Détection de communautés, étude comparative sur graphes réels

Emmanuel Navarro — Rémy Cazabet

IRIT, Université de Toulouse.
118 Route de Narbonne, 31062 Toulouse Cedex 9, France.
Email : {navarro, cazabet}@irit.fr

RÉSUMÉ. *De nombreux algorithmes de détection de communautés ont été proposés ces dernières années. La méthode d'évaluation habituellement employée consiste à retrouver des communautés connues sur des graphes générés automatiquement, graphes d'un réalisme limité. Dans cet article, nous essayons au contraire de comparer les résultats sur des grands graphes réels. Ne pouvant connaître la décomposition correcte de ces graphes, nous comparons d'abord les solutions des différents algorithmes entre elles, puis la robustesse de ces solutions sur des versions re-câblées des graphes. Nous montrons par ces moyens que, d'une part, les graphes réels ont des communautés beaucoup plus difficiles à détecter que les graphes générés (sur les graphes réels, les algorithmes ont, par exemple, tendance à trouver des « super-communautés » peu réalistes) et d'autre part, que les algorithmes actuellement les plus reconnus dans le domaine ne convergent pas vers une solution unique, contrairement à ce qu'ils font sur les graphes générés.*

ABSTRACT.

MOTS-CLÉS : *Détection de communautés, grands graphes de terrains, évaluation*

KEYWORDS: *Community detection, complex networks, evaluation*

1. Introduction

On sait depuis une dizaine d'années que la quasi totalité des grands graphes construits à partir de données réelles partagent des propriétés non triviales. Ces graphes ne sont ni complètement réguliers ni complètement aléatoires : le chemin le plus court entre deux sommets est généralement très court (propriété remarquable des graphes aléatoires) alors que le taux de clustering, ou de transitivité, est fort (indiquant une structure régulière) [WAT 98]. De plus ces réseaux présentent une forte disparité dans la répartition des degrés des sommets : peu de sommets sont extrêmement connectés alors qu'une grande majorité n'a que très peu de voisins [BAR 99].

Ces découvertes ont données lieu à de nombreux travaux. En particulier, le problème de *détection de communautés* intéresse un nombre toujours plus important de chercheurs : **Comment détecter, dans un graphe donné, les structures mésoscopiques porteuses de sens ?** Ces structures (des sous-ensembles de sommets) correspondent, par exemple, à des « concepts » dans un graphe lexical, à des « thématiques » dans un graphe de documents ou encore à des « communautés » [1] dans un réseau social. Notons que le terme *graph clustering* plutôt que *community detection* est parfois employé dans la littérature. Ce problème peut être vu comme un jeune héritier des problèmes de *clustering* en data-mining et du *problème de partitionnement* de graphe en informatique. Une des différences par rapport au clustering classique ou au partitionnement de graphe est que le nombre et la taille des clusters ne sont pas connus à l'avance. Aussi la structure particulière des grands graphes de terrain se doit d'être prise en compte pour développer et évaluer une méthode de détection de communautés.

Malgré le nombre important de travaux qui essayent de répondent à ce problème, il faut bien remarquer que la définition de « communauté » reste floue, pour reprendre [FOR 10] : *"The first problem in graph clustering is to look for a quantitative definition of community. No definition is universally accepted"*. Cependant, dans tous les domaines précédemment évoqués, les communautés recherchées semblent être relativement similaires, des ensembles de noeuds fortement liés entre eux et plus faiblement liés avec le reste du réseau. C'est pourquoi il existe de nombreux algorithmes dédiés à la détection de communautés de manière « universelle » . Ce sont ces derniers que nous évaluons dans cet article, afin de jauger leur pertinence et leur fiabilité sur des graphes réels.

Nous démarrons (section 2) ce papier par une rapide description des méthodes d'évaluation habituellement utilisées dans la littérature. Nous décrivons ensuite les algorithmes testés dans la suite du papier (section 3). Enfin les trois sections suivantes présentent les résultats de notre étude : sur des graphes artificiels (section 4), sur des graphes réels (section 5) et enfin face à des graphes réels artificiellement bruités (section 6). La section 7 conclut.

1. Notons que le terme « détection de communautés » provient de cette application privilégiée aux réseaux sociaux, néanmoins ce problème ne se limite pas à ces réseaux particuliers.

2. L'évaluation des algorithmes de détection de communautés

Étant donné le nombre toujours plus conséquent d'algorithmes de détection de communautés, des méthodes d'évaluation systématique sont nécessaires afin de comparer objectivement les méthodes entre elles. Les premiers travaux réalisés dans ce but consistaient à définir des graphes réguliers très simples dont la structure en communauté était imposée (chaque sommet est assigné à une communauté, puis un paramètre contrôle le ratio de liens internes aux communautés). Les algorithmes étaient alors évalués en comparant leurs résultats par rapport au partitionnement imposé à la construction, et ce, en fonction du paramètre définissant la « netteté » des communautés [GIR 02]. Le problème est que ces graphes sont peu réalistes, en particulier tous les sommets ont plus ou moins le même nombre de voisins. Des générateurs de graphes de tests plus complexes ont été introduits pour répondre à ce défaut [LAN 08, LAN 09a]. Ces générateurs produisent des graphes plus réalistes : loi de puissance dans la distribution des degrés, communautés de tailles différentes... L'idée reste cependant la même, utiliser une métrique pour quantifier l'écart entre le partitionnement idéal connu et le partitionnement trouvé par les différents algorithmes [LAN 09b].

