto & \sqrt{\frac{1+ax}{1+b/x}}\end{array}$$

On remarque qu'on peut ré-écrire $f(x) = \sqrt{\frac{x+ax^2}{x+b}}$. On calcule par ailleurs f' la dérivée

de f :

$$\begin{aligned} f'(x) &= \frac{1}{2(x+b)}\left(\frac{(2ax+1)\sqrt{x+b}}{\sqrt{x+ax^2}} - \frac{\sqrt{ax^2+x}}{\sqrt{x+b}}\right) \\ &= \frac{1}{2(x+b)\sqrt{x+ax^2}\sqrt{x+b}}\Big((2ax+1)(x+b) - (ax^2+x)\Big) \\ &= \frac{1}{2(x+b)\sqrt{x+ax^2}\sqrt{x+b}}\Big(ax^2+2abx+b\Big) > 0 \text{ étant donné que } x,a,b \in \mathbb{R}_+^* \end{aligned}$$

Ainsi, f est strictement croissante sur $\mathbb{R}_+^*$. En outre on a :

$f(x) \geq x \iff \sqrt{\frac{x+ax^2}{x+b}} \geq x \iff \frac{x+ax^2}{x+b} \geq x^2$ vu que la fonction carrée est croissante sur $\mathbb{R}_+^*$
$f(x) \geq x \iff x + ax^2 \geq x^3 + bx^2 \iff x^2 + (b-a)x - 1 \leq 0$ vu que $x > 0$.

On note x_{r_1} et x_{r_2} les racines du polynôme $x^2 + (b-a)x - 1$. On a alors

$$f(x) \geq x \iff x \in [x_{r_1}, x_{r_2}] \cap \mathbb{R}_+^*$$

Or, $\forall a, b > 0$, on a $x_{r_1} = \frac{(b-a)-\sqrt{(b-a)^2+4}}{2} < 0$ et $x_{r_2} = \frac{(b-a)+\sqrt{(b-a)^2+4}}{2} > 0$.

On a donc la table de variations suivante pour f :

x	$0^+ \ldots$	x_{r_2}	$\ldots \infty$
f		$\nearrow$	
$f(x) - x$	$+$	0	$-$

Soit une suite $\left(v^{(p)}\right)_{p\in\mathbb{N}}$ construite par récurrence sur f–$\forall p \in \mathbb{N}, v^{(p+1)} = f(v^{(p)})$–telle que $v^{(0)} > 0$. Alors

— Soit $v^{(0)} < x_{r_2}$. Alors, $\forall p \in \mathbb{N}, v^{(p+1)} = f(v^{(p)}) > v^{(p)}$. Donc cette suite est croissante. De plus, elle est majorée par x_{r_2}. En effet, f est croissante donc $\forall x \in]0, x_{r_2}], f(x) \leq f(x_{r_2}) = x_{r_2}$, avec x_{r_2} l'unique point fixe de f sur $\mathbb{R}_+{}^*$. Ainsi, $\left(v^{(p)}\right)_{p\in\mathbb{N}}$ converge vers x_{r_2}.

— Soit $v^{(0)} > x_{r_2}$. Alors, $\forall p \in \mathbb{N}, v^{(p+1)} = f(v^{(p)}) < v^{(p)}$. Donc cette suite est décroissante, et minorée par x_{r_2} le point fixe de f. Ainsi, $\left(v^{(p)}\right)_{p\in\mathbb{N}}$ converge vers x_{r_2}.

Donc toute suite récurrente construite sur f dont le point de départ est strictement positif

converge vers x_{r_2}. En particulier

$$u^{(p)} \underset{p\to\infty}{\longrightarrow} \frac{(b-a)+\sqrt{(b-a)^2+4}}{2}$$

□

Ainsi, on a

$$\forall i \in \{1..k\}, u_i^{(p)} \underset{p\to\infty}{\longrightarrow} \frac{(b_i-a_i)+\sqrt{(b_i-a_i)^2+4}}{2}$$

La convergence de la suite et la relation de récurrence démontrée sur $\widetilde{\mathbf{A}}^{(p)}$ font qu'on a bien démontré la propriété 10, à savoir que $\widetilde{\mathbf{A}}^{(p)} \underset{p\to\infty}{\longrightarrow} \overline{\mathbf{A}}$ comme définie à l'équation C.4.

En conclusion nous avons montré que l'équilibrage doublement stochastique de la matrice $\widetilde{\mathbf{A}}$ définie à l'équation (C.2) est la matrice $\overline{\mathbf{A}}$, définie par

$$\overline{\mathbf{A}} = \left[\begin{array}{ccc|ccc} \alpha_1\mathbf{I}_{m_1} & & & \widehat{\mathbf{A}}_1 & & \\ & \ddots & & & \ddots & \\ & & \alpha_k\mathbf{I}_{m_k} & & & \widehat{\mathbf{A}}_k \\ \hline \widehat{\mathbf{A}}_1^T & & & \beta_1\mathbf{I}_{n_1} & & \\ & \ddots & & & \ddots & \\ & & \widehat{\mathbf{A}}_k^T & & & \beta_k\mathbf{I}_{n_k} \end{array}\right]$$

avec $\forall i \in \{1,...,k\}, \widehat{\mathbf{A}}_i = \lambda_i \mathbf{e}_{m_i} \times \mathbf{e}_{n_i}^T$, et $\lambda_i, \alpha_i, \beta_i > 0$, et

$$\forall i \in \{1,...,k\}, \begin{cases} \lambda_i = \frac{\mu_i}{\sqrt{1+a_iu_i}\sqrt{1+b_i/u_i}} \\ \alpha_i = \frac{1}{1+a_iu_i} \\ \beta_i = \frac{1}{1+b_i/u_i} \end{cases}$$

où $a_i = n_i\mu_i$, $b_i = m_i\mu_i$, et $u_i = \frac{b_i-a_i+\sqrt{(a_i-b_i)^2+4}}{2}$.

