HEC Montréal

Détection de communautés robustes par regroupement de consensus

par
Ghislain Benoit

Sciences de la gestion
Option Intelligence d'affaires

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

Mars 2018

Table des matières

Table des figures

Résumé

Les méthodes computationnelles sont de plus en plus utilisées pour faire l'analyse de systèmes complexes afin d'y identifier des communautés. La plupart des méthodes utilisées présentement tel que les heuristiques de maximisation de modularité et les autres algorithmes non déterministes font face à un problème de limite de résolution. L'algorithme proposée tente de résoudre ce problème en réutilisant l'information de méthodes existante tel que LPA, LPAm et Louvain. Dans ce processus, les probabilités de coocurrence des sommets du graphes sont calculées à partir des partitions générées par un algorithme non déterministe. Ces probabilités sont ensuite utiliser pour créer un graphe de consensus sur lequel est appliqué une version agglomérative de l'algorithme de Kruskal pour obtenir un ensemble de partitions imbriquées. Les performances de la méthode proposée dans ce mémoire sont testées pour LPA, LPAm et une version plus gloutonne de Louvain à l'aide de graphes aléatoire ainsi qu'un cas classique de la théorie des graphes. La méthode proposée est capable d'améliorer les partitions obtenues par les algorithmes sources et fournie des deux niveaux hiérarchiques avec consistance. Comme le processus est parralèlisable, l'algorithme est en mesure de traiter des graphes contenant des millions de sommets dans un temps computationnel réaliste. Plusieurs modifications du processus ainsi que l'essai d'autres algorithmes sources reste ouvert à l'exploration.

Abstract

Computational methods are increasingly used to analyse complex systems in order to identify community structure. Most of the currently used methods like modularity optimizing heuristics and other non-deterministic algorithms face a resolution limit problem. The proposed algorithm tries to solve this problem by reusing information from currently existing methods including local-based LPA, LPAm and Louvain. The probabilities of node cooccurrence in the network are computed from partitions generated by a non-deterministic algorithm. These probabilities are used to create a consensus weighted graph on which is applied an agglomerative version of Kruskal's algorithm resulting in multiscale partitioning. Performance are tested for LPA, LPAm along with a greedier version of Louvain using both LFR hierarchical random graphs and a classical graph theory example. The method proposed is capable of improving output partitions in comparison to source algorithms on two-level embedded communities with consistency. Since the bulk of the process is parallelizable, partitions for graphs containing millions of nodes can be generated in reasonable computational time. Many possible modifications of the process as well as other source heuristics are left to be explored.

Chapitre 1

Introduction

1.1 Contexte

Le développement des technologies de l'information durant les dernières décénnies ont rendu disponible une très grande quantité de données en relation avec toutes les sphères de la société humaine. Bien que la théorie des graphes existe depuis des siècles, cette branche des mathématiques pouvant représenter une grande variété de systèmes complexes possède beaucoup d'applications dans les sciences humaines, naturelles et appliquées. Selon Scott (1991), la théorie des graphes a été utilisée pour faire l'analyse de réseaux sociaux à partir des années 30 par des chercheurs en psychologie tel que Kurt Lewin, Jacob Moreno et Fritz Heider. Les graphes font également apparition à la même époque dans des travaux en anthrolopologie et sociologie découlant de Radcliffe-Brown pour étudier les relations informationnelles et personnelles dans les systèmes sociaux. Mason et Verwoerd (2007) passent en revue les avancements de la théorie des graphes dans le domaine de la biologie et se penchent plus précisément sur l'utilité de la topologie de réseau sur l'étude des fonctions neurologiques ainsi que la structure des graphes pour étudier l'épidémiologie. Les graphes sont donc largement présents dans ce domaine tant au niveau macroscopique qu'au niveau microscopique. On les retrouve également en écologie et de plus en plus dans les réseaux de types biomoléculaires tel que les réseaux métaboliques, les réseaux d'interactions protéine-protéine et dans les réseaux de régulation génétique suite au développement du cribblage à haut débit.

Fortunato (2010) fait une revue extensive de la détection de communauté en réseau. Il explique que les graphes représentant des réseaux réels comportent à la fois de l'ordre et du désordre ; ils ne sont ni réguliers comme des treillis, ni homogènes comme les graphes aléatoires de Erdös et Renyi (1959). Pour mettre en contexte, ce dernier type de graphe fut créé en tant que modèle général afin de dériver un bon nombre de propriétés mathématiques. Les graphes aléatoires de Erdös et Renyi tendent à avoir une structure homogène puisque la probabilité qu'une arête soit placée entre deux sommets est équivalente pour chaque paire de sommets. Fortunato (2010) remarque que tout au contraire, les réseaux réels possèdent des groupes plus ou moins grands de sommets densément interconnectés tandis que les connections entre ces groupes de sommets sont plutôt faibles. Ils nomment cette propriété «structure de communauté» (*community structure*). Girvan et New-

man (2002) mentionnent aussi cette propriété qu'ils appellent également la transitivité de réseaux comme étant une propriété courante des réseaux réels. Cette propriété, mesurable par le coefficient de regroupement, indique la tendance pour trois sommets (A,B,C) de former une relation triangulaire avec une chaîne passant par $\overline{ABC}$ si A est connecté à B et B est connecté à C. «L'ami de mon ami est mon ami» tel que vulgarise l'article.

1.2 La définition d'une communauté

Ce travail se concentre donc sur la détection de communautés. Selon Lancichinetti, Fortunato et Radicchi (2008), une communauté, dans le cadre de la théorie des graphes, est un concept topologique pouvant représenter des regroupements ou modules de toute nature exprimés dans un graphe par des groupes de sommets. Parmis tout les types de réseaux étudiés, Fortunato (2010) fait la remarque que les communautés de sommets sont souvent caractérisées par des rôles ou fonctions communes. Il rajoute ici en plus des réseaux biologiques et sociaux mentionnés précédemment, les réseaux économiques, politiques, informatiques et technologiques. Bien qu'il existe plusieurs critères pour définir une communauté selon le contexte et la taille du réseau, les chercheurs dans le domaine s'entendent sur la notion qu'une communauté est un ensemble de sommets avec une densité interne élevée comparativement à l'extérieur tel que l'indique Newman (2006), Blondel et Al. (2008), Fortunato (2010) ainsi que Liu et Murata (2010) pour n'en citer que quelques uns. Cette définition est plutôt floue comme le remarque Raddicchi et Al. (2004). Ils notent en effet que cette définition qualitative porte «une difficulté intrinsèque dans l'interprétation des résultats sans informations non topologiques additionnelles» et rajoutent que les algorithmes existants ne font pas de discrimination objective à savoir quelle communauté en est vraiment une. Ils proposent donc une condition forte et faible pour définir une communauté ainsi qu'un algorithme prenant ces critères en considération. La condition forte (équation 1.1) étant quand, pour chaque sommet i du sous-graphe V représentant la communtauté et le degré étant défini la somme des poids des arêtes reliées à une composante, le degré interne k_i^{in} à la communauté V est supérieur au degré externe k_i^{in} de cette même communauté. Au sens faible, la somme des degrés internes pour tous les sommets doit être supérieure à la somme des degrés externes (équation 1.2).

$$k_i^{out} > k_i^{in}, \qquad \forall i \in V \tag{1.1}$$

$$\sum_{i \in V} k_i^{out} > \sum_{i \in V} k_i^{in}. \tag{1.2}$$

Plusieurs type de critères existent pour détecter une communauté mais ce mémoire se concentre sur les critères locaux (tel que les équations 1.1 et 1.2) et globaux (comme la modularité dont il sera question dans la prochaine section). Fortunato (2010) explique qu'un critère global tient compte du graphe dans son ensemble, dont chaque composante influe sur les autres malgré l'éloignement tandis qu'un critère local évalue uniquement un sous-graphe voir le voisinage direct d'un sommet.

Les critères locaux ont l'avantage de pouvoir éviter les problèmes de limites de résolution dont il sera question dans le prochain chapitre. En revanche, le calcul de critères globaux ont souvent une difficulté computationnelle plus faible et sont plus simples à appliquer.

Dans ce mémoire, le chapitre 2 présente une revue de certains critères de détection de communautés et les algorithmes utilisées pour en faire le calcul. Quelques applications de la détection de communautés seront présentés à titre d'exemple. Il sera question de limite de résolution et des méthodes utilisées pour remédier à ce problème. Les méthodes de regroupement par consensus sont ensuite discutés et enfin la problématique de ce mémoire sera définie. Le chapitre 3 présente l'idée générale de la nouvelle méthode. Le chapitre 4 est dédié à une article sur la méthode présentée et inclus les résultats aux tests effectués. La conclusion de ce mémoire est présentée au chapitre 5.

Chapitre 2

Revue de la littérature

2.1 La détection de communautés en théorie des graphes

La psychologie et la sociologie mathématique ont également développé des critères locaux pour identifier des communautés. Luce et Perry (1949) introduisent les concepts de cliques et de n-cliques, le premier se définit comme un groupe de sommets étant tous reliés les uns aux autres ou autrement dit, un sous-graphe complet maximal. L'article propose des méthodes matricielles afin d'identifier les cliques et en dérive des propriétés mathématiques. Mais la clique n'est efficace que pour un type précis de communauté dont les membres sont tous connus les uns des autres. Luce (1950) définit alors la clique généralisée ou n-clique comme étant un groupe de sommets reliés les uns aux autres par une distance de maximum n arêtes. Tout comme dans l'article précédent, l'article propose une méthode matricielle pour procéder à l'identification et dérive des propriétés pour les n-cliques. Les n-cliques sont certes plus flexibles que leur antécédant mais possèdent l'inconvénient d'avoir un diamètre potentiellement plus grand que n. De plus, une n-clique peut s'avérer être non connexe. Alba (1973) stipule que les deux méthodes matricielles proposées précedemment ne sont pas assez efficaces computationnellement pour identifier les communautés sur un grand ensemble de données. Son approche fait donc utilisation de la théorie des graphes pour proposer de nouvelles techniques d'identification. D'autres critères ont donc été développés par la suite dans la même branche afin d'élargir les représentations conceptuelles de communautés. Mokken (1979) introduit les n-clans et les n-clubs, tous deux des variantes des n-cliques. Les n-clans sont des sous-graphes dont les sommets sont reliés par une distance maximum de n [Mokken (1979)]. Contrairement aux n-cliques, les n-clans ne tiennent pas compte des sommets connectés via un autre sommet extérieur au clan (avec une distance inférieure à n). Les n-clubs quant à eux sont des sous-graphes maximaux avec un diamètre maximum de n. Les critères ci-présent dont la pertinence semble très intuitive constituent les premières tentatives de définir les communautés en théorie des graphes. Leur utilité en pratique est en revanche questionnable pour un graphe de grande taille.

Selon Fortunato (2010), la nécessité pour des algorithmes à plus faible difficulté computationnelle est apparue avec la croissance grandissante de données disponibles et leur capacité de

traitement. Un certain nombre de méthodes ont dont été développées au début des années 2000 apportant une nouvelle perspective dans le domaine de la détection de communautés. Le partitionnement de graphe existe déjà en informatique depuis plusieurs décennies. La méthode fut destinée à l'origine non pas à la détection de communautés mais plutôt pour des tâches techniques comme la division de tâches de travail pour des processeurs. Un de ses grands défauts tel qu'indiqué par Newman et Girvan (2003) est l'obtention de partition divisant le graphe en communautés ayant toujours le même nombre de sommets, un trait qui ne correspond pas à la réalité des modules en réseaux. Le partitionnement de graphe constitue néanmoins un point de départ pour une nouvelle branche de recherche.

