HEC MONTRÉAL

Analyse comparative de méthodes de détection de communautés dans les graphes avec algorithmes exacts

Par

Frédéric Doiron

Sciences de la gestion
(Intelligence d'affaires)

Mémoire présenté en vue de l'obtention du grade de maîtrise ès sciences (M.Sc.)

Août 2016

Pour ma femme et mon fils. Sans vous je ne suis rien.

To my wife and son. Without you I am nothing.

Résumé

L'intérêt récent envers la détection de communautés dans les graphes, ou de partitionnement de réseaux en communautés, a conduit à une augmentation dramatique du nombre de méthodes afin de les identifier, mais il existe encore des lacunes quant à l'analyse de leurs performances. Généralement, un graphe destiné à servir dans une analyse comparative est généré par un algorithme avec une structure de communauté pré-intégrée ; c'est cette structure que les méthodes testées tenteront de retrouver. L'erreur entre la partition générée et celle trouvée permet d'évaluer la performance des méthodes entre-elles. Cette erreur comporte deux composantes : l'erreur du critère de partitionnement et l'erreur introduite par la solution heuristique qui est inexacte. Dans ce mémoire, nous proposons d'utiliser la génération de colonnes pour obtenir une solution exacte, et ce afin d'éliminer le biais qui peut être causé par la solution heuristique. Nous utilisons l'indice de référence LFR de Lancichinetti et al. (2008) afin de tester la modularité de Girvan et Newman (2003), la densité de modularité de Li et al. (2008) et la coupe normalisée de Shi et Malik (2000) sur des graphes de 50 sommets. Ceux-ci seront générés avec différents paramètres : le degré moyen, la distribution des degrés, la distribution de la taille des communautés ainsi que le ratio d'arêtes intra et inter-communautés. Cette méthode est générale et pourrait s'appliquer à la comparaison d'autres critères de détection de communautés dans les graphes. Les résultats démontrent que 1) la performance des trois méthodes testées dépend de la structure du graphe et 2) la modularité et la densité de modularité sont sensibles à la disponibilité de l'information sur le nombre de communautés présentes dans les graphes générés.

Mots clés : génération de colonnes, réseaux, communautés, modularité, coupe normalisée, benchmark, algorithmes exacts, heuristique, segmentation, clustering

Abstract

The recent interest in detecting communities in graphs, or partitioning of networks into communities, has led to an increase in the methods to find them, but this has not been followed with proper benchmarking to assess their performance. Typically, the community structure is built into a generated graph and compared to the one found to assess the performance of the method. This error between the generated partition and the one found is due to the criterion used as well as the heuristic solution. In this thesis, we propose to use the powerful technique of column generation to obtain exact solutions and eliminate the bias that may be caused by the heuristic. We use the LFR benchmark (Lancichinetti et al., 2008) to test the modularity (Girvan et Newman, 2003), modularity density (Li et al., 2008) and normalized cut (Shi et Malik, 2000) on graphs with 50 nodes with varying degree and community size distribution, as well as varying average degree. This method of benchmarking can be easily generalized for other criteria of community detection in graphs. Results show that the performance of all methods depend on the structure of the graph, and that modularity and modularity density are sensitive to information about the number of clusters in the generated graphs.

Key words : column generation, networks, communities, modularity, normalized cut, benchmark, exact algorithm, heuristic, segmentation, clustering

Remerciements

Ce mémoire n'aurait jamais vu le jour sans la contribution de plusieurs personnes que j'aimerais remercier. Merci à mes parents et mes soeurs pour leur support inconditionnel lors d'une des périodes les plus difficiles de ma vie, mon épouse Carole pour son amour et le plus beau cadeau que j'ai reçu de ma vie : mon fils Olivier. Je remercie également mes amis, particulièrement Alex Morgan pour son expertise en rédaction anglaise et son amitié. Enfin, je suis reconnaissant de l'aide et l'expertise en théorie des graphes de Gilles Caporossi et de mes deux directeurs, Pierre Hansen et Sylvain Perron.

Acknowledgement

This thesis would not have been possible without the contribution and help of several people. I would like to thank my parents and my sisters for their unconditional support during one of the most difficult periods of my life, my wife Carole for her love and the greatest gift I have received : my son Olivier. I also would like to thank all my friends, especially Alex Morgan for his expertise in English and his friendship. Finally, the assistance and expertise in graph theory from Gilles Caporossi and my two advisors, Pierre Hansen and Sylvain Perron.

Table des matières

Liste des abréviations

- CPLEX : IBM ILOG CPLEX Optimization Studio
- D : Densité de Modularité
- EQ : Égal (*Equal*)
- GECMI : Logiciel de calcul de l'information mutuelle normalisée
- GERAD : Groupe d'étude et de recherche en analyse des décisions
- LE : inférieur ou égal (*less than or equal to*)
- LFR : Lancichinetti-Fortunato-Radicchi
- MI : Information mutuelle (*mutual information*)
- NMI : Information mutuelle normalisée (*normalized mutual information*)
- NP : non déterministe polynomial (*nondeterministic polynomial time*)
- Ncut : Coupe normalisée (*normalized cut*)
- Q : Modularité
- RMP : Problème maitre restreint (*restrcited master problem*)
- SP : Sous-problème

Table des figures

Chapitre 1

Introduction

1.1 Contexte

La théorie des graphes permet de mieux comprendre de nombreux systèmes complexes d'objets, autant naturels qu'artificiels. En représentant les objets par leurs relations, les graphes permettent une étude plus poussée (Fortunato, 2010). Les origines de la théorie des graphes peuvent être retracées à l'article de Leonhard Euler (Leonhard, 1741), qui a prouvé qu'il n'existait pas de solution au problème des ponts de Königsberg [1]. Euler a reformulé le problème en termes abstraits, représentant chaque rive et île par un point (sommet, noeud) et chaque pont reliant les masses par une ligne (arête). Ce concept de graphe comme objet mathématique simple peut être utilisé pour représenter des interactions sociales [Freeman (1992), Zachary (1977), Scott (2000)], des réseaux de transport, des phénomènes biologiques [Chen et Yuan (2006), Zhang (2009), Lusseau et Newman (2004)] et des réseaux d'information (Eriksen et al., 2003) et permet de les étudier en utilisant des outils et des techniques communes à

1. La ville de Königsberg (aujourd'hui Kaliningrad en Russie) était traversée par une rivière qui comprenait deux grandes îles. Sept ponts reliaient les deux îles aux deux rives. Le problème des sept ponts de Königsberg consiste à trouver un chemin à travers les sept ponts sans traverser l'un d'entre eux plus d'une fois. Le problème inclut la possibilité que le point de départ soit différent du point d'arrivée.

la théorie des graphes (Bollobás, 1998).

L'une des caractéristiques les plus intéressantes d'un graphe est sa structure sous-jacente où les sommets le composant peuvent être regroupés en communautés en fonction de leurs attributs propre. Ces groupes (ou communautés) de sommets peuvent être déduits à partir de la topologie du graphe ou des caractéristiques des sommets eux-mêmes (Fortunato, 2010). Par exemple, le regroupement des sommets en fonction de la topologie pourrait être effectué en utilisant les interactions sociales entre des individus, alors qu'une segmentation par caractéristiques des sommets pourrait consister à regrouper des entrepôts dans différente villes en fonction du type de marchandises qu'ils stockent.

Un graphe est un objet simple et sans ambiguïté, mais la segmentation (aussi appelé *clustering*) de ses sommets en communautés est très complexe. Il existe de nombreuses méthodes de segmentation, certaines se fondant sur la topologie, d'autre sur les caractéristiques des sommets, ou les deux. Chaque méthode peut donner des résultats différents et être plus ou moins efficace pour différents types de graphes. Il est donc important de choisir la meilleure méthode pour segmenter un graphe donné (Fortunato, 2010).

Généralement, ces méthodes sont testées par *benchmarking* (étalonnage, analyse comparative). La méthode sujette à l'analyse comparative est appliquée à un graphe avec une structure de communauté connue a priori, puis l'erreur de la méthode est calculée à partir de la différence entre la partition connue et la partition obtenue par la méthode testée (Girvan et Newman, 2003). Les graphes utilisés sont soit des graphes observés dans le monde réel ou soit générés par un ordinateur avec une structure pré-intégrée. De ce fait, il est possible de comparer les erreurs de chaque méthode de partitionnement et de classer leur efficacité pour chaque type de graphes étudié. L'analyse comparative devient particulièrement importante lorsque la structure de la communauté, ou même le nombre de groupes, ne sont pas connus *a priori* car certaines méthodes fonctionnent mieux sur certains types de graphes que d'autres (Lancichinetti et al., 2008).

Un défi supplémentaire se pose dans la mesure de l'erreur elle-même. De nombreux algorithmes de détection des communautés sont des problèmes NP-

difficiles, et le temps requis pour les résoudre peut devenir prohibitif même pour des graphes de petite ou moyenne taille. Il devient alors nécessaire de résoudre le problème en utilisant une heuristique, ce qui a pour effet d'échanger la garantie d'une solution optimale contre la rapidité d'exécution et une solution jugée "assez bonne" (Cafieri et al., 2012). Comme la solution exacte est inconnue lorsqu'une heuristique est utilisée, il est impossible de connaître l'écart entre celle-ci et la solution heuristique, ce qui ajoute à l'erreur lors de l'analyse comparative. Cet écart entre la solution heuristique et la solution exacte peut biaiser les benchmarks : une heuristique particulièrement mauvaise pourrait pénaliser une méthode qui aurait autrement bien performé, ou encore favoriser une méthode relativement mauvaise qui utilise une bonne heuristique. Ce biais peut seulement être éliminé en trouvant la solution exacte. L'objectif de ce mémoire est de trouver une solution à ce problème en produisant des analyses comparatives sans biais heuristique. Dans le reste de ce chapitre, nous allons définir le problème, le type de graphes étudié, ainsi que des concepts de base de la théorie des graphes.

1.2 Concepts de bases en théorie des graphes

Comme nous l'avons vu précédemment, un graphe est une structure abstraite et non-visuelle qui représente un ensemble d'objets et de relations entre ces objets. Même si un graphe est non visuel, il peut être représenté comme un schéma où chaque objet est un point (noeud ou sommet) et une relation est une ligne (arête ou arc) reliant deux points. Un graphe peut être représenté visuellement de nombreuses façons différentes : comme la position des sommets ou de la longueur des arêtes est arbitraire, seule la présence ou l'absence d'arêtes entre deux sommets est importante. Nous utiliserons la notation standard pour un graphe G comme étant $G = (V, E)$, qui est composé d'un ensemble V de **sommets** et d'un ensemble E d'**arêtes**. L'ensemble V contient n sommets notés v_j, tandis que l'ensemble E est composé de m arêtes $e_{ij} = \{v_i, v_j\}$. Chaque arête e_{ij} est composée d'une paire de sommets de l'ensemble V. Deux sommets v_i et v_j sont **adjacents** si $\{v_i, v_j\} \in E$. En d'autres termes, chaque

arête e_{ij} représente la relation entre deux sommets v_i et v_j.

Un graphe avec un seul sommet est appelé un **graphe trivial**, alors qu'un graphe sans arêtes (c'est-à-dire que l'ensemble E est vide) est appelé un **graphe vide** (*edgeless graph*). Les arêtes dans un graphe peuvent être **orientées** ou **non orientées** ; dans le cas d'un graphe orienté, une relation peut exister dans une direction mais pas nécessairement dans l'autre, alors que dans un graphe non orienté la relation est symétrique. Par exemple, dans un graphe composé de deux sommets a et b, les arêtes $\{a, b\}$ et $\{b, a\}$ sont les mêmes dans le cas d'un graphe non orienté, mais différent dans un graphe orienté. Dans un graphe orienté les arêtes peuvent porter le nom d'**arcs**. Figure 1.1 montre un exemple de graphe orienté et non orienté.

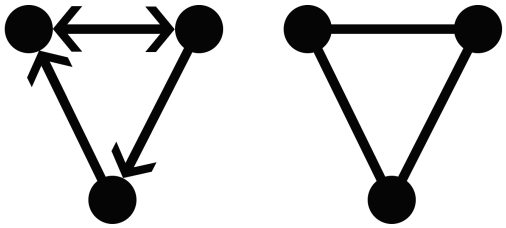

FIGURE 1.1 – Graphe orienté et non orienté

Un **graphe simple** est un graphe non orienté où une seule arête peut lier les deux mêmes sommets. Un **multigraphe** permet à plus d'une arête de relier deux sommets, mais ne permet pas les boucles (c'est-à-dire une arête reliant un des sommets à lui-même). Un **pseudographe** est un type de graphe permettant à la fois les boucles et plus d'une arête par paire de sommets. Voir Figure 1.2.

Le **degré** d'un sommet est la somme des arêtes provenant du sommet ou se terminant au sommet. Dans un graphe simple, le degré maximal d'un sommet est $n - 1$, ou une arête reliant le sommet à tous les autres sommets du graphe. La **matrice des degrés** D est une matrice diagonale n par n qui contient le degré de chaque sommet sur sa diagonale. La **densité** du graphe est représenté par le **degré moyen** : plus celui-ci est grand, plus le graphe est dense (plus d'arêtes), plus il est petit, moins le graphe est dense (moins d'arêtes). La

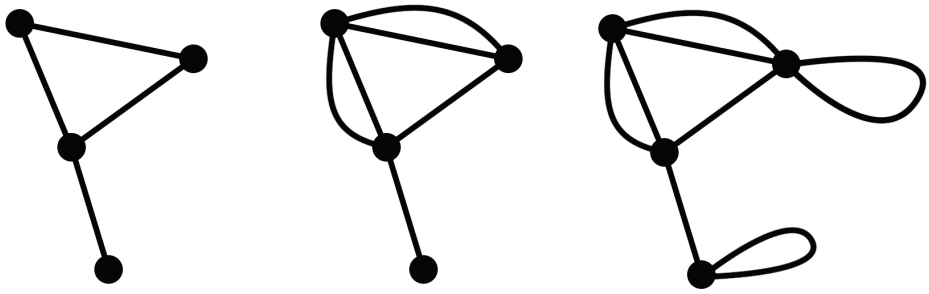

FIGURE 1.2 – Graphe simple, multigraphe et pseudographe

distance $d_G(i, j)$, ou distance géodésique, est le plus court chemin entre le sommet i et le sommet j compté en nombre d'arêtes. L'**excentricité** d'un sommet est la distance maximale entre celui-ci et tous les autres sommets. Le **diamètre** d'un graphe est l'excentricité maximale de tous les sommets du graphe.

Il est possible de représenter un graphe de façon matricielle à l'aide d'une **matrice d'adjacence** A de taille n par n. Chaque élément a_{ij} est le nombre d'arêtes liant le sommet i au sommet j. Dans le cas d'un graphe simple, la diagonale est égale à zéro (pas de boucles) et chaque élément non diagonal peut avoir une valeur de 0 ou 1. La **matrice laplacienne** L est une autre façon de représenter un graphe. Celle-ci est composée de la matrice des degrés moins la matrice d'adjacence : $L = D - A$.

Un **graphe régulier** est un graphe dans lequel chaque sommet a le même degré que tous les autres sommets du graphe. Un graphe est donc k-régulier si tous les sommets sont de degré k. Un **graphe complet** est un graphe simple où chaque sommet est lié à tous les autres sommets, c'est-à-dire qu'il contient toutes les arêtes possibles ($n(n-1)/2$, n étant le nombre de sommets). Un graphe est **connexe** s'il existe au moins un chemin entre chaque paire de sommets dans le graphe (Figure 1.3), sinon le graphe est **non connexe**. Un graphe peut être **pondéré** quand il y a une valeur numérique associé à chaque arête, appelé un poids ou pondération (Figure 1.4). Les poids dans un graphe pondéré représentent généralement des propriétés de l'arêtes telles que le coût, la capacité ou la longueur. Par exemple, si les sommets d'un graphe représentent

des villes et les arêtes représentent des voies ferrées, les poids sur les arêtes peuvent être la longueur ou la capacité de transport de chaque voie ferrée. Les graphes étudiés dans ce mémoire sont des graphes simples, connexes et non pondérés.

