Identification multi-échelle de la structure communautaire de très grands graphes

Vincent Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre

Identification multi-échelle de la structure communautaire de très grands graphes

Vincent Blondel[1†], Jean-Loup Guillaume[1,2‡], Renaud Lambiotte[1] et Etienne Lefebvre[1]

[1]*Dépt. d'Ingénierie Mathématique, Université catholique de Louvain, 4 avenue Georges Lemaitre,B-1348 Louvain-la-Neuve, Belgique.*
[2]*Université Pierre et Marie Curie, LIP6 (UMR 7606 CNRS), 104 avenue Kennedy, 75016 Paris, France.*

L'identification de sous-groupes denses dans les grands réseaux d'interactions est un problème complexe auquel on se retrouve confronté dès lors que l'on essaye de décrire précisément la structure d'un grand graphe. Le calcul de ces groupes, ou communautés, revient à chercher une partition de l'ensemble des sommets qui maximise une fonction de qualité telle que la modularité. Malheureusement, la maximisation de cette fonction est un problème NP-complet et nécessite donc l'utilisation de méthodes heuristiques pour pouvoir obtenir de bonnes partitions. Nous proposons ici une méthode très efficace pour résoudre ce problème sur de très grands graphes (centaines de millions de sommets) et nous en illustrons les résultats en la comparant à plusieurs approches classiques.

Keywords: réseaux d'interaction, communautés, algorithmique

1 Introduction

Le partitionnement de graphes est un problème classique en théorie des graphes qui consiste à regrouper les sommets d'un graphe par similarité, cette dernière étant généralement définie à partir de propriétés topologiques des sommets. L'étude des grands réseaux d'interactions a complexifié ce problème en refusant d'imposer *a priori* un nombre ou des tailles de groupes. La recherche dans ce domaine s'est alors orientée vers l'identification de groupes de sommets pour lesquels il existe un grand nombre de liens à l'intérieur des groupes et très peu entre les groupes. À l'heure actuelle, de nombreuses méthodes permettent de calculer des communautés sur des graphes avec plusieurs centaines de milliers de sommets en un temps raisonnable (de l'ordre de l'heure), par exemple [NG04, PL06]. Mais une seule méthode, proposée par Wakita et al. [WT07], permet de réellement dépasser la barre symbolique du million de sommets et les auteurs estiment que la limite de leur algorithme se situe à 10 millions de sommets.

Afin de comparer les résultats de ces méthodes, des travaux récents [New06] ont introduit une mesure de la qualité d'une partition, la modularité, qui indique à quel point une partition rend compte de la structure communautaire, c'est-à-dire identifie effectivement des groupes avec les contraintes de densité. Pour un graphe $G = (V,E)$ et une partition $P = \{C_1, \cdots, C_k\}$ des sommets de V, la modularité est formellement définie par $Q(P) = \sum_{C \in P} e_C - a_C^2$ où e_c représente la proportion de liens à l'intérieur de C et a_c la proportion de liens connectés à C, par rapport au nombre au nombre total de liens. Cette valeur est toujours inférieure à 1 et permet de comparer les qualités de deux partitions pour un même graphe et, en conséquence, permet de comparer les performances d'algorithmes de détection de communautés. Si cette mesure de qualité n'est pas la seule existante et a plusieurs lacunes [FB07], elle reste néanmoins la plus fiable et la plus communément utilisée. L'inconvénient majeur de cette fonction est que son optimisation est un problème NP-complet [BDG+06] et nécessite donc d'utiliser des méthodes approchées dès lors que l'on souhaite l'utiliser sur des graphes de grande taille.

†Tous les auteurs sont soutenus par l'Action de Recherche Concertée "Large Graphs and Networks" de la Direction de la recherche scientifique - Communauté française de Belgique.
‡J.-L. Guillaume est aussi soutenu par les projets MAPE et MAPAP.

Nous proposons ici une nouvelle approche qui, malgré sa simplicité apparente, permet d'extraire une structure hiérarchique en communautés pour des graphes de tailles très élevées, en des temps records et avec une qualité supérieure aux meilleurs algorithmes actuels. Plus précisément, cet algorithme permet de traiter des graphes avec plusieurs dizaines de millions de sommets en quelques minutes et n'est limité que par la quantité de mémoire vive disponible.

