Propagation de labels avec barrages sur de grands graphes en utilisant Apache Hadoop et Apache Spark (GraphX)

Jean−Philippe Attal [*,**] **— Maria Malek** [*]

[*] *EISTI: Ecole Internationale des Sciences du Traitement de l'Information, laboratoire Quartz, Cergy-France 95000*
Email: jal@eisti.eu
Email: mma@eisti.eu
[**] *Universite de Cergy-Pontoise CNRS France*
Email: jal@eisti.eu

RÉSUMÉ. *La propagation de labels est l'une des méthodes les plus rapides pour la détection de communautés, de complexité quasi−linéaire en terme d'arêtes. Il s'agit d'une méthode locale où chaque nœud possède son propre label qui change par interaction avec son voisinage. Malheureusement, cette méthode présente deux inconvénients majeurs. Le premier est qu'une mauvaise propagation peut mener à de trop grandes communautés. Le second inconvénient est l'instabilité de la méthode, ne donnant que très rarement le même résultat après chaque lancement. Dans cet article, nous proposons des algorithmes et une étude portant sur la propagation de labels en plaçant des barrages sur certaines arêtes dans le but d'éviter de mauvaises propagations. Une méthode d'ensemble learning basée sur l'alimentation d'une matrice de fréquence de co-apparition dans le but de stabiliser la propagation de labels est ensuite appliquée. Dans le but d'appliquer ces algorithmes sur de plus grands graphes, nous avons développé des versions parallèles et distribuées de la propagation de labels sur Hadoop utilisant le framework MapReduce et sur Apache Spark, solution "in-memory", qui utilise la notion de RDD (Resilient Distributed Dataset). Nous avons appliqué nos algorithmes sur différents graphes dont Amazon où nous avons pu faire une étude portant sur les communautés.*

ABSTRACT.

MOTS-CLÉS : *détection de communautés*$_1$, *propagation de labels*$_2$, *barrages*$_3$, *ensemble learning*$_4$, *Hadoop*$_5$, *Apache Spark*$_6$, *GraphX* $_7$, *MapReduce*$_8$, *RDD*$_9$

KEYWORDS:

1. Introduction

La plupart des réseaux représentant des systèmes réels montrent des caractéristiques propres comme des groupes de noeuds fortement liés entre eux (que l'on appelle des communautés) et peu avec le reste du graphe. Des études comparatives ont été effectuées par Fortunato et al. (Fortunato, 2010) ou encore par Kanawati (Yakoubi et Kanawati, 2014).

Dans cet article, nous exposons de nouveaux algorithmes de détection de communautés (et leurs variantes). Ces algorithmes sont basés sur la propagation de labels et sur la mesure de centralité d'intermédiarité des liens (appelée encore "edge betweenness").

2. L'approche par propagation de labels et détection de coeurs

La méthode de propagation de labels (Raghavan *et al.*, 2007) est basée sur la transmission d'informations d'un nœud à ses voisins. L'information étant un label. Cependant, l'algorithme de propagation de labels présente l'inconvénient d'être instable, ne donnant que rarement le même résultat après plusieurs lancements. Il est aussi caractérisé par un problème intrinsèque conduisant dans certains cas à de très grandes communautés (monstres). Cette méthode étant très instable, il serait souhaitable d'avoir une méthode de stabilisation pour pallier l'instabilité, nous utilisons dans ce sens la méthode de Seifi et al. (Seifi *et al.*, 2013). Nous nommerons cette méthode le *stabilisateur*.

3. Approche proposée

La propagation de labels avec barrages (PLAB) a pour objectif de détecter des communautés, utilisant à la fois l'information topologique globale du graphe et une méthode locale pour trouver les communautés. Les mesures issues de l'analyse des réseaux sociaux peuvent se révéler très utiles pour connaître l'importance de certaines arêtes ou nœuds au sein d'un graphe. Nous utilisons ici la mesure d'intermédiarité (Girvan et Newman, 2002) qui nous permettra de placer des barrages lors de l'exécution de la propagation de labels. Nous proposons deux grandes familles d'algorithmes avec barrages.