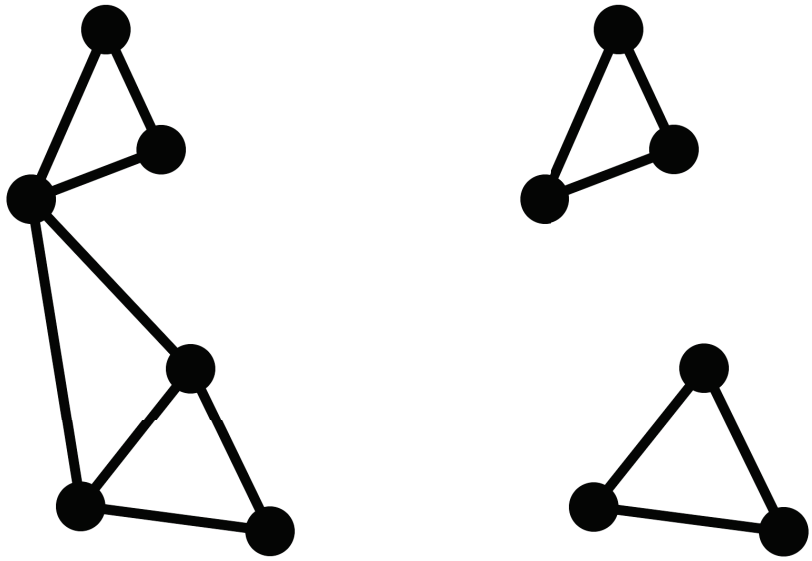

FIGURE 1.3 – Graphe connexe et non connexe

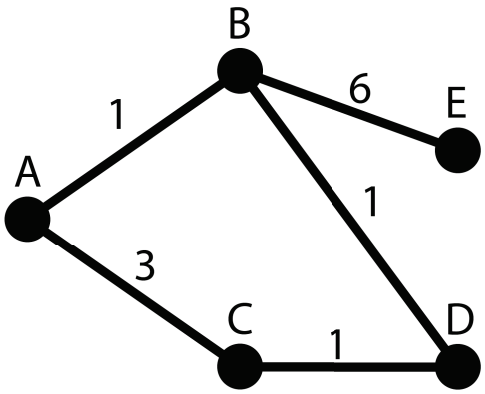

FIGURE 1.4 – Graphe pondéré

Un **sous-graphe** $G_S = (S, E_S)$ d'un graphe $G = (V, E)$ est composé d'un

ensemble de sommets S (soit un sous-ensemble de l'ensemble V) et d'un ensemble d'arêtes E_S (soit un sous-ensemble de l'ensemble E où toutes les paires d'arêtes lient des sommets contenus dans S). En d'autres termes, un sous-graphe contient certains sommets d'un graphe avec toutes les arêtes reliant ces sommets entre eux. La Figure 1.5 montre un sous-graphe G_S du graphe G. Un sous-graphe peut contenir tous les sommets du graphe ; dans un tel cas $G_S = G$.

Une **partition** d'un graphe est composée de sous-graphes disjoints deux à deux, non vides et dont l'union contient tous les sommets du graphe. En d'autres termes, le partitionnement d'un graphe en k parties divise les sommets en k sous-graphes tel que tous les sommets appartiennent à un seul et même sous-graphe. Une **communauté** est un ensemble de sommets d'un graphe. La définition de communauté est souvent contestée, mais est généralement définie comme étant un sous-graphe avec plus de relations intra-communautés que de relations inter-communautés (Fortunato, 2010). Le **degré interne** d'une communauté est le nombre d'arêtes de celle-ci, alors que le **degré externe** est le nombre d'arêtes liant les sommets de la communauté au reste du graphe. La **détection de communautés** dans un graphe consiste à trouver des structures qui répondent à un certain **critère**. Dans le cas de communautés avec chevauchement de sommets on parle de **couverture** (*cover*), où un sommet peut appartenir à plus d'une communauté à la fois. Quand les noeuds peuvent seulement appartenir à une seule communauté, la détection de communautés est un problème de partitionnement. La Figure 1.6 montre un graphe G partitionné en 4 communautés. La détection de communautés par partitionnement fera l'objet de ce mémoire.

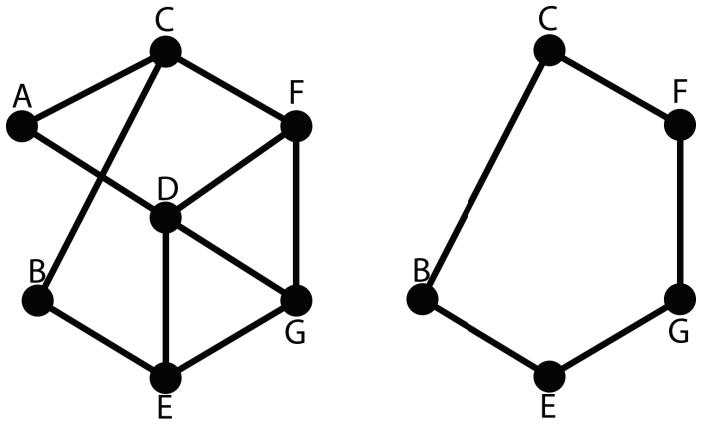

FIGURE 1.5 – Graphe et Sous-graphe

Une **clique** est un sous-graphe complet, c'est-à-dire un sous-graphe où chaque sommet dans le graphe est relié à tous les autres sommets par une arête. La clique maximale est le plus grand sous-graphe complet qui peut être retrouvé dans le graphe. En général, un **sous-graphe maximal** est un sous-graphe qui respecte une certaine condition. Par exemple, si on cherche une clique maximale on cherche en fait un sous-graphe maximal qui à la propriété d'être une clique.

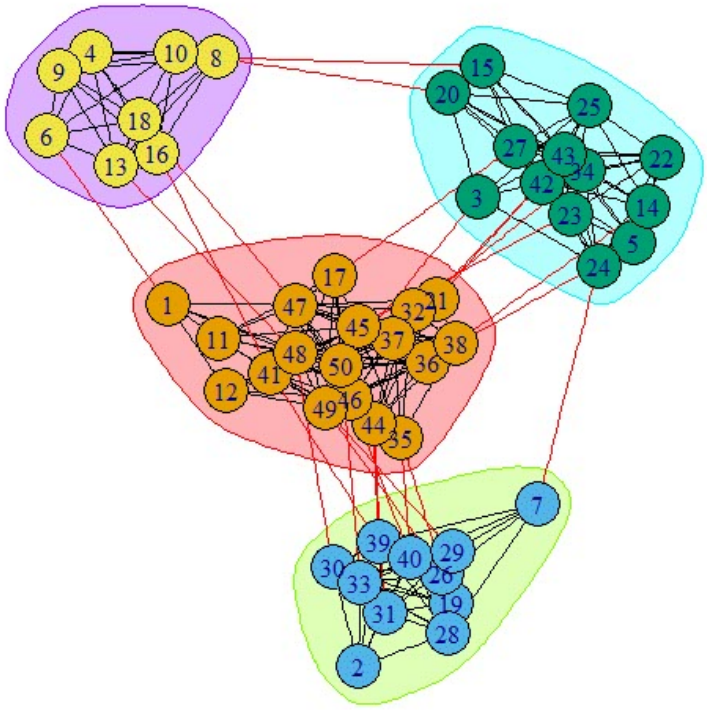

FIGURE 1.6 – Graphe divisé en communautés

Couper un graphe consiste à diviser celui-ci en deux sous-graphes. La **coupe** d'un sous-graphe représente les arêtes qui sont «coupées» et font partie de l'ensemble de la coupe : une arête couvre la coupe et appartient à l'ensemble de la coupe si ses sommets sont dans deux sous-graphes différents. La **taille de la coupe** est le nombre d'arêtes dans la coupe, ou le nombre d'arêtes qui relient le sous-graphe au reste du graphe. Si on retire les arêtes de l'ensemble de la coupe, le sous-graphe deviendrait alors déconnecté du reste du graphe.

Plusieurs critères (ou métriques) de détection de communautés prennent en compte les relations inter-communautés. Par exemple, le critère de la coupe minimum cherche à trouver une partition du graphe qui minimise la somme des coupes. D'autres critères utilisent les relations intra-communauté, par exemple un partitionnement qui maximise la densité d'arêtes à l'intérieur des communautés. Il existe une multitude de critères basée sur ces deux concepts ou sur d'autres encore, chacun ayant des caractéristiques et une performance différente sur différent types de graphes. Le chapitre 2 présente, entre autre, une

revue plus approfondie des critères de détection de communautés.

1.3 Solutions exactes et heuristiques

Le défi de trouver des communautés dans un graphe (c'est-à-dire partitionner un graphe) est difficile d'un point de vue computationnel (Cafieri et al., 2012). Une approche simple et naïve serait tout simplement d'énumérer toutes les combinaisons possibles de sous-graphes. Cependant, quand la taille du graphe augmente, le nombre de possibilités devient rapidement si grand que même les énumérer serait trop long, sans même parler d'identifier le meilleur partitionnement. Plutôt que d'utiliser cette recherche par force brute, une série d'algorithmes ont été développés pour résoudre le problème de la détection de communautés. Bien que plus efficaces, ces algorithmes sont encore soumis à des contraintes de temps et d'utilisation de mémoire informatique. Le temps et la mémoire requise peuvent devenir excessivement grands quand le nombre de sommets ou d'arêtes augmente. Il peut alors être nécessaire d'utiliser un algorithme d'approximation, ou heuristique, qui produira un résultat qui n'a pas la garantie d'optimalité, mais représente plutôt une approximation de la solution exacte (Fortunato, 2010).

Une méthode heuristique peut également être non déterministe, ce qui signifie qu'elle peut fournir un résultat différent pour le même problème lorsqu'elle est exécuté avec différents paramètres ou conditions de départ, alors qu'un algorithme exact donnera la solution exacte à chaque fois, quelles que soient les conditions de départ de l'algorithme. Bien que les heuristiques soient souvent la seule façon de procéder pour les graphes à grande échelle, il existe tout de même des méthodes exactes qui peuvent être utilisées sur des graphes de petite et moyenne taille.

La complexité d'un algorithme représente la quantité de ressources, comme le temps ou l'espace mémoire, utilisées pour le résoudre. L'approche la plus commune est de calculer le temps de calcul dans le pire des cas. Par exemple, pour un graphe de n sommets et m arêtes, un algorithme composé de $24n^3 + 2m^2 + 12$ étapes sera noté comme $O(n^3)$, seul le terme le plus significatif (n^3)

est utilisé. Un algorithme de complexité $O(n)$ croit de façon linéaire avec le nombre de sommets, alors qu'un algorithme de complexité $O(m^3)$ croit de façon cubique avec le nombre d'arêtes.

1.4 Problématique

Il existe une multitude de critères et d'algorithmes heuristiques ou exacts en recherche de communautés dans les graphes (Fortunato, 2010). Comme les communautés ne sont pas connues d'avance, il est difficile de savoir si un critère performe bien ou non. La difficulté majeure en détection de communautés est non seulement le développement d'algorithmes performants, mais également le choix d'un critère qui mesure la qualité d'une partition. Tester les différentes méthodes dans une analyse comparative, ou *benchmark*, dans laquelle les communautés sont connues *a priori* est l'une des seules façons de déterminer quel critère et quel algorithme doit être utilisé pour obtenir la meilleure partition. Ce mémoire fera valoir que, aux fins de l'analyse comparative, il est intéressant d'utiliser des méthodes exactes pour éliminer le biais heuristique discuté plus tôt. Nous allons donc utiliser des méthodes exactes basées sur la technique de génération de colonnes par programmation linéaire pour faire l'analyse comparative de différentes méthodes de partitionnement de graphes simples.

1.5 Plan du mémoire

Le chapitre 2 est consacré à la revue de la littérature. Le chapitre 3 présente la démarche méthodologique. Le chapitre 4 est composé de l'article intitulé "Column generation algorithm for exact benchmarking of community detection methods in simple graphs". L'introduction de l'article (section 4.2) est suivie de la méthodologie (section 4.4 à 4.6), les tests numériques et résultats (section 4.7) et conclusions (section 4.8). Le travail se termine par une courte conclusion générale.

Chapitre 2

Revue de la littérature

2.1 Définition de communauté

Identifier des communautés dans un graphe est un défi qui dépend en grande partie de la façon dont on définit la notion de communauté. Intuitivement, on s'attendrait à ce que les sommets se trouvant à l'intérieur de la communauté soient fortement liés entre eux, et qu'il y ait peu de liens vers le reste du graphe.

Plusieurs définitions de communautés sont basées sur l'étude des interactions sociales, comme par exemple le concept de clique formalisé par Luce et Perry (1949). Une clique est un sous-graphe maximal composé de sommets étant reliés avec tous les autres sommets du sous-graphe, formant ainsi une communauté extrêmement liée entre elle. Selon Alba (1973), la définition de clique est trop restrictive pour définir une communauté. Bien que plusieurs communautés contiennent des cliques, un sous-graphe composé de plusieurs sommets interreliés avec seulement une arête manquante pour être complet n'est pas une clique mais représente quand même une communauté très bien définie. Luce (1950) introduit la n-clique comme étant le sous-graphe maximal où chaque sommet est à une distance n ou moins de chacun des autres sommets du sous-graphe, produisant ainsi un objet moins restrictif que la clique. Mokken (1979) et Fortunato (2010) notent cependant qu'il y a deux problèmes associés à la n-clique : le diamètre d'une n-clique peut être supérieur à n, ou même infini

dans le cas d'une n-clique déconnectée, produisant ainsi des communautés sans grande cohésion interne. Le n-clan et le n-club (Mokken, 1979) tentent de résoudre les problèmes de la n-clique et sont définis respectivement comme étant une n-clique avec un diamètre égal ou inférieur à n et le sous-graphe maximal d'un diamètre n.

Fortunato (2010) differencie les définitions en utilisant les concepts de Wasserman et Faust (1994), soit la mutualité complète (*complete mutuality*), l'existence d'un chemin entre les sommets (*reachability*), le degré des sommets (*vertex degree*) et la comparaison entre la cohésion interne et externe (*comparison of internal versus external cohesion*). Le k-plex (Seidman et Foster, 1978) et le k-core (Seidman, 1983) utilisent les degrés des sommets pour identifier des communautés.

Plus récemment, Radicchi et al. (2004) ont introduit deux définitions de communautés fondées sur la cohésion interne et externe. Pour qu'une communauté soit considérée comme forte, chaque sommet de la communauté doit être connecté à plus de sommets à l'intérieur de la communauté qu'à l'extérieur, c.-à-d. le degré interne de chaque sommet doit être supérieur à son degré externe. Une communauté répond à la définition faible de Radicchi lorsque la somme de tous les degrés internes est supérieure à la somme de tous les degrés externes.

Pour une revue exhaustive des définitions de communautés, voir *Community detection in graphs* (Fortunato, 2010).

2.2 Critères globaux

Comme nous l'avons vu dans la section précédente, il n'y a pas de consensus quant à la définition de communauté. Souvent, une communauté est définie autour de la méthode utilisée pour la détecter. Dans cette section, nous passerons en revue les principales méthodes développées à ce jour dans la littérature.

Girvan et Newman (2003) ont introduit la modularité comme une fonction quantitative permettant de classer chaque façon possible de partitionner un

graphe donné. Elle est devenue très populaire au cours des dernières années, malgré ses limites (Fortunato et Barthelemy, 2007). La modularité, notée Q, repose sur l'idée qu'une communauté bien définie devrait avoir une densité plus élevée d'arêtes dans le sous-graphe qu'un sous-graphe similaire où les arêtes sont distribuées au hasard. Q peut varier de -1 à 1 inclusivement. Une valeur positive indique que la partition a une plus grande densité d'arêtes qu'aurait prédit un graphe aléatoire. En tant que tel, une partition avec une plus grande valeur de Q indique une «meilleure» partition qu'une partition avec une valeur de Q inférieure. Comme la modularité est une fonction quantitative, elle peut être maximisée jusqu'à un optimum, mais ce problème est NP-complet (Brandes et al., 2006) et est donc souvent résolu avec des heuristiques. Newman (2006) écrit la modulrité pour une seule communauté s sous la forme :

$$Q_s = [a_s - e_s]$$

où a_s est la proportion des arêtes inter-communauté de la communauté et e_s est la valeur espérée de la même quantité dans le graphe aléatoire. La modularité pour le graphe entier est simplement la somme des modularités de toutes les communautées s :

$$Q = \sum_s Q_s = \sum_s [a_s - e_s]$$

En gardant la sommation sur l'ensemble de la partition, nous pouvons ré-écrire la modularité sous la forme :

$$Q_s = \left[\frac{L(V_s, V_s)}{L(V, V)} - \left(\frac{L(V_s, V)}{L(V, V)} \right)^2 \right],$$

où $L(V_s, V_s)$ est deux fois le nombre d'arêtes du sous-graphe G_s, $L(V, V)$ est deux fois le nombre d'arêtes du graphe G et $L(V_s, V)$ est la somme des degrés des sommets du sous-graphe G_s (Li et al., 2008).

