

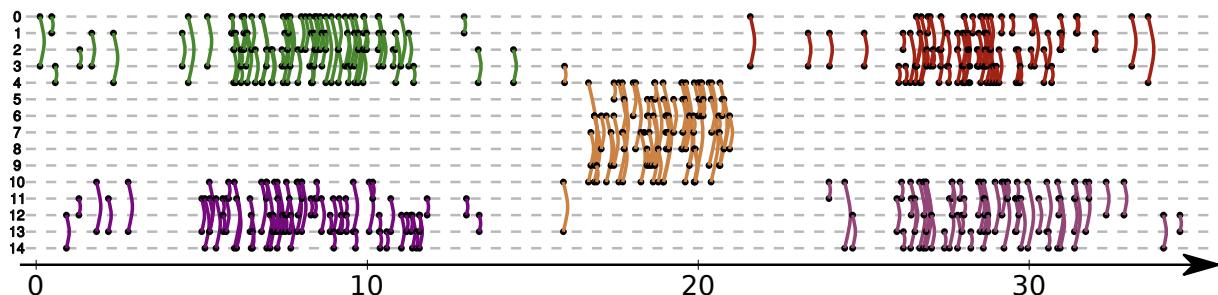
UNIVERSITY NAME

DOCTORAL THESIS

Groupes et Communautés dans les flots de liens: des données aux algorithmes

Auteur :
Noé GAUMONT

Directeurs :
Clémence MAGNIEN
Matthieu LATAPY



*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy
in the*

Research Group Name
Department or School Name

11 juillet 2016

Table des matières

Remerciements	ii
Introduction	1
1 État de l'art sur la détection de communautés et les réseaux dynamiques	3
1.1 Définition dans les graphes	4
1.1.1 Groupes, partitions et couvertures	4
1.1.2 Comparaison de partitions et couvertures	5
1.2 Communauté dans les graphes	7
1.2.1 Parititons de nœuds	7
Méthodes utilisant un modèle	7
Méthodes utilisant des marches aléatoires	10
1.2.2 Couverture de nœuds	10
Extension de méthodes existantes	11
Méthodes utilisant des communauté égocentré	12
1.3 Extension temporelle des graphes	13
1.3.1 Extensions avec pertes d'informations temporelles	14
Séries de graphes	14
Tenseur 3D	15
Graphes multicouches	16
1.3.2 Extension sans perte d'information temporelle	17
Graphe temporel	17
Flot de liens	18
1.4 Bilan	20
2 Flots de liens : extension temporelle des graphes	21
2.1 Définition	21
2.2 Sous-flots	22
2.3 Degré et densité	23
2.4 Liste des notations pour les flots de liens	25
2.5 Manipulation concrète des flots de liens	25
3 Étude de la structure d'une archive de courriels en tant que flot de liens	27
3.1 Prétraitement sur le jeu de données	27
3.2 Caractéristiques élémentaires des discussions	28
3.3 Étude des discussions en tant que sous-flots	30
3.3.1 Application de la Δ -densité	30
3.3.2 Répartition temporelle et structurelle des discussions	33
3.3.3 Flot quotient	35
3.3.4 Conclusion	36
3.4 Détection de structures denses	37
3.4.1 Méthode de détection	37
3.4.2 Comparaison des partitions	38
3.4.3 Conclusion	40

4 Détection de groupes pertinents dans les flots de liens	43
4.1 Travaux existants	44
4.2 Détection de groupes pertinent	45
4.2.1 Définition du voisinage	45
4.2.2 Définition de l'évaluation	47
4.2.3 Algorithme de calcul des scores	48
Voisinage au nœuds	48
Voisinage à la durée	49
Voisinage au temps de début	49
4.3 Jeux de données	50
4.4 Application	51
4.4.1 Évaluation des groupes candidats	52
Étude manuel d'un groupe de Socio Pattern	53
Étude manuel d'un groupde de Rollernet	54
4.4.2 Caractéristiques des groupes pertinents	55
Caractéristiques topologiques	56
Caractéristiques temporelles	57
4.5 Conclusion et perspective	57
4.5.1 Conclusion	57
4.5.2 Perspective	58
Amélioration de la détection des groupes pertinents	58
Amélioration de la validation des résultats	59
5 <i>Expected Nodes</i> : communautés de liens dans les graphes statiques	61
5.1 Travaux existants	62
5.2 Définition d' <i>Expected Nodes</i>	65
5.3 Comparaison	68
5.3.1 Cas du graphe complet	68
5.3.2 Graphe LFR	69
5.4 Calcul et optimisation	73
5.5 Conclusion	74
5.5.1 Perspective	75
Amélioration de l'optimisation d' <i>Expected Nodes</i>	75
Cas d'utilisation d' <i>Expected Nodes</i>	76
6 Vers des fonctions de qualité dans les flots de liens	79
6.1 Définition	79
6.2 Générateur de flots de liens avec structure communautaire	79
7 Conclusion	81
A Détection de groupes pertinents dans les flots de liens	83
A.1 Rollernet dataset	83
A.2 Baboon dataset	85
A.3 Reality mining dataset	86
Bibliographie	87

Introduction

Appuyer sur le fait que le temps est continu et que l'on veut garder toute l'information pour faire des descriptions super fines temporellement.

Chapitre 1

État de l'art sur la détection de communautés et les réseaux dynamiques

Sommaire

1.1	Définition dans les graphes	4
1.1.1	Groupes, partitions et couvertures	4
1.1.2	Comparaison de partitions et couvertures	5
1.2	Communauté dans les graphes	7
1.2.1	Parititons de nœuds	7
1.2.2	Couverture de nœuds	10
1.3	Extension temporelle des graphes	13
1.3.1	Extensions avec pertes d'informations temporelles	14
1.3.2	Extension sans perte d'information temporelle	17
1.4	Bilan	20

Dans cette thèse, nous étudions deux axes de recherches liés aux graphes qui sont orthogonaux. Mais avant de présenter ces deux axes de recherches, nous revenons dans la section 1.1 sur quelques définitions et notations utiles pour manipuler des graphes.

Le premier axe de recherche est la détection de structures dans les graphes et plus particulièrement de communautés. Une communauté est une partie du graphe de telle sorte qu'il existe beaucoup de liens à l'intérieur de la communauté et moins avec le reste du graphe. Il n'existe cependant aucune définition unique et exacte d'une communauté. La notion de communauté dépend du contexte et de la méthode. Malgré cette définition floue, des structures communautaires ont été trouvées dans de nombreux graphes dans plusieurs domaines tel que le réseau constitué des régions du cerveau [dRSKvdH14], un réseau de distribution d'eau [DDG⁺15] et un réseau d'interactions d'animaux [FW15]. Ces notions de communautés et les méthodes de détection sont définies dans la section 1.2.

Le second axe de recherche est la prise en compte du temps dans la théorie des graphes. La première approche fût de complètement ignorer l'information temporelle et de considérer que tout les liens apparaissent en même temps ; on parle alors de graphe agrégé. Cependant, il a été observé que la modélisation d'un réseau sous la forme d'un graphe agrégé peut ne plus être pertinente, notamment quand le réseau change trop au cours du temps [Hol15b]. Si tel est le cas, certaines structures présentes dans le graphe peuvent n'être que des artefacts de l'agrégation temporelle. Imaginons un graphe représentant les alliances politiques dans un pays sur une longue période. Si toutes les alliances politiques sont agrégées, alors les personnes changeant de parti politique seront faussement considérées comme influentes car connectées à plusieurs partis alors qu'elles peuvent ne plus avoir de contact dans leur ancien parti. Ainsi il n'est plus possible d'observer la répartition des alliances politiques dans ce genre de réseau si l'agrégation temporelle est trop importante [MRM⁺10].

Si on prend en compte le temps, alors il est possible de détecter des structures plus fine que dans le cas statique. En effet, il devient possible de détecter des instants où l'organisation générale change [RB10], de comprendre quels sont les instants où une personne est importante [MT15], ou bien de détecter des groupes temporels [CAH10]. Face à ces problèmes, plusieurs extensions de la théorie des graphes ont été proposées et elles sont présentées dans la section 1.3.

1.1 Définition dans les graphes

Un graphe G est défini par un couple (V, E) où V est un ensemble de nœuds et $E \subseteq V \times V$ est un ensemble de liens où chaque lien est une paire de nœuds. Sauf mention contraire, nous considérons des graphes non-orientés, c'est-à-dire pour toutes paires de nœuds $u, v \in V$, les liens (u, v) et (v, u) sont équivalents. Nous considérons également uniquement des liens non-pondérés, c'est-à-dire qu'un lien est soit présent soit absent. Enfin, nous ne considérons que des graphes simples, c'est-à-dire qu'il n'existe au maximum qu'un seul lien entre deux nœuds. Ceci est en opposition avec les graphes multiples où il peut exister plusieurs liens entre deux nœuds.

Par convention, nous notons $n = |V|$ le nombre de nœuds et $m = |E|$ le nombre de liens. Il est possible de représenter un graphe G par une matrice carrée A de taille n où l'élément $A_{i,j} \forall i, j \in \mathbb{N}^+ \leq n$ est égale à 1 si un lien relie les nœuds i et j et 0 sinon. Avec ces notations, nous définissons les notions suivantes :

Degré Le degré d'un nœud i , noté d_i , est égale au nombre de liens reliés à i : $d_i = \sum_{j \leq n} A_{i,j}$;

Densité La densité, $\delta(G)$ d'un graphe est la probabilité que deux nœuds soient reliés par un lien : $\delta(G) = \frac{2m}{n(n-1)}$;

Degré moyen Le degré moyen $\tilde{\delta}(G)$ diffère de la densité et est égale à : $\delta(\tilde{G}) = \frac{2m}{n}$;

Chemin Un chemin dans un graphe est une suite de liens $((u_1, v_1), \dots, (u_k, v_k))$ de telle sorte que $v_i = u_{i+1} \forall i \in [1, k-1]$;

Graphe connexe Les nœuds d'un graphe sont dit connexes s'il existe un chemin entre toutes les paires de nœuds du graphe ;

Composante Connexe Une composante connexe est un ensemble de nœuds $V' \subseteq V$ qui est connexe et maximal, c'est-à-dire qu'il n'est pas possible d'ajouter un nœud dans V' tel que V' soit toujours connexe ;

Arbre Un arbre est le graphe connexe le moins dense et il est constitué de $n - 1$ liens pour n nœuds ;

Clique Une clique, aussi appelée graphe complet, est le graphe le plus dense et elle est composée de $\frac{n(n-1)}{2}$ liens pour n nœuds ;

Sous-graphe Un graphe $G' = (V', E')$ est un sous-graphe de G si et seulement si $V' \subseteq V$ et $|E' \cap E| = |E'|$;

Graphe induit Le graphe induit par un ensemble de nœuds $V' \subseteq V$ est défini par V' et $E' = \{(u, v) \in E, u, v \in V'\}$.

1.1.1 Groupes, partitions et couvertures

En plus des nœuds et des liens, nous manipulons également des ensembles de liens et des ensembles de nœuds. Par convention, notons $X \subseteq V$ un ensemble de nœuds et $Y \subseteq E$ un ensemble de liens. Nous listons les notations utilisées dans le tableau 1.1.

Notation	Définition
$V(Y) = \{u, \exists(u, v) \in Y\}$	nœuds induits par les liens de Y
$n_X = X $	nombre de nœuds dans X
$l_{in}(X) = \{(u, v) \in E, u, v \in X\} $	nombre de liens entre les nœuds de X
$l_{out}(X) = \{(u, v) \in E, \cap(X \times V \setminus X)\} $	nombre de liens entre les nœuds de X et de $V \setminus X$
$l(X) = l_{in}(X) + l_{out}(X)$	nombre de liens reliés aux nœuds de X
$d_{in}(u, X) = \{(u, v) \in E, v \in X\} $	nombre de liens que partagent u avec X
$d_{in}(X) = \sum_{u \in X} d_{in}(u, X) = 2l_{in}(X)$	somme des degrés internes des nœuds dans X
$d_{out}(u, X) = \{(u, v) \in E, v \in V \setminus X\} $	nombre de liens que partagent u avec $V \setminus X$
$d_{out}(X) = \sum_{u \in X} d_{out}(u, X) = l_{out}(X)$	somme des degrés externes des nœuds dans X
$d(X) = d_{in}(X) + d_{out}(X)$	somme des degrés des nœuds dans X

TABLE 1.1 – Liste des notations utilisées pour un ensemble de nœuds X et un ensemble de liens Y .

Partitions et couvertures Nous définissons ici les structures de partitions et de couvertures appliquées au cadre spécifique des graphes. Une partition de nœuds, \mathcal{V} , est un ensemble d’ensemble de nœuds : $\mathcal{V} = \{V_1, \dots, V_k\}$ tel que $V_i \subseteq V \forall i \in [1, k]$. Une partition est soumise à deux contraintes :

1. La partition doit *recouvrir* l’ensemble des nœuds : $\bigcup_i (V_i) = V$.
2. Les ensembles de nœuds doivent être disjoints : $V_i \cap V_j = \emptyset \forall i, j \in [1, k], i \neq j$.

Afin de manipuler une partition, nous définissons $\mathcal{V}(u) = V_i$ si et seulement si $u \in V_i$.

Une couverture est une extension des partitions car elle relâche la contrainte sur l’intersection. Une couverture est parfois aussi appelée partition chevauchante. Ainsi, une couverture de nœuds, \mathcal{V} , est aussi un ensemble d’ensembles de nœuds. L’union des ensembles doit également être égale à V mais en revanche deux ensembles de nœuds peuvent partager un nœud : $\exists i, j \in [1, k], i \neq j \text{ } V_i \cap V_j \neq \emptyset$.

Nous avons détaillé les notions de partitions et couvertures de nœuds mais les même définitions valent pour les partitions et couvertures de liens.

1.1.2 Comparaison de partitions et couvertures

Il est intéressant de pouvoir comparer deux partitions ou deux couvertures entre elles. Le but est de calculer une similarité entre deux structures de telle sorte que la similarité est égale à 1 si les deux structures sont identiques et qu’elle soit égale à 0 ou -1 si elles sont complètement différentes. Il existe pour ce faire des méthodes tirant parti de la structure d’ensembles et celles provenant de la théorie de l’information.

Approches ensemblistes Il est possible de comparer deux ensembles X et Y en utilisant l’indice de Jaccard : $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. Avec l’indice de Jaccard, il est possible de mesurer la similarité entre deux partitions \mathcal{X} et \mathcal{Y} :

$$sim(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \max_{Y \in \mathcal{Y}} J(X, Y). \quad (1.1)$$

Avec cette formulation, la similarité n'est pas symétrique. C'est pourquoi la formule suivante lui est souvent préférée :

$$\text{sim}_{\text{moy}}(\mathcal{X}, \mathcal{Y}) = \frac{\text{sim}(\mathcal{X}, \mathcal{Y}) + \text{sim}(\mathcal{Y}, \mathcal{X})}{2} \quad (1.2)$$

Il existe d'autres méthodes pour symétriser la similarité, *e.g.* la moyenne harmonique. Cette méthode peut s'appliquer indifféremment aux partitions et aux couvertures.

Il existe également le *Rand Index* [Ran71] qui lui ne s'applique qu'aux partitions. Il mesure le nombre de paires de nœuds qui sont classées de la même manière dans les deux partitions ; c'est à dire pour deux nœuds u et v soit $\mathcal{X}(u) = \mathcal{X}(v)$ et $\mathcal{Y}(u) = \mathcal{Y}(v)$ soit $\mathcal{X}(u) \neq \mathcal{X}(v)$ et $\mathcal{Y}(u) \neq \mathcal{Y}(v)$. Plus formellement, soient a_{11} le nombre de paires de nœuds de telle sorte qu'ils soient dans le même ensemble dans les deux partitions, a_{00} le nombre de paires de nœuds de telle sorte qu'ils soient dans des ensembles différents dans les deux partitions et a_{10} (*resp.* a_{01}) le nombre de paires de nœuds de telle qu'ils soient dans le même ensemble dans \mathcal{X} (*resp.* \mathcal{Y}) et dans deux ensembles différents dans \mathcal{Y} (*resp.* \mathcal{X}). Avec ces notations, le *Rand Index* est défini de la manière suivante :

$$RI(\mathcal{X}, \mathcal{Y}) = \frac{a_{11} + a_{00}}{a_{11} + a_{01} + a_{10} + a_{00}} \quad (1.3)$$

Il se peut que, par chance, deux partitions classent de la même manière une paire de nœuds. C'est pourquoi une version ajustée du *Rand Index* (ARI) a été proposée [HA85]. Le *Rand Index* et sa version ajustée permettent de comparer des partitions. Porumbel *et al.* [PHK11] ont proposé l'*Omega Index* qui est une extension de l'ARI pour les couvertures.

Approche venant de la théorie de l'information On peut considérer que l'assignation d'un élément à un ensemble est une variable aléatoire. Dans ce cas, la probabilité d'un élément d'être dans un ensemble $X \in \mathcal{X}$ est $P(X) = n_X / |V|$. De manière similaire, la probabilité jointe est $P(X, Y) = |X \cap Y| / |V|$. Avec ces définitions, il est possible de calculer l'entropie d'une partition, $H(\mathcal{X})$, l'entropie conditionnelle, $H(\mathcal{X}|\mathcal{Y})$ et l'information mutuel $I(\mathcal{X}, \mathcal{Y})$. Cette dernière est définie par $I(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y})$. L'entropie, $H(\mathcal{X})$, et l'entropie conditionnelle, $H(\mathcal{X}|\mathcal{Y})$, sont définies dans le sens de Shanon par :

$$H(\mathcal{X}) = - \sum_{X \in \mathcal{X}} P(X) \log(P(X)), \quad H(\mathcal{X}|\mathcal{Y}) = - \sum_{X \in \mathcal{X}, Y \in \mathcal{Y}} P(X, Y) \log \frac{P(X, Y)}{P(Y)}. \quad (1.4)$$

Afin de normaliser l'information mutuel, Danon *et al.* [DDDGA05] ont défini l'information mutuel normalisée (NMI_{shanon}) :

$$NMI_{shanon}(\mathcal{X}, \mathcal{Y}) = \frac{2I(\mathcal{X}, \mathcal{Y})}{H(\mathcal{X}) + H(\mathcal{Y})}. \quad (1.5)$$

Lancichinetti *et al.* l'ont par la suite étendue pour prendre en compte les couvertures [LF09b]. Le choix de la normalisation dans le cas chevauchant semble cependant toujours ouvert [aMGH11, Zha15].

Résumé

Il est intéressant de noter que la littérature sur la comparaison de structures est assez restreinte comparé à celle sur la détection de communautés. En effet, il semble qu'uniquement 3 similarités soient couramment utilisées : la similarité se basant sur Jaccard, l'Omega index et la NMI. Surprenamment, tout les indices de similarité ont été étendus aux couvertures de manière assez convaincantes. Il est également important de noter l'existence d'implémentations librement accessible d'une majeure parties de ces métriques^{a b}.

a. <https://github.com/aaronmcdaid/Overlapping-NMI>

b. <http://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation>

1.2 Communauté dans les graphes

Ce champs de recherche est très vaste et il est illusoire de vouloir énumérer les méthodes existantes dans ce domaine car elles sont extrêmement nombreuses et les caractéristiques voulues d'une communauté peuvent varier selon le contexte [LLDM08, CGP11, YL15, JBP⁺15]. Les méthodes se séparent tout de même en deux catégories selon si elles capturent une partition ou une couverture de nœuds. Ces deux structures correspondent à deux visions possibles de l'organisation d'un graphe et du réseau sous-jacent. Nous présentons ces deux catégories dans les sous-sections suivantes. Il existe également une troisième catégorie qui est la détection de communautés que nous traitons dans le chapitre 5.

1.2.1 Parititons de nœuds

Afin de mieux comprendre ce que peut capturer une partition de nœuds, il est plus facile de partir d'un exemple. Dans l'étude de Stehlé *et al.* [SVB⁺11], des enfants d'une école primaire ont eu pendant 2 jours des capteurs enregistrant lorsque deux enfants sont à une distance de moins de 1 mètre 50. Ce dispositif permet de mesurer les interactions entre élèves et de construire le graphe des relations à l'école. Un lien existe entre deux élèves si ils ont interagi au moins une fois ensemble. Une illustration du graphe obtenu est visible dans la figure 1.1. La classe de chaque élève est également connue. Comme chaque élève appartient à une et une seule classe, les classes forment une partition des élèves. Cette partition est une bonne structure communautaire car on remarque que les élèves d'une même classe interagissent beaucoup entre eux mais peu avec les élèves des autres classes. Cela se remarque particulièrement bien pour la classe 3A. Il existe beaucoup de liens entre les élèves de la classe 3A mais aucun avec les élèves de la classe 5A par exemple.

Afin de capturer des partitions de nœuds, beaucoup de méthodes existent. Il y a d'ailleurs régulièrement des états de l'art qui sont publiés [For10, PC13, MV13, HBG14]. Afin d'aider le lecteur, nous détaillons ici quelques unes des méthodes les plus utilisées.

Méthodes utilisant un modèle

Comme une communauté est souvent définie comme devant être très connectée, le problème est de trouver une fonction capable d'évaluer la connectivité d'une communauté. Pour ce faire, il faut définir une métrique qui mesure la connectivité, *e.g.* le nombre de liens dans un groupe. Puis, il est nécessaire de définir comment évaluer cette métrique.

Il est possible d'utiliser une métrique en la normalisant par ses bornes minimum et maximum. Avec une métrique normalisée entre 0 et 1, un groupe est alors considéré comme très connecté si son évaluation est supérieure à un certain seuil. Par exemple, le nombre de lien

1. Image provenant de <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0023176>.

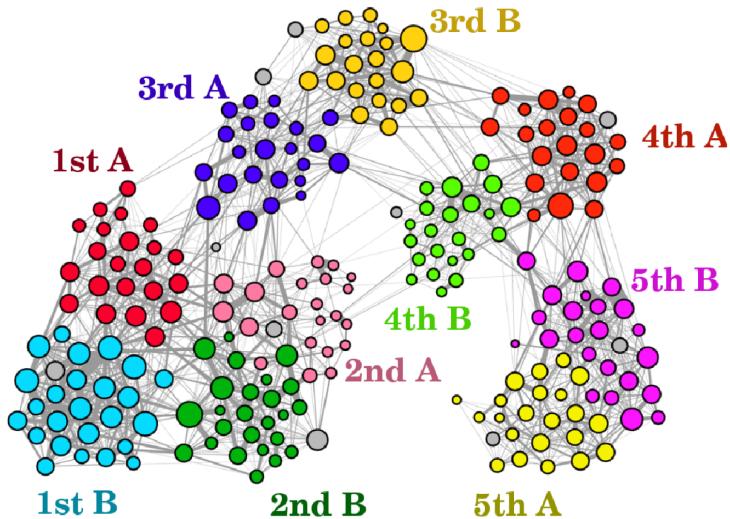


FIGURE 1.1 – Graphe de contact des enfants d'une école primaire. L'épaisseur du lien représente la durée de communication entre deux élèves. La couleur représente la classe de chaque élève. Les professeurs sont en gris.¹

entre n nœuds peut être normalisé par le nombre de liens dans un arbre et dans une clique de taille n . Cette approche n'est cependant pas adaptée pour la recherche de communautés car elle ne tient pas compte de la structure du graphe². Prenons l'exemple du graphe constitué d'une unique clique. Dans ce graphe, tout groupe de nœuds est également une clique et par conséquent tout groupe de nœuds a une évaluation parfaite de 1. Chaque groupe serait donc une très bonne communauté selon cette évaluation. Or, le graphe constitué d'une unique clique ne possède pas de structure communautaire contrairement au graphe dans la figure 1.1. Il est donc nécessaire de trouver une autre approche.

Plutôt que de normaliser une métrique par ses valeurs minimum et maximum, il est intéressant de considérer l'écart à une valeur attendue. L'idée est la suivante : quelle serait la valeur attendue de la métrique considérée si le graphe n'avait pas de structure communautaire. Le problème est alors de "retirer" la structure communautaire du graphe ou bien de définir un ensemble de graphes similaires au graphe initial mais n'ayant pas de structure communautaire. Ce processus définit alors ce qu'on appelle un *modèle nul*. Déetecter des communautés se traduit ainsi en trouver des groupes qui s'éloignent du modèle nul où il n'existe pas de communauté.

Ce changement est astucieux car il est relativement aisé de définir des graphes n'ayant pas de structure. Il suffit de créer des graphes complètement aléatoires [ER59] où les liens du graphes sont tirés de manière uniforme. Afin que les graphes aléatoires puissent être comparé au graphe initial, des contraintes sont généralement ajoutées et donnent lieu à différents modèles nuls. Le premier modèle nul de graphe est celui de Erdős-Rényi [ER59] où le nombre de liens et le nombre de nœuds des graphes aléatoires doivent être les mêmes que dans le graphe initial. Un autre modèle très couramment utilisé est le modèle de configuration [BC78]. Dans ce modèle, la distribution des degrés est également fixe. Le modèle de configuration est utilisé car il a été observé que les graphes provenant de données réelles ont une distribution des degrés très éloignées d'une distribution uniforme. C'est pourquoi l'ajout de la contrainte sur les degrés permet de considérer des graphes plus proches du graphe initial. Il existe bien évidemment d'autres modèles possibles considérant d'autres contraintes [New09].

2. Cette approche est plus appropriée dans la recherche des groupes les plus denses [BBC⁺15].

Modularité La modularité [NG04] est une fonction qui associe à chaque partition de nœuds une valeur entre -1 et 1 . Plus la valeur de modularité d'une partition est élevée, plus la partition est censée capturer une bonne structure communautaire. La modularité est définie de la manière suivante pour une partition \mathcal{C} :

$$Q(\mathcal{C}) = \frac{1}{2m} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta_{\mathcal{C}(i)=\mathcal{C}(j)}, \quad (1.6)$$

où $\delta_{\mathcal{C}(i)=\mathcal{C}(j)}$ est égale à 1 si i et j sont dans la même communauté et 0 sinon. Il s'agit pour deux nœuds d'une même communauté de comparer la présence ou l'absence d'un lien, A_{ij} , à la probabilité que ces deux nœuds soient reliés dans le modèle de configuration, $\frac{d_i d_j}{2m}$. L'idée sous-jacente est que les nœuds d'une communauté devraient partager plus de liens qu'espérés dans le modèle de configuration. De très nombreux travaux ont par la suite étudié les caractéristiques de la modularité et son optimisation. Tout d'abords, il a été montré que l'optimisation de la modularité est un problème NP-Complet [BDG⁺07]. Il est donc nécessaire de recourir à des heuristiques afin de trouver rapidement une partition proche de l'optimum. Parmi l'ensemble des algorithmes existants, l'algorithme de Louvain [BGLL08] est un des plus rapides. Il existe également des variantes de cet algorithme [HBP15, Tra15].

D'autres travaux se sont attachés à l'étude de la modularité. Il a été montré que la modularité souffre du problème de *réolution limite* [FB07, LF11] car le modèle de configuration presuppose une répartition uniforme des tailles des communautés. Il a par ailleurs été montré que la modularité n'offre pas de maximum clair et que beaucoup de partitions différentes ont des évaluations proches [GdMC10]. Pour répondre à ces problèmes, il existe différentes variantes de la modularité [RB06, DYB10].

Surprise La fonction Surprise [AM11, TAD15] est une autre fonction de qualité qui se base quant à elle sur le modèle de Erdös-Rényi pour évaluer la surprise d'observer un groupe de nœuds relié par l liens.

Stochastic Block Model Nous avons défini précédemment la notion de modèle nul permettant de se comparer à l'absence de communautés. Il est également possible de modéliser une structure communautaire puis de vérifier *a posteriori* si ce modèle pourrait être à l'origine du graphe observé. Le problème de détection de communauté est alors un problème d'inférence qui est traité avec des outils statistiques tel que le *stochastic block model* (SBM) [HLL83, NS01]. L'idée derrière le SBM est la suivante : la probabilité que deux nœuds soient reliés dépend uniquement de leur groupe respectif. Si le graphe a une structure communautaire, alors deux nœuds d'une même communauté devraient avoir une forte chance d'être connectés. À l'inverse, deux nœuds de deux communautés différentes devraient avoir une probabilité assez faible d'être connectés. Le SBM est défini par de nombreux paramètres : le nombre de groupes, l'assignation d'un groupe à chaque nœud et les probabilités d'interactions entre les groupes. Avec un jeu de paramètres donné, il est possible de calculer la vraisemblance que ce jeu de paramètres soit à l'origine du graphe. Trouver une partition de nœuds dans ce contexte est alors équivalent à trouver le jeu de paramètre qui est le plus vraisemblable à l'origine du graphe.

Dans le SBM, tous les nœuds sont considérés comme équivalents en particulier vis-à-vis du degré ce qui n'est pas le cas dans beaucoup de graphes provenant de données réelles. C'est pourquoi une version tenant compte du degré des nœuds a été proposée : le Degree-corrected Stochastic Block Model (DBSM) [KN11]. Enfin d'après de récents travaux [New16], il semblerait que le DBSM et l'optimisation de la modularité soient liés.

Méthodes utilisant des marches aléatoires

Il existe d'autres approches que celles utilisant un modèle nul ou un modèle génératif. Notamment, il y a les méthodes utilisant les marches aléatoires [PL05, RB08]. Ces méthodes tirent parti du fait que si une communauté est densément connectée alors un marcheur aléatoire devrait y rester assez longtemps. En particulier, la méthode Infomap [RB08] repose sur une idée très élégante qui est la suivante. Une partition de nœuds est une carte du graphe et, en ce sens, elle doit aider sa lecture.

Une carte est efficace si elle permet de mieux comprendre l'objet d'étude en réduisant sa complexité. Dans une carte d'un pays, les départements découpent l'espace en zones disjointes et la majorité des routes se trouvent à l'intérieur des départements. Dans une carte, il est courant qu'un nom de ville soit unique dans un département mais qu'un même nom puisse être utilisé dans plusieurs départements. De plus, un voyageur se déplaçant aléatoirement sur les routes a peu de chance de sortir d'un département. Pour décrire, *a posteriori*, l'ensemble des villes traversées par ce voyageur, il suffit alors de donner le département initial puis la liste des villes visitées. Il n'est pas nécessaire de répéter le département pour chaque ville si le voyageur n'en est pas sorti. En ce sens, la découpe d'un pays en département permet de réduire la complexité de la description du voyage. Il s'agit donc d'un problème lié à la théorie de l'information et de sa compression. De manière similaire, Rosval *et al.* utilise des marcheurs aléatoires se déplaçant sur les nœuds du graphe et les communautés comme zones du graphe. Si les communautés sont bien formées, alors les marcheurs aléatoires restent bloqués à l'intérieur des communautés et la description de leur marche est courte. La longueur de cette description devient alors la signature de la partition et plus la signature est courte, meilleure la partition est.

Le phénomène de résolution limite a également été étudié dans le cadre de Infomap [KR15]. Il semble que cet effet existe également dans Infomap mais qu'il soit beaucoup moins prononcé.

Autres méthodes Il existe bien d'autres méthodes pour détecter des communautés en tant que partition de nœuds. Il y a les méthodes spectrales [DM04, MT09] qui se basent sur les vecteurs propres de la représentation d'un graphe sous la forme d'une matrice. De manière moins formelle, il existe également les méthodes de propagation de labels (LPA) [RAK07, LLJT14]. Dans ces méthodes, chaque nœud a initialement un label puis à chaque itération chaque nœud prend comme label un des labels de ses voisins. En général, un nœud prend comme label celui qui est le plus présent parmi ses voisins. Au bout d'un certain nombre d'itérations ou une fois à l'équilibre, il ne reste que quelques labels dans le graphe et ils représentent les communautés.

Résumé

Il existe de très nombreuses méthodes pour la détection de communautés en tant que partition de nœuds. Ils semblent que les méthodes d'optimisation de **modularité**, la méthode **infomap** et les **Stochastic Block Model** soient les plus utilisées dans la littérature.

