Détection de communautés dans les réseaux sociaux

Rapport de projet de troisième année

Dimitri Lozeve Teven Le Scao

15 mars 2017

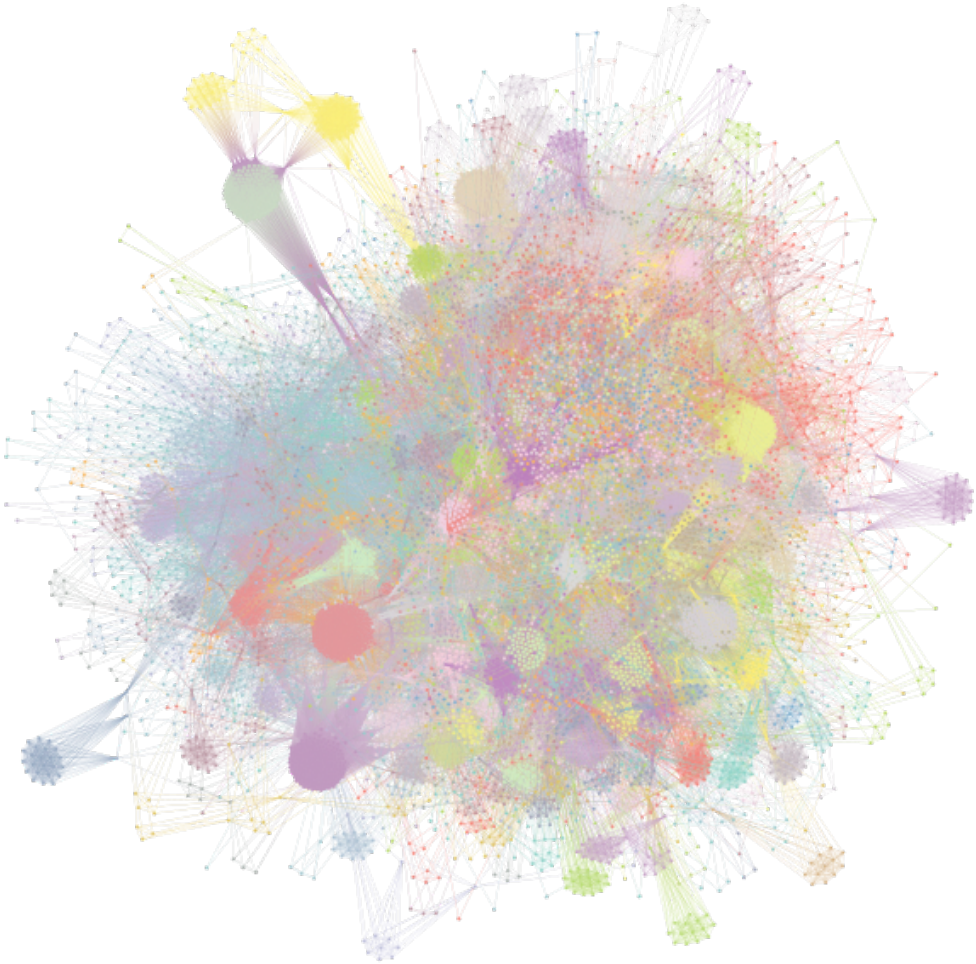


Table des matières

1 Introduction **3**

2 État de l'art **4**
2.1 Notion de communauté 4
2.2 Clustering spectral 5
2.3 Clustering par modularité 6

3 Le Stochastic Block Model est-il proche de modèles réels ? **8**
3.1 Justesse de l'approximation par un SBM donné 8
3.2 Justesse de la ressemblance avec un SBM en général 9

4 Validation croisée **11**
4.1 Validation croisée en tant que mesure de qualité 11
4.1.1 Méthode générale et illustration 11
4.1.2 Validation k-fold 11
4.1.3 Implémentation 13
4.1.4 Résultats 14
4.2 Comparaison de méthodes par validation croisée 15
4.3 Validation croisée et *ground-truth* 15

5 Applications : Prestashop **17**
5.1 Les données Prestashop 17
5.1.1 Objectif de l'étude 17
5.1.2 Description des données 17
5.1.3 Construction des graphes 18
5.2 Étude des graphes de Prestashop 19
5.2.1 Répartition des degrés 19
5.2.2 Application de la validation croisée 20
5.2.3 Détection de communautés chez les utilisateurs et les produits de Prestashop 21
5.3 Conclusion sur le problème de la recommandation 22

6 Conclusion **24**

Bibliographie **25**

1 Introduction

La science des réseaux est un outil précieux pour de nombreuses disciplines scientifiques. De nombreux systèmes réels tombent en effet dans son champ d'application : on pourrait ainsi citer des domaines aussi divers que la sociologie, qui repose sur les interactions entre une multitude d'êtres humains, la biologie, quand par exemple elle cherche à comprendre le cerveau ou les chaînes alimentaires, ou enfin l'informatique, qui par nature fait communiquer différents éléments entre eux.

Ce dernier domaine fournit, en plus d'exemples d'application, les outils de calcul nécessaires à traiter des graphes aussi grands que, par exemple, ceux des réseaux sociaux en ligne. Devant des objets de cette taille et complexité, il devient difficile pour un être humain de saisir la structure du réseau — ou même de le représenter !

Pour exhiber cette structure, une notion pertinente est celle de communauté ; une communauté, c'est un ensemble de points qui jouent le même rôle au sein du réseau, que ce rôle soit donné par une information supplémentaire au graphe lui-même ou qu'il faille le lire de l'ensemble des points. Répartir ainsi le graphe nous donne non seulement une meilleure compréhension, mais nous aussi l'accès à d'autres outils. En effet, cette condensation de l'information nous permet de mieux traiter des questions comme la diffusion dans le réseau ou la recommandation d'éléments. C'est donc la partition de réseaux en communautés qui sera l'objet de notre projet.

Toute une littérature existe déjà pour apporter des réponses ; dans une première partie, nous ferons un résumé succinct de la manière dont ce problème est traité aujourd'hui, avant de remettre en question une hypothèse fondamentale de la majorité de cette littérature. Nous tenterons ensuite d'introduire une mesure de qualité spécifiquement adaptée au problème de la recommandation, puis conclurons avec l'étude d'un réseau sur lequel a travaillé notre tuteur, Laurent Massoulié, que nous remercions ici pour son temps et sa pédagogie.

2 État de l'art

2.1 Notion de communauté

Intuitivement, une communauté est un sous-graphe du réseau qui est à la fois dense en liens internes et peu relié au reste du graphe. Pour formaliser cette définition, deux notions imbriquées existent [9] :

- Un sous-graphe est une communauté *forte* si chacun de ses points a plus de liens avec l'intérieur qu'avec l'extérieur
- Un sous-graphe est une communauté *faible* si chaque point a plus de liens avec l'intérieur qu'avec n'importe quelle autre communauté.

Cette première définition tient compte des liens existants entre points du réseau ; cependant, si on considère un réseau existant comme la réalisation aléatoire d'un modèle de graphe donné, l'information est contenue dans la *probabilité* de chaque lien. Cependant, il est dans les faits difficile d'obtenir ce modèle qui représente la nature parfaitement. Pour tester différentes méthodes, il est donc fréquent de recourir au *Stochastic Block Model* ou SBM. Ce modèle est construit de la manière suivante :

- Un ensemble de N points est partitionné en k blocs B_i
- On choisit une matrice P de taille $k \times k$
- Chaque couple de points $(x \in B_i\,, y \in B_j)$ a une probabilité $P_{i,j}$ d'être relié.

Quand la majorité du poids de P se trouve sur la diagonale, chaque bloc ainsi implanté représente une communauté ; c'est le cas dit *assortatif*. Cependant, d'autres cas existent : si la majorité du poids est entre les blocs et que ces derniers ont donc plus de liens externes, la configuration est dite *disassortative* (figure 2.1).

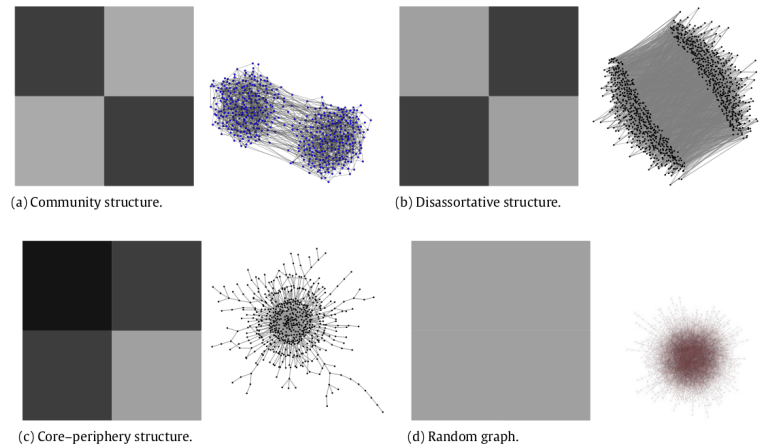


Fig. 2.1 : Structure des graphes : assortativité et disassortavtivité.