L'article de Girvan et Newman (2002) et les travaux qui s'en sont suivis marquent un tournant important dans la détection de communautés en terme d'innovation et d'influence. Les méthodes hiérarchiques traditionnelles sont décrites dans ce premier article comme un graphe dont l'ensemble des arêtes est vide et dont on connecte les sommets au fur et à mesure selon une matrice de poids W_{ij} pour chaque paire de sommets i, j. La façon de calculer les poids peut varier. Selon l'article, cette méthode n'arrive pas à déceler des partitions de qualité sur certains types de graphes connus pour en posséder. De plus, les sommets reliés à un module par un seul lien sont souvent séparés en une communauté à part. Les méthodes hiérarchiques traditionnelles comportent aussi l'inconvénient d'avoir à choisir un point de coupure dans le processus de fusion des communautés, donc devoir choisir le nombre de communautés désiré. L'algorithme proposé par Girvan et Newman (2002) fait l'inverse de ces méthodes, il divise les arêtes à chaque itération en partant du graphe d'origine en se basant sur l'intermédiarité des arêtes (edge betweenness). La mesure utilisée dans cet article provient des travaux de Freeman (1977) sur la centralité noeuds, adapté pour les liens par Newman (2001) et parallèlement par Brandes(2001). Cette méthode, quoique trop difficile computationnellement pour être utilisée sur les graphes de grande taille, arrivent à obtenir d'excellents résultats sur les exemples classiques du club de karate de Zachary et sur un réseau de Football collégial américain ainsi que sur des graphes générés aléatoirement par ordinateur.

Plus tard, Girvan et Newman (2003) offrent une description et une série de tests plus approndie sur les méthodes divisives basées sur l'intermédiarité des liens. L'article propose trois méthodes différentes pour calculer l'intermédiarité, soit par algorithme de plus court chemin, par réseaux de résistances et par marches aléatoires. La méthode par réseaux de résistance, inspirée de la physique électrique, est équivalente à étudier le courant dans un circuit de résistance auquel on branche un courant pour chaque chaque paire de points i, j, (Klein et Randic (1993)). L'intérêt de cet article réside surtout dans l'utilisation de la modularité afin de mesurer la qualité des partitions obtenues. Cette fonction de qualité est aujourd'hui très largement utilisée et elle serait la plus populaire selon Fortunato (2010). La modularité, représentée par l'équation 2.1, indique la densité des liens internes aux communautés ou partitions d'un graphe comparativement au modèle nul, c'est-à-dire la densité espérée si il y avait réaffectation aléatoire des arêtes du graphe. L'équation 2.1 s'applique pour toutes les paires de sommets i et j, m est le poids total des liens du graphe. A_{ij} est le poids de l'arête entre les sommets i et j, k_i et k_j représentent les degrées des sommets i et j. $\delta(c_i, c_j)$ est appelé le delta de Kronecker, il équivaut 1 si i et j font partie de la même communauté sinon 0 .

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{2.1}$$

Newman (2004) revient sur la modularité et propose de l'optimiser directement. Mais puisque l'obtention d'un optimum global est très difficile computationnellement, des méthodes heuristiques pour les graphes doivent être utilisées. Cet article décrit ainsi un algorithme agglomératif afin d'optimiser la modularité sans toutefois avoir une garantie d'optimalité. La méthode consiste en une liste de sommets ayant chacun leur propre communauté et procède à fusionner chaque sommet un par un avec le sommet apportant le plus grand gain jusqu'à ce qu'aucune amélioration ne soit possible. L'algorithme fut testé sur des graphes aléatoires en plus de graphes réels classiques à la littérature en comparaison avec l'algorithme présenté dans Girvan et Newman (2003). Sur les graphes aléatoires, les performances dépendent du ratio de liens intercommunautaires du graphe. En terme du pourcentage de sommets classifiés correctement, Newman (2004) observe que l'algorithme de maximisation de modularité présenté performe mieux quand ce dernier ratio est élevé tandis que l'algorithme de Girvan et Newman (2003) possède un léger avantage quand ce ratio est faible. Ce fait fut également observé dans les réseaux réels.

Dans un autre article par Clauset, Newman et Moore (2004), l'algorithme de maximisation de modularité présenté apporte une réduction de la difficulté computationnelle. Cet algorithme opère selon les mêmes principes que Newman (2004) sauf qu'il utilise trois différentes structures de données rendant le processus plus efficace et permettant ainsi de traiter un graphe de plus de 400 000 sommets dans un temps raisonnable.

Duch et Arenas (2005) développent une algorithme de maximisation de la modularité par optimisation extrémale. Les tests effectués sur des graphes aléatoires montrent que l'algorithme arrivent avec une modularité plus évelée que les algorithmes précédant sur les graphes classiques et une meilleure fraction de sommets correctement classifiés sur des graphes aléatoires.

Newman (2006) développe une méthode différente pour optimiser la modularité en faisant appel à une structure matricielle permettant l'application d'un algorithme spectral. Des tests effectués sur des réseaux réels montrent que cette méthode parvient à obtenir des partitions avec une modularité plus élevée que les algorithmes développés précédemment sur les réseaux de plus grande taille et une modularité presqu'aussi élevé que Duch et Arenas (2005) sur les réseaux de petite taille avec une difficulté computationnelle plus faible. L'algorithme est par contre plus difficile computationnellement que celui de Clauset, Newman et Moore (2004).

Celui de Blondel et al. (2008) est l'exemple le plus notoire en ce qui est du partitionnement par maximisation de la modularité. L'article propose une heuristique ayant la capacité de traiter des graphes ayant des centaines de millions de sommets en un temps raisonnable. La méthode fonctionne en deux phases. Les sommets sont avant tout triés dans un ordre aléatoire. La première phase consiste ensuite à explorer le voisinage de chaque sommet et d'assigner à chaque itération le voisin

apportant le plus grand gain en termes de modularité dans la communauté du sommet exploré. La deuxième phase débute quand aucune amélioration n'est possible lors de la première phase et procède à la construction d'un nouveau graphe dans lequel les sommets de chaque communauté sont fusionnés ensemble dans un hypergraphe avec une boucle pondérée représentant les liens intracommunautaires pour chaque communauté et un lien pondéré pour tous les liens intercommunautaires entre deux communautés. L'algorithme revient alors à la première phase et le processus dans son ensemble s'arrête encore une fois quand aucune amélioration n'est possible, donc nul besoin de savoir au préalable le nombre de communautés désirées. La nature agglomérative de la phase 2 fait diminuer la taille du graphe et rend la tâche beaucoup plus simple computationnellement pour chaque répétition subséquente. L'algorithme a été testé sur plusieurs réseaux dont Zachary mais aussi d'autres réseaux de taille très grande et fait preuve d'une rapidité et d'une efficacité sans précédent au moment où l'article a été publié en ce qui attrait à l'optimisation de la modularité. Tout comme les autres méthodes de cette catégorie, l'optimum global n'est pas garanti. Il s'agit tout de même d'un des algorithmes les plus utilisés dans le domaine de la détection de communauté aujourd'hui.

La modularité comporte toutefois des faiblesses. Barthélemy et Fortunato (2006) démontrent que la modularité optimale possède une limite de résolution, i.e. puisque l'indice dépend du nombre global de liens, les plus petites communautés peuvent se perdre dans un effet d'échelle. La modularité peut donc parfois ignorer les plus petits modules et ainsi offrir en revanche de plus grandes communautés. Les auteurs proposent plusieurs solutions telles que modifier l'équation de la modularité pour en faire une moyenne plutôt qu'une somme ou encore faire utilisation de critères locaux au lieu d'un critère global. Il faut aussi mentionner que les algorithmes dit «gourmands» couramment utilisés pour optimiser la modularité offre des résultats inconsistents car ils dépendent de l'ordre dans lequel la liste des sommets est traitée. De plus, puisque la taille du réseau influence les valeurs de la modularité, celle-ci ne peut être utilisée que pour comparer différentes partitions d'un même réseau.

Raghavan, Albert et al. (2007) présentent un algorithme de propagation d'étiquette (LPA). Tout comme les algorithmes de maximisation de modularité, il n'est pas requis de savoir la taille et le nombre de communautés désirées au préalable. Chaque sommet possède initialement une étiquette correspondant à son identifiant. Une altération locale de l'étiquette de chaque sommet est ensuite effectuée selon l'étiquette la plus commune chez les sommets voisins. L'article démontre que l'algorithme opère avec une difficulté computationnelle presque linéaire. L'algorithme arrive à converger rapidement sur des solutions à hautes valeurs en modularité. La validité des résultats est également vérifiée par comparaison avec des graphes classiques bien connus, soit Zachary et Football mentionnés précédemment. En revanche, l'algorithme de propagation d'étiquette donne des résultats extrêmement instables, et tout comme indique Barber et Clark (2008), l'optimum global du problème est une solution triviale où tous les sommets font partie de la même communauté et la fonction à optimiser dans LPA ne correspond pas nécessairement à une propriété désirée pour le réseau, ce qui remet en question d'autres solutions au voisinage de l'optimum global. Barber et Clark proposent donc plusieurs variantes du LPA, soit le LPAr, LPAb, LPAm et une version

hybride avec LPA suivi de LPAm. Dans le processus du LPA, quand la plus grande fréquence des étiquettes voisines est équivalente pour un sommet, l'algorithme choisit le statu quo. Si le statu quo ne fait pas partie des options, une étiquette est choisie aléatoirement parmi les plus fréquentes. Le LPAr fonctionne de façon très similaire au LPA, sauf que dans cette dernière situation, l'algorithme choisi toujours une étiquette au hasard au lieu du statu quo. Quant à LPAm, il possède un critère de sélection d'étiquette avec un terme maximisant la modularité locale. Finalement, le LPAb modifie le critère de sélection afin de maximiser la modularité bipartite dans le voisinage du sommet. Des tests effectués sur des graphes générés aléatoirement montre clairement que LPAm arrive avec des partitions offrant une modularité supérieure à LPA, LPAr, LPAb et l'algorithme hybride. Il en va de même pour l'information mutuelle normalisée entre les solutions connues et les solutions obtenues par l'algorithme. Les tests effectués sur les graphes réels classiques sont par contre inconclusifs, chaque algorithme semble performer plus ou moins bien selon le graphe utilisé en terme de modularité.

Hollocou et al. (2017) développent un algorithme utilisant une métrique globale similaire à la conductance. Le fonctionnement est très simple et d'une complexité computationnelle linéaire, donc très rapide. Dans un graphe vide où chaque sommet est une communauté, l'algorithme parcourt une liste d'arêtes triées aléatoirement, modifie le degré des sommets aux deux extrémités du lien, et joint les communautés ensemble si le degré total des deux sommets ne dépasse pas le seuil maximum établi. Les tests effectués sur des réseaux réels montrent une meilleure précision que Louvain et les autres algorithmes couramment utilisés pour les très grands réseaux. Pour les réseaux de taille plus modeste par contre, la précision de leur algorithme est nettement inférieure.

Une autre méthode souvent mentionnée est l'algorithme Infomap par Rosvall et Bergstrom (2011). Cet algorithme fait appel à une équation carte généralisée (*generalized map equation*). Emprunté de la théorie de l'information, ce concept est utilisé afin d'obtenir une hiérarchie de partitions à niveau variable. Les paramètres de cette équation récursive sont obtenus par marches aléatoires.Plusieurs méthodes heuristiques peuvent être utilisées pour résoudre les équations afin d'obtenir une valeur minimale. Dans le cas présenté, l'algorithme est similaire à la méthode Louvain, c'est-à-dire qu'en première phase une liste de sommets triée aléatoirement est parcourue et chaque opération vise à trouver le voisin avec lequel le sommet sélectionné constitue une communauté minimisant la valeur de l'équation de carte jusqu'à ce qu'un minimum soit atteint. L'algorithme procède alors de façon récursive à une nouvelle phase essentiellement identique mais à un niveau supérieur, soit avec les communautés et sous-communautés. Pour chaque phase, l'ordre aléatoire des sommets ou modules est réeffectué après chaque parcours. Cet algorithme possède l'avantage d'arriver à des partitions hiérarchisées sans avoir à choisir les nombres de communautés ou sous-communautés ni le nombre de niveau de hiérarchie. Les résultats obtenus sont par contre inconsistents tout comme les autres heuristiques d'optimisation de fonction de qualité.