1.2.2 Couverture de nœuds

Jusqu'à maintenant, nous avons considéré les communautés comme des partitions de nœuds. Or, les partitions sont très restrictives et ne peuvent pas capturer toutes les situations possibles. Prenons l'exemple d'un graphe reflétant des interactions de personnes. Il existe des communautés qui sont disjointes comme le travail et la famille mais bien souvent des personnes appartiennent à plusieurs groupes, voir l'exemple dans la figure 1.2. Ainsi le groupe des personnes faisant du sport et le groupe des personnes travaillant ensemble peuvent ne pas être disjoints. Si tel est le cas, alors il n'est plus possible de représenter ces communautés avec une partition. Il est nécessaire de manipuler une couverture de nœuds. Ainsi dans l'exemple dans la figure 1.2, les nœuds rouges appartiennent à deux groupes au lieu d'un seul.

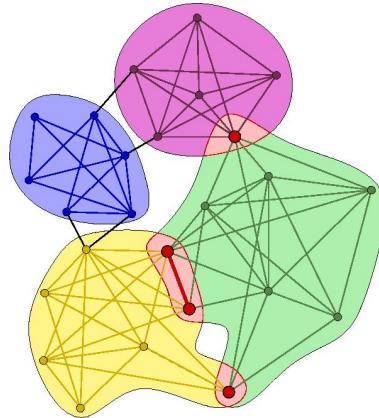


FIGURE 1.2 – Exemple graphe avec une structure communautaire chevauchante représentée par les couleurs³.

Une fois encore, la littérature est très vaste dans ce domaine et nous ne ferons pas une liste exhaustive des méthodes existantes. Pour une liste plus exhaustive, il existe de nombreux état-de-l'art dans le domaine [Kan14, XKS13, BCS15, HDF14].

Une des premières méthodes de détection de couverture de nœuds est la *Clique Percolation Method* (CPM) [PDFV05]. L'algorithme CPM repose sur le principe de transitivité qui serait à l'origine des communautés : si i et j sont reliés par un lien, alors un nœud k qui serait relié à i aurait une forte chance d'être également connecté à j . Il s'agit de la formalisation du proverbe "Les amis de mes amis sont mes amis". Si ce principe est réellement à l'origine des communautés, alors les communautés doivent être composées de plusieurs cliques. C'est pourquoi CPM cherche l'ensemble des cliques d'une taille k donnée⁴ puis fusionne toutes les cliques qui partagent suffisamment de nœuds, en général $k - 1$. Comme cette méthode est relativement couteuse, Kumpula *et al.* [KKKS08] ont repris le même mécanisme en optimisant le mode de calcul.

Extension de méthodes existantes

La majorité des méthodes existantes pour les partitions ont été adaptées pour manipuler les couvertures de nœuds. Il existe notamment plusieurs extensions de la modularité [SCCH09, NMCM09]. Cependant ces extensions ne reposent plus sur un modèle nul car elles introduisent des termes de normalisations. Par conséquent, ces extensions sont beaucoup moins utilisées que la modularité initiale.

Stochastic Block Model En revanche, le SBM s'adapte très bien aux couvertures de nœuds. Une extension du SBM est le Mix-Membership Stochastic Block Model (MMSBM)[ABFX08] qui permet à un nœuds d'avoir plusieurs groupes. Dans [GB13], les auteurs utilisent la même méthode mais en échantillonnant le graphe initial afin de réduire le cout de calcul. Il existe d'autres méthodes à base de modèle génératif [BKN11, YL13]. Ces méthodes se basent sur des graphes d'affiliations[Bre74]. Un graphe d'affiliation est un graphe biparti entre les nœuds d'une part et les communautés de l'autre part. Dans la méthode de Ballet *et al.* [BKN11], ce graphe d'affiliation est pondéré et la pondération représente la propension d'un nœud à créer des liens dans un groupe donné tant dis que pour Yanget *al.* il s'agit du facteur d'appartenance du nœuds au groupe. Une fois le graphe d'affiliation défini, il est nécessaire de savoir recréer la matrice d'adjacence et, en ce sens, ces méthodes se rapprochent des méthodes de factorisation

3. Image provenant de https://en.wikipedia.org/wiki/Clique_percolation_method.

4. En générale, k est compris entre 3 et 5 pour des raisons de coût de calcul

en matrices non-négatives [LS99]. Le but de la factorisation non-négative d'une matrice donnée est d'être capable de trouver deux matrices dont les entrées sont non-négatives tel que leur combinaison permettre de retrouver la matrice initiale.

Jusque récemment ce genre de technique généralisant le SBM ne pouvait s'appliquer qu'à des graphes relativement petits. Il semble cependant que les méthodes récentes [GB13, YL13] arrivent à traiter des graphes ayant énormément de liens, de l'ordre $10^9 \sim 10^{12}$ liens.

Propagation de label Des méthodes ont étendus la propagation de label aux couvertures de nœuds [Gre10, XSL11]. Le principal changement est qu'un nœud ne stocke plus un unique label mais soit plusieurs labels soit des fréquences d'apparition de labels. Ainsi à la fin de l'algorithme, il suffit de choisir les labels les plus fréquents. Cependant ce mécanisme de propagation change radicalement l'idée initiale. En effet, la diffusion d'un label n'est plus directement contrainte par la diffusion des autres labels. Dans cette nouvelle configuration, il est possible qu'un label soit présent dans tous les nœuds. De plus selon les auteurs de ces méthodes, des communautés non connexes peuvent apparaître ce qui nécessite l'ajout d'un mécanisme de *post-processing*.

Infomap Il s'agit sûrement de la méthode étant la moins "déformée" avec le SBM par l'adaptation aux couvertures de nœuds. En effet, il suffit de relâcher la contrainte sur le fait qu'un nœud n'appartienne qu'à un seul groupe. Il est toujours possible de décrire un trajet par un nom de groupe puis une liste de nœuds visités dans le même groupe. Ainsi, la notion de longueur de description est toujours valide dans ce contexte. Ce travail d'extension a été fait par Esquivel et al. [ER11].

Méthodes utilisant des communauté égocentré

On a vu avec la modularité qu'il est assez difficile de définir une fonction de qualité évaluant une couverture de nœuds en fonction des caractéristiques des groupes. C'est pourquoi certaines méthodes s'affranchissent complètement de fonction de qualité globale et se concentrent sur des propriétés locales. Dans cette philosophie, il y a les méthodes se basant sur des fonctions de proximité et celle utilisant des fonctions de qualité locales.

Dans le premier cas, le but est de mesurer la proximité entre tout les nœuds et un nœud appelé égo. Puis une fois les nœuds ordonnés selon leur similarité, il suffit de définir une coupe pour séparer les nœuds appartenant à la même communauté que l'égo et les autres. Le choix de la coupe se fait, en générale, en fonction de la présence de forte décroissance dans le profil de la similarité. Pour obtenir une couverture de nœuds, il est nécessaire d'appliquer ce processus égocentré sur plusieurs égos. Danisch et al. [DGG12] utilise une méthode de diffusion pour mesurer la similarité, alors que Whang et al [WGD13] utilise le *Personalized Page Rank*. Ces méthodes ne sont cependant pas à même d'évaluer les communautés qu'elles trouvent.

D'autres méthodes prennent le contre pied des méthodes de proximité en utilisant des fonctions de qualité locales. Il s'agit d'algorithmes qui initialement considèrent de très petites communautés puis essayent de les étendre itérativement en ajoutant des nœuds. Afin de ne pas étendre un groupe indéfiniment, un ajout n'est fait que si l'améliore la fonction de qualité locale. Ainsi dans ce type d'algorithme, il y a principalement deux critères importants : le choix des communautés initiales et le critère d'évaluation. Dans la majorité des cas, chaque nœud constitue une communauté. OSLOM[LRRF11] diffère de cette situation car OSLOM utilise comme point de départ une partition trouvée par l'algorithme de Louvain ou par infomap.

Les fonctions de qualité applicables sont très nombreuses et nous n'en mentionneront que trois. Tout d'abord, il est possible d'utiliser le degré relatif [LWP08] d'un groupe comme critère. Il s'agit du ratio entre le nombre de l_{in} liens internes à un groupe de nœuds et du nombre

de liens total $l_{in} + l_{out}$: $\frac{l_{in}}{l_{in} + l_{out}}$. En effet, plus le degré relatif est élevé, plus le groupe est dense comparé à son voisinage et plus il a de forte chance d'être une bonne communauté. Cette formulation est assez proche de la conductance ou coupe normalisée[SM00] :

$$\phi = \frac{l_{out}}{\min(2(l_{in} + l_{out}), m - 2(l_{in} - l_{out}))} \quad (1.7)$$

Ces deux notions se rapportent à la notion de densité vis-à-vis de leur voisinage. Il également possible d'évaluer une communauté locale par rapport aux nombre de triangles présents dans la communauté. C'est ce que mesure la cohesion [FCF11] pour un groupe de taille k :

$$cohesion = \frac{\Delta_3}{\binom{k}{3}} \times \frac{\Delta_3}{\Delta_3 + \Delta_2}, \quad (1.8)$$

où Δ_3 est le nombre de triangles dont les 3 nœuds sont à l'intérieur du groupe et Δ_2 est le nombre de triangles dont uniquement 2 nœuds sont à l'intérieur du groupe.

D'autres méthodes d'initialisation ainsi que fonctions de qualité sont détaillées dans l'article de Kanawati [Kan14]. Ces méthodes semblent très prometteuses car elles permettent de traiter efficacement de très grand graphe tout comme les LPA mais elles profitent des fonctions de qualité locales qui permettent d'une certaine manière d'évaluer le résultat obtenu.

Résumé

La littérature sur la détection de couverture de nœuds est encore très récente. Il est donc délicat de commencer à tirer des conclusions. Cependant, il semble que les extensions de la modularité ne soient pas réellement capable de trouver des couvertures de nœuds. En plus d'infomap, c'est surtout les méthodes utilisant des modèles génératifs et celles utilisant des fonctions de qualité locales qui semblent les plus à même de capturer des couvertures de nœuds pertinentes. Les modèles génératifs ne semblent en effet plus limiter à de petits graphes. Les méthodes utilisant des fonctions de qualité locales sont quant à elle très rapide et le choix de la fonction de qualité permet de s'adapter au contexte.

1.3 Extension temporelle des graphes

Les graphes sont utilisés pour représenter une situation ou résoudre un problème réel. Dans la section précédente, nous nous sommes attachés à présenter une partie des méthodes considérant le problème de la détection de communautés recouvrantes ou non. Cependant toutes ces méthodes reposent sur la validité de la représentation du problème par un graphe. Cette hypothèse est justifiée dans énormément de cas mais elle ne l'est plus lorsque l'objet d'étude évolue beaucoup.

Afin d'illustrer ce propos, nous appuyons sur le même exemple de réseau de personnes que nous avons présenté dans la sous-section 1.2.1 et qui est illustré dans la figure 1.1. Dans cet exemple, deux élèves sont reliés dans le graphe si ils ont interagi au moins une fois au cours de la capture. Comme la capture n'a duré que deux jours, il est fort probable que les élèves n'aient pas changé d'habitude durant la capture. En étudiant le graphe agrégé, il est possible de mettre en avant l'organisation générale des élèves mais déjà de l'information a été perdue. En effet, il n'est pas possible de différencier le comportement observé le matin de celui observé le soir car l'ensemble des interactions sont agrégées dans le graphe. Ce n'est pas tout, imaginons que la capture ait duré plusieurs mois voir plusieurs années. Dans ce cas de figure, il ne sera plus possible de dégager un comportement général des élèves car il aura changé durant ce laps de temps. En particulier, de nouveaux groupes d'amis se seront formés et d'autres auront disparu. Tout ces changements impactent le graphe agrégé et tendent au fur et à mesure à le rapprocher

à une clique ce qui le rend complètement inexploitable. Le temps est donc une dimension qu'il est nécessaire de prendre en compte.

Le domaine de l'épidémiologie est un très bon exemple de l'importance du temps. Un modèle épidémique modélise la propagation d'une maladie dans une population en fonction des contacts qui existent. Il est donc tout naturel de s'appuyer initialement sur un graphe où les nœuds représentent des personnes et les liens représentent les interactions entre personnes. En plus du graphe, il est nécessaire d'ajouter l'état de chaque nœud, *e.g.* sain ou infecté. Ainsi, un nœud sain ne peut devenir infecté que s'il est relié à un nœud infecté dans le graphe. Ce genre de modèle mettent donc en avant un chemin de diffusion, *e.g.* une suite de personnes transmettant la maladie à l'autre.

Afin d'illustrer ce phénomène, les premiers travaux [VBB08] s'appuient sur un graphe d'interactions de personnes agrégant toute l'information temporelle. Or, la prise en compte du temps dans ce contexte est primordiale car il est possible que le chemin d'infections simulé dans le graphe agrégé ne soit pas réalisable si l'on prend en compte le temps. Imaginons 3 personnes A, B, C tel que A et B interagissent à l'instant 1, B et C interagissent à l'instant 2. Dans le graphe agrégé, il est possible pour C d'infecter A via B alors que dans la réalité cela n'est possible. L'ajout du temps impacte donc fortement les résultats obtenus dans le contexte épidémiologique et ce phénomène est d'ailleurs très étudié [GPBC15, KKP⁺11, JPKK14, HK14, HL14, SWP⁺14, PJHS14].

Une fois reconnue l'importance du temps, il est nécessaire de trouver un nouveau formalisme étendant la théorie des graphes pour en tenir compte. Nous présentons maintenant différentes extensions possibles de la théorie des graphes, dans les sous-sections 1.3.1 et 1.3.2. Nous détaillons également comment la recherche de communautés se transpose dans ces nouveaux formalismes et plus généralement quels sont les problèmes qu'ils permettent de résoudre. Des états-de-l'art dans ce domaine ont d'ailleurs déjà été esquissés [BBC⁺14, CA14, HKW14].

1.3.1 Extensions avec pertes d'informations temporales

Séries de graphes

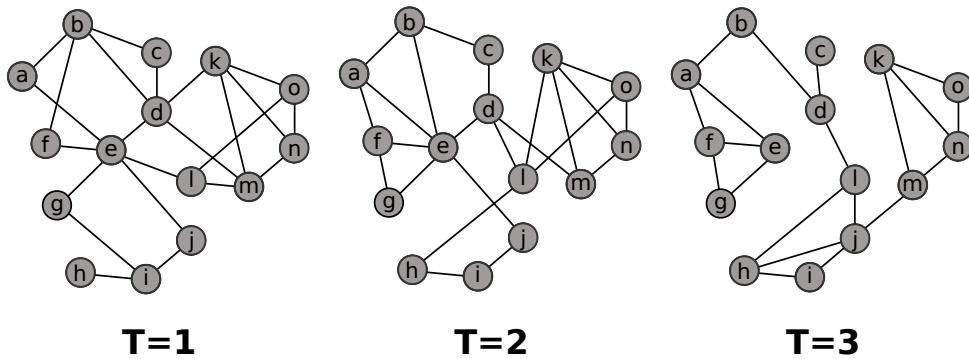


FIGURE 1.3 – Exemple de série de graphes sur trois intervalles de temps.

La première solution qui a été apportée ne prend le temps en compte que partiellement. Il s'agit de manipuler une série de graphe où chaque graphe représente le réseau durant un intervalle de temps donné. Ainsi, il est possible d'appliquer les outils de la théorie des graphes sur chaque intervalle. Cependant chaque intervalle de temps est représenté par un graphe agrégé. Il y a donc toujours une perte d'information. Plus formellement, une série de graphe est définie par $\mathcal{G} = \{G_i\}_{i < T}$ où T est un entier, voir l'illustration 1.3.

Cette définition initiale laisse le choix sur la découpe du temps en intervalle. Il est possible de choisir des intervalles de taille égales ou non et disjoints ou chevauchant [WFAG12]. Utiliser

des intervalles à durée variable permet de mieux tenir compte de la dynamique. Par exemple lors de l'étude d'interactions de personnes, il est très courant d'observer une très faible activité la nuit et une plus forte la journée. Un graphe agrégeant ce qui se passe sur un intervalle de 5h sera vide dans le premier cas et peut être trop dense dans le second. La détection d'intervalle pertinent est donc un vaste sujet de recherche [RB10, KKB⁺12, RPB13, CBW13, PC15, DVR16].

La notion même de temps est importante. Albano *et al.* [AGL14] propose d'utiliser une autre mesure que la seconde mais plutôt le nombre de changements comme mesure du temps. Ainsi, le temps n'avance pas si aucun changement n'a lieu et au contraire il avance beaucoup si énormément de changements apparaissent. Cette manière de procéder se rapproche des travaux de Lamport [Lam78] dans les systèmes distribués.

Détection de communautés Dans ce contexte de série de graphes, il y a eu assez tôt des méthodes de détection de communautés [HKKS04, SFPY07, LCZ⁺08, APU09]. Il est, en effet, assez naturel d'appliquer une méthode de détection statique sur chaque graphe puis d'essayer de faire du suivi de communautés. Le suivi de communautés consiste à comprendre comme une communauté donnée évolue, étant donné une série de graphes et une série de partitions. Palla *et al.* [PBV07] ont été parmi les premiers à décrire les évolutions possibles d'une communauté. Muni d'indicateur de similarité tel que l'indice de Jaccard, voir 1.1.2, il est possible de trouver les communautés les plus proches aux instants précédent et suivant. À partir de ces informations, 5 type d'évolutions d'une communauté sont définis :

Naissance Une nouvelle communauté apparaît.

Agrandissement La communauté continue d'exister et s'agrandit.

Fusion Deux communautés fusionnent pour donner lieu à une nouvelle communauté.

Division Une communauté se sépare en deux nouvelles communautés à l'étape suivante.

Mort Une communauté cesse d'exister dans les graphes suivants.

Ce type de méthodes souffre souvent de l'instabilité des méthodes de détection [AG10, HBG14]. Si les partitions changent complètement entre deux graphes consécutif, alors il est difficile de faire un réel suivi de communautés. C'est pourquoi des méthodes essayent de forcer une certaine stabilité de la partition en ajoutant un coût de transition [CKT06, CKU13, KBV15]. Une approche détournée pour garder une certaine stabilité est d'utiliser la partition trouvée précédemment comme base de recherche pour l'intervalle suivant [LRRF11].

Des extensions du Stochastic Block Models ont également été proposées par différents auteurs dans le cadre des séries de graphes. Yang *et al.* [YCZ⁺11] sont parmi les premiers à considérer ce cas de figure. Ils considèrent que la probabilité d'un lien entre deux communautés est fixe et que ce qui change est l'affiliation des noeuds au cours du temps. Ce processus de changement de communauté suit alors une chaîne de markov cachée. À l'inverse, Corneli *et al.* [CLR16] considèrent une partition de noeuds fixe tout au long du temps et c'est l'activité entre deux communautés qui change selon l'intervalle de temps. Xu *et al.* [XH14] permettent à l'affiliation et à l'activité entre deux communautés de changer selon l'intervalle de temps. Cependant, il semblerait que cette relaxation se fasse au dépens de l'identifiabilité des paramètres d'après Matias *et al.* [MM15]. Ils ont donc proposé une nouvelle méthode afin résoudre ce problème. De plus, leur méthode permet de traiter des séries de graphes pondérés.

Tenseur 3D

Les tenseurs 3D ne sont pas en soi différent des séries de graphes. Il est possible de voir un graphe comme une matrice carré d'adjacence. Il est donc normal de concevoir une série de graphes comme un tenseur appartenant à \mathcal{R}_{nnT} . Ce changement de point de vue permet ainsi d'appliquer les méthodes d'algèbre linéaire, notamment la décomposition de tenseur. C'est

la méthode proposée par Gauvin *et al.* [GPC14] pour étudier la structure communautaire des interactions d'élèves. Cependant ce genre de décomposition semble moins expressive que les SBM.

Graphes multicouches

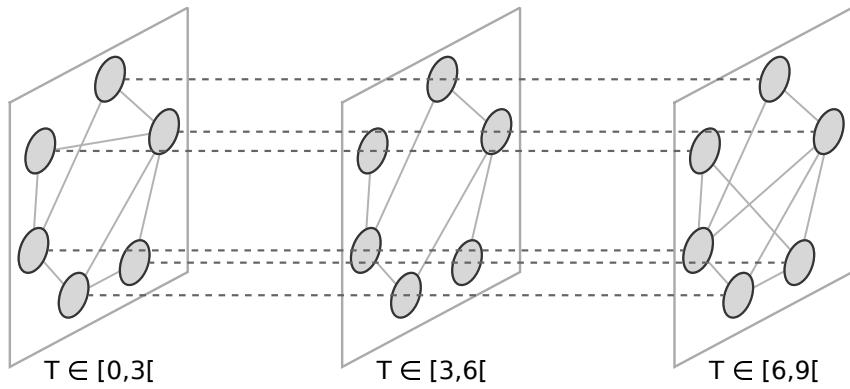


FIGURE 1.4 – Graphe multicouche avec trois couches représentant le temps. Les liens pleins (resp. pointillés) sont les liens intra-couches (resp. inter-couches).

La construction de graphes multicouches (*multilayer* ou *multiplex*) est proche de l'idée des séries de graphes. Tout comme les séries de graphes, des nœuds et des liens sont créés à chaque intervalle pour représenter ce qui s'est déroulé durant l'intervalle de temps. Cependant, le graphe multicouche ajoute des liens entre les nœuds de deux intervalles, voir l'illustration 1.4. Par conséquent, un graphe multicouche est un graphe où il existe deux types de liens, les liens intra-couches et les inter-couches et il existe T répliques d'un même nœud, une par intervalle de temps. Les liens intra-couches représentent un connexion entre deux nœuds durant un intervalle de temps. Les liens inter-couches représentent un lien entre deux nœuds sur deux intervalles différents. Ces derniers sont utilisés pour identifier un même nœud sur plusieurs intervalles et sont en générales limités à relier deux couches consécutives.

Les graphes multicouches représentent les données évoluant dans le temps mais ils modélisent aussi très bien d'autres situations. Par exemple, ils permettent de représenter facilement les différents moyens de transport dans une ville où chaque moyen de transport (bus, voiture, métro ...) est représenté par une couche. Plusieurs travaux [DSRC⁺13, KAB⁺14, BBC⁺14] décrivent les graphes multicouches et leurs applications.

Détection de communautés Grâce au formalisme de graphe multicouche, il est possible de traiter le temps de manière un peu plus fine que dans les séries de graphes car il permet de mieux suivre l'évolution des nœuds. Comme un graphe multicouche est un graphe, il est possible d'adapter les méthodes existantes pour tenir compte des différents types de liens. C'est le cas d'infomap [DLAR14], de la modularité [MRM⁺10, BPW⁺13, BPW⁺16] et du SBM [SSTM15, Pei15].

Résumé

Les séries de graphes, les tenseurs et les graphes multicouches permettent de prendre en compte le temps tout en autorisant l'utilisation de méthodes conçues sur un graphe statique. Cette liberté d'utilisation a un coût. Ces approches reposent sur une découpe du temps en sous-intervalles durant lesquelles le temps n'est plus pris en compte afin d'obtenir un graphe statique. Or, il peut être délicat de définir ces intervalles de temps. De plus, la construction des graphes agrégés entraîne une perte d'information temporelle et cela impacte la précision temporelle des structures communautaires qui sont manipulables. Il n'est pas envisageable d'augmenter le nombre d'intervalle de temps car cela induirait d'une part des graphes agrégés avec très peu de liens et d'autre part le temps de calcul serait très fortement impacté.

1.3.2 Extension sans perte d'information temporelle

Graphe temporel

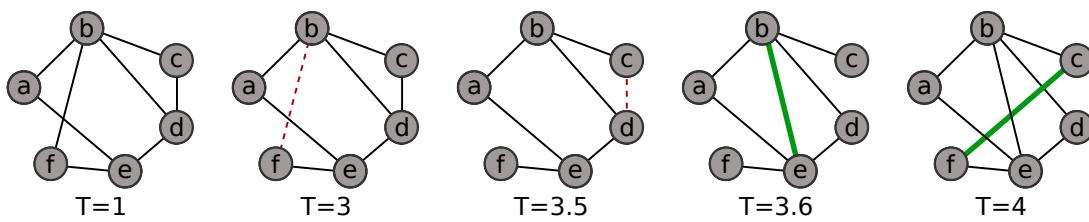


FIGURE 1.5 – Graphe temporel avec des ajouts de lien représentés en trait épais vert et des retraits de lien représentés par des liens pointillés rouge.

Les graphes temporels (*Time Varying Graph* ou *Evolving Graph*) permettent de tenir compte de l'ensemble de l'information temporelle. Pour cela au lieu de considérer des intervalles de temps, ils considèrent l'ensemble des modifications qui affectent le graphe : les ajouts et retraits de liens. En pratique, cela revient à considérer sur chaque lien une fonction de présence en fonction de temps qui vaut 1 à un instant t si le lien existe à cet instant et 0 sinon. Ainsi, il est possible de connaître la structure de graphe à chaque instant. Ce formalisme est présenté dans différents travaux [CFQS11, WZF14] et illustré dans la figure 1.5. Dans cette figure, on voit apparaître l'ordre de modification du graphe. Tout d'abord, les liens (b, f) et (c, d) disparaissent puis les liens (b, e) et (f, c) apparaissent chacun leur tour.

Détection de communautés Dans un graphe temporel, une structure de graphe existe à chaque instant. Il est donc possible de calculer après chaque modification l'évolution d'une métrique. Par exemple, il est possible de calculer après l'ajout d'un lien le nouveau degré interne des nœuds impactés par ce changement. En fonction de l'évolution de cette métrique, il est alors décidé d'ajouter ou retirer un nœud voir même de fusionner deux communautés. Li *et al.* [LHB⁺12] se base sur le nombre de liens que partage un nœud avec les communautés environnantes. Ainsi, un nœud est toujours dans la communauté avec laquelle il partage le plus de liens. Shanget *al.* [SLX⁺14], Cordeiro *et al.* [CSG16] et Sunet *al.* [SHZ⁺14] se basent sur l'évolution de la modularité. Cependant, ces approches ne permettent pas l'ensemble des évolutions de communauté possibles, en particulier l'apparition d'une nouvelle communauté. C'est pourquoi l'évolution de la structure courante peut mener à une structure ayant une faible qualité. Une autre approche a été proposée par Cazabet *et al.* [CAH10] afin d'améliorer l'évolution de la partition. Ils utilisent une métrique locale basée sur le nombre de chemins de longueur 2 existant entre un nœud et une communauté. Après chaque modification, ils considèrent également la possibilité de créer une nouvelle communauté sous la forme d'une petite clique. Ainsi, ils assurent une meilleure qualité de la partition au cours de l'évolution.

Flot de liens

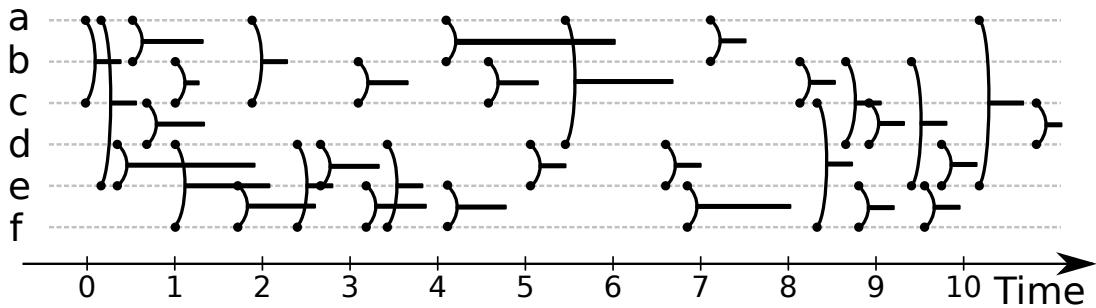


FIGURE 1.6 – Flot de lien entre 6 nœuds, représenté sur l’axe des ordonnées, au cours du temps représenté sur l’axe des abscisses. Dans l’exemple, il existe un lien entre a et b durant l’intervalle $[4, 6]$.

Dans les graphes temporels, toute l’information temporelle est gardée. Cependant, l’idéologie derrière cette méthode est qu’il existe une structure de graphe à chaque instant. Cette hypothèse n’est pas toujours vérifiée, en particulier lorsque les liens apparaissent et disparaissent très rapidement. C’est le cas des appels téléphoniques qui durent rarement plus d’une heure ou bien de manière plus frappante avec les SMS et les courriels qui n’ont même pas de durées. Dans ces contextes, il n’est pas possible d’assumer qu’à chaque instant une structure de graphe existe. Il faut donc un formalisme et des mesures qui s’adaptent à ce contexte. C’est pour répondre à ce besoin que le formalisme de flot de liens a été pensé. Le but est de construire un objet ne présupposant aucune structure et qui stocke toute l’information disponible. Plusieurs travaux [HS13, Hol15b, Hol15a] font un tour d’horizon des méthodes existantes pour étudier les flots de liens qui sont souvent appelés *temporal networks* mais il n’existe, à notre connaissance, qu’un seul travail donnant un fondement théorique solide au flot de liens. Il s’agit de Bataagel *et al.* [BP15] qui se basent sur l’algèbre. Malheureusement avec une formulation purement algébrique, il semble difficile de transcrire l’ensemble des notions de graphes pour l’instant.

Prenons l’exemple des appels téléphoniques, c’est-à-dire qui parle avec qui à quelle heure et pendant combien de temps. Un appel peut donc être représenté par un quadruplet (b, e, u, v) où b (resp. e) est le début (resp. la fin) de l’appel et u et v représentent des personnes. Il est donc possible de modéliser des appels téléphoniques par un ensemble de quadruplet. En faisant cela, aucune information n’est perdue et, en ce sens, flots de liens et graphes temporels sont équivalents. Cependant, ce changement de perspective induit une réflexion différente selon le formalisme considéré.

Les différences dans les méthodes de représentation des graphes temporels 1.5 et celle des flots de liens 1.6 illustrent bien ce changement de perspective, bien que les deux figures ne représentent pas le même réseau. Dans l’exemple de la figure 1.6, les nœuds sont représentés sur l’axe des ordonnées et le temps sur l’axe des abscisses. Un lien dans cette visualisation est représenté par : un arc verticale reliant deux axes de nœuds et un trait horizontal représentant la durée du lien. Ainsi dans l’exemple, il existe un lien entre les nœuds a et b dans l’intervalle $[4, 6]$.

Dans un graphe temporel, on abordera plus souvent les questions d’évolution de communautés de nœuds ou de l’importance d’un noeud. Dans un flot de liens, on s’intéressera plus souvent au temps nécessaire pour que deux nœuds soient de nouveau en contact ou à l’importance d’un lien. De manière très manichéenne et inexacte, le formalisme de graphe temporel pousse à étudier les nœuds et leur évolution alors que celui de flot de liens mets plus l’accent sur les liens.

Avec ce formalisme, la majorité des travaux sont encore descriptifs car ce type d'objet n'a jamais été étudié sous cet angle. Il existe de nombreux travaux étudiant le temps séparant l'apparition de deux liens pour un nœud [MSMA08, MSCA09]. Il semblerait que ces temps inter-contacts soient très hétérogènes et que de nombreuses connexions apparaissent dans un faible intervalle de temps suivie de longues durées sans activité. On parle alors de temps inter-contact *bursty*. Les effets des temps inter-contacts sur les phénomènes de diffusions et de marches aléatoires ont été très étudiés [KKP⁺11, KKBK12, SBBPS12, RB13]. Il semble cependant ne pas y avoir de conclusion définitive sur le sujet car la diffusion peut être accélérée ou ralentie par les temps inter-contacts selon la structure sous-jacente.

Il existe également quelques méthodes qui s'intéressent plus à la structure des flots de liens. Des études [KKK⁺11, KKKS13] s'intéressent à la présence de motif. Un motif dans un graphe est un sous-graphe comme le triangle qui est l'un des motifs les plus étudiés. Dans les flots de liens, le temps est également pris en compte dans les motifs. Il y a donc plusieurs variantes temporelles d'un même motif dans un graphe. Prenons l'exemple d'un chemin entre quatre nœuds A, B, C et D qui est représenté dans le graphe par $\{(A, B), (B, C), (C, D)\}$. Dans un flot de liens, il existe deux variantes à ce motif soit $\{(t_1, A, B), (t_2, B, C), (t_3, C, D)\}$ soit $\{(t_1, A, B), (t_2, C, D), (t_3, B, C)\}$ avec $t_1 < t_2 < t_3$. Dans un cas, une information peut être propagée au fur et à mesure de A vers D tandis que dans l'autre ce n'est pas possible. L'étude de la fréquence d'apparition de ces motifs dans le cas temporelle permet d'observer si le flot de liens a une structure particulière.