Il existe une variété d'algorithmes afin de maximiser la modularité et nous en

verrons quelques-uns. Le premier étant un algorithme glouton par Girvan et Newman (2003), où chaque communauté est réunie de façon à ce que chaque jonction successive augmente la valeur de Q. D'autres travaux par Clauset et al. (2004) ont amélioré cet algorithme hiérarchique, lui permettant d'exécuter avec une complexité de $O(n \log n)$ où n est le nombre d'arêtes. Danon et al. (2006) notent que l'algorithme glouton tend à favoriser les communautés de plus grande taille aux dépens des communautés de plus petite taille. Ils notent que les communautés détectées dans les graphes réels ont souvent des tailles hétérogènes et proposent donc une normalisation qui tente d'éliminer ce biais pour les grandes communautés. Un autre algorithme glouton a été développé par Blondel et al. (2008) pour utilisation sur des graphes pondérés. Guimera et al. (2004) ont adapté l'algorithme de recuit simulé (Kirkpatrick et al., 1983) à la modularité, et Duch et Arenas (2005) ont utilisé l'heuristique d'optimisation extrémale développé par Boettcher et Percus (2001). L'optimisation extrémale permet d'obtenir la même précision que la méthode de recuit simulé avec une moins grande complexité (Fortunato, 2010). Finalement, Aloise et al. (2010) ont comparé la performance de plusieurs méthodes exactes de maximisation de la modularité par génération de colonnes.

Bien que l'optimisation de la modularité soit devenue très populaire, Fortunato et Barthelemy (2007) ont identifié certains problèmes associés à cette méthode. Comme la modularité dépend d'un modèle nul aléatoire, elle est sensible à la taille du graphe. En effet, si le réseau devient assez grand, le nombre attendu d'arêtes entre deux communautés du modèle nul peut devenir inférieur à un. Dans ce cas, la présence d'une seule arête entre les deux communautés forcera la jonction des communautés par l'algorithme car le gain en modularité sera très grand. Fortunato et Barthelemy (2007) concluent que, dépendamment de la taille du graphe et du degré moyen, la modularité peut échouer à détecter certaines petites communautés qui seraient autrement bien définies.

En raison de ces limites de résolution, un certain nombre de méthodes alternatives à la modularité ont vu le jour depuis. Li et al. (2008) on introduit la densité de modularité, ou valeur D, qui n'utilise pas le concept de modèle nul. Elle est définie comme étant la somme, sur toutes les communautés, de la dif-

férence entre les degrés internes et externes des sommets dans la communauté, divisée par la taille de la communauté. La valeur D peut être écrite comme suit :

$$D = \sum_s D_s = \sum_s \left[\frac{L(V_s, V_s) - L(V_s, \bar{V}_s)}{|V_s|} \right],$$

où $L(V_s, V_s)$ est deux fois le nombre d'arêtes du sous-graphe V_s, $L(V_s, \bar{V}_s)$ est le nombre d'arêtes inter-communauté de V_s et $|V_s|$ le nombre de sommets de V_s (Li et al., 2008).

Afin de résoudre les limites de la modularité, Li et al. (2008) ont ajouté un paramètre λ qui peut être variable. Bien que Li et al. (2008) affirment que la densité de la modularité ne divise pas les cliques, Costa (2014) montre que cela peut être le cas pour certains exemples, et propose une modification basée sur la définition faible de communautés de Radicchi et al. (2004). Cette modification garantit également qu'aucune communauté avec une densité de modularité négative puisse exister, ce qui était possible dans la formulation originale de Li et al. (2008). Costa (2014) commente également que la preuve de Li et al. (2008) selon laquelle l'optimisation de la densité de modularité est un problème NP-complet n'est pas convenable, et que leurs résultats pour le réseau du club de karaté de Zachary (Zachary, 1977) sont erronés.

Notons également toutes les méthodes basées sur le concept de propagation d'étiquette (*label propagation*) tel que l'algorithme de **?**.

2.3 Critères basés sur la coupe

Il existe une variété de méthodes basées sur le partitionnement de graphe par la coupe, où un graphe est divisé en éléments plus petits qui forment des communautés. Nous avons précédemment défini la coupe dans la section 1.2 ; de façon mathématique la coupe $\delta(S)$ d'un sous-graphe G_s avec l'ensemble d'arêtes $S \subseteq V$ est définie comme le sous-ensemble d'arêtes $(i, j) \in E$ tel que $|\{i, j\} \cap S| = 1$. La coupe minimum (*minimum cut*) d'un graphe est la coupe

qui sépare le graphe en deux sous-graphes disjoints et contient un nombre minimal d'arêtes. La somme des coupes est la somme de la coupe pour tous les sous-graphes de G. En règle générale, l'objectif consiste à partitionner le graphe de manière à minimiser la somme des coupes :

$$\min Cut(G) = \sum_{s} \delta(s).$$

Une façon courante d'y parvenir est de couper les sous-graphes en deux de façon successive (bissection itératives) jusqu'à ce que le nombre de communautés spécifié soit atteint. L'algorithme Kernighan-Lin (Kernighan et Lin, 1970) est une telle méthode avec une complexité de $O(n^2\ log\ n)$ où n est le nombre de sommets dans le graphe. Un inconvénient majeur des méthodes de bissection itératives est que la première coupe peut affecter la qualité de la solution finale, car il est impossible de revenir en arrière et fusionner deux sous-graphes. D'autres méthodes reposent sur les propriétés du graphe dans son ensemble, telles que le spectre de la matrice d'adjacence dans le cas de la séparation spectrale ou le spectre de la matrice laplacienne dans le cas de la méthode de bissection spectrale (Barnes, 1982). Enfin, la coupe normalisée (Shi et Malik, 2000) entre deux sous-graphes avec les ensembles d'arêtes O et P est définie comme :

$$Ncut(O,P) = \frac{cut(O,P)}{assoc(O,G)} + \frac{cut(O,P)}{assoc(P,G)},$$

où $cut(O,P)$ est la coupe entre les communautés O et P du graphe G, $assoc(O,G)$ est la somme des arêtes inter-communauté de O et $assoc(P,G)$ est la somme des arêtes inter-communauté de P. En utilisant les mêmes définitions vues précédemment :

$$Ncut = \sum_{s} Ncut_s = \sum_{s} \left[\frac{L(V_s, \bar{V}_s) - \frac{L(V_s,V_s)}{2}}{L(V_s,V)} \right],$$

où $L(V_s,V_s)$ est deux fois le nombre d'arêtes du sous-graphe V_s, $L(V_s,\bar{V}_s)$ est le nombre d'arêtes inter-communauté de V_s et $L(V_s,V)$ est la somme des degrés

des sommets du sous-graphe G_s.

La coupe normalisée essaie de répondre au problème de la minimisation des coupes qui tends à favoriser des sous-graphes plus petit. La normalisation retire, en théorie, le biais pour les sous-graphes petits. L'optimisation de la coupe normalisée est un problème NP-complet (Shi et Malik, 2000).

Les méthodes basées sur le partitionnement de graphe souffrent de certains inconvénients : le nombre de communautés doit être spécifié à l'avance et, lorsque le nombre de communautés demandé est impair, la partition est obtenue en divisant l'un des groupes de deux (Fortunato, 2010).

2.4 Génération de graphes

Afin d'évaluer la performance d'un algorithme, il est nécessaire de le comparer à d'autres algorithmes de partitionnement, et ce dans une grande variété de structures de graphes. Fortunato et Barthelemy (2007) notent qu'il existe un manque de repères appropriés, ou encore que de nouveaux algorithmes sont souvent testés sur un petit nombre de problèmes de la littérature. Il note également que le choix de l'algorithme à utiliser est parfois arbitraire. Dans cette section, nous donnerons un aperçu des méthodes existantes d'analyse comparative de la littérature.

Dans de nombreux cas, l'analyse comparative est basée sur des graphes réels ; l'algorithme mis à l'essai tente de récupérer une partition qui est connue à l'avance. L'un des exemples les plus connus est le club de karaté de Zachary (Zachary, 1977) qui représente 34 membres d'un club de karaté avec 78 arêtes représentant les interactions sociales enregistrées en dehors du club par le chercheur. Au cours de l'étude, un conflit entre l'instructeur en chef et l'administrateur a eu pour effet de scinder le club en deux groupes. La Figure 2.1 illustre le graphe des interactions entre les 34 membres du club de karaté. Notons également le graphe du réseau social des dauphins par Lusseau et Newman (2004) et le graphe *American College Football* par Girvan et Newman (2002).

Il existe de nombreux autres graphes dans la littérature, mais ils souffrent

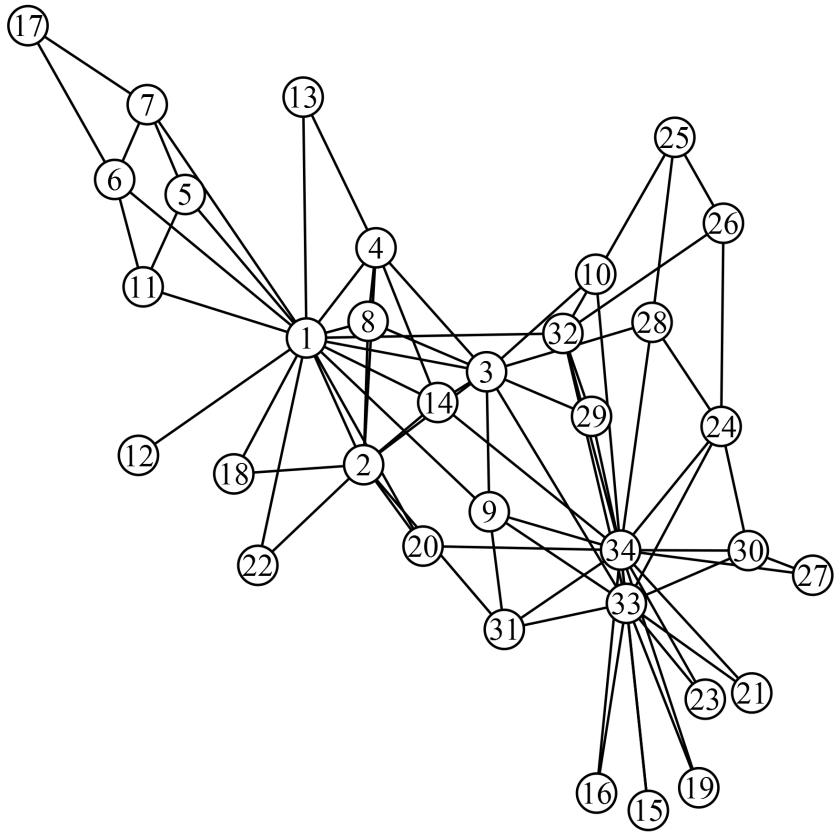

FIGURE 2.1 – Graphe du club de karaté de Zachary

tous du même problème : la structure des communautés n'est pas connue a priori car plusieurs bonnes partitions différentes peuvent exister. Il devient donc nécessaire de générer des graphes avec une structure connue à l'avance : l'algorithme mis à l'essai tentera donc de récupérer cette structure originale.

De nombreuses méthodes de génération de graphes existants sont basées sur le modèle de l-partitions plantées (*planted l-partition model*) par Condon et Karp (2001). Le modèle partitionne n sommets en l groupes ayant chacun g sommets. Les sommets d'un même sous-graphe sont inter-reliés par une arête avec la probabilité de P_{in} et liés vers l'extérieur avec une probabilité de P_{out}.

Un cas particulier du modèle de l-partitions plantées est le *benchmark* GN par

Girvan et Newman (2002) où les graphes sont générés avec 128 sommets et assignés à quatre groupes de 32 sommets chacun. Les sommets sont connectés de sorte à ce que le degré moyen est de 16. Le nombre moyen d'arêtes qui connecte les sous-graphes au reste du graphe est de Z_{out}. Lorsque $Z_{out} = 8$, chaque sommet a le même nombre de connexions intra-communauté qu'inter-communauté. Quand Z_{out} prend des valeurs de plus de 8, les communautés générées ne répondent plus à la définition faible de Radicchi et al. (2004). Le GN peut donc être utilisé pour générer des graphes avec des communautés très clairement définies (Z_{out} tend vers 0) ou des communautés moins bien définies (Z_{out} supérieur à 8). La performance d'un algorithme testé par le GN est déterminée par la fraction de sommets classifiés correctement par rapport à la partition initiale générée.

Bien que le GN soit largement utilisé, Lancichinetti et al. (2008) notent que sa structure n'est pas nécessairement similaire aux graphes observés dans le monde réel. Le degré de chaque sommet est le même que tous les autres sommets et la taille des communautés est identique, cependant les graphes réels ont souvent des degrés moyens et des tailles de communautés hétérogènes. Brandes et al. (2003) ont conçu le générateur de partition aléatoire gaussienne (*gaussian random partition generator*) comme étant une version modifiée du modèle de l-partitions plantées de Condon et Karp (2001). Dans cette version, la taille des communautés est hétérogène et suit une distribution gaussienne. Conséquemment, les degrés sont distribués de façon hétérogène car le degré attendu d'un sommet dépend du nombre de sommets dans sa communauté. Danon et al. (2005) ont modifié le benchmark de GN (Girvan et Newman, 2002), ajoutant une structure hiérarchique de deux niveaux. Le graphe est composé de 256 sommets répartis en 16 microcommunautés de taille égale et 4 macrocommunautés composées chacune de 4 microcommunautés. Ce *benchmark* utilise trois paramètres : Z_{in}^1 est le nombre attendu d'arêtes reliant un sommet à un autre sommet de sa microcommunauté, Z_{in}^2 est le nombre attendu d'arêtes reliant un sommet à un autre sommet de la même macrocommunauté alors que Z_{out} est le nombre attendu d'arêtes vers l'une des trois autres macrocommunautés. Le degré moyen est la somme des trois paramètres et est fixé à 18.

Une autre variante du modèle de l-partitions plantées est le *benchmark* LFR par Lancichinetti, Fortunato et Radicchi (Lancichinetti et al., 2008). Les auteurs supposent que le degré moyen et la taille de la communauté sont des distributions hétérogènes qui suivent une loi de puissance avec un exposant τ_1 et τ_2 respectivement. Clauset et al. (2004) et Guimerà et al. (2003) ont tous deux montré que ces distributions sont bien approximées par une loi de puissance avec les exposants $2 \leq \tau_1 \leq 3$ et $1 \leq \tau_2 \leq 2$ (Lancichinetti et al., 2008). Les graphes sont construits de la manière suivante :

(1) Choisir le degré moyen k, les paramètres τ_1 et τ_2 ainsi que le paramètre de mélange (*mixing parameter*) μ. Le paramètre de mélange est défini comme étant la fraction des arêtes inter-communauté alors que $1 - \mu$ représente la fraction des arêtes intra-communauté.

(2) Un degré est attribué à chaque sommet à partir d'une distribution de loi de puissance avec l'exposant τ_1. Les deux valeurs extrêmes de la distribution sont choisies afin que le degré moyen soit égal à k.

(3) Chaque communauté reçoit sa taille à partir d'une distribution de loi de puissance avec l'exposant τ_2 de telle sorte que la somme de la taille de toutes les communautés soit égale au nombre de sommets dans le graphe.

(4) Chaque sommet n'est initialement dans aucune communauté, ou «sans-abri». À chaque itération, un sommet est attribué à une communauté aléatoirement. Le sommet reste dans la communauté si son degré interne est plus petit que la taille de la communauté, sinon il devient sans-abri à nouveau. Ce processus se poursuit ; si une communauté est déjà pleine, un sommet aléatoire est expulsé et devient sans-abri à nouveau. Les étapes sont répétées jusqu'à ce que tous les sommets soient affectés à une communauté (c.-à-d. qu'il ne reste plus de sommets sans-abri).

(5) Le graphe passe par un processus de «recâblage» dans lequel des arêtes sont déplacées successivement. Le recâblage du graphe continue jusqu'à ce que les contraintes du paramètre de mélange μ soient satisfaites. Au cours du processus de recâblage, le degré de chaque sommet reste constant mais la proportion des arêtes internes et externes est modifiée. La proportion d'arêtes internes et externes de chaque sommet n'égale

pas nécessairement μ, mais sa distribution suit une loi normale avec un pic prononcé (voir *Benchmark graphs for testing community detection algorithms*, FIG 8 de Lancichinetti et al. (2008)).

Le LFR a été étendu pour les cas de graphes avec chevauchement des communautés, les graphes orientés et les graphes pondérés (Lancichinetti et Fortunato, 2009). Selon Lancichinetti et al. (2008), l'algorithme de création de graphe est très rapide et permet de créer des graphes de plus d'un million de sommets.

2.5 Métrique de comparaison

Chaque méthode d'analyse comparative que nous avons vu jusqu'à maintenant exige un moyen de comparer la partition originale générée à celle trouvée par l'algorithme mis à l'essai. Meila (2007) classifie les différentes mesures en trois catégories : groupe correspondant, comptage de paires et théorie de l'information. Pour une liste plus complète des mesures, voir Meila (2007) et Fortunato (2010).