C.1.2 Trouver la structure par blocs de la matrice initiale

Nous avons montré qu'il est possible de faire converger $\widetilde{\mathbf{A}}$, la matrice augmentée de $\mathbf{A}$ définie équation (C.1), vers la matrice $\overline{\mathbf{A}}$ doublement stochastique de la propriété 10.

Nous allons maintenant vérifier qu'il est possible de récupérer l'information sur la structure par blocs de lignes et de colonnes de $\mathbf{A}$ grâce aux vecteurs propres dominants de la matrice $\overline{\mathbf{A}}$.

Propriété 11. *Soit* $\mathbf{x}$ *un vecteur propre de* $\overline{\mathbf{A}}$ *associé à la valeur propre* 1*. Alors le vecteur composé des* m *premières coordonnées de* $\mathbf{x}$ *est une combinaison linéaire des vecteurs caractérisant les blocs de lignes de* $\mathbf{A}$*. De même le vecteur composé des* n *dernières coordonnées de* $\mathbf{x}$ *est une combinaison linéaire des vecteurs caractérisant les blocs de colonnes de* $\mathbf{A}$*.*

Démonstration. Soit $\mathbf{x}$ un vecteur propre de $\overline{\mathbf{A}}$ associé à la valeur propre $\theta = 1$. On note $\mathbf{y}_1 \in \mathbb{R}^{m_1}$ le vecteur contenant les coordonnées de $\mathbf{x}$ correspondant au premier bloc de lignes, $\mathbf{y}_2 \in \mathbb{R}^{m_2}$ les coordonnées de $\mathbf{x}$ correspondant au deuxième bloc de lignes,... De la même façon, on note $\mathbf{z}_1 \in \mathbb{R}^{n_1}$ le vecteur contenant les coordonnées de $\mathbf{x}$ correspondant au premier bloc de colonnes, etc., de sorte que l'on puisse écrire $\mathbf{x}$ sous la forme :

$$\mathbf{x} = [{\mathbf{y}_1}^T, ..., {\mathbf{y}_k}^T, {\mathbf{z}_1}^T, ..., {\mathbf{z}_k}^T]^T$$

On a donc

$$\overline{\mathbf{A}}\mathbf{x} = \theta\mathbf{x} \iff \left[\begin{array}{ccc|ccc} \alpha_1\mathbf{I}_{m_1} & & & \widehat{\mathbf{A}}_1 & & \\ & \ddots & & & \ddots & \\ & & \alpha_k\mathbf{I}_{m_k} & & & \widehat{\mathbf{A}}_k \\ \hline \widehat{\mathbf{A}}_1^T & & & \beta_1\mathbf{I}_{n_1} & & \\ & \ddots & & & \ddots & \\ & & \widehat{\mathbf{A}}_k^T & & & \beta_k\mathbf{I}_{n_k} \end{array}\right] \begin{bmatrix} \mathbf{y}^1 \\ \vdots \\ \mathbf{y}^k \\ \mathbf{z}^1 \\ \vdots \\ \mathbf{z}^k \end{bmatrix} = \theta \begin{bmatrix} \mathbf{y}^1 \\ \vdots \\ \mathbf{y}^k \\ \mathbf{z}^1 \\ \vdots \\ \mathbf{z}^k \end{bmatrix}$$

Ainsi,

$$\overline{\mathbf{A}}\mathbf{x} = \theta\mathbf{x} \iff \forall p \in \{1, ..., k\}, \begin{cases} \forall i \in \{1..m_p\}, & \lambda_p \sum_{t=1}^{n_p} \mathbf{z}_p(t) = (\theta - \alpha_p)\mathbf{y}_p(i) \\ \forall j \in \{1..n_p\}, & \lambda_p \sum_{s=1}^{m_p} \mathbf{y}_p(s) = (\theta - \beta_p)\mathbf{z}_p(j) \end{cases} \tag{C.5}$$

Le terme de gauche dans les deux équations ne dépendant pas de i, respectivement de j, on peut écrire :

$$\overline{\mathbf{A}}\mathbf{x} = \theta\mathbf{x} \implies \forall p \in \{1...k\}, \begin{cases} \text{soit } \mathbf{u}^p(1) = ... = \mathbf{u}^p(m_p); & \text{soit } \theta = \alpha_p \\ \text{soit } \mathbf{v}^p(1) = ... = \mathbf{v}^p(n_p); & \text{soit } \theta = \beta_p \end{cases} \tag{C.6}$$

Or en reprenant les notations de l'équation C.4, on a

$$\forall p \in \{1, ..., k\}, \begin{cases} \lambda_p = \dfrac{\mu_p}{\sqrt{1 + a_p u_p}\sqrt{1 + b_p/u_p}} \\ \alpha_p = \dfrac{1}{1 + a_p u_p} \\ \beta_p = \dfrac{1}{1 + b_p/u_p} \end{cases} \tag{C.7}$$

avec $u_p = \dfrac{(b_p - a_p) + \sqrt{(a_p - b_p)^2 + 4}}{2}$. En remarquant que $u_p > 0$, on obtient $\alpha_p, \beta_p < 1$. Puisque $\theta = 1$ est valeur propre de $\overline{\mathbf{A}}$ ($\overline{\mathbf{A}}\mathbf{e} = \mathbf{e}$), cela contraint les vecteurs propres de $\overline{\mathbf{A}}$ associés à $\theta = 1$ à être constants par morceaux, et en particulier sont des combinaisons linéaires des vecteurs caractérisant les lignes et les colonnes de $\mathbf{A}$.