Détection de structures Il existe peu de méthodes détectant des structures décrivant l'ensemble d'un flot de liens. Rozenshtain *et al.* [RTG14] se penchent sur la détection de la zone la plus dense dans les flots de liens. Ils permettent de capturer un ensemble de nœuds et plusieurs intervalles de temps disjoints tel que ces nœuds sur ces intervalles aient le degré moyen le plus élevé dans le graphe agrégé sur ces intervalles. Bien que la mesure utilisée ne tienne pas directement compte du temps, leur méthode permet ainsi de mettre en évidence une partie du flot de liens.

Une méthode de type Stochastic Block Model a été proposée par Matias *et al.* [MRV15]. Elle est très proche de celle proposée par Corneli *et al.* [CLR16]. En effet, l'affiliation des nœuds est fixe et c'est l'activité d'une communauté qui change au cours du temps. La grosse différence ici est que l'activité varie de manière continue dans le temps. Ainsi l'apparition d'un lien dépend de la réalisation d'un processus de poisson non homogène qui change selon les communautés. Cependant, leur méthode ne permet pas de considérer les changements de communauté.

Résumé

Les graphes temporels et les flots de liens ne souffrent pas d'agrégation temporelle. Ils ont donc un pouvoir expressif plus important que les formalismes présentés précédemment. Formellement, flots de liens et graphes temporels sont équivalents dans le sens où un graphe temporel peut être représenté en un flot de liens et *vice versa*. En revanche, ils diffèrent dans le point de vue considéré. Dans un graphe temporel, il existe une structure de graphe représentant le réseau à chaque instant et les métriques de graphes sont pertinentes. Dans un flot de liens, il n'existe pas de structure pertinente à un instant donné et par conséquent les métriques de graphes ne sont pas pertinentes dans ce contexte. Cette différence implique d'utiliser des mesures différentes et, en particulier, de créer de nouvelle métrique pour les flots de liens. Cela explique notamment pourquoi il n'existe pour l'instant qu'assez peu de travaux traitant de la structure des flots de liens. Cette différence de perspective entraîne également un déplacement du centre d'intérêt. Dans les graphes temporels, on aura plutôt tendance à étudier les nœuds des graphes et leur évolution. À l'inverse, on s'intéresse plutôt aux liens et leur répartition dans les flots de liens.

Ainsi, les deux formalismes coexistent et répondent à des besoins différents.

1.4 Bilan

Dans un graphe statique, il existe de très nombreuses méthodes capturant une structure des nœuds soit via une partition soit via une couverture. L'abondance de méthodes existantes s'explique par la diversité des définitions de communautés existantes. Les structures capturées permettent de mieux comprendre l'organisation générale du graphe mais aussi de mieux comprendre le type d'un nœud.

Avec l'émergence de nouvelles données incorporant l'information temporelle, il est pertinent d'adapter la théorie des graphes pour prendre en compte le temps. L'extension temporelle des graphes est un champ de recherche très récent et il n'y a pas à douter que de nombreuses méthodes de détection de communautés dans ce contexte vont voir le jour. Cependant, tout les formalismes existants ne sont pas équivalents et ils limitent parfois les solutions possibles.

Il y a d'une part les formalismes se rapprochant de la théorie des graphes : séries de graphes, tenseurs 3d et graphes multicouches. Ces formalismes sont proches des graphes statiques et il est même possible d'y appliquer des méthodes statiques. En revanche, la prise en compte du temps n'est que partielle. Il y a toujours une forme d'agrégation temporelle et il n'est pas possible d'avoir une vision très fine de l'objet d'étude. De plus, les solutions existantes reposent sur le suivi de communautés qui ne semble pas être un problème résolu.

D'autre part, les formalismes de graphes temporels et de flots de liens capturent toute l'information temporelle. Ils ne souffrent donc pas de perte d'information. Ces deux formalismes, bien qu'équivalents, ne présupposent pas la même structure sur les données sous-jacentes. Dans un graphe temporel, les liens durent assez longtemps et par conséquent il existe une structure de graphe pertinente à chaque instant. Dans les flots de liens, les liens sont plus courts et il n'existe aucune structure de graphe à un instant donné. Par conséquent, ces deux formalismes répondent à des situations différentes. Par exemple, les études des temps inter-contacts sont très majoritairement conduites en utilisant le formalisme de flot de liens. Il semble donc plus aisés d'utiliser un flot de liens qu'un graphe temporel lorsque l'on souhaite étudier la structure des interactions.

Au cours de cette thèse, nous nous intéressons à la structure communautaire que peuvent former les liens dans le temps. Nous ne pouvons pas utiliser les formalismes utilisant une agrégation temporelle car l'identité du lien est perdue et le formalisme de graphe temporel considère des situations assez différentes de celles que nous étudions. C'est pourquoi le formalisme de flot de liens semble être le plus adapté pour étudier la structure des liens. Il n'existe que peu de travaux définissant formellement les flots de liens et traitant de leur structure. C'est pourquoi dans le chapitre 2, nous définissons plus formellement ce qu'est un flot de liens ainsi que les métriques utilisées tout au long de cette thèse avant de présenter nos travaux sur la structure des liens dans les chapitres suivants.

Chapitre 2

Flots de liens : extension temporelle des graphes

Sommaire

2.1	Définition	21
2.2	Sous-flots	22
2.3	Degré et densité	23
2.4	Liste des notations pour les flots de liens	25
2.5	Manipulation concrète des flots de liens	25

2.1 Définition

Nous avons décrit rapidement le formalisme de flot de liens dans le chapitre précédent. Nous nous attachons maintenant à définir plus formellement les flots de liens et quelques notions utilisées dans le reste de cette thèse. Un flot de liens est défini par un triplet $L = (T, V, E)$ où $T = [\alpha, \omega]$ est un intervalle de temps, V un ensemble de n nœuds et $E \subseteq T \times T \times V \times V$ un ensemble de m liens. Les liens de E sont des quadruplets (b, e, u, v) , signifiant que la paire de nœuds (u, v) est connectée sur l'intervalle $[b, e] \subseteq [\alpha, \omega]$. Nous dénotons la durée du flot par $\bar{L} = \omega - \alpha$. De manière analogue, la durée d'un lien $l = (b, e, u, v) \in E$ est notée $\bar{l} = e - b$. $\beta(E) = \min_{(b,e,u,v) \in E}(b)$ et $\psi(E) = \max_{(b,e,u,v) \in E}(e)$ sont respectivement l'apparition du premier lien et la disparition du dernier lien dans le flot de liens.

Nous considérons les flots de liens non orientés et sans boucle, *i.e.* $(b, e, u, v) = (b, e, v, u)$ et $u \neq v$. Enfin de manière analogue aux graphes et les multigraphes, nous définissons les flots de liens simples. Un flot de liens est simple si pour tout $l_1 = (b, e, u, v) \in E$ et $l_2 = (b', e', u, v) \in E$, $[b, e] \cap [b', e'] = \emptyset$ si $l_1 \neq l_2$. Dans les graphes, il est possible de transformer un multigraphe en graphe simple. Nous définissons également cette opération que nous nommons simplification : $\sigma(L)$. Afin de définir la simplification, nous nous aidons de la fonction de présence $\zeta_L(u, v, t)$ d'un flot de liens qui est égale à 1 si au moins un lien existe dans le flot L entre u et v à l'instant t et 0 sinon. $L' = (T', V', E') = \sigma(L)$ est la simplification de $L = (T, V, E)$ si et seulement si L' est simple, $T' = T$, $V' = V$ et si $\forall u, v \in V, \forall t \in T, \zeta_L(u, v, t) = \zeta_{L'}(u, v, t)$.

Il est parfois nécessaire d'augmenter la durée des liens. C'est notamment utile lorsque les liens sont de la forme (t, t, u, v) , ce qui est le cas lorsqu'on étudie des envoies de courriels par exemple. Nous notons $\xi(L, \Delta) = L'$ le fait d'augmenté de Δ la durée de chaque lien. Il y a plusieurs moyen d'augmenter de Δ un intervalle $[b, e]$. Nous considérons l'ajout symétrique c'est à dire qu'un intervalle $[b, e]$ est transformé en l'intervalle $[b - \Delta/2, e + \Delta/2]$.

Enfin, il est parfois intéressant d'agréger l'information temporelle pour créer un graphe statique. Un graphe $G = (V, E') = G(L)$ est le graphe agrégé d'un flot de liens si $\forall (u, v) \in E', \exists b, e \in T$ tel que $(b, e, u, v) \in E$.

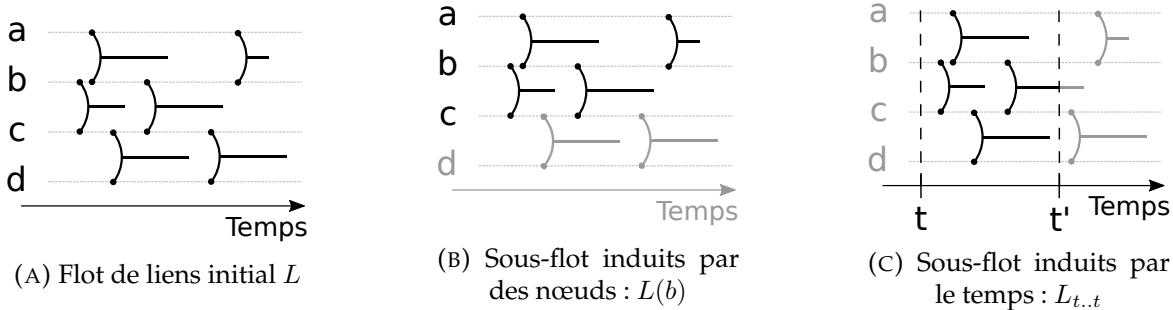


FIGURE 2.1 – Exemple de différents sous-flots, (B) et (C), du flot initial en (A). Les liens en noirs sont les liens sélectionnés dans le sous-flot.

2.2 Sous-flots

Dans les graphes, il existe la notion de sous graphes que nous étendons également aux flots de liens. Un flot de liens L' est un sous-flot de L , $L' \subset L$, si $\forall u, v \in V, \forall t \in T, \zeta_{L'}(u, v, t) \leq \zeta_L(u, v, t)$. Cette notion est en particulier utile pour définir des sous-flots induits par différent éléments. Nous illustrons ces notions dans les figure 2.1a 2.1b et 2.1b avec le flot initial dans la figure 2.1a.

Nous définissons $L(E')$, le sous-flot de L induit par un ensemble de liens $E' \subset E : L(E') = ([\beta(E'), \psi(E'), V(E'), E'])$.

Nous définissons $L(S)$, le sous-flot de L induit par un ensemble de paires noeuds $S \in V^2 : L(S) : L(S) = ([\beta(E'), \psi(E')], V', E')$ avec $E' = \{(b, e, u, v) \in E, (u, v) \in S\}$. Par convention, on note $L(v) = L(\{v\} \times V)$, le sous-flot induit par un nœud. Un exemple de sous-flot induit par un nœud est dans la figure 2.1b.

Enfin, nous définissons $L_{t..t'}$, le sous-flot de L induit par l'intervalle de temps $[t, t'] \subset [\alpha, \omega] : L_{t..t'} = ([t, t'], V, E')$ avec $E' = \{(b', e', u, v), \exists (b, e, u, v) \in E, b' = \max(b, t), e' = \min(e, t')\}$. Il est également possible de définir $L_{t..t'}$ via la fonction de présence : $\forall u, v \in V, \forall x \in [t, t'] \zeta_{L_{t..t'}}(u, v, x) = \zeta_L(u, v, x)$. Un exemple de sous-flot induit par un intervalle de temps est présenté dans la figure 2.1c. Il est intéressant de noter que le sous-flot $L_{t..t'}$ est équivalent au graphe statique à l'instant t , que nous notons $G(L_t)$.

Enfin, il est aussi possible de combiner ces notions. Par exemple avec $V' \subset V, L_{\alpha'..{\omega'}}(V'^2)$ est le sous-flot correspondant aux liens entre les nœuds de V' sur l'intervalle $[\alpha', \omega']$.

La construction d'un sous-flot induit par un ensemble de paires de nœuds peut être faite de manière linéaire au nombre de liens dans le sous-flot. Pour un ensemble de paires de nœuds $S = V_1 \times V_2$, il suffit d'itérer sur l'ensemble des liens qui sont reliés aux nœuds de V_1 et de vérifier que l'autre nœud du lien appartient à V_2 ce qui se fait en $O(\log(|V_2|))$ pour chaque lien. Cela peut être fait rapidement si chaque nœud a la connaissance des ces liens. Il faut tout de fois distinguer le cas où $S = V_1 \times V$ car il n'est alors pas nécessaire de faire la vérification d'appartenance à V .

Pour l'intervalle de temps, la situation est plus compliquée car il n'est pas facile de connaître l'ensemble des liens existent à un instant donné. Il est uniquement possible de les trier par ordre d'apparition ou de disparition. Il n'y a pas d'ordre total sur la présence qui permettrait de considérer uniquement les liens appartenant au sous-flot. Pour qu'un lien, $l = (b, e, u, v)$ appartienne au sous-flot temporel sur $[t, t']$, il doit remplir une des ces conditions, qui ne sont pas exclusives :

1. $t \leq b \leq t'$ le lien commence dans l'intervalle $[t, t']$.
2. $t \leq e \leq t'$ le lien fini dans l'intervalle $[t, t']$.
3. $b \leq t \wedge t' \leq e$ le lien couvre tout l'intervalle $[t, t']$.

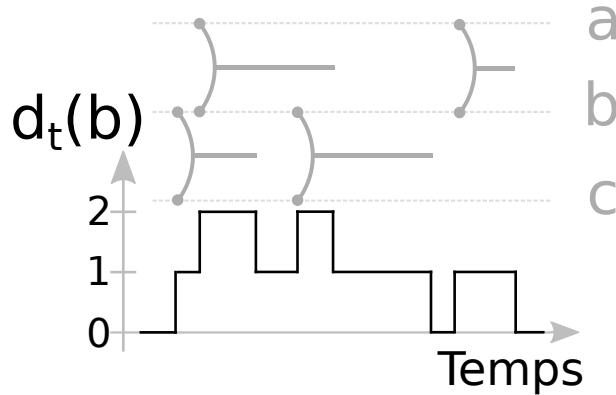


FIGURE 2.2 – Degré temporel du nœud b dans le flot de liens dans la figure 2.1a qui est également rappelé en grisé dans cette figure.

Avec ces conditions, on peut tirer deux conclusions. Pour qu'un lien $l = (b, e, u, v)$ puisse appartenir au sous-flot, il est nécessaire que $e \geq t$. Il n'est donc pas nécessaire de considérer les liens ayant un temps de fin inférieur à t . De manière analogue, il n'est pas nécessaire de considérer les liens ayant un temps de début supérieur t' . Cependant ces deux conditions ne peuvent pas être combinées pour restreindre l'espace de recherche. Une manière de construire le sous-flot temporel est d'itérer sur tous les liens dans l'ordre temporel d'apparition et d'arrêter le parcours dès qu'un lien a un temps de début supérieur à t' . Il faut cependant tester l'appartenance de chaque lien parcouru. De manière analogue, il est possible d'itérer sur les liens dans l'ordre inverse de disparition des liens et de s'arrêter dès qu'un lien a une temps de fin inférieur à t .

Dans le pire des cas, ces méthodes sont donc linéaire sur le nombre total de liens dans le flot et non dans le sous-flot comme précédemment. Il n'y a alors que deux optimisations possibles. Il est possible de choisir le sens du parcours si les deux ordres sont disponibles. Si la plus longue durée de liens, \bar{l}_{max} , est connue, alors il est possible de commencer la recherche à partir du premier lien respectant $b + \bar{l}_{max} \geq t$.

2.3 Degré et densité

Nous avons défini des outils pour manipuler et extraire un flot de liens. Nous nous intéressons maintenant à étendre quelques notions de connexités qui existe dans le graphe, en particulier le degré et la densité.

Il est assez trivial de définir le degré temporel d'un nœud u de la manière suivante :

$$d(t, u) = d_t(u) = |L_t(u)| = \sum_{v \in V} \zeta_L(u, v, t). \quad (2.1)$$

Le degré temporel n'est alors pas une simple valeur mais une fonction qui dépend du temps. Comme les liens apparaissent et disparaissent de manière instantanée, la fonction de degré est une fonction constante par morceaux. Un exemple de degré temporel est présenté dans la figure 2.2.

De manière analogue aux graphes, il est également possible de définir le degré temporel d'un ensemble de nœuds ou d'un ensemble de liens :

$$d_t(V') = \sum_{v \in V'} d_t(v) = 2|L_t(V'^2)| = 2|\{(b, e, u, v) \in E, u, v \in V', b \leq t \leq e\}|, \quad (2.2)$$

$$d_t(E') = 2|L_t(E')| = 2|\{(b, e, u, v) \in E', b \leq t \leq e\}|. \quad (2.3)$$

Avec ces définitions, il est aussi possible définir les notions de degré temporel interne, d_t^{in} , et externe d_t^{out} . Les degrés temporels d'un nœuds, d'un ensemble de nœuds ou de liens peuvent se calculer rapidement. Pour ce faire, il faut d'abord transformer les liens (b, e, u, v) en une suite de modifications de la forme $(b, u, v, +1)$ et $(e, u, v, -1)$ ce qui a une complexité en $O(m)$. Une fois cette liste créée, il suffit d'ordonner ces modifications dans l'ordre temporelle, $O(2m\log(2m))$, puis d'itérer sur l'ensemble des modification en sommant au fur et à mesure les apparitions et disparitions de lien, $O(2m)$.

Les degrés sont des fonctions du temps mais il est souvent intéressant de regarder la valeur moyenne du degré :

$$d_{t..t'}(u) = \frac{1}{t' - t} \int_t^{t'} d(u, t) dt = \sum_{l \in L_{t..t'}(u)} \frac{\bar{l}}{t' - t}. \quad (2.4)$$

Lorsque cela n'est pas ambigu, nous notons $d_{\alpha..\omega}(u) = d(u)$. Il est intéressant de noter dans cette formulation que si tout les liens durent tout au long du flot de liens alors le degré dans le graphe agrégé et le degré moyen dans le flot de liens sont égaux, c'est-à-dire $d_{\alpha..\omega}(u) = d_{G(L)}(u), \forall u \in V$.

À partir du degré, il est possible de définir beaucoup de notions différentes et notamment la densité. Pour rappel, la densité dans un graphe est définie par $\delta(G) = 2m/(n(n - 1)) = d(V)/n(n - 1)$ et est égale à la probabilité qu'il existe un lien entre 2 nœuds. Si l'on transpose l'idée au formalisme de flot de liens, la densité dans un flot de liens est la probabilité qu'il existe un lien entre 2 nœuds à un instant donné aléatoire. Cela se traduit par la formule suivante :

$$\delta(L) = \frac{2 \sum_{l \in E} \bar{l}}{n(n - 1)(\omega - \alpha)}. \quad (2.5)$$

Il se trouve que cette formulation est complètement équivalente à la densité moyenne des graphes $G(L_t)$:

$$\begin{aligned} \frac{1}{\omega - \alpha} \int_\alpha^\omega \delta(G(L_t)) dt &= \frac{1}{\omega - \alpha} \int_\alpha^\omega \frac{d(t, V)}{(n - 1)} dt = \frac{1}{\omega - \alpha} \int_\alpha^\omega \frac{\sum_{u \in V} d(t, u)}{n(n - 1)} dt = \frac{\int_\alpha^\omega \sum_{u \in V} d(t, u) dt}{n(n - 1)(\omega - \alpha)} \\ &= \frac{\sum_{u \in V} \int_\alpha^\omega d(t, u) dt}{n(n - 1)(\omega - \alpha)} = \frac{\sum_{u \in V} \sum_{l \in L_{\alpha..\omega}(u)} \bar{l}}{n(n - 1)(\omega - \alpha)} = \frac{\sum_{u \in V} \sum_{l \in L(u)} \bar{l}}{n(n - 1)(\omega - \alpha)} = \frac{2 \sum_{l \in E} \bar{l}}{n(n - 1)(\omega - \alpha)}. \end{aligned}$$

Pour arriver à ce résultat, nous utilisons la relation entre degré temporel moyen et somme des durées de l'équation 2.4 et le fait que la somme des degrés soit égale à deux fois le nombre de liens.

La notion de densité que nous avons définie est donc cohérente avec notre notion de degré et traduit le même concept que dans les graphes. Ainsi, la densité dans les flots de liens est aussi comprise entre 0 et 1. Enfin, cette formulation de densité est une généralisation de la densité proposée par Viard *et al.* [VL14] qui ne considérait que les liens sans durée.

Le calcul de la densité d'un flot est linéaire en le nombre de liens car elle est dépendante du calcul de $d_{\alpha..\omega}(V)$ qui est fait de manière linéaire.

2.4 Liste des notations pour les flots de liens

Symbol	description
L	Flot de liens
T	intervalle de temps
V	ensemble de nœuds
E	ensemble de liens : (b, e, u, v)
n	nombre de nœuds
$ L , E $	nombre de liens dans le flot
$\beta(E)$	temps d'apparition du premier lien
$\psi(E)$	temps de disparition du dernier lien
$\xi(L, \Delta)$	Flot de liens où chaque lien dure Δ
$\sigma(L)$	Simplification du flot de liens L
$L(V'^2)$	sous-flot induits par les nœuds de V'
$L_{t..t'}$	sous-flot induits par l'intervalle $[t, t']$
$d_t(v)$	degré de v à l'instant t
$d_t(V)$	Somme des degrés des nœuds dans V à l'instant t
$d_{t..t'}(v)$	degré moyen de v sur $[t, t']$
$d(v)$	degré moyen v sur T
$\delta(L)$	densité du flot

TABLE 2.1 – Liste des notations pour les flots de liens

2.5 Manipulation concrète des flots de liens

Nous avons défini formellement quelques notions pour les flots de liens. Afin de manipuler ces notions simplement, nous avons mis en place une librairie capable de les calculer simplement. Le but est de fournir une implémentation généraliste qui soit simple d'accès. Ainsi, il sera possible à n'importe qui de calculer ces notions. La démarche, bien que beaucoup plus modeste, est similaire à ce qui est fait avec les graphes et networkx¹.

L'implémentation est en C++ pour être rapide avec un export en *python* pour faciliter l'utilisation. Le code de cette implémentation est ligne² et la documentation également³. Il est par exemple possible avec la librairie de lire un flot de liens et de calculer la densité moyenne d'un sous-ensemble de nœuds sur un intervalle arbitraire. Comme le but de cette librairie est d'être généraliste, l'implémentation n'est pas optimisée en espace pour un calcul spécifique. Ainsi il est aisément d'étendre la librairie en écrivant un nouveau calcul.

Un intérêt de cette implémentation est de permettre, en python, la génération de visualisation⁴, voir le dessin dans la figure 2.3. Parmi les possibilités offertes par la visualisation, il est possible de n'afficher qu'une partie des nœuds ou de ne garder qu'un sous intervalle de temps. De plus, il est possible de choisir la couleur des liens de manière manuelle ou en fonction d'une partition de liens.

La lisibilité de ce genre de visualisation est très dépendant de l'ordre attribué aux nœuds sur l'axes des ordonnées. C'est pourquoi avec notre outil, il est également possible de fixer un ordre arbitraire. Comme il peut être fastidieux d'écrire un ordre, nous avons implémenté un algorithme rudimentaire pour améliorer l'ordonnancement des nœuds dans la visualisation.

1. <https://networkx.github.io/>

2. XXX

3. XXX

4. Pour l'instant, uniquement un export en svg est possible.

A Mettre

A Mettre

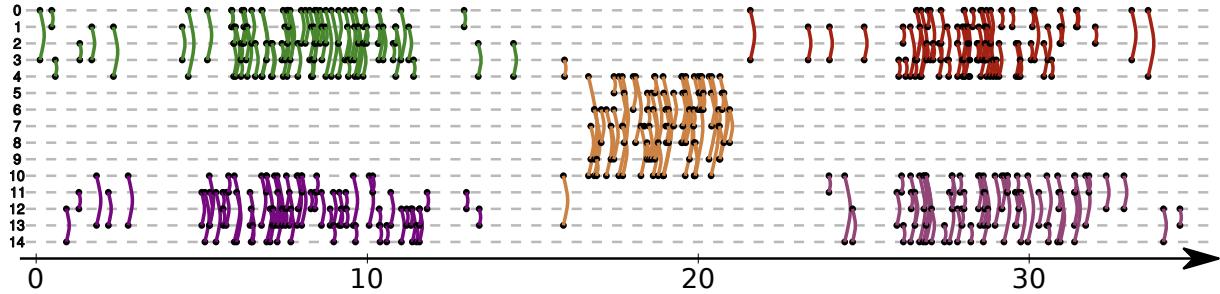


FIGURE 2.3 – Flot de lien entre 6 nœuds, représenté sur l’axe des ordonées, au cours du temps représenté sur l’axe des abscisses. Dans l’exemple, il existe un lien entre a et b durant l’intervalle $[4, 6]$.

Avant de pouvoir améliorer une visualisation, il est nécessaire de pouvoir quantifier la complexité de la visualisation actuel. Empiriquement, on se rend vite compte que ce sont les long traits verticaux qui rende pénible la lecture et qu’il faut limiter.

C’est pourquoi nous décidons d’évaluer une ordonnancement en fonction de la somme des longueurs des traits représentant les liens. Avec la fonction d’ordre $Ordre : V \mapsto \mathbb{N}$, trouver le meilleur ordonnancement se résume à résoudre :

$$Ordre^* = \arg \min_{Ordre} \sum_{(b,e,u,v) \in E} |Ordre(u) - Ordre(v)| \quad (2.6)$$

Malheureusement, trouver l’optimum ne semble pas trivial et il n’est pas envisageable de tester l’ensemble des ordre. C’est pourquoi, nous utilisons un algorithme probabiliste qui teste certain nombre de permutations aléatoire. Une permutation est appliquée si elle améliore l’évaluation de la visualisation. Il s’agit bien sur d’une première approche qu’il est possible d’améliorer.

Une piste possible pour améliorer cette approche naïve serait la construction d’un ordre potentiellement proche de l’optimum. Pour y arriver, il serait intéressant de placer côté à côté les nœuds qui partagent le plus de liens et ainsi construire itérativement une première solution.

Chapitre 3

Étude de la structure d'une archive de courriels en tant que flot de liens

Sommaire

3.1	Prétraitement sur le jeu de données	27
3.2	Caractéristiques élémentaires des discussions	28
3.3	Étude des discussions en tant que sous-flots	30
3.3.1	Application de la Δ -densité	30
3.3.2	Répartition temporelle et structurelle des discussions	33
3.3.3	Flot quotient	35
3.3.4	Conclusion	36
3.4	Détection de structures denses	37
3.4.1	Méthode de détection	37
3.4.2	Comparaison des partitions	38
3.4.3	Conclusion	40

Nous nous intéressons ici à une archive de courriels publiquement disponibles¹. Cette archive contient l'ensemble des courriels échangés par différent utilisateurs pour résoudre un problème survenu lors de l'utilisation de Debian. Typiquement, une personne ayant un problème lors de l'installation envoie un courriel à la liste afin de demander de l'aide. Toute personne inscrite sur la liste reçoit ce courriel et peut y répondre ce qui donne lieu à une discussion visible par tous. Ces discussions ont déjà été étudiées dans le passé [DLCA07, SSA06, WWL⁺14] mais cela a été fait en utilisant des méthodes statiques uniquement.

Or, ces données se représentent naturellement sous forme de flot de liens en associant chaque personne à un nœud et chaque courriel entre deux personnes à un lien dans le flot de liens à l'instant où le courriel a été envoyé. L'avantage de ces données de communications est que nous connaissons la discussion (*thread*) dans laquelle a lieu chaque message. Une discussion est un ensemble de courriels dont tout les messages répondent à un message précédent de la discussion excepté pour le premier qui a initié la discussion et que nous appelons *racine*. Ainsi, nous étudions la structure des discussions dans le flot liens représentant les courriels envoyés sur la liste.

Utiliser le formalisme de flot de liens est particulièrement intéressant car cette lite de diffusion existe depuis 1994. L'aspect temporel des discussions est donc important.

3.1 Prétraitement sur le jeu de données

Bien qu'accessible sur internet, ce jeu de données nécessite un ensemble de traitements avant de pouvoir exploiter les 724985 courriels que contenait l'archive en janvier 2015. Tout

1. <https://lists.debian.org/debian-user/>

d'abords, les données ne sont pas sous la forme d'un flot de liens avec la structure des conversations. Les données sont accessibles via le site internet et ne sont pas structurées. Pour avoir ces informations sous la forme d'un flots de liens, un script d'extraction a été développé [URL](#). Lors de l'extraction, 2269 courriels n'ont pas pu être pris en compte car certaines informations étaient manquantes ou mal formées, typiquement à cause de la date ou d'un fuseau horaire non reconnu.

Une fois les informations brutes récupérées, il faut les transformer en un flot de liens cohérent. Pour chaque message m , nous extrayons son auteur $a(m)$, l'instant $t(m)$ auquel le message a été posté², le message auquel il répond $p(m)$ trouvé via le champ IN-REPLY-TO, son destinataire $a(p(m))$ et la discussion $D(m)$ dans laquelle il apparaît. Comme les messages *racines* ne répondent à aucun autre, nous imposons $p(m) = m$. L'ensemble de liens du flot est donc $\{(t(m), a(m), a(p(m)))\}_m$. Nous ne prenons pas en compte la direction des liens.

Une fois le flot créé, il est encore nécessaire de vérifier sa cohérence. Un message peut être filtré pour différentes raisons : le courriel apparaît avant le message auquel il est censé répondre, le message auquel il répond n'est pas présent dans l'archive, l'auteur et le destinataire sont la même personne. Cette dernière condition permet notamment d'éviter la présence de boucles dans le flot. Cela concerne principalement les *racines*. Il s'agit de vérifications simples auquel il faut ajouter les vérifications sur la cohérence de la structure des discussions. Ainsi, une discussion est entièrement retirée du jeu de données s'il manque la *racine*, si un de ses messages a été retiré à l'étape précédente. Après ces vérifications, environ 7% des discussions sont retirées. À cela, il faut également tenir compte de notre temps d'observation qui est partiel. En effet, une discussion dont le dernier message a lieu 1 semaine avant la fin de la capture peut ne pas être terminée. De même, une discussion durant très longtemps n'a qu'une faible probabilité d'être capturée en entier. Pour corriger ces biais, nous filtrons également les discussions ayant débutées trop récemment ou durant trop longtemps. La limite pour considérer une discussion trop récente ou trop longue a été fixé à 4 ans (1.26×10^8 s) car nous avons constaté qu'uniquement quelques discussions dépassées ce seuil sur la distribution de durées des discussions dans la figure 3.1. Toutes les discussions qui ont débutées moins de 4 ans avant la capture du jeu de données ne sont donc pas prises en compte.

Une fois tout ces messages filtrés, nous obtenons un flot de liens avec 316 569 liens entre 34 648 personnes pendant presque 19 ans et 116 999 discussions. Mis à part les 237 664 messages de début de discussion, ce sont 168 482 courriels qui ont été filtrés soit environ 23%. La majorité des courriels filtrés l'ont été car ils appartiennent à une discussions trop récente.

3.2 Caractéristiques élémentaires des discussions

Les caractéristiques les plus élémentaires des discussions sont le nombre de courriels, le nombres de personnes, le nombre de paires de personnes distinctes en interaction directes et leur durée. Dans la figure 3.1, sont présentées les distributions cumulatives inverses de ces quantités et on remarque qu'elles sont toutes hétérogènes. On remarque que les données filtrées ne diffèrent pas qualitativement des données brutes.

La distribution des durées des discussions montre que la majorité des discussions dure environ une journée ou moins (10^5 secondes équivaut à moins de 28 heures). Par ailleurs, on remarque qu'il n'existe que quelques discussions durant plus d'un an.

Ces premières observations sont nécessaires mais pas suffisantes pour comprendre les caractéristiques d'une discussion. Nous avons également étudié la corrélation entre ces différentes notions et une partie d'entre elles sont présentées dans la figure 3.2.