Une notion générale de communauté, qui s'étend à ces cas, est celle d'ensemble de points se comportant de la même manière et pas forcément particulièrement liés entre eux : on peut

par exemple penser à un graphe bipartite proies/prédateurs. Pour vérifier la pertinence d'une méthode, plutôt que de vérifier si elle produit des communautés au sens donné ci-dessus, on peut tester si elle retrouve la structure des graphes SBM. De plus, quand l'on dispose d'un a priori sur le graphe (par exemple la *ground truth* des communautés partitionnées par d'autres critères), on peut vérifier si les communautés obtenues ont un sens dans la réalité. Ce sont là les principaux *benchmarks* par lesquels un algorithme est évalué :

- Retrouve-t-il la structure des communautés plantées par le Stochastic Block Model ?
- Donne-t-il des résultats cohérents sur des graphes avec une *ground truth* a priori des communautés ?

Deux grandes catégories rassemblent la majorité des méthodes : le clustering spectral et l'optimisation de modularité.

2.2 Clustering spectral

Les méthodes qui tombent dans cette catégorie utilisent les propriétés spectrales du graphe. Typiquement, le spectre de nombreuses matrices associées au graphe consiste d'une masse de valeurs propres densément réparties, ainsi que de quelques valeurs isolées, plus éloignées. Souvent, il est possible d'exhiber un lien entre les vecteurs correspondant à ces valeurs propres et les propriétés de structure du graphe, comme la structure en communautés. Chaque coordonnée des vecteurs propres correspondant à un point du graphe, en choisissant d'isoler k vecteurs propres particuliers, on peut projeter l'ensemble des points du graphe dans $\mathbb{R}^k$ Dans cette représentation, la i-ème coordonnée du j-ème vecteur propre est la j-ème coordonnée dans $\mathbb{R}^k$ du point correspondant au i-ème point du graphe. Une fois ce travail effectué, des méthodes classiques permettent de clusteriser le nuage de points obtenus car on est maintenant dans le cadre d'un espace vectoriel. *In fine*, le grand avantage de ces méthodes est d'employer des outils mathématiques préexistants et légers en temps de calcul : la décomposition en vecteurs propres de matrices en général symétriques, et le clustering de points dans $\mathbb{R}^k$.

Les premières matrices pour lesquelles cette méthode a été développée sont :

- La matrice d'adjacence, dont la case $i\,, j$ vaut 1 s'il y a un lien entre les points i et j et 0 sinon [16]
- Le Laplacien, dont la case $i\,, j$ vaut $\text{degré}(i)$ si $i = j$, -1 s'il y a un lien entre les points i et j et 0 sinon.

C'est avec notre implémentation du clustering par matrice d'adjacence que nous obtenons le résultat présenté à la figure 2.2.

Cependant, les méthodes spectrales qui leur sont associées ont le problème de ne pas saturer la limite théorique de détection de communautés [13] : en effet, des arguments de physique statistique sur le Stochastic Block Model montrent que quand le ratio entre le degré externe (la probabilité pour un point de former un lien avec un point extérieur à son bloc donné) et le degré interne (la probabilité de former un lien avec un point intérieur) est trop proche de 1 dans un sens ou dans l'autre, il est impossible d'inférer la structure du graphe [4]. La version la plus poussée du clustering spectral, qui retrouve les communautés jusqu'à la limite théorique, utilise la Hessienne de l'énergie libre de Bethe, une quantité qui représente l'énergie libérable en séparant le graphe, donc une matrice symétrique positive. Si cette matrice a toutes ses valeurs propres positives, c'est que le graphe est « stable » et non fissionnable ; sinon, c'est que le vecteur propre correspondant à la valeur négative pointe dans la direction de points qu'on peut enlever pour former leur propre cluster [33].

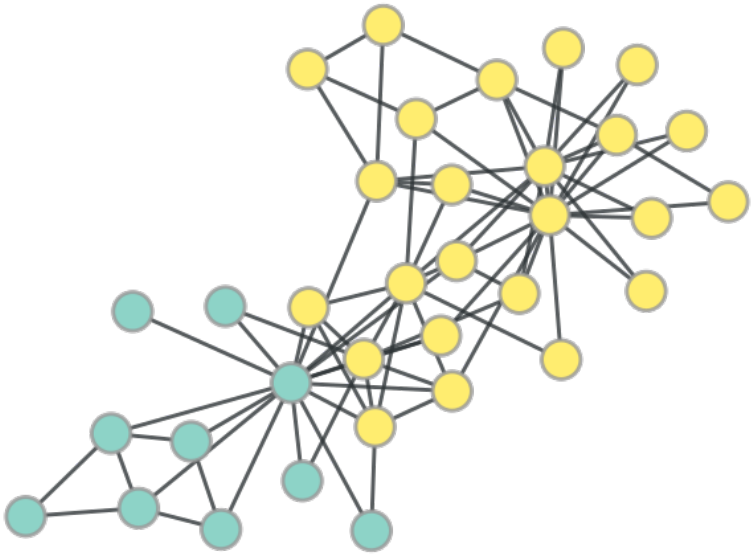


FIG. 2.2 : Clustering spectral par matrice d'adjacence sur un exemple de réseau.

2.3 Clustering par modularité

Ces méthodes consistent à optimiser une quantité appelée modularité qui représente, pour une partition donnée, le nombre d'arêtes supplémentaires à l'intérieur des groupes par rapport la situation ou les arêtes sont réparties aléatoirement en conservant les degrés de chaque point. Plus la modularité est grande, plus la structure du graphe est organisée, par opposition à un graphe aléatoire. L'expression générale de la modularité, qui consiste en une somme sur les couples de points i et j du graphe, chacun appartenant à un bloc C_i et C_j est [25] :

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta(C_i, C_j)$$

Avec m le nombre d'arêtes du graphe, A la matrice d'adjacence et P une matrice qui représente la matrice d'adjacence moyenne d'un graphe avec ces caractéristiques. En général, on choisit $P_{ij} = \frac{k_i k_j}{2m}$ (la probabilité qu'il y ait une arête entre deux points de ce degré) ce qui se nomme le modèle de configuration et donne [25] :

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j)$$

Optimiser la modularité directement sur toutes les partitions possibles du graphe est difficile computationellement ; pour cette raison, l'heuristique la plus utilisée est celle dite de Louvain [2]. Cette dernière consiste en le processus suivant :

- On part de la partition du graphe en singletons.
- Pour chaque point, on considère si le mettre dans la communauté d'un de ses voisins augmente la modularité ; si oui, on le met dans celle du voisin qui maximise la modularité. On passe ensuite au point suivant dans une liste arbitraire.

- Une fois la liste parcourue, on considère s'il existe un changement de ce type qui augmente la modularité. Si oui, on recommence.

Les avantages de cette approche (nommément, son efficacité sur de très grands réseaux, la possibilité de réeffectuer un clustering du graphe induit par les communautés afin d'avoir un clustering hiérarchique à plusieurs niveaux (figure 2.3), et son succès sur les réseaux à *ground-truth* de test habituels) en ont fait l'algorithme standard de recherche de communautés [9].

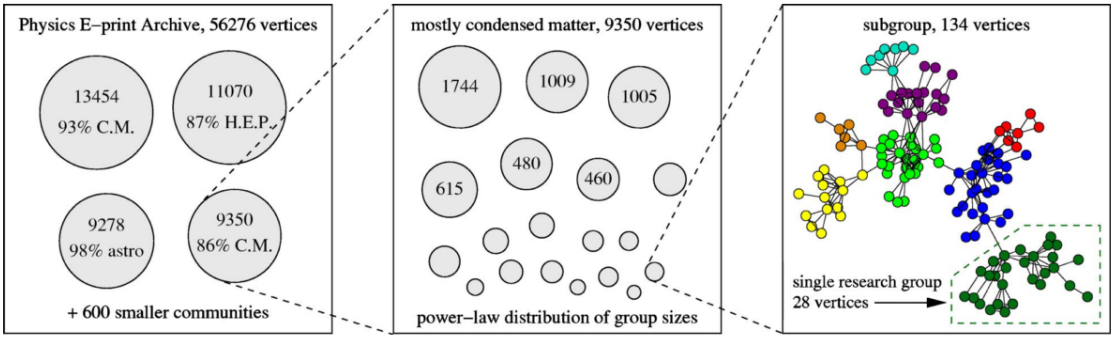


FIG. 2.3 : Clustering hiérarchique d'un réseau de collaboration de physiciens.