Nous montrons maintenant que tout vecteur correspondant à une combinaison linéaire des vecteurs caractérisant les blocs de lignes peut être compléter afin d'obtenir un vecteur propre de $\overline{\mathbf{A}}$ associé à $\theta = 1$, de même pour les colonnes.

Choisissons maintenant $\mathbf{y}$ une combinaison linéaire des vecteurs caractérisant les blocs de lignes de $\mathbf{A}$, autrement dit $\mathbf{y} = [\varphi_1 \mathbf{e}_{m_1}{}^T, ...\varphi_k \mathbf{e}_{m_k}{}^T]^T$, avec $\varphi_1, .., \varphi_k$ des scalaires.

En posant

$$\forall p \in \{1, ..., k\}, \psi_p = \frac{u_p\sqrt{1 + b_p/u_p}}{\sqrt{1 + a_p u_p}}\varphi_p$$

puis en construisant le vecteur $\mathbf{z}$ tel que

$$\mathbf{z} = [\psi_1 \mathbf{e}_{n_1}{}^T, ..., \psi_k \mathbf{e}_{n_k}{}^T]^T,$$

et en réécrivant les μ_p, α_p et β_p de l'équation (C.5) par leurs valeurs en fonction de $a_p = \mu_p n_p$, b_p, u_p rappelés à l'équation (C.7), on voit que $\mathbf{y}$ et $\mathbf{z}$ vérifient l'équation C.5 avec $\theta = 1$. Ainsi, $\mathbf{x} = [\mathbf{y}^T, \mathbf{z}^T]^T$ est un vecteur propre de $\overline{\mathbf{A}}$ associé à la valeur propre $\theta = 1$.

De même, en fixant d'abord $\mathbf{z} = [\psi_1 \mathbf{e}_{n-1}{}^T, ..., \psi_k \mathbf{e}_{n_k}{}^T]^T$, puis

$$\mathbf{y} = [\varphi_1 \mathbf{e}_{m_1}{}^T, ..., \varphi_k \mathbf{e}_{m_k}{}^T]^T \text{ avec } \forall p \in \{1...k\}, \varphi_p = \frac{\sqrt{1 + a_p u_p}}{u_j\sqrt{1 + b_p/u_p}}\psi_p,$$

on a $\mathbf{x} = [\mathbf{y}^T, \mathbf{z}^T]^T$ vecteur propre de $\overline{\mathbf{A}}$ associé à $\theta = 1$. □

Remarque 20. *La démonstration de cette propriété permet en outre de montrer que si $\overline{\mathbf{A}}$ possède des valeurs propres négatives, alors les vecteurs propres associés doivent aussi permettre de mettre en évidence la structure par blocs de $\mathbf{A}$. En effet, en établissant*

l'équation (C.6), on a démontré que, si une valeur propre θ est différente de tous les α_p et β_p, $\forall p \in \{1, ..., k\}$, alors les vecteurs propres qui lui seront associés sont nécessairement constants par morceaux. Et vu que $\forall p \in \{1, ..., k\}, \alpha_p, \beta_p > 0$, ce résultat est en particulier vrai pour les valeurs propres négatives, si elles existent.

Dans la propriété précédente, on a démontré que si $\overline{\mathbf{A}}$ est l'équilibrage doublement stochastique de $\widetilde{\mathbf{A}}$, la matrice augmentée de la matrice $\mathbf{A}$ définie à l'équation (C.1), alors les vecteurs propres de $\overline{\mathbf{A}}$ associés à 1 seront constants par morceaux, chaque partie constante permettant de détecter un des blocs de lignes ou de colonnes de la matrice initiale $\mathbf{A}$. En outre si un vecteur $\mathbf{y} \in \mathbb{R}^m$ est constant par morceaux, chaque partie constante coïncidant avec un bloc de lignes de $\mathbf{A}$, il existe un vecteur $\mathbf{z} \in \mathbb{R}^n$ tel que $\mathbf{x} = [\mathbf{y}^T, \mathbf{z}^T]^T$ soit un vecteur propre de $\overline{\mathbf{A}}$ associé à 1. De même si $\mathbf{z} \in \mathbb{R}^n$ est un vecteur constant par morceaux dont chaque partie constante coïncide avec un bloc de colonnes de $\mathbf{A}$.

Bibliographie

[1] R. Barriot, P. Langendijk-Genevaux, Y. Quentin, and G. Fichant, "Graph partitioning in genomic data analysis," in *High performance linear and nonlinear methods for scale applications, Workshop Innovative clustering methods for large graphs and block methods*, CIMI Semester, 2015.

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

[3] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software*, vol. 38, no. 1, pp. 1–14, 2011.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, Mar. 2003.

[5] N. Asher, J. Hunter, M. Morey, B. Farah, and S. Afantenos, "Discourse structure and dialogue acts in multiparty dialogue : the STAC corpus," in *Proceedings of the 20th International Conference on Language Resources and Evaluation*, pp. 2721 – 2727, 2016.

[6] M. Qi, "Use of deep learning approaches for the prediction of discourse srtuctures," Master's thesis, Université Paul Sabatier, Toulouse, 2018.

[7] S. Fortunato and D. Hric, "Community detection in networks : A user guide," *CoRR*, vol. abs/1608.00163, 2016.

[8] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 9, pp. 2658–2663, 2004.