2.2 Applications de la détection de communautés

L'application la plus évidente de la détection de communauté est probablement l'analyse des réseaux sociaux. On peut y voir par exemple un potentiel de ciblage de clients ou bien encore un outil pour comprendre la dynamique entre différents groupes. Les applications de la détection de communautés sont cependant beaucoup plus diverses. Trois articles sont passés en revue dans cette section pour en donner des exemples.

Isogai (2014) utilise un algorithme de maximisation de la modularité afin de faire du regroupement (*clustering*) sur des graphes d'actions japonaises avec des arêtes pondérés par corrélation calculée à partir du modèle ARMA-GARCH. Il utilise un algorithme spectral développé par Newman (2006) pour maximiser la modularité dans le but d'obtenir des résultats de hautes valeurs. Il tente de remédier au problème de limite de résolution en utilisant l'algorithme d'optimisation de façon récursive sur les sous-graphes correspondant aux partitions identifiées suite à la première étape du processus. Selon l'auteur, cette classification constituerait un outil pour assister à la gestion du risque remplaçant le coefficient de Pearson. L'utilisation de ce dernier assumerait à tort la distribution normale du rendement des actifs. Il nécessite souvent en plus une analyse factorielle préalable pour réduire la dimensionalité de la matrice dont la complexité croît de façon polynomiale. La méthode par détection de communautés permet d'éviter ces deux derniers problèmes.

La détection de communauté est aussi utilisée en neurologie. Par exemple, Zhang et al. (2008) construisent un graphe à partir de données sur l'épaisseur corticale du cerveau humain récoltées par imagerie à résonnance magnétique et y appliquent l'algorithme de maximisation modularité de Clauset, Newman et Moore (2004) adapté pour les communautés avec tailles hétérogènes par Diaz-Guilera et Arenas (2006). Selon leurs analyses, les communautés trouvées ont une correspondance très proche aux différentes régions correspondant aux différentes fonctions du cerveau. Les auteurs arrivent donc avec une représentation du cerveau par six modules densément interconnectés. Selon eux, les partitions pourraient aider à la compréhension du développement cérébral ainsi que de certains troubles neuro-psychiatriques.

En épistémologie, Claveau et Gingras (2016) appliquent la méthode Louvain dans un processus dynamique sur des graphes de citations construits à partir des données tirées du service d'indexation *Web of Sciences*. Les graphes couvrent les documents publiés en sciences économiques de 1956 à 2014. Le poids entre chaque arête est déterminé par une mesure de couplage bibliométrique exprimée par l'équation (2.2), où n_{ab} indique le nombre de documents communs cités par les deux articles et n_a, n_b, le nombre de documents total cité par a ou b. Pour assurer une stabilité intertemporelle, les auteurs utilisent la méthode Louvain pour maximiser la première période et ensuite sauvegardent le partitionnement obtenu suite au premier cycle de l'algorithme selon lequel les communautés des sommets existant toujours à la période suivante sont initialisées. Pour juger si un regroupement modifié par l'ajout ou le retrait de sommets à la période suivante constitue encore le même regroupement, les auteurs utilisent un critère mesurant le ratio du nombre de documents existants dans une même communauté durant les deux périodes par rapport au nombre de

documents existants dans cette communauuté durant la période initiale. Ils arrivent donc avec des groupes de clusters qui se créent, évoluent, et disparaissent au travers le temps. L'application d'un algorithme d'exploitation textuel permet de vérifier la cohérence des regroupements et leur évolution. Selon les auteurs, les articles correspondent aux différentes écoles de pensées ou spécialités. Ces quelques exemples concrets montrent l'utilité que peut avoir la détection de communauté mais ses applications sont beaucoup plus étendues.

$$\frac{n_{ab}}{\sqrt{n_a n_b}} \tag{2.2}$$

2.3 Approches pour faire face au problème de la limite de résolution

Tel que mentionné précédemment, la maximisation de la modularité possède une limite de résolution. Les tailles de communautés dans les réseaux réels sont très variables, et suivent parfois une loi de puissance. Vu l'hétérogénéité de la densité et des tailles des communautés, Isogai (2014) remarque que "le modèle nul ne peut pas ajuster la résolution adéquatemment pour détecter à la fois le grand et les petits regroupements". Différentes méthodes ont été proposées pour faire face au problème. Une première approche serait de faire une résolution multiple de la modularité, comme par exemple utiliser les premières partitions comme sous-graphes et réoptimiser. Lancichinetti et Fortunato (2011) indiquent par contre que la limite persiste après résolution multiple et est suceptible de persister avec toute méthode ayant un critère d'optimisation global étant donné un graphe d'assez grande taille, ce qui inclut OSLOM et Infomap. Il faudrait donc se tourner vers les méthodes locales afin d'arriver avec une définition plus précise des communautés dans les réseaux hétérogènes.

Les travaux de Yin et al. (2017) sont un exemple récent d'un algorithme qui tend à être plus local en évaluant la conductance des motifs. À partir d'un sommet d'origine, l'algorithme peut trouver un regroupement (appelé *motifs* par les auteurs) minimisant la conductance. La méthode est une adaptation de l'algorithme *APPR* (Andersen, Chung et Lang (2006)) pour traiter la conductance de motifs. Elle possède la capacité de traiter des graphes dirigés contrairement aux autres algorithmes utilisant ce type de métrique. Si cet algorithme arrive à éviter le problème de résolution, il n'arrive par contre pas à construire des partitions et exploiter la nature hiérarchique d'un graphe.

Chakraborty et al. (2014) utilisent une métrique locale basée sur la connectivité des sommets qu'ils nomment la permanence. Cette métrique est l'aggrégat de deux mesures. La première tient compte du nombre de connections internes par rapport au nombre maximum de connections vers une autre communauté, le terme est ensuite normalisé entre 0 et 1 par le degré total de la communauté. De ce premier terme est soustrait 1 moins le coefficient de regroupement interne qui

évalue la densité des connections internes pour arriver à la permanence, dont les valeurs peuvent varier entre -1 et 1 selon les auteurs. Quoique la permanence est une mesure locale, l'algorithme proposé vise quand même à maximiser la permanence globale du graphe, donc le problème initial d'éviter les limites de résolutions n'est pas nécessairement réglé. Des tests de cet algorithme sur des graphes aléatoires LFR ainsi que sur quelques graphes classiques démontrent une précision comparable aux algorithmes préexistants sur les réseaux ayant des communautés bien définies. L'algorithme offre une précision relativement plus élevée sur les graphes avec des réseaux difficiles à identifier, tel que sur les graphes aléatoires LFR ayant une valeur de μ élevée.

Bien que ces algorithmes ont leurs utilités, les deux prochains articles révèlent davantage la structure hiérarchisée des communautés. Or, les systèmes réels regorgent de grandes communautés divisibles en plus petites communautés tel qu'observé dans les réseaux métaboliques par Ravasz et al. (2002). Gomez et al. (2015) développent un algorithme à deux étapes combinant à la fois les méthodes divisives et agglomératives existantes en se basant respectivement sur une mesure locale d'«intermediarité» et une mesure de similarité. Ces mesures sont calculées au début de chaque itération, suivi par la phase divisive puis agglomérative aboutissant sur un graphe ayant la forme d'une forêt ou autrement dit, un groupe d'arbres disjoints. À la fin de l'itération, chaque arbre représente une partition. L'algorithme au final offre une série de partitions chacune imbriquées dans la partition précédente tel que dans les algorithmes classiques de regroupement hiérarchiques, offrant ainsi beaucoup plus d'information qu'une simple partition obtenue par l'optimisation d'une fonction objectif mais ayant par contre une plus grande difficulté computationnelle et créant aussi le nouveau problème du choix des partitions à utiliser. Des tests sur des graphes réels et simulés montrent une précision comparable aux algorithmes de détection de communautés couramment utilisées.

Bodlaj et Batagelj (2015) développent également une mesure de dissimilarité locale pour algorithme agglomératif classique. Cet algorithme peut fonctionner sur les graphes normaux ainsi que sur les «*line graphs*» et peut donc fusionner les sommets ou les arêtes ainsi que fournir des partitions imbriquées ou avec chevauchements de communautés. Les tests effectués par les auteurs rendent l'algorithme difficilement comparable à d'autres méthodes mais démontrent tout de même que l'algorithme arrive avec une validation de communautés relativement bonne sur des réseaux connus.

2.4 Les méthodes d'ensembles

En plus des méthodes explorées jusqu'à présent, il est possible d'appliquer d'autres méthodes en aval afin d'améliorer les solutions existantes. Ghosh et Strehl (2002) proposent l'utilisation de méthodes d'ensembles en analyse de regroupement afin d'améliorer la qualité et la robustesse des résultats. Ils offrent une définition théorique de ce qu'ils appellent le problème d'ensemble de regroupements (*Cluster Ensemble Problem*) qui consiste à utiliser l'information obtenu par plusieurs partitions afin d'arriver à une solution finale. Topchy et al. (2005) démontrent par des tests

empiriques que les méthodes d'ensembles offrent des résultats valides et rendent les algorithmes de regroupement beaucoup moins sensibles au bruit et aux valeurs extrêmes. Certaines des méthodes d'ensembles qu'ils proposent arrivent d'ailleurs à améliorer la précision des solutions sur des graphes amputés. Ces méthodes ouvrent également la porte à la parallélisation et à l'intégration de différents algorithmes de clustering.

Les méthodes de regroupement par consensus, un autre nom attribué aux méthodes d'ensembles, existe aussi en théorie des graphes dans le contexte de la détection de communautés. Deux implémentations par Kwak, Moon et al. (2011) et Lancichinetti et Fortunato (2012) se servent du regroupement par consensus pour calculer la probabilité de coocurrence des sommets. Cette dernière décrit le nombre de fois où deux sommets sont apparus dans la même communauté divisé par le nombre total de partitions. Dans le cas de ces deux articles, cette probabilité est réincorporée dans le graphe pour remplacer les poids des arêtes et le processus est ainsi répété jusqu'à convergence (c'est-à-dire jusqu'à ce que tous les poids équivalent 0 ou 1). Compte tenu qu'un graphe ne possède pas une grande densité, la réutilisation de la probabilité de coocurrence des sommets en tant que poids pour les arêtes permet de réduire grandement la difficulté computationnelle en comparaison avec une matrice de coocurrence. Comme une matrice de coocurrence possède n^2 champs pour n sommets, le nombre de case mémoire à traiter augmente quadratiquement à mesure que le nombre de sommets dans le graphe augmente. Par ailleurs, les deux algorithmes sont sensiblement les mêmes avec l'exception d'une étape de filtration ajoutée par Lancichinetti et Fortunato (2012). Ces derniers rajoutent une section sur la détection de communautés dynamiques et explorent la possibilité d'utiliser le regroupement par consensus à cette fin. Des tests effectués par Kawk, Moon et al. (2011) démontrent la consistence de l'algorithme. De l'autre coté, Lancichinetti et Fortunato (2012) procèdent à des tests de précision en utilisant à la base plusieurs algorithmes de maximisation de modularité mais aussi d'autres algorithmes non déterministes tel que la propagation d'étiquette, infomap et OSLOM. Ces tests effectués sur des graphes aléatoires à degrés et tailles de communautés variables *LFR Benchmark* conçus par Lancichinetti, Fortunato et Raddicchi (2008) montrent une amélioration de la précision des algorithmes avec l'utilisation du consensus comparativement à l'algorithme source.