Cependant, la modularité souffre d'un problème de résolution dû au choix de P : dans les cas où celui-ci est uniformément petit devant 1, par exemple si pour la plupart des points i du graphe $k_i << \sqrt{m}$ dans le modèle de configuration, la modularité perd en précision et est parfois plus haute pour des associations de communautés qui sont contre-productives [2].

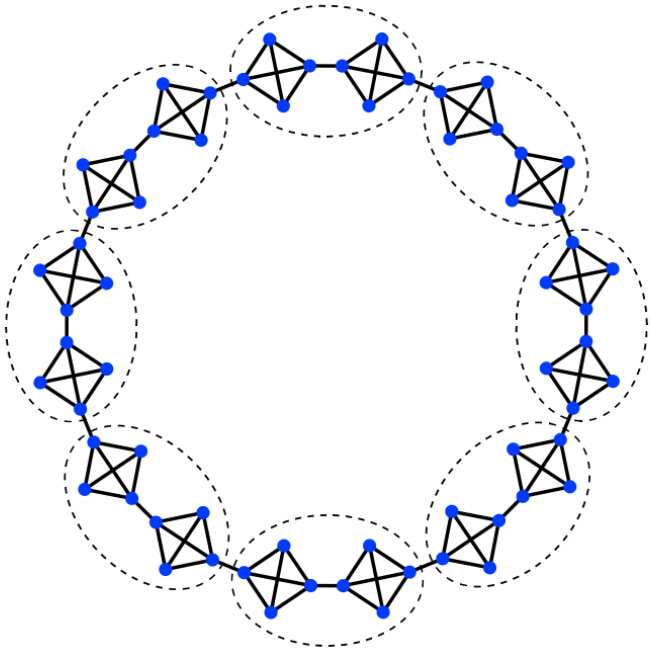


FIG. 2.4 : Un exemple où la modularité peut échouer : le cercle de cliques.

3 Le Stochastic Block Model est-il proche de modèles réels ?

3.1 Justesse de l'approximation par un SBM donné

L'intégralité des articles que nous avons consultés pour se documenter sur des méthodes les a testées d'abord de manière systématique sur des graphes SBM pour en vérifier la pertinence, puis compare avec la *ground truth* de graphes supervisés. Cette approche pose deux problèmes : d'abord, le fait qu'une méthode puisse fonctionner sur les SBM mais ne pas donner exactement la *ground-truth* de graphes réels montre bien que ce modèle n'est pas universel. Ensuite, s'il ne l'est pas, le fait de construire les algorithmes de manière à ce qu'ils le traitent bien n'est-il pas une forme d'*overfitting* dangereuse ?

Pour effectuer cette vérification sur un graphe qui pourrait ne pas ressembler à un SBM, nous allons utiliser une méthode que nous n'avons pas présentée dans la partie précédente, car peu utile dans la pratique. C'est l'inférence statistique : étant fixé un modèle de graphes (par exemple, le Stochastic Block Model), en supposant qu'un graphe donné fait partie de cette famille, on peut inférer les paramètres du modèle particulier qui l'a construit. Une première approche est de vérifier sur des graphes réels si cette approximation correspond bien à la *ground truth* ; cependant la *ground truth* ne correspond pas toujours à la structure topologique du graphe, et même en testant sur un graphe SBM, donc pour lequel on s'attend à de bons résultats, l'inférence peut trouver un modèle complètement différent qui soit tout aussi valable si on est dans la zone évoquée par [4] où il est impossible de remonter aux communautés implantées.

Pour ces raisons, nous introduisons une mesure d'à quel point un Stochastic Block Model est proche de la topologie du graphe. Pour un couple i , j de blocs du modèle observés, le SBM est un bon modèle à deux conditions : d'abord, s'il arrive à bien approcher la probabilité effectivement observée, et ensuite, s'il arrive à former des blocs assez grands qui vérifient toujours la condition de probabilité. Pour prendre les deux en compte, nous partons de l'observation que la probabilité observée est en fait la réalisation d'une grande moyenne ($n_i \times n_j$ termes) de variables de Bernoulli indépendantes. Il y a donc un écart d'ordre maximal$\sqrt{n_i n_j}$ entre la valeur observée et la valeur réelle de d_{ij} En additionnant cette incertitude et l'écart réel, on arrive à la formule suivante :

$$\text{Loss}_{ij} = \left| d_{ij} - \hat{d_{ij}} \right| + \frac{1}{\sqrt{n_i n_j}}$$

La figure 3.1 décrit l'écart des blocs i et j à leur probabilité du SBM, et l'incertitude si celui-ci a construit de trop petits clusters. Cependant, ce modèle a le gros défaut de devoir d'abord inférer un Stochastic Block Model du graphe à tester. Est-il possible d'observer la structure du graphe directement pour savoir s'il est possible qu'il ressemble à n'importe quel graphe SBM ?

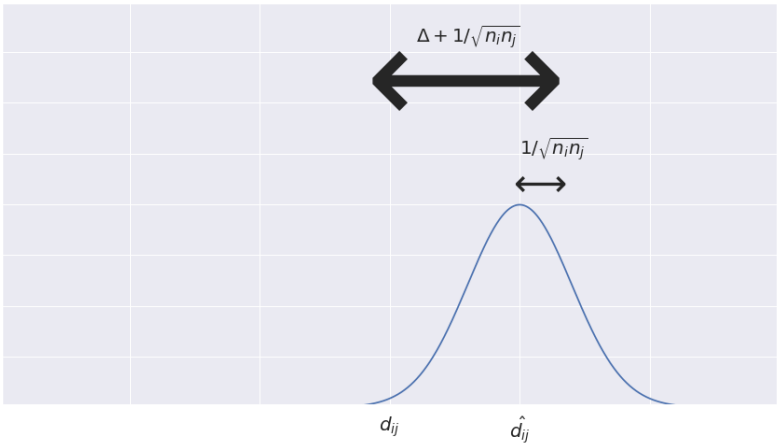


FIG. 3.1 : Somme de l'erreur et du risque.

3.2 JUSTESSE DE LA RESSEMBLANCE AVEC UN SBM EN GÉNÉRAL

Le travail de la partie précédente nous a quand même permis d'observer un phénomène typique des Stochastic Block Models : en effet, pour n'importe quel point donné, la variable aléatoire qui régit son degré est une somme de variables de Bernoulli indépendantes ; en fait, on a pour $(B_i)_{1,\dots,k}$ les blocs du SBM et $x \in B_i$:

$$\deg(x) = \sum_{j=1}^{k} X_j$$

Avec les X_i des variables binomiales de probabilité d_{ij} et de nombre de réalisations $|B_j|$.

En particulier, la partie extrême de la répartition des degrés doit suivre une loi de cette type. Elle est donc nécessairement à décroissance exponentielle dans des grands graphes, quand à la fois on tire un très grand nombre de réalisations et que le théorème central limite s'applique. Un graphe dont la distribution des degrés est dite *heavy-tailed* c'est-à-dire à décroissance moins rapide qu'exponentielle est donc très éloigné du comportement d'un graphe SBM.

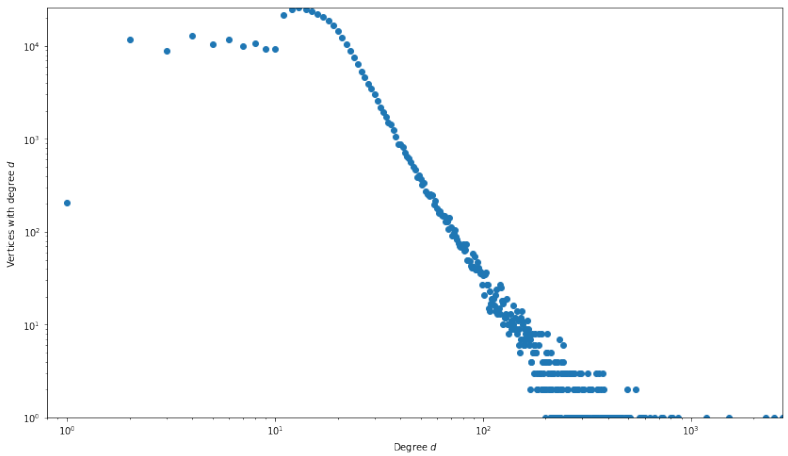


FIG. 3.2 : Distribution des degrés d'un dataset de relations entre produits d'Amazon.

Il est tentant de faire passer une droite par la courbe de la figure 3.2 ; cependant, le problème est qu'elle est extrêmement bruitée à l'extrémité, et ne tient pas compte des valeurs nulles qui

devraient se trouver infiniment basses. Pour résoudre ce problème, nous avons calculé à la place la fonction de répartition des degrés F ; si elle suit une loi de puissance, alors la densité d aussi (figure 3.3).