Cependant, ces générateurs de graphes ne visent pas à modéliser les phénomènes dont résultent les graphes réels, ils cherchent simplement à imiter certaines de leurs propriétés tout en imposant une structure en communauté. Dès lors il est raisonnable de penser que ces générateurs ne produisent pas des graphes équivalents aux graphes réels. Cela pourrait nous amener à surestimer la fiabilité des algorithmes de détection de communautés actuels, la structure en communautés des graphes réels étant certainement plus complexe que celle des graphes générés. Nous comparons donc, dans cet article, les résultats trouvés par les algorithmes sur des graphes réels. Le problème est qu'il n'existe pas de graphes réels de grande taille pour lequel un partitionnement de référence est connu, nous comparons donc les résultats des algorithmes entre eux.

Comparaison de deux partitionnements. La métrique la plus courante dans le littérature pour comparer deux partitionnements est l'*Information Mutuelle Normalisée* (IMN). Cette mesure ne permet pas de comparer des partitionnements avec recouvrements (c.a.d. partitionnements où des sommets peuvent appartenir à plusieurs communautés) or l'un des algorithmes que nous testons détecte des communautés imbriquées. Nous utilisons donc une variante de cette métrique [LAN 09c] permettant de comparer des partitionnements avec recouvrements. Nous notons $IMN(A, B)$ la valeur de cette -pseudo- information mutuelle entre les partitionnements A et B. $IMN(A, B) = 0$ signifie que A et B sont indépendants, ils n'ont aucune information commune. $IMN(A, B) = 1$ signifie que les deux partitionnements sont identiques. Pour plus de détails nous renvoyons à [LAN 09c].

3. Les Algorithmes étudiés

Le nombre d'algorithmes comparés est une limite indéniable de notre étude, néanmoins il est impossible d'évaluer toutes les méthodes publiées. Nous avons choisi des méthodes récentes et reconnues comme efficaces. Afin de les tester sur de grands graphes, il était nécessaire que ces algorithmes soient performants en terme de temps

de calcul et de consommation mémoire (par exemple, l'algorithme historique de Girvan et Newman [GIR 02], ayant une complexité en $O(n^3)$, ne peut-être utilisé sur des graphes pouvant contenir des milliers de sommets). Enfin, nous avons choisi des algorithmes représentants différentes approches. Deux d'entre eux sont basés sur des marches aléatoires (WalkTrap et InfoMap), deux sur une optimisation de la modularité (Louvain et FastGreedy) et un dernier sur une recherche de motifs locaux (CFinder). L'objectif étant de faire éventuellement apparaître des similitudes ou différences entre ces approches. Pour un état de l'art plus complet des méthodes existante nous renvoyons à [SCH 07, FOR 10].

CFinder [PAL 05] : Cette méthode est basée sur une recherche de motifs locaux. Une communauté est définie comme une chaîne de k-cliques adjacentes. Une k-clique est un sous-ensemble de k sommets tous adjacents les uns aux autres, et deux k-cliques sont adjacentes si elles partagent $k-1$ sommets. L'avantage immédiat d'une telle approche est la détection de communautés avec recouvrement, un sommet pouvant appartenir à plusieurs k-cliques non forcément adjacentes. C'est la seule méthode évaluée ici qui permette un recouvrement des communautés. Une limite de cet algorithme est qu'il nécessite un paramétrage : la valeur de k (la taille des communautés à considérer). Nous avons choisi une valeur de k = 4, généralement acceptée comme la plus efficace. Les tests que nous avons menés semblent montrer que modifier cette valeur influe fortement sur les résultats trouvés mais peu sur la tendance générale de ces résultats comparés aux autres algorithmes.

Walktrap [PON 07] : Utilise une distance entre sommets basée sur des marches aléatoires [GAU 04]. Une marche aléatoire courte, partant d'un sommet donné, tend à rester dans la (les) communauté(s) de ce sommet. Ainsi la distance entre les résultats de deux marches aléatoires partant de deux sommets distincts, révèle efficacement l'appartenance commune ou non de ces sommets à une même communauté. Cette distance permet à cette méthode de partitionner le graphe par l'intermédiaire d'un algorithme de clustering hiérarchique.