La première est basé sur l'optimisation d'une fonction de qualité qui peut par exemple être la modularité ou la conductance. Nous alimentons K matrices de co−appartenance avec différents niveaux de barrages. Nous calculons les composantes connexes pour chaque matrice, et sélectionnons la solution ayant le meilleur score selon la fonction de qualité choisie. Nous avons nommé cet algorithme la **M**ultiple **P**ropagation de **L**abels avec **B**arrages et **S**tabilisation avec optimisation d'une fonction de qualité (MPLBS)

La deuxième famille consiste à alimenter une seule matrice de co-appartenance avec différents niveaux de barrages et calculer les composantes connexes. Nous nommons cette méthode l'algorithme de **P**ropagation de **L**abels avec **B**arrages utilisant la **s**tabilisation (PLBS).

La complexité de ces algorithmes est gouvernée par le calcul de la centralité d'intermédiarité, qui est en $\mathcal{O}(n^2)$, n tant le nombre de noeuds. Nous exposons les algorithmes MPLBS 1 et PLBS 2 :

Algorithme 1 MPLBS

Paramètres : Un graphe $G = (V, E)$, un seuil α, $\mathcal{N}$ le nombre d'essais, Δ le pas
Sortie : communautés de G

1: Calcul de la centralité d'intermédiarité de G.
2: **Pour** $i = 0$ à 1 avec un pas de Δ **Faire**
3: Mettre des barrages sur les $\Delta \times |E|$ (β) arêtes ayant les plus grandes valeurs d'intermédiarité
4: Calcul du *stabilisateur* avec α en utlisant la propagation de labels
5: **Fin Pour**
6: **Retourner** La partition avec le meilleur score de la fonction de qualité choisie par l'utilisateur : $P = \{P_1, ..., P_{\mathcal{C}}\}$.

La complexité du MPLBS est en $\mathcal{O}(n^2) + \mathcal{O}(\frac{1}{\Delta} \times \mathcal{N} \times k \times (m - \beta m))$.

Algorithme 2 PLBS

Paramètres : Un graphe $G = (V, E)$, un seuil α, $\mathcal{N}$ le nombre d'essais, Δ le pas
Sortie : communautés de G

1: Calcul de la centralité d'intermédiarité de G.
2: Allocation d'une matrice de fréquence de co-appartenance vide
3: **Pour** $i = 0$ à 1 avec un pas Δ **Faire**
4: Mettre des barrages sur les $\Delta \times |E|$ (β) arêtes ayant les plus grandes valeurs d'intermédiarité .
5: Lancer $\mathcal{N}$ fois la propagation de labels avec un nombre différent de barrages.
6: Remplir la matrice de fréquence de co-appartenance avec les résultats des différentes propagations de labels.
7: **Fin Pour**
8: Créer un nouveau graphe $G' = (V, E')$ en partant de $P_{ij}^{\mathcal{N}}$avec des arêtes dont la pondération est égale ou supérieure à α
9: Creér une partition P en considérant les $\mathcal{C}$ composantes connexes.
10: **Retourner** La partition $P = \{P_1, ..., P_{\mathcal{C}}\}$.

La complexité du PLBS est en $\mathcal{O}(n^2) + \mathcal{O}(\frac{1}{\Delta} \times \mathcal{N} \times k \times (m - \beta m))$.

4. Traitement des grands graphes

Les graphes, pouvant représenter des systèmes complexes, peuvent être également de très grandes tailles. Appliquer certains algorithmes sur des graphes de grandes tailles peut conduire à des problèmes de mémoire, et à un échec du processus. Il est ainsi souhaitable de pouvoir décomposer le travail sur plusieurs machines, en parallélisant l'algorithme et en distribuant les données.