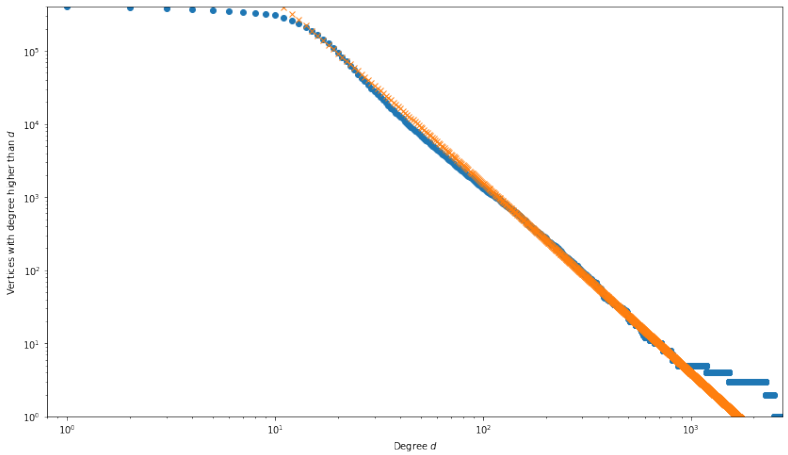


FIG. 3.3 : Fonction de répartition des degrés.

La droite fittée en orange correspond presque exactement à la partie rectiligne de la courbe, ce qui suggère une loi de puissance pour la densité (d'exposant très proche de -3). On remarquera d'ailleurs que son extrémité à droite est convexe : la distribution des degrés décroît asymptotiquement encore moins vite qu'une loi de puissance ! Finalement, sur 10 graphes testés (trois petits graphes standard chargés par défaut dans notre outil d'analyse, *graph-tool*, le grand graphe de relations entre produits d'Amazon et un de ses sous-ensembles, un graphe que nous avons tiré nous-mêmes de données Youtube et un de sous-ensemble, un graphe portant sur les produits et un autre sur les utilisateurs du site de vente en ligne Prestashop, et le graphe des emails d'un centre de recherche de l'UE) seuls les trois petits graphes pré-chargés ont passé ce test ; il semblerait que la plupart des graphes de grande taille (> 100 000 noeuds) que l'on trouve dans la nature soient de distribution des degrés *heavy-tailed* et donc incompatibles avec le Stochastic Block Model.

4 Validation croisée

Afin de qualifier une méthode, nous avons vu à la section 2 qu'il existait de nombreuses fonctions de qualité, telles que la modularité, l'overlap, etc. Cependant, une technique classique en Machine Learning, la validation croisée *(cross-validation)* semble assez peu représentée dans la littérature.

Nous avons alors implémenté la validation croisée dans un but de comparaison de méthodes de clustering, et afin de déterminer en particulier la validité du Stochastic Block Model.

4.1 Validation croisée en tant que mesure de qualité

4.1.1 Méthode générale et illustration

La première validation croisée que nous avons implémentée consiste à masquer une partie des arêtes du graphe. Le nouveau graphe, qui a donc le même nombre de sommets mais moins d'arêtes, est ensuite réparti en communautés en utilisant une technique existante (clustering spectral, modularité, minimisation de Block Model par exemple).

Pour extraire une mesure de qualité, nous prenons ensuite les arêtes supplémentaires. Chaque arête dont les deux extrémités sont dans un même bloc contribue positivement à la mesure, tandis que les autres ne contribuent pas. On parvient ainsi à extraire une mesure de la capacité du clustering de réunir dans ses blocs la plupart des relations du réseau.

Prenons pour exemple un réseau de matchs de football américain universitaire durant la saison 2000 [5]. Le graphe initial est représenté en figure 4.1.

Nous choisissons ensuite de conserver 90% des arêtes dans le graphe « train », les arêtes restantes constituant le graphe « test » (figures 4.2a et 4.2b).

Ensuite, il suffit d'appliquer la méthode de clustering choisie au réseau « train », par exemple ici une minimisation de Block Model (figure 4.3).

On obtient enfin une qualité de 0.726.

4.1.2 Validation k-fold

Cette première version est très sensible au choix initial des graphes « train » et « test ». En effet, la proportion d'arêtes retirées est souvent faible, et le choix initial aléatoire rend la mesure de qualité finale inutile.

Pour pallier à ce problème, la solution classique consiste à employer une validation en k plis *(k-fold cross-validation)*. Celle-ci subdivise l'ensemble de départ (ici, les arêtes du graphe initial), en une partition à k éléments, où k est un entier positif. On considère ensuite chacun des segments tour à tour comme l'ensemble « test », en calculant donc la mesure de qualité k fois. La mesure de qualité finale sera la moyenne de toutes les qualités intermédiaires calculées.

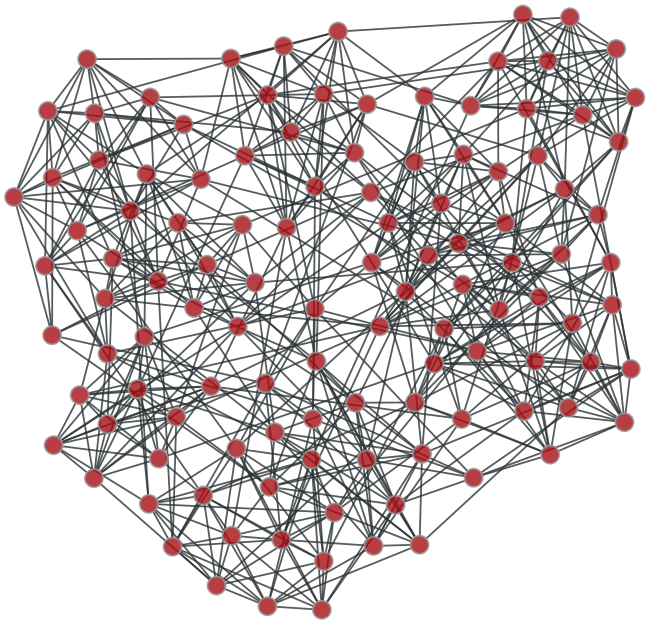
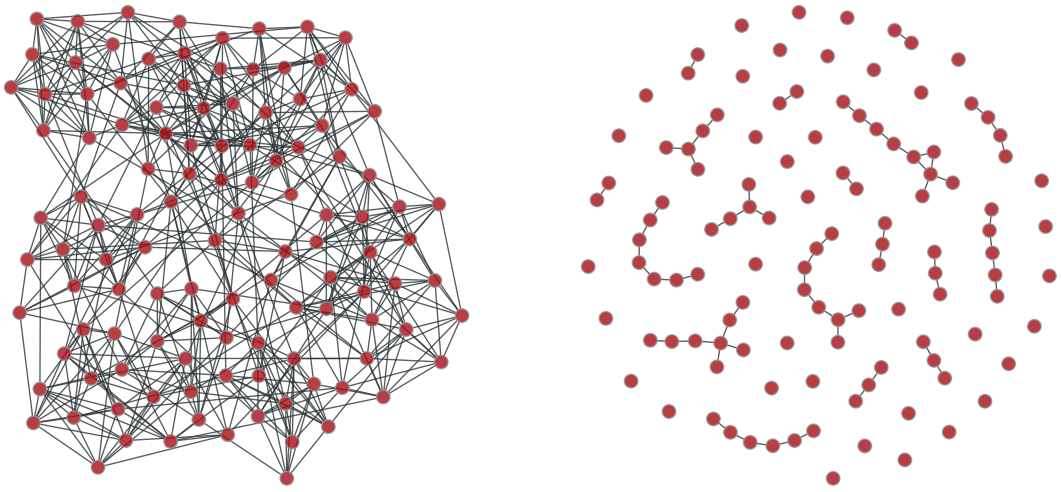


FIG. 4.1 : Réseau de matchs de football américain universitaire durant la saison 2000.



(a) Training Graph (b) Testing Graph

FIG. 4.2 : Réseau « train » et « test » extraits du réseau « football ».

FIG. 4.3 : Minimisation de Block Model sur le réseau « football train ».

