PALM: Un algorithme parallèle pour extraire des clusters de liens dans les réseaux sociaux

Erick Stattner*, Reynald Eugenie*, Martine Collard*

* Université des Antilles
Laboratoire LAMIA
{erick.stattner, reynald.eugenie, martine.collard}@univ-antilles.fr

Résumé. Dans cet article, nous nous intéressons à l'optimisation du processus de recherche de clusters de liens. Nous proposons en particulier l'algorithme PALM (Stattner et al., 2017), qui vise à améliorer l'efficacité du processus d'extraction par l'exploration conjointe de plusieurs zones de l'espace de recherche. Ainsi, nous commençons par démontrer que l'espace des solutions forme un treillis de concepts. Nous proposons ensuite une approche qui explore en parallèle les branches de ce treillis tout en réduisant l'espace de recherche en s'appuyant sur différentes propriétés. Les bonnes performances de notre algorithme sont démontrées en le comparant avec l'algorithme d'extraction d'origine.

1 Introduction

Dans cet article, nous nous intéressons à l'optimisation d'une approche récente de clustering de réseau appelée la recherche de *liens conceptuels*. Il s'agit d'une nouvelle approche qui effectue des clusters de liens en exploitant la structure et les attributs des noeuds pour identifier les liens fréquents entre des groupes de noeuds au sein desquels les attributs sont communs. Ce travail est motivé par le fait que l'algorithme d'extraction d'origine effectue la recherche des clusters de liens de manière séquentielle, sans prendre en compte les parallélisations possibles.

Ainsi dans cet article, nous présentons PALM (Stattner et al., 2017), un algorithme parallèle qui vise à améliorer l'efficacité du processus d'extraction des liens conceptuels en explorant simultanément plusieurs zones de l'espace de recherche. Pour cela, nous démontrons que l'espace des solutions forme un treillis de concepts. Nous proposons ensuite une approche qui explore en parallèle les branches du treillis tout en réduisant l'espace de recherche en s'appuyant sur certaines propriétés des liens conceptuels. L'efficacité de l'algorithme est démontré en l'appliquant à un réseau de télécommunications et en comparant les performances avec l'algorithme d'extraction d'origine. Les résultats obtenus montrent un gain significatif sur le temps de calcul.

2 État de l'art

La recherche de clusters dans les réseaux sociaux, aussi appelée *clustering de réseaux*, est une des approches les plus répandues de la modélisation descriptive de réseaux. L'objectif est

d'identifier des groupes de noeuds qui satisfont certaines propriétés. Les principales familles de méthodes pour le clustering de réseaux sociaux peuvent être classifiées comme suit.

(i) Le clustering basé sur les liens fait référence à une famille de méthodes qui recherche une partition des noeuds, appelé *communauté*, en tenant uniquement compte de la structure du réseau (Fortunato, 2010; Blondel et al., 2008). L'objectif est de décomposer le réseau en plusieurs communautés, définies comme des groupes de noeuds fortement connectés. Les algorithmes tentent ainsi d'identifier les groupes qui maximisent les liens intra-communautaires tout en minimisant les liens inter-communautaires (Newman, 2006).

(ii) Le clustering hybride est une approche de clustering de réseau qui vise à tenir compte des attributs des noeuds durant la phase d'extraction des clusters (Zhou et al., 2009; Yoon et al., 2011). En effet, dans de nombreuses applications la définition classique d'une *communauté* ne permet pas de comprendre pleinement les structures étudiées. Ainsi, les techniques de clustering hybrides tentent d'identifier les groupes de noeuds densément connectés, qui possèdent en plus une similitude dans leurs attributs.

(iii) La recherche de liens conceptuels est une nouvelle approche qui exploite les informations disponibles à la fois sur la structure du réseau et les attributs des noeuds, dans le but d'identifier l'ensemble des attributs les plus fréquemment connectés dans le réseau (Stattner et Collard, 2012). De tels groupes fournissent une connaissance sur les attributs qui structurent les liens au sein du réseau.

3 Liens conceptuels

Soit $G = (V, E)$ un réseau, dans lequel V est l'ensemble des noeuds et E l'ensemble des liens avec $E \subseteq V \times V$. L'ensemble V est défini comme une relation $R(A_1, ..., A_p)$ où chaque A_i est un attribut et $|R| = p$. Ainsi, chaque noeud $v \in V$ est défini par le tuple $(a_1, ..., a_p)$ où $\forall k \in [1..p], v[A_k] = a_k$ correspond à la valeur de l'attribut A_k du noeud v.

Un item est une expression logique $A = x$ où A est un attribut et x une valeur. L'item vide est noté $\emptyset$. Un itemset est une conjonction d'items, par exemple *($A_1 = x$ et $A_2 = y$ et $A_3 = z$)*, qu'on note par simplicité (xyz). Quand un itemset est une conjonction de k items non-vides, on parle de k-itemset.