Nous présentons cet algorithme dans la Section 2, puis nous présentons des résultats comparatifs pour plusieurs graphes dans la Section 3, avant de conclure et de présenter quelques pistes de recherche.

2 Méthode de détection de communauté

La plupart des méthodes de détection de communautés se basent sur l'intuition qu'une structure communautaire est par nature hiérarchique, c'est-à-dire qu'une communauté est elle-même composée de sous-communautés qui sont à leur tour composées de sous-sous-communautés et ainsi de suite. Ces méthodes peuvent être classées en deux grandes familles : les approches agglomératives et les méthodes divisives. Les méthodes agglomératives (par exemple [PL06]) tentent de fusionner de manière récursive des petites communautés en de plus grandes en effectuant les choix sur la base d'une mesure de proximité entre les communautés, avec comme point de départ des communautés atomiques ne contenant qu'un sommet. Les méthodes divisives (par exemple [NG04, WT07]) vont au contraire tenter d'identifier les liens inter-communautaires et les supprimer pour isoler petit à petit les communautés.

Dans tous les cas, la sortie est un arbre représentant la structure hiérarchique et il faut noter qu'avec ces méthodes la structure hiérarchique est enrichie à chaque fusion ou séparation.

2.1 Algorithme

Notre approche se situe dans la famille des méthodes agglomératives et se compose de plusieurs étapes similaires, chacune tentant de créer un ensemble de meta-communautés en fusionnant des communautés obtenues à l'étape précédente. L'objectif de chaque étape, que l'on nomme *passe* dans la suite, est donc de calculer un graphe de meta-communautés à partir d'un graphe de communautés. Initialement, on considère un ensemble de communautés atomiques contenant chacune un seul sommet du graphe que l'on souhaite décomposer. Nous allons maintenant décrire une passe de l'algorithme, c'est-à-dire comment fusionner des communautés en meta-communautés.

Soit $G = (V, E, w)$ un graphe de communautés pondéré (w est une fonction associant un poids à chaque lien) dont les sommets sont les communautés calculées à la passe précédente. On commence par placer chaque sommet i dans un groupe distinct $c[i]$ puis, pour chaque sommet i du graphe, on calcule, pour tous les voisins j de i, le gain de modularité que l'on obtiendrait en déplaçant i du groupe $c[i]$ au groupe $c[j]$. Si l'un des gains est positif, on déplace effectivement i dans le groupe $c[j]$ qui maximise ce gain, sinon on laisse i où il se trouvait. Au fur et à mesure du déplacement des sommets, des groupes de taille plus grosse vont se former. Comme l'ordre dans lequel les sommets sont considérés a une influence sur les groupes obtenus finalement, on répète cette étape (que l'on nommera *itération*) tant qu'il est possible d'améliorer la modularité. Une fois atteint un maximum de la modularité, l'algorithme est terminé et les groupes obtenus sont en fait les meta-communautés que l'on utilisera à la passe suivante : il suffit de construire un nouveau graphe de communautés pondéré dont les sommets sont les groupes et dont les liens (y compris des boucles) sont pondérés naturellement par la somme des poids des liens entre les sommets des groupes.

L'efficacité de cette méthode tient au fait qu'il faut en pratique peu d'itérations pour atteindre un maximum local de la modularité et que chaque opération élémentaire de calcul de gain de modularité peut se faire en appliquant la formule suivante :

$$\Delta Q(C,i) = \left[\frac{\sum_{in}+k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

qui s'obtient simplement à partir de la formule générale de la modularité, et où $\sum_{in}$ représente la somme des poids des liens dans C, $\sum_{tot}$ la somme des poids des liens attachés à C, k_i la somme des poids des liens attachés à i, $k_{i,in}$ la somme des poids des liens de i à C, et m la somme des poids des liens du graphe.