4.1.3 IMPLÉMENTATION

```
def quality_crossval(g, a, verbose=False):
    """Applies a cross-validation to the graph g and computes a quality
    measure from it.  The array a should contain 1 on test edges, and
    0 on train edges.
    """
    n = g.num_edges()
    if (verbose):
        print("Initial graph number of edges:", n)
    # training graph
    filt_train = g.new_edge_property("bool")
    filt_train.a = a.astype(int)
    g_train = gt.GraphView(g, efilt=filt_train)
    g_train = gt.Graph(g_train, prune=True)
    if (verbose):
        print("Training graph number of edges:", g_train.num_edges())
    # minimize the blockmodel on the training graph
    if (verbose):
        print("Minimizing the blockmodel on the training graph...")
    state = gt.minimize_blockmodel_dl(g_train)
    # testing graph (remaining edges)
    filt_test = g.new_edge_property("bool")
    filt_test.a = (1 - a).astype(int)
    g_test = gt.GraphView(g, efilt=filt_test)
    g_test = gt.Graph(g_test, prune=True)
    if (verbose):
        print("Testing graph number of edges:", g_test.num_edges())
    # compute the quality
    if (verbose):
        print("Computing quality score...")
    quality = 0
    for e in g_test.get_edges():
        # if e[0] and e[1] are in the same block, quality++
```

```
33          blocks = state.get_blocks().a
34          if (blocks[e[0]] == blocks[e[1]]):
35              quality = quality + 1
36      quality = quality / g_test.num_edges()
37      if (verbose):
38          print("Quality score =", quality)
39      return quality
40
41
42  def kfold(g, k, verbose=False):
43      """Applies k-fold cross-valdiation to graph g."""
44      if (verbose):
45          print(k, "-fold cross validation on graph", g)
46      n = g.num_edges()
47      a = np.zeros(n)
48      for i in range(n):
49          a[i] = i % k
50      q = 0
51      for i in range(k):
52          q = q + quality_crossval(g, (a != i).astype(int))
53      q = q / k
54      return q
```

4.1.4 Résultats

Pour le graphe « football », nous avons calculé la qualité de validation croisée pour k variant de 1 à 20.

Nous avons ensuite testé la même méthode pour un graphe aléatoire suivant le Stochastic Block Model, avec 10 blocs, une probabilité d'arête interne de 0.999 et externe de 0.001 (figure 4.4).

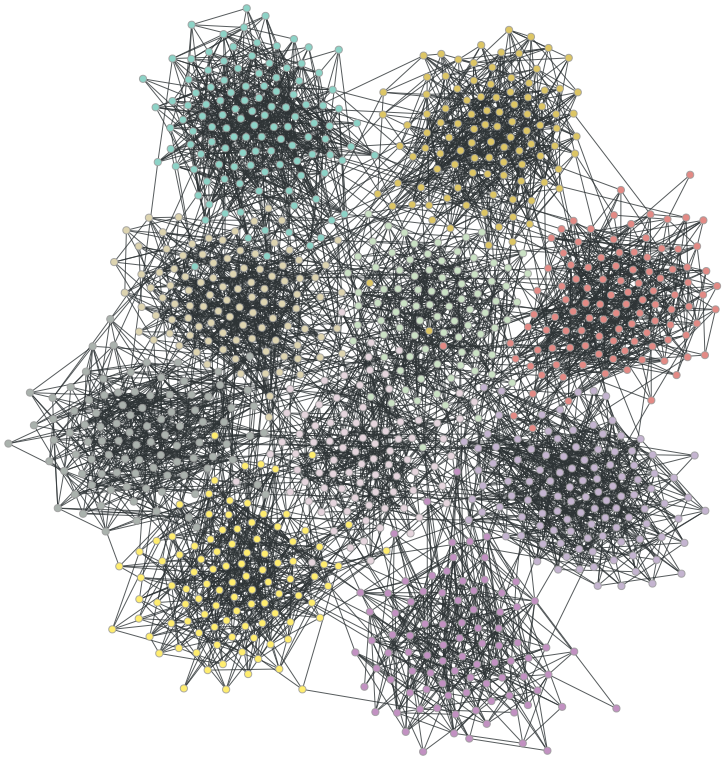


Fig. 4.4 : Stochastic Block Model, 10 blocs, probabilité interne 0.999 et externe 0.001.

4.2 Comparaison de méthodes par validation croisée

Il est possible d'exploiter la méthode de validation croisée pour comparer différentes méthodes de clustering. Ceci permettrait en effet de mesurer la robustesse de différentes méthodes lorsque l'on retire de l'information au graphe (en retirant des arêtes).

Pour cela, nous avons sélectionné deux méthodes récentes déjà implémentées dans des bibliothèques standard en Python :

- la méthode de Louvain [2], qui est une méthode d'optimisation de la modularité. Cet algorithme a l'avantage d'avoir une complexité assez faible en comparaison avec d'autres méthodes (de l'ordre de $\mathcal{O}(n\log(n))$). Cette méthode est implémentée en Python dans la bibliothèque `community`[1].
- la minimisation de Block Model [31]. Cette méthode consiste à minimiser l'entropie d'un Block Model sur le graphe, et a une complexité en $\mathcal{O}(n\log^2(n))$. Cette méthode a également l'avantage d'être implémentée de manière très efficace (parallélisée) dans la bibliothèque `graph-tool`.

Nous avons considéré des graphes aléatoires (Stochastic Block Models) et des graphes réels pour comparer ces deux algorithmes grâce à la validation croisée. Dans ce tableau, `football` est le réseau de matchs de football américain vu précédemment, `email-enron` est le graphe des communications par email chez Enron (rendu public lors d'une enquête judiciaire) [14], et `condmat` est un réseau de collaborations scientifiques par des chercheurs en matière condensée (récolté en 2005) [30].

Les résultats sont présentés dans la table 4.1. Le réseau SBM 1 est un Stochastic Block Model de taille 1000, avec une probabilité d'arête interne à un cluster de 0.99 et externe de 0.001. Le SBM 2 est similaire, mais de taille 2000 avec des probabilités interne et externe de 0.9 et 0.1 respectivement.

Réseau	Méthode de Louvain	Minimisation de Block Model
`football`	0.7419	0.6774
`email-enron`	0.7517	0.2907
`condmat`	0.6995	0.5492
SBM 1	0.8431	0.8039
SBM 2	0.1188	0.8514

Tab. 4.1 : Résultats de la validation croisée sur des exemples de réseaux, avec les méthodes de Louvain et de minimisation de Block Model.

4.3 Validation croisée et *ground-truth*

Il peut être difficile d'établir la valeur d'une mesure de qualité, en particulier lorsque celle-ci provient d'une technique de validation croisée. En effet, plusieurs méthodes s'offrent à nous :

- comparer la mesure de qualité à d'autres mesures de qualité déjà établies. Cependant, chaque mesure est sensible à certaines spécificités des graphes, et ignore d'autres aspects. Par ailleurs, l'immense variété des modèles de graphes aléatoires ne permet pas d'obtenir un ensemble de réseaux et de clusters adaptés pour un *benchmarking* efficace. Le modèle établi le plus courant, le Stochastic Block Model, ne ressemble que très superficiellement aux réseaux réels (comme établi en section 3), et évaluer l'intérêt d'une mesure sur ce modèle seul risque de biaiser les mesures de qualité en faveur des graphes réels ressemblant le plus à un Stochastic Block Model.

1. Disponible sur https://github.com/taynaud/python-louvain/. Documentation : http://perso.crans.org/aynaud/communities/

- L'autre technique consiste à utiliser des réseaux réels, dont on dispose par avance d'un clustering « naturel ». Cette approche a deux inconvénients : d'une part, ces réseaux sont difficiles à trouver et à relever. Même lorsqu'il est possible d'étiqueter les sommets d'un graphe, il existe souvent plusieurs manières d'établir des clusters (on peut penser par exemple aux utilisateurs d'un réseau social : doit-on les catégoriser par centres d'intérêts, par lieu de résidence, par secteur professionnel ?). Le deuxième inconvénient provient de l'étude de la mesure de qualité elle-même : pour ce faire, il est nécessaire d'obtenir une grande diversité de graphes (en taille, en nombre de clusters, etc.). Ceci rajoute donc à la difficulté d'en trouver quelques-uns.

Dans cette étude, nous ne pouvons pas appliquer nos méthodes à de nombreux réseaux étiquetés (munis de *ground-truths*) directement. Nous pourrions donc nous restreindre à deux exemples de graphes étiquetés, librement disponibles sur Internet[2] [38].

- Le premier représente un sous-ensemble du réseau social de Youtube. Une arête représente un lien d'amitié entre deux utilisateurs. Ceux-ci peuvent par ailleurs créer des groupes autour d'intérêts communs, et ensuite se déclarer membre d'un ou plusieurs groupes. Ce réseau est constitué d'une seule composante connectée, où chaque utilisateur est membre d'un seul groupe (les groupes de moins de 3 membres ont été supprimés). Le réseau de Youtube contient alors 1 134 890 sommets et 2 987 624 arêtes.
- Le second réseau provient d'Amazon. Il a été récolté en utilisant la fonctionnalité « les utilisateurs qui ont acheté ceci ont également acheté cela ». Si un produit est fréquemment acheté avec un autre produit, une arête est tracée entre ces deux produits. La catégorie fournie par Amazon fait office de *ground-truth*. Le réseau a 334 863 sommets et 925 872 arêtes.