La corrélation entre la durée et le nombre de courriels, dans la figure 3.2 partie gauche, met en évidence que plus une discussion est grande en nombre de courriels plus elle dure

2. Cet instant est convertit en *timestamp* en tenant compte des fuseaux horaires.

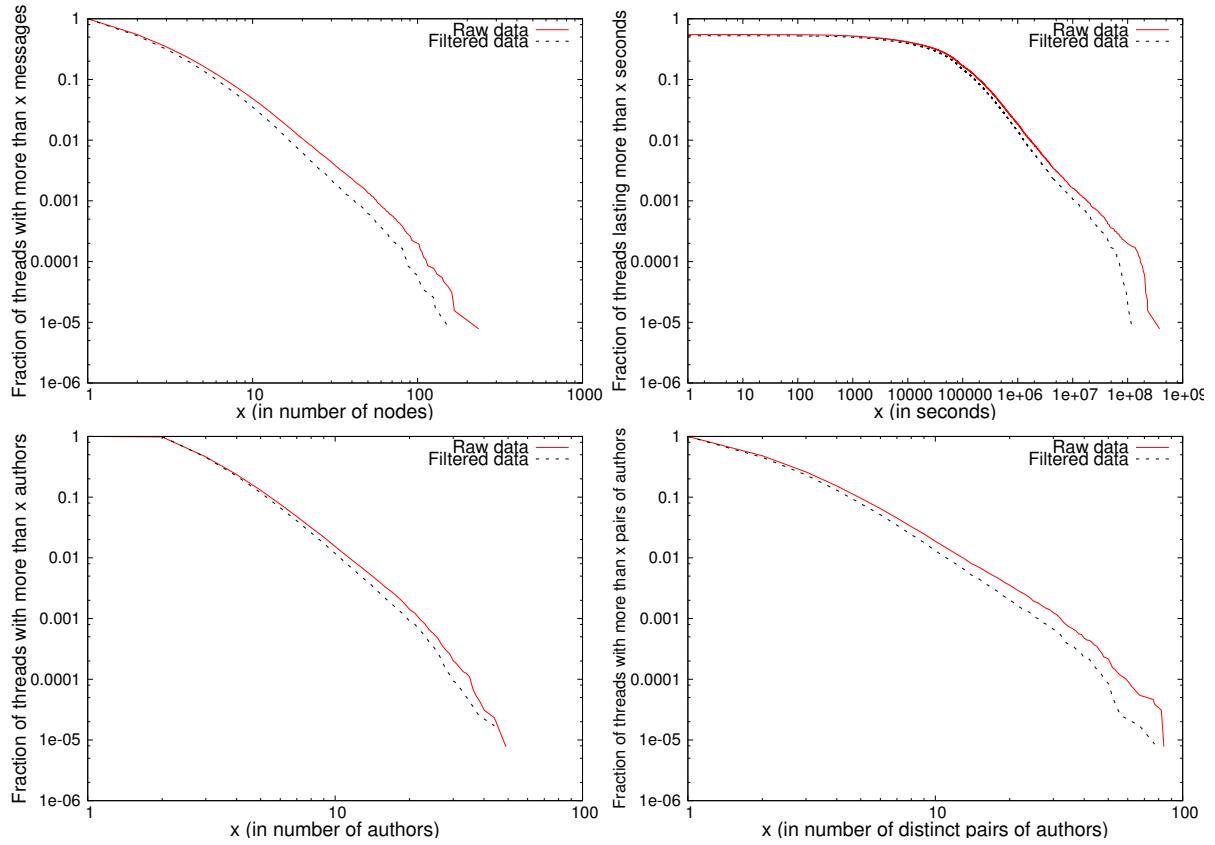


FIGURE 3.1 – Distribution cumulative inverse de différentes caractéristiques pour les données brutes (ligne pleine) et filtrées (ligne pointillé). En haut à gauche : nombre de courriels dans une discussion ; en haut à droite : durée d'une discussion ; en bas à gauche : nombre de personnes dans une discussion ; en bas à droite : nombre de paires d'auteurs distinct dans une .

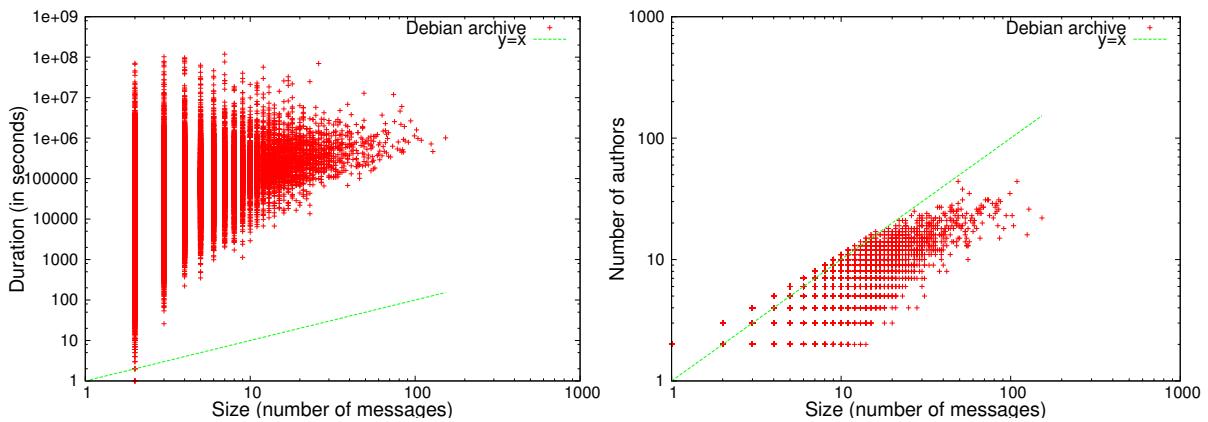


FIGURE 3.2 – Gauche : Corrélation entre le nombre de courriels et la durée d'une discussion. Droite : Corrélation entre le nombre de courriels et le nombre d'auteurs dans une discussion.

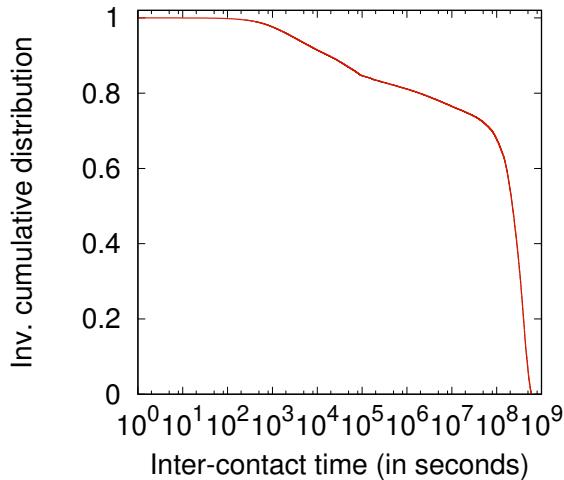


FIGURE 3.3 – Distribution des temps inter-contacts dans le fil de discussions.

longtemps, ce qui est attendu. Par contre, on observe que les petites discussions ont des durées très variables. Dans la partie droite de la figure 3.2 présentant la corrélation entre le nombre de courriels et d'auteurs, on observe un autre fait attendu [DLCA07] qui est qu'une discussion est constituée, en général, de plus de messages que de participants. Ainsi lors d'une discussion, c'est un petit nombre de personnes qui échangent potentiellement beaucoup de messages.

Enfin, il est intéressant d'observer la dynamique des échanges entre deux personnes. Soit $\tau(u, v) = (t_{i+1} - t_i)_{i=0..k+1}$ la séquence des temps inter-contacts des k liens entre les nœuds u et v , où t_0 est le temps entre α et le premier lien et t_{k+1} est le temps entre le dernier lien et ω . Il s'agit du temps écoulé avant que deux personnes se contactent à nouveau, indépendamment peu importe la conversation. Dans la figure 3.3 est représentée la distribution cumulative inverse du temps inter-contacts. 21% des temps inter-contacts sont inférieurs à 30 jours (2.6×10^6 s). Ce chiffre bien que relativement faible est tout de même important car il s'agit de discussions ouvertes où tout le monde peut participer. En particulier, une personne peut envoyer une demande d'aide à un moment donné et ne plus jamais échanger avec les mêmes personnes. Or, on observe que 21% des contacts sont renouvelés en moins de 30 jours. La participation est donc relativement élevée.

3.3 Étude des discussions en tant que sous-flots

3.3.1 Application de la Δ -densité

Jusqu'à maintenant aucune notion intrinsèquement liée aux flots de liens n'a été utilisée pour caractériser les discussions. Le but est d'évaluer si cette structure de flot peut se rapprocher d'une structure communautaire. Comme dit précédemment, les communautés sont souvent définies comme étant des structures devant être densément connectées. C'est pourquoi nous nous attachons à étudier la densité des discussions.

Comme ces données se modélisent par un flot de liens où les liens n'ont pas de durée, nous étudions la densité non pas dans le flot de liens initial mais dans $\sigma(\xi(\Delta, L))$ pour différentes valeurs de Δ entre 1 seconde et 20 ans. La notation $\delta(\sigma(\xi(\Delta, L)))$ est cependant très lourde et nous simplifions par $\delta_\Delta(L)$ et nous parlons donc de Δ -densité. Tout d'abord dans la figure 3.4 est représentée la Δ -densité globale du flot. En couvrant un spectre aussi large de Δ , on observe que la Δ -densité est croissante avec Δ mais surtout on observe bien la convergence de Δ -densité vers 3.139×10^{-4} , la densité du graphe agrégé, lorsque Δ est proche de $\omega - \alpha$.

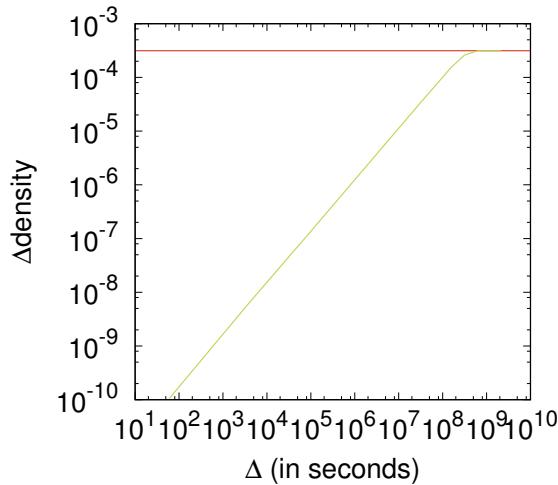


FIGURE 3.4 – Évolution de la Δ -densité (en vert) du flot de liens pour Δ de 60 seconde à 20 ans. En rouge, la densité dans le graphe agrégé.

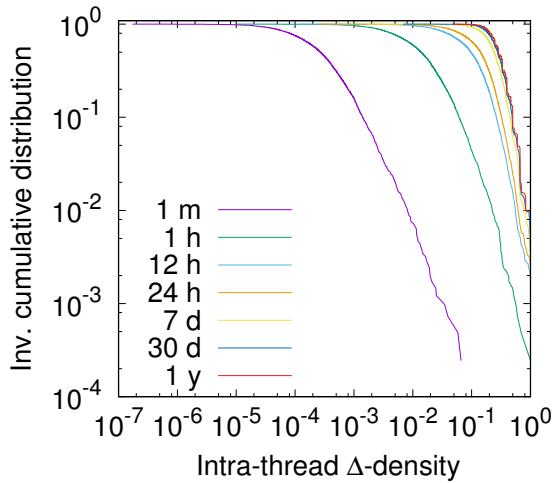


FIGURE 3.5 – Distribution cumulative inverse de la Δ -densité des discussions pour différentes valeurs de Δ s.

Cependant, la Δ -densité du flot n'apporte que peu d'informations en elle-même. Elle est surtout utile pour comparer les valeurs de Δ -densité des sous-flots que sont les discussions, c'est-à-dire $\delta(\sigma(\xi(\Delta, D_i)))$. Ainsi dans la figure 3.5, est présentée la distribution cumulative inverse de la Δ -densité des discussions pour différentes valeurs de Δ . On remarque que les différentes valeurs de Δ ne semblent pas influencer qualitativement la distribution de Δ -densité. Cette courbe met surtout en évidence que les discussions sont des structures beaucoup plus denses que le flot. En effet, la densité médiane des discussions est, selon la valeur de Δ , entre 2.69×10^{-4} et 0.28 alors que le flot a une Δ -densité variant entre 1.05×10^{-10} et 3.42×10^{-5} . La Δ -densité des discussions est donc en moyenne 10⁵ fois plus élevée que celle du flot. Bien que notable, ce fait est attendu notamment car le flot dure beaucoup plus longtemps et concerne beaucoup plus de nœuds que les discussions.

Afin d'aller plus loin dans l'étude de cette structure, il faut revenir à une définition plus précise de ce qu'est une bonne communauté. En soi, une valeur de densité n'est pas suffisante pour définir une structure communautaire. En effet, une discussion ayant une densité de 0.8 peut ne pas être une communauté tandis qu'une autre ayant une densité proche de zéro peut être une communauté. Il faut définir un point de comparaison pour effectivement affirmer

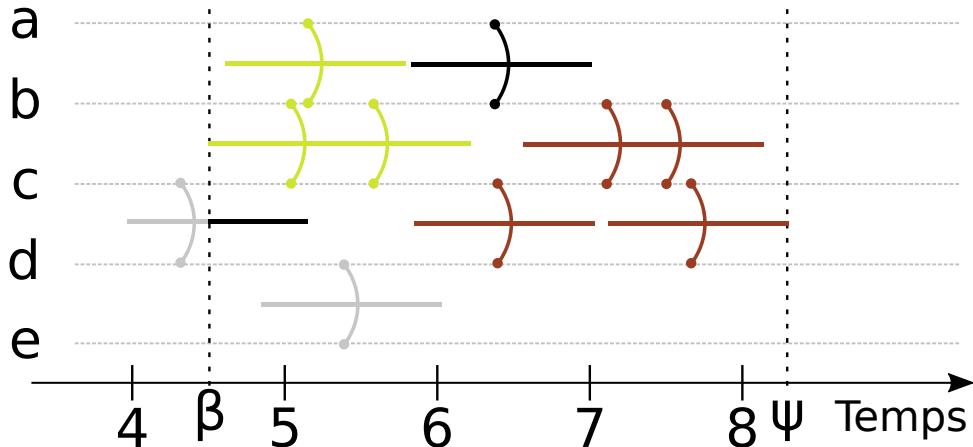


FIGURE 3.6 – Le flot entre les discussions verte et rouge est constitué des liens ou partie de liens en noir. Les liens en gris ne sont pas pris en compte.

qu'une structure est particulièrement dense. La prise en compte de la densité globale est un début mais n'est pas suffisante.

Une autre définition d'une communauté est qu'elle devrait être plus densément connectée à l'intérieur qu'avec les autres communautés adjacentes. Pour un graphe $G = (V, E)$ et une communauté C_i de la partition $C = \{C_j\}_{j_1..k}$ de V en k communautés, cela se traduit par le calcul de la densité entre les communautés, $\delta^{inter}(C_i)$:

$$\delta^{inter}(C_i) = \frac{1}{|C| - 1} \sum_{j, i \neq j} \frac{|\{(u, v) \in E \text{ t.q. } u \in C_i \text{ and } v \in C_j\}|}{|C_i| \cdot |C_j|}. \quad (3.1)$$

Il s'agit tout simplement de la probabilité qu'un lien existe entre un nœud de C_i et un nœud d'une autre communauté. Encore une fois, cette notion n'a pas de sens direct dans le formalisme de flot de liens et il est nécessaire de l'adapter. Pour ce faire, nous définissons la Δ -densité inter discussions entre deux discussions D_i et D_j : $\delta_\Delta^{inter}(D_i, D_j)$. Soit $E_\Delta = \xi(\Delta, E)$ et $L_{inter}(D_i, D_j) = (T', V', E')$ avec $V' = V(D_i \cup D_j)$, $T' = [\beta(D_i \cup D_j), \psi(D_i \cup D_j)]$, et $E' = E_\Delta \setminus (D_i \cup D_j)$. Avec ces définitions, $\delta_\Delta^{inter}(D_i, D_j)$ est égale à $\delta_\Delta^{inter}(D_i, D_j) = \delta(L_{inter}(D_i, D_j))$. Il s'agit donc de la densité du flot inter discussions qui est constitué des liens entre les nœuds induits par D_i et D_j qui n'appartiennent ni à D_i ni à D_j . Dans la figure 3.6, un exemple de flot inter discussion est représenté.

Afin d'obtenir la Δ -densité inter discussions entre D_i et tout les autres discussions, nous utilisons la moyenne des densité inter discussion entre D_i et les autres discussions, soit :

$$\delta_\Delta^{inter}(D_i) = \frac{1}{|C| - 1} \sum_{j, i \neq j} \delta_\Delta^{inter}(D_i, D_j). \quad (3.2)$$

La distribution cumulative inverse de la Δ -densité inter discussions est présentée dans la figure 3.7 pour différentes valeurs de Δ . Bien que similaire, le comportement de la Δ -densité inter discussions diffère qualitativement de celui de la Δ -densité. La Δ -densité inter discussions croît également en fonction de Δ mais il y a toujours une différence notable entre $\Delta = 1 \text{ mois}$ et $\Delta = 1 \text{ an}$ ce qui n'est pas le cas pour la Δ -densité. Cette différence est normale car lors du calcul de Δ -densité le nombre de liens considérés est fixe peut importe Δ alors qu'il croît avec Δ lors du calcul de Δ -densité inter discussions. Cet effet est visible dans la figure 3.6. Le lien (c, d) qui apparaît peut avant β n'est pas pris en compte si Δ est proche de 0 alors qu'il est en parti pris en compte lorsqu'un Δ plus grand est considéré, ce qui est le cas dans la figure.

Un autre facteur est aussi la durée considérée qui est plus longue que la durée des discussions.

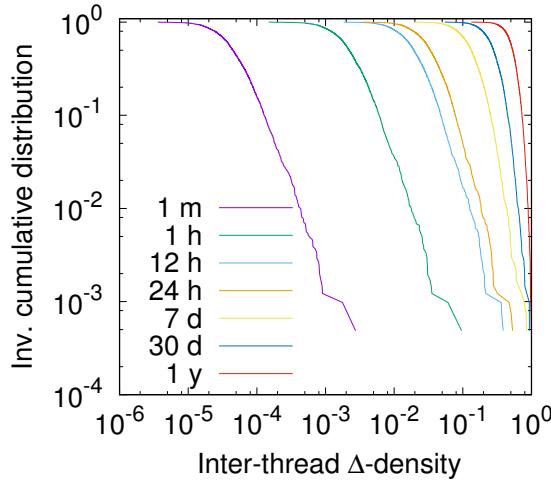


FIGURE 3.7 – Distribution cumulative inverse de la Δ -densité inter discussions pour différentes valeurs de Δ s.

Afin de comparer plus aisément Δ -densité et Δ -densité inter discussions, la corrélation entre ces deux mesures est présentée dans la figure 3.8 pour différentes valeurs de Δ . On remarque que les discussions sont effectivement plus denses intérieurement qu’avec les autres discussions. La différence est de plusieurs ordres de grandeur lorsque Δ est petit et elle diminue lorsque Δ croît. Pour $\Delta = 20 \text{ ans}$ dans la figure 3.8d, la différence n’est plus visible car à cette échelle de temps, l’ancrage temporel des discussions n’est plus décisif. On remarque tout de même que pour $\Delta = 1 \text{ an}$, la différence reste notable.

3.3.2 Répartition temporelle et structurelle des discussions

Nous avons étudié la densité des discussions et entre les discussions mais il est également intéressant d’observer comment ces discussions sont réparties topologiquement et temporellement. Pour étudier la répartition des discussions dans le temps, nous construisons un graphe d’intervalle [LB62] $X = (V_X, E_X)$ représentant le chevauchement temporel. Chaque discussion du flot devient un nœud de V_X et le lien (i, j) existe dans E_X si les discussions D_i et D_j correspondantes ont eu lieu au même instant, i.e. $[\alpha_i, \omega_i] \cap [\alpha_j, \omega_j] \neq \emptyset$. De manière similaire, nous définissons le graphe de chevauchement topologique $Y = (V_Y, E_Y)$. Les nœuds de ce graphe représentent encore une fois les discussions du flot et un lien existe entre deux discussions si au moins une personne a participé aux deux, i.e. $V(D_i) \cap V(D_j) \neq \emptyset$.

Ces deux graphes sont constitués de 116 999 nœuds et d’environ 2 millions de liens pour le graphe de chevauchement temporel et d’environ 63 millions de liens pour le graphe de chevauchement topologique. Par construction, ces graphes contiennent beaucoup d’informations sur les relations entre les discussions.

Dans la figure 3.9(gauche), est représentée la corrélation entre le degré d’une discussion dans le graphe de chevauchement temporel X et sa durée. Il y a une corrélation évidente entre ces deux notions lorsque les discussions ont une durée supérieure à 10^5 secondes. Plus une discussion dure longtemps, plus elle a de chance d’avoir lieu en même temps que beaucoup d’autres discussions. On observe également que, même pour les discussions durant moins d’un jour (8.6×10^4 s), il peut y avoir jusqu’à une centaine d’autres discussions actives sur la même période.

La figure 3.9(droite) présente la corrélation entre le degré d’une discussion dans le graphe de chevauchement topologique Y et son nombre de participants. La corrélation est moins nette mais il y a tout de même une tendance. Par contre, on remarque de manière frappante que

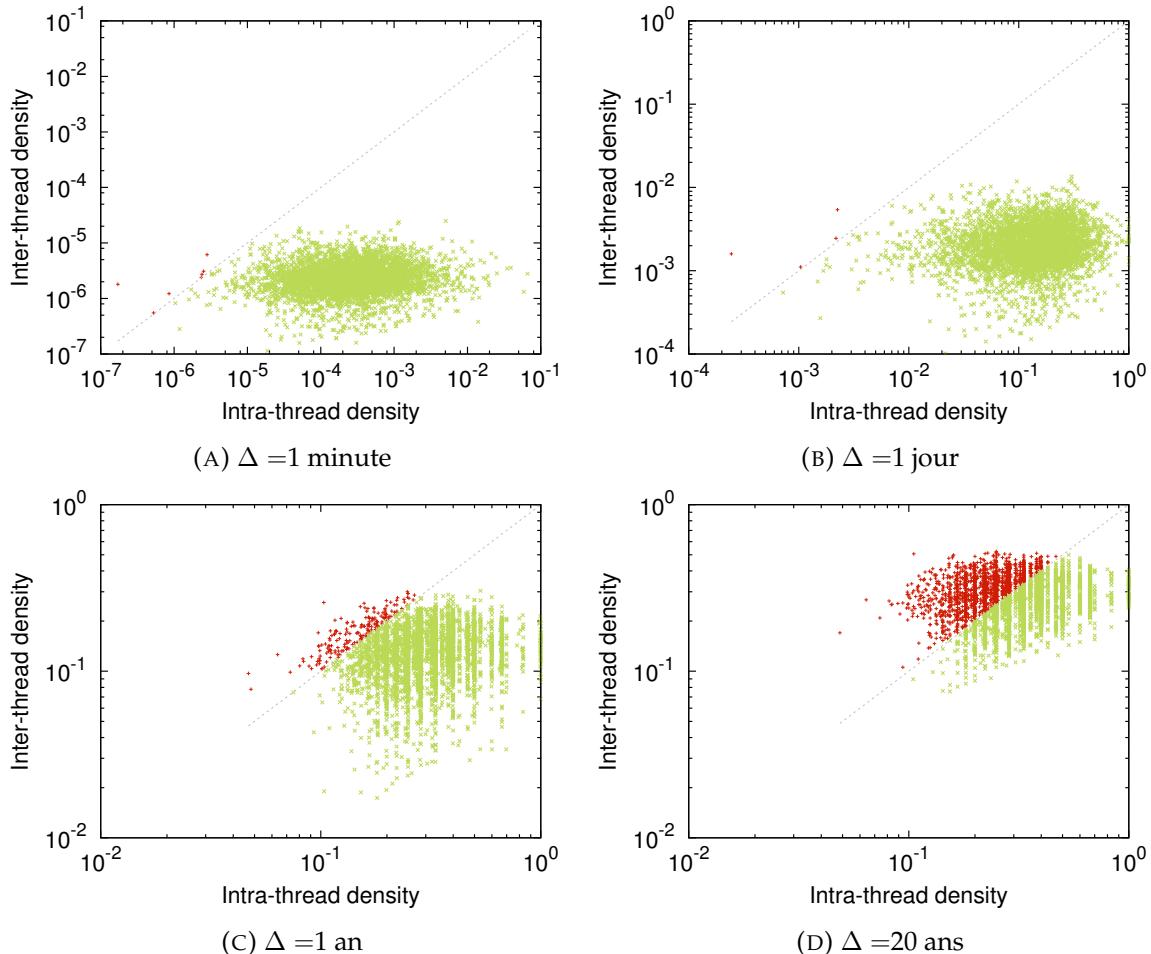


FIGURE 3.8 – Corrélations entre Δ -densité et Δ -densité inter discussions pour différentes valeurs de Δ . Une discussion est en vert (resp. rouge) si elle a une Δ -densité plus (resp. moins) élevée que sa Δ -densité inter discussions.

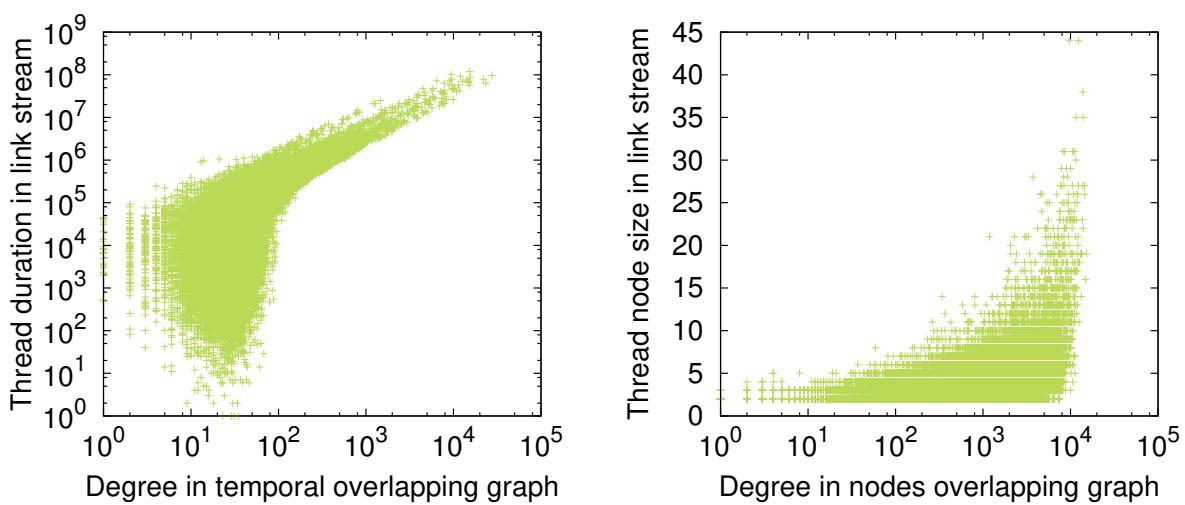


FIGURE 3.9 – Gauche : Corrélation entre le degré des discussions dans le graphe de chevauchement temporel et leur durée. Droite : Corrélation entre le degré des discussions dans le graphe de chevauchement topologique et leur nombre de participants.

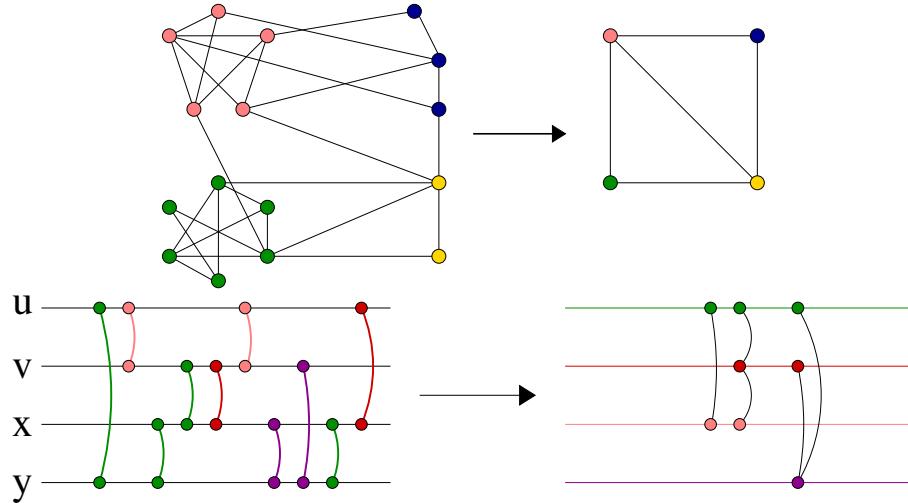


FIGURE 3.10 – Haut : Exemple de graphe ayant une structure communautaire et son graphe quotient associé. Bas : Exemple d'un flot de lien avec une structure ainsi que son flot quotient associé.

même une petite discussions peut partager des nœuds avec les énormément d'autres discussions.

3.3.3 Flot quotient

Le graphe quotient est une autre notion clef pour étudier les relations entre les communautés d'un graphe $G = (V, E)$. Soit une partition $C = \{C_i\}_{1..k}$ des nœuds de G en k communautés, chaque communauté est représentée dans le graphe quotient \bar{G} par un nœuds dans V . Il y a un lien entre deux communauté C_i et C_j dans E si il existe au moins un lien entre un nœuds de C_i et un nœuds de C_j . Voir une illustration sur la figure 3.10. Il est possible d'ajouter un poids sur les liens de \bar{G} égale au nombre de liens reliant les communautés. Le graphe quotient permet de facilement étudier, dans un graphe, les relations entre les communautés.

Nous étendons ici cette notion de graphe quotient aux flots de liens. Nous définissons le flot quotient, $Q = (T_Q, V_Q, E_Q)$, induit par une partition $P = \{P_i\}_{1..k}$ en k sous-flots de la manière suivante. Chaque sous-flot P_i est représenté par un nœud dans V_Q . Il existe un lien (t, P_i, P_j) dans E_Q si il existe $(t_1, u, v) \in P_i$, $(t_2, u, v') \in P_j$ et $(t_3, u, v'') \in P_i$ avec $t_1 \leq t_2 \leq t_3$. En d'autre termes, il y a un lien dans E_Q si un nœud u a un lien dans P_j qui apparaît entre deux autres de ses liens du groupe P_i .

Le flot quotient induit par les discussions dans le jeu de données contient 12 281 269 liens impliquant 68 524 discussions différentes. Comme le jeu de données contient 116 999 discussions, il y a donc 48 475 discussions sans lien et qui ne seront pas prises en compte par la suite. Ce nombre de discussions non-reliées est élevé comparé à ce qui est obtenu dans un graphe. En effet dans un graphe, un nœud de degré 0 correspond à une communauté qui est une composante connexe (ou un union de composantes connexes). En ajoutant l'information temporelle, les discussions sont séparées par le temps dans flot. C'est pourquoi un grand nombre de discussions n'ont pas de liens dans le flot quotient. Ce phénomène est d'autant plus vrai pour les petites discussions.

Il faut aussi noter qu'il y a environ 20 fois plus de liens dans le flot quotient que dans le flot initial. Cela est normal car un lien dans le flot peut donner lieu à plusieurs liens dans le flot quotient. Ce cas est visible dans la figure 3.10. Le lien (x, y) du groupe violet du flot à gauche donne lieu au lien (*violet, rouge*) et au lien (*violet, vert*) dans le flot quotient à droite.