[9] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 888–905, Aug. 2000.

[10] S. Mouysset, H. Zbib, S. Stute, J.-M. Girault, J. Charara, J. Noailles, S. Chalon, I. Buvat, and C. Tauber, "Segmentation of Dynamic PET Images with Kinetic Spectral Clustering," *Physics in Medicine and Biology*, vol. 58, pp. 6931–6944, 2013.

[11] M. Girolami, "Mercer kernel-based clustering in feature space," *Trans. Neur. Netw.*, vol. 13, pp. 780–784, May 2002.

[12] C. Laclau, I. Redko, B. Matei, Y. Bennani, and V. Brault, "Co-clustering through optimal transport," in *ICML*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 1955–1964, PMLR, 2017.

[13] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, (New York, NY, USA), pp. 89–98, ACM, 2003.

[14] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein, "Spectral Biclustering of Microarray Data : Coclustering Genes and Conditions," *Genome Res.*, vol. 13, no. 4, pp. 703–716, 2003.

[15] M. Onizuka, T. Fujimori, and H. Shiokawa, "Graph partitioning for distributed graph processing," *Data Science and Engineering*, vol. 2, pp. 94–105, Mar 2017.

[16] Y. Zhu and A. H. Sameh, "How to generate effective block jacobi preconditioners for solving large sparse linear systems," in *Advances in Computational Fluid-Structure Interaction and Flow Simulation : New Methods and Challenging Computations* (Y. Bazilevs and K. Takizawa, eds.), (Cham), pp. 231–244, Springer International Publishing, 2016.

[17] B. L. Chamberlain, "Graph partitioning algorithms for distributing workloads of parallel computations," tech. rep., 1998.

[18] S. Acer, E. Kayaaslan, and C. Aykanat, "A recursive bipartitioning algorithm for permuting sparse square matrices into block diagonal form with overlap," *SIAM J. Scientific Computing*, vol. 35, 2013.

[19] T. Drummond, I. Duff, R. Guivarch, D. Ruiz, and M. Zenadi, "Partitioning Strategies for the Block Cimmino Algorithm," *Journal of Engineering Mathematics*, vol. Hors-série, p. (on line), août 2014.

[20] F. Sukru Torun, M. Manguoglu, and C. Aykanat, "A novel partitioning method for accelerating the block cimmino algorithm," *SIAM Journal on Scientific Computing*, vol. 40, 10 2017.

[21] E. Vecharynski, Y. Saad, and M. Sosonkina, "Graph partitioning using matrix values for preconditioning symmetric positive definite systems," *SIAM J. Scientific Computing*, vol. 36, no. 1, pp. A63–A87, 2014.

[22] M. Idel, "A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps," *arXiv e-prints*, p. arXiv :1609.06349, Sep 2016.

[23] D. Hartfiel and J. Spellman, "A role for doubly stochastic matrices in graph theory," *Proceedings of The American Mathematical Society - PROC AMER MATH SOC*, vol. 36, 12 1972.

[24] P. B. Slater, "Hubs and Clusters in the Evolving U. S. Internal Migration Network," *arXiv e-prints*, p. arXiv :0809.2768, Sep 2008.

[25] E. Seneta, *Non-Negative Matrices and Markov Chains*, ch. 2.6,4.1. Springer, 2006.

[26] P. A. Knight, "The sinkhorn-knopp algorithm : Convergence and applications," *SIAM J. Matrix Anal. Appl.*, vol. 30, pp. 261–275, Mar. 2008.

[27] P. A. Knight, D. Ruiz, and B. Uçar, "A Symmetry Preserving Algorithm for Matrix Scaling," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, p. 25, 2014.

[28] A. Lancichinetti and S. Fortunato, "Community detection algorithms : A comparative analysis," *Physical Review E*, vol. 80, pp. 056117+, Nov. 2009.

[29] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics : Theory and Experiment*, vol. 10, p. 10008, Oct. 2008.

[30] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific J. Math.*, vol. 21, pp. 343–348, 1967.

[31] A. Y. Ng, M. I. Jordan, and Y. 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, (Cambridge, MA, USA), pp. 849–856, MIT Press, 2001.

[32] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, p. 036104, 2006.

[33] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, (New York, NY, USA), pp. 269–274, ACM, 2001.

[34] R. A. Brualdi, *Combinatorial Matrix Classes*. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2006.

[35] A. Berman and R. J. Plemmons, *Chapt. 2 - Nonnegative Matrices*, pp. 26 – 62. Academic Press, 1979.

[36] A. Pothen and C.-J. Fan, "Computing the block triangular form of a sparse matrix," *ACM Transactions on Mathematical Software*, vol. 16, pp. 303–324, 1990.

[37] P. Conde-Césepdes and J. Marcotorchino, "Modularisation et recherche de communautés dans les réseaux complexes par unification relationnelle," *Revue des Nouvelles Technologies de l'Information*, vol. Apprentissage Artificiel et Fouille de Données vol. RNTI-A-6, pp. 71–98, 2012.

[38] F. Wang, P. Li, A. C. König, and M. Wan, "Improving clustering by learning a bi-stochastic data similarity matrix," *Knowledge and Information Systems*, vol. 32, pp. 351–382, Aug 2012.

[39] F. Gargiulo, A. Caen, R. Lambiotte, and T. Carletti, "The classical origin of modern mathematics," *EPJ Data Science*, vol. 5 :26, 2016.

[40] V. Calderon, R. Barriot, Y. Quentin, and G. Fichant, "Crossing isorthology and microsynteny to resolve multigenic families functional annotation," *22nd International Workshop on Database and Expert Systems Applications*, pp. 440–444, 2011.