Hadoop est une structure logicielle ayant pour but le développement d'application distribuées et échelonnables (dont la volumétrie des données peut être à différents niveaux), sur un très grand nombre de machines. Cela permet ainsi de pouvoir travailler sur de grandes quantités de données. C'est sur cette structure logicielle qu'a été implémenté le paradigme de programmation *MapReduce* par Google (Dean et Ghemawat, 2008). Un autre framework open source de calcul distribué est Apache Spark, utilisant la notion de RDD (resilient distributed dataset), collection d'objets distribués immutable. Nous avons développé des versions modifiées du PLBS et du MPLBS sous Hadoop et sous Spark.

5. Expérimentations et analyse

Nous avons expérimenté nos algorithmes sur des réseaux connus de la littérature, comme le club de Zachary (Zachary, 1977), un réseau footballistique (Girvan et Newman, 2002), un réseau de livres politiques américains (Krebs, 2004), un réseau de dauphins (Lusseau *et al.*, 2003), mais également des graphes de plus grandes tailles comme Amazon, You tube et DBLP. Nous exposons les résultats de nos algorithmes via une étude comparative, en utilisant les algorithmes de Louvain (Blondel *et al.*, 2008), de Walktrap (Walk) (Pons et Latapy, 2006), de Givan-Newman (GN) (Girvan et Newman, 2002), de Leung et al. (Leung *et al.*, 2009), de LPA et de DPA (Šubelj et Bajec, 2011) et de Licod (Kanawati, 2011). Nous utilons pour notre évaluation la conductance Φ, la modularité Q, le NMI, l'indice de rand ajusté (ARI) et la pureté.

Les expérimentations ont montré que l'ajout de barrages permettait l'augmentation de la qualité de partionnement. MPLBS et PLBS donnent des résultats très bons en terme de NMI (Ana et Jain, 2003) vis à vis d'autres algorithmes de la littérature.

6. Conclusion et perspectives

Dans cet article, nous avons exposé de nouveaux algorithmes basés sur la propagation de labels. Nos expérimentations ont montré que notre méthode hybride, liant propagation de labels et barrages issus de l'intermédiarité des arêtes permettait l'obtention de résultats satisfaisants. Nous avons également observé qu'alimenter une matrice de fréquences de co−apparition en faisant varier le nombre de barrages permettait d'augmenter la qualité des communautés obtenues. De même, alimenter K matrices de fréquences de co−apparition avec différents niveaux de barrages, et considérant la

Experiences avec d'autres algorithmes													
Algorithms	Q	Φ	NMI	ARI	Pureté	$\#$	*Algorithms*	Q	Φ	NMI	ARI	Pureté	$\#$
Foot $\#11$							*Dol* $\#2$						
Louvain	0.6021	0.2778	0.855	0.7277	0.9217	9	Louvain	0.5185	0.2451	0.5109	0.3274	0.9677	5
GN	0.6005	0.290	0.8788	0.7781	0.8347	10	GN	0.5193	0.2001	0.5541	0.3949	0.9838	5
Spin	0.6052	0.293	0.8922	0.8165	0.86956	10	Spin	0.5285	0.2339	0.5864	0.3734	1.0	5
Walk	0.6038	0.2951	0.8874	0.8154	0.9217	10	Walk	0.4888	0.1798	0.5372	0.4167	0.9516	4
Leung	0.5689	0.353	0.900	0.8361	0.9217	13	Leung	0.5189	0.2708	0.4811	0.2908	0.9677	6
LPA	0.5946	0.2974	0.8865	0.7787	0.8544	10.39*	LPA	0.4831	0.1720	0.6226	0.5024	0.9790	3.85*
DPA	0.606		0.897				DPA	0.529		0.774			
LICOD	0.49		0.83	0.69		16	LICOD	0.35		0.41	0.32		2
PLBS	0.5822	0.3377	**0.9371**	**0.9154**	0.9652	17	**PLBS**	0.3237	0.1682	0.578	0.6689	0.9677	8
PLAB	0.5985	0.3167	0.9289	0.9004	0.9391	13	**PLAB**	0.3761	0.0549	**0.9429**	**0.9563**	1.0	3
MPLBS	0.6019	0.3102	0.9269	0.8893	0.9304	12	**MPLBS**	0.4569	0.0431	0.5979	0.6671	0.8476	2