Soient m et sm deux itemsets. Si $sm \subset m$, on dit que sm est un sous-item de m et que m est un super-itemset de sm. Par exemple $sm = xy$ est un sous-itemset de $m = xyz$. Nous notons I_V l'ensemble des itemsets construits à partir des noeuds de V.

Definition 1. Lien conceptuel Posons m_1 et m_2 deux itemsets de I_V et V_{m_1}, V_{m_2}, respectivement les ensembles de noeuds dans V qui satisfont m_1 et m_2. Nous notons $E_{(m_1,m_2)}$ le cluster de liens connectant des noeuds de V_{m_1} à des noeuds de V_{m_2}, c'est-à-dire :

$$E_{(m_1,m_2)} = \{e \in E \; ; \; e = (v_1, v_2) \quad v_1 \in V_{m_1} \;\; et \;\; v_2 \in V_{m_2}\} \tag{1}$$

Le cluster $E_{(m_1,m_2)}$ est appelé "*lien conceptuel*" dans la mesure où il ne s'agit pas d'un lien du réseau, mais d'un cluster de liens entre deux groupes de noeuds qui peuvent être vus comme des "*concepts*", au sens de l'analyse de concepts formels, c'est-à-dire des objets qui partagent des attributs communs. Par exemple, si m_1 est l'itemset cd et m_2 est l'itemset efj, le *lien conceptuel* $E_{(m_1,m_2)} = (cd, efj)$ inclut tous les liens entre les noeuds qui vérifient cd et qui sont connectés à des noeuds qui vérifient efj. Nous notons L_V l'ensemble des liens conceptuels construits à partir des itemsets de I_V.

Le *support* du lien conceptuel $E_{(m_1,m_2)}$ de L_V, est le pourcentage de liens appartenant à $E_{(m_1,m_2)}$, i.e. $supp(E_{(m_1,m_2)}) = \frac{|E_{(m_1,m_2)}|}{|E|}$. Le lien conceptuel entre les itemsets m_1 et m_2 est fréquent si le support de $E_{(m_1,m_2)}$ est plus grand qu'un *seuil de support* minimum β donné, i.e. $supp(E_{(m_1,m_2)}) > \beta$.

Ainsi, nous définissons LC comme l'ensemble des clusters de liens fréquents (les liens conceptuels fréquents) extraits du réseau G.

$$LC = \bigcup_{m_1 \in I_V, m_2 \in I_V} \{E_{(m_1,m_2)} \; ; \; supp(E_{(m_1,m_2)}) > \beta\} \tag{2}$$

Definition 2. Sous-lien conceptuel Soient deux itemsets sm_1 et sm_2 respectivement sous-items de m_1 et m_2. Le lien conceptuel (sm_1, sm_2) est dit *sous-lien conceptuel* de (m_1, m_2) et nous notons $(sm_1, sm_2) \subseteq (m_1, m_2)$. Symétriquement, (m_1, m_2) est le *super-lien* de (sm_1, sm_2).

Propriété 1. Fermeture descendante Si un lien conceptuel est fréquent, tous ses sous-liens le sont également. De la même façon, si un lien est non-fréquent, tous ses sur-liens sont également non-fréquents.

Preuve. Soient sm_1 et sm_2 respectivement des sous-itemsets de m_1 and m_2. Les propriétés $V_{m_1} \subseteq V_{sm_1}$ et $V_{m_2} \subseteq V_{sm_2}$ se vérifient. En conséquence, $(m_1, m_2) \subseteq (sm_1, sm_2)$ et donc $|(m_1, m_2)| \leq |(sm_1, sm_2)|$.

Definition 3. Liens conceptuels fréquents maximaux. Soit β un seuil de support donné, nous appelons *liens conceptuels fréquents maximaux*, tout lien conceptuel fréquent pour lequel il n'existe aucun super-lien qui soit également fréquent.

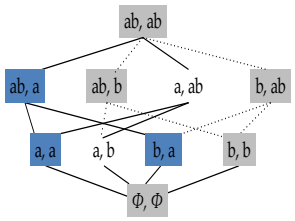


FIG. 1 – *Exemple de treillis de concepts avec (ab ;b) fréquent et maximal, et (b,b) non-fréquent (ainsi les branches non pertinentes, en pointillées, sont élaguées)*

Ainsi, la relation $\subseteq$ qui définit un ordre partiel sur I_V peut être étendue à L_V. En effet, comme le montre la Figure 1, $(L_V, \subseteq)$ induit un treillis de concepts qui peut etre utilisé pour extraire les liens conceptuels fréquents et maximaux.