[41] P. Conde-Céspedes, "Modélisation et extension du formalisme de l'analyse relationnelle mathématique à la modularisation des grands graphes," *PhD Thesis, Université Pierre et Marie Curie*, 2013.

[42] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004.

[43] J. Oswinski and S. Zadrozny, "Structuring a regional problem : aggregation and clustering in orderings," *Appl. Stoch. Models and Data An.*, vol. 2, pp. 83–95, 1986.

[44] P. Michaud and J. Marcotorchino, "Modèles d'optimisation en analyse des données relationnelles," *Math. Sci. hum*, vol. 67, pp. 7–38, 1979.

[45] C. Zahn, "Approximating symmetric relations by equivalence relations," *SIAM Journal on Applied Mathematics*, vol. 12, pp. 840–847, 1964.

[46] L. Lovász *et al.*, "Random walks on graphs : A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1–46, 1993.

[47] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[48] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13–23, 2009.

[49] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proceedings of the National Academy of Sciences*, vol. 107, no. 29, pp. 12755–12760, 2010.

[50] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, p. 016107, Jan 2011.

[51] M. E. J. Newman, "Equivalence between modularity optimization and maximum likelihood methods for community detection," *Phys. Rev. E*, vol. 94, p. 052315, Nov 2016.

[52] N. Veldt, D. F. Gleich, and A. Wirth, "A correlation clustering framework for community detection," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (Republic and Canton of Geneva, Switzerland), pp. 439–448, International World Wide Web Conferences Steering Committee, 2018.

[53] R. Campigotto, P. Conde-Céspedes, and J. Guillaume, "A generalized and adaptative method for community detection," tech. rep., 2014.

[54] M. E. Newman, "Analysis of weighted networks," *Physical review E*, vol. 70, no. 5, p. 056131, 2004.

[55] M. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 70, p. 056131, 2004.

[56] S. Cafieri, P. Hansen, and L. Liberti, "Locally optimal heuristic for modularity maximization of networks," *Physical Review E*, vol. 83, p. 056105, 2011.

[57] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Transactions on knowledge and data engineering*, vol. 20, pp. 172–188, 2008.

[58] S. Fortunato and M. Barthélemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[59] M. Girvan and M. Newman, "Community structure in social and biological networks," *Proc. Natl. Acad. Sci., USA*, vol. 99, pp. 7821–7826, 2002.

[60] E. Demaine and N. Immorlica, "Correlation clustering with partial information," *Proceedings of the 6th international workshop on approximation algorithms for combinatorial optimization problems and 7th international workshop on randomization and approximation techniques in computer science*, 2003.

[61] P. Conde-Céspedes and J. Marcotorchino, "Comparison of linear modularization criteria of networks using relational metrics," *45emes journées de Statistiques*, 2013.

[62] P. A. Knight and D. Ruiz, "A fast algorithm for matrix balancing," *IMA Journal of Numerical Analysis*, vol. 33, pp. 1029—1047, 2013.

[63] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, p. 066111, Dec. 2004.

[64] Z. Yang, R. Algesheimer, and C. J. Tessone, "A comparative analysis of community detection algorithms on artificial networks," in *Scientific reports*, 2016.

[65] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, p. 046110, Oct. 2008.

[66] A. L. N. Fred and A. K. Jain, "Robust data clustering.," in *CVPR (2)*, pp. 128–136, IEEE Computer Society, 2003.

[67] J. M. Kumpula, J. Saramäki, K. Kaski, and J. Kertész, "Limited resolution in complex network community detection with potts model approach," *The European Physical Journal B*, vol. 56, pp. 41–45, Mar 2007.

[68] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[69] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[70] D. Fritzsche, V. Mehrmann, D. B. Szyld, and E. Virnik, "An SVD approach to identifying metastable states of Markov chains," *Electronic Transactions on Numerical Analysis*, vol. 29, pp. 46–69, 2008.

[71] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[72] J. Lei and A. Rinaldo, "Consistency of spectral clustering in stochastic block models," *The Annals of Statistics*, vol. 43, no. 1, pp. 215–237, 2015.

[73] G. Frobenius, "Ueber matrizen aus nicht negativen elementen," *Sitzungsber. Königl. Preuss. Akad. Wiss*, p. 456–477, 1912.

[74] O. Perron, "Zur theorie der matrices," *Mathematische Annalen*, vol. 64, no. 2, pp. 248–263, 1907.

[75] B. Bekka, "Le théorème de Perron-Frobenius, les chaines de Markov et un célèbre moteur de recherche," 2007.

[76] A. Pothen, H. D. Simon, and K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[77] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[78] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[79] B. Slininger, "Fiedler's theory of spectral graph partitioning," 2018.

[80] L. Labiod and M. Nadif, "A unified framework for data visualization and coclustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 2194–2199, Sep. 2015.

[81] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *SIAM J. Numer. Anal.*, vol. 2, no. 2, pp. 205–224, 1965.

[82] G. Plassman, "A survey of singular value decomposition methods and performance comparison of some available serial codes," Tech. Rep. NASA, Langley Research Center, August 2005.

[83] C. Musco and C. Musco, "Randomized block krylov methods for stronger and faster approximate singular value decomposition," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 1396–1404, Curran Associates, Inc., 2015.

[84] G. W. Stewart, *Matrix Algorithms*. Society for Industrial and Applied Mathematics, 2001.

[85] A. L. Dulmage and N. S. Mendelsohn, "Coverings of bipartite graphs," *Canadian Journal of Mathematics*, vol. 10, pp. 517–534, 1958.