Louvain [BLO 08] : Méthode d'optimisation de la modularité. La modularité est une fonction de mesure de la qualité d'un partitionnement, introduite initialement par Girvan et Newman, pour choisir une coupe privilégiée dans un dendrogramme issu d'un clustering hiérarchique [GIR 02]. La modularité mesure le nombre d'arêtes à l'intérieur des communautés moins ce même nombre obtenu sur un graphe aléatoire (c.a.d. sans structure) de même taille mais gardant exactement la même distribution de degrés. L'idée est que dans un graphe aléatoire, certains liens ont naturellement une forte chance d'exister (notamment entre 2 noeuds de très fort degrés). L'existence de ce lien dans le graphe réel ne sera donc pas un argument pertinent pour considérer que ces 2 noeuds sont effectivement dans la même communauté. Au contraire, l'existence d'un lien entre 2 noeuds ayant peu de chances d'être liés dans le graphe aléatoire sera un argument fort pour les regrouper. Louvain utilise une méthode d'optimisation gloutonne de la modularité. Au départ chaque sommet est dans sa propre communauté, puis chaque sommet prend la communauté d'un de ces voisins de tel sorte que le gain de modularité soit maximal. Cette opération est répété plusieurs fois sur l'ensemble

des sommets jusqu'à ce qu'aucun gain ne soit possible. Toute l'opération est ensuite répété sur le graphe des communautés.

Infomap [ROS 08] : Comme WalkTrap cette méthode exploite le fait qu'un marcheur suivant aléatoirement les arêtes du graphe a tendance à rester bloqué dans les communautés. Si on décrit un parcours aléatoire sur le graphe comme une séquence de numéros indiquant soit le code de la communauté dans laquelle entre le marcheur, soit le code du sommet sur lequel il arrive (code propre à la communauté courante), alors un bon partitionnement doit permettre de compresser au mieux cette séquence, les changements de communautés devant être rares. Le nombre de bits utiles, en moyenne, pour coder un pas du marcheur est défini théoriquement (pour un partitionnement donné). Cela permet de mesurer la qualité d'un partitionnement. La détection de communautés en elle même est ensuite réalisée par une méthode d'optimisation similaire à Louvain.

FastGreedy [CLA 04] : Comme Louvain cette méthode est basée sur une optimisation gloutonne de la modularité. En partant du partitionnement le plus fin (une communauté par noeud), les communautés sont rassemblées progressivement si cela permet d'augmenter la modularité [2]. Un des point fort de cette méthode est sa rapidité d'exécution.

4. Comparaison sur graphes générés

Tester un algorithme de détection de communautés sur des graphes dont le partitionnement correct est connu est la première étape permettant de les valider. Récemment des générateurs de graphes (*LFR benchmark*) ont été proposés [LAN 08, LAN 09a]. Ils produisent des graphes ayant des propriétés proches des graphes réels (distribution des degrés et taille des communautés en loi de puissance, par exemple), mais dont le partitionnement en communautés est connu (le graphe étant construit de manière à respecter ces communautés). Bien que ces graphes ne soient pas équivalents aux graphes réels (nous illustrons cela par la suite), il est intéressant de mesurer l'efficacité des algorithmes sur ces graphes, avant de les tester sur des graphes réels.

Nous avons réalisé un certain nombre d'expériences, en faisant varier les paramètres suivants : n le nombre de noeuds du graphe, $\langle k \rangle$ le degré moyen des noeuds, c la taille des communautés (plage de tailles possibles), μ le *mixing parameter* i.e. pour chaque sommet le ratio entre le nombre de liens à l'extérieur et le nombre à l'intérieur de sa communauté. Nous avons décidé d'un graphe de référence, puis nous avons fait varier ses paramètres un à un, et comparé à nouveau les résultats des algorithmes. Par manque de place nous ne donnons ici que les analyses de ces résultats [3].

---

2. Notons que, en somme, FastGreedy est une méthode de clustering hiérarchique agglomératif, ce qui n'est pas le cas de Louvain (qui est aussi une méthode d'optimisation gloutonne). Deux sommets rassemblés lors d'une étape de FastGreedy appartiendrons toujours à la même communauté, ce n'est pas forcément le cas avec Louvain.

3. Les tableaux de chiffres et figures donnant les résultats peuvent être trouvé à l'adresse : `http://www.irit.fr/~Emmanuel.Navarro/MARAMI_comdet/`

Graphe de référence. $n = 5000$, $\langle k \rangle = 15$, $c = [10, 50]$, $\mu = 0,3$. On observe que InfoMap et WalkTrap trouvent la solution exacte. Louvain trouve une solution presque identique ($IMN = 0.99$). CFinder est assez proche également (0.93), l'erreur venant principalement de la fausse détection de petites communautés. FastGreedy est très éloigné des autres résultats, il n'a trouvé que peu de communautés, peu pertinentes, dont notamment des « super-communautés » , allant jusqu'à 500 noeuds.

Petit graphe. $n = 1000$. Les mêmes résultats semblent apparaître, à ceci près que FastGreedy a des résultats plus proches des autres ($IMN \simeq 0.5$), mais a toujours une tendance à produire des super-communautés.