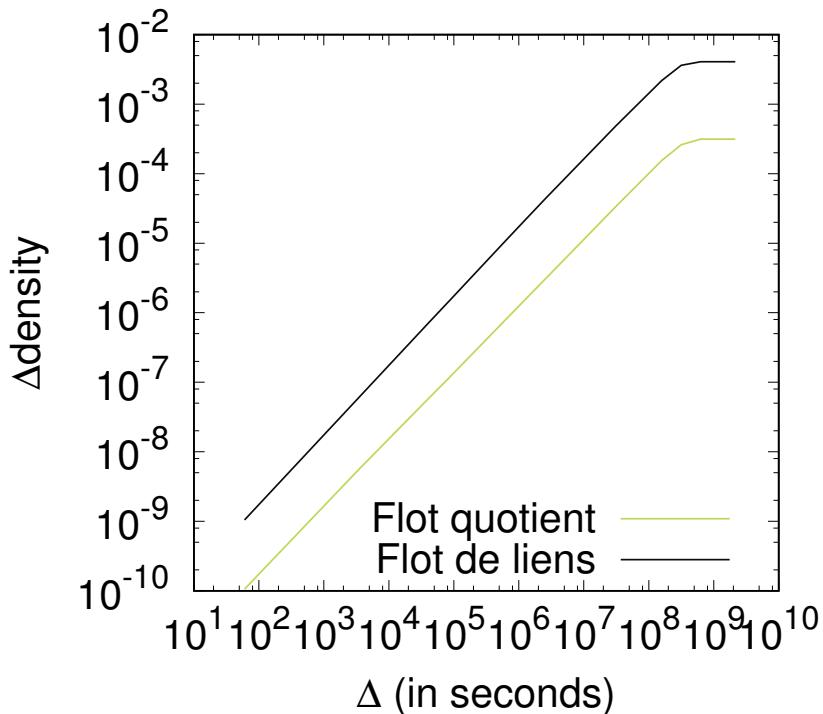


FIGURE 3.11 – Δ -densité du flot de liens et du flot de liens quotient en fonction de Δ pour $\Delta = 1mn, 1h, 12h, 1j, 7j, 30j, 1\text{ an}$ et 20 ans .

La figure 3.11 présente la Δ -densité du flot de liens initial et du flot quotient pour différentes valeurs de Δ . Le flot initial et le flot quotient ont le même comportant de densité mais le flot quotient est moins Δ -dense que le flot initial. Ce résultat diffère par rapport à ce qui est obtenu dans un graphe. Cela est dû au nombre de noeuds qui augmente dans le flot quotient. Mais le flot quotient contient tout de même beaucoup de liens. En effet, le degré moyen dans le flot quotient est moyenne 25 fois plus élevé que dans le flot.

3.3.4 Conclusion

Nous avons utilisé le modèle de flot de liens pour étudier une archive de courriels provenant du projet Debian. Grâce au modèle de flot de liens, nous avons étudié des notions clefs pour mieux comprendre la répartition temporelle et topologique des discussions. Nous avons étudié la notion de Δ -densité sur les discussions en elles mêmes. Puis, nous avons étudié les relations entre les discussions avec la Δ -densité inter discussions, les projections en graphe de chevauchement temporel ou topologique et le flot quotient.

Cette étude repose en grande partie sur la notion de Δ -densité qui nécessite un paramètre fixé arbitrairement. Nous avons à chaque fois testé un ensemble de valeurs de Δ variant d'une seconde jusque parfois 20 ans et, lors de ces tests, aucune valeurs Δ caractéristique n'a pu être identifiée. Il semble donc que la Δ -densité soit relativement robuste vis-à-vis de Δ dans ce contexte.

Nous avons tout d'abord observé que les discussions forment une structure plus dense que le flot de liens. De manière encore plus forte, nous avons constaté, grâce à la Δ -densité inter discussion, que les discussions sont plus denses en interne qu'en externe. C'est une caractéristique importante des communautés que l'on trouve dans les graphes mais qui n'avait pas été observée dans un contexte temporel. À partir de ces observations, nous avons également observé les relations entre les discussions. Via le graphe de chevauchement temporel, nous avons validé le fait que différentes discussions ont lieu en même temps et que par conséquent une

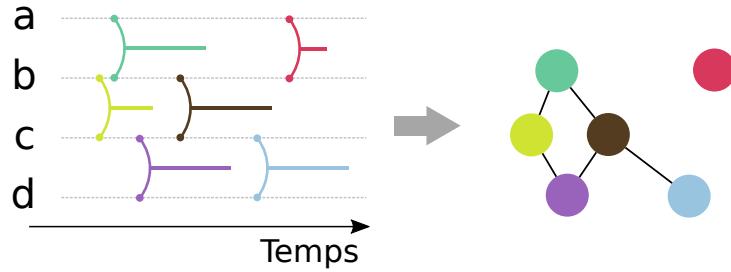


FIGURE 3.12 – Transformation d'un flot de liens avec 4 noeuds (a-d) et 6 liens à gauche en un graphe à droite à 6 noeuds. La couleur d'un noeuds dans le graphe indique le lien du flot qu'il représente.

agrégation temporelle entraînerait une perte d'information. De même via le graphe de chevauchement topologique, on remarque que la structure est très recouvrante sur les noeuds, rendant ainsi l'utilisation de partitions statiques de noeuds difficilement envisageable pour décrire les discussions.

3.4 Détection de structures denses

À partir du constat que les discussions forment une structure particulière, il est naturel d'essayer de les retrouver automatiquement. Pour y parvenir, il faut un moyen capable de trouver des sous-flots-denses dans le flots. C'est à dire une méthode capturant des groupes de liens qui soient proche temporellement et topologiquement. Il serait tentant d'optimiser directement la densité dans le flot mais ce n'est pas envisageable car un groupe constitué d'un unique lien a une densité de 1. Il faut donc trouver une autre méthode. C'est pourquoi nous avons construit une autre projection du flot en un graphe statique afin d'y appliquer une méthode de détection de communautés. Le problème est alors de réussir à créer une transformation de telle sorte que les informations temporelles et topologiques ne soient pas complètement détruites.

3.4.1 Méthode de détection

Afin de créer une transformation du flot vers un graphe, nous définissons un autre flot de liens, \mathcal{L} , dont les liens ont une durée. À partir de \mathcal{L} , nous créons un graphe non-orienté et non-pondéré $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Chaque lien du flot est représenté par un noeud. Deux liens (b, e, u, v) et $(b', e', u', v') \in \mathcal{L}$ sont connectés dans le graphe s'ils partagent un noeud et si les intervalles s'intersectent, i.e. $\{u, v\} \cap \{u', v'\} \neq \emptyset$ et $[b, e] \cap [b', e'] \neq \emptyset$, voir figure 3.12. Ainsi, un lien dans le graphe représente une connexion structurelle et temporelle entre deux liens du flot de liens. Les groupes denses dans le graphe représentent donc des groupes de liens connectés temporellement et topologiquement dans le flot.

Il faut donc trouver une manière d'ajouter une durée à chaque lien pour créer \mathcal{L} . Lors du calcul de la densité dans la section 3.3.1, nous avions ajouté une durée arbitraire Δ . Ici, il n'est pas très pertinent d'appliquer la même logique. En effet, si on utilise un Δ faible, alors il n'y aura que très peu de liens dans \mathcal{E} et les noeuds représentant les liens d'une discussions ne seront pas forcément connexes. Il paraît illusoire d'espérer retrouver les discussions dans \mathcal{G} si elles ne sont même pas connexes. Si Δ est très grand alors toute information temporelle est perdue et cela revient à calculer le line graphe du graphe agrégé. C'est pourquoi nous adoptons une autre manière d'ajouter une durée sur les liens.

Pour chaque message m , nous connaissons $p(m)$, le message auquel il répond dans la discussion. Nous définissons alors les liens de \mathcal{L} qui ont une durée de la manière suivante :

$(t(p(m)), t(m), a(m), a(p(m)))_m$. Ainsi, deux messages, m_1 et m_2 , se succédant dans une discussion sont par définition reliés topologiquement car $a(m_1) = a(p(m_2))$. Ces deux messages sont aussi reliés temporellement car nous avons la relation suivante :

$$\begin{aligned}[t(p(m_1)), t(m_1)] \cap [t(\mathbf{p}(\mathbf{m}_2)), t(m_2)] &= \\ [t(p(m_1)), t(m_1)] \cap [t(\mathbf{m}_1), t(m_2)] &= [t(m_1)] \neq \emptyset.\end{aligned}$$

Par construction, une discussion est donc représentée dans \mathcal{G} par un ensemble connexe de nœuds. Un fois \mathcal{G} construit, on peut appliquer un algorithme de détection de communautés.

Avec cette construction, \mathcal{G} contient plus d'1 millions de liens entre 316 569 nœuds pour les 116 999 discussions présentes. Sur ce graphe, nous avons appliqué l'algorithme de Louvain [BGLL08] qui optimise la modularité. D'autres algorithmes peuvent également être appliqués s'ils capturent des groupes de nœuds disjoints et qu'ils passent à l'échelle. Les groupes trouvés par Louvain sont des communautés dans \mathcal{G} . Par conséquent, ils sont censé être densément connectés dans \mathcal{G} . Comme un lien de \mathcal{G} correspond à une connexion temporelle et topologique dans le flot, on peut espérer qu'ils correspondent à des groupes denses dans le flot.

3.4.2 Comparaison des partitions

Avant de comparer la structure des discussions, D , et la partition, \mathfrak{D} , trouvée par la méthode de Louvain sur \mathcal{G} , il est nécessaire de décrire cette dernière. Dans la figure 3.13, les distributions cumulatives inverses du nombre de liens, du nombre nœuds et de leur durée sont présentées pour les groupes de \mathfrak{D} . Pour rappel, les même données sont représentées pour les discussions. On remarque tout de suite que \mathfrak{D} contient des groupes beaucoup plus gros en nombre de nœuds et de liens alors qu'ils ont des durées similaires.

Ces deux structures sont donc très différentes mais cela pourrait être dû à l'algorithme de Louvain qui n'est pas adapté pour trouver des groupes denses. C'est pourquoi, nous avons également observé la densité des groupes de D et \mathfrak{D} dans le flot \mathfrak{L} . Le résultat est visible dans la figure 3.13d. Comme les liens de \mathfrak{L} ont une durée, il est possible d'utiliser directement la densité au lieu de la Δ -densité utilisée précédemment. On remarque que les groupes de \mathfrak{D} , bien que plus gros, sont plus denses que les groupes de D . Cependant la distribution cumulative inverse cache les effets de la taille sur la densité. Or à nombre de liens égale (entre 2 et 160), on remarque que les groupes trouvés par Louvain sont plus denses en moyenne ,0.46 contre 0.38. La médiane est également plus élevée : 0.34 contre 0.33. En revanche, les plus gros groupes ($|\mathfrak{D}_j| > 160$) trouvés par Louvain ont une densité plus faible ce qui peut être dû à leur taille.

Si les groupes de \mathfrak{D} sont plus denses, c'est peut être car ils regroupent plusieurs discussions de D dans un groupe. Pour comparer deux partitions, l'indice de Jaccard est classiquement utilisé pour calculer la *précision* et le *rappel* qui sont définis de la manière suivante :

$$\text{précision}(\mathfrak{D}_j) = \max_i \frac{|\mathfrak{D}_j \cap D_i|}{|\mathfrak{D}_j|}, \quad \text{rappel}(D_i) = \max_j \frac{|\mathfrak{D}_j \cap D_i|}{|D_i|}.$$

Dans la figure 3.14, est présenté la *précision* des groupes et le *rappel* des discussions en fonctions de leur taille. Chaque point représente la moyenne du *précision* (resp. *rappel*) pour les groupes (resp. discussions) d'une taille donnée. On voit qu'il y a un important rappel et ce même pour les grandes discussions, ce qui veut dire qu'en générale une discussion D_i est totalement incluse dans un groupe \mathfrak{D}_j . En revanche, la précision est très faible car un groupe \mathfrak{D}_j contient plusieurs discussions, ce qui est cohérent avec la taille très importante des groupes de \mathfrak{D}_j .

Il semble donc que la partition \mathfrak{D} soit proche de D mais que ses groupes soient plus gros. Pour circonvenir à ce problème, nous appliquons de manière récursive l'algorithme de Louvain sur chaque graphe induit par un groupe \mathfrak{D}_j . Ce processus permet de subdiviser de manière

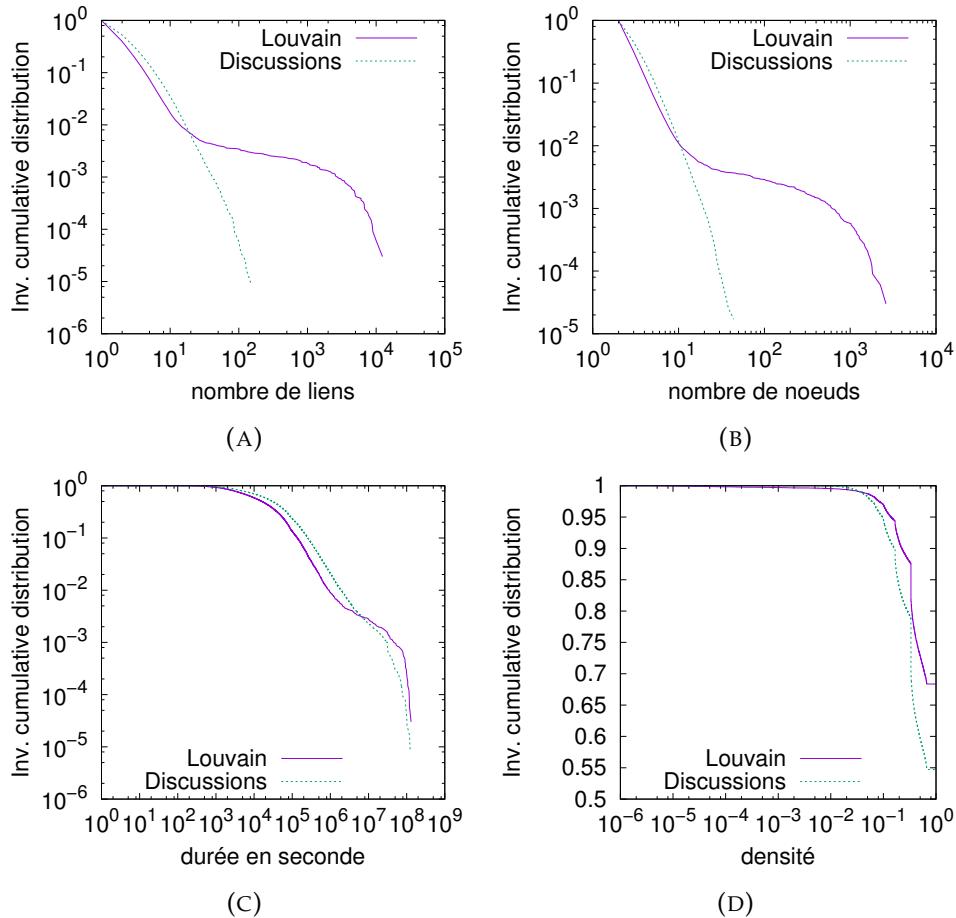


FIGURE 3.13 – Distribution cumulative inverses du nombre de liens (a), du nombre de nœuds (b), de la durée (c) et de la densité (d) pour les groupes trouvés par Louvain et les discussions.

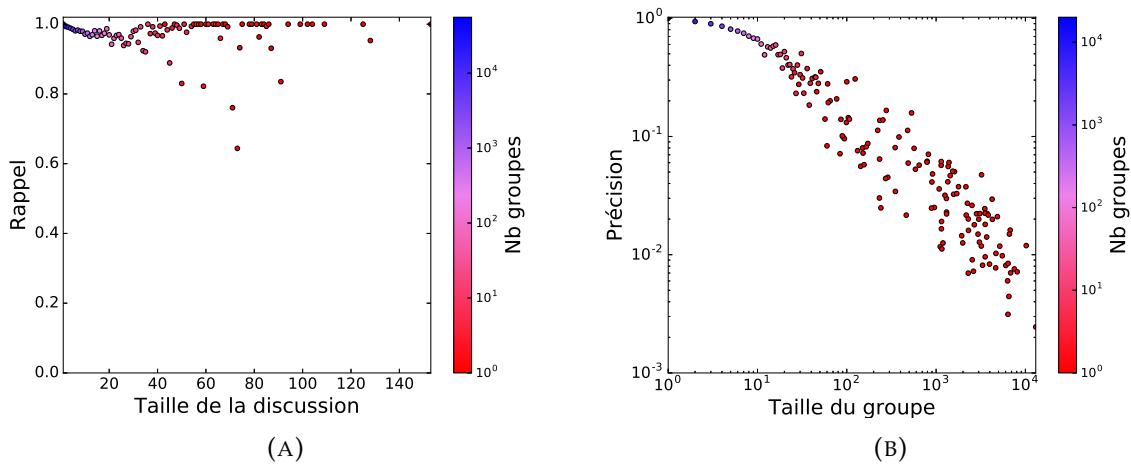


FIGURE 3.14 – (A) Rappel des discussions vis-à-vis des groupes trouvés par Louvain. (B) Précision des groupes trouvés vis-à-vis des discussions. Chaque point représente la moyenne du rappel (resp. précision) pour les discussions (resp. groupes) ayant la même taille. La couleur du point indique le nombre de groupes ayant la même taille.

récursive chaque groupe \mathfrak{D}_j . Soit $\mathfrak{D}_{j'}(h)$ un groupe trouvé au niveau h , par construction, il est inclus dans un groupe trouvé au niveau $h - 1$, c'est à dire $\mathfrak{D}_{j'}(h) \subseteq \mathfrak{D}_j(h - 1)$. Le niveau 0 est la première partition trouvée par l'algorithme de Louvain dans \mathfrak{G} .

Soit $D_i \in D$, notons $\mathfrak{D}_{\tilde{j}}(h)$ avec $h \in \mathbb{N}$ le groupe trouvé par la méthode de Louvain au niveau h qui soit le plus proche de D_i au niveau h , c'est-à-dire $|\mathfrak{D}_{\tilde{j}}(h) \cap D_i| = \max_j |\mathfrak{D}_j(h) \cap D_i|$. Avec ces définitions, on observe la relation suivante : $\mathfrak{D}_{\tilde{j}}(h) \cap D_i \subseteq \mathfrak{D}_{\tilde{j}}(h - 1) \cap D_i$. La définition de *rappel* de l'équation 3.4.2 n'est donc pas adaptée pour les niveaux inférieurs et nous l'adaptions de la manière suivante :

$$\text{rappel}(D_i, h) = \max_j \frac{|\mathfrak{D}_j(h) \cap D_i|}{|D_i \cap \mathfrak{D}_{\tilde{j}}(h - 1)|}. \quad (3.3)$$

Ainsi, le *rappel* au niveau h prends en compte le maximum d'élément qu'il est possible de trouver à ce niveau. La définition de *précision* ne pose quant à elle pas de problème. La figure 3.15 représente le *rappel* adapté et la *précision* pour le premier et deuxième niveau de récursion de la même manière que pour la figure 3.14. On remarque que, dès le premier niveau, le rappel baisse et que ce phénomène s'amplifie fortement au niveau suivant. Cela implique que les discussions ne sont plus incluses dans un groupe mais au contraire réparties dans plusieurs. La précision quant à elle augmente légèrement mais cela est dû à la baisse de la taille des groupes trouvés. Il semble donc qu'il ne soit pas possible avec cette approche de retrouver automatiquement les discussions.

3.4.3 Conclusion

Nous avons avec cette méthode mis en évidence des groupes denses. Les groupes trouvés sont plus gros et plus denses que la structure des discussions. Cependant, ces observations ne remettent pas en cause les conclusions faites dans la section 3.3 pour plusieurs raisons. Tout d'abord, les flots de lien étudiés ne sont pas exactement les-mêmes ($L \neq \mathcal{L}$). Ce changement de flot est nécessaire pour le fonctionnement de la méthode de détection. Ensuite, les deux structures ne sont pas complètement différentes car les groupes trouvés semblent en fait agréger plusieurs discussions. Malheureusement, nous n'avons pas réussi avec notre méthode à isoler chaque discussion malgré notre approche récursive. Pourtant, il semble que la structure trouvée ai du sens au vu des valeurs de densité des groupes.

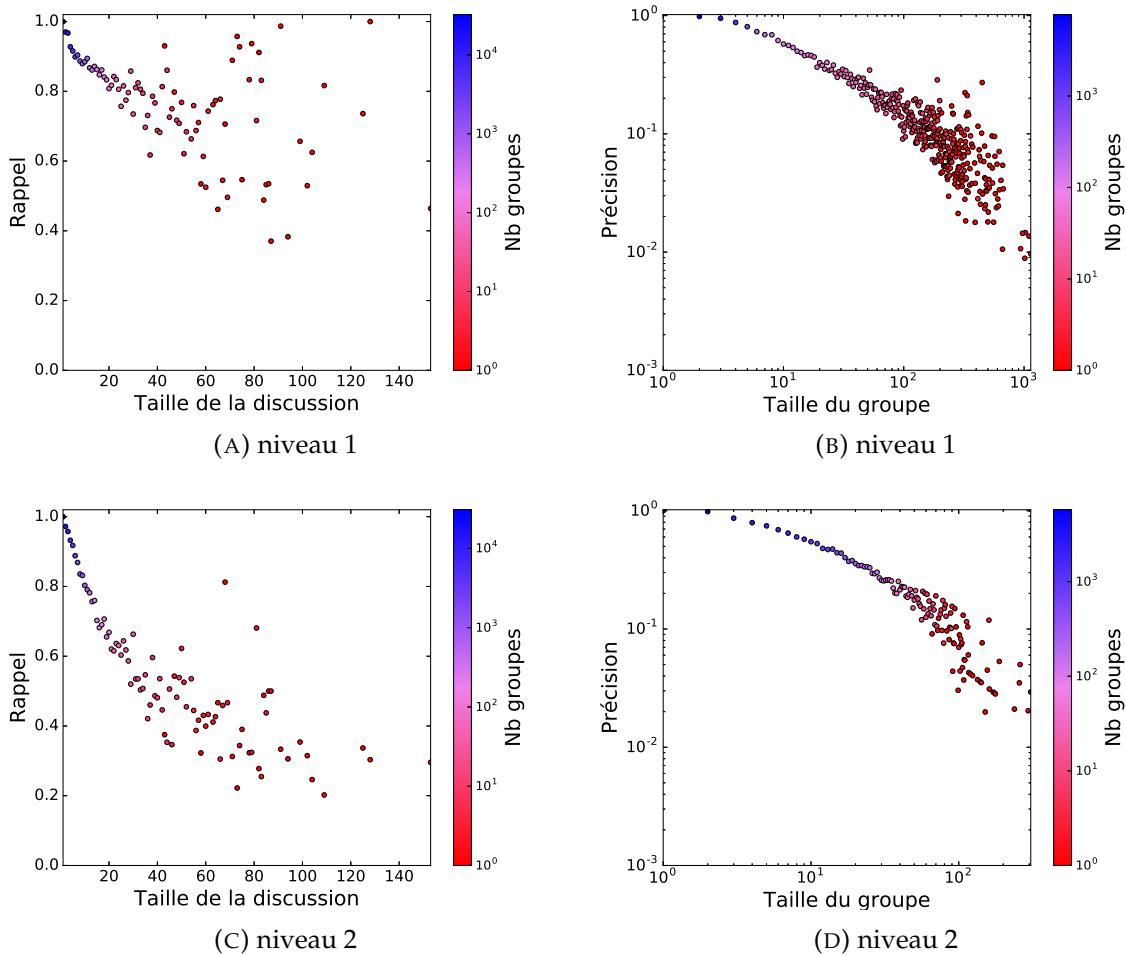


FIGURE 3.15 – (A,B,C) Rappel des discussions vis-à-vis des groupes trouvés par Louvain à différent niveaux récursif. (B,D,F) Précision des groupes trouvés vis-à-vis des discussions. Chaque point représente la moyenne du rappel (resp. précision) pour les discussions (resp. groupes) ayant la même taille. La couleur du point indique le nombre de groupes ayant la même taille.

Chapitre 4

Détection de groupes pertinents dans les flots de liens

Sommaire

4.1	Travaux existants	44
4.2	Détection de groupes pertinent	45
4.2.1	Définition du voisinage	45
4.2.2	Définition de l'évaluation	47
4.2.3	Algorithme de calcul des scores	48
4.3	Jeux de données	50
4.4	Application	51
4.4.1	Évaluation des groupes candidats	52
4.4.2	Caractéristiques des groupes pertinents	55
4.5	Conclusion et perspective	57
4.5.1	Conclusion	57
4.5.2	Perspective	58

Nous avons vu précédemment qu'il était possible de trouver des groupes de liens dans un flots de liens via l'intermédiaire d'une projection du flot de liens en un graphe statique. Parmi l'ensemble de ces groupes, il se peut que certains capturent des structures plus importantes et plus pertinentes que d'autres. Le but ici est de réussir à extraire parmi ces groupes ceux qui sont pertinents. En ce sens, nous nous posons la question de la décomposition partielle d'un flot de liens en sous-flots pertinents. Il s'agit donc d'un problème légèrement différent de la détection de partition car certains liens peuvent n'appartenir à aucun groupe et les groupes doivent être pertinents.

Toute la difficulté pour réussir à trouver les groupes pertinents revient à trouver une définition convaincante de la pertinence. La notion de densité est une première approche mais ce n'est pas suffisant. En effet comme on a pu le voir précédemment, il peut exister de très grands groupes peu dense et de petits groupes très denses. Faire la différence entre ces deux catégories en utilisant juste la densité n'est pas possible. C'est pourquoi, nous utilisons une notion de voisinage. Un groupe est alors pertinent si il est plus dense que son voisinage. Ainsi, un groupe peu dense peut être pertinent si il est plus dense que son voisinage. Comme le but est de pouvoir décrire le flot de liens, nous nous limitons aux groupes ayant une taille supérieur à un minimum. Un exemple de flots avec une décomposition en groupes pertinent est dans la figure 4.1.

Afin de trouver des groupes pertinents, nous procédons de la manière suivantes :

- Nous construisons la projection du flot de liens en un graphe statique ;
- Nous appliquons sur cette projection un algorithme de détection de communautés afin d'obtenir une partition des liens du flot de liens ;
- Nous ne gardons dans la partition que les groupes qui sont considérés comme pertinent, c'est-à-dire ceux qui sont assez gros et qui sont plus dense que leur voisinage.

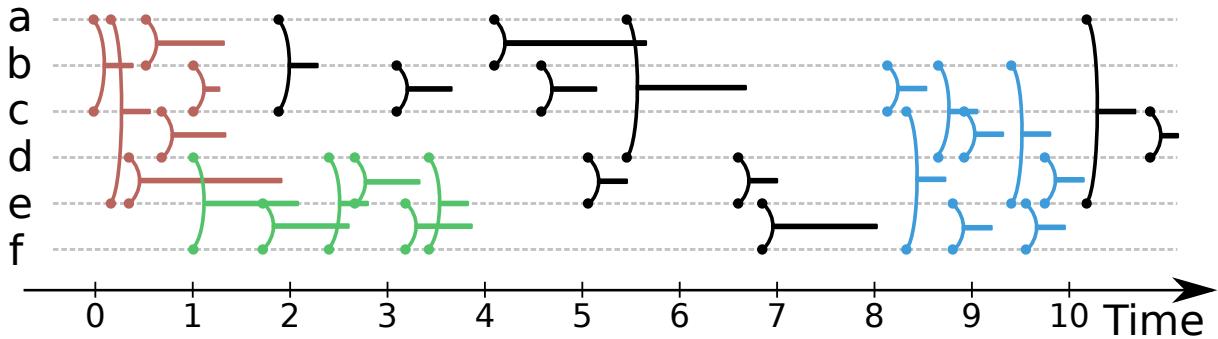


FIGURE 4.1 – Exemple de flots de liens avec 3 groupes denses représenté par la couleur des liens (rouge, vert et bleu).

Afin de montrer la pertinence de cette approche, nous l’appliquons sur différents jeux de données et essayons de faire une analyse manuelle de certains groupes trouvés. Par ailleurs, nous montrons que les groupes que nous détectons ne le sont pas par une méthode statique.

Le chapitre est organisé de la manière suivante. Dans la section 4.1, nous revenons sur les travaux existants qui traitent de sujets similaires. Puis dans la section 4.2, nous présentons notre méthode de détection et de validation de groupes pertinents. Enfin, les jeux de données que nous utilisons sont présentés dans la section 4.4 et les résultats associés dans la section 4.3.

4.1 Travaux existants

Comme nous l’avons vu dans le chapitre 1, il existe assez peu de méthodes cherchant une structure dans un formalisme sans perte d’information.

Mitra *et al.* [MTR12] et Speidel *et al.* [STM15] ont développé une méthode de détection de communauté mais dans les réseaux diachronique. Dans un tel réseau, ce sont les nœuds qui ont un *timestamp* et non les liens. Ce cas de figure s’applique parfaitement aux citations car une citation relie bien deux publications qui sont chacune apparues à une date spécifique. Cependant ce genre de construction est différente n’est pas équivalente au formalisme de flot de liens. C’est pourquoi nous ne pouvons utiliser la même méthode.

Ils existent également des méthodes capturant la partie la plus dense du réseau. C’est le cas de Bogdanov *et al.* [BMS11] qui considèrent encore une autre variante d’extension de graphe. Ils considèrent un graphe dont la topologie n’évolue pas mais dont le poids des liens change en fonction du temps entre -1 et 1. Ce formalisme est encore une fois très différent de celui de flot de liens. De plus, la notion de densité qu’ils utilisent ne tient pas vraiment compte du temps. En effet, un groupe de nœuds sur intervalle est évalué par la somme des liens entre ces nœuds sur tout l’intervalle.

Epasto *et. al* [ELS15] capture le sous-graphe le plus dense dans un graphe temporel. Ainsi, le temps est bien pris en compte dans le formalisme. Cependant, la méthode capture le sous-graphe à un instant t qui est le plus dense. La notion de densité est donc statique car elle ne considère que la structure de graphe à l’instant t .

Enfin il existe les travaux de Rozenshtein *et. al* [RTG14] qui se rapprochent le plus de notre méthode. Leur méthode considère le formalise de flots de liens et ne suppose pas de structure de graphe à un instant donné. Elle capture un groupe de nœuds et plusieurs intervalles disjoints de telle sorte que ce groupe de nœuds dans le graphe agrégé sur ces intervalles ait le degré moyen le plus élevé. Ainsi même si la méthode permet de capturer un groupe de nœuds sur des intervalles, elle évalue un candidat sur le graphe agrégé pour appliquer une métrique de graphe. Le temps n’a donc pas d’influence sur l’évaluation. Par ailleurs, leur méthode repose sur plusieurs paramètres qui doivent être fixés *a priori* : le nombre maximum d’intervalles

disjoint et la durée cumulée maximum de ces intervalles. Le premier paramètre permet d'éviter d'utiliser une infinité d'intervalle qui seraient chacun centré sur un lien. Le second paramètre permet d'éviter de considérer tout le graphe agrégé et ainsi d'obliger la prise en compte du temps. Ainsi, leur méthode dépend de manière non triviale sur le choix des paramètres.

Pour résumer, il existe des méthodes cherchant la structure la plus dense dans un contexte dynamique ; soit en considérant un graphe temporel soit en considérant un flot de liens. Cependant, ces méthodes utilisent des métriques de graphes pour évaluer un groupe de noeuds. Le temps n'est donc complètement pris en compte dans l'évaluation. De manière plus importante, ces méthodes évaluent de manière absolue sans prendre en compte de notion de voisinage. Enfin, nous capturons plusieurs groupes de liens alors qu'ils capturent un groupe noeud. Certaines des méthodes existantes peuvent être étendues pour capturer les k groupes de noeuds les plus denses mais cela se fait au prix de l'ajout d'un paramètre pour contrôler le chevauchement entre deux groupes capturés.

4.2 Détection de groupes pertinent

Nous avons défini dans le chapitre précédent 3.4.1 une transformation du flot de liens en un graphe statique. Sur cette transformation, nous appliquons un algorithme de détection de communautés. Nous considérons les communautés trouvées par l'algorithme comme des groupes potentiellement pertinents. Nous les appelons donc groupes candidats.

L'ensemble des groupes candidats ne sont pas forcément pertinents. Tout d'abord, les groupes d'une partition peuvent avoir des tailles très variées. En particulier, il peut y avoir beaucoup de très petits groupes de 1 ou 2 liens. Ces groupes ne sont pas prise en compte car nous fixons une limite arbitraire sur la taille minimum des groupes.