4 Algorithme PALM

L'algorithme d'origine effectuait la recherche des liens conceptuels en 4 étapes. (i) Génération des 1-itemsets candidats (ii) Génération des 1-liens conceptuels fréquents, à partir des 1-itemsets candidats (iii) Génération des t-itemsets candidats à l'itération t > 1 (iv) Génération des t-liens conceptuels à l'itération t > 1, à partir des t-itemsets candidats

Les parties (iii) et (iv) sont répétées jusqu'à ce qu'il n'y ait plus de nouveaux candidats générés. Bien que cet algorithme permette effectivement d'extraire les liens conceptuels fréquents et maximaux, la recherche s'effectue de manière séquentielle. De plus, l'exploration de l'espace de recherche n'est pas totalement optimisée, ce qui rend l'extraction des liens conceptuels particulièrement coûteuse en temps pour de grands jeux de données.

La première amélioration apportée par PALM concerne la phase de création des candidats. Dans le précédent algorithme, la génération des t-itemsets candidats est effectuée en fusionnant les itemsets m_1 et m_2 possédant t-2 items communs et tel que $(m_1, m_2) \in LI^2_{(t-1)}$ ou $(m_1, m_2) \in RI^2_{(t-1)}$. Or, deux itemsets ont k items en commun si et seulement s'ils partagent au moins un sous-itemset de taille k. Ainsi, en conservant la structure du treillis de concepts nous pouvons explorer les (t-2)-itemsets fréquents et fusionner leurs sous-itemsets respectifs.

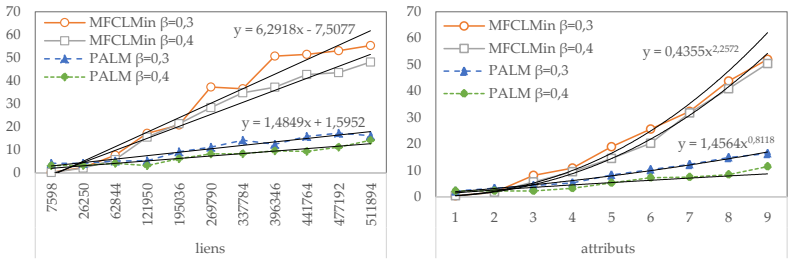
A l'étape (iv) de l'ancien algorithme, l'ensemble des liens conceptuels testés est constitué de toutes les combinaisons possibles faites à partir des itemsets de LI_{cand} et des itemsets de RI_{cand}. La fréquence de chacune de ces combinaisons est ensuite évaluée. Cette étape est la plus coûteuse en temps. Afin de réduire le temps de calcul nécessaire, nous réduisons le nombre de liens conceptuels candidats à tester en exploitant la propriété 1 de fermeture.

Enfin, plusieurs étapes de la recherche sont indépendantes et peuvent être parallélisées. En effet dans l'étape (i), nous explorions tous les noeuds du réseau afin d'extraire les 1-itemsets. Dans PALM les noeuds sont répartis sur **T** threads, chargés chacun de l'extraction des 1-itemsets de sa portion de noeuds, puis de la fusion dans une liste globale. Dans la partie (ii), les nouveaux clusters candidats sont générés par jointure des éléments précédents. Cette étape a également été parallélisée en répartissant le travail sur **T** threads.

5 Résultats expérimentaux

Le jeu de données utilisé est un réseau d'appels fourni par un opérateur de téléphonie mobile local. Il représente des abonnés le 1e juin 2009 de 5h à 15h. Les noeuds sont les abonnés et les liens sont les appels passés sur la période. Le réseau a une structure scale-free et est composé d'environ 246 000 noeuds et 510 000 liens collectés sur les 10h d'étude.

Dans une première approche, nous avons étudié l'évolution du temps de calcul (en secondes) avec différents seuils ($\beta = 0.3$ and $\beta = 0.4$) pour les deux algorithmes (cf Figure 2).

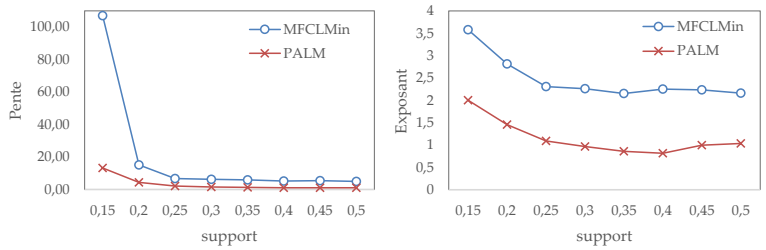


(a) (b)

FIG. 2 – *Temps de calcul (sec.) de MFCLMin et PALM selon (a) liens et (b) attributs*

Tout d'abord, nous observons que pour les deux seuils utilisés, le temps de calcul est toujours plus petit pour l'algorithme PALM. Nous observons cependant des tendances communes. En effet, le temps de calcul croît linéairement avec la taille du réseau, alors qu'il peut être approché par une fonction de puissance quand le nombre d'attributs augmente. Ces tendances ont été observées pour plusieurs seuils de support.