Ces graphes sont imposants et requièrent des puissances de calcul ainsi que des quantités de mémoire conséquentes. En voulant comparer la mesure de qualité issue de la validation croisée avec la *ground-truth*, nous nous sommes donc heurtés aux limites informatiques du matériel dont nous disposons. En effet, sur un ordinateur récent, adapté pour les tâches courantes en Data Science, avec un processeur de 3.6 GHz et 16 Go de RAM, les calculs de communautés dépassaient rapidement les limites de la mémoire disponible.

En fait, l'expérimentation nous a permis de réaliser que les limites informatiques étaient atteintes pour des réseaux qui atteignaient approximativement la taille du réseau Amazon ci-dessus (soit environ 300 000 sommets). Pour des réseaux de taille supérieure, seule les statistiques immédiates étaient possibles (comme la distribution des degrés par exemple), et ce malgré l'utilisation d'une bibliothèque fortement optimisée comme `graph-tool`[3].

2. *Stanford Large Network Dataset Collection* : `https://snap.stanford.edu/data/`
3. `https://graph-tool.skewed.de/`

5 Applications : Prestashop

5.1 Les données Prestashop

Prestashop[1] est un logiciel d'e-commerce vendu à des entreprises souhaitant mettre en place des boutiques en ligne sur leur site Internet.

Grâce à notre tuteur, Laurent Massoulié, et au partenariat entre l'Inria et Prestashop, nous avons pu obtenir l'accès à des données issues de ventes en ligne via leur logiciel.

5.1.1 Objectif de l'étude

Dans l'optique de Prestashop, il ne s'agit pas de faire de la détection de communautés pour des réseaux sociaux. Les techniques de *clustering* sont développées dans le but d'améliorer les algorithmes de recommandation.

Ceux-ci se traduisent, du point de vue de l'utilisateur, par une sélection d'articles mis en avant qui pourraient l'intéresser. Du point de vue du commerçant, il s'agit d'une forme de publicité ciblée.

Il existe de nombreux algorithmes de recommandation, dont la plupart, très simples, se basent sur les catégories évidentes, objectives, des produits.

Ici, l'objectif serait de complémenter ces méthodes existantes par de la détection de communautés. Il est important de remarquer que cette technique peut être appliquée aussi bien au réseau des utilisateurs qu'à celui des produits : on peut en effet recommander à un utilisateur donné des produits achetés par un utilisateur qui lui ressemble, ou au contraire, des produits similaires à ceux qu'il a déjà achetés.

5.1.2 Description des données

Les données sont constituées de deux tableaux :

- Le premier contient les informations du catalogue du magasin en ligne. Chaque ligne est un produit (ici des livres), et donne des informations sur :
  - l'ID du livre ;
  - son titre ;
  - sa catégorie (identifiée par un numéro) ;
  - une liste de « précisions », qui ajoutent des catégories supplémentaires, qui peuvent être de nature très différente (« Ouvrage broché », « Français », « Actes Sud », etc).
- Le deuxième contient les données des utilisateurs et de leurs transactions :
  - l'ID de l'utilisateur ;
  - l'ID de l'item (produit en vente) ;
  - la date ;
  - le type de transaction (« Purchase », « AddShopCart », « Click », « Recommendation-Click »).

1. http ://www.prestashop.com/

5.1.3 Construction des graphes

Nous avons construit plusieurs graphes à partir des données disponibles.

Le premier est basé sur le catalogue. Chaque sommet du graphe représente un produit, et nous avons tracé une arête entre deux produits si ceux-ci partageaient au moins une « Précision » en commun.

Les deux graphes suivants sont plus intéressants pour notre étude et sont fondés sur les données des transactions. En effet, dans l'optique d'un système de recommandation, il est intéressant de mettre en relation les utilisateurs et les produits afin de pouvoir catégoriser ces derniers. Il serait alors envisageable de proposer à un utilisateur des articles achetés par un autre utilisateur de sa catégorie. À l'inverse, si un acheteur a apprécié un livre, il sera sans doute intéressé par un article similaire, au-delà du critère évident de la catégorie arbitraire du magasin.

Le graphe des utilisateurs (noté `users` dans la suite) relie deux acheteurs s'ils ont acheté un article en commun. Il contient 8 943 sommets et 301 006 arêtes, lorsque l'on conserve seulement la plus grande composante connectée (il y a en effet de nombreux clients qui n'ont aucun article en commun avec les autres). Ce réseau est représenté à la figure 5.1.

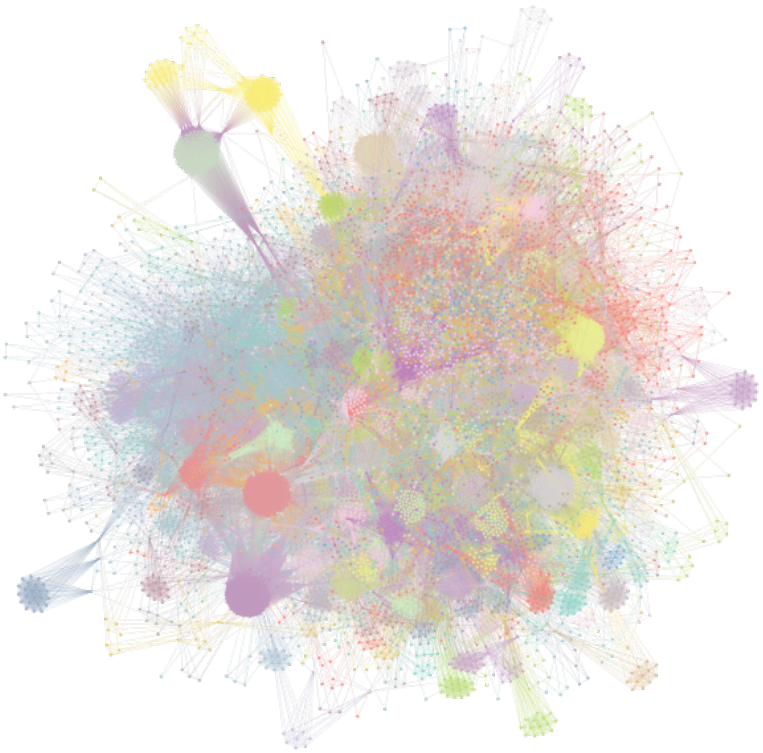


Fig. 5.1 : Graphe des utilisateurs. On observe de nombreuses cliques, qui correspondent à des articles achetés par un grand nombre d'utilisateurs, qui sont donc tous reliés entre eux.

Le graphe des produits (noté `items`) est similaire : deux produits sont reliés entre eux si au moins un acheteur les a achetés tous les deux. En ne conservant que la plus grande composante connectée (pour les mêmes raisons que précédemment), on obtient un graphe de 3 896 sommets et 41 824 arêtes, visible en figure 5.2.

Afin de créer ces graphes, nous avons parcouru la liste des produits (respectivement des utilisateurs), et ajouté toutes les paires d'utilisateurs (resp. de produits) qui ont ce produit (resp. utilisateur) en commun :

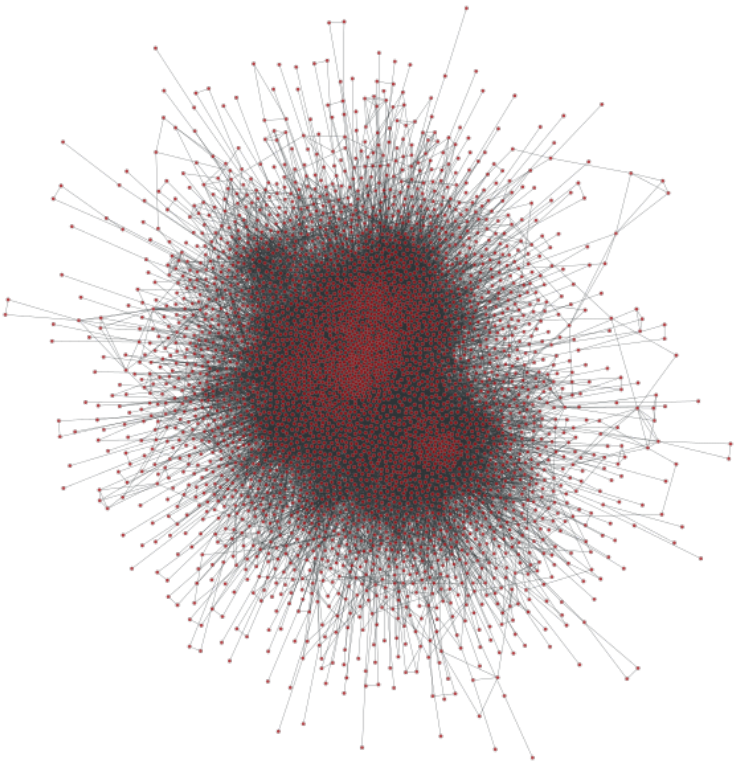


FIG. 5.2 : Graphe des produits.

```
users = []
for currentItem in purchases.ItemID.unique():
    g = purchases[purchases.ItemID == currentItem]
    users = users + list(itertools.combinations(g.UserID.unique(), 2))
users
```

Ceci renvoie une liste de paires de sommets, qu'on peut ensuite interpréter comme une liste d'arêtes, et importer facilement dans `graph-tool` ou `networkx`.