Emprunté à la théorie de l'information, le concept de l'information mutuelle (*Mutual Information*, MI) est défini comme étant une mesure de la dépendance statistique entre deux variables aléatoires. En d'autres termes, elle mesure la quantité d'informations que l'on peut apprendre à propos d'une variable à partir d'une autre variable. La MI est adimensionnelle et utilise généralement le bit comme unité de mesure : une valeur élevée de la MI signifie que les deux variables partagent beaucoup d'information alors qu'une valeur de zéro signifie que les deux variables sont indépendantes (MacKay, 2005). L'information mutuelle de deux variables aléatoires X et Y peut être écrite comme :

$$I(X,Y) = \sum_{x,y} P(x,y)\ log\left(\frac{P(x,y)}{P(x)P(y)}\right),$$

où $P(x,y)$ est la fonction de distribution de probabilité jointe de X et Y, $P(x)$ et $P(y)$ sont les distributions marginales de X et Y respectivement. Dans le cadre de la comparaison de différentes partitions d'un graphe, l'information

mutuelle est utilisée de cette façon : si deux partitions sont similaires, la quantité d'informations nécessaires pour en déduire l'une de l'autre est très petite (grandes valeurs de MI), alors que si les partitions n'ont presque rien en commun, la quantité d'information requise est très élevée (petites valeurs de MI).

L'information mutuelle a ses inconvénients : selon Fortunato (2010) et Danon et al. (2005) une partition χ et une partition χ' créées en séparant par bissection certaines communautés de χ auraient la même valeur de MI, et ce même si les partitions χ et χ' sont très différentes. Danon et al. (2005) utilisent l'information mutuelle normalisée (*Normalized Mutual Information*, NMI) pour corriger ce problème :

$$I_{norm}(X,Y) = \frac{-2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} N_{ij}\log(\frac{N_{ij}N}{N_i N_j})}{\sum_{i=1}^{c_X} N_{i\bullet}\log(\frac{N_i}{N}) + \sum_{j=1}^{c_Y} N_{\bullet j}\log(\frac{N_j}{N})},$$

où le nombre de communautés trouvés par l'algorithme est c_Y, le nombre de communautés généré par le *benchmark* est c_X, la somme de la rangée i de la matrice de confusion N_{ij} est $N_{i\bullet}$, et la somme sur la colonne j est $N_{\bullet j}$. La NMI est égale à 1 si les partitions sont identiques et est égale à 0 si elles sont indépendantes.

Deux facteurs peuvent expliquer la différence entre la partition trouvée par un algorithme et la partition originale générée : le choix du critère et le biais introduit par une solution heuristique. Comme la plupart des algorithmes actuels utilisent une solution heuristique (Fortunato, 2010) nous n'avons pas trouvé beaucoup de littérature portant sur ce biais.

2.6 Solutions exactes

Il existe une grande variété d'algorithmes pour identifier des communautés dans un graphe, mais la grande majorité d'entre eux sont de nature heuristique (Fortunato, 2010). Dans cette section, nous allons donner un bref aperçu des méthodes exactes utilisées sur les critères définis précédemment.

Wu et Leahy (1993) ont appliqué la coupe minimale aux domaines de segmentation d'images et ont résolu, jusqu'à un optimum global, des graphes de 2000 sommets en utilisant l'algorithme Gomory-Hu (Gomory et Hu, 1961). Wang et al. (2008) ont proposé d'adapter un arbre à un graphe, puis de résoudre cet arbre exactement en utilisant le critère de la coupe normalisée. Grötschel et Wakabayashi (1989, 1990) ont proposé un algorithme exact pour le partitionnement utilisant le critère de la clique, qui a ensuite été adapté à la maximisation de la modularité par Aloise et al. (2010). Aloise et al. (2010) ont également utilisé la formulation de Xu et al. (2007) et ont proposé deux autres formulations utilisant la génération de colonnes et la génération de colonnes stabilisées. Brandes et al. (2008) ont formulé la maximisation de la modularité comme étant un problème de programmation en nombres entiers et ont résolu deux problèmes test jusqu'à l'optimalité.

Chapitre 3

Méthodologie

Dans ce chapitre, nous présentons le problème de partitionnement d'ensemble, la génération de colonnes de façon générale, le logiciel CLUSTOPT, l'implémentation des tests numériques ainsi que les contributions principales de l'auteur.

3.1 Formulation mathématique du problème de partitionnement d'ensemble

Si on définit $H = \{S_t \subset V : S_t \neq \emptyset\}$ comme étant tous les sous-ensembles possibles non-vides S_t de l'ensemble de sommets V, on peut representer un problème de minimisation sous la forme :

$$\min \sum_{t:S_t \in H} c_t z_t, \tag{3.1}$$

où c_t est le «coût» de l'ensemble S_t, et z_t est une variable binaire qui indique si S_t est choisi ($z_t = 1$) ou non ($z_t = 0$). En ajoutant les contraintes suivantes

où $x_{it} = 1$ si le sommet $i \in S_t$ et 0 sinon :

$$\sum_{t:S_t \in H} x_{it} z_t = 1 \quad \forall i \in V \tag{3.2}$$

$$z_t \in \{0, 1\} \quad \forall S_t \in H, \tag{3.3}$$

le problème de partitionnement consiste à trouver la combinaison de sous-ensembles S_t dont la somme des coûts c_t est la plus petite possible et dont l'union contient tous les sommets du graphe. La taille de l'ensemble H de sous-graphes non-vides possibles dépend du nombre de sommets $(2^n - 2)$ et croit de façon exponentielle, devenant rapidement trop grande. Par exemple, un graphe de $n = 300$ sommets contient environ 2×10^{90} possibilités, ce qui est plus élevé que le nombre estimé d'atomes d'hydrogène dans l'univers observable (10^{80}). Nous avons donc recours à la génération de colonnes pour résoudre ce problème.

3.2 Génération de colonnes

Cette section contient une brève introduction de la génération de colonnes. Pour plus de détails, voir Desrosiers et Lübbecke (2005).

La génération de colonnes est basée sur l'idée que, à optimalité, seulement quelques-unes des variables z_t du problème seront non négatives, donc toutes les variables ne doivent pas nécessairement être prises en considération pour résoudre le problème. En supposant un problème linéaire de minimisation et des variables continues, la génération de colonnes se déroule comme suit : un nouveau problème appelé le problème maitre restreint (*Restricted Master Problem*, RMP) est créé et contient uniquement un sous-ensemble des variables du problème d'origine. Celui-ci est résolu de façon exacte. Les valeurs duales des contraintes du RMP sont passées à un second problème appelé le sous-problème (SP) qui est ensuite résolu. Le but du sous-problème est de trouver de nouvelles variables (colonnes) à inclure dans le RMP. Lorsque la fonction objective du SP a une valeur négative, une variable ayant un coût réduit né-

gatif qui pourrait améliorer le RMP est trouvée et est ensuite ajoutée au problème maitre restreint. Le RMP est alors résolu à nouveau, générant ainsi de nouvelles valeurs duales qui sont passées au sous-problème, et le processus se répète jusqu'à ce qu'aucune nouvelle colonne ne soit trouvée par le SP. Le sous-problème peut être résolu de manière heuristique tant et aussi longtemps que des colonnes de coût réduit négatif sont trouvées. Dans le cas contraire, le sous-problème doit être résolu de manière exacte afin de prouver qu'il n'existe aucune colonne de coût réduit négatif absente du RMP courant, ou à trouver une telle colonne dans le cas où l'heuristique aurait échoué à trouver une telle colonne. Lorsque le sous problème est résolu de façon exacte et qu'aucune nouvelle colonne est trouvée, nous pouvons alors conclure que la solution actuelle du problème maître restreint est optimale.

Nous présenterons maintenant la formulation mathématique du problème maitre et du sous problème dans le cas d'un problème de partitionnement. En utilisant les équations (3.1) - (3.3) vues plus haut, si on remplace la contrainte (3.3) par

$$z_t \geq 0, \quad \forall S_t \in T, \tag{3.4}$$

La borne supérieure $z_t \leq 1$ étant déduite de la contrainte (3.2), on obtient une relaxation linéaire du problème de partitionnement. Selon le principe de la dualité lagrangienne, chaque problème linéaire est associé à un autre problème appelé le dual. Ce problème dual a le même nombre de variables que le problème original (primal) a de contraintes et autant de contraintes que le primal a de variables. Le problème dual s'écrit :

$$\begin{aligned} &\max \sum_{i=1}^{n} \lambda_i \\ S.\grave{A}. \quad &\sum_{i=1}^{n} x_{it}\lambda_i \geq c_t \quad \forall t \in T, \\ &\lambda_i \in \mathbb{R} \quad \forall i = 1 \ldots n. \end{aligned}$$

Selon le théorème de la dualité, la solution optimale du problème primal $(z_1^*, z_2^*, \ldots, z_T^*)$ et la solution optimale du problème dual $(\lambda_1^*, \lambda_2^*, \ldots, \lambda_n^*,)$ sont

identiques :

$$\sum_{t \in T} c_t z_t^* = \sum_{i=1}^{n} \lambda_i^*.$$

Si on définit la relaxation linéaire du problème initial (3.1)-(3.2),(3.4) comme étant le problème maitre (MP), le problème maitre restreint (RMP) est le même problème que le MP, mais avec un nombre plus petit de variables (colonnes). Le sous-problème (SP) est défini comme étant le problème dual du RMP.

Selon le type de critère, le problème de minimisation peut être un problème de maximisation, par exemple la maximisation de la modularité et la maximisation de la densité de modularité. Dans ce cas, les signes seront inversés. Cette formulation est présentée dans le chapitre 4.

3.3 CLUSTOPT

La résolution par génération de colonnes a été effectuée à l'aide de CLUSTOPT, un logiciel développé au GERAD [1] sous la supervision de M. Gilles Caporossi, M. Pierre Hansen et M. Sylvain Perron. Ce logiciel est un code générique pour la résolution de problèmes de partitionnement de graphes à l'aide de la génération de colonnes. Le problème maitre restreint est résolu de façon exacte avec le logiciel *IBM ILOG CPLEX Optimization Studio* (ci-après dénommé CPLEX). L'utilisateur de CLUSTOPT doit alors implanter sa méthode exacte et heuristique pour résoudre le sous-problème. Pour ce qui est des problèmes à l'étude dans ce mémoire, les itérations exactes du sous-problème sont résolues avec CPLEX dans le cas de la modularité et avec une librairie développée par Hansen et Meyer (2009) pour la résolution de problèmes quadratiques en variables binaires pour les deux autres critères. Les détails sont disponibles dans le chapitre 4. Les itérations non-exactes du sous-problème sont résolues

1. Groupe d'étude et de recherche en analyse des décisions

à l'aide d'une heuristique VNS (*variable neighborhood search*) (Mladenovic et Hansen, 1997; Hansen et Mladenovic, 2001). Dans l'éventualité où une solution est fractionnaire (pas 0 ou 1), CLUSTOPT utilise un algorithme de séparation et d'évaluation (*Branch and Bound*) de Ryan et Foster (1981) pour obtenir une solution entière. D'autres améliorations ont été ajoutées dans le logiciel, dont le prix multiple (*multiple pricing*) où tous les optimums locaux avec un coût réduit négatif trouvés par l'heuristique VNS sont ajoutés au RMP, le départ à chaud (*warm start*) où une heuristique VNS trouve une bonne solution initiale et la stabilisation de du Merle et al. (1999) pour accélérer la convergence de l'algorithme de génération de colonnes.

3.4 Mise en œuvre

Tel que décrit dans le chapitre 2, le LFR nécessite que les paramètres suivants soient fixés : le nombre de sommets n, le degré moyen k, le paramètre de mélange μ, et les exposants négatifs des distributions en loi de puissance des degrés des sommets et de la taille des communautés (τ_1 et τ_2 respectivement). Comme discuté dans la section précédente, la complexité de calcul de la génération de colonnes dépend, en grande partie, du nombre de sommets dans le graphe. Il faut donc choisir une taille de graphe qui est à la fois résoluble dans un délai raisonnable, et de taille comparable à des graphes de la littérature. Après quelques essais préliminaires, le nombre de sommets $n = 50$ a été choisi comme étant un bon équilibre entre ces deux contraintes. Bien qu'il soit possible de résoudre des graphes ayant plus de 50 sommets, le temps de calcul dépend également de la densité d'arêtes (degré moyen) du graphe ; plus le degré moyen k augmente, plus le temps de calcul devient grand. Des tests à petite échelle dans CLUSTOPT semblent confirmer ceci. Après avoir étudié une sélection de graphes de la littérature, nous avons identifié trois valeurs de degrés moyen $k = \{5, 7.5, 10\}$ qui représentent bien les caractéristiques de graphes réels d'environ 50 sommets. Cette gamme de paramètres nous permet d'examiner le comportement des critères à différentes densités d'arêtes.

Une des caractéristiques les plus intéressantes du *LFR Benchmark* est sa ca-

pacité à générer des sommets avec des degrés de différentes valeurs et des communautés de tailles variables. Dans un grand nombre de *benchmark* antérieurs, tels que le GN de Girvan et Newman (2002), le degré et la taille des communautés reste fixe, tandis que les graphes du monde réel sont souvent mieux estimés par une distribution hétérogène en loi de puissance. Clauset et al. (2004) et Guimerà et al. (2003) ont montré que des exposants de $3 \leq \tau_1 \leq 2$ et $2 \leq \tau_2 \leq 1$ pour les deux distributions correspondent le mieux aux graphes du monde réel. Nous utilisons la même gamme de paramètres que Lancichinetti et al. (2008), à savoir $\tau_1 = \{3, 2\}$ et $\tau_2 = \{1, 2\}$, pour explorer la performance de tous les critères aux extrêmes de ce qui est généralement rencontrés dans les graphes de la littérature. Le paramètre de mélange μ varie entre 0,1 et 0,6 par incréments de 0,05. Cela nous donne 12 combinaisons possibles de k, τ_1 et τ_2, chacune avec 11 valeurs de μ. Afin d'éviter que des valeurs aberrantes viennent biaiser les résultats, chaque point de données est composé de la moyenne de 50 graphes, pour un total de 6600 graphes résolus pour chacun des trois critères.

Chaque graphe est généré à l'aide du logiciel LFR de Lancichinetti et al. (2008) fourni librement en ligne sur le site de Santo Fortunato [2]. Nous avons compilé celui-ci avec gcc [3]. Les graphes sont ensuite résolus par CLUSTOPT et l'information mutuelle normalisée (NMI) est calculée en utilisant le logiciel GECMI par Esquivel et Rosvall (2012), également disponible librement en ligne [4]. Les résultats sont présentés dans le chapitre suivant.

3.5 Contributions de l'auteur

Cette section décrit brièvement les contributions de l'auteur dans l'article présenté dans le chapitre 4. Bien que les critères de la modularité et de la coupe normalisée avaient déjà été mis en place dans le code de CLUSTOPT, la densité de modularité n'était pas pleinement implémentée au début du projet.

2. http ://sites.google.com/site/santofortunato/inthepress2
3. GNU Compiler Collection
4. http ://bitbucket.org/dsign/gecmi/wiki/Home

En utilisant le code écrit par Régis Bardet, un ancien stagiaire de l'équipe, la densité de modularité a été implémentée et testée en utilisant principalement le code existant de la coupe normalisée. Les contraintes de Ryan-Foster de l'algorithme de séparation et d'évaluation, initialement absentes du code original de la densité de modularité, ont été ajoutées. Ce travail en C ++ a été fait sous la supervision et avec l'aide de M. Sylvain Perron. Ayant déterminé que la densité de la modularité était fonctionnelle à l'aide de quelques graphes faits à la main, l'auteur a étudié les différents types d'analyses comparatives de la littérature ainsi que les métriques de performance existantes. Un pipeline pour les tests numériques a également été conçu et mis en œuvre ; les détails de celui-ci sont décrits plus loin dans cette section. Grâce à l'exécution de ces problèmes dans CLUSTOPT, une série de bogues ont été découverts, en particulier dans l'algorithme de séparation et d'évaluation ; cette caractéristique de CLUSTOPT est utilisée de façon relativement rare et n'avait pas été soumise à une série de tests composés de milliers de graphes. La mise au point a été effectuée principalement par M. Perron ; l'auteur de ce mémoire a testé chaque version successive du programme avec les problèmes générés jusqu'à ce qu'aucun bogue ne puisse être trouvé dans cette phase de validation.