Grandes communautés. $c = [30, 100]$, Louvain, WalkTrap et InfoMap trouvent toujours des résultats extrêmement proches de la solution exacte. Les résultats de FastGreedy sont aussi faibles que pour le graphe de référence, toujours avec le même défaut. CFinder au contraire, semble s'effondrer, l'écart à la solution exacte étant très important ($IMN = 0.4$). On observe que CFinder a tendance à sur-diviser les communautés, problème inverse à celui de FastGreedy.

Communautés mal définies. $\mu = 0.5$ On a ici affaire à ce qui est considéré comme un problème « compliqué » , chaque noeud ayant autant de liens à l'intérieur de sa communauté qu'à l'extérieur. Avec un IMN de 0.99 avec la solution exacte, InfoMap est de loin le plus proche. Louvain et WalkTrap ne sont pas très éloignés, avec 0.90 environ, cependant l'IMN entre Louvain et WalkTrap est plus faible, 0.80 environ. On peut remarquer que Louvain a tendance à sur-diviser alors que WalkTrap, au contraire, trouve de trop grandes communautés. FastGreedy est toujours très éloigné et CFinder a un IMN de 0.50 avec la solution exacte.

Graphe très denses $\langle k \rangle = 25$. À part FastGreedy, tous les algorithmes trouvent une solution très proche de la solution exacte. FastGreedy est toujours très éloigné ($IMN = 0.10$). Notons que le temps de calcul de CFinder devient rapidement très important quand la densité moyenne augmente.

Graphe peu dense. $\langle k \rangle = 7$. InfoMap et WalkTrap trouvent un résultat assez proche, avec une IMN de 0.85. En revanche, Louvain et CFinder ont tous les deux tendance à sur-diviser dans ce cas, obtenant des IMN faibles avec la solution exacte (respectivement 0.45 et 0.20). Leurs découpages ne sont cependant pas les mêmes, l'IMN entre eux étant de 0.20.

Ces quelques tests effectués sur des graphes générés permettent de se rendre compte que selon les propriétés du graphe, certains algorithmes fonctionnent mieux que d'autres. CFinder par exemple semble être peu efficace quand il s'agit de trouver des grosses communautés, Louvain est moins à l'aise sur les graphes peu denses que sur ceux très denses. On note aussi que certains algorithmes semblent être plus éloignés que d'autres de manière générale. (FastGreedy et InfoMap ont toujours des résultats très différents par exemple. Ici, on peut qualifier les résultats d'InfoMap de « meilleurs » en comparaison à la solution exacte). Il est donc intéressant maintenant de comparer ces résultats avec ceux que l'on peut obtenir sur des graphes réels.

5. Comparaison sur graphes réels

Nous avons observé le comportement des algorithmes sur les trois graphes réels suivants. **DicoSyn** un réseau de synonymie des verbes français, **Wiktionary** un réseau de synonymie des verbes anglais (issue du site Wiktionary [4] [NAV 09]), et **CA HepTH** un réseau de co-citations (les sommets sont des auteurs, et deux auteurs sont liés si ils ont été co-auteurs d'un article [LES 07]). Le tableau 1 présente les principales caractéristiques de ces graphes.

Tableau 1. *Pédigrées des graphes utilisés.* *n : nombre de sommets, m : nombre d'arêtes, $\langle k \rangle$ degré moyen, n_{lcc} : nombre de sommets de la plus grande partie connexe, C : coefficient de clustering, L_{lcc} : distance moyenne entre deux sommets sur la plus grande partie connexe, λ : coefficient de la loi de puissance approximant la distribution des degrés avec un coefficient de corrélation r^2.*

	n	m	$\langle k \rangle$	n_{lcc}	C	L_{lcc}	λ	r^2
Wiktionary	7339	8293	1.13	4285	0.106	8.98	-2.4	0.94
DicoSyn	9147	51423	5.62	8993	0.142	4.20	-1.9	0.91
CA-HepTh	9875	25973	2.63	8638	0.284	5.95	-2.3	0.91

Tableau 2. *Comparaison des résultats de cinq algorithmes sur trois graphes réels. Chaque case donne l'IMN entre deux partitionnements. Le nom des algorithmes n'est pas indiqué sur les colonnes, la $i^{ème}$ colonne (pour chaque graphe) correspondant à l'algorithme présenté sur la $i^{ème}$ ligne.*

	DicoSyn (Verbe, fr)					Wiktionary (Verbe, en)					CA HepTH				
Louvain	.	0.31	0.52	0.50	0.08	.	0.78	0.78	0.70	0.23	.	0.60	0.69	0.56	0.26
InfoMap	0.31	.	0.28	0.44	0.03	0.78	.	0.85	0.80	0.20	0.60	.	0.64	0.62	0.18
WalkTrap	0.52	0.28	.	0.48	0.04	0.78	0.85	.	0.83	0.16	0.69	0.64	.	0.65	0.17
FastGreedy	0.50	0.44	0.48	.	0.02	0.70	0.80	0.83	.	0.09	0.56	0.62	0.65	.	0.09
CFinder	0.08	0.03	0.04	0.02	.	0.23	0.20	0.16	0.09	.	0.26	0.18	0.17	0.09	.