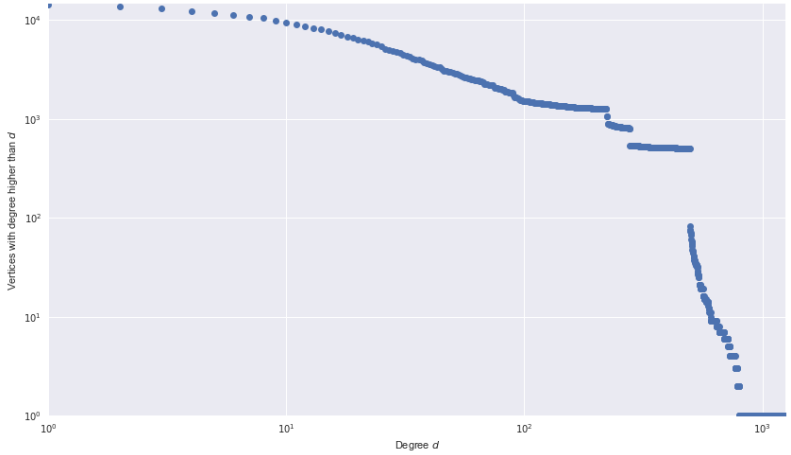
Notons que dans toute la suite, on a conservé uniquement la plus grande composante connectée de chaque réseau. Cela permet d'exclure les quelques dizaines d'utilisateurs ou de produits seuls ou en couple, qui perturbent les algorithmes de détection de communautés.

5.2 Étude des graphes de Prestashop

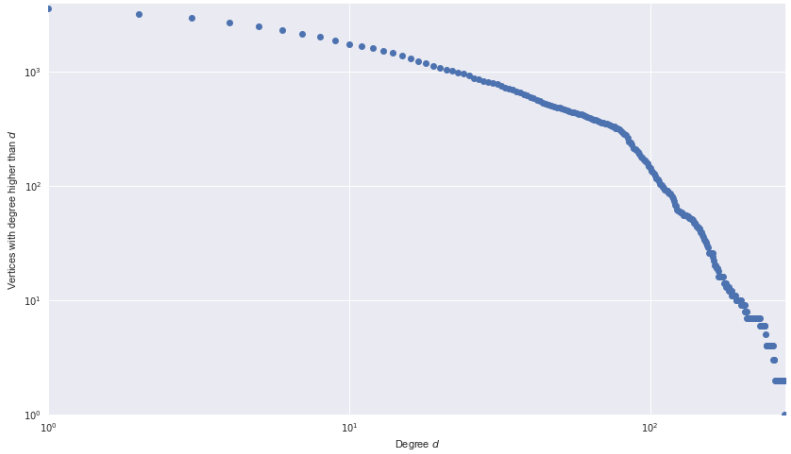
5.2.1 Répartition des degrés

Une représentation de la répartition des degrés des graphes Prestashop est donnée dans la figure 5.3.

On constate que les deux graphes ne suivent pas une distribution en *power law* classique, en particulier le graphe `users`, qui exhibe un comportement extrêmement atypique. Ceci provient des nombreuses composantes très fortement connectées et des cliques qui se sont formées autour de produits achetés par un très grand nombre d'utilisateurs. Celles-ci conduisent à des accumulations de sommets pour certains degrés, et à des ruptures de continuité dans la répartition.



(a) `users`



(b) `items`

FIG. 5.3 : Répartition des degrés des sommets des graphes de Prestashop.

5.2.2 APPLICATION DE LA VALIDATION CROISÉE

De la même manière qu'à la section 4.2, nous pouvons utiliser la validation croisée pour comparer les méthodes de Louvain et de minimisation de Block Model sur les graphes de Prestashop (table 5.1).

		`users`	`items`
Méthode de Louvain	Nombre de clusters	51	17
	Mesure de qualité	0.9369	0.6372
Minimisation de Block Model	Nombre de clusters	190	91
	Mesure de qualité	0.7163	0.3556

TAB. 5.1 : Résultats de la validation croisée sur les graphes de Prestashop, avec les méthodes de Louvain et de minimisation de Block Model.

On constate aisément que la technique de minimisation de Block Model a tendance à créer un trop grand nombre de clusters, ce qui rend le résultat extrêmement sensible à une information manquante. Cela peut s'expliquer par la structure très différentes des modèles de graphes aléatoires

traditionnels, en particulier pour le graphe `users`.

5.2.3 Détection de communautés chez les utilisateurs et les produits de Prestashop

L'analyse précédente montre que la méthode de Louvain est plus résistante au manque d'information. Comme les graphes de Prestashop sont par nature incomplets (manque de données collectées, faible échantillon d'utilisateurs et de produits, évolution constante des transactions), la méthode de Louvain serait donc plus adaptée pour une interprétation des communautés obtenues.

Les résultats pour les graphes `users` et `items` de Prestashop sont visibles en figure 5.4.

(a) `users` (b) `items`

Fig. 5.4 : Communautés dans les graphes de Prestashop, par la méthode de Louvain.

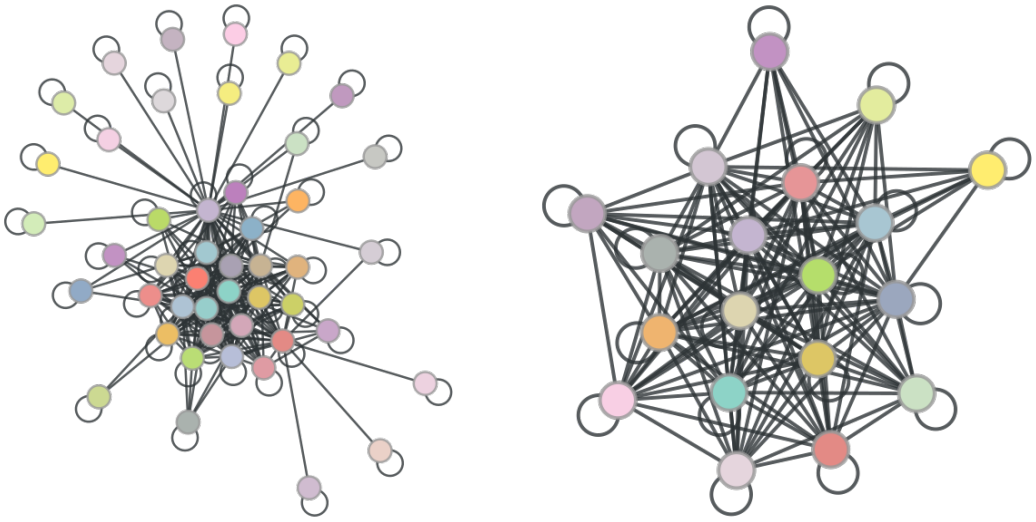
Ceci correspond à la répartition suivante :

Réseau	Nombre de clusters
`users`	46
`items`	19

Pour terminer l'étude de ce partitionnement, on peut déduire le graphe par blocs pour chacun des réseaux (figure 5.5).

On voit que la structure de ces deux graphes est extrêmement différente. Le réseau `users` possède une multitude de communautés de taille variée. Néanmoins, la plupart de ces clusters sont petits et reliés seulement faiblement au reste du graphe. On observe sur le graphe des blocs (figure 5.5a) un ensemble de blocs fortement connectés entre eux, qui concentrent à eux seuls la plupart des utilisateurs dans une « méta-communauté », tandis que les autres communautés en restent éloignées. Cela peut sans doute s'expliquer par le très grand nombre de cliques présentes dans le réseau, qui constituent à elles seules des petites communautés, qui peuvent n'être que très faiblement liées au reste du réseau.

Le graphe des produits, quant à lui, a une structure plus proche d'un graphe uniformément aléatoire (de type Erdös-Rényi). On retrouve ainsi assez peu de communautés, d'assez grande taille, et qui jouent des rôles similaires dans le graphe par blocs (figure 5.5b).



(a) `users` (b) `items`

FIG. 5.5 : Graphe des communautés pour les réseaux de Prestashop.

On voit donc que grâce à leur nature très différente, une même méthode de détection de communautés (ici Louvain) a permis d'exhiber des clusters très différents. Il serait alors intéressant d'utiliser cette disparité pour proposer des algorithmes de recommandation différents, suivant que l'on se place du point de vue de l'utilisateur ou du produit. Ainsi, une combinaison de ces deux graphes permettrait d'éviter les travers habituels des méthodes de recommandation, qui ont tendance à rester enfermées dans une catégorie particulière.