Comme le nombre de graphes à générer (6600) et le nombre de graphes à résoudre (19800) sont grands, une certaine automatisation a dû être mise au point. Les logiciels CLUSTOPT, GECMI et LFR mentionnés précédemment ne sont pas nativement capables d'automatisation sans une modification de leur code ; chaque itération doit être exécutée manuellement via une ligne de commande. Chaque logiciel utilise également une façon différente d'enregistrer les graphes, ce qui aggrave ce problème. Afin de résoudre ces difficultés, nous avons utilisé l'environnement R [5] pour établir un pipeline reliant les différents logiciels entre eux et automatiser les tâches de génération, de résolution et d'évaluation des graphes. Tout d'abord, l'ensemble des paramètres est spécifié dans R et les commandes pour le logiciel LFR sont écrites dans un script qui est ensuite exécuté. Les fichiers résultants sont ensuite importés dans R

5. http ://www.r-project.org

pour un traitement ultérieur. Comme le logiciel LFR peut ne pas obtenir de solutions convergentes, un mécanisme de vérification des erreurs a été codé dans R pour ré-exécuter le logiciel LFR et obtenir un graphe valide. Pour les petites valeurs de μ et de k, il est possible que le logiciel LFR génère des graphes qui sont non connexes ; nous ne savons pas si cela est un comportement visé par Lancichinetti et al. (2008) ou non. Cette vérification a été ajoutée au code R ; comme le LFR est non déterministe, le programme doit simplement être ré-exécuté jusqu'à ce que tous les graphes générés soient connexes.

Une fois les graphes vérifiés et importés, une autre fonction R convertit ceux-ci dans le format compatible avec CLUSTOPT. Les lignes de commande d'exécution de CLUSTOPT sont écrites dans un autre script et exécutées à distance en utilisant cinq ordinateurs au GERAD. Les solutions optimales pour chaque graphe sont ensuite importées dans R. Cette étape nécessitait une grande quantité de méthodes de contrôle d'erreurs, en particulier dans la phase de validation du projet. Les scripts devaient terminer automatiquement les tâches qui tournaient en boucles infinies. Diverses fonctions R ont été ajoutées afin de vérifier l'absence d'erreurs telles que les fichiers de logs invalides ou les partitions manquantes, invalides ou incomplètes. Comme plusieurs instances de CLUSTOPT étaient exécutées en parallèle, une autre couche d'automatisation a dû être ajoutée pour surveiller par SSH avec le logiciel MTPuTTY.

Les solutions optimales et les partitions générées originales ont ensuite été converties avec R dans un format compatible avec GECMI et les commandes ont été écrites dans un troisième fichier script. Encore une fois, une vérification des erreurs a été codée en R pour identifier les cas où GECMI a échoué à générer une valeur de NMI. Un autre contrôle a été mis en place permettant d'identifier les valeurs NMI qui étaient différentes de la moyenne par une grande marge ; ces graphes en particulier ont été marqués pour une analyse plus en détail au cas où la différence était due à un bogue dans le code R ou CLUSTOPT.

La phase de validation du pipeline est composée de graphes avec les mêmes paramètres que mentionné plus haut, mais inclut également les valeurs de μ de 0.6 à 0,8. Ceci augmente la possibilité d'une solution fractionnaire nécessitant un appel à l'algorithme de séparation et d'évaluation. Dans cette phase de

validation, CLUSTOPT a calculé la solution optimale à partir d'une solution initiale aléatoire. Une fois toutes les erreurs éliminées, les graphes ont été re-générés et résolus avec les paramètres finaux ; les résultats de ceux-ci sont présentés dans le chapitre 4. Afin de gagner du temps, nous avons fourni à CLUSTOPT la partition générée par LFR comme solution initiale. Nous avons pris soin de vérifier que les résultats finaux ne différaient pas significativement de ceux de la phase de validation, ce qui pourrait indiquer que l'utilisation de la partition générée en tant que solution initiale aurait faussé les résultats. Comme attendu pour un algorithme exact comme la génération de colonnes, ce ne fut pas le cas.

Chapitre 4

Column generation algorithm for exact benchmarking of community detection methods in simple graphs

Column generation algorithm for exact benchmarking of community detection methods in simple graphs

Frédéric Doiron*

HEC Montreal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7

Pierre Hansen† and Sylvain Perron‡

GERAD and HEC Montreal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7

(Dated: September 24, 2016)

Abstract

The recent interest in detecting communities in graphs has led to an increase in the methods to find them, but this has not been followed with proper benchmarking to assess their performance. Typically, the community structure is built-in a generated graph and compared to the one found to assess the performance of the method. The error between the generated partitions and the partitions found is due to the criterion used as well as the heuristic solution. We propose to use the powerful technique of column generation to obtain exact solution and eliminate the bias that may be caused by the heuristic. We use the LFR benchmark by Lancichinetti et al. [Physical Review E **78**, 4 (2008)] to test the modularity by Girvan & Newman [Physical Review E **69**, 026113 (2004)], modularity density by Li et al. [Physical Review E **77**, 3 (2008)] and normalized cut by Shi & Malik [IEEE Trans. Pattern Anal. Mach. Intell. **22**, 8 (2000)] on graphs with 50 nodes with varying degree and community size distribution, as well as varying average degree. Results show that the performance of all methods depend on the structure of the graph, and that modularity and modularity density are sensitive to information about the number of clusters in the generated graphs.

* frederic.doiron@hec.ca

† pierre.hansen@gerad.ca

‡ sylvain.perron@gerad.ca

I. INTRODUCTION

Many complex natural and artificial systems of objects can be better understood through the study of graphs and the presence of communities (also called modules or clusters) within them. Graphs, or networks, are composed of vertices representing entities, and edges representing relations joining pairs of vertices [1–4]. Community detection, or clustering, is an important part of data analysis and data mining, providing numerous insights on a variety of topics. There exists a wide range of definitions of communities and methods of identifying them, but comparatively few ways to compare their performance against each other [5–7]. Worse yet is the possibility of the benchmark results themselves being biased by the method since, in an overwhelming majority of cases, it is a heuristic and not an exact method; a good criterion of community detection could be punished by a particularly bad heuristic solution and vice-versa. Having an exact solution with a guarantee of optimality for benchmarking solves this problem of separating possible inadequacies of the model from eventual errors resulting from the use of heuristics.

Radicchi et al. [8] defined a *community in the strong sense* as a subgraph in which all vertices have larger indegree than outdegree. As an inner edge contributes by two to the sum of the indegrees and a cut edge contributes by one to the sum of outdegrees, the number of inner edges in a community in the weak sense must be at least as large as half the number of cut edges. As cut edges contribute to the sum of degrees of two communities, this definition entails that for the network as a whole the number of inner edges is larger than the number of cut edges. Recently, the *community in the weak sense* definition has been expanded in several ways. One may consider the difference for each community of the sum of indegrees and the sum of outdegrees. Then summing these contributions for all communities gives a multiway cut problem. The *normalized cut* aims to minimize the sum of cuts between two communities divided by the sum of inter-community edges [9]. Another approach is to normalize the contribution of each community by dividing it by its number of vertices [10]. The resulting function, to be maximized, is called *modularity density*.

Alternatively, contributions of communities may be divided by their number of edges [11]. Finally, one may consider maximizing, in a divisive hierarchical method, the minimum ratio of the number of edges in a community divided by the number of cut edges [12].

A different, and currently mainstream, approach was inaugurated by Newman and Girvan also in 2004 [13]. They propose to find a partition of V which maximizes the sum over all modules of the number of inner edges minus the expected number of such edges assuming that they are drawn

at random with the same distribution of degrees as in G.

This paper will use exact column generation algorithms to benchmark three criteria found in the literature: modularity density, normalized cut and modularity. The following section contains a brief overview of the definitions and notations in graph theory. The column generation algorithm is presented in Sec.III. The reformulation into column generation problems of the three criterion under study is presented in Sec.IV. Sec.V presents the *LFR benchmark* and its parameters as well as the *normalized mutual information* metric. Numerical tests and results are discussed in Sec.VI. Conclusions are drawn in Sec.VII.

II. DEFINITIONS AND NOTATIONS

We denote a graph $G = (V, E)$, where V is a set of of n vertices and E a set of m edges. Set E contains edges $e_{ij} = \{v_i, v_j\}$ representing a line joining a pair of vertices v_i and v_j; only the presence or absence of the line is important, its shape is irrelevant. If there is at most one edge between any two vertices then the graph is simple, otherwise it is a multigraph. A loop $e_{ii} = \{v_i, v_i\}$) is an edge linking a vertex v_i to itself. The degree k_i of v_i is the number of edges incident with v_i. The adjacency matrix $A = (a_{ij})$ is a $n \times n$ matrix such that $a_{ij} = 1$ if vertices v_i and v_j in V are joined by an edge and equals 0 otherwise.

A vertex-induced subgraph $G_s = (S, E_s)$ of a graph $G = (V, E)$ is a graph with vertex set $S \subseteq V$ and edge set E_s equal to all edges whose endpoints are both in S. A partition of V is composed of induced subgraphs that are nonempty, pairwise disjoint and their union contains all vertices in V. Many heuristics and exact algorithms aim at finding a partition of clusters (or communities, or modules). One usually seeks clusters which contain more intra-community edges (with both vertices in the same cluster) than cut edges (with vertices in different clusters). The degree k_i of v_i can be split in two: the indegree ${k_i}^{in}$ or number of neighbors within its community and the outdegree ${k_i}^{out}$ or number of neighbors outside its community.

III. COLUMN GENERATION

Column generation is a powerful technique of linear programming which allows for exact solutions of linear programs with a number of columns exponential in the size of the input (there may be billions of them and, in some cases, many more). To this effect, it follows the usual steps of the

simplex algorithm, with the exception of finding an entering column with a negative reduced cost (in case of minimization) which is done by solving an auxiliary problem. The precise form of this last problem depends on the type of problem under study. It is often a combinatorial optimization or a global optimization problem. It can be solved heuristically as long as a column with a reduced cost of the required sign can be found. When this is no longer the case, an exact algorithm for the auxiliary problem must be applied either to find a column with the adequate reduced cost sign, undetected by the heuristic, or to prove that there is no such column and hence the linear programming relaxation is solved. Column generation has proven to be very useful in the solution of large clustering problems, e.g., minimum sum-of-squares clustering [14–16] or exact modularity maximization [17]. For clustering problems with an objective function additive over the clusters, the columns correspond to all subsets of V, i.e., to all nonempty modules.

Define X an $n \times u$ matrix where $x_{it} = 1$ if vertex v_i belongs in module t and 0 otherwise. The problem of dividing the network G into u modules $T = \{t_1, t_2, \ldots, t_u\}$ with $0 < \sum_{i=1}^{n} x_{it} < n$ can be written as

$$\min \sum_{t \in T} c_t z_t \tag{1}$$

$$\text{subject to} \sum_{t \in T} x_{it} z_t = 1 \qquad \forall i = 1, \ldots, n \tag{2}$$

$$z_t \in \{0, 1\} \qquad \forall t \in T. \tag{3}$$

Constraints (2) expresses that each entity must belong to one and only one module and (3) that modules must be selected entirely or not at all. If the integrality constraints (3) are replaced by

$$z_t \geq 0, \forall t \in T, \tag{4}$$

the upper bound $z_t \leq 1$ being implied by constraint (2), one obtains a *relaxation* of (1) - (3) which is a linear program.

Recall that to any primal linear program is associated another linear program called its dual. This dual program has as many variables as the primal has constraints and as many constraints as the primal has variables.

The dual of the relaxation of (1)-(3),(4) can be written

$$\max \sum_{i=1}^{n} \lambda_i \tag{5}$$

$$\text{s.t.} \sum_{i=1}^{n} x_{it}\lambda_i \geq c_t \qquad \forall t \in T, \tag{6}$$

$$\lambda_i \in \mathbb{R} \qquad \forall i = 1 \ldots n. \tag{7}$$

The objective function (5) of the *dual* problem (5)-(7) is equal to the sum of all dual variables. The constraints (6) express that the sum of dual variables associated with the entities of any community must be at least as large as its primal solution value. Finally, the constraints (7) express the fact that the dual variables are unrestricted in sign.

From the duality theorem of linear programming, the optimal solutions $(z_1^*, z_2^*, \ldots, z_T^*)$ of the primal and $(\lambda_1^*, \lambda_2^*, \ldots, \lambda_n^*,)$ of the dual have the same value:

$$\sum_{t \in T} c_t z_t^* = \sum_{i=1}^{n} \lambda_i^*. \tag{8}$$

We define the problem (1)-(2),(4) as the *master problem*. This relaxation of the clustering problem can in principle be solved by a linear programming package such as CPLEX. However, for larger networks, the number of variables may become so large that we cannot even explicitly state them all. To solves instances where n is large we resort to column generation, where the columns correspond to all subsets of V, i.e., to all nonempty modules. We therefore us a *restricted master problem* (RMP) which contains only a subset of variables (columns) from the master problem. Columns that contribute to improving the objective function are said to be of *negative reduced cost* and are added to the problem as needed in each iteration. The reduced cost of a column t is defined as :

$$\text{Reduced cost} = c_t - \lambda_0 - \sum_{i=1}^{n} x_{it}\lambda_i$$

where λ_i are the current values of the dual variables of the continuous relaxation of the restricted master problem. In column generation an iteration consists of:

- solving the RMP to determine the dual and objective function values
- finding, if it exists, column(s) with a negative reduced cost and adding it to the RMP.

Such column(s) with negative reduced cost in case of minimization are found by solving the *auxiliary problem* which is another minimization problem. If the optimal solution of the auxiliary problem is zero, no other negative reduced columns can be found. In the case of a maximization problem, the signs are simply inverted and one seeks a column with *positive reduced cost*.

It is well known that column generation algorithms suffer from slow convergence particularly when the optimal solution is *degenerate*, i.e., when such a solution has many variables equal to 0, which is the case for clustering problems. Column generation algorithms also suffer from the plateau effect, i.e., the optimal solution keeps the same value for several or many iterations [18]. To alleviate these defects, one can use a variant of the stabilization methods for column generation due to du Merle et al. [19], which we previously called *focussed column generation* [17].

IV. CRITERION AND REFORMULATION

A. Modularity

The modularity approach was introduced in 2004 by Newman and Girvan [13]. They propose to find a partition of V which maximizes the sum over all modules of the number of inner edges minus the expected number of such edges assuming that they are drawn at random with the same distribution of degrees as in G. In [13] the following precise definition of modularity is given:

$$Q = \sum_s Q_s = \sum_s [a_s - e_s]$$

where a_s is the fraction of all edges that lie within module s and e_s is the expected value of the same quantity in a graph in which the vertices have the same expected degrees but edges are placed at random. For a single cluster s, we can write the modularity as:

$$Q_s = \left[\frac{L(V_s, V_s)}{L(V, V)} - \left(\frac{L(V_s, V)}{L(V, V)} \right)^2 \right],$$

Where $L(V_s, V_s)$ is twice the number of edges in V_s, $L(V, V)$ is twice the number of edges of G and $L(V_s, V)$ is the sum of degrees of all nodes in V_s [10]. A maximum value of Q near to 0 indicates that the network considered is close to a random one (barring fluctuations), while a maximum value of Q near to 1 indicates strong community structure. Observe that maximizing modularity gives an optimal partition together with the optimal number of modules. In [17] we have compared four algorithms for modularity maximization, two based on reduction to clique

partitioning, while the other two are based on a direct formulation using a mixed integer convex quadratic programing model proposed by Xu, Tsoka and Papageorgiou [20]. The mixed integer column generation algorithm appeared to be the best choice, we will therefore use it in this paper. From [17], the auxiliary problem is close to the formulation of Xu et al. [20] where a single community is determined at a time:

$$\begin{aligned}
\max_{y\in\mathbb{B}^n, D\in\mathbb{R}} \quad & \sum_r \frac{y_r}{m} - \left(\frac{D}{2m}\right)^2 - \sum_{i=1}^{n} \lambda_i x_i \\
\text{s.t.} \quad & D = \sum_i k_i x_i \\
& y_r \leq x_i \qquad\qquad \forall r = \{i,j\} \in E \\
& y_r \leq x_j \qquad\qquad \forall r = \{i,j\} \in E,
\end{aligned}$$

Where edges are indexed by r and vertices by i (or j), m denotes the number of edges in the module and D is the sum of degrees k_i of the vertices in the module. Variable y_r is equal to 1 if edge r belongs to the community which maximizes the objective function and to 0 otherwise. Similarly, x_i is equal to 1 if the i^{th} vertex belongs to the community and 0 otherwise. The objective function is equal to the modularity of the community to be determined minus the scalar product of the current value λ_i of the dual variables times the indicator variable x_i. This is a mixed integer quadratic problem with $n + m$ binary variables and 1 continuous variable, in the objective function, subject to $2m + 1$ linear constraints. There is a single nonlinear term which is concave, in the objective function.