D'autres auteurs font également utilisation de cette méthode pour diverses fins. Pereira, Carvalho et Nascimento (2016) remarquent une lacune de méthodes s'adressant au problème de limite de résolution pour les graphes dirigés et développent par conséquent un algorithme de regroupement par consensus. Leur méthode arrive à retrouver les communautés de graphes aléatoires LFR de façon autant sinon plus précise que OSLOM et INFOMAP. Zhang et Moore (2014) génèrent des partitions à hautes valeurs en modularité à l'aide d'un algorithme de propagation de message (*belief propagation algorithm*) avec lesquels ils appliquent un algorithme de regroupement par consensus pour obtenir des partitions hiérarchisées. Burgess et Al. (2016) se servent du consensus pour imputer des arêtes manquants dans les graphes incomplets. Liang Zong-Wen et al. (2014) créent un graphe avec poids consensus (tel que Kwak, Moon et al. (2011) et Lancichinetti et Fortunato (2012)) à partir de partitions générées par LPA afin de traiter avec le problème d'instabilité de l'algorithme. Ils font ensuite l'adaptation du critère d'optimisation du LPA pour y inclure les poids

consensus, créant ainsi le LPAcw. Des tests démontrent que l'algorithme arrive à retrouver des partitions à haute modularité sur les réseaux classiques et supplante LPA en termes de précision sur les graphes LFR Benchmark.

Chapitre 3

Méthodologie

La littérature sur la détection de communautés en théorie des graphes a connu une explosion durant les dernières années. Les algorithmes actuels ont la capacité de traiter des réseaux contenant des dizaines voir centaines de millions d'arêtes tout en arrivant à des solutions d'une qualité assez bonne pour en faire utilisation. Il existe toutefois encore certaines lacunes quant à la stabilité et la précision des algorithmes utilisées. C'est le cas par exemple de la méthode Louvain, soit la méthode la plus connue, qui comporte un problème de limite de résolution en plus d'arriver à des résultats différents à chaque exécution. Cet article explore les façons efficaces de traiter ces problèmes. Pour ce faire, un algorithme de détection de communauté non déterministe est utilisé pour générer un grand nombre de partitions qui déterminent la coocurence des arêtes du graphe. Cette nouvelle mesure permet un regroupement hiérarchique par agglomération. L'algorithme proposé est simple, parallélisable, obtient des résultats riches et peut s'adapter à un grand nombre d'autres algorithmes. Le regroupement par consensus arrive à améliorer les résultats des algorithmes sources testés sur des graphes LFR Benchmark hiérarchiques à deux niveaux.

3.1 Algorithmes Sources

Cette section montre les procédures par lesquelles l'algorithme est implanté. Le travail a été effectué en utilisant le language C++. Pour débuter, la modularité d'un graphe, souvent exprimé par l'équation 2.1 du chapitre précédent, peut également être exprimée par l'équation 3.1. Pour des raisons pratiques, l'équation 3.1 est utilisée pour ce travail puisque la sommation s'effectue sur les communautés et permet un recalcul rapide de la modularité suite à des changements locaux. Ici, i représente une communauté, in_i et out_i représentent respectivement le nombre d'arêtes ayant les deux extremités ou une seule dans i, m est la somme des poids des liens du graphe.

$$Q = -\sum_i \left[\left(\frac{2in_i + out_i}{2m} \right)^2 - \frac{in_i}{m} \right] \tag{3.1}$$

La première étape de l'algorithme est la génération d'un certain nombre de solutions par lesquelles seront calculés la probabilité de coocurrence pour chaque paire de sommets connectés. Ce travail évalue les performances de ce processus avec trois méthodes différentes, soit une version modifié de Louvain dans laquelle les sommets sont fusionnées directement après chaque altération de communauté ainsi que deux algorithmes de propagation d'étiquette. Chaque méthode est testée avec le cas où la partition générée avec une modularité maximale est utilisée. Louvain est une heuristique très rapide qui recherche une partition maximisant la modularité. Elle ne garantit par contre pas une solution optimale, ce qui n'est pas l'enjeu de ce travail qui cherche à obtenir des résultats stables et détaillés en utilisant entre autre la modularité. La méthode Louvain, et encore plus dans cette version, est très gourmande et dépend de l'ordre selon lequel les sommets sont arrangés initialement. Cette propriété est intéressante puisqu'elle permet une exécution massive qui permet de générer un grand nombre de partitions. Malgré les faiblesses de la modularité, cette méthode est pertinente dans ce travail puisqu'elle arrive à fournir une distribution aléatoire de solutions de bonne qualité avec une difficulté computationnelle relativement basse. La version de Louvain utilisée dans cette article est décrite ci-dessous. Les modifications apportés amène une plus grande variabilité, une plus grande vitesse d'exécution et une légère baisse de la modularité en moyenne.

Algorithme : Louvain Modifié

Données : $G = (V, E)$
Soit une partition initiale où chaque sommet v constitue sa propre communauté ;
Soit Q, la modularité courante du graphe ;
Soit q, la modularité du graphe après fusion de n et v ;
Créer une liste triée aléatoirement L de tous les sommets voisins de v ;
pour tous *v de L* **faire**
    Créer une liste l_n de tous les voisins de v ;
    pour tous *Élements n de* l_n **faire**
        $n' = \emptyset$;
        $\tilde{q} = -\infty$;
        si $q > \tilde{q}$ **alors**
            $n' \leftarrow n$;
            $\tilde{q} \leftarrow q$;
    si $\tilde{q} > Q$ **alors**
        Fusionner v et n' ;
        Ajouter le sommet créé par la fusion à L_v ;
        Retirer v et n' de L_v ;
        $Q \leftarrow \tilde{q}$;

La méthode suivante utilisée pour générer des partitions est l'algorithme de propagation d'étiquettes (LPA) développé par Raghavan et al. (2007). Ce type d'algorithme est également de type non deterministe, ce qui est nécéssaire afin de pouvoir être utilisé en regroupement par consensus. Ces processus sont beaucoup plus instables que la méthode Louvain, particulièrement LPA. Néanmoins, cette variance constitue ici une propriété intéressante dans le contexte où le résultat final fait utilisation de la coocurrence des sommets. Ces algorithmes sont également beaucoup plus rapides mais nettement moins bons quand vient à maximiser la modularité, ce qui n'est pas un problème pour les mêmes raisons mentionnées précédemment.

Dans le modèle de la propagation d'étiquettes, une étiquette correspond à une communauté. À l'état initial de l'algorithme, chaque sommet possède sa propre étiquette. Ensuite à chaque itération, le sommet sélectionné se voit assigné l'étiquette la plus fréquente dans son voisinage. L'algorithme s'arrête quand aucun changement d'étiquette n'est possible. Selon Raghavan et al. (2007), LPA possède une difficulté computationnelle presque linéaire, ce qui en fait donc un des algorithmes de détection de communauté les plus rapides. L'équation 3.2 définit le critère de sélection d'étiquette après chaque itération pour LPA. L_x^{new} représente la nouvelle étiquette assignée au sommet x, A_{ix}, le poids de l'arête entre les sommets i et x. $\delta(l_i, l)$ représente le Delta de Kronecker qui prend la valeur de 1 si la nouvelle étiquette est la même que pour le sommet i et prend la valeur de 0 autrement,

$$L_x^{new} = \arg\max_b \left(\sum_{i=1}^{n} A_{ix} \delta(l_i, l) \right) \tag{3.2}$$

Le LPAm est une variante du LPA par Barber et Clark (2009) qui prévient les solutions triviales tel qu'avoir le graphe en entier étant une seule partition. Cette modification du critère entraîne l'algorithme à maximiser la modularité localement. Comme l'indique Liu et Murata (2010), ce critère de sélection entraîne une perte de variabilité puisqu'elle peut restreindre la déroulement du processus dans un parcours de solutions locales maximisant la modularité. Le critère de sélection pour LPAm est exprimé par l'équation 3.3 où L_x^{new} représente la nouvelle étiquette assignée au sommet x, $B_{ix} = A_{ix} - \frac{k_i k_x}{2m}$ (voir équation 2.1) et $\delta(l_i, l)$ représente le delta de Kronecker (également expliqué au chapitre 2).

$$L_x^{new} = \arg\max_b \left(\sum_{i=1}^{n} B_{ix} \delta(l_i, l) \right), \tag{3.3}$$

3.2 Regroupement par consensus

Suivant le déroulement d'une des trois méthodes qui viennent d'être décrites peut être calculé la proportion de chaque paire de sommets connectés faisant partie de la même partition. Ce pourcentage, connu sous le nom de probabilité de coocurrence, peut être interprété comme étant la probabilité que les deux sommets soit dans la même commmunauté suite à l'application d'un algorithme source. Puisque l'intervalle de cette valeur se situe en $[0,1]$, la soustraction de 1 à cette proportion définit une métrique de dissimilarité qui peut être exprimée dans une matrice de dissimilarité de taille $n \times n$. L'algorithme décrit dans ce mémoire utilise cette dissimilarité en tant que poids dans un nouveau graphe, éliminant ainsi l'information quant à la coocurrence de sommets n'étant pas directement connecté par une arête. Cette façon de procéder nous permet par contre d'opérer sur une liste de liens et rend le processus beaucoup plus efficace sur les graphes creux. La reconstruction du graphe est suivie par une méthode de regroupement classique comparable à un algorithme naïf opérant dans un espace graphe (note : l'algorithme est presque qu'identique à Kruskal (1956) pour trouver des arbres recouvrant à poids minimal avec l'exception que chaque paire de sommets est fusionnée au lieu d'être placée dans la même communauté). L'algorithme est décrit ci-dessous. Vu la nature agglomérative de cet algorithme, il faut définir la dissimilarité des arêtes dans l'éventualité où plusieurs sommets d'un regroupement ont des liens vers un autre regroupement puisque la dissimilarité ne possède pas les mêmes propriétés qu'un poids d'arête conventionnel qui peuvent être simplement additionnés ensemble. Dans le cas de la probabilité de coocurrence, des valeurs dépassant 100% ne sont pas adéquates. Le cas présent utilise le lien simple, ce qui veut dire que l'arête définissant la plus grande probabilité de coocurrence est choisi lors des fusions. Le lien simple est connu pour être rapide et consistant pour détecter des regroupements de haute densité tant que ces communautés sont bien divisées.

Données : $G = (V, E)$
Algorithme : Classification ascendante hiérarchique par le lien simple
Résultat : Hiérarchie de partitions
Trier la liste d'arête en ordre croissant de dissimilarité **tant que** $|V| > 1$ **faire**
    pour tous *Élements e de E* **faire**
        Fusionner les deux sommets ayant la dissimilarité la plus basse ;

Le déroulement de cet algorithme est illustré dans l'annexe de l'article. Il est à noter que plusieurs sommets peuvent posséder la même dissimilarité, donc l'ordre de liste d'arêtes peut différer légèrement d'une exécution à l'autre si un tri aléatoire est effectué à priori.

3.3 Graphes aléatoires et information mutuelle normalisée

Cette section fait la revue de certain concepts utilisés pour procéder à la validation des résultats provenant des algorithmes de détection de communautés. Outre les graphes basés sur des cas réels, beaucoup d'auteurs utilisent des générateurs de graphes aléatoires ayant une structure connu afin de pouvoir comparer avec les résultats de l'algorithme. Newman et Girvan (2003) développent un générateur de graphes avec un nombre de communautés fixes. La distribution de la taille des communautés ainsi que celle des degrés des sommets sont également homogènes. À l'aide de ce modèle, Lancichinetti, Radicchi et Fortunato (2008) développent le générateur de graphe *LFR Benchmark* dont la distribution des degrés des sommets et de la taille des communautés suivent une loi de puissance. L'utilisation des graphes *LFR Benchmark* est courante dans la littérature, le modèle est également adapté pour générer des graphes dirigés et des graphes avec communautés imbriqués à deux niveaux.

