Détection de structures de communautés dans les grands réseaux d'interactions

Pascal Pons[†]

LIAFA (CNRS – Université Paris 7), 2 place Jussieu 75251 Paris Cedex 05

Les parties localement denses de graphes globalement peu denses (*communautés*) apparaissant dans la plupart des réseaux d'interactions réels jouent un rôle important dans de nombreux contextes. Nous proposons une mesure de la similarité structurelle (au sens des communautés) entre sommets basée sur des marches aléatoires. Nous montrons ensuite comment calculer efficacement cette distance. Nous l'utilisons alors dans un algorithme de clustering hiérarchique qui fonctionne en temps $O(mn^2)$ et en espace $O(n^2)$ dans le pire des cas, cependant il tire profit des caractéristiques des réseaux réaliste, la complexité en temps pour la plupart des cas pratiques est alors de $O(n^2 \log n)$ (n et m sont respectivement le nombre de sommets et d'arêtes du graphe). Les résultats expérimentaux montrent que notre algorithme produit des structures de communautés de meilleures qualité que celles obtenue grâce aux algoritmes existants, tout en restant compétitif du point de vue du temps de calcul.

Keywords: communautés, grands réseaux d'interactions, marches aléatoires, clustering

1 Introduction

L'existence dans les réseaux d'interactions de zones plus densément connectées que d'autres découle souvent de la présence dans le graphe d'une structure de communautés. Ce type de structure intervient dans de nombreux réseaux d'interactions et apporte de l'information sur leur organisation. Les communautés peuvent avoir des interprétations différentes suivant le type de réseau considéré (réseaux sociaux, réseaux biologiques, réseaux d'information, réseaux de télécommunication, ...).

La notion de communauté dans un graphe est cependant difficile à définir formellement. C'est pourquoi toutes les approches récentes ont utilisé une notion intuitive des communauté : une communauté est alors vue comme un ensemble de sommets dont la densité de connexions internes est plus forte que la densité de connexions vers l'extérieur (sans pour autant définir de seuil formel). Le but est alors de trouver une partition des sommets en communautés vérifiant ce critère (sans savoir a priori le nombre de telles communautés). Le domaine a récemment reçu un vif regain d'intérêt avec l'arrivée de nouveaux algorithmes pouvant être classées en deux grandes familles :

- Les approches séparatives [GN02, NG04, RCC$^+$04] scindent le graphe en plusieurs communautés en retirant progressivement les arêtes reliant deux communautés distinctes. Les arêtes sont retirées une à une, à chaque étape les composantes connexes du graphe obtenu sont identifiées à des communautés. Le processus est répété jusqu'au retrait de toutes les arêtes. On obtient alors une structure hiérarchique de communautés. Les méthodes existantes diffèrent par la façon de choisir les arêtes à retirer.
- Les approches agglomératives [New04, DMn04] utilisent une approche s'apparentant à celle du clustering hiérarchique dans lesquelles les sommets sont regroupés itérativement en communautés. Notre approche en fait partie.

2 Evaluation de la similarité et de la différence des sommets grâce à des marches aléatoires courtes

Nous considérons un graphe non orienté $G = (V,E)$ possédant $n = |V|$ sommets et $m = |E|$ arêtes. Afin de pouvoir grouper les sommets du graphe par communautés, nous allons introduire une distance entre

[†]pons@liafa.jussieu.fr

sommets (qui évalue leur proximité structurelle) basée sur des marches aléatoires dans le graphe.

2.1 Marches aléatoires dans un graphe

Nous allons nous intéresser à un processus de marche aléatoire (ou de diffusion) dans le graphe G. Le temps est discrétisé ($t = 0, 1, 2, \ldots$). À chaque instant, un marcheur est localisé sur un sommet et se déplace à l'instant suivant vers un sommet choisi aléatoirement et uniformément parmi les sommets voisins. La suite des sommets visités est alors une marche aléatoire, et la probabilité de transition du sommet i au sommet j est à chaque étape : $P_{ij} = \frac{A_{ij}}{d(i)}$ (où A est la matrice d'adjacence du graphe et $d(i)$ le degré du sommet i).

Ceci définit la matrice de transition P de la chaîne de Markov correspondante. Nous pouvons aussi écrire $P = D^{-1}A$ en introduisant la matrice diagonale des degrés des sommets D ($D_{ii} = d(i)$ et $D_{ij} = 0$ pour $i \neq j$). Nous noterons $P^t_{ij} = (P^t)_{ij}$ la probabilité d'aller du sommet i au sommet j en t étapes.