Par ailleurs, l'algorithme de Louvain optimise de manière gloutonne la modularité dans la transformation du flot de liens en un graphe statique et non une métrique de flot de liens directement. C'est pourquoi, certains candidats peuvent être une bonne communauté dans le graphe mais un groupe non pertinent dans le flot de liens. Cet effet est d'autant plus présent que la modularité ne prend pas en compte le voisinage d'un groupe. Par exemple dans la figure 4.1, les liens dans l'intervalle [4, 8] forment une composante connexe dans la transformation et ils sont considérés comme une communauté par l'algorithme de Louvain. Or, ce candidat n'est pas un groupe pertinent car il est beaucoup moins dense que les liens durant l'intervalle [0, 4] ou l'intervalle [7, 11]. C'est pourquoi il est nécessaire de valider ou de filtrer les groupes candidats pour ne garder que les groupes pertinents.

4.2.1 Définition du voisinage

Dans le chapitre 2.3, nous avons défini la densité d'un flot de liens. Pour la densité d'un groupe de liens, il faut donc calculer la densité du sous-flots induit par ces liens. Il est donc possible de calculer la densité d'un candidat. Seulement, la densité en elle même n'est pas un bon critère pour évaluer la pertinence d'un candidat. En effet, un candidat constitué d'un unique lien a une densité de maximal de 1. C'est pourquoi il est également nécessaire de tenir compte du voisinage. Ainsi, plus le candidat a une densité plus importante que son voisinage, plus le candidat est un groupe pertinent.

Il est donc nécessaire de définir une notion de voisinage qui puisse tenir compte à la fois de la structure et du temps. Pour ce faire, il faut utiliser une autre définition de la densité pour comprendre quels sont les paramètres qui influent sur la densité. Jusqu'à maintenant pour un candidat C_i , nous considérons la densité du sous-flot induit par ses liens, $\delta(L(C_i))$. Pour faire apparaître l'influence de la structure et du temps, nous utilisons $\delta(L_{\beta(C_i) \dots \psi(C_i)}(V(C_i)^2))$ que nous simplifions par $\delta(V(C_i), \beta(C_i), \bar{C}_i)$ où $\bar{C}_i = \psi(C_i) - \beta(C_i)$. Il s'agit de la densité des

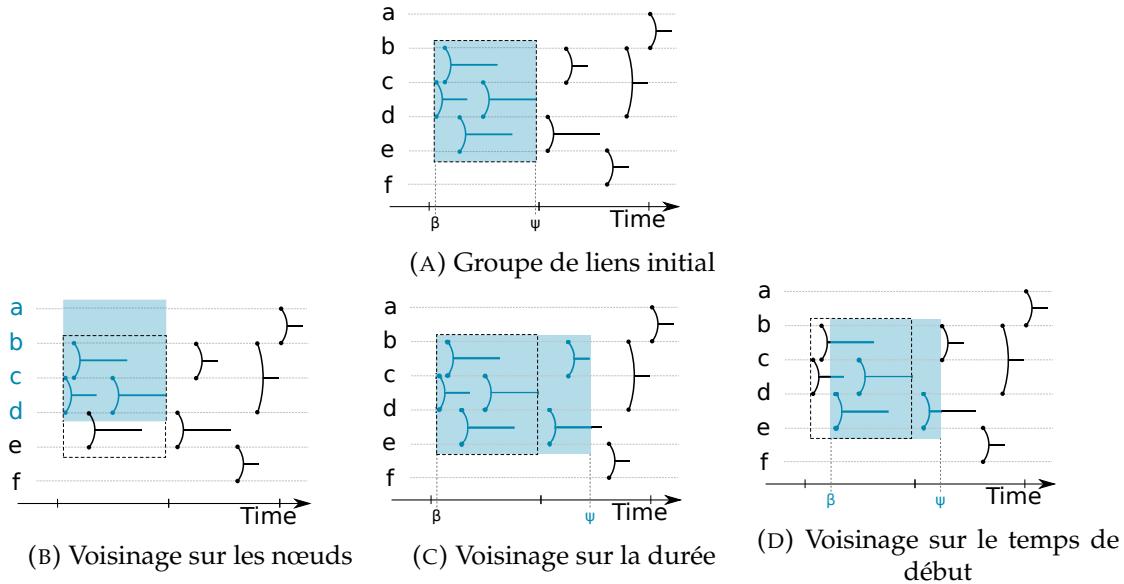


FIGURE 4.2 – Exemple d'un groupe de liens en (A) et de ses différents voisins en (B), (C) et (D). La zone en bleu est la zone prise en compte lors du calcul de la densité. La zone en pointillé représente la zone considérée lors du calcul de la densité du groupe initial.

nœuds induits par C_i sur la durée du groupe. Ces deux formulations ne sont pas équivalentes car les deux sous-flots ne sont pas équivalents en particulier $L(C_i) \subseteq L_{\beta(C_i)..\psi(C_i)}(V(C_i)^2)$. De cette inclusion découle le fait $\delta(L(C_i)) \leq \delta(V(C_i), \beta(C_i), \bar{C}_i)$. C'est le cas dans l'exemple de la figure 4.1 si l'on veut calculer la densité des liens noirs. Dans ce cas, l'ensemble de nœuds induit par ces liens est égal à V et l'intervalle de temps est égal à $[2, 11]$. Le sous-fLOT induit par V sur $[2, 11]$ comprend l'ensemble des liens noirs mais aussi des liens **verts** et **bleus**.

Cependant avec cette formulation, on fait apparaître 3 dimensions : l'ensemble de nœuds, le temps de début et la durée. Il devient alors aisément de considérer comme voisins les sous-flots qui diffèrent sur une seule dimension : soit le temps de début, soit la durée soit les nœuds. Cette différence entre les voisins est illustrée dans la figure 4.2.

Pour le voisinage au nœuds, il est impossible de considérer l'ensemble de tous les ensemble de nœuds car il est beaucoup trop grand. De plus, il est probable que ces groupes de nœuds ne soient pas connectés si le flot de liens est peu dense. C'est pourquoi, si $V(C_i)$ contient k nœuds, nous nous limitons à l'ensemble des groupes de k nœuds qui partagent $k - 1$ nœuds avec $V(C_i)$. De cette manière, uniquement des groupes de nœuds similaires au candidat sont considérés. Cette comparaison nous semble plus stricte mais plus équitable que celles utilisant l'ensemble des groupes de nœuds ou des groupes de nœuds pris aléatoirement. Il serait possible de considérer un nombre moins important de nœuds partagés mais nous ne l'avons pas fait à cause de la complexité de calcul. En effet, partager $k - 1$ nœuds entraîne la création $k(|V| - k)$ sous-flots pour le voisinage au nœuds pour un unique candidat.

Pour le voisinage au temps de début (resp. à la durée), nous considérons toutes les valeurs possibles de temps début (resp. durée) dans un intervalle donné. Chacune de ces valeurs donne lieu à un sous-fLOT induit ayant les mêmes nœuds et la même durée (resp. temps de début) que le sous-fLOT du groupe candidat. Ensuite, nous calculons la densité de chacun de ces sous-flots voisins. Plus formellement, l'ensemble des densités des sous-flots voisins se définit de la manière suivante. Soit I_1 et I_2 deux intervalles et C_i un groupe de liens. Les densités des sous-fLOT dans le voisinage au temps de début sont alors définies par la fonction $\delta(V(C_i), x, \bar{C}_i)$, $\forall x \in I_1$. Les densités des sous-fLOT dans le voisinage de la durée sont alors définies par la fonction $\delta(V(C_i), \beta(C_i), y)$, $\forall y \in I_2$.

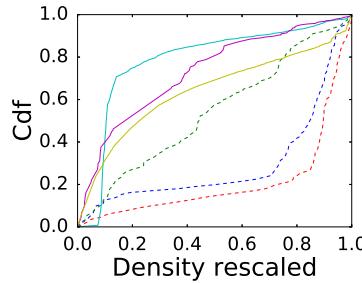


FIGURE 4.3 – Distribution cumulative (*cdf*) des valeurs de densité renormalisées des sous-flots voisins dans le voisinage à la durée. En train plein (resp. pointillé), candidats du jeux de données Rollernet (resp. Socio pattern).

L'intervalle utilisé pour le voisinage au temps de début $[\alpha, \omega - \bar{C}_i]$. Pour le voisinage à la durée, nous utilisons l'intervalle $[0.8\Delta_{min}, 1.2\Delta_{max}]$, où Δ_{min} (resp. Δ_{max}) est la plus petite (resp. grande) durée de tous les candidats considérés initialement. Nous utilisons cet intervalle pour deux raisons. Tout d'abord, il contient l'ensemble des durées acceptables quand nous l'appliquons à nos jeux de données. Mais surtout, nous avons également testé l'intervalle $[1 - \omega - \alpha]$ qui est beaucoup plus grand. Cela change les résultats de manière quantitative mais pas de manière qualitative. En effet lorsque la durée considérée est proche de $\omega - \alpha$ alors la densité est très proche de 0. Comme ces valeurs sont peu informative, nous préférons nous limiter à un intervalle un peu plus restreint.

Au final, nous avons pour chaque voisinage les valeurs de densités des sous-flots voisins. Pour le voisinage au nœuds, il y a $k(|V| - k)$ valeurs de densité différentes. Pour le voisinage au temps de début (resp. à la durée), nous avons une fonction de la densité qui dépend du temps de début (resp. de la durée).

4.2.2 Définition de l'évaluation

Nous évaluons chaque candidat en comparant sa densité à celle des sous-flots dans chaque voisinage. Si un candidat est vraiment pertinent alors il devrait avoir une densité significativement plus élevée que la plupart des sous-flots voisins. Si la densité des sous-flots voisins suivait une distribution connue, par exemple une loi normale, alors cela reviendrait à évaluer si la densité du candidat est éloignée de la moyenne en observant le *z-score*. De nos observations, les distributions de densités des sous-flots voisins ne suivent pas la même distribution pour différents candidats. Par exemple dans la figure 4.3, des distributions cumulatives de la densité pour le voisinage à la durée pour 3 groupes candidats des jeux de données Socio pattern et Rollernet¹ sont présentées. Ces distributions sont très différentes et il n'est pas donc pas justifié d'assumer une unique distribution sous-jacente. C'est pourquoi nous utilisons les *percentiles* qui représentent la proportion des valeurs de densités qui sont inférieures à la densité du candidat.

Pour le voisinage au nœuds qui est représenté par un ensemble, S , de valeurs de densité, le score d'un candidat C_i ayant une densité, δ^* , est :

$$p_{noeud}(C_i) = \frac{\sum_{\delta \in S} \mathbf{1}_{\delta \leq \delta^*}}{|S|}, \quad (4.1)$$

où $\mathbf{1}$ est la fonction indicatrice. Pour le voisinage au temps de début et à la durée qui sont représentés par des fonctions, les scores sont respectivement :

1. Ces jeux de données sont présentés dans la section suivante 4.3.

$$p_{\text{début}}(C_i) = \frac{1}{\omega - \bar{C}_i - \alpha} \int_{\alpha}^{\omega - \bar{C}_i} \mathbf{1}_{\delta(V(C_i), z, \bar{C}_i) < \delta^*} dz, \quad (4.2)$$

$$p_{\text{durée}}(C_i) = \frac{1}{1.2\Delta_{\max} - 0.8\Delta_{\min}} \int_{0.8\Delta_{\min}}^{1.2\Delta_{\max}} \mathbf{1}_{\delta(V(C_i), \beta(C_i), z) < \delta^*} dz. \quad (4.3)$$

Un score de 15% pour un type de voisinage signifie que la densité du candidat est plus faible que 85% des sous-flots voisins de ce type et que par conséquent le candidat ne devrait pas être considéré comme pertinent.

Pour résumer, un candidat est évalué par un triplet constitué des scores obtenus pour chaque voisinage. Un candidat est considéré comme pertinent si tous ces scores sont plus élevés que des seuils définis manuellement. La définition de ces seuils est difficile et dépend des caractéristiques du flot de liens. C'est pourquoi nous ne les fixons pas *a priori* mais *a posteriori* après avoir examiner la distribution des scores. Le choix des seuil est décris plus amplement dans la section 4.4.

4.2.3 Algorithme de calcul des scores

Nous avons défini formellement les trois voisinages mais il faut encore définir un algorithme efficace pour les calculer. Si le problème n'est pas crucial pour le voisinage aux noeuds, il est plus délicat lorsqu'on considère le voisinage au temps de début ou à la durée car le temps est continu.

Voisinage au noeuds

Pour ce voisinage, il est nécessaire d'analyser des sous-flots durant le même intervalle, $[t, t']$, mais pour différents ensembles noeuds, V' . Il existe plusieurs stratégies possibles pour calculer la densité de ces sous-flots mais dans tous les cas il n'est pas nécessaire de créer explicitement le sous-flot. Il suffit de pouvoir itérer sur les liens le composants pour calculer leur densité, voir la section 2.3.

La première méthode consiste en manipuler chaque sous-flot de manière indépendante et de calculer leur densité. Pour extraire le sous-flot induit par un ensemble de noeuds sur un intervalle, il est judicieux de d'abord considérer le sous-flot sur les noeuds puis d'intégrer le temps car la construction du sous-flot induit par un ensemble de noeuds est plus rapide. Le calcul de la densité d'un sous-flot induit par un ensemble de noeuds, V' sur intervalle de temps $[t, t']$ est alors dépendant du nombre de liens existant entre les noeuds de V' avant t' et de la taille de V' : $C_{V'} = O(|L_{\alpha..t'}(V'^2)| \log(|V'|))$. $|L_{\alpha..t'}(V'^2)|$ correspond au coût nécessaire pour parcourir l'ensemble des liens ayant un de leur noeuds dans V' qui sont potentiellement dans $[t, t']$. Pour chacun de ces liens, il faut également tester que l'autre noeud du lien appartienne également à V' ce qui se fait en $\log(|V'|)$. Pour calculer l'ensemble du voisinage pour un groupe de noeuds de taille k , il est alors nécessaire de répéter ce processus pour chaque ensemble de noeuds ce qui donne une complexité de $O(k(|V| - k)\tilde{C}_{V'})$. Cette méthode est facilement parallélisable. C'est cette méthode que nous avons implémentée dans sa version non parallèle.

La deuxième méthode consisterait en tirer parti de la ressemblance des ensembles de noeuds dans le voisinage. La première étape est alors de calculer $d_{t..t'}(V'^2)$, le degré moyen du groupe de noeuds initial. Puis lors de l'évaluation d'un autre groupe de noeuds V'' , nous savons que V' et V'' ne diffère que sur un noeud, v' qui est remplacé par v'' . Pour calculer la densité de V'' sur $[t, t']$, il suffit de mettre à jour $d_{t..t'}(V'^2)$ en fonction des liens qui ne sont plus pris en compte et des nouveaux liens pris en compte. C'est à dire qu'il faut considérer $|L_{\alpha..t'}(v' \cap V')|$ suppressions de liens et $|L_{\alpha..t'}(v'' \cap V'')|$ ajouts de liens. Il faut de même répéter ce processus pour chaque

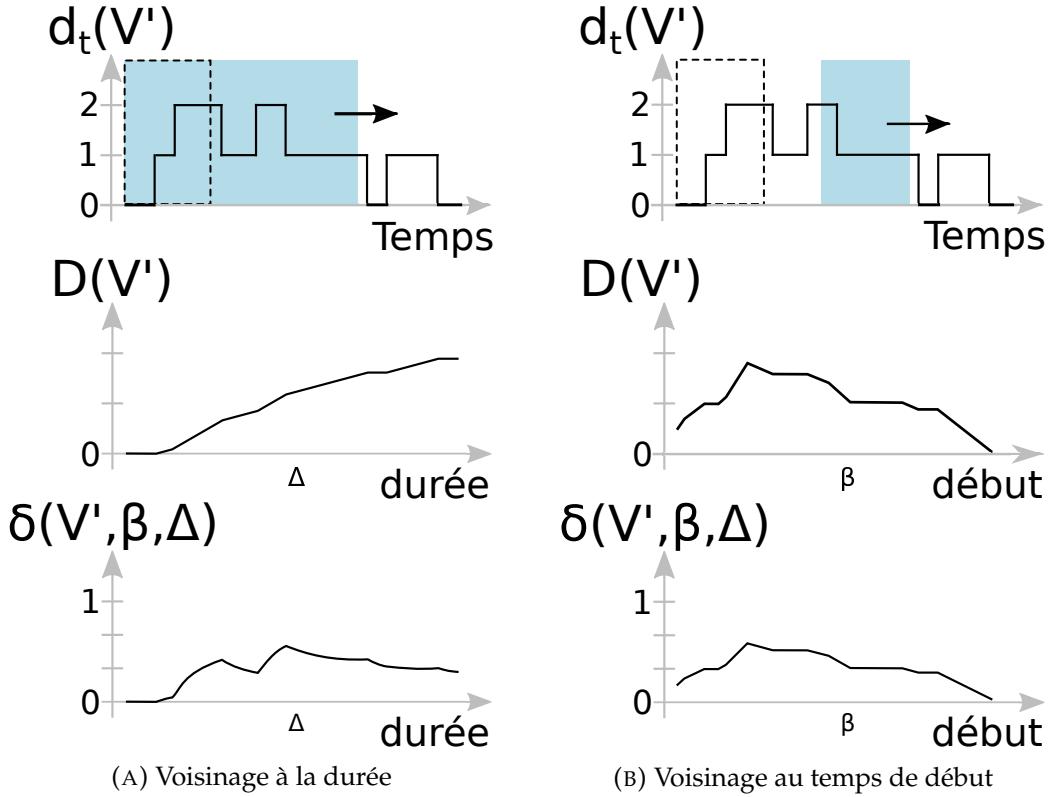


FIGURE 4.4 – Calcul du voisinage à la duré en (A) et au temps de début en (B) pour un groupe ayant un degré temporel donné. L'exemple en question est le même que celui dans la figure 2.2.

groupe de noeuds pour évaluer l'ensemble du voisinage. Cette méthode peut également être parallélisée.

Voisinage à la durée

Pour le voisinage à la duré, l'ensemble de noeuds est fixe et il faut de manière continue augmenter l'intervalle du sous-flot. Pour réussir à faire cela, il est nécessaire de se baser encore une fois sur le degré temporel de l'ensemble de noeuds en question. La densité dépend d'une part de la somme des degrés et d'autre part de la la durée de l'intervalle. Comme les degrés temporels sont des fonctions constantes par morceaux, la somme des degrés, qui est l'intégrale des degrés, est une fonction linéaire par morceaux dont les points de changement sont les instants où un lien apparaît ou disparaît. C'est pourquoi, il est possible de définir la densité en fonction de la durée par une fonction par morceaux, voir l'exemple dans la figure 4.4a. L'idée est de mettre à jour au fur et à mesure la somme des degrés en fonction des liens tout en temps compte de la durée du sous-flots qui augmente linéairement. La complexité est donc $O(|L_{t..t+\Delta_{max}}(V'^2)|)$ si le degré de temporel du groupe est connu.

Voisinage au temps de début

Pour le voisinage au temps début, l'approche est similaire. Il faut mettre à jour la somme des degrés. La principale différence est qu'il faut cette fois tenir compte de l'apparition et de la disparition des liens. Le parcours est donc un peu plus complexe mais la durée est fixe. C'est pourquoi la densité en fonction du temps de début est une fonction linéaire par morceaux. Il existe d'ailleurs une unique bijection entre la somme des degrés et la densité qui

Datasets	$ V $	$ E $	$\omega - \alpha$
Socio pattern	180	19774	9 jours
Rollernet	62	15803	3 heures
Reality Mining	94	44975	9 mois
Babouins	28	95616	14 jours

TABLE 4.1 – nombre de nœuds $|V|$, nombre de liens $|E|$ et durée de chaque jeu de données.

est $\delta(V', \beta, \Delta) = \frac{D(V')}{|V'|(|V| - 1)}$. Voir l'exemple dans la figure 4.4a. La complexité est donc $O(|L_{\alpha..w-\Delta}(V'^2)|)$ où Δ est la duré du groupe.

4.3 Jeux de données

Nous appliquons notre méthodes sur 4 jeux de données différents. La table 4.1 listent le nombre de nœuds, le nombre de liens et la durée de chaque jeu de données.

Socio Pattern [FB14]² contient le réseaux dynamique d'étudiant d'une classe préparatoire à Marseilles en 2012. 180 étudiants de 5 classes ont porté pendant 9 jours des capteurs de proximités. Ainsi, il est possible de savoir quand 2 interagissent. Dans ce jeu de donnée, la classe de chaque étudiant est connue.

Rollernet [TLB⁺09] a été collecté durant une randonné roller ayant eu lieu en 2066. Il s'agit d'un ayant regroupé 2500 participants. Parmi eux, 62 étaient équipé d'appareils *bluetooth* qui enregistrent l'ensemble des appareils *bluetooth* disponibles. Nous nous limitons uniquement aux contacts ayant eu lieu entre les 62 personnes portant un capteur. Des métadonnées sont également associés et notamment le rôle de chaque personne. Un rôle peut être membre de l'organisation à l'avant du peloton ou membre d'une association de roller.

Reality Mining [EPL09] représente le réseau d'interactions entre 94 personnes du *MIT Media Laboratory* entre Septembre 2004 et Juin 2005. Parmi les 94 participants, 68 évoluent dans le même bâtiment (90% étudiants, 10% employés) alors que le reste sont des étudiants de l'école de commerce de l'université. Ce jeu de donné a été récolté grâce à des téléphones *bluetooth* prêtés aux étudiants.

Babouins [CKW15, SPFCC15] contient la position *GPS* de 28 babouins (*Papio anubis*) dans le centre de recherche Mpala au Kenya. Ces 28 babouins représentent 80% d'une troupe. Chaque babouin est équipé d'un collier muni d'un capteur *GPS* qui enregistre la position toute les secondes. Nous transformons ces données en un flot de liens en créant un lien de entre deux babouins si ils sont à moins de 1 mètre 50 durant les dernières 10 secondes pour lisser les potentielles imprécisions du *GPS*. Le code est disponible en ligne³ et est écrit en *rust*.

Même si ces jeux de données sont des réseaux d'interactions face-à-face, ils sont différents dans leur dynamique. Par exemple, le jeu de donnée Rollernet a environ le même nombre de liens que celui de Socio Pattern alors qu'il dure seulement 3 heures contre 9 jours pour Socio Pattern. Enfin, nous n'incluons pas le jeu de données de courriels utilisé précédemment car

3. <https://bitbucket.org/nGaumont/baboonstreamextractor>

Datasets	$ \mathcal{C} $	$\langle V \rangle$	$\langle L \rangle$
Socio Pattern	12532 (155)	2 (9)	1 (15)
Rollernet	559 (75)	2 (31)	1 (194)
Reality Mining	5737 (474)	2 (12)	1 (36)
Babouin	37671 (1249)	2 (7)	1 (16)

TABLE 4.2 – $|\mathcal{C}|$: nombre de groupes dans la partition, $\langle |V| \rangle$ médiane du nombre de nœuds dans un groupe, $\langle |L| \rangle$ médiane du nombre de liens dans un groupe. Les valeurs en parenthèses correspondent à la valeur quand uniquement les groupes d’au moins 10 liens sont pris en compte.

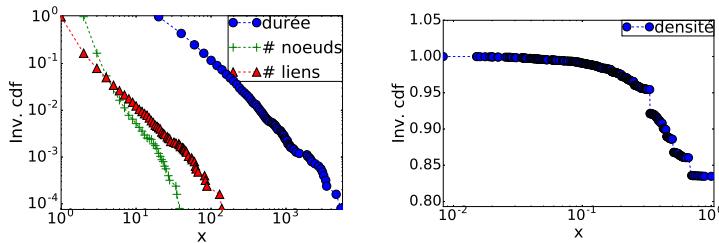


FIGURE 4.5 – Distribution cumulative inverse du nombre de liens, de nœuds et de la durée en (A) et de la densité en (B) pour les candidats trouvés par la méthode de Louvain dans le jeu de données Socio Pattern.

notre méthode nécessite un flot de liens avec durées. Nous pourrions utiliser un flot de liens avec une durée artificiellement mais le choix de la durée impacte les scores.

4.4 Application

Pour chaque jeu de données, nous appliquons notre méthode pour capturer les groupes d’interactions pertinents. Comme présenté dans la section 4.2, la première étape est de trouver une partition, \mathcal{C} , des liens. Quelques statistiques obtenues pour chaque partition sont listées dans la table 4.2, notamment les nombres de nœuds et de liens médians. La première chose qui est frappante est qu’il existe énormément de très petits groupes. On remarque également que seul le jeu de données Rollernet se distingue des autres en ayant des groupes plus gros. Cette différence est probablement due au flot de liens qui contient beaucoup de liens pour uniquement 3 heures de durée.

Afin d’avoir une vision plus précise pour le jeu de données Socio Pattern, nous présentons dans la figure 4.5 les distributions cumulatives inverses du nombre de nœuds, du nombre de liens, de la durée et de la densité des groupes. Les distributions sont toutes hétérogènes sauf pour la densité. Les irrégularités dans la distribution de la densité sont dues aux petits candidats ; typiquement un groupe de 2 liens entre 3 nœuds ont une densité de 0.33 et un groupe avec un seul lien⁴ a une densité de 1. Les distributions pour les autres jeux de données sont en appendice pages 83 à 86 dans les figures A.1,A.5et A.9 Pour ces jeux de données, les distributions du nombre de liens, du nombre de nœuds et de la durée sont également hétérogènes mais les distributions de densité sont légèrement moins irrégulières.

Comme les très petits groupes de liens ont un intérêt limité pour décrire un flot de liens, nous commençons par écarter les groupes ayant moins de 10 liens. Même si cela filtre énormément de groupes, il reste tout de même beaucoup de groupes dont il faut encore évaluer la pertinence.

4. Ils représentent 83% de tous les candidats.

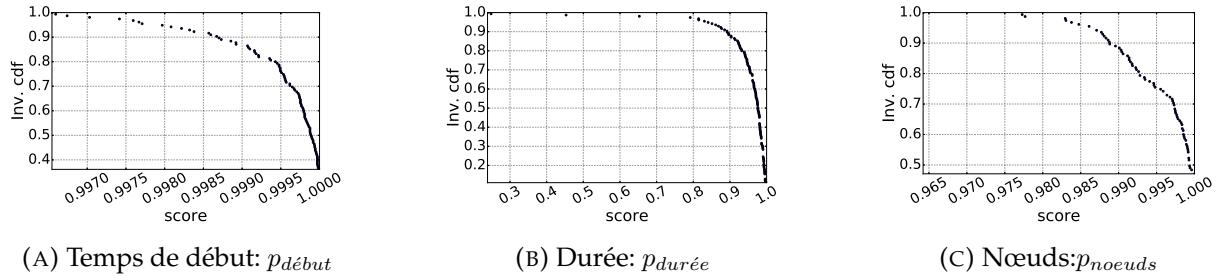


FIGURE 4.6 – Distribution cumulative inverse des scores pour chaque voisinage dans le jeu de données Socio Pattern.

	p_{noeuds}	$p_{début}$	$p_{durée}$	N_c	Représentativité	temps d'exécution
Socio Pattern	0.98	0.998	0.85	136	17.7%	2 min
Rollernet	0.9	0.7	0.6	37	95.4%	4 min
Reality Mining	0.97	0.98	0.8	80.9%	394	1h
Babouin	0.95	0.99	0.85	1023	39%	52 min

TABLE 4.3 – p_{noeuds} , $p_{début}$ et $p_{durée}$: seuils utilisés pour chaque voisinage, N_c nombre de groupes capturés, proportion de liens appartenant à un groupe pertinent, et temps d'exécution pour chaque jeu de données

4.4.1 Évaluation des groupes candidats

Pour séparer les groupes qui sont pertinents des autres, nous utilisons des seuils. Un candidat est ensuite considéré comme pertinent si ses scores sont au dessus des seuils choisis. Baisser les valeurs des seuils entraîne la capture de plus de groupes comme pertinents mais cela ne change pas les évaluations des groupes candidats. Ainsi, si un groupe est capturé pour un seuil donné alors il le sera également pour tout autre seuil inférieur. C'est pourquoi les seuils sont fixés après avoir étudié les distributions cumulatives inverses de score pour chaque voisinages. Elles sont représentées pour le jeu de données Socio Pattern dans la figure 4.6. Pour ce jeu de données, les scores sont très élevés, c'est-à-dire proche de 1, peu importe le voisinage considéré. On remarque également que les distributions cumulatives inverses chutent toutes à partir de certaines valeurs de scores et nous les utilisons comme seuils.

Nous utilisons pour le voisinage au temps de début, à la durée et aux noeuds les valeurs de seuils suivantes : 0.998, 0.85 et 0.98. Pour les autres jeux de données, les distributions de scores sont en appendice page 83 à 86 dans les figures A.2,A.6et A.10. Elles sont similaires à celles obtenues pour Socio Pattern mis à part pour le jeu de données Rollernet dont les scores sont plus faible. Cette différence est sûrement par la structure plus dense du flot de liens globale dans le cas de Rollernet. Enfin, la table 4.3 liste les seuils utilisés pour chaque jeu de données.

Un candidat est écarté si un de ses scores est en dessous du seuil correspondant. C'est pourquoi il est important d'analyser quels sont les groupes filtrés par quels voisinages. Les corrélations entre chaque voisinage sont présentées pour le jeu de données Socio Pattern dans la figure 4.7. Les trois voisinages filtrent des candidats. Ils ne sont donc pas redondant. Même si un score élevé dans un voisinage est souvent corrélé avec un score élevé dans les autres, il arrive qu'un candidat est un score parfait dans un voisinage et score très faible dans les autres. Les corrélations entre chaque voisinage pour les autres jeux de données sont présentées en appendice pages 84 à 86 dans les figure A.3,A.7et A.11.

Nous présentons maintenant une analyse manuel pour deux candidats extraits des jeux de données Socio Pattern et Rollernet, afin d'illustrer les scores définis précédemment. Nous avons choisi ces jeux de données pour une analyse manuelle car, en plus des données brutes, des métadonnées sont également associées ce qui aident la validation des résultats.

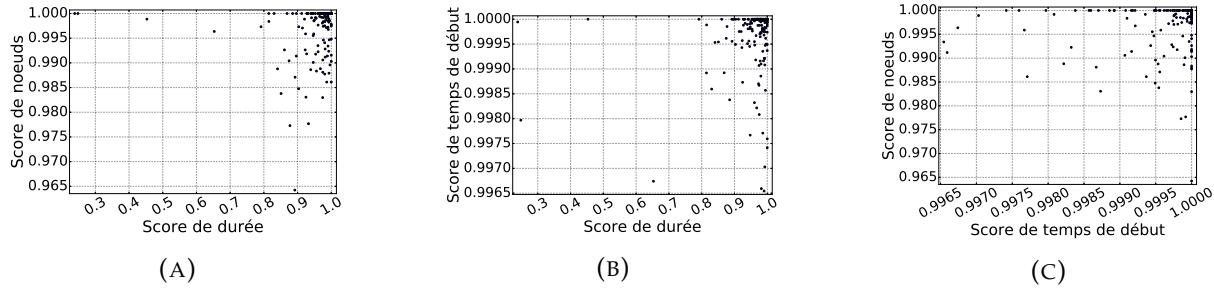


FIGURE 4.7 – Corrélations des scores selon le voisinage dans le jeu de données Socio Pattern.

Étude manuel d'un groupe de Socio Pattern

Dans les données Socio Pattern, la classe de chaque participant est connue et nous connaissons également quand commencent les premiers cours et quand ont lieu les pauses. Le groupe que nous considérons contient 50 liens entre 17 nœuds et dure environ 15 minutes, ce qui en fait un des plus gros groupes trouvés. Comme ce groupe commence à 7h44 le deuxième lundi de la capture, il précède le premier cours de la journée qui a lieu à 8h. De plus parmi les élèves participant à ce groupe, 16 font parti de la même classe. C'est pourquoi il est fort probable que ce groupe corresponde à un regroupement des élèves avant le premier cours.

Cette analyse est une première étape mais ne suffit pas. En effet, il est possible que ce regroupement ait en réalité concerné d'autres nœuds, duré plus longtemps ou même commencé à un autre instant. C'est pourquoi nous avons développé les notions de voisinages et nous avons calculé les scores associé à ce groupe.