Danon, Diaz-Guilera et Al. (2005) passent en revue différentes méthodes de détection de communauté et proposent de changer la mesure de validation utilisée par Newman et Girvan (2003) pour l'information mutuelle normatilisée (NMI). Le NMI est une mesure d'information de Shannon commune entre deux partitions, normalisée entre 0 et 1. Le concept est basée sur l'entropie de Shannon couramment utilisée en théorie de l'information. Suite à la publication de l'article, plusieurs auteurs tel que Zhang et Moore (2014) et Lancichinetti, Raddichi et Fortunato (2008) ont adopté le NMI comme mesure. L'équation (3.4) exprime le NMI entre les partitions A et B, représenté par $I(A,B)$, où N dénote somme des éléments de la matrice de confusion pour les ensembles de partitions A et B avec C_A et C_B indiquant le nombre de communautés dans chacune. Les termes $N_{i.}$, $N_{.j}$ et N_{ij} dénotent respectivement la somme des éléments de la rangée i, la somme des éléments de la colonne j et le nombre de sommets de la communauté i présents dans la communauté j. Le logarithme utilisé dans le cadre de ce mémoire est de base 2.

$$I(A,B) = \frac{-2\Sigma_{i=1}^{C_A}\Sigma_{i=1}^{C_B}N_{ij}log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\Sigma_{i=1}^{C_A}N_{i.}log(\frac{N_{i.}}{N}) + \Sigma_{i=1}^{C_B}N_{.j}log(\frac{N_{.j}}{N})} \tag{3.4}$$

Chapitre 4

Article

Finding cores of communities using consensus clustering

Ghislain Benoit, Gilles Caporossi and Sylvain Perron

December 5th, 2017

Abstract

Computational methods are increasingly used to analyse complex systems in order to identify community structure. Most of the currently used methods like modularity optimizing heuristics and other non-deterministic algorithms face a resolution limit problem. The proposed algorithm tries to solve this problem by reusing information from currently existing methods including local-based LPA and LPAm. The probabilities of node cooccurrence in the network are computed from partitions generated by a non-deterministic algorithm. These probabilities are used to create a consensus weighted graph on which is applied an agglomerative version of Kruskal's algorithm resulting in multiscale partitioning. Performance are tested for LPA, LPAm along with a greedier version of Louvain using both LFR hierarchical random graphs and a classical graph theory example. The method proposed is capable of improving output partitions in comparison to source algorithms on two-level embedded communities with consistency. Since the bulk of the process is parallelizable, partitions for graphs containing millions of nodes can be generated in reasonable computational time. Many possible modifications of process as well as other source heuristics are left to be explored.

4.1 Community Detection

The abundance of information in recent decades has raised a lot of interest for all subjects related to data analysis. Graph theory, which has been around for centuries in mathematics, is now being used to represent complex systems in a large range of natural, human and applied sciences. While a very large number of problems has been defined and solved using graph theory, we focus on one specific problem which is the detection of communities.

The informal definition of a community in graph theory is a topological concept that can represent a wide variety of things in the real world such as groups, clusters, modules or many other things according to context. Groups of nodes will often share common properties or serve similar

functions. Analysis of social media is perhaps the most obvious application of community detection. Other applications include clustering of stocks in finance, neurons in neurology, citation networks in the field of epistemology and much more.

Although criteria used by community identification algorithms can vary according to context, goals and complexity of the problem, one commonly accepted notion of a community is of a set of nodes with high internal density in comparison to the outside. There is no precise mathematical definition for a community. Raddichi et Al. (2004) has proposed a weak and a strong condition for defining communities based on comparison between internal and external degrees. There exists many algorithms used to identify community structures. Clique based methods has been in use since the 1950s in mathematical sociology and psychology. In computer sciences, graph partitioning has existed since the 1970s. Works by Girvan and Newman based on edges betwenness led to modularity as function for judging the quality of partitions. Modularity then gained a lot of attention in the community detection literature.

Modularity is essentially a comparison between a graph and its null model, which is a random rearrangement of the edges constrained by expected edge degree of the original graph. Despite being one of the best known quality function, modularity has a bias towards merging smaller communities into large ones. In a real case application, an optimal modularity solution is questionable. It is possible that sub-optimal solutions composed of more numerous communities offer a better portrayal of the network than a few very large communities in some cases.

Methods were subsequently proposed for modularity maximization such as Blondel et Al.(2008), Newman (2004), Clauset, Newman et Moore (2004), Guimera et Amaral (2005) as well as Duch et Arenas (2005) just to name a few. . Furthermore, greedy modularity maximization algorithms produce inconsistent results because it depends on its initial node order. Modularity can only be used for comparing different partitions of the network itself and cannot be compared with other networks since its values are affected by the size of the network. The label propagation methods are also widely used as very fast local-based greedy algorithms. The method proposed here has been tested with two label propagation methods and will therefore be covered in the methodology section. One last algorithm worth mentionning here is Infomap which uses random walks to detect patterns which in term can be used to identify hierarchical communities .

In the following sections, we take a look at the resolution limit problem and consensus clustering. The basics of the proposed method is then presented along with the results of benchmark tests. We conclude this article with a discussion.

4.2 The Resolution Limit Problem and Consensus Clustering

As just mentionned, modularity itself has a resolution limit. As Isogai (2014) explains, it is difficult for the nul model to have a resolution that satisfies itself both large and small scale clusters.

This is problematic because real graphs do not necessarily have homogeneous community sizes. Different methods have been proposed to face this problem. A basic approach could be multiscale resolution of modularity, that is reoptimising modularity on the community partitions first obtained as subgraphs. However, Lancichinetti et Fortunato (2011) point out that resolution limit persists even with multiscale resolution and possibly with any method having a global optimisation criterion, and that would include OSLOM and Infomap. This points out to local-based critera algorithms for more precise community in heterogeneous networks.

The algorithm proposed by Hollocou et al. (2017) has a linear computationnal complexity, which is extremely low. The metric used is not exactly conductance but has a strong mathematical resemblance. Tests on real networks has shown a better precision than Louvain and other commonly used algorithms for very large networks. However, for networks of a more modest size, the precision of their algorithm is clearly lower .

Yin et Al. (2017) propose a measure of partition quality based on motif conductance that can deal with directed graphs unlike other algorithms using this type of metric . Chakraboty et Al. (2014) introduce a metric called *permanence* that considers both internal and external connectivity of the vertex. The algorithm they propose has a precision that is comparable to commonly used algorithms on well defined community and a better precision for communities that are harder to define such as LFR Benchmark graphs with having a high mean ratio between nodes degrees external to their communities and their total degrees. Although these algorithms seem promising, they can only offer one single partition at the end of the process. Real complex systems abound with larger communities that can be subdivided into smaller ones as observed by Ravasz et Al. (2002) in metabolic networks .

Also recently, Gomez et Al. (2015) developed a two stages algorithm which divides the graph by edge betweenness and applies a kruskal-like algorithm to link edges based on a dissimilarity measure. This create a forest graph containing information from each step of the divide process which enables hierarchical clusters. This feature is interesting because it can look within a large community such as one would get from an algorithm with resolution limit and identify community cores. Bodlaj et Batagelj (2015) developed two hierarchical graph clustering algorithms using a local-based monotonous dissimilarity metric. The first version of the algorithm joins nodes based on link dissimilarity, the second version is based on line graphs and joins edges instead of nodes. This algorithm is able to offer overlapping as well as scalable partitions.

Ghosh et Strehl (2002) discuss cluster ensembles methods in which multiple processes are used to generate a certain number of partitions. Partitions are then combined by a consensus function. Topchy et Al. (2005) observe that ensemble methods make results much less sensible to noise and extreme values. Moreover, data integration methods are known to generate more stable and consistent outputs according to both Grosh et Strehl (2002) as well as Lancichinetti et Fortunato (2012), possibly obtaining previously unattainable solutions (Topchy et Al. (2005)).

There exists applications of cluster ensembles problems pertaining to community detection based on pairwise membership probability, which is defined by the probability of two nodes being in the same community. Lancichinetti et Fortunato (2012) along with Kwak, Moon et Al. (2011) developed consensus clustering algorithms with this property reincorporated into graph model as an edge attribute. In both cases, the algorithm repeats consensus clustering, changing the link of the graph until all link weights has converged to 0 or 1. The algorithm described by Lancichinetti et Fortunato (2012) has a filtering step before each iteration in which edges with weights inferior to a certain value are considered noisy and removed. This reduces computing time since it makes the graph more sparse. Consensus clustering has been tested with many modularity maximizing methods including Louvain as well as with other clustering algorithms. The tests have been done on both real-world applications and randomly generated graphs such as LFR benchmark graphs (based on the planted l-partition model by Condon et Karp (2001) adapted for creating benchmark graphs by Girvan et Newman (2008) and implemented for variable degrees and community sizes by Lancichinetti, Fortunato and Radicchi (2012)). Results were more accurate when comparing consensus algorithms with source algorithms according to measured normalized mutual information (NMI) between obtained partitions and planted partitions of random graph models or otherwise with the actual solution if is it known for real cases according to Nascimento, Santos and Carvalho (2016), Lancichinetti et Fortunato (2012), Zhang et Moore (2014) as well as Wen, Jian-Ping et Fan (2014). Other studies have used consensus clustering to compute scalable and overlapping communities (Zhang et Moore (2014)) as well as dealing with dynamic community problems (Lancichinetti and Fortunato (2012)) and imputing edges in incomplete graphs (Burgess, Adar et Carafella (2016)). More recently, consensus clustering has been adapted for directed graphs by Nascimento, Santos and Carvalho (2016).

We have seen recent development pertaining to consensus clustering which constitute the basis of the method presented in this article along with other methods aiming to resolve the resolution limit problem. The use of consensus clustering in conjunction with a local-based algorithm such as LPA can deliver a resolution-free set of partitions while preserving the quality of modularity on a broader level. The following section presents the application of a classical hierarchical clustering algorithm to pairwise membership graphs obtained from consensus clustering. The proposed algorithm can scale down communities to its core nodes or any other degree of dissimilarity. Combining solutions can show core nodes contained into their larger community. In addition, single partitions obtained with simulated cut points are to be tested on LFR benchmarks graphs. Results can be obtained at any level of hierarchical clustering offering more versatility than other algorithms.

4.3 Proposed Method

This section exposes the procedures by which this algorithm is implemented. The basics of sources algorithms tested here are presented followed by the consensus clustering process and testing methods. This work was programmed using C++. The two following equations are equi-

valent and are used to define modularity which will be used both as a criterion for one of the source algorithm and for choosing a cut point in the subsequent hierarchical classification phase.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{4.1}$$

Equation 4.1 is taken from Blondel et Al. (2008). It is computed for all pair of vertex i and j, m is the total edge weight of the graph. A_{ij} is the weight of the link between i and j, k_i and k_j represents the degree of vertex i and j. $\delta(c_i, c_j)$ is called the Kronecker delta and equals 1 if i and j are in the same community and 0 otherwise .

$$Q = -\sum_{i} \left[\left(\frac{2in_i + out_i}{2m} \right)^2 - \frac{in_i}{m} \right] \tag{4.2}$$

In equation 4.2 above, taken from Aloise et Al. (2010), i represents a cluster, in_i, the number of edges that are internal to i, out_i, the number degree of edges that are extermnal to i and m, the total edge weight of the graph. For practical purposes, we use Equation 4.2 which is a sum on all communities and allows to quickly recalculate modularity following local changes. The first step of the algorithm is to generate a number of solutions which will be used to compute the pairwise membership probability of both ends from each link. This article evaluates performance of three different methods, that is a modified version of Louvain which fuses nodes after each community alteration and two label propagation algorithms. Each method is compared with the outcome against the case in which the single highest modularity generated partition was used. Louvain is a fast heuristic that seeks the maximal modularity partition but does not guarantee an optimal solution. Louvain is a typical greedy algorithm. It is non deterministic in nature since it depends on the initial order of its randomly ordered list of nodes. This property allows the Louvain method to be executed a large number of times to generate partitions. Despite the flaws described earlier with modularity, Louvain is pertinent because it can provide a random distribution of quality solutions at relatively low computational cost. The modified version of Louvain tested in this article is described below. It is more greedy than Louvain since it aggregates nodes after each step which results in faster execution, increased variability and a slight loss of modularity on average.