D'un point de vue algorithmique, il suffit de stocker des tableaux contenant les $\sum_{in}$ et $\sum_{tot}$ et une correspondance sommet/groupe pour pouvoir calculer ce gain de manière très efficace, typiquement en un temps proportionnel au degré de i.

Il faut aussi noter qu'à une passe donnée, les itérations sont faites tant qu'il y a un gain de modularité, et que l'on effectue de nouvelles passes tant qu'il y a un gain. L'observation expérimentale du fonctionnement de l'algorithme montre que les dernières itérations d'une passe et les dernières passes n'apportent qu'un gain marginal mais sont coûteuses en temps. En arrêtant systématiquement une passe si la dernière itération n'a apporté qu'un gain marginal ou en arrêtant l'algorithme si une passe n'a pas abouti à un gain supérieur à un seuil fixé (typiquement 0.0001), on limite simplement le nombre d'itérations et de passes. Cette optimisation simple a été utilisée dans les expériences présentées plus bas.

3 Résultats expérimentaux

Dans cette section, nous présentons des résultats expérimentaux visant à comparer notre approche à trois des approches les plus classiques et parmi les plus efficaces. Il s'agit de méthodes à base de marches aléatoires, notée *P*L (Pons-Latapy [PL06]), d'optimisation de la modularité, notée *CMN* (Clauset-Moore-Newman [CNM04]), et de la méthode la plus efficace actuellement qui est une optimisation de la précédente, notée *WT* (Wakita-Tsurumi [WT07]). Malheureusement, le code de cette dernière méthode n'étant pas disponible, nous avons estimé le temps qu'aurait pris cet algorithme sur nos graphes de test en vérifiant la validité de nos estimations d'après les résultats fournis dans l'article. Les résultats en termes de modularité devraient être proches de ceux de l'approche *CMN* car elle en reprend les principes.

Le tableau 1 contient des résultats comparatifs pour les 4 méthodes sur des graphes de taille variable allant de 34 sommets à 118 millions de sommets. Tous les calculs ont été effectués avec les codes fournis par les auteurs des articles respectifs qui ont été compilés et testés sur une même machine (bi-opteron 2.2k avec 24Go de RAM). D'après les auteurs de *Wakita-Tsurumi*, leur méthode devrait pouvoir s'appliquer sur des graphes ayant jusqu'à 10 millions de sommets, mais ils ne l'ont pas testée sur des graphes ayant plus de 5.5 millions de sommets.

	Karate Club	Arxiv	Internet	Web NDU	Web Base 2001
	n=34 / m=77	n=9k / m=24k	n=70k / m=351k	n=325k / m=1M	n=118M / m=1B
CMN	0.40 / 0s	0.77 / 3.6s	0.69 / 799s	0.93 / 84mn	- / -
PL	0.42 / 0s	0.76 / 3.3s	0.73 / 575s	0.89 / 111mn	- / -
WT	- / 0s	- / 0s	- / 8s	- / 52s	- / -
Notre approche	0.42 / 0s	0.81 / 0s	0.78 / 1s	0.93 / 3s	0.98 / 152mn

TAB. 1: Résultats obtenus sur plusieurs graphes de taille croissante. La première ligne décrit les graphes (nombre de sommets, nombre de liens), les lignes suivantes donnent la modularité et le temps de calcul pour 4 approches parmi les plus efficaces actuellement. Pour l'approche de *WT* les résultats en termes de modularité ne sont pas disponibles et les temps sont estimés uniquement. Des références décrivant les graphes peuvent être trouvées dans [PL06] et [BV04]. L'algorithme nécessite environ 10Go de mémoire pour traiter le graphe Web Base 2001, principalement utilisés pour le stockage du graphe lui-même.

D'un point de vue comparatif, la méthode que nous proposons est supérieure aux approches précédentes à tous points de vue. Les gains en termes de modularité ne sont pas très élevés mais il sont tout de même positifs sur tous les graphes de grande taille là où les autres approches offrent des résultats variables. Mais le gain le plus important concerne le temps de calcul, les résultats étant presque instantanés sur des graphes ayant moins d'un million de sommets. Pour des graphes plus gros, le temps de calcul reste très faible et notre approche est la seule utilisable pour des graphes ayant plusieurs dizaines de millions de sommets.