[86] J. Canny, "A computational approach to edge detection," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–698, 1986.

[87] J. Groß, "On the product of orthogonal projectors," *Linear Algebra and its Applications*, vol. 289, no. 1, pp. 141 – 150, 1999.

[88] G. H. Golub and C. F. Van Loan, *Matrix Computations*, ch. Computing Subspaces with the SVD. The Johns Hopkins University Press, third ed., 1996.

[89] J. P. Bagrow, "Communities and bottlenecks : Trees and treelike networks have high modularity," *Physical Review E*, vol. 85, no. 6, p. 066118, 2012.

[90] 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*, vol. 12, pp. 2825–2830, 2011.

[91] S. Mouysset, J. Noailles, and D. Ruiz, "Using a global parameter for gaussian affinity matrices in spectral clustering," in *VECPAR*, vol. 5336 of *Lecture Notes in Computer Science*, pp. 378–390, Springer, 2008.

[92] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, (New York, NY, USA), pp. 1177–1178, ACM, 2010.

[93] J. H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

[94] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231, AAAI Press, 1996.

[95] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang, "Spectral redemption in clustering sparse networks," *Proceedings of the National Academy of Sciences*, vol. 110, no. 52, pp. 20935–20940, 2013.

[96] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, (New York, NY, USA), pp. 89–98, ACM, 2003.

[97] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, pp. 604–632, Sept. 1999.

[98] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006.

[99] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, Apr. 1998.

[100] M. Manguoglu, M. Koyutürk, A. H. Sameh, and A. Grama, "Weighted matrix ordering and parallel banded preconditioners for iterative linear system solvers," *SIAM J. Scientific Computing*, vol. 32, no. 3, pp. 1201–1216, 2010.

[101] Y. Saad and M. H. Schultz, "Gmres : A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, July 1986.

[102] I. S. Duff and J. Koster, "On algorithms for permuting large entries to the diagonal of a sparse matrix," *SIAM J. Matrix Anal. Appl.*, vol. 22, pp. 973–996, July 2000.

[103] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, Dec. 1998.

[104] I. Duff, R. Guivarch, D. Ruiz, and M. Zenadi, "The Augmented Block Cimmino Distributed method," *SIAM Journal on Scientific Computing*, vol. 37, p. (on line), juillet 2015.

[105] A. Kongthon, C. Sangkeettrakarn, S. Kongyoung, and C. Haruechaiyasak, "Implementing an online help desk system based on conversational agent," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, MEDES '09, (New York, NY, USA), pp. 69 :450–69 :451, ACM, 2009.

[106] R. Kumar, S. Lattanzi, and P. Raghavan, "An algorithmic treatment of strong queries," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, (New York, NY, USA), pp. 775–784, ACM, 2011.

[107] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA : McGraw-Hill, Inc., 1986.

[108] G. Tambouratzis, S. Sofianopoulos, and M. Vassiliou, "Language-independent hybrid Mt with Presemt," in *Proceedings of HYTRA-2013 Workshop, held within the ACL-2013 Conference*, (Sofia, Bulgaria), pp. 123–130, ACL, 2013.

[109] T. Van de Cruys, "La génération de poésies en français," in *Conférence sur le Traitement Automatique des Langues Naturelles*, TALN 2019, pp. 113–123, 2019.

[110] J. Pennington, R. Socher, and C. Manning, "Glove : Global vectors for word representation," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 1532–1543, 2014.

[111] C. Osgood, G. Suci, and P. Tenenbaum, *The Measurement of meaning*. Urbana : : University of Illinois Press, 1957.

[112] C. Fabre and A. Lenci, "Distributional Semantics Today Introduction to the special issue," *Traitement Automatique des Langues*, vol. 56, no. 2, pp. 7–20, 2015.

[113] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[114] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, pp. 203–208, Jun 1996.

[115] E. Rostand, "Cyrano de Bergerac," 1897.

[116] P. D. Turney and P. Pantel, "From frequency to meaning : Vector space models of semantics," *J. Artif. Int. Res.*, vol. 37, pp. 141–188, Jan. 2010.

[117] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[118] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, and Y. Matsuo, "Towards an open-domain conversational system fully based on natural language processing," in *Proceedings of the 25th International Conference on Computational Linguistics : Technical Papers*, pp. 928–939, 2014.

[119] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 654–664, 2017.

[120] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[121] D. P. Kingma and J. Ba, "Adam : A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[122] T. Pellegrini and S. Mouysset, "Inferring phonemic classes from CNN activation maps using clustering techniques," in *Annual conference Interspeech (INTERSPEECH 2016)*, (San Francisco, United States), pp. pp. 1290–1294, Sept. 2016.

[123] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA., 1967.

[124] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means : spectral clustering and normalized cuts," in *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 551–556, ACM, 2004.

[125] S. Mouysset, J. Noailles, and D. Ruiz, "Using a global parameter for gaussian affinity matrices in spectral clustering," in *High Performance Computing for Computational Science-VECPAR 2008*, pp. 378–390, Springer, 2008.

[126] S. Afantenos, E. Kow, N. Asher, and J. Perret, "Discourse parsing for multi-party chat dialogues," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 928–937, 2015.

[127] J. Perret, S. Afantenos, N. Asher, and M. Morey, "Integer linear programming for discourse parsing," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pp. 99–109, June 2016.

[128] M. Taboada and W. C. Mann, "Rhetorical Structure Theory : looking back and moving ahead," *Discourse Studies*, vol. 8, no. 3, pp. 423–459, 2006.