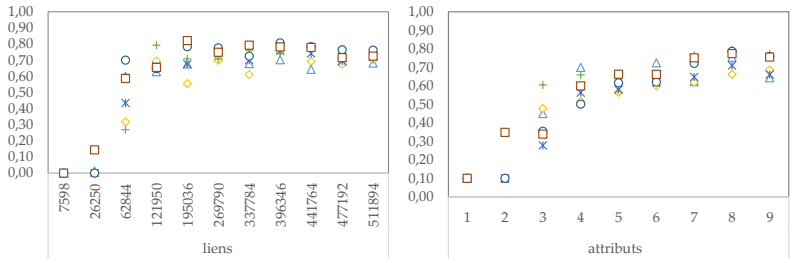
Pour mieux comprendre l'évolution du temps de calcul, nous nous sommes focalisés sur l'évolution de (a) la pente du temps de calcul et (b) l'exposant de la fonction de puissance. La Figure 3 montre les résultats avec $\beta \in [0.15..0.5]$.



(a) (b)

FIG. 3 – *Évolution de (a) la pente et (b) l'exposant de la courbe du temps de calcul*

Si nous nous concentrons sur l'évolution de la pente, nous pouvons observer que pour les deux algorithmes, la pente du temps de calcul croît quand le seuil de support décroît. Cependant contrairement à l'algorithme d'origine, PALM explore plus rapidement l'espace des solutions, en particulier pour de petits seuils de support. Un comportement similaire est observé si nous nous intéressons à l'exposant de la courbe du temps de calcul.



(a) (b)

FIG. 4 – *Gain sur le temps de calcul comparé à l'algorithme d'origine pour divers seuils de support selon (a) le nombre de liens et (b) le nombre d'attributs*

Pour aller plus loin, nous avons cherché à comprendre quel était le gain sur le temps de calcul apporté par PALM (cf Figure 4). Dans notre contexte, le gain est défini comme la proportion de temps économisée par rapport à l'algorithme d'origine. Le gain fourni par l'algorithme PALM croît rapidement avec la taille du réseau et semble se stabiliser. En effet, le gain est d'environ 75% et reste stable même avec un nombre de liens élevé. Les mêmes tendances sont observées pour les résultats obtenus selon le nombre d'attributs.

6 Conclusion

Dans cet article, nous nous sommes intéressés à l'optimisation de la recherche des liens conceptuels. (i) Nous avons formellement décrit la notion de liens conceptuels et montré que l'espace de solutions forme un treillis de concepts. (ii) Nous avons proposé l'algorithme PALM, qui vise à améliorer l'efficacité du processus d'extraction en explorant simultanément plusieurs zones de l'espace de solutions, tout en réduisant l'espace de recherche. (iii) Enfin, nous avons implémenté l'algorithme en Java et nous avons évalué ses performances sur un réseau de communications téléphoniques. L'efficacité de l'algorithme a été démontré en comparant les performances par rapport à l'algorithme d'origine. Comme perspectives, nous voulons étendre ce travail de parallélisation en distribuant les calculs sur des clusters de machines et utiliser les concepts du big data dans le but d'adapter l'algorithme proposé à des frameworks big data.

Références

Blondel, V., J. L. Guillaume, R. Lambiotte, et E. Lefebvre (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment 2008*.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports 486*, 75–174.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences 103*(23), 8577–8582.

Stattner, E. et M. Collard (2012). Social-based conceptual links : Conceptual analysis applied to social networks. *IEEE Conference on Advances in Social Networks Analysis and Mining*.

Stattner, E., R. Eugenie, et M. Collard (2017). Palm : A parallel mining algorithm for extracting maximal frequent conceptual links from social networks. In *International Conference on Database and Expert Systems Applications*, pp. 259–274. Springer.

Yoon, S.-H., S.-S. Song, et S.-W. Kim (2011). Efficient link-based clustering in a large scaled blog network. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pp. 71 :1–71 :5. ACM.

Zhou, Y., H. Cheng, et J. Yu (2009). Graph clustering based on structural/attribute similarities. *VLDB Endowment 2*(1), 718–729.

Summary

In this paper, we focus on the optimization of the search for cluster of links. In particular, we propose PALM (Stattner et al., 2017), a parallel algorithm that aims to improve the efficiency of the extraction by simultaneously exploring several areas of the search space. For this purpose, we begin by demonstrating that the solution space forms a concept lattice. Then, we propose an approach that explores in parallel the branches of the lattice while reducing the search space based on various properties of the clusters. We demonstrate the efficiency of the algorithm by comparing the performances with the original extraction approach. The results obtained show a significant gain on the computation time.