Comparaison des IMN entre algorithmes. Le tableau 2 donne les résultats des comparaisons entre algorithmes sur ces trois graphes réels. Il apparait que les différents algorithmes trouvent des résultats relativement différents, ils ne « s'entendent » visiblement pas sur un découpage identique. Notons que ces résultats sont très hétérogènes, par exemple sur Wiktionary, l'IMN moyenne si l'on exclu CFinder est de 0.79 alors qu'elle est de 0.42 sur DicoSyn. Ces écarts moins importants sur Wiktionary peuvent s'expliquer par le fait que ce graphe est très peu dense et qu'il est composé d'un nombre important de petites composantes connexes (2, 3 sommets) formant des communautés assez faciles à détecter. La seconde remarque que l'on peut formuler est l'importante différence entre CFinder et les autres algorithmes. Dans le cas des graphes générés (Section 4) CFinder était généralement comparable aux autres, sauf quand les communautés étaient très grandes [5]. À l'inverse FastGreedy donnait des résultats différents des autres algorithmes sur les graphes générés, alors qu'ici ses résultats semblent comparables à ceux d'InfoMap, Louvain et WalkTrap. Enfin les

4. Wiktionary (`www.wiktionary.org`) est un dictionnaire collaboratif libre et gratuit.

5. CFinder permet un recouvrement des communautés et il ne classifie dans aucune communauté les sommets n'appartenant à aucune 4-clique, cela explique certainement l'écart important observé ici. Notons que cet écart ne signifie pas que les résultats de CFinder sont moins bon que les autres. Ils diffèrent, mais rien ne nous permet de juger un résultat meilleur qu'un autre.

méthodes basées sur des marches aléatoires et celles basées sur la modularité ne se différencient pas dans cette étude.

Tableau 3. *Nombre de communautés (N) et taille relative de la plus grande ($\%_{max}$)*

	DicoSyn		Wiktionary		CA-HepTh	
	N	$\%_{max}$	N	$\%_{max}$	N	$\%_{max}$
Louvain	812	7.7%	2130	0.5%	2052	4.9%
InfoMap	794	1.1%	1609	0.6%	1517	0.8%
WalkTrap	910	18.5%	1490	9.1%	1295	16.9%
FastGreedy	186	26.5%	1075	5.0%	551	15.9%
CFinder	317	46.4%	70	0.2%	1177	3.7%

Tailles des communautés détectées. Le tableau 3 donne pour chaque méthode et chaque graphe le nombre de communautés (N), et la taille relative (c.a.d. par rapport au nombre de sommets du graphe) de la plus grande des communautés ($\%_{max}$). Là encore, les écarts entre les méthodes sont importants. Aussi ils varient en fonction des graphes, en particulier ils sont moins importants sur Wiktionary. Mis à part sur Wiktionary, en ne considérant que les algorithmes considérés par l'IMN comme les plus semblables (c.a.d. en excluant CFinder), on constate que le nombre de communautés peut varier d'un facteur 1 à 4 (FastGreedy et Louvain), et la taille de la plus grosse communauté d'un facteur 1 à 10. Pour avoir une IMN relativement proche malgré ces écarts importants, c'est probablement que les algorithmes trouvent de nombreux résultats relativement similaires (des communautés "faciles" à séparer), et sont en fort désaccord sur d'autres communautés, beaucoup plus difficiles à distinguer.

La figure 1 donne la distribution de la taille des communautés. Ces distributions semblent suivre une loi de puissance jusqu'à une certaine taille critique (les méthodes détectent des super-communautés qui « cassent » la loi de puissance). Seuls InfoMap et CFinder (pour certain graphes) échappent à ce phénomène, et retournent une « belle » loi de puissance. CFinder trouve plus de petites communautés (mais aucune très petite non plus, à cause de la limite imposée par la valeur de k). Cependant, une fois encore, il semble difficile de tirer des conclusions générales valables pour tous les graphes.

Le phénomène intéressant qui ressort donc de cette analyse est celui des « super-communautés » . Certains algorithmes, sur certains graphes, trouvent en effet des communautés comprenant 15, 20, voire 45% des sommets. Nous pouvons dire que ces super-communautés sont le signe d'un échec de l'algorithme à partitionner correctement le graphe. D'une part ces résultats sont peu exploitables, et d'autre part (pour les graphes étudiés) ces communautés ne semblent pas être porteuses de sens. Par exemple, une analyse rapide des résultats d'InfoMap sur DicoSyn montre clairement des partitions « sensées » (verbes proches sémantiquement), alors que la communauté de plus de 2000 verbes proposée par FastGreedy n'est clairement pas compréhensible.