The label propagation algorithm (LPA) by Barber and Clark (2008) also possesses the non-deterministic property required for generating partition ensembles. LPA is much less stable in comparison with the Louvain method. This higher variance of potential output is actually an interesting property in the context of pairwise membership probability generation. Label propagation algorithms presented in this article are also much faster than Louvain but does not give better solutions than Louvain in terms of modularity. This is not a problem since our stated goal is not

Modified version of Louvain
Data : $G = (V, E)$
Let each node v be its own community
Let Q be current graph modularity
Let q be graph modularity after fusion of v and n
Create a randomly ordered list L of all nodes v
for all $v \in L$ **do**
  Create a list l_n of neighbors to v
  $n' = \emptyset$
  $\tilde{q} = -\infty$
  for all Elements n of l_n **do**
    if $q > \tilde{q}$ **then**
      $n' \leftarrow n$
      $\tilde{q} \leftarrow q$
    end if
  end for
  if $\tilde{q} > Q$ **then**
    Fuse v and n'
    Add the node created by the fusion to L_v
    Remove v and n' from L_v
    $Q \leftarrow \tilde{q}$
  end if
end for

modularity maximization.

In the label propagation model, a label corresponds to community. At the initial state of LPA, every node has its own label. During each iteration, nodes are assigned the most frequent label in its neighbourhood. The algorithm stops when no labels are left to modify. According to Raghavan et Al (2007), LPA is the fastest community detection algorithm so far with near linear time complexity. Equation 4.3 describes how labels are chosen for each node iteration where L_x^{new} represents the new label assigned to node x, A_{ix}, the link weight between nodes i and x. $\delta(l_i, l)$ represents Kronecker's delta which takes a value of 1 if current label is the same as node i and 0 otherwise ,

$$L_x^{new} = \arg\max_b \left(\sum_{i=1}^{n} A_{ix} \delta(l_i, l) \right). \tag{4.3}$$

Developped by Liu et Murata (2010), LPAm is a slight variant of the original LPA that prevents trivial solutions such as the whole graph being one partition by maximising modularity in the neighborhood of x. As the authors point out, the criterion for LPAm label selection often

leads the algorithm into a path of solutions that maximize local modularity . Equation 4.4 shows the selection criterion for LPAm where L_x^{new} represents the new label assigned to node x where $B_{ix} = A_{ix} - \frac{k_i k_x}{2m}$ (same as in equation 4.1) and $\delta(l_i, l)$ represent Kronecker's delta (same as in equation 4.3).

$$L_x^{new} = \arg\max_b \left(\sum_{i=1}^{n} B_{ix} \delta(l_i, l) \right), \tag{4.4}$$

The initial phase of the algorithm presented in this article is to launch an adequate source algorithm a large number of times in order to obtain the necessary information to execute a hierarchical classification process on a consensus graph. In the next phase, the proportion linked vertices contained within the same partition can be calculated. This percentage, known as pairwise membership probability, can be interpreted as probability that these two vertices are in the same community after following the application of source algorithm. Since this is bound $[0,1]$, substracting this proportion by 1 consequently defines a metric for dissimilarity that can be expressed in a $n \times n$ size matrix. The algorithm described here adds dissimilarity as an edge attribute, discarding information between nodes sharing no edges. Operating on a list of edges has the advantage of being much more efficient than a dissimilarity matrix on a sparse graph. The reconstruction of the original graph is followed by a classical clustering comparable to a naive algorithm operating in graph space (It is the same as Kruskal (1956) algorithm for finding minimum spanning trees with the exception that the pair of nodes connected by the selected link at each step are fused together). This clustering algorithm is described below. Dissimilarity of edges has to be defined in the event where pairs of nodes that have links to another cluster are grouped together because it does not have the same properties as link weights that are typically added together. Doing so with this metric for dissimilarity will most likely result in values over 100% which does not make much sense. In this case, the algorithm applies a single linkage, which means the algorithm chooses the link with the highest pairwise membership probability over the other ones. Single linkage is known for being fast and consistent when detecting high density clusters provided that there exists a separation between communities that is clear enough .

Data: $G = (V, E)$
Sort list of edges by increasing dissimilarity values
while $|V| > 1$ **do**
  For all Elements e of E **do**
    Fuse the two nodes whose link have the lowest dissimilarity
  End for
end while

The execution of this algorithm is illustrated in the appendix. It should be noted that since multiple links can have the same dissimilarity, the order in which the list of edges is sorted can slightly

change the results on multiple runs if the list of edges is randomly sorted before the process.

4.4 Results

A lot of raw information can be extracted from this algorithm. Since it is a hierarchical clustering method, a partition can be obtained at each step of the process. Partitions at the beginning of the process can identify the core of communities as the link fused always has maximal strength. Early clique-like solutions can be combined with later larger communities to express hierarchical communities. This is illustrated in Figure 10 with the famous Zachary's Karate Club example. This solution is similar to what is obtained by other algorithms, with additionnal information on core relationships between nodes.

Single partitions are obtained by choosing a cut point in the sequence. This can be done with definable mathematical criteria or also intuitively by graph scree with values at given time of the process such as in Figure 1. Here, two cut points have been simulated for the sake of testing the algorithm on hierarchical LFR Benchmark Graphs. The first is done by taking the partition with maximum modularity, the second one by stopping when all links that connect nodes with 100% pairwise membership are fused. This last cut point provides a solution with smaller and more numerous communities in comparison with the first one. Figure 1 shows for the specific case of Zachary's Karate club the relation between the number of components in the graph with modularity, dissimilarity of the last fused link (defined in this case as 1 minus the pairwise membership probability), the number of nodes contained in the largest component and the number of trivial components (or the number single node communities). We can see that modularity consistenly grows and peaks when the hierarchical process stops at 4 communities before dropping. We can also note that nodes fused early on in the process virtually always end in the same communities. Those core communities are fused together towards the end of the process. Nodes that does not clearly belong to any community are only fused at the end of the process.

Different large sets of hierarchical LFR Benchmark graphs with fixed parameters have been generated using the application supplied by Santo Fortunato on his personal website[23]. Those parameters are the numbers of nodes, the average degree, the maximum degree and power law exponents γ for the degree distribution and β for community distribution. A variant of the application, also supplied by Fortunato was used to generate hierarchical graphs. This particular graph generator creates graphs with two levels of communities, that is micro-communities embedded into a macro one. In this case, there is a set of parameters for each level. The possible range of values for each parameter depends on the value of other parameters in order to generate graphs. Each generated graph is tested for connectivity, disconnected graphs are discarded.

Danon et al. (2005) propose to measure statistical information shared between random graph generator solutions and this algorithm's output partitions can be measured by normalized mutual information. Mutual information is the mutual entropy between two random variables measured

/home/kaizoku/Documents/article/graphes.jpg

Figure 1 – Execution of a hierarchical consensus graph clustering algorithm with simple link on the famous example of Zachary's Karate Club. Number of consensus partitions = 100

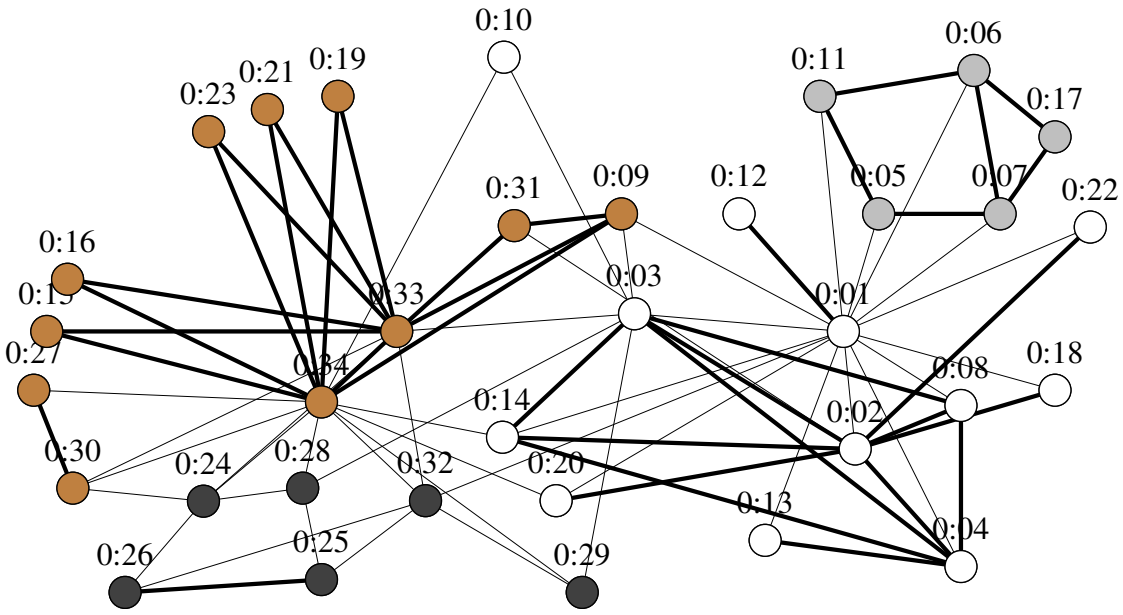


Figure 2 – Solution when number of communities = 4 (chosen by graph scree using figure 1). Edges with dissimilarity < 0.01 are displayed in bold. Number of consensus partitions = 100

in bits. This measure is then normalized to fit between 0 and 1 (see equation below), 0 indicating that the partitions are entirely different and 1 indicating that the partitions are identical. Equation 4.5 expresses NMI, where N denotes the sum of elements of the confusion matrix for partitions A and B with C_A and C_B indicating the number of communities in A and B. Terms $N_{i.}$, $N_{.j}$ and N_{ij} is respectively the sum of elements in row i, the sum of elements in column j and the number of nodes from community i existing in community j. Other authors such as Zhang et Moore (2014) and Lancichinetti, Radicchi et Fortunato (2008) have used NMI for this purpose.

$$I(A,B) = \frac{-2\Sigma_{i=1}^{C_A}\Sigma_{i=1}^{C_B}N_{ij}log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\Sigma_{i=1}^{C_A}N_{i.}log(\frac{N_{i.}}{N}) + \Sigma_{i=1}^{C_B}N_{.j}log(\frac{N_{.j}}{N})} \tag{4.5}$$

In previous articles using LFR Benchmark Graphs, the average NMI between the partition obtained by community detection algorithms and the generated partitions is expressed in function of the mixing parameter μ which is the average ratio between a node's degree outside its community in respect to its total degree. Values over 0.6 are often not tested because for any values of μ greater

than 0.5, the generated communities are not expected to respect the weakest quantitative definition of a community offered by Raddichi et al. (2004), or in other words, the sum of the outside degrees for each node of a community is expected to be higher than the sum of degrees for each node within the community. Keeping this range for μ ensures that the generated communities do have a relevant community structure that is detectable and suitable for testing. In the case of two-level hierarchical graphs, μ_1 is the average ratio between node's degree outside of its macro and micro communities compared to total. μ_2 corresponds to the average ratio of the node's degree within its macro community but outside its micro community. The value $1-\mu_1-\mu_2$ is therefore equal to the ratio of the node's degree within its micro community. We chose to test a few cases where the values of $\mu_1+\mu_2$ does not exceed 0.6 while maintaining a broad distribution of μ parameters.