2.2 Communautés et marches aléatoires courtes

Nous allons considérer des marches aléatoires de longueur fixée t suffisamment courte pour ne pas atteindre le régime stationnaire mais suffisamment longue pour collecter des informations globales sur le voisinage du point de départ de la marche (en pratique nous prenons $3 \leq t \leq 6$). Nous supposons donc connaître pour tous sommets i et j les probabilités P^t_{ij} d'aller de i à j en t pas. Chaque probabilité P^t_{ij} apporte de l'information sur i et j. Ainsi si nous voulons comparer deux sommets i et j nous utiliserons l'information fournie par les probabilités P^t_{ik}, P^t_{jk}, P^t_{ki} et P^t_{kj} pour tout sommet k. Cependant, il est facile de montrer que $d(i)P^t_{ij} = d(j)P^t_{ji}$, cette relation nous montre que ces deux probabilités apportent exactement la même information.

Pour comparer les sommets i et j nous pourrons donc nous contenter de comparer les vecteurs $P^t_{i\bullet}$ et $P^t_{j\bullet}$ correspondant aux lignes i et j de la matrice P^t. Pour celà nous nous appuyons sur le fait que les sommets d'une même communauté ont tendance à "voir" les sommets éloignés de la même façon, ainsi si i et j sont dans la même communauté et k dans une autre communauté il y a de fortes chances que $P^t_{ik} \simeq P^t_{jk}$. Nous devons aussi tenir compte de l'influence du degré $d(j)$ du sommet d'arrivée sur la probabilité P^t_{ij} (cette probabilité est d'ailleurs directement proportionnelle au degré lorsque $t \to \infty$). Dans cette optique, nous introduisons la distance r_{ij} suivante :

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{ik} - P^t_{jk})^2}{d(k)}} = \left\| D^{-\frac{1}{2}} P^t_{i\bullet} - D^{-\frac{1}{2}} P^t_{j\bullet} \right\| \tag{1}$$

Nous généralisons naturellement cette distance entre les communautés (ensembles de sommets) en utilisant les probabilités $P_{C\bullet} = \frac{1}{|C|} \sum_{i \in C} P_{i\bullet}$ pour toute communauté $C \subset V$.

Théorème 1 *La distance r_{ij} s'exprime directement en fonction des vecteurs propres à droite v_α de P associés aux valeurs propres λ_α.*

$$r^2_{ij} = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2 \tag{2}$$

Cette formule établit un lien entre notre distance et les propriétés spectrales de la matrice de transfert, ainsi son comportement est principalement dominé par les vecteurs propres de P liés aux plus grandes valeurs propres. Soulignons que de nombreux travaux ont montré ou constaté que les propriétés structurelles du graphe sont capturées par ces vecteurs propres [SEMS04, DMn04]. Cependant ces approches ont toutes l'inconvénient de devoir calculer explicitement les valeurs propres et les vecteurs propres associés. Notre approche s'appuie sur les mêmes fondements théoriques mais vise à s'affranchir du calcul pénalisant des valeurs propres, comme nous allons le voir dans le paragraphe suivant.

2.3 Calcul de la distance

La définition donné par l'équation (1) de r_{ij} permet un calcul en $O(n)$ à partir des probabilités P_{ik} et P_{jk}. Le calcul exact de ces différentes probabilités peut être fait en calculant la matrice P^t. Les graphes rencontrés en pratique sont généralement peu denses, il est alors judicieux de calculer chaque vecteur $P^t_{i\bullet}$

en multipliant t fois le vecteur $P^0_{i\bullet}$ ($P^0_{i\bullet}(k) = P^0_{ik} = \delta_{ik}$) par P^T. Chaque multiplication demande $O(m)$, le calcul de chaque $P^t_{i\bullet}$ demande donc un temps $O(tm)$.

Suivant la ressource mémoire, nous pourrons soit calculer tous les vecteurs probabilités en un temps $O(tnm)$ et les stocker en mémoire pour pouvoir ensuite faire chaque calcul de distance en un temps $O(n)$, ou bien calculer à la volée chaque distance en un temps $O(tm)$.

3 Description de l'algorithme

Le but est maintenant de construire une structure de communautés en accord avec notre distance. Nous allons pour ceci proposer une variante de la méthode de clustering hiérarchique de Ward.

3.1 Principe de notre algorithme

Nous allons partir d'une partition du graphe en n communautés contenant chacune un seul sommet et fusionner successivement des communautés jusqu'à obtenir une seule communauté correspondant au graphe entier de la façon suivante :

- Choisir deux communautés à fusionner selon la procédure décrite en 3.3.
- Fusionner ces deux communautés en une nouvelle communauté.
- Mettre à jour les distances entre communautés.
- Calculer et mémoriser un paramètre de qualité Q présenté en 3.2

Nous obtenons ainsi une suite de n partitions des sommets en communautés $P_0 \dots P_{n-1}$ parmi lesquelles il va falloir choisir la meilleure (maximisant Q).