Rôle des petites communautés. Un certain nombre d'indices permettent de penser que la détection est plus simple pour les petites communautés que pour les grandes. Nous avons comparé l'IMN entre algorithmes (comme dans le tableau 2) mais en gardant seulement les grandes communautés ou seulement les petites. Le tableau 4

Tableau 4. *Effet des petites communautés sur l'accord entre les méthodes (sur DicoSyn). A gauche, en ne gardant que les « grosses » communautés ; à droite, que les « petites » .*

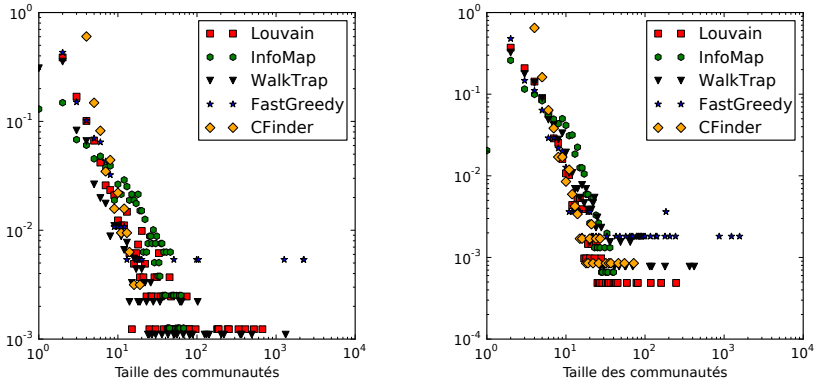
	$\|c\| > 6$					$\|c\| \leq 6$				
Louvain	.	0.19	0.27	0.09	0.11	.	0.38	0.57	0.56	0.05
InfoMap	0.19	.	0.20	0.22	0.06	0.38	.	0.34	0.49	0.02
WalkTrap	0.27	0.20	.	0.14	0.06	0.57	0.34	.	0.52	0.03
FastGreedy	0.09	0.22	0.14	.	0.00	0.56	0.49	0.52	.	0.01
CFinder	0.11	0.06	0.06	0.00	.	0.05	0.02	0.03	0.01	.

donne les résultats pour DicoSyn [6]. On constate effectivement que si l'on supprime les petites communautés, les valeurs d'IMN s'effondrent. (Particulièrement flagrant sur FastGreedy, qui semble ne devoir sa ressemblance aux autres algorithmes qu'aux petites communautés) Au contraire, si l'on ne garde que les petites communautés, les résultats restent stable voire s'améliorent.

« Netteté » des communautés trouvées. La dernière expérimentation que nous avons menée consiste à étudier la « Netteté » des communautés trouvées c.a.d. leur proportion de liens internes par rapport au nombre total de liens de leurs sommets. Par exemple, dans le cas d'une clique de 3 sommets ayant chacun un lien vers l'extérieur, le ratio serait de $3/(3+3) = 0,5$. La figure 2 donne les nuages de points montrant l'éventuelle corrélation entre ce ratio et la taille des communautés, ainsi que la distribution de ce ratio et de la taille des communautés. On voit que les profils sont très différents entre graphes réels et graphes générés. Les communautés des graphes générés ont toutes le même ratio (imposé par construction) alors que cela ne semble pas être le cas sur les graphes réels. Un nombre important de communautés sur les graphes réels ont un ratio de 1, ce sont des composantes connexes généralement de faible taille. Cependant les profils sont très différents entre les graphes. Sur CA-HepTH et Wiktionary le ratio de liens entrant est très variable, peu importe la taille des communautés. Mais le plus surprenant est sur DicoSyn où les ratios sont toujours extrêmement faibles, sauf les petites communautés qui elles ont un ratio très variable. Sur le même graphe, on peut constater que FastGreedy, par exemple, propose un résultat très différent : la plupart des communautés ont un ratio largement plus élevé que pour InfoMap (donc moins de liens inter-communautés), en contrepartie elles sont beaucoup plus étendues et semblent moins pertinentes (super-communautés). Il ressort de manière certaine que les communautés des graphes réels sont définies moins nettement que celles produites par les générateurs.

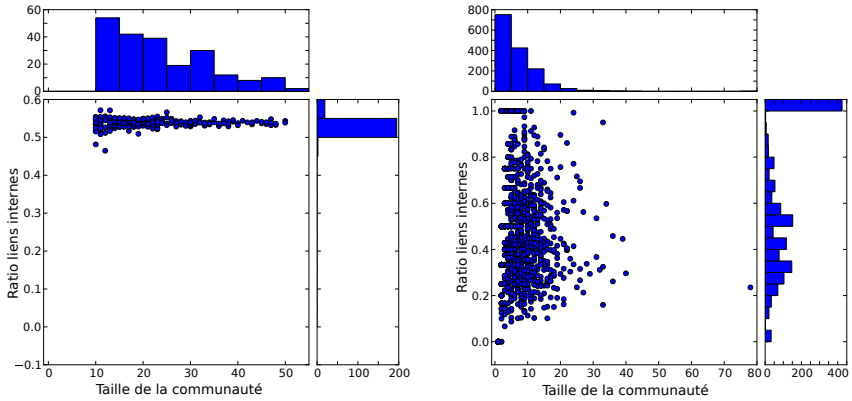
Conclusion. Les tests réalisés dans cette section montrent clairement que la détection de communautés sur les graphes réels est encore un problème complexe pour lequel il n'existe pas de solution satisfaisante. Tout d'abord, nous avons constaté que les algorithmes trouvent des résultats très différents (en particulier pour les grosses communautés). Aussi, nous avons montré que ces résultats varient fortement d'un graphe à l'autre. Cela nous conduit à penser que pour étudier les graphes réels, il