Le temps de calcul avec notre approche est fortement concentré sur les premières passes, les deux premières prenant environ 95% du temps sur tous les exemples. Pour le graphe Web Base 2001 en particulier, la structure communautaire est si forte qu'après la première passe il n'y a presque plus d'évolution. Sur d'autres graphes de taille comparable mais moins communautaires notre approche pourrait donc prendre plus de temps tout en restant largement raisonnable.

Outre les résultats expérimentaux validant la qualité de l'algorithme, d'autres avantages doivent être soulignés :

- il est possible de décomposer des graphes pondérés ou orientés en adaptant la définition de la modularité, ce qui n'est pas le cas de toutes les méthodes. Il est aussi possible d'utiliser d'autres fonctions que la modularité tant qu'elles sont calculables localement ;
- un autre problème des approches actuelles, mis en évidence par Barthelemy et Fortunato, est lié à la modularité elle-même qui est souvent plus élevée si les communautés sont de grande taille [FB07], ce qui impose de fusionner de petites communautés très denses même si cela paraît contre-intuitif. Notre approche faisant seulement des modifications locales va généralement créer des petites communautés à une passe donnée qui seront fusionnées dans les passes ultérieures. Les sous-communautés sont donc toutes directement disponibles dans la structure de l'arbre fournie par l'algorithme si l'on souhaite observer le graphe à une autre résolution qui paraît plus intuitive à l'utilisateur ;
- enfin, la méthode n'utilise pas de connaissance globale sur le graphe, chaque opération élémentaire consistant simplement à regarder la liste d'adjacence des sommets les uns après les autres. Il serait en particulier possible de ne conserver que les tableaux $\sum_{in}$, $\sum_{tot}$ et la correspondance sommet/communauté en mémoire et de lire le graphe sur disque, ce qui permettrait de traiter des grands graphes même sur des machines peu puissantes.

4 Conclusions et perspectives

Nous avons présenté une méthode de détection de communautés basée sur des principes simples d'optimisation locale de la modularité qui offre des résultats supérieurs à toutes les méthodes existantes en terme de modularité et surtout en terme de temps de calcul, ce qui permet de traiter des graphes de taille inégalée. De plus, l'approche ne faisant pas de calculs globaux sur le graphe cela permet de lire le graphe sur disque de manière séquentielle si la mémoire vive n'est pas suffisante pour stocker le graphe. Enfin plusieurs heuristiques simples peuvent être appliquées pour diviser les temps de calcul par un rapport 2 sur de grands graphes sans perte notable de modularité.

Parmi les travaux futurs, de nombreux points doivent être éclaircis concernant la méthode elle-même, notamment afin de comprendre l'influence de l'ordre dans lequel sont traités les sommets à chaque itération et l'impact de plusieurs heuristiques envisagées. Ensuite, la méthode doit être utilisée dans des contextes réels et notamment des études en cours ou à venir vont utiliser cette méthode sur des graphes de contacts téléphoniques ou des graphes d'échanges pair-à-pair. Ces études permettent en particulier de mettre en évidence des phénomènes communautaires forts et de pouvoir observer ces communautés à différentes échelles.

Références

[BDG+06] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing modularity is hard, 2006. arXiv :physics/0608255.

[BV04] P. Boldi and S. Vigna. The webgraph framework i : compression techniques. In *WWW '04 : Proceedings of the 13th international conference on World Wide Web*, pages 595–602, New York, NY, USA, 2004. ACM.

[CNM04] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70 :066111, 2004.

[FB07] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *PROC.NATL.ACAD.SCI.USA*, 104 :36, 2007.

[New06] M.E.J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103 :8577, 2006.

[NG04] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69 :026113, 2004.

[PL06] P. Pons and M. Latapy. Computing communities in large networks using random walks (long version). *Journal of Graph Algorithms and Applications (JGAA)*, 10-2 :191–218, 2006.

[WT07] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks, 2007. arXiv :cs/0702048v1.