B. Normalized Cut

Introduced by Shi and Malik [9], the normalized cut is a measure of goodness based on the cut :

$$Ncut(O,P) = \frac{cut(O,P)}{assoc(O,V)} + \frac{cut(O,P)}{assoc(P,V)}$$

where $cut(O,P)$ is the cut between partition O and P, $assoc(O,V)$ is the total connections from nodes in O to all the nodes in the graph V and $assoc(P,V)$ is the same for partition P. Using the same notation as before:

$$Ncut = \sum_s Ncut_s = \sum_s \left[\frac{L(V_s, \bar{V_s}) - \frac{L(V_s,V_s)}{2}}{L(V_s,V)} \right],$$

where $L(V_s, \bar{V}_s)$ is the sum of vertex outdegrees of s, $L(V_s, V_s)$ is twice the number of edges in V_s, and $L(V_s, V)$ is the sum of degrees of all nodes in V_s. We can also write the Normalized cut problem as :

$$\min_{x \in \mathbb{B}^n} \sum_{t \in T} \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij} x_{it}(1 - x_{jt})}{\sum_{i=1}^n k_i x_{it}},$$

where A the adjacency matrix, k_i the degree of node i and X the $n \times m$ assignment matrix where $x_{it} = 1$ if node i is in cluster t and 0 otherwise. With $\lambda_1, \ldots, \lambda_n$ the dual values of the master problem, the column generation auxiliary problem can be written as:

$$\min_{x \in \mathbb{B}^n} \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i(1 - x_j)}{\sum_{i=1}^n k_i x_i} - \sum_{i=1}^n \lambda_i x_i - \lambda_0,$$

where x_i is a binary variable that equals 1 if the node i belongs in the cluster and 0 otherwise, λ_i are the n dual variables associated with constraint (2) and λ_0 is the dual variable associated with the constraint on the number of clusters. The first term auxiliary problem is quadratic non-convex function on a linear function while the second is a linear function. Equivalently:

$$\min_{x \in \mathbb{B}^n} \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i(1 - x_j) - \sum_{i=1}^n \lambda_i x_i \sum_{i=1}^n k_i x_i - \lambda_0 \sum_{i=1}^n k_i x_i}{\sum_{i=1}^n k_i x_i}.$$

Therefore the auxiliary problem consists in checking if the following problem admits a strictly negative optimal solution:

$$\min_{x \in \mathbb{B}^n} \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i(1 - x_j) - \sum_{i=1}^n \lambda_i x_i \sum_{i=1}^n k_i x_i - \lambda_0 \sum_{i=1}^n k_i x_i.$$

After simplifications, we obtain an unconstrained, non-convex quadratic 0-1 program.

C. Modularity Density

Introduced by Li et al. [10], modularity density, or D value, aims to address the well known resolution limits of modularity [21]. Using the same definitions as with the modularity, the D value can be written as:

$$D = \sum_s D_s = \left[\frac{L(V_s, V_s) - L(V_s, \bar{V}_s)}{|V_s|} \right],$$

where $L(V_s, \bar{V}_s)$ is the sum of vertex outdegrees of s and $|V_s|$ is the number of vertices in V_s. It can be formulated as an integer nonlinear programming model [10]:

$$D = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i(1-x_j)}{\sum_{i=1}^{n} x_i},$$

where $a_{ij} = 1$ if vertex v_i and vertex v_j are adjacent and 0 otherwise. $x_i = 1$ if vertex i is in the cluster t and equals 0 otherwise. Similarly, $x_j = 1$ if vertex j is in the cluster t and zero otherwise. As with the normalized cut, the column generation auxiliary problem for modularity density can be written as:

$$\max_{x\in\mathbb{B}^n} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i(1-x_j)}{\sum_{i=1}^{n} x_i} - \sum_{i=1}^{n}\lambda_i x_i - \lambda_0.$$

The auxiliary problem of the modularity density consists in checking if the following problem admits a strictly negative optimal solution:

$$\max_{x\in\mathbb{B}^n} 2\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i - \sum_{i=1}^{n}\lambda_i x_i \sum_{i=1}^{n} x_i - \lambda_0 \sum_{i=1}^{n} x_i.$$

Similarly to the normalized cut, this is an unconstrained, non-convex quadratic 0-1 program.

D. Heuristic and exact solution

In all three cases, the auxiliary problem is first solved with a *Variable Neighborhood Search* (VNS) heuristic [22, 23] as long as a column with a positive (negative) reduced cost can be found. VNS is a metaheuristic, i.e., a framework for building heuristics, based on the idea of systematic change of neighborhood during the search. It explores progressively larger neighborhoods of the incumbent (or best known) solution in a probabilistic way. Therefore, often favorable characteristics of the incumbent will be kept and used to obtain promising neighboring solutions. VNS applies a local search routine repeatedly to get from these neighboring solutions to local optima. When this is no longer the case, an exact method is called to find such a column or prove that there are no more. For the modularity, this is done by CPLEX, while for the normalized cut and the modularity density this is done by a library for unconstrained 0-1 quadratic programming developed by Hansen and Meyer [24]. If the optimal solution of the relaxed problem is integer, which is often the case, it corresponds to a partition maximizing the objective function. Should the integrality constraint (3) not be met, a branch-and-bound is applied to compute the optimal integer solution.

V. BENCHMARK

A. Generating graphs with built-in community structure

In order to test the three criteria against one another it is necessary to compare them in a wide variety of graph structures; this is achieved by generating graphs using a computer with a built-in community structure that each criterion will attempt to recover. Many existing methods of generating graphs are based on the *planted l-partition model* by Condon and Karp [25]. The model partitions n vertices into l groups containing g vertices each. Vertices within the same group g are linked by an edge with the probability P_{in} and linked to another group with the probability P_{out}. One special case of the planted l-partition model is the GN benchmark by Girvan and Newman [26] where graphs are generated with 128 vertices and assigned to four groups of 32 vertices each. Brandes et al.[27] designed the *Gaussian random partition generator* as a modified version of the planted l-partition model where the size of communities are heterogeneous and follow a Gaussian distribution. A further modification of the planted l-partition model is the *LFR Benchmark* by Lancichinetti et al.[6]. For a thorough review of available benchmarks, see [5]. In the LFR, the degree of each node is taken from a power law distribution with exponent τ_1 while the size of each community is taken from power law distribution with exponent τ_2. The *mixing parameter* μ determines the fraction of inter-community edges (the cut), $1-\mu$ the fraction of intra-community edges, while the average degree of the generated network is k.

As the LFR benchmark attempts to emulate features found in real-world graphs, we will use it for our comparison. In smaller networks for a combination of low μ and low average degree "k" it is possible for the LFR algorithm to generate disconnected graphs; we have chosen to exclude these and keep only connected graphs in our analysis.

B. Comparison metrics

Once the generated graphs are solved, the solution obtained from each criteria must be compared to the original generated partition. To do this, a comparison metric must be used. Meilă [28] distinguishes three categories of metrics: comparing by counting pairs, comparing by set matching and the measured based on information theory. See [28] and [5] for a more complete list.

The *mutual information* (MI) is a useful concept from information that is defined as a measure of the mutual dependence between two random variables; a high value of MI means the two

variables share a lot of information while a value of zero means the two variables are independent [29]:

$$I(X,Y) = \sum_{x,y} p(x,y)\ log\left(\frac{p(x,y)}{p(x)p(y)}\right),$$

where $P(x,y)$ is the joint probability distribution of X and Y, $p(x)$ and $p(y)$ the marginal probability distribution of X and Y. An obvious problem with the MI as a metric for comparing two partition is that it will declare partition A and A' to be the same if A' is composed of the same clusters as A with some of them divided into sub-clusters [5].

Danon et al. [30] proposed the use of *normalized mutual information* (NMI) as a reliable metric for comparing partitions in clustering [6, 7, 31]. When comparing two partitions X and Y, the NMI can be written as:

$$I_{norm}(X,Y) = \frac{-2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} N_{ij}\log(\frac{N_{ij}N}{N_iN_j})}{\sum_{i=1}^{c_X} N_{i\bullet}\log(\frac{N_i}{N}) + \sum_{j=1}^{c_Y} N_{\bullet j}\log(\frac{N_j}{N})},$$

where c_x and c_y are the number of clusters of X and Y respectively, the sum over the row i of the confusion matrix N_{ij} is $N_{i\bullet}$, the sum over the column j of N_{ij} is $N_{\bullet j}$. The NMI varies between 1 and 0 inclusively; an NMI of 1 means that the partitions are identical and a value of 0 means that they are totally independent [30]. An obvious advantage of the normalized mutual information is that comparison between two partitions with different numbers of clusters is possible. Due to the normalized mutual information's flexibility and accuracy we will use it as a metric.

VI. NUMERICAL TESTS AND RESULTS

The benchmark of modularity, density of modularity and the normalized cut was done on networks generated by the LFR software[1][6] with $n = 50$ nodes, an average degree of $k = \{5, 7.5, 10\}$, exponents $\tau 1 = \{2, 3\}$, $\tau 2 = \{1, 2\}$ and mixing parameter $\mu = \{0.10, \ldots, 0.60\}$ in 0.05 increments. As the number of nodes n is fixed, the number of communities depends on their sizes which are in turn determined by the power law distribution. The partitions found by the column generation algorithm are then compared to the partitions generated by the benchmark and the normalized mutual information (NMI) is calculated using the implementation by Esquivel and Rosvall[2] [32]. The column generation algorithm is the same as used in [17], but with the added

[1] Available online on Santo Fortunato's website: http://sites.google.com/site/santofortunato/inthepress2

[2] The GECMI software is available online: http://bitbucket.org/dsign/gecmi/wiki/Home

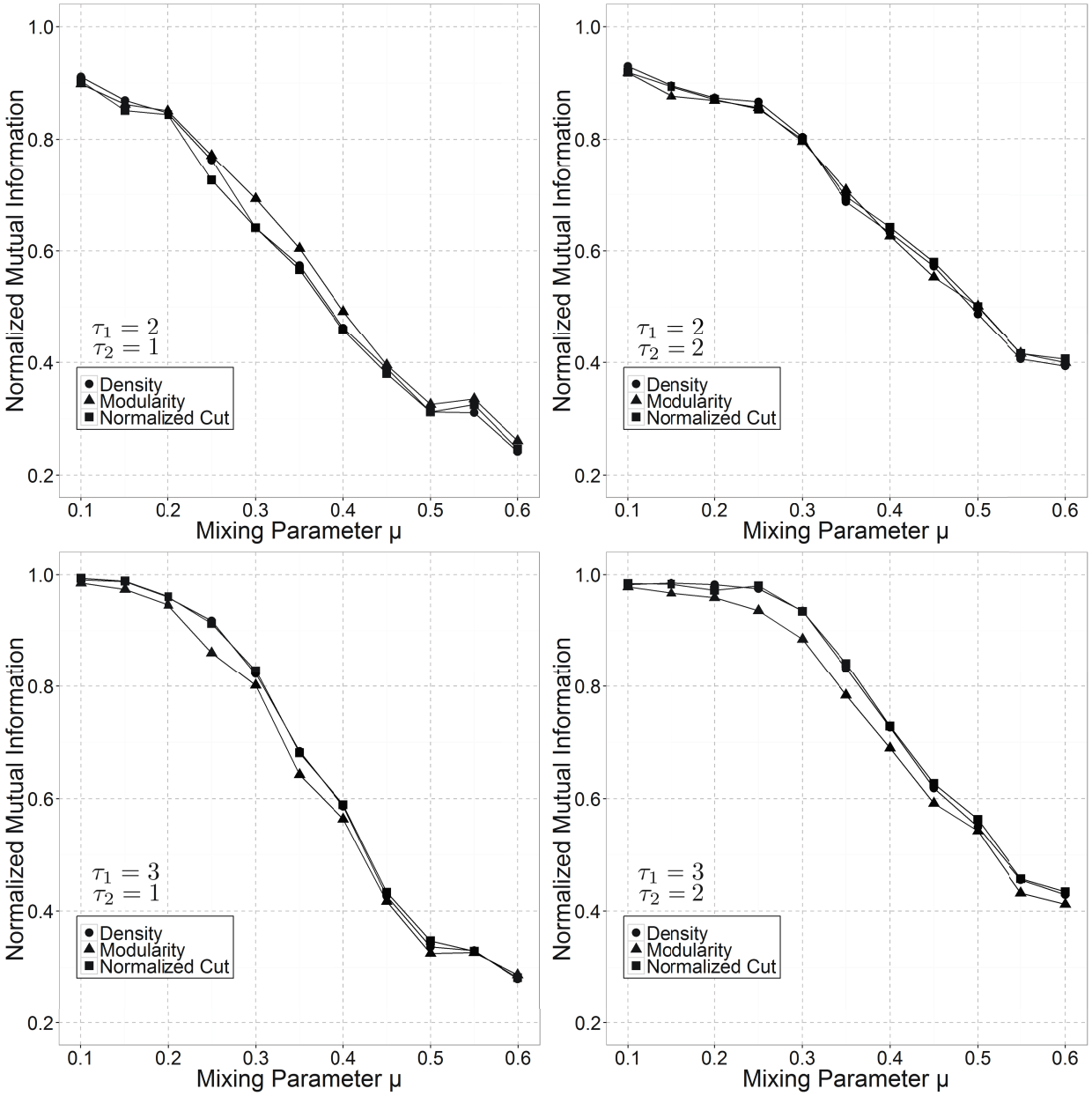

Figure 1: Results of the benchmark of all three algorithms for all values of $\tau_1 = 3$ and $\tau_2 = 1$ with the average degree $k = 5$

resolution of the normalized cut and modularity density auxiliary problems and was implemented in C++. The precision set for the column generation algorithm is of $1\mathrm{e}{-4}$ and the linear programs are solved using IBM ILOG CPLEX Optimization Studio 12.6.1.

Figure 1 plots the normalized mutual information of each criterion for all values of μ when the average degree $k = 5$ and for all values of $\tau_1 = \{2, 3\}$ and $\tau_2 = \{1, 2\}$. Each corner shows a different combination of the power law exponents; for the case where both the node degrees

and community size is more heterogeneously distributed ($\tau_1 = 2$, $\tau_2 = 1$), the case where both are more homogeneously distributed ($\tau_1 = 3$, $\tau_2 = 2$) as well as the cases where only one is heterogeneous but the other is homogeneous: ($\tau_1 = 2$, $\tau_2 = 2$) and ($\tau_1 = 3$, $\tau_2 = 1$). The normalized mutual information is the metric of how well each of the three criterion recovered the partition built-in by the LFR benchmark; higher values indicate that the criterion performed better than a lower value. An NMI of 1 means that the recovered partition and the original partition are identical. As the mixing parameter μ increases, the communities are less well defined; it can be interpreted as a "difficulty" parameter. Each data point in the figure represent the average of 50 graphs solved for that particular combination of parameters and criterion.

Fig. 2 and 3 mirror Fig. 1 for the cases with larger average degrees ($k = 7.5$ and $k = 10$). Modularity maximization does not require a number of clusters to be specified in advanced unlike the modularity density and normalized cut; we have therefore added a parameter that restricts all three methods to the number of clusters generated by the LFR. Figures 1 to 3 use the equal (EQ) values.

From the figures, it appears that:

i. All methods work better as the graph becomes more densely connected (the average degree k increases). Results are particularly poor in the case where $k = 5$. The LFR algorithm attempts to generate the graph so that each node is set to the value μ, but this is impossible for all nodes in general. Instead, the distribution of μ values is distributed as a bell curve around the specified value of μ. The peak of the curve is more pronounced as the average degree k increases. See Fig. 3 in *Benchmark graphs for testing community detection algorithms* by [6]. In the case where $k = 5$ the community structure is not very clear, even at low values of μ. This indicates that a sparse graph is more difficult to correctly partition than a dense one.

ii. For $k = 7.5$ and $k = 10$ (Figures 2 and 3) there is a significant drop in the normalized mutual information when the mixing parameter $\mu > 0.4$. Recall that when $\mu > 0.5$ the communities generated no longer satisfies the definition of a *community in the strong sense* by Radicchi et al. [8].

iii. The normalized cut outperforms the other two methods in almost all cases when $k > 5$ and the number of clusters is known *a priori*.

iv. The modularity performs worse of the three in most cases, especially in the range $0.25 >$

$\mu > 0.4$ with the exception of Figure 1 when $\tau_1 = 2$ and $\tau_2 = 1$.

v. The modularity density generally ranks second when $\mu \leq 0.4$. In the range $0.4 > \mu \leq 0.6$ the drop in NMI becomes more pronounced.

vi. As the degree distribution gets more homogeneous (τ_1 increases) for $k = 5$ the three methods perform better. This is also true for $k = 7.5$ and $k = 10$ when $\mu \leq 0.35$ and $\mu \leq 0.4$ respectively. Heterogeneity of degrees is an important feature of real world graphs, but we see that the three criteria studied perform worse when this is true. This shows the importance of using benchmarks that emulate the real world better.

vii. All methods yield better results as the communities get closer to each other in size; as τ_2 increases, the power law distribution of community size gets steeper. This seems to indicate that a graph with more heterogeneously distributed community sizes is more difficult to partition accurately.