3.2 Evaluation de la qualité d'une partition du graphe en communautés

Nous utilisons pour évaluer la qualité de la partition P_k la modularité $Q(P_k) = Q_k$ introduite dans [NG04, New04] :

$$Q_k = \sum_{C \in P_k} (e_C - a_C^2) \tag{3}$$

où e_C est la fraction d'arêtes internes à la communauté C et a_C est la fraction d'arêtes ayant une extrémité dans la communauté C. Cette quantité est calculable en un temps $O(m)$ et peut être mise à jour après chaque fusion en $O(1)$. Nous retenons comme résultat de notre algorithme la partition du graphe possédant la meilleure modularité.

3.3 Choix des communautés à fusionner

Pour diminuer le nombre de cas à considérer et ainsi la complexité, nous envisagerons uniquement les fusions de communautés ayant au moins une arête entre elles. Nous utilisons ensuite le principe de l'algorithme de clustering hiérarchique suivant la méthode de Ward [War63]. Cette méthode fusionne les communautés de telle sorte à minimiser à chaque étape la moyenne σ de la distance au carré de chaque sommet à sa communauté :

$$\sigma(P_k) = \frac{1}{n} \sum_{C \in P_k} \sum_{i \in C} r_{iC}^2 \tag{4}$$

Nous devons alors connaitre à chaque étape les valeur $\Delta\sigma$ des variations de σ après chaque fusion possibles de communautés. Celles-ci peuvent être efficacement obtenues grâce au théorème suivant :

Théorème 2 *La variation $\Delta\sigma(C_1, C_2)$ de σ lors d'une fusion de deux communautés C_1 et C_2 en une nouvelle communauté $C_1 \cup C_2$ est :*

$$\Delta\sigma(C_1, C_2) = \frac{1}{n} \frac{|C_1||C_2|}{|C_1| + |C_2|} r_{C_1 C_2}^2 \tag{5}$$

3.4 Complexité

Le calcul initial des probabilités $P^t_{i\bullet}$ pour tous les sommets i se fait en $O(tmn)$. À chaque étape de fusion, le maintien de la structure de donnée codant les communautés se fait après chaque fusion en temps constant et le calcul de $P^t_{(C_1\cup C_2)\bullet}$ se fait en temps $O(n)$. Ces deux étapes demandent donc pour les $n-1$ étapes un temps total en $O(n^2)$. La partie la plus coûteuse est le calcul des nouvelle valeurs de $\Delta\sigma(C_1\cup C_2, C_3)$. À chaque fusion, il faut recalculer les distances entre la nouvelle communauté créée et chacune de ses communautés voisines en temps $O(n)$.

Théorème 3 *Le nombre total de calculs de distances effectué lors des fusion est au plus* $2Hm$ *où* H *est la hauteur de la structure arborescente des communautés trouvées par l'algorithme.*

La complexité totale de l'algorithme est donc $O(mnH)$, soit $O(mn^2)$ dans le pire des cas. Cependant les graphes réalistes sont peu denses (m = $O(n)$) et leurs communautés s'organisent généralement de manière hiérarchique, de plus l'algorithme de Ward que nous utilisons a tendance à créer des communautés de tailles équilibrées. Ces deux remarques font qu'en pratique la structure arborescente hiérarchique des communautés s'approche du cas favorable où $H = O(\log n)$ qui correspond à une performance de l'algorithme $O(n^2 logn)$.

4 Conclusion

Nous avons proposé et implémenté (disponible sur [www]) un algorithme efficace de détection de communautés dans les grands réseaux d'interactions basé sur l'étude des marches aléatoires dans les graphes. Notre approche s'applique à des graphes de grandes tailles qui ne pouvaient pas être traités par la plupart des algorithmes existants. Ainsi, nous avons pu obtenir les structures de communautés de graphes de tailles allant jusqu'à 100 000 sommets. Cette courte présentation ne peut malheureusement contenir ni les nombreuses améliorations et optimisations que nous avons proposées ni les tests de comparaison que nous avons effectués. Ces tests ont souligné la qualité des partitions trouvées par notre algorithme par rapport à celles trouvées par les autres algorithmes existants.

Références

[DMn04] Luca Donetti and Miguel A. Muñoz. Detecting network communities : a new systematic and efficient algorithm. *arXiv :cond-mat/0404652*, 2004.

[GN02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12) :7821–7826, 2002.

[New04] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(6) :066133, 2004.

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(2) :026113, 2004.

[RCC+04] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *PNAS*, 101(9) :2658–2663, 2004.

[SEMS04] Ingve Simonsen, Kasper Astrup Eriksen, Sergei Maslov, and Kim Sneppen. Diffusion on complex networks : a way to probe their large-scale topological structures. *Physica A : Statistical Mechanics and its Applications*, 336(1-2) :163–173, May 2004.

[War63] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[www] `http://www.liafa.jussieu.fr/~pons/`.