[129] S. D. Afantenos and N. Asher, "Testing sdrt's right frontier," in *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, (Stroudsburg, PA, USA), pp. 1–9, Association for Computational Linguistics, 2010.

[130] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič, "Non-projective dependency parsing using spanning tree algorithms," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, (Vancouver, British Columbia, Canada), pp. 523–530, Association for Computational Linguistics, Oct. 2005.

[131] Y. i Chu and T. Liu, "On the shortest arborescence of a directed graph," 1965.

[132] J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards*, vol. 71N, pp. 233–240, 1967.

[133] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[134] W.-H. Yang, D.-Q. Dai, and H. Yan, "Biclustering of microarray data based on singular value decomposition," in *Emerging Technologies in Knowledge Discovery and Data Mining*, (Berlin, Heidelberg), pp. 194–205, Springer Berlin Heidelberg, 2007.

[135] S. Borgatti and M. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, pp. 375–395, 11 1999.

[136] J. Yang, M. Zhang, K. Ning Shen, X. Ju, and X. Guo, "Structural correlation between communities and core-periphery structures in social networks : Evidence from twitter data," *Expert Systems with Applications*, vol. 111, 12 2017.

[137] P. A. Knight and A. Aleidan, "Bipartivity Measures and Methods," in *28th Biennal Conference on Numerical Analysis*, (Glasgow), 2019.

[138] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Bipartite graph partitioning and data clustering," in *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, (New York, NY, USA), pp. 25–32, ACM, 2001.

[139] I. Duff, P. Knight, L. le Gorrec, S. Mouysset, and D. Ruiz, "Uncovering hidden block structure for clustering," Rapport de recherche IRIT/RR–2018–04–FR, IRIT, Université Paul Sabatier, Toulouse, mars 2018.

[140] R. A. Brualdi, "Convex sets of non-negative matrices," *Canadian Journal of Mathematics*, vol. 20, p. 144–157, 1968.

[141] M. Benzi, E. Estrada, and C. Klymko, "Ranking hubs and authorities using matrix functions," *Linear Algebra and its Applications*, vol. 438, no. 5, pp. 2447 – 2474, 2013.

Contributions

Conférences internationales avec actes et comité de sélection

— L. le Gorrec, S. Mouysset, I. Duff, P. Knight, and D. Ruiz, "Uncoverring Hidden Block Structure for Clustering", in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, (Würzburg), 2019

Conférences internationales avec comité de sélection

— I. Duff, P. Knight, L. le Gorrec, S. Mouysset, and D. Ruiz, "Algorithm based on spectral analysis to detect numerical blocks in matrices", in *Sparse Days*, 2017

— I. Duff, P. Knight, L. le Gorrec, S. Mouysset, and D. Ruiz, "An Algorithm Based on Spectral Analysis to Detect Block Structures on Matrices", in *SIAM Workshop on Combinatorial Scientific Computing*, (Bergen), 2018

— L. le Gorrec, S. Mouysset, and D. Ruiz, "Doubly-stochastic scaling of adjacency matrices for community detection", in *Conference on Graphs, Networks, and their Applications*, (Moscow), 2019

— L. le Gorrec, S. Mouysset, and D. Ruiz, "Doubly-stochastic scaling of adjacency matrices for community detection", in *28th Biennal Conference on Numerical Analysis*, (Glasgow), 2019

— L. le Gorrec, S. Mouysset, I. Duff, P. Knight, and D. Ruiz, "Détection automatiques de structures blocs sur des matrices", in *Conférence sur l'Apprentissage Automatique*, (Toulouse), 2019

— L. le Gorrec, S. Mouysset, and D. Ruiz, "Doubly-stochastic scaling of adjacency matrices for community detection", in *5th International Conference on Computational Social Science*, (Amsterdam), 2019

Rapports de recherche

— L. le Gorrec, S. Mouysset, and D. Ruiz, "Evaluation de la qualité des découpages en communautés dans le cas de graphes non orientés pondérés", IRIT/RR—2018—06—FR, Université Paul Sabatier, Toulouse, juin 2018

— L. le Gorrec, S. Mouysset, and S. Afantenos, "Evaluation d'un réseau BiLSTM pour l'identification des actes de dialogue par des méthodes de clustering", IRIT/RR-–2019-–04-–FR, Université Paul Sabatier, Toulouse, avril 2019

— I. Duff, P. Knight, L. le Gorrec, S. Mouysset, and D. Ruiz, "Uncoverring hidden block structure for clustering", IRIT/RR—2018-–04—FR, Université Paul Sabatier, Toulouse, mars 2018

Résumé : La détection de structures par blocs dans les matrices est un enjeu important. D'abord en analyse de données, où les matrices sont classiquement utilisées pour représenter des données, par exemple via les tables de données ou les matrices d'adjacence. Dans le premier cas, la détection d'une structure par blocs de lignes et de colonnes permet de trouver un co-clustering. Dans le second cas, la détection d'une structure par blocs diagonaux dominants fournit un clustering. En outre, la détection d'une structure par blocs est aussi utile pour la résolution de systèmes linéaires car elle permet, notamment, de rendre efficace des préconditionneurs type Block Jacobi, ou de trouver des groupes de lignes fortement décorrélés en vue de l'application d'un solveur type Block Cimmino.