Voisinage aux nœuds

Pour ce voisinage, le groupe a un score de 0.987. Cela implique que ce groupe a une densité plus importante 98% des sous-flots ayant des nœuds similaires et exactement le même intervalle de temps. Dans le cas de ce groupe à 17 nœuds, il y a $2771 = 17(180 - 17)$ sous-flots dans le voisinage aux nœuds. Ce score élevé nous permet d'être raisonnablement sûr que cet ensemble de nœuds est véritablement important dans cet intervalle de temps.

Voisinage au temps de début

Pour ce voisinage, la densité en fonction du temps de début du sous-fLOT voisin est présentée dans la figure 4.8a. Les cycles circadiens sont clairement ainsi que la fin de semaine. Par ailleurs, on remarque qu'avec une densité de 0.04 le groupe est plus dense que tous les sous-flots ayant la même durée et les mêmes nœuds mais un temps de début différent. C'est pourquoi le groupe obtient un score de 1 pour ce voisinage.

Voisinage à la durée

Pour finir, le groupe a un score de 0.95 pour le voisinage à la durée et donc le groupe a capturé une durée particulièrement pertinente. La densité en fonction de la durée des sous-flots voisins est présentée dans la figure 4.8b. La densité du sous-fLOT est légèrement plus importante si une durée plus longue est utilisée. Ce n'est pas complètement surprenant car la durée du sous-fLOT impacte la densité.

Par ailleurs, il est important de noter deux chose lorsque l'on observe cette courbe. Tout d'abord, on se rend compte que considérer des durées plus longue n'est pas vraiment utile. En effet pour ces jeux de données, la fonction de densité tend vers 0 lorsque la durée est très

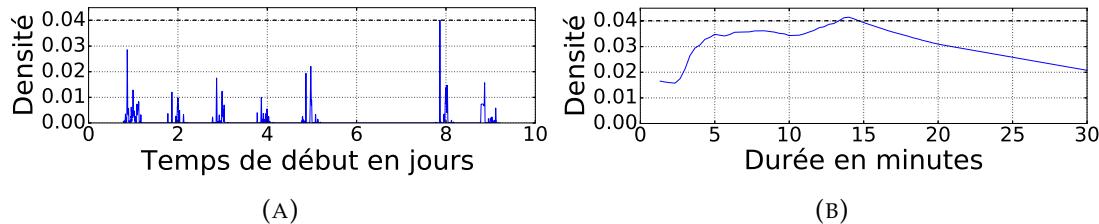


FIGURE 4.8 – Les figures représentent les densités du voisinage d'un groupe dans Socio Pattern. En trait plein : la fonction $d(V(C_i), z, \bar{C}_i)$ en (A) et la fonction $d(V(C_i), \beta(C_i), y)$ en (B) . En pointillé, est rappelée la densité du groupe.

longue. C'est pourquoi, augmenter l'intervalle de durée pris en compte ne fait qu'augmenter les scores sans apporter d'information. Par ailleurs avec une durée arbitraire pour un sous-flot du voisinage, il est possible qu'il n'existe aucun groupe de liens exhibant exactement cette durée. En effet, un sous-flot induit par un ensemble de noeuds, V' sur un intervalle donné, I , peut mener à considérer des liens seulement une partie de leur durée. C'est le cas notamment si il existe un lien, $(b, e, u, v) \in E$ tel qu'il relie des noeuds de V' et qu'il commence pendant l'intervalle I mais finisse après. C'est pourquoi trouver un groupe de liens avec un score parfait pour la durée serait très surprenant.

Enfin, nous avons de plus fait un parcours extensif de l'ensemble des densité $d(V(C_i), y, z)$ pour $y \in [\alpha, \omega - \bar{C}_i]$ et $z \in [0.8\Delta_{min}, 1.2\Delta_{max}]$ pour ce groupe en particulier et nous obtenons qu'il est très souvent le plus dense.

Étude manuel d'un groupde de Rollernet

Dans le jeu de donnée Rollernet, nous connaissons le rôle des participants. Certains des participants ne sont connus que comme membres d'une association de roller mais d'autres ont des rôle plus précis tel que membre organisateur à l'arrière gauche du peloton. Ce genre d'information aide lors de l'étude d'un groupe. Le groupe que nous considérons contient 38 liens, 9 noeuds et dure pendant environ 5 minutes. Ce groupe de liens commence juste au début de la randonnée et 8 des membres du groupes font parti des membres organisateurs à l'arrière de la randonnée. Le dernier fait parti des membres organisateurs mais à l'avant. Ce groupe pourrait indiquer une discussion rapide avant le début du tour. Comme pour le groupe de Socio pattern, nous analysons les scores obtenus par ce groupe.

Voisinage aux noeuds

Pour ce voisinage, le groupe a un score de 0.99 ce qui indique une fois de plus que cet ensemble de noeuds est important lorsque l'on considère le même intervalle de temps.

Voisinage au temps de début

Pour le voisinage au temps de début, le groupe a un score de 0.85, voir la figure 4.9a. Comme le jeu de données Rollernet est plus dense que Socio Pattern, la densité est beaucoup plus stable en fonction du temps de début. Cela se reflète d'ailleurs sur le score obtenu qui est plus faible. Cependant comme le groupe de liens en question apparaît 5 minutes après le début de la randonnée, il correspond tout de même à un maximum local de la densité.

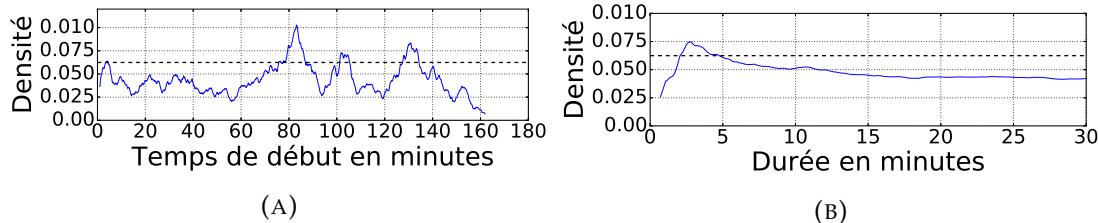


FIGURE 4.9 – Les figures représentent les densités du voisinage d'un groupe dans Rollernet. En trait plein : la fonction $d(V(C_i), z, \bar{C}_i)$ en (A) et la fonction $d(V(C_i), \beta(C_i), y)$ en (B) . En pointillé, est rappelée la densité du groupe.

Voisinage à la durée

Pour le voisinage à la durée, le groupe a un score de 0.86, voir la figure 4.9b. Encore une fois, le profil de la densité est plus lisse que dans le cas de Socio Pattern et une durée un peu plus courte donnerait lieu à une densité plus élevée. Enfin on remarque que, même si Rollernet est beaucoup plus dense que Socio Pattern, la densité décroît tout de même en fonction de la durée considérée.

Même si ces scores ne sont pas optimum et sont plus faibles que ceux obtenus dans l'exemple précédent, il faut les considérer dans le contexte du flot de liens de Rollernet qui est plus dense. Comme Rollernet est globalement dense, il est moins surprenant d'observer des sous-flots qui soient denses.

Pour résumer, il y a 12532 groupes candidats initialement dans le cas de Socio Pattern. Parmi eux, 155 ont plus de 10 liens et 136 sont considérés comme pertinents. C'est 136 groupes représentent 3507 liens soit 17.7% du flot de liens initial. La table 4.3 résume également le nombre de final de groupes pertinents capturé pour chaque jeux de données. On remarque notamment que même si peu de groupes sont capturés, ils représentent un très grande proportion des liens du flots de liens.

Tous ces groupes ont des scores plus élevés que les seuils que nous avons fixé mais pour la plupart ils n'ont pas des scores parfaits. Cela implique qu'il existe des sous-flots voisins ayant une densité plus importante. C'est pourquoi nous avons également comparé la densité obtenue par le groupe à celle obtenue par le meilleur des sous-flots dans un voisinage donné. Nous observons que dans 75% des cas la densité des groupes est à moins de 20% de l'optimale. C'est écart est relativement stable selon les jeux de données et les voisinages.

Les groupes que nous considérons comme pertinent ont donc une densité proche de l'optimum quand bien même cet optimum n'est pas atteignable par un groupe de liens. De plus, nous avons considéré les voisinages de manière séparée. Or, il se peut que le sous-flot optimum dans le voisinage au temps de début ne soit pas l'optimum dans le voisinage à la durée.

4.4.2 Caractéristiques des groupes pertinents

Afin d'avoir une vision plus précises des groupes capturés comme pertinents, nous présentons encore une fois la distribution cumulative inverse du nombre de liens, de nœuds, de la durée et de la densité sont présentées dans la figure 4.10. Les distributions sont encore une fois hétérogènes sauf pour la densité. Les distributions pour les autres jeux de données sont en appendice pages 84 à 86 dans les figures A.4,A.8et A.12 En plus de ces distributions, nous avons également étudié comment ces groupes de liens se répartissent dans le temps et la topologie de l'ensemble du flot de liens.

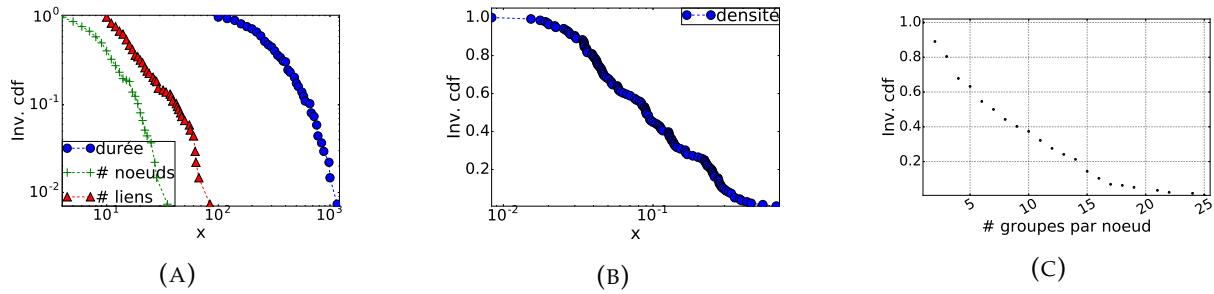


FIGURE 4.10 – Distributions cumulatives inverses du nombre de liens, de nœuds et de la durée en (A), de la densité en (B), du nombre de groupes par nœud en (C) pour les groupes capturés par notre méthode dans les données Socio Pattern.

Caractéristiques topologiques

Pour la répartition topologique des groupes, la distribution du nombre de groupes pertinent par nœuds est présenté dans la figure 4.10c pour le jeu de donnée Socio Pattern. Tous les nœuds appartiennent à au moins un groupe et quelques nœuds appartiennent même à plus de 20 groupes. Avec une moyenne de 7.4 groupes par nœuds, les groupes pertinents forment donc une structure très recouvrante sur les nœuds. Pour les autres jeux données voir les figures A.4c, A.8c et A.12c. Nous observons également sur ces jeux de données une structure très chevauchante sur les nœuds, en particulier pour le jeu de donnée Babouin. Cette différence est sûrement due au relative faible nombre de nœuds comparé au nombre de liens, 28 nœuds pour 95616 liens.

Il serait possible de penser que cette structure très chevauchante aurait pu être détectée par une méthode sur capturant des communautés chevauchantes de nœuds dans un graphe. Enfin de tester cette hypothèse, nous créons un graphe agrégé pondéré tel que le poids d'un lien entre deux nœuds soit égale à la somme des durées liens entre ces deux nœuds dans le flot de liens. Comme algorithme de détection de communauté chevauchante, nous avons utilisé l'algorithme BIGCLAM proposé par Yang *et al.* [YL13] que nous avons décrits dans le chapitre 1. Cette méthode ne considère que des graphes non pondérés. C'est pourquoi il est nécessaire de transformé le graphe pondéré en graphe non pondéré. Comme aucune valeur singulière de pondération n'est apparue lors de l'étude de la distribution des poids des liens, nous avons créé plusieurs graphes non pondérés selon une règle simple. La règle est la suivante : un lien existe dans le graphe non pondéré si le poids associé au lien est plus important ou égale à $\lambda\%$ des poids de tous les liens du graphe. Ainsi en faisant varier λ , il est possible de prendre en compte dans une certaine mesure l'information temporelle. Une fois les graphes construits, nous appliquons l'algorithme BIGCLAM sur chaque graphe et obtenons des partitions chevauchantes de nœuds.

Nous utilisons l'indice de Jaccard afin de comparer les partitions chevauchantes obtenues par BIGCLAM et celle définie par les nœuds induits par les groupes pertinents. L'indice de Jaccard, défini dans la section 1.1, nous permet de trouver pour chaque groupe pertinent quel est le groupe trouvé par BIGCLAM qui est le plus proche. La table 4.4 liste les similarités médianes et maximums entre les groupes pertinents et ceux trouvés par BIGCLAM selon la valeur de λ utilisée. Lorsque l'on étudie cette table, on remarque que les groupes pertinents et les groupes trouvés par BIGCLAM sont différents car les indices de Jaccard médians sont très faibles. Il y a toutefois quelques groupes pertinents qui sont retrouvés par BIGCLAM dans certains jeux de données, seulement cela n'arrive que pour quelques valeurs de λ et très peu de groupes. C'est pourquoi notre méthode a permis de mettre en valeur une structure des nœuds qui n'est pas retrouvée par une méthode classique sur le graphe agrégé.

	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.5$	$\lambda = 0.7$
Socio pattern	0.30 (0.55)	0.30 (0.55)	0.30 (0.55)	0.30 (0.55)	0.29 (0.54)	0.33 (0.85)
Rollernet	0.40 (0.52)	0.39 (0.64)	0.41 (0.54)	0.37 (0.54)	0.31 (0.57)	0.26 (0.75)
Reality Mining	0.42 (0.76)	0.42 (0.77)	0.36 (0.88)	0.42 (0.86)	0.38 (0.87)	0.36 (1)
Baboons	0.33 (0.95)	0.38 (1)	0.40 (0.84)	0.46 (0.94)	0.46 (0.93)	0.44 (0.85)

TABLE 4.4 – Indice de Jaccard médian (et maximum) entre les groupes pertinents et les groupes trouvés par BIGCLAM dans le graphe agrégé où les liens ayant un poids inférieur à $\lambda\%$ des liens ont été retirés. Les indices de Jaccard supérieurs à 0.8 sont en gras.

Caractéristiques temporelles

Pour l'aspect temporel, nous avons également étudié le chevauchement temporel entre les groupes pertinents. Pour ce faire, nous regardons la proportions du temps durant laquelle il n'existe aucun groupe présent. Pour tous les jeux de données excepté Rollernet, entre 82% et 95% du temps il n'y a aucun groupes présents. Cette proportion élevée s'explique par la nature des jeux de données qui couvrent de longue périodes, ce qui inclut des nuits où aucun liens n'apparaît. Pour le jeu de donnée Rollernet, cette proportion de temps sans groupes pertinents n'est que de 15% car il est beaucoup plus dense.

De manière plus importante, il existe également des instants où plusieurs groupes pertinents sont présents. Dans le jeu de donnée Socio Pattern, c'est notamment le cas lors des instants importants de la journée tel que les pauses ou le repas du midi. Durant ces instants, il peut y avoir jusque 4 groupes présents en même temps. Ces instants importants sont visibles via notre méthode mais ils sont bien évidemment facilement identifiable par d'autres méthodes plus simple tel que le degré temporel. De manière globale, la structure temporelle des groupes pertinents est très différentes de la structure topologique car une part importante du temps n'est impacté par aucun groupe pertinent. Il est toute fois intéressant de noter que les deux structures sont chevauchantes.

4.5 Conclusion et perspective

4.5.1 Conclusion

Dans ce chapitre, nous nous sommes intéressé à la détection de groupes de liens pertinents dans un flot de liens. À l'inverse des méthodes existantes, nous considérons pleinement l'information temporel et nous n'utilisons aucune agrégation pour évaluer nos résultats. Cette différence a été permise par l'utilisation du formalisme de flot de liens qui permet la définition de métrique mélangeant vraiment structure et temps. De plus, notre méthode ne capturent pas des ensembles de noeuds et des intervalles de temps mais des ensembles de liens.

Pour capturer des groupes de liens, nous avons proposé une transformation du flot de liens en un graphe statique non pondéré tel que les liens sont transformés en noeuds dans le graphe. Deux noeuds dans le graphe sont relié si les liens correspondant dans le flot ont au moins un noeud et un instant de temps en commun. C'est pourquoi, rechercher une communauté dans le graphe statique permet de trouver des groupes liens connectés temporellement et structuruellement. Pour ce faire nous utilisons l'algorithme de Louvain qui permet de capturer des partitions de noeuds dans un graphe. Nous obtenons ainsi un ensemble de groupes de liens potentiellement pertinents.

Afin de trouver les groupes pertinents parmi l'ensemble des groupes de la partition trouvée par Louvain, nous avons proposé un critère évaluant le sous-flot défini par les liens du groupes. Ce critère est basé sur la densité du sous-flot et sur plusieurs notions de voisinages. Ainsi, un

sous-flot est jugé pertinent en fonction de scores comparant la densité du sous-flot et la densité des sous-flots proches temporellement et topologiquement. Un sous-flot peut être caractérisé par les nœuds participants à ce sous-flot, l'instant d'apparition du premier lien du sous-flot et la durée du sous-flot. Il est donc naturel de considérer trois voisinages. Chaque voisinage est alors composé de sous-flots différent du sous-flot initial sur un seul aspect : l'ensemble de nœuds, le temps début ou la durée. Le score d'un groupe de liens dans un voisinage est ainsi défini par le nombre de sous-flot qui sont moins denses que lui. Finalement, le groupe de liens est jugé pertinent si ces scores sont plus élevés que des seuils qui sont fixés *a posteriori*. Ainsi, modifier ces seuils ne modifient les groupes candidats mais uniquement les groupes qui sont jugés pertinents.

Nous avons appliqué notre méthode sur quatre jeux de données réelles représentant des interactions entre individus. Deux d'entre eux sont interactions d'étudiant, un est un réseau de participant à une randonnée roller à Paris et le dernier est un réseau de babouins. Ils proviennent ainsi de contextes radicalement différents. Cela se traduit notamment sur la densité des flots de liens avec le jeu de données Rollernet ne durant que trois heures mais ayant autant de liens que Socio Pattern qui dure 9 jours. Pour tous ces jeux de données, nous avons réussi à capturer des groupes ayant des scores très élevés.

La validation de ces résultats est délicate car il n'existe pas de vérité de terrains sur ces données et il n'existe aucun algorithme similaire. Cependant il existe des métadonnées pour deux des jeux de données. De cette manière, nous avons pu analyser manuellement quelques groupes considérés comme pertinent par notre méthodes. Nous avons validé que les groupes étudiées correspondent à des événements existants dans ces jeux de données comme le regroupement d'étudiants avant le premier cours de la semaine. Nous avons également étudié comment les groupes pertinents se répartissent dans le temps et la topologie du réseau. Nous avons observé qu'ils sont très chevauchant sur les nœuds qu'ils concernent qu'une fraction du temps.

Afin de tester la nouveauté de la structure que nous capturons, nous avons appliqué un algorithme de détection de communautés chevauchante, BIGCLAM, sur le graphe agrégé. Il apparaît que parmi l'ensemble des groupes pertinents seulement quelques un sont retrouvés. Ce résultat illustre l'importance de prendre en compte le temps lors de l'étude d'un réseau.

4.5.2 Perspective

Notre méthode repose sur la projection du flot de liens en un graphe statique, d'une méthode de détection de communauté, une taille de groupe minimum et des seuils qui sont fixé *a posteriori*. Les axes de recherches associer à ces travaux sont principalement de deux types : l'amélioration de la détection des groupes pertinents et l'amélioration de la validation des groupes détectés.

Amélioration de la détection des groupes pertinents

Notre méthode n'est pas encore complètement automatique car il est nécessaire de fixer les seuils de score et la taille minimum d'un groupe. La taille minimum ne semble pas être très délicat à choisir. En revanche, les seuils nécessitent une plus grande connaissances des données. C'est pourquoi, il serait intéressant de pouvoir automatiser leur choix. Lors de l'application de notre méthode, nous avons à chaque fois observé une forte décroissance dans la distribution des scores. Une méthode prometteuse serait donc de se baser sur la détection de point d'infection qui peuvent être caractérisés par une dérivé seconde nulle. Cependant, le comportement que nous avons observé n'est pas forcément universel et il faudrait donc étudier des flots de liens provenant de différents contextes avant de généraliser cette approche.

Il serait également intéressant de remettre en question la méthode de détection utilisée. Nous utilisons l'algorithme de Louvain sur la projection du flot de liens en un graphe statique. D'une part, il est possible d'autres algorithmes de détection de communautés en tant que partition nœuds tels que ceux listés dans la sous section 1.2.1. Mais il est également possible de considérer les couvertures comme une liste de candidats potentiels. Il serait par exemple possible d'appliquer BIGCLAM. D'autre part, il est également nécessaire de réfléchir l'impact sur les groupes candidats de l'utilisation de tels algorithmes sur la projection du flot en un graphe car il est possible qu'un groupe pertinent ne soit pas détectable dans la projection. En effet, un lien dans la projection nécessite un chevauchement temporel des liens dans le flot de liens. Si deux liens sont séparés par un intervalle de temps, alors ils ne seront pas reliés dans la projection peu importe la durée les séparant. Il serait donc intéressant de détecter directement les groupes pertinents.

Il n'est pas possible, à partir de la projection, de calculer la densité et les voisinages d'un groupe de lien. Il est donc nécessaire de manipuler directement un flot de liens. Comme l'évaluation d'un groupe de liens est locale, une approche locale intégrant au fur et à mesure des liens dans un groupe candidat est envisageable. Le but est alors d'améliorer un groupe courant jusqu'à obtenir un équilibre de Nash, c'est à dire qu'il n'est pas possible d'améliorer un score sans dégrader les deux autres. Sans aller jusque la détection de groupes pertinents de manière autonomes, il serait également intéressant d'essayer d'améliorer localement les groupes candidats proposé par l'algorithme de Louvain.

Amélioration de la validation des résultats

Nous avons comparé nos résultats à une méthode statique et non la littérature dans les flots de liens car les méthodes de l'état de l'art manipulent principalement des séries de graphes. La détection de communautés et la recherche de groupes pertinent sont proches mais le changement de formalisme impacte fortement l'objet d'étude. Dans une série de graphes, il y a une agrégation temporelle. L'identité d'un lien est donc perdue lors de l'agrégation et il est ainsi délicat de comparer des groupes de liens dans un flot de liens et des groupes de nœuds dans une série de graphes. Toute fois, avec différentes transformations du flot en série de graphes, il serait intéressant de calculer les scores obtenus par chaque sous-flots induit par les nœuds d'une communauté sur l'intervalle de temps où la communauté a été détectée dans la série de graphes.

Cette approche permettrait de comparer plus en profondeur les résultats mais pas de les valider. Comme il n'existe pas de données avec une vérité de terrain, une approche prometteuse serait de créer des modèles génératifs de flots de liens permettant de définir des contraintes sur la densité. Ainsi, il serait alors possible de tester de manière quantitative notre méthode. Cela permettrait de plus d'ouvrir la voie pour des méthodes de détection de communautés dans les flots de liens.

Chapitre 5

Expected Nodes : communautés de liens dans les graphes statiques

Sommaire

5.1	Travaux existants	62
5.2	Définition d' <i>Expected Nodes</i>	65
5.3	Comparaison	68
5.3.1	Cas du graphe complet	68
5.3.2	Graphe LFR	69
5.4	Calcul et optimisation	73
5.5	Conclusion	74
5.5.1	Perspective	75

Les structures communautaires dans les graphes ont été beaucoup étudiées lorsqu'elles concernent les nœuds, voir le chapitre 1. Dans une moindre mesure, elles ont également été étudiées lorsqu'elles concernent les liens. Par exemple dans un réseau social, chaque personne a plusieurs centres d'intérêt : famille, sport, politique... Lorsque deux personnes interagissent, la communication a lieu dans un contexte bien particulier. Bien que les personnes aient plusieurs centres d'intérêts, la raison de leur communication est souvent unique. Il semble donc qu'une information importante soit intrinsèquement liée au lien. Via la recherche de partitions de liens d'un graphe, c'est cette information que nous cherchons à capturer.

Afin d'illustrer ce que capture une partition, prenons l'exemple d'un réseau de personnes fictif où le contexte de l'interaction est connu. Certaines personnes communiquent ensemble car elles sont de la même **famille**, pratiquent le même **sport**, jouent au **go** ensemble ou bien encore car elles **travaillent** ensemble. Nous illustrons cet exemple dans la figure 5.1 où les interactions

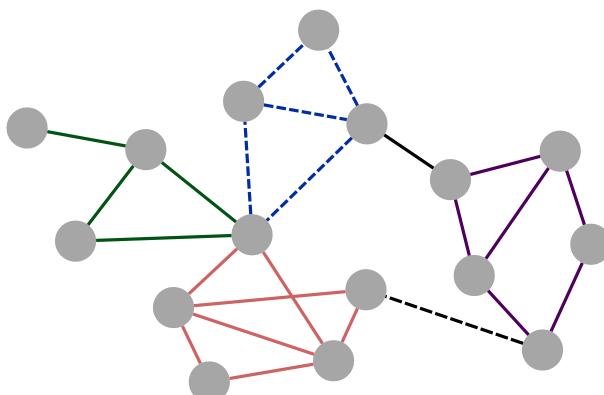


FIGURE 5.1 – Exemple de réseau de personnes avec une structure communautaire sur les liens qui est représentée par la couleur et le style de chaque lien.

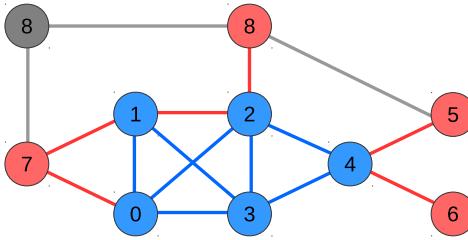


FIGURE 5.2 – Exemple d'un groupe de liens L (en bleu). Les liens rouges sont les liens adjacents L_{out} . Les nœuds internes V_{in} sont en bleu et les nœuds adjacents V_{out} en rouge.

sont colorées en fonction de leur contexte. Il est intéressant de noter que les interactions d'un même type se regroupent ensemble. Un nœud peut avoir des interactions de plusieurs types ; c'est le cas du nœud central qui a des interactions de type **famille**, **sport** et **go**. De même, certaines interactions font le lien entre différents groupes. C'est le cas du lien pointillé noir qui relie le **sport** et le **travail** ce qui pourrait se traduire par le financement de l'équipe par l'entreprise. Ainsi, les partitions de liens peuvent capturer des situations assez variées.

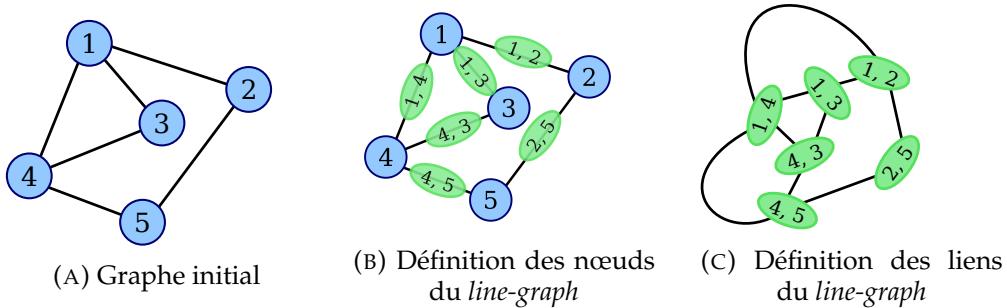
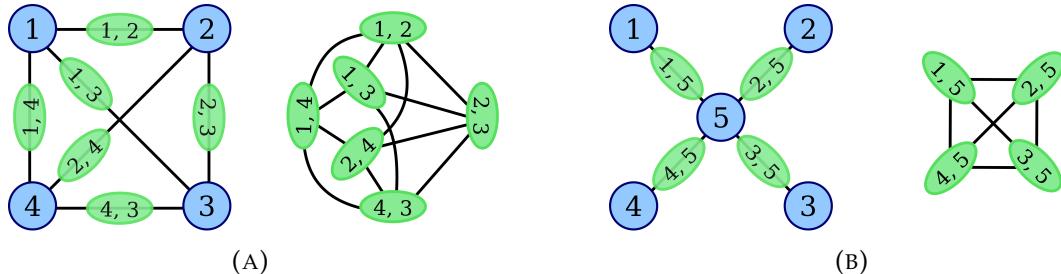
Il est également possible de manipuler des partitions chevauchantes ou couvertures. Dans ce cas, chaque nœud peut appartenir à plusieurs communautés. Pour répondre à ce problème de nombreux algorithmes ont été proposés pour la détection et l'évaluation de couvertures de nœuds, voir la section 1.2.2. Les couvertures sont une généralisation des partitions et aucune méthode ne fait encore consensus pour les évaluer car leur structure est complexe, voir la sous-section 1.2.2. Les partitions de liens, quant à elles, restent des objets plus simples à manipuler. De plus, elles permettent de mettre en avant une autre structure ayant également du sens.

Il apparaît donc que les partitions de liens sont des objets à part entiers. Pour ce faire, il est nécessaire d'adapter les outils d'analyse pour évaluer directement les partitions de liens. Nous développons ici une approche similaire à ce qui est fait pour les partitions de nœuds et la modularité [NG04]. Le but est de créer une fonction de qualité permettant d'évaluer une partition de liens d'un graphe.

Dans la suite, nous utiliserons les notations suivantes. Soit $G = (V, E)$ un graphe non-orienté avec V l'ensemble des nœuds de taille n et $E \subseteq V \times V$ l'ensemble des liens de taille m . Le degré d'un sommet u de G est noté $d_G(u)$. Une partition des liens en k groupes est notée $\mathcal{L} = (L_1, L_2, \dots, L_k)$ avec $L_i \subseteq E \forall i$, $L_i \cap L_j = \emptyset \forall i \neq j$ et $\bigcup_i L_i = E$. Pour un groupe de liens $L \in \mathcal{L}$, on pose $V_{in} = \{u \in V, \exists (u, v) \in L\}$ l'ensemble des nœuds internes au groupe L , $V_{out} = \{u \in V \setminus V_{in}, (u, v) \in E \wedge v \in V_{in}\}$ représente les nœuds adjacents au groupe L et enfin $L_{out} = \{(u, v) \in E \setminus L, u \in V_{in} \vee v \in V_{in}\}$ l'ensemble des liens adjacents au groupe L (voir la figure 5.2).

5.1 Travaux existants

Une approche naïve serait de transformer le graphe initial en un *line-graph*. Un *line-graph* est un graphe où chaque lien du graphe initial est transformé en un nœud dans le *line-graph*. Deux nœuds du *line-graph* sont reliés si les liens correspondants ont au moins un nœud en commun, voir la figure 5.3. Comme un *line-graph* est un graphe classique, on peut y appliquer toutes les méthodes déjà existantes, *e.g.* les algorithmes de détection de communautés. Ainsi, une partition des nœuds du *line-graph* représente une partition des liens du graphe initial. Sur l'exemple de la figure 5.3, les liens $(1, 4)$, $(1, 3)$ et $(4, 3)$ du graphe forment un triangle dans le *line-graph* et pourraient être capturés comme étant une communauté. Ces liens dans le graphe initial forment également un triangle et peuvent être considérés comme une communauté valide.

FIGURE 5.3 – Exemple de construction du *line-graph*¹.FIGURE 5.4 – Exemple de construction du *line-graph* d'une clique en (A) et d'une étoile en (B).

Cependant pour que ce type de méthode fonctionne, il est nécessaire que le *line-graph* résultant puisse être analysé comme un graphe. En particulier, il est nécessaire que la notion de communauté dans le *line-graph* ait un sens dans le graphe initial. Or, un *line-graph* a une structure très différente du graphe initial.