5.3 Conclusion sur le problème de la recommandation

Nous avons étudié dans ce projet l'intérêt des méthodes de détection de communautés, et en particulier de la validation croisée, pour le problème de la recommandation.

La validation croisée nous a permis d'évaluer différentes méthodes de clustering, et en particulier leur robustesse face à une perte d'information (sous la forme d'un manque de données sur les relations entre les différents éléments du réseau). Elle permet donc de comprendre que les différents algorithmes conduisent à des stratégies de recommandation distinctes. En particulier, cela va influer sur la capacité du système à recommander plus ou moins de produits en dehors de la catégorie classique de l'utilisateur.

Néanmoins, la validation croisée est-elle réellement une approche utile dans un objectif de recommandation ? Elle ne semble réellement utile que dans des cas extrêmes ou très limites, où l'absence d'information est la caractéristique limitante. Notre conclusion est en fait que c'est le cas des techniques de détection de communautés en général. Étant donné que celles-ci s'appuient uniquement sur la structure du graphe, sans a priori sur la nature des sommets et des arêtes, elles ne peuvent rivaliser avec un simple étiquetage, une *ground-truth*, des sommets du réseau.

La détection de communauté peut toutefois complémenter un système de catégorisation plus classique, quand on n'a pas d'autres informations, ou quand étiqueter les éléments devient irréalisable, ou encore quand une partie du graphe parasite le reste.

Les méthodes de clustering sont toujours assez opaques, et donc difficiles à interpréter. Or un système de recommandation, pour être efficace, a besoin de comprendre quels sont les points communs entre les éléments d'un même cluster.

Finalement, les problématiques de détection de communautés sont des exercices théoriques extrêmement intéressants pour la compréhension de la structure des réseaux qui nous entourent. Néanmoins, dans une optique de recommandation, l'étude des données de Prestashop nous a montré que l'intérêt résidait surtout pour complémenter un système de catégories classiques, par exemple pour favoriser les recommandations de produits avec plus de variété.

6 Conclusion

Ce projet a été pour nous l'occasion de nous familiariser avec une branche des mathématiques appliquées, la théorie des graphes, pour l'instant absente de notre cursus. Le problème de la détection de communautés en particulier, poussé par son intérêt fort à la fois pour le monde académique et l'industrie, a engendré une littérature très vaste et une multitude d'algorithmes ; sans prétendre en avoir fait le tour, nous pensons avoir une bonne vue d'ensemble du problème et nous espérons en avoir exploré des aspects nouveaux.

D'abord, en étudiant la validité de l'usage du Stochastic Block Model et en exhibant une condition suffisante pour qu'un graphe soit très différent de ce modèle, nous remettons en cause la manière de procéder de la littérature sur le sujet, c'est-à-dire d'abord étudier l'algorithme en cours de développement sur des graphes obtenus pas ce modèle et ensuite le tester sur des graphes tirés de problèmes réels. En effet, la grande majorité des graphes sociaux de grande taille que nous avons étudiés ne correspondent pas à ce modèle.

Ensuite, nous avons tenté, sur le conseil de notre tuteur, d'introduire la cross-validation comme méthode de vérification de la justesse d'une décomposition en communautés ; l'idée, venant d'autres domaines de l'apprentissage statistique, est d'estimer la valeur de la partition en communautés par sa capacité à prédire de nouveaux liens. Bien qu'elle soit prometteuse, nous avons dû abandonner cette idée car elle est impraticable sur des graphes de taille même raisonnable.

Enfin, nous avons souhaité appliquer ces techniques au problème de la recommandation posé par une société extérieure, Prestashop, qui opère un service de vente en ligne. La technique dite de Louvain a été un succès pour analyser la structure en communauté du graphe ; en exhibant et isolant de petits clusters très denses, elle permet d'effectuer des recommandations pertinentes sur ces clusters. Cependant, sur la composante plus diluée du graphe, en construisant des communautés de taille trop grandes pour être pratiques, les techniques de ce type sont moins efficaces. On remarque en cela qu'elles sont complémentaires avec des techniques de recommandation classique, comme le filtrage collaboratif, qui sont parasitées par les petits groupes qui captent la majorité de l'information. Une combinaison des deux semble donc être une approche prometteuse.

Bibliographie

[1] David Aldous and James Allen Fill. Reversible Markov Chains and Random Walks on Graphs. 2002 :1–516, 1999.

[2] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. 2008. 2.3, 2.3, 4.2

[3] Garrett Dash Nelson and Alasdair Rae. An Economic Geography of the United States : From Commutes to Megaregions. *Plos One*, 11(11) :e0166083, 2016.

[4] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 84(6), 2011. 2.2, 3.1

[5] Tim S Evans. Clique graphs and overlapping communities. *Journal of Statistical Mechanics : Theory and Experiment*, 2010(12) :P12037, 2010. 4.1.1

[6] Santo Fortunato. Quality functions in community detection. *Optimization*, 6601 :16–18, 2007.

[7] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5) :75–174, 2010.

[8] Santo Fortunato and Claudio Castellano. Community structure in graphs. *Computational Complexity : Theory, Techniques, and Applications*, 9781461418 :490–512, 2012.

[9] Santo Fortunato and Darko Hric. Community detection in networks : A user guide. *Physics Reports*, 659 :1–42, 2016. 2.1, 2.3

[10] M Girvan and M E J Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12) :7821–6, 2002.

[11] Prem K Gopalan and David M Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences of the United States of America*, 110(36) :14534–9, 2013.

[12] Lucas G S Jeub, Prakash Balachandran, Mason A. Porter, Peter J. Mucha, and Michael W. Mahoney. Think locally, act locally : Detection of small, medium-sized, and large communities in large networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 91(1) :1–29, 2015.

[13] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences of the United States of America*, 110(52) :20935–20940, 2013. 2.2

[14] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks : Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1) :29–123, 2009. 4.2

[15] L Lovász. Random walks on graphs : A survey. *Combinatorics Paul Erdos is Eighty*, 2(Volume 2) :1–46, 1993.

[16] Ulrike Von Luxburg. A Tutorial on Spectral Clustering A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(March) :395–416, 2006. 2.2

[17] Fragkiskos D. Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks : A survey, 2013.

[18] Travis Martin, Brian Ball, and M. E J Newman. Structural inference for uncertain networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 93(1), 2016.

[19] Morten Mørup and Mikkel N. Schmidt. Bayesian Community Detection. *Neural Computation*, 24(9) :2434–2456, 2012.

[20] M E Newman, S H Strogatz, and D J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64(2 Pt 2) :026118, 2001.

[21] M E J Newman. Mixing patterns in networks. *Physical Review E*, 67(2) :026126, 2003.

[22] M. E J Newman. Analysis of weighted networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 70(5 2), 2004.

[23] M. E J Newman. Fast algorithm for detecting community structure in networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 69(6 2), 2004.

[24] M. E J Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 74(3), 2006.

[25] M E J Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23) :8577–8582, 2006. 2.3

[26] M E J Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature Communications*, 7(May) :1–16, 2016.

[27] 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 2), 2004.

[28] M. E J Newman and Tiago P. Peixoto. Generalized Communities in Networks. *Physical Review Letters*, 115(8), 2015.

[29] M. E J Newman and Gesine Reinert. Estimating the Number of Communities in a Network. *Physical Review Letters*, 117(7), 2016.

[30] Mark EJ Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2) :404–409, 2001. 4.2

[31] Tiago P. Peixoto. Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2014. 4.2

[32] Federico Ricci-Tersenghi, Adel Javanmard, and Andrea Montanari. Performance of a community detection algorithm based on semidefinite programming. *Journal of Physics : Conference Series*, 699 :012015, 2016.

[33] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral Clustering of Graphs with the Bethe Hessian. *arXiv preprint arXiv :1406.1880*, (1) :1–8, 2014. 2.2

[34] Chuan Shi, Yanan Cai, Philip S. Yu, Zhenyu Yan, and Bin Wu. A Comparison of Objective Functions in Network Community Detection. *2010 IEEE International Conference on Data Mining Workshops*, pages 1234–1241, 2010.

[35] Amanda L. Traud, Eric D. Kelsic, Peter J. Mucha, and Mason a. Porter. Comparing Community Structure to Characteristics in Online Collegiate Social Networks. *SIAM Review*, 53(3) :17, 2008.

[36] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of Facebook networks. *Physica A : Statistical Mechanics and its Applications*, 391(16) :4165–4180, 2012.

[37] Adam Wierzbicki, Ulrik Brandes, Frank Schweitzer, and Dino Pedreschi. Advances in network science : 12th international conference and school, NetSci-X 2016 Wroclaw, Poland, January 11–13, 2016 proceedings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9564 :111–125, 2016.

[38] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 2015. 4.3