Dans cette thèse, nous centrons notre analyse sur la détection de blocs diagonaux dominants par permutations symétriques des lignes et des colonnes. De nombreux algorithmes pour trouver ces structures ont été créés. Parmi eux, les algorithmes spectraux jouent un rôle crucial, et se divisent en deux catégories. La première est composée d'algorithmes qui projettent les lignes de la matrice dans un espace de faible dimension composé des vecteurs propres dominants avant d'appliquer une procédure de type k-means sur les données réduites. Ces algorithmes ont le désavantage de nécessiter la connaissance du nombre de classes à découvrir. La deuxième famille est composée de procédures itératives qui, à chaque itération, cherchent la k-ième meilleure partition en deux blocs. Mais pour les matrices ayant plus de deux blocs, la partition optimale en deux blocs ne coïncide en général pas avec la véritable structure. Nous proposons donc un algorithme spectral répondant aux deux problèmes évoqués ci-dessus. Pour ce faire, nous prétraitons notre matrice via un équilibrage bi-stochastique permettant de stratifier les blocs. D'abord, nous montrons les bénéfices de cet équilibrage sur la détection de structures par blocs en l'utilisant comme prétraitement de l'algorithme de Louvain pour détecter des communautés dans des réseaux. Nous explorons aussi plusieurs mesures globales utilisées pour évaluer la cohérence d'une structure par blocs. En adaptant ces mesures à nos matrices bi-stochastiques, nous remarquons que notre équilibrage tend à unifier ces mesures.

Ensuite, nous détaillons notre algorithme basé sur les éléments propres de la matrice équilibrée. Il est construit sur le principe que les vecteurs singuliers dominants d'une matrice bi-stochastique doivent présenter une structure en escalier lorsque l'on réordonne leurs coordonnées dans l'ordre croissant, à condition que la matrice ait une structure par blocs. Des outils de traitement du signal, initialement conçus pour détecter les sauts dans des signaux, sont appliqués aux vecteurs pour en détecter les paliers, et donc les séparations entre les blocs. Cependant, ces outils ne sont pas naturellement adaptés pour cette utilisation. Des procédures, mises en place pour répondre à des problèmes rencontrés, sont donc aussi détaillées.

Nous proposons ensuite trois applications de la détection de structures par blocs dans les matrices. D'abord la détection de communautés dans des réseaux, et le préconditionnement de type Block Jacobi de systèmes linéaire. Pour ces applications, nous comparons les résultats de notre algorithme avec ceux d'algorithmes spécifiquement conçus à cet effet. Enfin, la détection des actes de dialogues dans un discours en utilisant la base de données STAC qui consiste en un chat de joueurs des «Colons de Catane» en ligne. Pour ce faire nous couplons des algorithmes de clustering non supervisés avec un réseau de neurones BiLSTM permettant de prétraiter les unités de dialogue.

Enfin, nous concluons en entamant une réflexion sur la généralisation de notre méthode au cas des matrices rectangulaires.

Mots clés : Équilibrage bi-stochastique - Méthodes spectrales - Classification non supervisée - Détection de communautés - Traitement automatique du langage naturel

Abstract : The detection of block structures in matrices is an important challenge. First in data analysis where matrices are a key tool for data representation, as data tables or adjacency matrices. Indeed, for the first one, finding a co-clustering is equivalent to finding a row and column block structure of the matrix. For the second one, finding a structure of diagonal dominant blocks leads to a clustering of the data. Moreover, block structure detection is also usefull for the resolution of linear systems. For instance, it helps to create efficient Block Jacobi precoditioners or to find groups of rows that are strongly decorrelated in order to apply a solver such as Block Cimmino.

In this dissertation, we focus our analysis on the detection of dominant diagonal block structures by symmetrically permuting the rows and columns of matrices. Lots of algorithms have been designed that aim to highlight such structures. Among them, spectral algorithms play a key role. They can be divided into two kinds. The first one consists of algorithms that first project the matrix rows onto a low-dimensional space generated by the matrix leading eigenvectors, and then apply a procedure such as a k-means on the reduced data. Their main drawbacks is that the knowledge of number of clusters to uncover is required. The second kind consists of iterative procedures that look for the k-th best partition into two subblocks of the matrix at step k. However, if the matrix structure shows more than two blocks, the best partition into two blocks may be a poor fit to the matrix groundtruth structure. Hence, we propose a spectral algorithm that deals with both issues described above. To that end, we preprocess the matrix with a doubly-stochastic scaling, which leverages the blocks. First we show the benefits of using such a scaling by using it as a preprocessing for the Louvain's algorithm, in order to uncover community structures in networks. We also investigate several global modularity measures designed for quantifying the consistency of a block structure. We generalise them to make them able to handle doubly-stochastic matrices, and thus we remark that our scaling tends to unify these measures.

Then, we describe our algorithm that is based on spectral elements of the scaled matrix. Our method is built on the principle that leading singular vectors of a doubly-stochastic matrix should have a staircase pattern when their coordinates are sorted in the increasing order, under the condition that the matrix shows a hidden block structure. Tools from signal processing-that have been initially designed to detect jumps in signals-are applied to the sorted vectors in order to detect steps in these vectors, and thus to find the separations between the blocks. However, these tools are not specifically designed to this purpose. Hence procedures that we have implemented to answer the encountered issues are also described.

We then propose three applications for the matrices block structure detection. First, community detection in networks, and the design of efficient Block Jacobi type preconditioners for solving linear systems. For these applications, we compare the results of our algorithm with those of algorithms that have been designed on purpose. Finally, we deal with the dialogue act detection in a discorsre, using the STAC database that consists in a chat of online players of « The Settlers of Catan ». To that end we connect classical clustering algorithms with a BiLSTM neural network taht preprocesses the dialogue unities.

Finally, we conclude by giving some preliminary remarks about the extension of our method to rectangular matrices.

Key words : Doubly-stochastic scaling - Spectral methods – Clustering - Community detection - Natural langage processing