6. Les résultats sont similaires sur les autres graphes

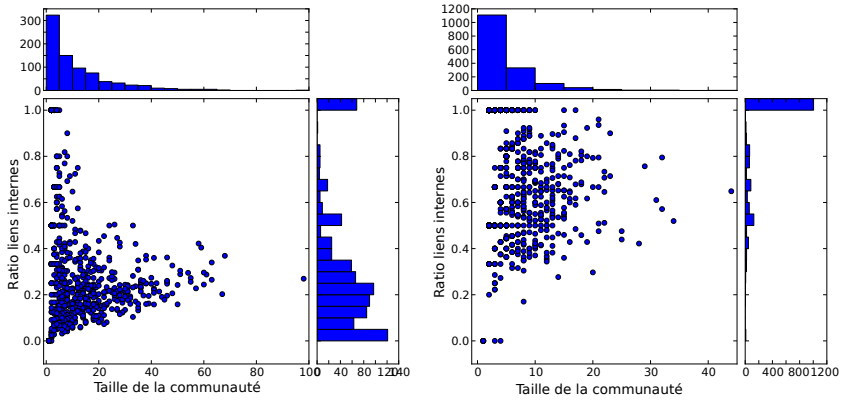


(a) DicoSyn (b) CA-HepTH

Figure 1. *Distribution de la taille des communautés.* Echelle logarithmique.



(a) Graphe généré par LFR (b) CA-HepTH



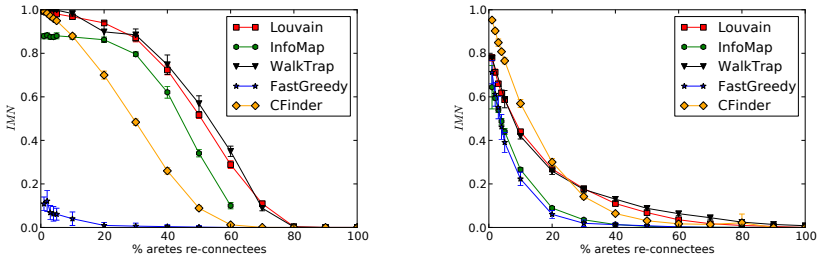
(c) DicoSyn (d) Wiktionary

Figure 2. *Taille et « Netteté » des communautés trouvées par InfoMap.*

pourrait d'une part être intéressant de recouper les résultats des différents algorithmes (par exemple en regroupant des sommets s'ils se trouvent dans une même communauté avec trois algorithmes différents), et d'autre part de proposer des solutions moins strictes et plus robustes, par exemple en donnant une appartenance graduelle des sommets aux communautés et permettant aux sommets d'appartenir à plusieurs communautés.

6. Robustesse des méthodes

Dans cette section nous tentons d'observer le comportement des algorithmes lorsque l'on bruite les graphes : comment varient les résultats lorsqu'un certain pourcentage des arêtes est reconnecté aléatoirement ? La figure 3 présente l'IMN entre le partitionnement obtenu sur le graphe initial et celui obtenu quand une certaine proportion des arêtes est re-cablée aléatoirement, et ce pour le graphe généré par LFR (graphe de référence utilisé en Section 4) et pour DicoSyn.



(a) Graphe généré par LFR (b) DicoSyn

Figure 3. *Qualité du partitionnement retourné sur un graphe de plus en plus bruité (par rapport au partitionnement de départ).* *Pour chaque proportion d'arêtes reconnectées* 20 *graphes ont été calculés, chaque point donne la moyenne de l'*IMN*, l'écart-type est représenté par une barre verticale.*

Là encore, le comportement des algorithmes est très clairement différent sur les graphes générés et sur les graphes réels. Alors que les résultats sont peu sensibles à un faible bruit sur les graphes générés (sauf pour FastGreedy qui manifestement est inefficace au moins sur ces graphes, voir section 4), ils le sont extrêmement sur les graphes réels. Ce résultat est à mettre en parallèle à ceux donnés par la figure 2 : beaucoup des communautés des graphes réels sont bien moins nettes que celles des graphes générés. Notons que cela est moins vrai pour CFinder qui est le moins « robuste » sur les graphes générés et paradoxalement le meilleur sur les graphes réels. Ce comportement est certainement lié à sa capacité à détecter le recouvrement entre communautés (recouvrement absent des graphes générés).

7. Conclusion

Nous avons évalué cinq méthodes reconnues de détection de communautés sur des graphes réels, en comparant leurs résultats respectifs et en analysant leurs comportements sur des graphes bruités. Bien que cette étude présente plusieurs limites, en particulier concernant le nombre de graphes utilisés et d'algorithmes évalués, il en ressort que le problème de détection de communautés est plus complexe sur des graphes réels que sur les graphes artificiels habituellement utilisés pour l'évaluation.