We then removed the restriction of equality to the generated number of clusters on modularity and allowed it to be less or equal (LE). Figure 4 shows the results of modularity for LE compared to EQ. Figure 5 shows the same LE and EQ comparison for modularity density. From the figures, we can see that:

i. In nearly all cases, the modularity performs best (higher values of *normalized mutual information*) when the number of communities is known and set *a priori* (EQ). The difference in NMI between LE and EQ also tends to increase as the average degree k decreases.

ii. In the case of modularity density EQ performs better than LE when $\tau_1 = 3$ (the degrees are more homogeneously distributed) and $\mu \leq 0.4$. It is not exactly clear what impact the distribution of the cluster size (τ_2) has on the difference between the equal (EQ) and less or equal (LE) cases. In general, for the case $k = \{7.5, 10\}$ EQ performs better when $\mu \leq 0.35$.

This seems to indicate that the performance of modularity and modularity density maximization is sensitive to the information on the number of clusters; in almost all cases the results are improved by restricting both methods to the number of clusters generated by the benchmark.

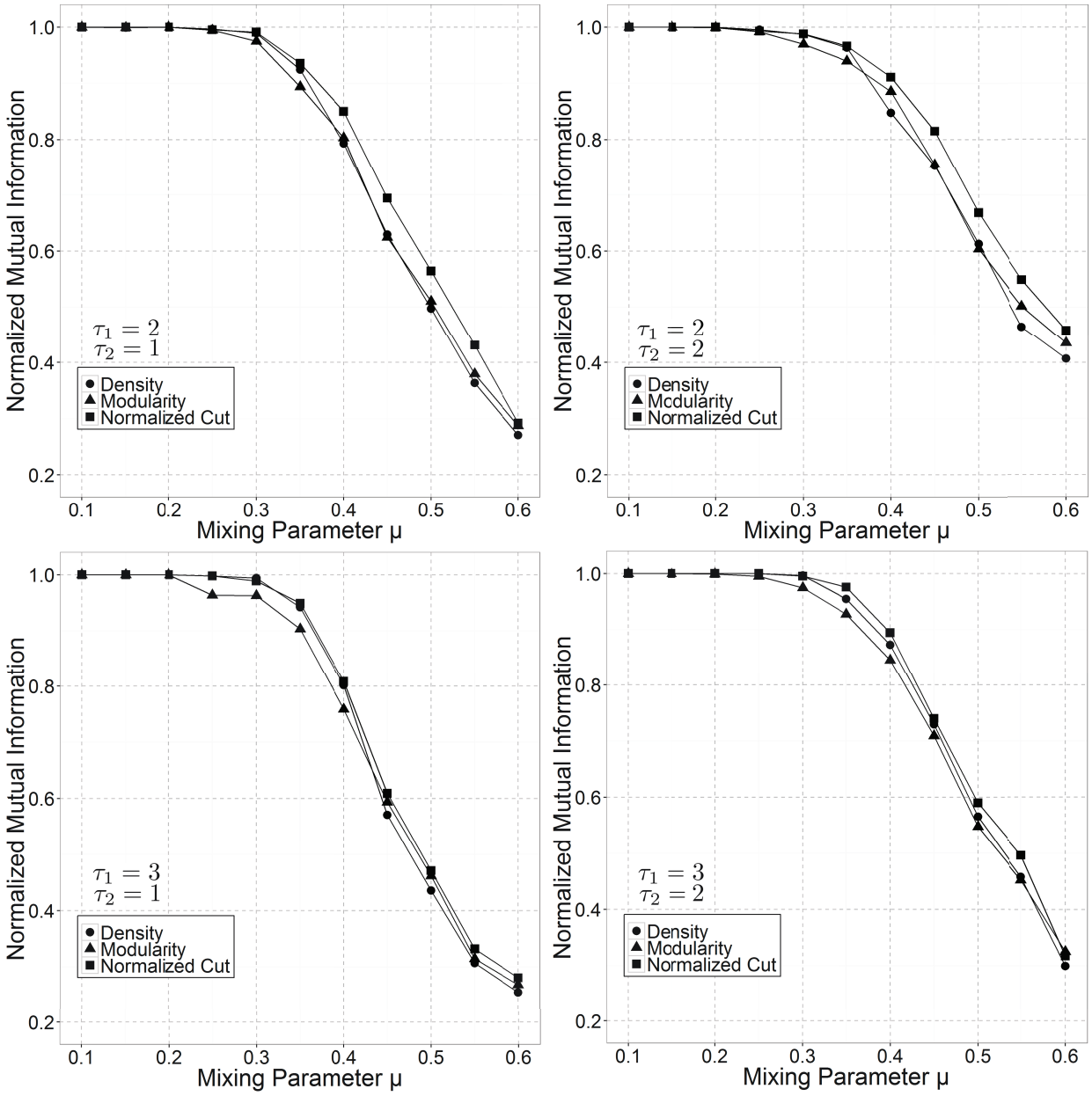

Figure 2: Results of the benchmark of all three algorithms for all values of τ_1 and τ_2 with the average degree $k = 7.5$

VII. CONCLUSIONS

In this paper, we have proposed two new exact column generation algorithms for the maximization of modularity density by Li et al. [10] and minimization of the normalized cut by Shi and Malik [9]. We have tested these methods of community detection as well as the maximization of modularity by Newman and Girvan [13] using the LFR benchmark by Lancichinetti et al. [6]

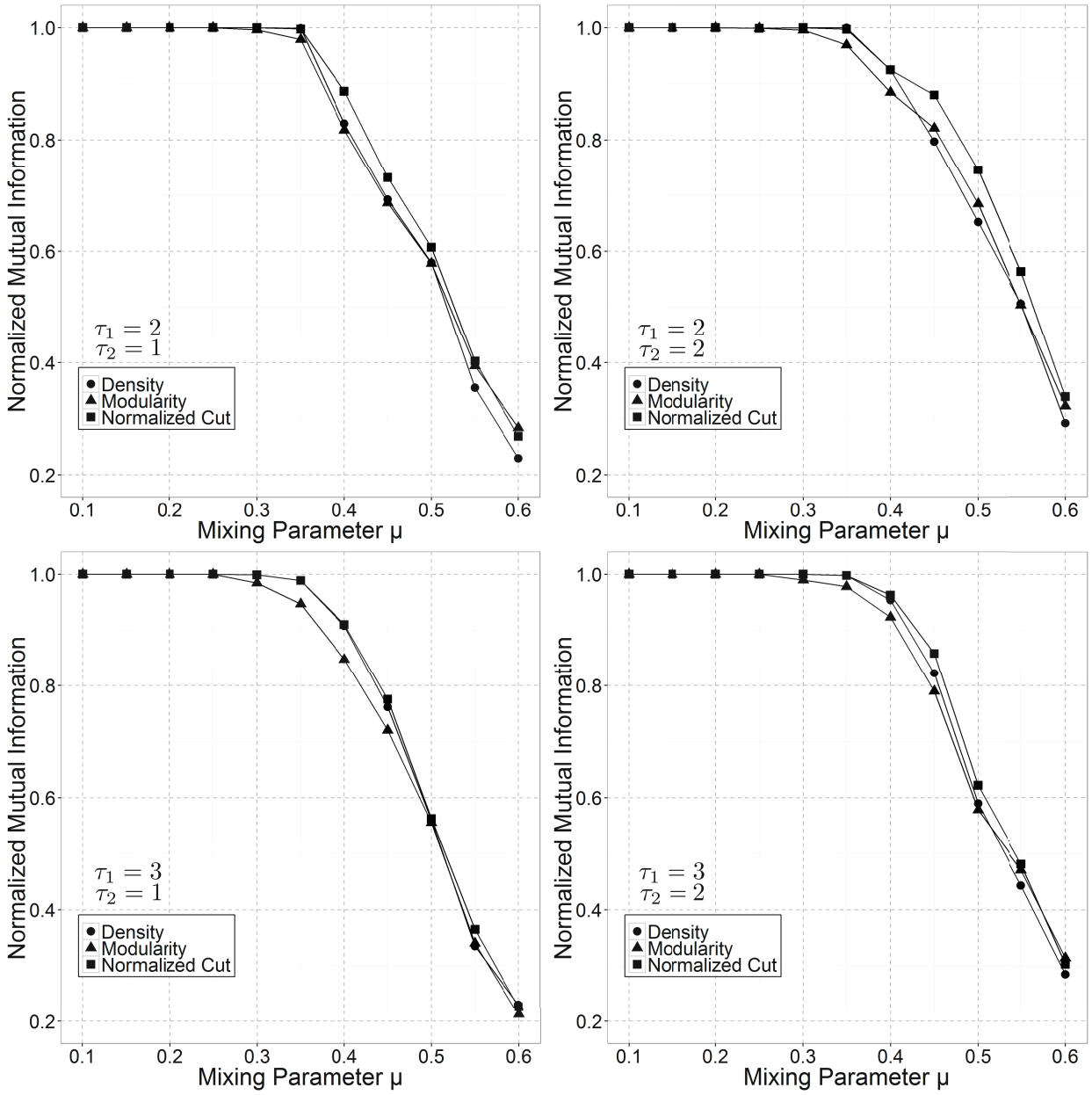

Figure 3: Results of the benchmark of all three algorithms for all values of τ_1 and τ_2 with the average degree $k = 10$

and the Normalized Mutual Information as a metric on graphs of varying average degree, degree distribution and community size distribution. The use of exact algorithms allows a comparison which does not depend on the choice of heuristic, isolating the criterion themselves from this bias. This approach of using the LFR benchmark and column generation could be used for any criterion that is calculated as the sum of a value for each community; the only significant difficulty resides in the formulation and solving of the auxiliary problem. We have showed that the performance

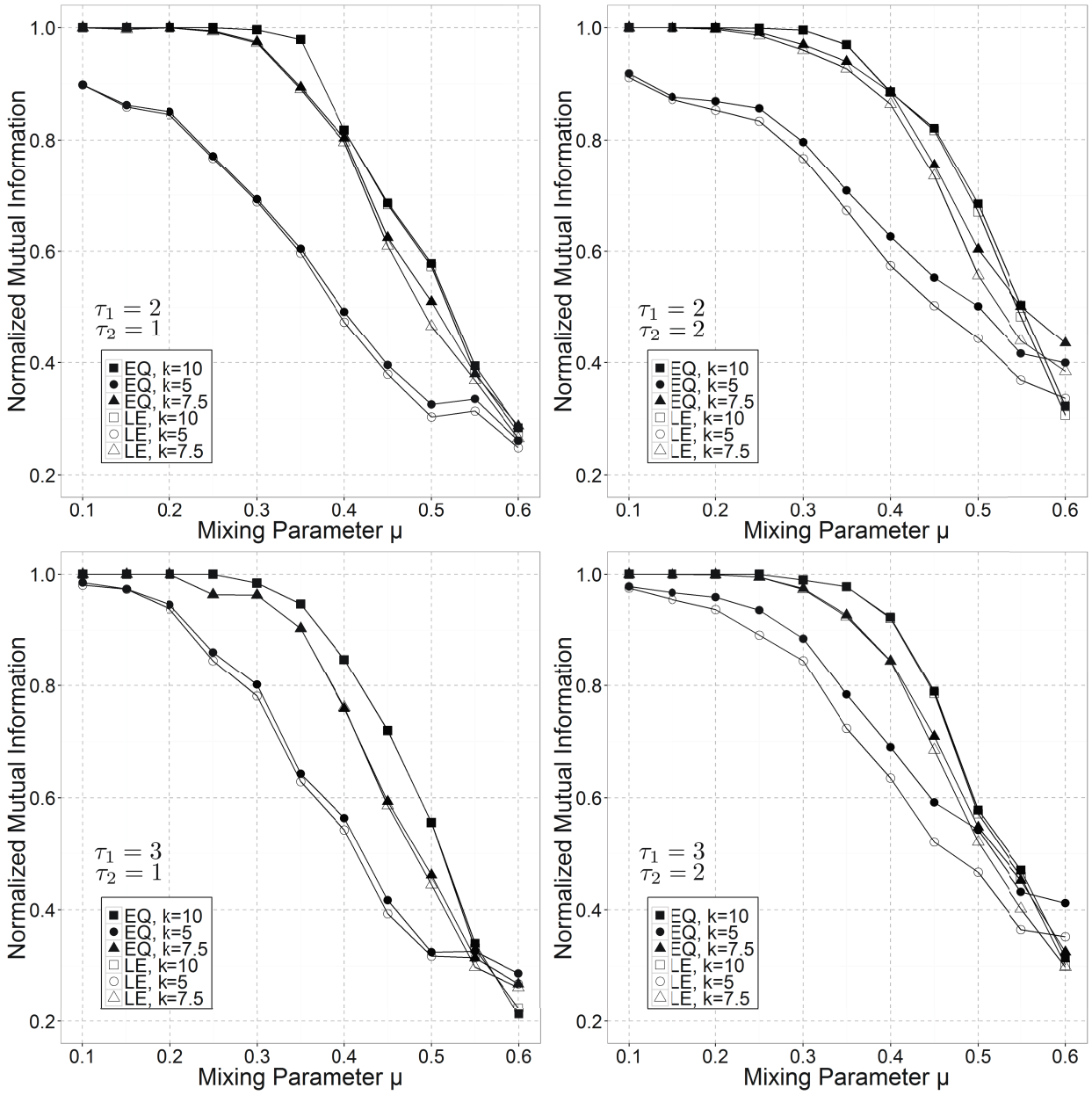

Figure 4: Results of modularity maximization for all values of τ_1, τ_2 and k when the number of clusters is equal to the generated number (EQ) and when less or equal (LE)

of the three methods depend heavily on the average degree, as well as the distribution of both the degrees and the community sizes. It is also clear that, in most cases, the modularity and modularity density yield better results when the number of clusters is known and set as a parameter *a priori*.In cases where the mixing parameter is low ($\mu < 0.35$) the performance of all three criteria are quite similar. This seems to give an advantage to the modularity maximization as it does not require the number of clusters to be specified in advance, while the modularity density and normalized cut

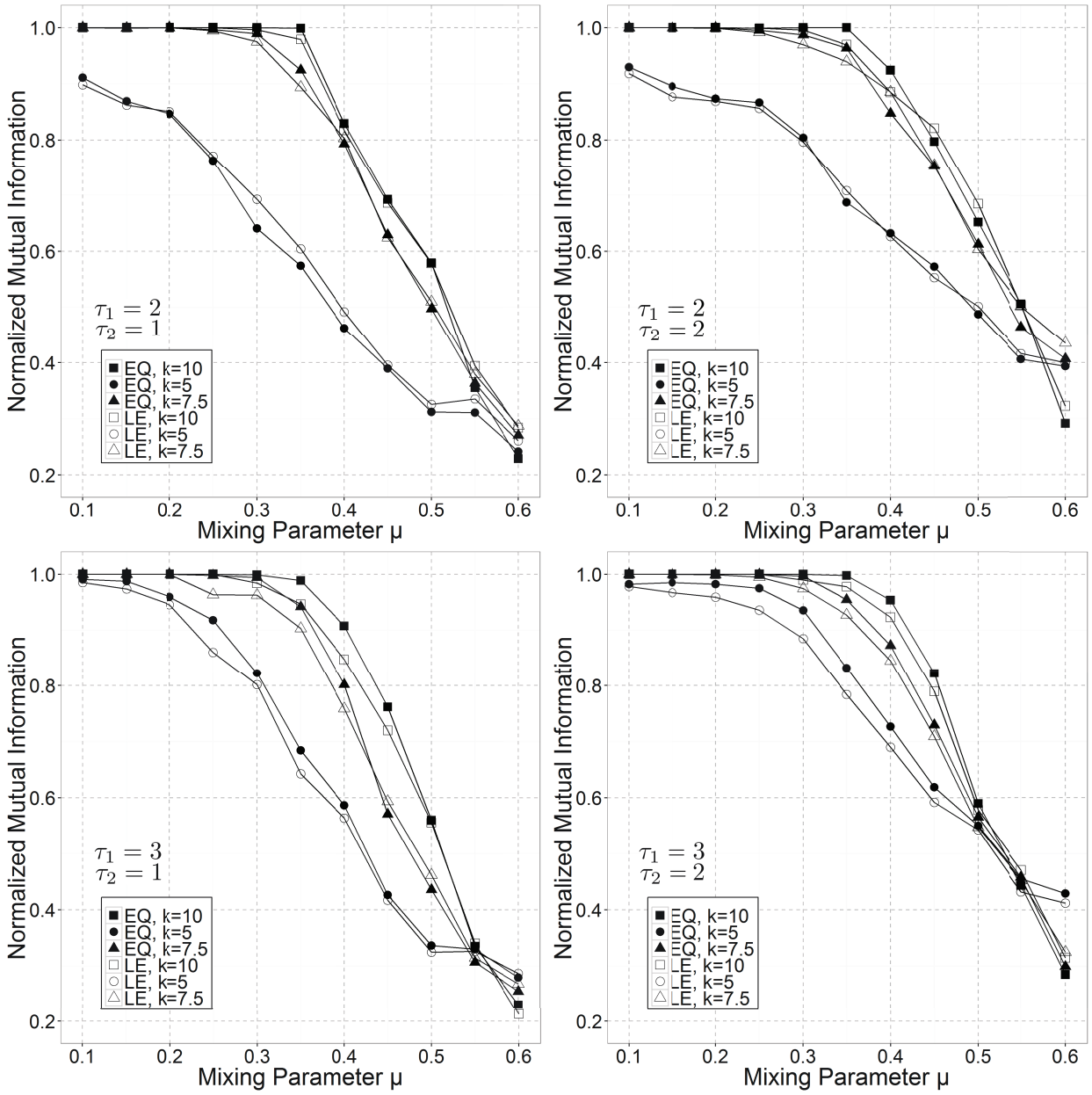

Figure 5: Results of modularity density maximization for all values of τ_1, τ_2 and k when the number of clusters is equal to the generated number (EQ) and when less or equal (LE)

both require this. Future research topics include (i) further study of the auxiliary problem that may lead to improvement and larger solvable graphs, (ii) a comparison of heuristic and exact solutions to determine the extent of the bias introduced by the heuristic solutions, (iii) further benchmarking

of additional methods of community detection and the effects of graph size on their performance.