The following result tables show mean NMI between hierarchical LFR Benchmark Graph solutions and output partitions for LPA, LPAm and the modified version of Louvain using consensus clustering. We will refer to those as consensus LPA, consensus LPAm and consensus Louvain. The LFR graphs have been generated 10 000 times with 120 consensus partitions, 1 000 nodes, an average degree of 10, a maximum degree of 50, a minimum micro community size of 5, a maximum micro community size of 25, a minimum macro community size of 50 and a maximum macro community of 250. The exponent parameter for node degree and community degree for power law distribution are 2 and 1 respectively. Within the table, the macro and micro row corresponds to the macro community partition and micro community partition. Columns maxQcut and robcut designate the cut points used for obtaining the output partition in the hierarchical process. MaxQcut is when modularity is at its highest, robcut is when all link fused have a pairwise membership of 1. We will refer to those as modularity cut and robustness cut. Column maxQ is shown for comparison, it corresponds to the maximum modularity partition obtained during the initial step.

μ_1, μ_2	LPA		
01, 01	maxQ	maxQcut	robcut
macro	0.715821	0.749218	0.683029
micro	0.97534	0.944333	0.971007

μ_1, μ_2	LPAm		
0.1, 0.1	maxQ	maxQcut	robcut
macro	0.67976	0.703083	0.460626
micro	0.966799	0.977756	0.706084

μ_1, μ_2	Modified Louvain		
0.1, 0.1	maxQ	maxQcut	robcut
macro	0.74344	0.780496	0.617062
micro	0.845769	0.893314	0.843329

Figure 3

In figure 3, values of $\mu correspond to the case where communites are clearly defined but the distinction between mi$

μ_1, μ_2	LPA		
0.2, 0.4	maxQ	maxQcut	robcut
macro	0.99999	0.999986	0.603614
micro	0.630306	0.630316	0.798011
μ_1, μ_2	LPAm		
0.2, 0.4	maxQ	maxQcut	robcut
macro	0.619388	0.706246	0.441029
micro	0.770748	0.709661	0.761102
μ_1, μ_2	Modified Louvain		
0.2, 0.4	maxQ	maxQcut	robcut
macro	0.878305	0.969472	0.635487
micro	0.581679	0.630387	0.691462

Figure 4

Figure 4 shows the results when macro communities are well defined but not micro communities. The distinction between the micro and macro communities is a little bit clearer than in the previous table. Max modularity and modularity cut offers a nearly perfect score for macro communities with LPA. As expected, results here show that modularity cut and robustness cut respectively improves the NMI for macro and micro communities with the exception of LPAm micro communities.

μ_1, μ_2	LPA		
0.3, 0.3	maxQ	maxQcut	robcut
macro	0.925166	0.910677	0.516819
micro	0.639135	0.67444	0.821784
μ_1, μ_2	LPAm		
0.3, 0.3	maxQ	maxQcut	robcut
macro	0.551922	0.561904	0.440239
micro	0.796827	0.74496	0.761452
μ_1, μ_2	Modified Louvain		
0.3, 0.3	maxQ	maxQcut	robcut
macro	0.712432	0.823894	0.511748
micro	0.523383	0.640306	0.745721

Figure 5

The case illustrated in figure 5 is when communities are not clearly defined but distinction between macro and micro levels within the communities are clear. Here, consensus modified Louvain is the only case where both modularity cut and robustness cut respectively improves NMI for macro and micro communities as intended. Consensus LPA with robustness cut offers

a large improvement for micro communities and dominates other consensus algorithm in. Yet, it slightly underperforms on macro communities in comparison with the max modularity partition. Consensus LPAm in this case offers improvements for macro communities and underperforms on micro communities compared to the max modularity partition.

μ_1, μ_2	LPA		
0.4, 0.2	maxQ	maxQcut	robcut
macro	0.573968	0.618444	0.491177
micro	0.746343	0.844966	0.826036
μ_1, μ_2	LPAm		
0.4, 0.2	maxQ	maxQcut	robcut
macro	0.495458	0.449505	0.440859
micro	0.812876	0.754664	0.761547
μ_1, μ_2	Modified Louvain		
0.4, 0.2	maxQ	maxQcut	robcut
macro	0.399609	0.534495	0.454994
micro	0.492576	0.703939	0.769135

Figure 6

Case showed in figure 6 corresponds to a harder situation where macro communities are not well defined and the distinction between micro and macro communities is not clear. Consensus LPA again dominates other consensus algorithms but is dominated on macro communities by the max modularity partition. LPAm seems unefficient compared to max modularity, conversely Consensus modified Louvain offers a large improvement on max modularity.

μ_1, μ_2	LPA		
0.1, 0.5	maxQ	maxQcut	robcut
macro	1	1	0.750649
micro	0.629461	0.629461	0.722458
μ_1, μ_2	LPAm		
0.1, 0.5	maxQ	maxQcut	robcut
macro	0.68349	0.840752	0.440261
micro	0.738609	0.670707	0.760915
μ_1, μ_2	Modified Louvain		
0.1, 0.5	maxQ	maxQcut	robcut
macro	0.956461	0.994013	0.801072
micro	0.612675	0.628628	0.65478

Figure 7

Figure 7 above has clearly defined macro communities with a clear distinction between the macro and micro levels. Here, all consensus algorithms show improvements on maximum modularity except for LPA on the macro level where both maximum modularity and modularity cut scores perfect.

μ_1, μ_2	LPA		
0.5, 0.1	maxQ	maxQcut	robcut
macro	0.421816	0.508302	0.487862
micro	0.781011	0.875482	0.825187
μ_1, μ_2	LPAm		
0.5, 0.1	maxQ	maxQcut	robcut
macro	0.461157	0.38577	0.440979
micro	0.819433	0.758059	0.761617
μ_1, μ_2	Modified Louvain		
0.5, 0.1	maxQ	maxQcut	robcut
macro	0.16393	0.371174	0.445728
micro	0.484049	0.730157	0.768228

Figure 8

Figure 8 is the hardest case to solve with macro communities badly defined overall and the macro and micro levels barely distinguishable. We see consensus LPA again dominating other algorithms in terms of NMI and both LPA and Louvain showing improvements over maximum modularity. LPAm in this case is inefficient.

Overall, establishing the cut point at maximal modularity or at 100% pairwise membership in the hierarchical process outperforms the maximal modularity partition for micro and macro communities and for all three base algorithms, with a few exceptions. Even when the maximum modularity partition outperforms the others, it has an NMI which is in a close range from the others while in some cases the maximum modularity partition clearly underperforms. Partitions obtained from establishing the cut point to pairwise membership values inferior to 1 have a strong NMI on average in relation to the other two criteria for the case of micro communities while the maximum modularity cut point tends to perform better for macro communities. The proposed consensus clustering technique is especially useful with LPA, improving it to the point of beating this modified Louvain in all cases except when (μ_1, μ_2) equals $(0.1, 0.1)$ but sometimes dominated by LPA maximum modularity. When it comes to consensus clustering, it seems that LPA's instability becomes a great advantage because it provides much more information about community structure when executed a large number of times. Consensus LPAm on the other hand is the least precise.

4.5 Discussion

In this paper, we propose to apply a parallelizable vote technique to the problem of finding embedded communities in networks. Namely, applying a large number of times an algorithm that yields a portion of randomness in its results, and using a vote approach provides better results than each of the individual run of the algorithm. This principle is exactly the same as used in random forests by Breiman (2001) and proves to be efficient in improving stability and precision of LPA, LPAm and Louvain. The proposed algorithm has two steps. In the first one, a non-deterministic algorithm for finding communities in networks is used, either LPA, LPAm or a simplified version of the Louvain's algorithm is used repeatedly to identify some strong patterns in the data. The second step exploits the results from the first within a hierarchical clustering scheme. This last step is used to identify communities with various level of homogeneity, which leads to the identification of embedded communities. The current work is still exploratory and still needs to be refined. Further testing is required to demonstrate whether this method in conjunction with LPA is an adequate solution to the resolution limit problem even if it seems theoretically viable. Testing methods for accuracy also need improvements. The choice of the algorithms to use in the first step, is still a question to answer. A lot of alternatives can be tried, including a combination of different ones. If the choice of hierarchical clustering for the second step seems natural, the best criterion to use remains to be identified (Ward, single linkage or complete linkage for instance). Nevertheless, the potential of vote methods for community detection in networks is now clear.

The methods of consensus clustering used in this paper have also been applied in an upcoming study following up the work of Claveau et Gingras (2016). This article provides an epistemological analysis of dynamic communities found by executing Louvain on a series of economic citation graphs constructed from *Web of Sciences*. A similar process has been implemented, this time using this consensus clustering method which is more stable and has the ability to reveal community cores embedded in communities. Preliminary subjective analysis of clusters by text mining validification shows communities and cores which are coherent to one another and to the studied discipline (Political Sciences in this case) with seemingly best results obtained with the modified version of Louvain as source.

Chapter 5

Conclusion et application du mémoire

Dans ce mémoire, nous avons mis en œuvre une méthode qui vise à améliorer la robustesse des résultats de détection de communautés. Pour ce faire, nous avons programmé une version modifiée de l'algorithme de Blondel , ainsi que LPA et LPAm. Les résultats obtenus par ces algorithmes lancés un grand nombre de fois sont ensuite compilés et exploités dans le cadre d'une classification hiérarchique. Si l'idée d'utiliser un grand nombre de réplications pour améliorer la performance de la méthode n'est pas nouvelle, à notre connaissance, notre approche qui consiste à compiler les résultats à l'aide de classification hiérarchique est novatrice et elle permet d'identifier des communautés naturelles et imbriquées.

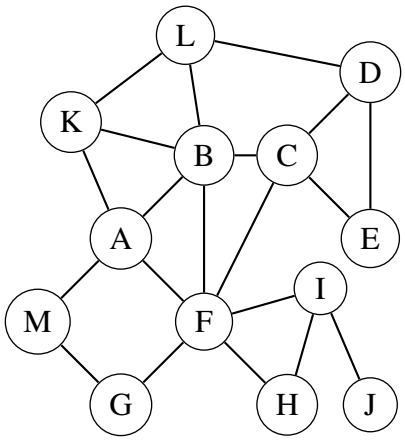
Ce mémoire offre donc un moyen efficace pour améliorer la stabilité et la précision des algorithmes tout en tentant d'éviter le problème de limite de résolution. Les tests effectués avec les graphes aléatoires LFR démontrent clairement la validité de cette méthode avec deux des trois algorithmes sources mentionnés si l'on compare à l'aide de information mutuelle normalisée les partitions générées avec la structure de communautés connue de ces graphes. La méthode s'avère particulièrement efficace avec LPA, un algorithme de propagation d'étiquette extrêmement rapide (de difficulté computationnelle linéaire) mais naturellement instable. La méthode présentée dans ce mémoire semble prendre avantage de cette instabilité pour soutirer un maximum d'information sur la structure des communautés. Les tests montrent également une amélioration de la précision des résultats avec Louvain modifié. Cette amélioration est moins claire avec LPAm. Le point de coupure maximisant la modularité performe surtout bien avec les macro-communautés tandis le point de coupure au moment où la cooccurence des arètes est de 100% semble offrir des partitions plus précises pour détecter les micro-communautés. Le temps computationnel supplémentaire requis par le grand nombre d'exécutions est en partie réglé par la possibilité de parallélisation de la charge de travail. Simple à implémenter, la méthode est également compatible avec n'importe quel algorithme de détection de communauté non déterministe par partition, ce qui ouvre la porte à un large éventail de possibilité pour des travaux futurs incluant l'utilisation de plusieurs algorithmes sources. La méthode hiérarchique utilisée en phase finale comporte également plusieurs options à explorer, tel que l'utilisation du lien complet au lieu du lien simple, ou encore la distance de Ward. L'utilisation de LPA en consensus semble viable d'un point de vue théorique pour résoudre le problème de limite de résolution étant donné la localité de son critère de sélection mais une

vérification appropriée est nécessaire.