L'accord entre les méthodes est faible sur les graphes réels alors qu'il est généralement important sur les graphes d'évaluation. Aussi les résultats sont bien moins robustes sur les graphes réels que sur les graphes d'évaluation. Et enfin les algorithmes on tendance à trouver, sur les graphes réels, des « super-communautés » peu réalistes. Cela nous permet de conclure que les méthodes testées ne permettent pas d'obtenir des communautés réellement fiables sur des graphes réels. Remarquons que quatre des méthodes étudiées ne détectent pas de communautés avec recouvrement (comme la plupart des méthodes de la littérature). Or il est probable que les communautés des graphes réels présentent un fort taux de recouvrement (par exemple, dans un graphe de synonymie, la plupart des mots sont polysémiques). Ce point semble primordial pour rendre les résultats plus pertinents. Il serrait donc intéressant de prolonger cette étude en évaluant d'autres méthodes que CFinder acceptant un recouvrement des communautés, par exemple [ALV 06, CAZ 10].

8. Bibliographie

[ALV 06] ALVAREZ-HAMELIN J. I., DALL'ASTA L., BARRAT A., VESPIGNANI A., « k-core decomposition : a tool for the visualization of large scale networks », *Advances in Neural Information Processing Systems*, vol. 18, 2006, p. 41–50.

[BAR 99] BARABÁSI A., ALBERT R., « Emergence of Scaling in Random Networks », *Science*, vol. 286, n° 5439, 1999, p. 509–512.

[BLO 08] BLONDEL V. D., GUILLAUME J., LAMBIOTTE R., LEFEBVRE E., « Fast unfolding of communities in large networks », *Journal of Statistical Mechanics : Theory and Experiment*, vol. 10, 2008.

[CAZ 10] CAZABET R., AMBLARD F., HANACHI C., « Using network's evolution to detect strongly overlapping communities », *MARAMI 2010*, octobre 2010.

[CLA 04] CLAUSET A., NEWMAN M. E. J., MOORE C., « Finding community structure in very large networks », *Phys. Rev. E*, vol. 70, n° 6, 2004.

[FOR 10] FORTUNATO S., « Community detection in graphs », *Physics Reports*, vol. 486, n° 3-5, 2010.

[GAU 04] GAUME B., « Balades aléatoires dans les petits mondes lexicaux », *I3 Information Interaction Intelligence*, vol. 4, n° 2, 2004.

[GIR 02] GIRVAN M., NEWMAN M. E. J., « Community structure in social and biological networks », *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, n° 12, 2002, p. 7821–7826.

[LAN 08] LANCICHINETTI A., FORTUNATO S., RADICCHI F., « Benchmark graphs for testing community detection algorithms », *Phys. Rev. E*, vol. 78, n° 4, 2008, page 046110.

[LAN 09a] LANCICHINETTI A., FORTUNATO S., « Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities », *Phys. Rev. E*, vol. 80, n° 1, 2009, page 016118.

[LAN 09b] LANCICHINETTI A., FORTUNATO S., « Community detection algorithms : A comparative analysis », *Phys. Rev. E*, vol. 80, n° 5, 2009, page 056117.

[LAN 09c] LANCICHINETTI A., FORTUNATO S., KERTÉSZ J., « Detecting the overlapping and hierarchical community structure in complex networks », *New Journal of Physics*, vol. 11, n° 3, 2009, page 033015.

[LES 07] LESKOVEC J., KLEINBERG J., FALOUTSOS C., « Graph evolution : Densification and shrinking diameters », *ACM Trans. Knowl. Discov. Data*, vol. 1, n° 1, 2007, page 2, ACM.

[NAV 09] NAVARRO E., SAJOUS F., GAUME B., PRÉVOT L., HSIEH S., KUO I., MAGISTRY P., HUANG C.-R., « Wiktionary and NLP : Improving synonymy networks », *Proceedings of the 2009 ACL-IJCNLP (workshop)*, Singapore, August 2009, p. 19–27.

[PAL 05] PALLA G., DERENYI I., FARKAS I., VICSEK T., « Uncovering the overlapping community structure of complex networks in nature and society », *Nature*, vol. 435, n° 7043, 2005, p. 814–818.

[PON 07] PONS P., « Détection de communautés dans les grands graphes de terrain », PhD thesis, 2007.

[ROS 08] ROSVALL M., BERGSTROM C. T., « Maps of random walks on complex networks reveal community structure », *Proceedings of the National Academy of Sciences*, vol. 105, n° 4, 2008, p. 1118–1123.

[SCH 07] SCHAEFFER S. E., « Graph clustering », *Computer Science Review*, vol. 1, n° 1, 2007, p. 27–64.

[WAT 98] WATTS D. J., STROGATZ S. H., « Collective Dynamics of Small-World Networks », *Nature*, vol. 393, 1998, p. 440–442.