Tableau 1. *Résultats comparatifs avec d'autres algorithmes ($\#$ signifie le nombre de communautés et $*$ signifie que le résultat est une moyenne sur* 100 *lancements car instable.)*

Foot $\#11$	PLAB	PLBS	MPLBS	*Dol* $\#2$	PLAB	PLBS	MPLBS
α	$\{0.6\}$	$\{0.5\}$	$\{0.5\}$	α	$\{0.6\}$	$\{0.5\}$	$\{0.3-0.5\}$
β	$\{0.05\}$	$-$	$\{0.3\}$	β	$\{0.05\}$	$-$	$\{0.3\}$

Tableau 2. *Paramétrisation de nos algorithmes*

matrice donnant une partition via ces composantes connexes avec une mesure comme la modularité donnait aussi de très bons résultats. Nous développons actuellement une mesure d'analyse des réseaux sociaux pour approximer la betweenness pour de très grands graphes utilisant Hadoop et Spark.

Remerciements

Ce projet est financé par Square Predict, projet "investissement d'avenir" pour le Big Data et les assurances.

7. Bibliographie

Ana L., Jain A. K., « Robust data clustering », *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, IEEE, p. II-128, 2003.

Blondel V., Guillaume J., Lambiotte R., Mech E., « Fast unfolding of communities in large networks », *J. Stat. Mech*p. P10008, 2008.

Dean J., Ghemawat S., « MapReduce : simplified data processing on large clusters », *Communications of the ACM*, vol. 51, n° 1, p. 107-113, 2008.

Fortunato S., « Community detection in graphs », *Physics Reports*, vol. 486, n° 3, p. 75-174, 2010.

Girvan M., Newman M. E. J., « Community structure in social and biological networks », *Proceedings of the National Academy of Sciences*, vol. 99, n° 12, p. 7821-7826, 2002.

Kanawati R., « Licod : Leaders identification for community detection in complex networks », *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, IEEE, p. 577-582, 2011.

Krebs V., « Books about US politics : », , `http://www.orgnet.com/`, 2004.

Leung I. X., Hui P., Lio P., Crowcroft J., « Towards real-time community detection in large networks », *Physical Review E*, vol. 79, n° 6, p. 066107, 2009.

Lusseau D., Schneider K., Boisseau O. J., Haase P., Slooten E., Dawson S. M., « The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations », *Behavioral Ecology and Sociobiology*, vol. 54, n° 4, p. 396-405, 2003.

Pons P., Latapy M., « Computing Communities in Large Networks Using Random Walks », *Journal of Graph Algorithms and Applications*, vol. 10, n° 2, p. 191-218, 2006.

Raghavan U. N., Albert R., Kumara S., « Near linear time algorithm to detect community structures in large-scale networks », *Physical Review E*, vol. 76, n° 3, p. 036106, 2007.

Seifi M., Junier I., Rouquier J.-B., Iskrov S., Guillaume J.-L., « Stable community cores in complex networks », *Complex Networks*, Springer, p. 87-98, 2013.

Šubelj L., Bajec M., « Unfolding communities in large complex networks : Combining defensive and offensive label propagation for core extraction », *Physical Review E*, vol. 83, n° 3, p. 036103, 2011.

Yakoubi Z., Kanawati R., « Licod : A leader-driven algorithm for community detection in complex networks », *Vietnam Journal of Computer Science*, vol. 1, n° 4, p. 241-256, 2014.

Zachary W., « An information flow model for conflict and fission in small groups », *Journal of Anthropological Research*, vol. 33, p. 452-473, 1977.

Zaharia M., Chowdhury M., Franklin M. J., Shenker S., Stoica I., « Spark : cluster computing with working sets », *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, p. 10-10, 2010.