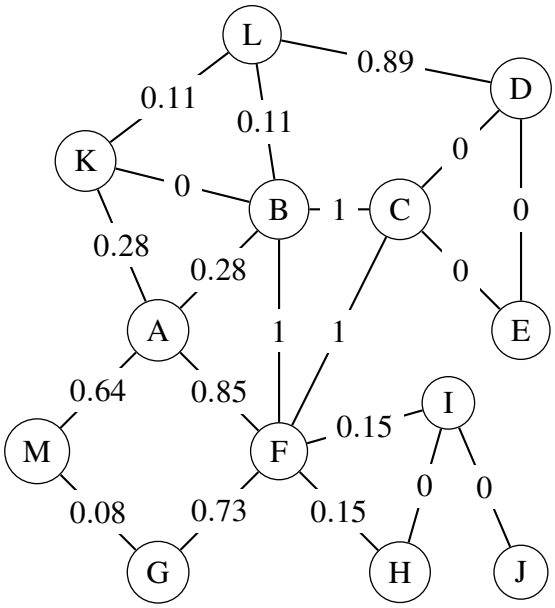
Les méthodes utilisées dans le cadre cet article ont également été utilisées dans une étude suivant les travaux de Claveau et Gingras (2016) mentionné précédemment sur un corpus de sciences politiques. Un processus similaire fut implanté pour obtenir des résultats dynamiques tel qu'expliqué précédemment mais en utilisant le regroupement par consensus offrant des résultats beaucoup plus stables et avec la possibilité d'obtenir les noyaux des communautés imbriquées dans celle-ci. Une analyse préliminaire et subjective suite à une validation par une analyse textuelle des titres contenus dans les regroupements montrent une cohérence entre les communautés et leurs noyaux ainsi qu'avec la discipline étudiée, les meilleurs résultats étant obtenus avec la version modifié de Louvain. L'algorithme peut s'exécuter dans un temps raisonnable (1h) pour des réseaux ayant jusqu'à 16 millions d'arêtes. Ce temps d'exécution s'allonge à plus d'une semaine sur un graphe avec plus de 100 millions d'arêtes. Quoique la méthode consensuelle requiert davantage de travail, elle s'exécute quand même rapidement en comparaison avec les algorithmes gourmands couramment utilisés pour des graphes de cette envergure puisque la charge de travail supplémentaire peut être dédoublée en différents fils d'exécution.

Appendix

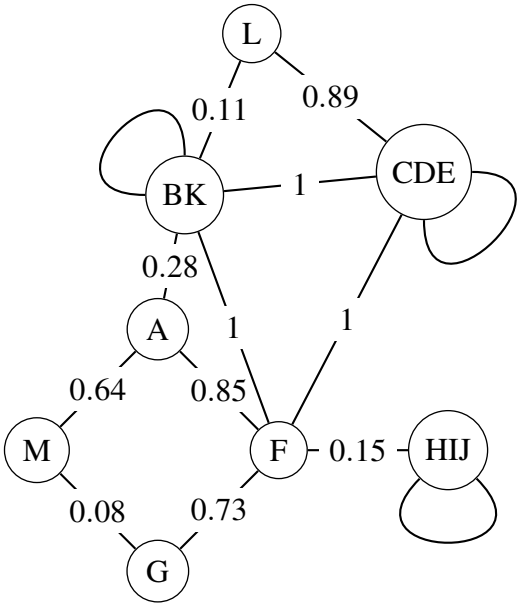
Figure 1 – Illustrated example of the algorithm's execution



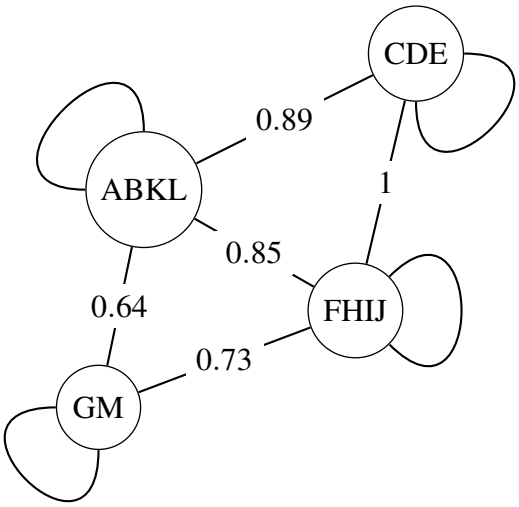
Representation of a simple graph



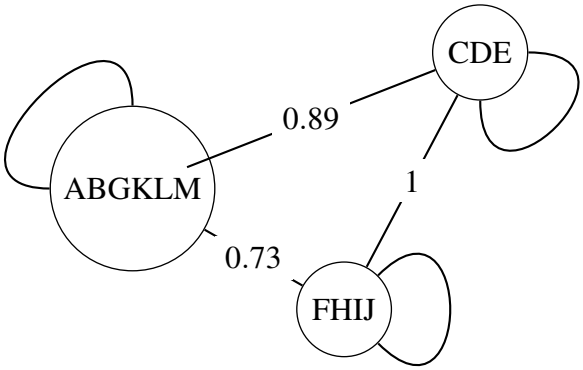
Link dissimilarity established with random Louvain partitions



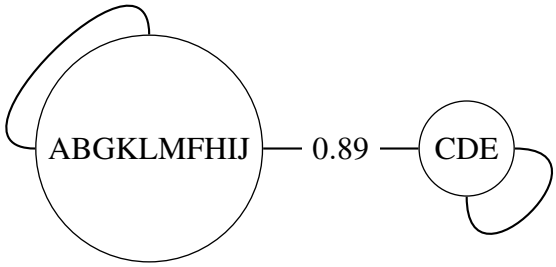
Fusion of link with dissimilarity of 0



Fusion of link with dissimilarity < 0.5



Fusion of link with dissimilarity < 0.7



Fusion of link with dissimilarity < 0.85

Bibliography

[1] A.Condon and R.M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18:116–140, March 2001].

[2] Richard D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.

[3] Daniel Aloise, Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Leo Liberti, and Sylvain Perron. Column generation algorithms for exact modularity maximization in networks. *Physics Review*, 82, 2010.

[4] M. J. Barber and J. W. Clark. Detecting network communities by propagating labels under constraints. *Physical Review*, 80, 2009. Art. ID 026129.

[5] Marc Barthélemy and Santo Fortunato. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 104:36–41, 2006.

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

[7] Jernej Bodlaj and Vladimir Batagelj. Hierarchical link clustering algorithm in networks. *Physical Review*, E 91, 2015. Art. ID 062814.

[8] U Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25:163–177, 2001.

[9] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[10] Matthew Burgess, Eytan Adar, and Michael Cafarella. Link-prediction enhanced consensus clustering for complex networks. June 2015. [Online; http://arxiv.org/abs/1506.01461; accessed 12-feb-2016].

[11] Sonia Cafieri, Alberto Costa, and Pierre Hansen. Adding cohesion constraints to models for modularity maximization in networks. *Journal of Complex Networks*, November 2014.

[12] Tanmoy Chakraborty, Sriram Srinivasan, and Niloy Ganguly. On the permanence of vertices in network communities. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1396–1405, August 2014.

[13] Zhang J. Chen, Yong He, Pedro Rosa-Neto, Jurgen Germann, and Alan C. Evans. Revealing modular architecture of human brain structural networks by using cortical thickness from mri. *Cerebral Cortex (New York, NY)*, 10:2373–2381, 2008.

[14] Y. Choi, Y.-H. Eom, H. Jeon, H. Kwak, and S. Moon. Consistent community identification in complex networks. *Journal of the Korean Physical Society*, 59:3128–3132, November 2011.

[15] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physics Review E70*, pages 321–330, 2004.

[16] François Claveau and Yves Gingras. Macrodynamics of economics: A bibliometric history. *History of Political Economy*, 48:551–592, 2016.

[17] Leon Danon, Albert Diaz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, November 2006.

[18] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, Volume 2005:219–228, 2005.

[19] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physics Review*, 72, 2005.

[20] P. Erdös et A. Rényi. On random graphs i. *Publicationes Mathematicae*, 6:290–297, 1959.

[21] R.D. Luce et A.D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14:95–116, 1949.

[22] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:74–174, February 2010.

[23] Santo Fortunato. Santo Fortunato's home page software. 2016. [https://sites.google.com/site/santofortunato/inthepress2; accessed 17-june-2016].

[24] Linton Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, March 1977.

[25] Joydeep Ghosh and Alexander Strehl. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 4:583–617, December 2002.

[26] Roger Guimera and Luıs A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[27] Daniel Gómez, Edwin Zarrazola, Javier Yáñez, and Javier Montero. A divide-and-link algorithm for hierarchical clustering in networks. *Information Sciences*, 316:308–328, 2015.

[28] Alexandre Hollocou, Thomas Bonald, Julien Maudet, and Marc Lelarge. A linear streaming algorithm for community detection in very large networks. *CoRR*, abs/1703.02955, March 2017.

[29] Takashi Isogai. Clustering of japanese stock returns by recursive modularity optimization for efficient portfolio diversification. *Journal of Complex Networks*, 2:557–584, 2014.

[30] J.A.Hartigan. Consistency of single linkage for high-density clusters. *Journal of American Statistical Association*, 76:388–394, 1981.

[31] D.J. Klein and M.Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95, 1993.

[32] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *PNAS - Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[33] Andrea Lancichinetti and Santo Fortunato. Limits of modularity in community detection. *Physical Review*, 84, 2011.

[34] Andrea Lancichinetti and Santo Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, March 2012.

[35] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review*, 78, 2008.

[36] Andres Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review*, E80, 2009.

[37] X. Liu and T. Murata. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 389:1493–1500, 2010.

[38] R.D. Luce. Connectivity and generalized cliques in sociometric group. *Psychometrika*, 15:169–190, 1950.

[39] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *IET Systems Biology*, 2:89–119, 2007.

[40] M.Girvan and M.E.J. Newman. Community structure in social and biological networks. *PNAS, Proceedings of the National Academy of Sciences (USA)*, 99:7821–7826, June 2002.

[41] Robert J. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13:161–173, 1979.

[42] M.E.J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, 64, 2001.

[43] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physics Review E69*, June 2004.

[44] M.E.J. Newman. Modularity and community structure in networks. *PNAS*, 103:8577–8582, 2006.

[45] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *PNAS*, 103:8577–8582, 2003.

[46] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *PNAS - Proceedings of the American Mathematical Society*, 101:2658–2663, 2004.

[47] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale network. *Physical Review*, 76, 2007. Art. ID 036106.

[48] R.Andersen, F. Chung, and K.Lang. Local graph partitionning using pagerank vectors. *FOCS 2016*, pages 475–486, 2006.

[49] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical organization of modularity in metabolic networks. *Sciencemag*, 297, 2002.

[50] Martin Rosvall and Carl Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoA ONE*, 6, April 2011. e18209.

[51] Camila Pereira Santos, Desiree Maldonado Carvalho, and Maria C.V. Nascimento. A consensus graph clustering algorithm for directed networks. *Expert Systems with Applications*, 54:212–135, 2016.

[52] J Scott. Social network analysis: A handbook. pages 8–9, 1991.

[53] Alexander Topchy, Anil K. Jain, and William Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1866–1881, 2005.

[54] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. *CoRR*, abs/cs/0702048, 2007. [Online; http://arxiv.org/abs/cs/0702048; accessed 17-june-2016].

[55] Lian Zong Wen, Li Jian-Ping, Yan Fan, and Athina Petropulu. Detecting community structure using label propagation with consensus weight in complex network. *Chinese Physics B*, 23, 2014. [Number 8].

[56] Hao Yin, Jure Leskovec, Austin R. Benson, and David F. Gleich. Local higher-order graph clustering. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564, August 2017.

[57] Pan Zhang and Cristopher Moore. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *PNAS, Proceedings of the National Academy of Sciences (USA)*, 111:18144–18149, 2014.