---

[1] M. E. J. Newman, *Networks: An Introduction*. Oxford: Oxford University Press, 2010.

[2] C. Berge, *Graphs and Hypergraphs*. Amsterdam: Elsevier Science Ltd., 1985.

[3] S. Wasserman and K. Faust, *Social network analysis: methods and applications*. Cambridge: Cambridge University Press, 1994.

[4] J. L. Gross and J. Yellen, *Discrete Mathematics and Its Applications*. Chapman and Hall/CRC, 2013.

[5] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, Feb. 2010.

[6] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 78, no. 4, pp. 1–5, 2008.

[7] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, no. 1, pp. 1–8, 2009.

[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 of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004.

[9] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Ieee Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[10] Z. Li, S. Zhang, R. S. Wang, X. S. Zhang, and L. Chen, "Quantitative function for community detection," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 77, no. 3, pp. 1–9, 2008.

[11] A. D. Medus and C. O. Dorso, "Alternative approach to community detection in networks," *Phys. Rev. E*, vol. 79, p. 066111, Jun 2009.

[12] S. Cafieri, P. Hansen, and L. Liberti, "Edge ratio and community structure in networks," *Phys. Rev. E*, vol. 81, p. 026105, Feb 2010.

[13] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 026133, 2004.

[14] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, vol. 79, pp. 191–215, 1997.

[15] O. Du Merle, P. Hansen, B. Jaumard, and N. Mladenovic, "An interior point algorithm for minimum sum-of-squares clustering," *Siam Journal on Scientific Computing*, vol. 21, pp. 1485–1505, APR 27 2000.

[16] D. Aloise, P. Hansen, and L. Liberti, "An improved column generation algorithm for minimum sum-of-squares clustering," *Mathematical Programming*, vol. 131, no. 1, pp. 195–220, 2012.

[17] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti, "Column generation algorithms for exact modularity maximization in networks," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 82, no. 4, pp. 1–9, 2010.

[18] F. Vanderbeck and M. Savelsbergh, "A generic view of dantzig-wolfe decomposition in mixed integer programming," *Operations Research Letters*, vol. 34, no. 3, pp. 296 – 306, 2006.

[19] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen, "Stabilized column generation," *Discrete Mathematics*, vol. 194, pp. 229–237, 1999.

[20] G. Xu, S. Tsoka, and L. Papageorgiou, "Finding community structures in complex networks using mixed integer optimization," *Eur. Physical Journal B*, vol. 60, pp. 231–239, 2007.

[21] S. Fortunato and M. Barthélemy, "Resolution limit in community detection.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 1, pp. 36–41, 2007.

[22] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[23] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[24] P. Hansen and C. Meyer, "Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1267–1290, 2009.

[25] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures and Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[26] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[27] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on Graph Clustering Algorithms," in *European Symposium on Algorithms* (Springer Berlin Heidelberg, ed.), pp. 568–579, 2003.

[28] M. Meila, "Comparing clusterings-an information based distance," *Journal of Multivariate Analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[29] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms David J.C. MacKay*, vol. 100. 2005.

[30] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 09, pp. P09008–P09008, 2005.

[31] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips, "Tolerating the Community Detection Resolution Limit with Edge Weighting," 2009.

[32] A. V. Esquivel and M. Rosvall, "Comparing network covers using mutual information," *arXiv*, p. 8, 2012.

Chapitre 5

Conclusion

Dans ce mémoire, nous avons tenté de mesurer la performance de trois critères de détection de communautés dans les graphes simples. Nous avons identifié la possibilité d'un biais introduit par les heuristiques et avons donc cherché à éliminer celui-ci par des algorithmes exacts. Nous avons reformulé de façon non-entière la modularité de Girvan et Newman (2003), la densité de modularité de Li et al. (2008) et la coupe normalisée de Shi et Malik (2000) afin de les résoudre de façon exacte à l'aide d'un algorithme de génération de colonnes. Un algorithme de séparation et d'évaluation a ensuite été utilisé pour obtenir des solutions entières zéro-un. Ayant validé le fonctionnement de CLUSTOPT, nous avons ensuite généré une série de graphes à l'aide du logiciel LFR de Lancichinetti et al. (2008). Ces graphes sont générés avec une structure de communauté selon des paramètres comme le dégré moyen, la distribution des dégrés, la distribution de la taille des communautés et la proportion d'arêtes inter et intra-communauté. Nous avons partitionné ces graphes pour chacun des trois critères et les avons comparés à la partition d'origine avec le logiciel GECMI d'Esquivel et Rosvall (2012). Cette méthode d'analyse comparative par algorithmes exacts pourrait s'appliquer à d'autres critères de détection de communautés dans les graphes.

Nous avons montré que la performance des trois méthodes dépend fortement du degré moyen ainsi que de la distribution des degrés et de la taille des

communautés. Il est également clair que, dans la plupart des cas, la densité de modularité et la modularité donnent de meilleurs résultats lorsque le nombre de communautés est connu et défini à l'avance. Des avenues de recherche futures pourraient comprendre :

- Une étude plus approfondie du sous-problème de la génération de colonnes qui peut conduire à une amélioration du temps de calcul ainsi que de rendre possible l'analyse de graphes de plus grande taille.

- Une comparaison des solutions heuristiques et exactes afin de déterminer l'étendue du biais introduit par les solutions heuristiques.

- Une analyse comparative sur un nombre plus élevé de méthode de détection de communautés et une série de tests sur des graphes de différente taille afin de voir comment la taille d'un graphe affecte la performance des méthodes.

Bibliographie

Alba, R. D. A graphtheoretic definition of a sociometric clique. *The Journal of Mathematical Sociology*, 3(1) :113–126, 1973. ISSN 0022-250X. doi : 10.1080/0022250X.1973.9989826.

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., et Liberti, L. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4) :046112, October 2010. ISSN 1539-3755. doi : 10.1103/PhysRevE.82.046112.

Barnes, E. R. An Algorithm for Partitioning the Nodes of a Graph. *SIAM. J. on Algebraic and Discrete Methods*, 3 :541–550, 1982.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., et Lefebvre, E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 10008(10) :6, 2008. ISSN 1742-5468. doi : 10.1088/1742-5468/2008/10/P10008.

Boettcher, S. et Percus, A. G. Extremal optimization for graph partitioning. *Physical Review E*, 64(2) :026114, 2001. ISSN 1063-651X. doi : 10.1103/PhysRevE.64.026114.

Bollobás, B. *Modern Graph Theory*. Springer-Verlag, New York, 1998. ISBN 0387900357 0387900365 (pbk) 3540900365 (Berlin pbk) FG - 0. doi : 978-14419-8566-8.

Brandes, U., Gaertler, M., et Wagner, D. Experiments on Graph Clustering Algorithms. In Springer Berlin Heidelberg, editor, *European Symposium on Algorithms*, pages 568–579, 2003.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., et Wagner, D. On Modularity : NP-Completeness and Beyond. (001907) :33, 2006.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., et Wagner, D. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2) :172–188, 2008. ISSN 10414347. doi : 10.1109/TKDE.2007.190689.

Cafieri, S., Hansen, P., et Liberti, L. Improving heuristics for network modularity maximization using an exact algorithm. *Discrete Applied Mathematics*, 2012.

Chen, J. et Yuan, B. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 22(18) :2283–2290, 2006. ISSN 13674803. doi : 10.1093/bioinformatics/btl370.

Clauset, A., Newman, M. E. J., et Moore, C. Finding community structure in very large networks. *Cond-Mat/0408187*, 70 :066111, 2004.

Condon, A. et Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18 :116–140, 2001.

Costa, A. Some remarks on modularity density. *arXiv :1409.4063 [physics]*, 2014.

Danon, L., Díaz-Guilera, A., Duch, J., et Arenas, A. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, (9) :219–228, 2005. ISSN 1742-5468. doi : 10.1088/1742-5468/2005/09/P09008.

Danon, L., Diaz-Guilera, A., et Arenas, A. Effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics : Theory and Experiment*, 11010(1) :6, 2006. ISSN 1742-5468. doi : 10.1088/1742-5468/2006/11/P11010.

Desrosiers, J. et Lübbecke, M. E. A primer in column generation. *Column Generation*, 3 :1–32, 2005.

du Merle, O., Villeneuve, D., Desrosiers, J., et Hansen, P. Stabilized column generation. *Discrete Mathematics*, 194(1-3) :229–237, 1999. ISSN 0012365X.

Duch, J. et Arenas, A. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2) :2–5, 2005. ISSN 1539-3755. doi : 10.1103/PhysRevE.72.027104.

Eriksen, K., Simonsen, I., Maslov, S., et Sneppen, K. Modularity and Extreme Edges of the Internet. *Physical Review Letters*, 90(14) :5, 2003. ISSN 0031-9007. doi : 10.1103/PhysRevLett.90.148701.

Esquivel, A. V. et Rosvall, M. Comparing network covers using mutual information. *arXiv*, page 8, 2012.

Fortunato, S. Community detection in graphs. *Physics Reports*, 486(3-5) :75–174, February 2010. ISSN 03701573. doi : 10.1016/j.physrep.2009.11.002.

Fortunato, S. et Barthelemy, M. Resolution limit in community detection. *Proc Natl Acad Sci USA*, 104(1) :36–41, 2007. ISSN 0027-8424. doi : 10.1073/pnas.0605965104.

Freeman, L. C. On the sociological concept of "group" : a empirical test of two models. *American Journal of Sociology*, 98 :152–166, 1992.

Girvan, M. et Newman, M. E. J. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12) :7821–6, 2002. ISSN 0027-8424. doi : 10.1073/pnas.122653799.

Girvan, M. et Newman, M. E. J. Finding and evaluating community structure in networks. *Cond-Mat/0308217*, pages 1–16, 2003. ISSN 1063651X. doi : 10.1103/PhysRevE.69.026113.

Gomory, R. E. et Hu, T. C. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4) :551–570, 1961. doi : 10.1137/0109047.

Grötschel, M. et Wakabayashi, Y. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1) :59–96, 1989. ISSN 1436-4646. doi : 10.1007/BF01589097.

Grötschel, M. et Wakabayashi, Y. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1) :367–387, 1990. ISSN 1436-4646. doi : 10.1007/BF01580870.

Guimerà, R., Danon, L., Diaz-Guilera, A., Giralt, F., et Arenas, A. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(6) :065103+, 2003. ISSN 1063651x.

Guimera, R., Sales-Pardo, M., et Amaral, L. A. N. Modularity from fluctuations in random graphs and complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 70(2 2) :1–4, 2004. ISSN 15393755. doi : 10.1103/PhysRevE.70.025101.

Hansen, P. et Mladenovic, N. Variable neighborhood search : Principles and applications. *European Journal of Operational Research*, 130(3) :449–467, 2001.

Hansen, P. et Meyer, C. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. *Discrete Applied Mathematics*, 157(6) : 1267–1290, 2009. ISSN 0166218X. doi : 10.1016/j.dam.2007.12.008.

Kernighan, B. W. et Lin, S. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49 :291–307, 1970. ISSN 1538-7305. doi : 10.1002/j.1538-7305.1970.tb01770.x.

Kirkpatrick, S., Gelatt, C. D., et Vecchi, M. P. Optimization by Simulated Annealing. 220(4598) :671–680, 1983.

Lancichinetti, A. et Fortunato, S. Benchmarks for testing community detection algorithms on directed and weighted, 2009. ISSN 15393755.

Lancichinetti, A., Fortunato, S., et Radicchi, F. Benchmark graphs for testing community detection algorithms. *Physical Review E - Statistical, Nonli-*

near, and Soft Matter Physics, 78(4) :1–5, 2008. ISSN 15393755. doi : 10.1103/PhysRevE.78.046110.

Leonhard, E. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

Li, Z., Zhang, S., Wang, R.-S., Zhang, X.-S., et Chen, L. Quantitative function for community detection. *Physical Review E*, 77(3) :036109, March 2008. ISSN 1539-3755. doi : 10.1103/PhysRevE.77.036109.

Luce, R. D. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15(2) :169–190, 1950. ISSN 00333123. doi : 10.1007/BF02289199.

Luce, R. et Perry, A. A method of matrix analysis of group structure. *Psychometrika*, (1) :95–116, 1949. ISSN 0033-3123. doi : 10.1007/BF02289146.

Lusseau, D. et Newman, M. E. J. Identifying the role that animals play in their social networks. *Proceedings of the Royal Society B : Biological Sciences*, 271 (Suppl_6) :S477–S481, 2004. ISSN 0962-8452. doi : 10.1098/rsbl.2004.0225.

MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms David J.C. MacKay*, volume 100. 2005. ISBN 9780521642989. doi : 10.1198/jasa.2005.s54.

Meila, M. Comparing clusterings-an information based distance. *Journal of Multivariate Analysis*, 98(5) :873–895, 2007. ISSN 0047259X. doi : 10.1016/j.jmva.2006.11.013.

Mladenovic, N. et Hansen, P. Variable neighborhood search. *Computers and Operations Research*, 24(11) :1097–1100, 1997.

Mokken, R. J. Cliques, clubs and clans. *Quality & Quantity*, 13(2) :161–173, 1979. ISSN 00335177. doi : 10.1007/BF00139635.

Newman, M. E. J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23) :8577–82, 2006. ISSN 0027-8424. doi : 10.1073/pnas.0601602103.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., et Parisi, D. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9) :2658–2663, 2004. ISSN 0027-8424. doi : 10.1073/pnas.0400054101.

Ryan, D. M. et Foster, B. a. An integer programming approach to scheduling, 1981.

Scott, J. *Social Network Analysis : A Handbook*, volume 3. SAGE Publications Ltd, London, 2000. ISBN 0761963391. doi : 10.1370/afm.344.

Seidman, S. B. Network structure and minimum degree. *Social Networks*, 5 (3) :269–287, 1983. ISSN 03788733. doi : 10.1016/0378-8733(83)90028-X.

Seidman, S. B. et Foster, B. L. A graph-theoretic generalization of the clique concept. *The Journal of Mathematical Sociology*, 6(August 1970) :139–154, 1978. ISSN 0022-250X. doi : 10.1080/0022250X.1978.9989883.

Shi, J. et Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, 2000. ISSN 01628828. doi : 10.1109/34.868688.

Wang, J., Jia, Y., Hua, X.-s., Zhang, C., et Quan, L. Normalized Tree Partitioning for Image Segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference*, pages 1–8, 2008. doi : 10.1109/CVPR.2008.4587454.

Wasserman, S. et Faust, K. *Social network analysis : methods and applications*. Cambridge University Press, Cambridge, 1994.

Wu, Z. et Leahy, R. An optimal graph theoretic approach to data clustering : Theory and its application to image segmentation, 1993. ISSN 0162-8828.

Xu, G., Tsoka, S., et Papageorgiou, G. L. Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B*, 60(2) :231–239, 2007. ISSN 1434-6036. doi : 10.1140/epjb/e2007-00331-0.

Zachary, W. An information flow model for conflict and fission in small group. *Journal of Anthropological Research*, 33 :452–473, 1977.

Zhang, A. *Protein Interaction Networks*. Cambridge University Press, Cambridge, 2009. ISBN 9780521888950. doi : 10.1017/CBO9780511626593.