Prenons pour l'exemple, la clique qui est la meilleure communauté possible et l'étoile qui est une des pires communautés possibles. Le but est d'observer comment ces structures sont transformées dans le *line-graph*. Ces situations sont représentées dans la figure 5.4. L'étoile dans la figure 5.4b est transformée en une clique de 4 noeuds. Une des pires structures communautaires d'un graphe est transformée dans le *line-graph* en la meilleure structure communautaire. En effet, chaque noeud de degré k du graphe initial donne lieu à une clique de taille k dans le *line-graph*. Les cliques du *line-graph* ne sont donc pas forcément des communautés dans le graphe. Dans le cas de la clique de taille 4 dans la figure 5.4a, on remarque que le *line-graph* est composé de 6 noeuds et de uniquement 12 liens. Plus généralement une clique de taille n dans le graphe initial donne lieu dans le *line-graph* à $\frac{n(n-1)}{2}$ noeuds et $(n-1)n$ liens. Ainsi, plus la clique est grande dans le graphe, moins la structure résultante dans le *line-graph* est dense. Les cliques du graphe sont donc moins denses que ses étoiles, lorsqu'on les observe dans le *line-graph*.

Il n'est donc pas trivial d'utiliser le *line-graph* pour trouver des communautés de liens. Il est nécessaire d'adapter les outils à ce type de graphe.

Il existe d'autres méthodes que le *line-graph* pour la détection et l'évaluation de partitions de liens.

Les algorithmes de propagation de labels ont été étendus pour capturer des partitions de liens [YWW13] Cependant, ils ne permettent pas d'évaluer la qualité de la partition obtenue.

Il y a des méthodes évaluant une partition de liens via la transformation de la partition en une couverture de noeuds [HWW⁺13, LRK⁺14, WLWT10]. Une transformation classiquement

1. Image provenant de https://en.wikipedia.org/wiki/Line_graph.

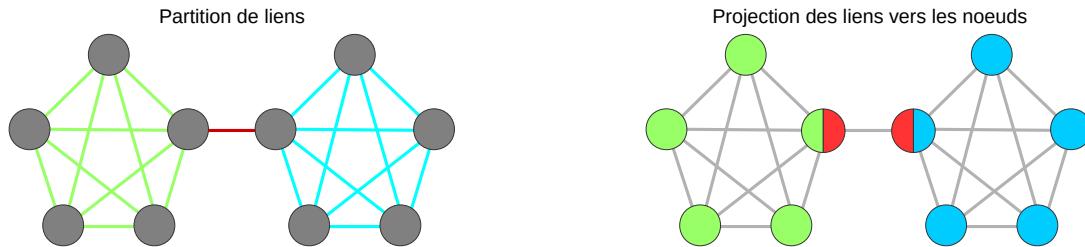


FIGURE 5.5 – Transformation d'une partition de liens à gauche en couverture de nœuds à droite. La couleur représente un groupe.

utilisée est qu'un nœud dans la couverture prend comme communautés l'ensemble des communautés de ses liens, voir la figure 5.5. Il serait alors tentant de considérer que les partitions de liens et les couvertures de nœuds sont équivalentes. Ainsi pour évaluer une partition de liens, il suffirait de transformer la partition en couverture. Or, ce changement n'est pas anodin. D'une part, les couvertures de nœuds permettent de modéliser beaucoup plus de situations car il n'y a aucune contrainte sur les couvertures. D'autre part, transformer une partition de liens en couverture de nœuds impose des contraintes sur la couverture résultante.

Par exemple dans l'exemple de la figure 5.5, il n'est pas évident que la communauté **rouge** constituée des deux nœuds centraux soit une communauté légitime. Selon le contexte, elle peut être considérée comme un artefact de la transformation. La transformation d'une partition n'est donc pas un acte neutre. Cet aspect a d'ailleurs été mis en avant par Esquivel *et al* [ER11]. Face à ce problème, nos travaux ainsi que quelques méthodes existantes proposent des méthodes évaluant directement les partitions de liens.

Ahn *et al.* [ABL10] sont parmi les premiers à avoir proposé une méthode détectant les communautés de liens. Leur méthode *link clustering* est une méthode hiérarchique d'agglomération. Elle construit un dendrogramme en agglomérant de manière itérative les groupes de liens en fonction de leur similarité, calculée par l'indice de Jaccard. Afin de décider de la coupe du dendrogramme et de la partition résultante, la fonction *Partition Density* est utilisée. Pour une partition de liens donnée \mathcal{L} , la *Partition Density* est définie de la manière suivante :

$$D(\mathcal{L}) = \frac{\sum_{L \in \mathcal{L}} |L| D(L)}{m}, \text{ avec } D(L) = \frac{|L| - \min_D(|V_{in}|)}{\max_D(|V_{in}|) - \min_D(|V_{in}|)}, \quad (5.1)$$

où $\min_D(N) = N - 1$ est le nombre minimum de liens nécessaire pour relier N nœuds et $\max_D(N) = \frac{N(N - 1)}{2}$ est le nombre maximum de liens qu'il puisse exister entre N nœuds. Malgré son nom la *Partition Density* n'est pas une densité mais le nombre de liens du groupe normalisé par le nombre de liens minimum et maximum possible pour un groupe de $|V_{in}|$ nœuds. Après simplification, on obtient la formule suivante :

$$D(L) = 2 \frac{|L| - (|V_{in}| - 1)}{(|V_{in}| - 1)(|V_{in}| - 2)}. \quad (5.2)$$

Par convention, un groupe constitué d'un unique lien, et qui n'a donc que deux nœuds internes, a une qualité nulle.

D'autres chercheurs [LZW⁺13, SCF⁺13] ont par la suite utilisé la *Partition Density* comme fonction à optimiser dans des algorithmes génétiques. Leurs solutions semblent pour l'instant difficilement utilisable car leurs algorithmes reposent sur de nombreux critères et sont limités à de petit graphes.

Par ailleurs, la *Partition Density* ne peut pas être directement appliquée aux graphes pondérés. Une première proposition de Kim [Kim14] a été faite dans ce sens.

Evans *et al.* [EL09] proposent trois fonctions de qualité pour évaluer les partitions de liens. Leurs fonctions de qualité sont basées sur trois marches aléatoires qui se déroulent sur les liens du graphe. L'approche est similaire à la modularité car la modularité peut également être définie à l'aide d'une marche aléatoire sur les nœuds du graphe [DYB10]. Leurs trois fonctions de qualité peuvent être calculées et optimisées sur le graphe mais les auteurs ont montré que l'on pouvait, de manière complètement équivalente, utiliser la modularité sur des *line-graph* pondérés (LG_1 , LG_2 , LG_3). Ainsi, il suffit de construire le *line-graph* approprié puis d'utiliser un algorithme existant d'optimisation de la modularité tel que l'algorithme de *Louvain* [BGLL08].

Pour construire les *line-graphs* LG_1 , LG_2 et LG_3 , nous définissons $B \in \mathcal{M}_{n,m}$ la matrice d'incidence du graphe G : un élément $B_{i\alpha}$ de cette matrice $|V| \times |E|$ est égal à 1 si le lien α est relié au nœud i et 0 sinon. Les matrices LG_1 , LG_2 et LG_3 sont alors définies de la manière suivante :

	$x = 1$	$x = 2$	$x = 3$
$LG_x(\alpha, \beta)$	$B_{i\alpha}B_{i\beta}(1 - \delta_{\alpha\beta})$	$\sum_{i \in V, d_G(i) > 1} \frac{B_{i\alpha}B_{i\beta}}{d(i) - 1}$	$\sum_{i,j \in V, d(i)d_G(j) > 0} \frac{B_{i\alpha}A_{ij}B_{j\beta}}{d(i)d(j)}$

Soit $k_x(\alpha) = \sum_\beta LG_x(\alpha, \beta)$ le degré pondéré dans le line graphe LG_x du nœud représentant le lien α et $W_x = \sum_{\alpha, \beta \in |E|} LG_x(\alpha, \beta)$ la somme des poids des liens. Pour $x \in \{1, 2, 3\}$, la fonction de qualité $Evans_x$ est définie de la manière suivante :

$$Evans_x(\mathcal{L}) = \frac{1}{W_x} \sum_{L_i \in \mathcal{L}} \sum_{e_1, e_2 \in L_i^2} LG_x(e_1, e_2) - \frac{k_x(e_1)k_x(e_2)}{W}. \quad (5.3)$$

Kim *et al.* [KJ11] ont exploré une extension du concept de *Minimum Length Description* (MDL) introduit par Rosvall *et al.* [RB08] qui est une méthode provenant de la théorie de l'information. Cette extension de la MDL évalue directement une partition de liens, contrairement à l'extension proposée par Esquivel *et al.* [ER11]. Un avantage de leur méthode est de pouvoir comparer la qualité d'une partition de liens et d'une partition de nœuds avec leur MDL respective. Cependant, leur méthode ne semble favoriser les communautés de liens que dans des cas très limités.

Résumé

Il existe des méthodes pour capturer et évaluer des partitions de liens dans un graphe. Deux d'entre elles semblent faire consensus pour l'instant. D'une part, la *Partition density* est une fonction de qualité comparant le nombre de liens observé avec les nombres minimum et maximum de liens possibles entre les mêmes nœuds induits. D'autre part, les fonctions de qualité $Evans_x$ se basent quant à elles sur des marches aléatoires sur les liens et d'une certaine manière sur un processus similaire à la modularité. Il n'existe cependant aucune méthode utilisant une comparaison d'une métrique à ce qui est attendu dans un modèle nul. Or, ce processus, qui est à l'origine de la modularité, permet de mettre en avant des structures qui diffèrent du modèle nul et qui sont donc intéressantes à capturer.

5.2 Définition d'Expected Nodes

Une idée souvent utilisée lors de la détection de communautés de nœuds est qu'une communauté devrait avoir beaucoup de liens en interne. Pour ce faire, la modularité compare le nombre de liens partagés par un groupe de nœuds au nombre attendu dans le modèle de configuration, que nous avons évoqué dans la section 1.2.1. Pour notre fonction de qualité *Expected Nodes*, nous utilisons également le modèle de configuration mais avant de définir formellement *Expected Nodes*, il est utile d'avoir une définition informelle de la fonction de qualité.

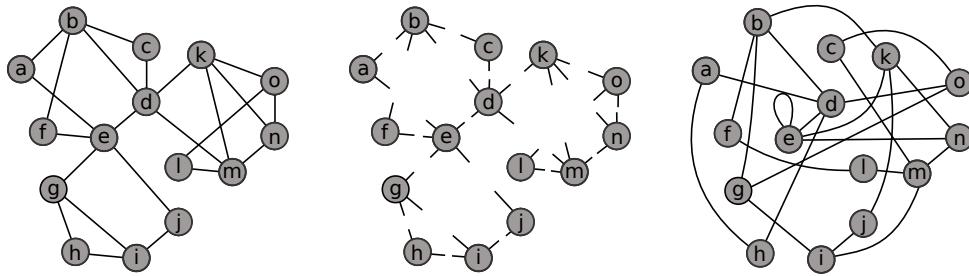


FIGURE 5.6 – Exemple d'une génération du modèle de configuration avec à gauche le graphe initial, au milieu la mise en avant des demi-liens et à droite une génération aléatoire du modèle de configuration

Le but est d'évaluer un groupe de liens. Afin qu'un groupe de liens soit évalué comme une bonne communauté, les liens devraient induire un nombre relativement faible de nœuds internes. En effet, plus le nombre de nœuds internes est faible, plus le groupe de liens ressemble à une clique. De manière similaire à la modularité, nous utilisons le configuration modèle pour calculer le nombre de nœuds internes attendu dans le modèle de configuration. Si le groupe de liens a moins de nœuds internes qu'attendu alors cela indique que le groupe de liens est plus dense et qu'il devrait donc avoir une évaluation élevée.

Il est donc nécessaire de calculer l'espérance du nombre de nœuds interne, μ_G , d'un groupe de liens, L , dans le modèle de configuration. Afin d'y parvenir, il est nécessaire de comprendre en détail le modèle de configuration. Le modèle de configuration permet de mélanger les liens d'un graphes de tel sorte que les nœuds aient toujours autant de liaisons que dans le graphe initial. Un moyen de se représenter ce mélange est présenté dans la figure 5.6. De manière imagée, il s'agit de couper tous les liens en deux pour créer des demi-liens. Puis, il suffit de reconnecter aléatoirement tous les demi-liens pour obtenir une génération du modèle de configuration. Donc afin d'étudier comment un groupe de $|L|$ liens est transformé dans le configuration modèle, il est nécessaire d'étudier comment $2|L|$ demi-liens sont mélangés dans le modèle.

Un nœud est interne à L si au moins un de ses liens appartient à L . Ainsi pour calculer la probabilité qu'un nœud soit interne dans le modèle de configuration, il faut donc calculer la probabilité qu'au moins un de ses demi-liens soit choisi lors du tirage aléatoire et sans remise de $2|L|$ demi-liens parmi les $2|E|$ demi-liens du graphe. Soit B_u la variable aléatoire correspondant au nombre de fois où le nœud u est tiré. Cette variable suit une loi hypergéométrique $B_u \sim \mathcal{G}(2|E|, d_G(u), 2m)$. Avec cette notation, on définit μ_G de la manière suivante :

$$\mu_G(|L|) = \sum_{u \in V} \mathbb{P}(B_u \geq 1) = \sum_{u \in V} 1 - \frac{\binom{2|E|-d(u)}{2|L|}}{\binom{2|E|}{2|L|}}. \quad (5.4)$$

Voici quelques propriétés de la fonction $\mu_G(|L|)$:

- La fonction μ_G dépend uniquement de la séquence de degrés $\{d_G(v)\}_{v \in V}$ et du nombre de liens.
- Pour une distribution de degrés donnée, la fonction $\mu_G(|L|)$ est une fonction croissante de $|E|$.
- Si $L = E$, alors le nombre de nœuds attendus est bien égal à $|V|$.
- On a $\mu_G(1) \leq 2$, en effet le modèle nul n'interdit pas la présence de boucle.

Avec μ_G , nous pouvons définir la qualité *interne*, Q_{in} d'un groupe de liens L :

$$Q_{in}(L) = \frac{\mu_G(|L|) - |V_{in}(L)|}{\mu_G(|L|)}. \quad (5.5)$$

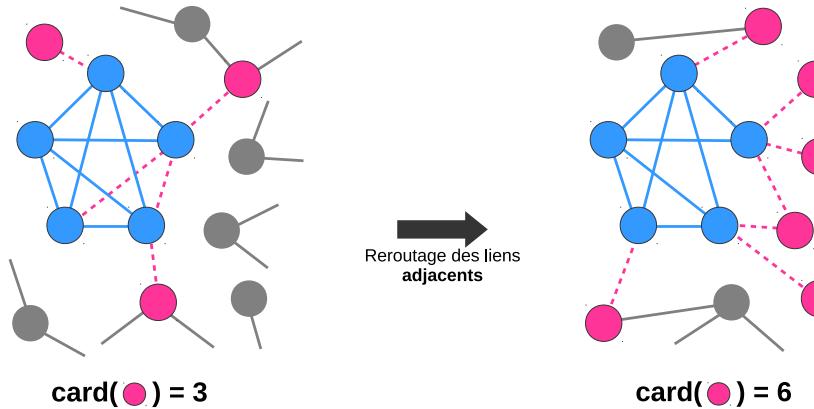


FIGURE 5.7 – Groupe de liens L en bleu et ces liens adjacents en rose pointillés dans le graphe initial à gauche. À droite, une réalisation du modèle de configuration où L a été figé.

Avec cette formulation, pour un groupe de taille $|L|$, plus le nombre de nœuds internes est faible, plus Q_{in} sera élevée.

Q_{in} permet d'évaluer la qualité interne d'un groupe mais il faut aussi tenir compte du voisinage. En effet, observer une clique à l'intérieur d'une autre clique n'est absolument pas surprenant. C'est pourquoi, nous définissons également une qualité externe. Le but est d'évaluer comment sont répartis les liens et nœuds adjacents. Pour ce faire, nous allons également comparer le nombre de nœuds adjacents observé au nombre espéré dans le modèle de configuration. Cependant à l'inverse de la qualité interne, la qualité externe est **mauvaise** si jamais le nombre de nœuds adjacents est plus faible qu'espéré. En effet, si il y a beaucoup de liens adjacents pour peu de nœuds, alors cela indique que le voisinage du groupe est dense et devrait être inclus dans le groupe. Le cas idéal est que chaque lien adjacent soit relié à un nœud différent.

Pour un nœud interne u , soit $\bar{d}(L, u) = \sum_{v \in V} \mathbf{1}_{(u,v) \in E \setminus L}$ le degré de u limité aux liens adjacents et $\bar{d}(L) = \sum_{u \in V_{in}(L)} \bar{d}(L, u)$. L'espérance du nombre de nœuds adjacents est calculé comme le nombre de nœuds tirés lorsque $\bar{d}(L)$ demi-liens sont choisis aléatoirement et sans remise dans le modèle de configuration où les liens de L ont été préalablement retirés. Ce graphe aléatoire a la distribution de degrés suivante : $\{d_{G \setminus L}(u)\}_{u \in V}$ où $G \setminus L = (V, E \setminus L)$. Dans ce cas, on ne tire pas aléatoirement un lien mais uniquement un demi-lien car l'autre demi-lien est un des demi-liens reliés aux nœuds internes. L'espérance du nombre de nœuds adjacents se définit de la manière suivante :

$$\mathbb{E}[\bar{d}(L)] = \mu_{G \setminus L}(\bar{d}(L)/2). \quad (5.6)$$

Une illustration de ce processus est présentée dans la figure 5.7. Sur cette illustration, le groupe L a un très mauvais voisinage et cela se reflète par un nombre de nœuds adjacents observés plus faible qu'espéré. En particulier, se sont les liens adjacents reliant deux nœuds internes qui pénalisent l'évaluation car ils comptent chacun pour deux demi-liens adjacents.

Comme il est intéressant de pénaliser les groupes ayant de mauvais voisinage mais qu'un bon voisinage n'est pas suffisant pour définir une bonne communauté, nous bornons à 0 la qualité externe :

$$Q_{ext}(L) = \min \left(0, \frac{|V_{out}(L)| - \mu_{G \setminus L}(\bar{d}(L)/2)}{\mu_{G \setminus L}(\bar{d}(L)/2)} \right). \quad (5.7)$$

Enfin, nous définissons *Expected Nodes* pour un groupe L :

$$Q(L) = 2 \frac{|L|Q_{in}(L) + |L_{out}|Q_{ext}(L)}{|L| + |L_{out}|}. \quad (5.8)$$

La qualité interne est due aux liens de L et la qualité externe aux liens adjacents. C'est pourquoi nous pondérons Q_{in} par $|L|$ et Q_{ext} par $|L_{out}|$.

Nous détaillons maintenant certaines propriétés des formules 5.7 et 5.8 découlant des propriétés de μ_G en nous appuyant sur des exemples. En s'intéressant aux nœuds adjacents V_{out} , on pénalise la présence de liens adjacents entre deux nœuds internes et la présence de nœuds adjacents fortement connectés avec les nœuds internes à L . Prenons le cas extrême où L est une clique qui est incluse dans une plus grande clique alors que le reste du graphe est quelconque. Dans cette situation, $Q_{in}(L)$ est maximum car le groupe est une clique. En revanche, $Q_{ext}(L)$ va fortement pénaliser la qualité du groupe car il y a dans ce cas beaucoup de liens adjacents pour très peu de nœuds adjacents. Ainsi, la Q_{ext} permet de pénaliser les nœuds adjacents ayant beaucoup de liens avec L . L'idée est que ces liens adjacents devraient être intégrés à L .

Bien évidemment, la solution n'est pas de systématiquement intégrer les liens adjacents car intégrer un lien adjacent peut également faire baisser Q_{in} . Par exemple, un lien reliant deux groupes denses disjoints peut avoir une qualité positive. Cette situation est visible dans la figure 5.5 partie gauche. Ce lien tout seul a d'une part une qualité interne positive car $\mu_G(1) \leq 2$ et d'autre part il a une qualité externe nulle car chaque nœud adjacent n'est relié qu'à un seul lien adjacent ce qui est le cas idéal. Dans une situation plus générale, la qualité d'un lien isolé dépend du nombre de triangles dans lequel il se trouve. Moins le nombre de triangle est élevé, meilleure sera la qualité externe.

Enfin il est intéressant de noter que la qualité du groupe contenant tout les liens est nulle. En effet comme dit précédemment, $Q_{in}(E)$ est égal à zéro, voir l'équation 5.4. De plus, si le groupe contient tout les liens alors il n'y a plus de liens adjacents et donc seule $Q_{in}(E)$ influe sur $Q(E)$.

Nous définissons *Expected Nodes* pour une partition de liens \mathcal{L} comme la moyenne pondérée de la qualité de chaque groupe :

$$Q_G(\mathcal{L}) = \frac{\sum_{L \in \mathcal{L}} |L|Q(L)}{|E|}. \quad (5.9)$$

D'autres choix de pondération pour Q_{in} , Q_{ext} et Q_G ont été testés en utilisant le nombre de nœuds au lieu du nombre de liens mais elles ont été abandonnées lors des tests qui sont présentés dans la section 5.3.

5.3 Comparaison

Nous évaluons maintenant *Expected Nodes* en utilisant deux jeux de test. Sur ces jeux de tests, nous appliquons également des fonctions de qualités reconnues : *Partition Density* [ABL10] et les fonctions de qualité proposées par Evans *et al.* [EL09] que nous nommons *Evans1*, *Evans2* et *Evans3*. Pour chaque graphe de test, nous créons empiriquement plusieurs partitions de liens et nous évaluons chaque partition avec toutes les fonctions de qualité.

5.3.1 Cas du graphe complet

Le premier jeu de test est assez simple puisqu'il s'agit d'un graphe complet. Le but est de vérifier que *Expected Nodes* n'a pas un comportement dégénéré. Nous étudions un graphe

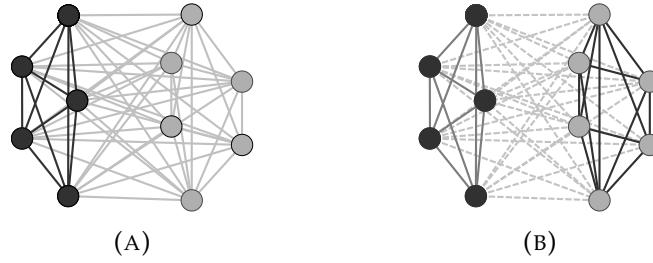


FIGURE 5.8 – Deux partitions de liens pour un graphe complet à 10 nœuds avec $p = 5$: (a) partition en deux groupes et (b) partition en trois groupes. Les nœuds noirs sont les nœuds appartenant à V' et la couleur d'un lien correspond à son groupe.

complet de 100 nœuds². Sur ce graphe, nous définissons plusieurs partitions. La première est la partition triviale où tout les liens sont dans un unique groupe. Nous définissons également deux familles de partitions : une séparant les liens en deux groupes et une séparant les liens en 3 groupes. Soit V' un ensemble de p nœuds où p est un paramètre $p < |V|$. Les deux familles de partitions placent les liens de $V' \times V'$ dans un groupe. Pour la partition en 2 groupes, tous les autres liens sont mis dans un second groupe. Pour la partition en 3 groupes, les liens de $V' \times V \setminus V'$ sont dans un second groupe et les liens de $V \setminus V' \times V \setminus V'$ sont dans un troisième. Ces répartitions sont illustrées dans la figure 5.8.

Comme le graphe est un graphe complet, la meilleure solution est d'avoir un seul groupe contenant l'ensemble des liens, *i.e.* la partition triviale devrait avoir une meilleure évaluation que les autres partitions. La figure 5.9 présente les résultats. Pour chaque valeur de p et chaque fonction de qualité, nous calculons les évaluations des partitions en deux et en trois groupes ainsi que l'évaluation de la partition triviale. L'évaluation de la partition triviale n'est pas dépendante de p et n'est calculée qu'une seule fois. Tout d'abord, la construction de cet exemple est symétrique pour la partition en deux groupes. En effet, la partition en deux groupes lorsque p est égal à 24 est complètement équivalente à la partition en deux groupes lorsque p est égal à 76. Par ailleurs, il est possible de définir formellement la qualité de chacune de ces partitions selon toutes les fonctions de qualité. Cependant, la complexité des équations, surtout pour *Expected Nodes* et *Evans_x*, rend le résultat difficilement interprétable.

De manière assez surprenante, les fonctions *Evans1* et *Evans2* ne passent pas ce test car elles évaluent la partition en deux ou trois groupes comme meilleure que la partition triviale. Selon la *Partition Density*, *Expected Nodes* et *Evans3*, la partition triviale est la meilleure des partitions. La fonction *Evans3* diffère légèrement des deux autres car elle a une amplitude plus faible ($\approx 10^{-3}$).

5.3.2 Graphe LFR

Nous utilisons maintenant un jeu de test plus évolué. Il n'existe pas à notre connaissance de générateur de graphe aléatoire avec une structure communautaire sur les liens. C'est pourquoi, nous utilisons le générateur proposé par Lancichinetti *et al.* [LF09a]. Ce générateur aléatoire permet de générer des graphes ayant une structure communautaire chevauchante sur les nœuds. Comme nous voulons évaluer une partition de liens, il est nécessaire de transformer cette vérité de terrain. Nous introduisons deux transformations de la couverture des nœuds en deux partitions de liens, *TA* et *TB*, voir figure 5.10.

Reprenons l'exemple d'un réseau d'interactions de personnes où chaque personne appartient à différents groupes. Afin de simplifier l'exemple, nous considérons qu'il n'existe que

2. Nous avons obtenu des résultats similaires pour un graphe de 500 nœuds.

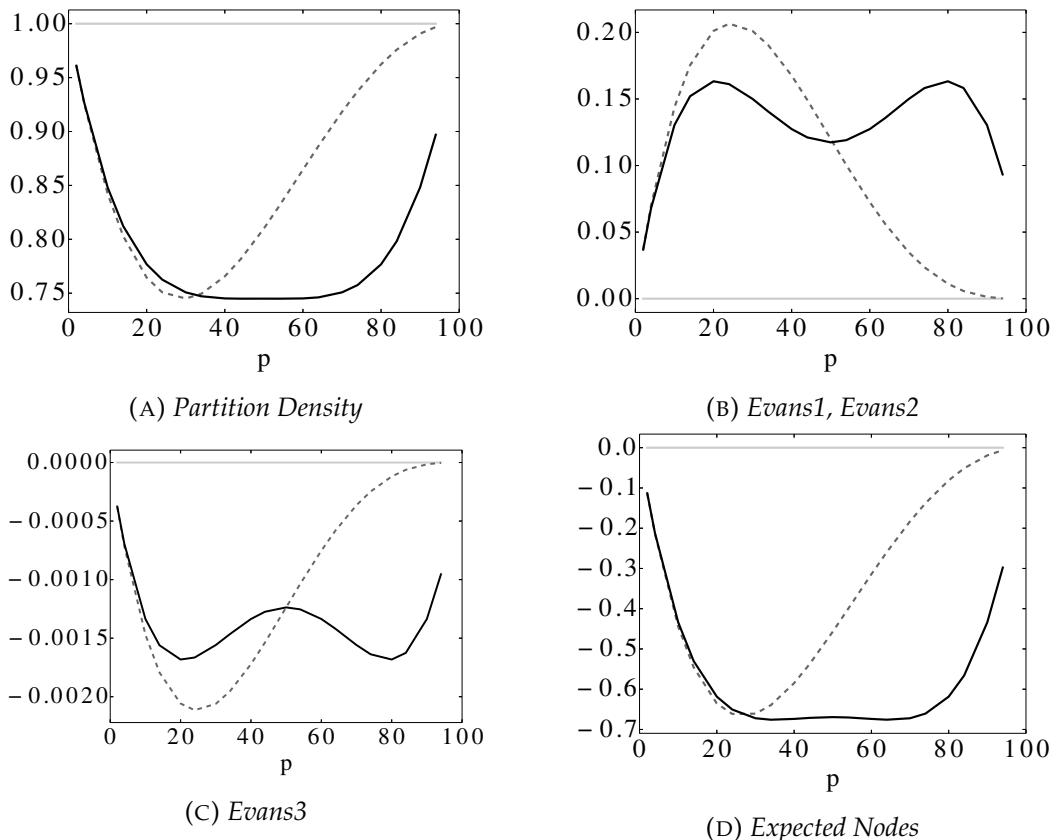


FIGURE 5.9 – Évaluation des 5 fonctions de qualité sur un graphe complet de 100 nœuds pour trois type de partitions. Les partitions testées sont présentées dans la section 5.3.1. Par définition, les résultats pour *Evans1* et *Evans2* sont identiques. Les lignes en gris, noir et pointillées représentent respectivement la partition triviale, les partitions en deux groupes et les partitions en trois groupes.

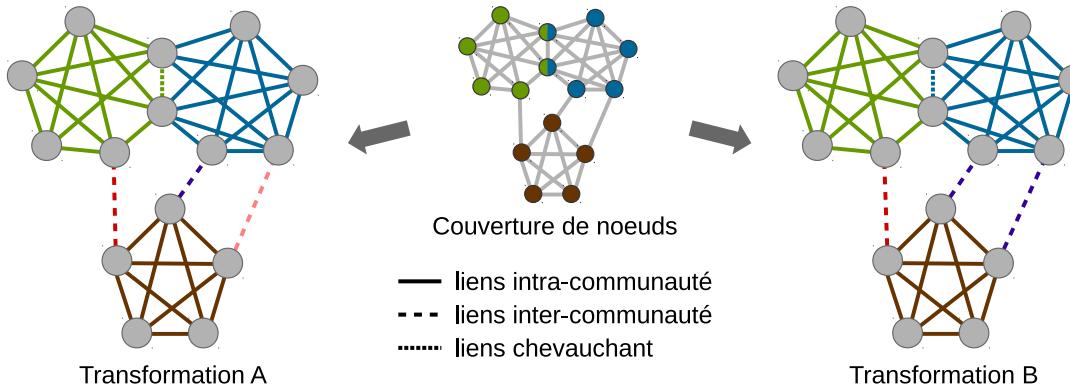


FIGURE 5.10 – Construction de TA et TB depuis une couverture de nœuds. La couleur des liens indique leur groupe.

deux groupes : *travail* et *amis*. Le but est de déterminer le type d'une interaction à partir des groupes des personnes.

Si deux personnes partagent un seul groupe, par exemple *travail*, alors il est clair que l'interaction entre ces deux personnes devraient également être de type *travail*.

Si deux personnes ne partagent aucune communauté, l'une est dans *travail* et l'autre dans *amis*, alors plusieurs choix sont possible pour le type de l'interaction. Soit l'interaction a un type *travail-amis* car on considère que toutes les interactions reliant deux personnes des groupes *travail* et *amis* sont de même types. Soit l'interaction a son propre type car on considère qu'elle est unique.

Enfin, deux personnes peuvent partager plus qu'une communauté si les deux sont dans les communautés *travail* et *amis*. Dans ce cas, l'interaction peut être due à l'un de ces deux groupes indépendamment.

On définit maintenant de manière plus formelle cette transformation qui également illustrée dans la figure 5.10. Soit $u, v \in V$, $C_{u,v}$ désigne l'intersection des communautés de u et v dans la couverture et $U_{u,v}$ désigne leur union. Nous définissons le groupe d'un lien $(u, v) \in E$ dans TA et TB de la manière suivante :

intra-communauté si $|C_{u,v}| = 1$ alors (u, v) est dans la communauté $C_{u,v}$;

inter-communauté si $|C_{u,v}| = 0$ alors dans TA , (u, v) appartient à sa propre communauté.

Dans TB , le lien appartient à la communauté $U_{u,v}$, qui contient l'ensemble des liens (u', v') tel que $U_{u',v'} = U_{u,v}$;

chevauchant si $|C_{u,v}| > 1$ alors le lien (u, v) appartient aléatoirement à une des communautés appartenant à $C_{u,v}$.

Pour générer le graphe, nous avons appliqué un jeu de paramètre classiquement utilisé dans la littérature [For10]. Ainsi, nous avons généré des graphes de 500 nœuds ayant un degré moyen de 25, un degré max de 50 et 10 nœuds appartenant à deux communautés et des communautés ayant une taille comprise entre 20 et 100. Le degré est tiré selon une loi exponentielle de paramètre -2 et la taille des communautés a -1 comme paramètre. Enfin, 90% des liens se sont à l'intérieur d'une communauté et les 10% restant sont répartis de manière aléatoire. Avec ces paramètres, il y a en moyenne 5620 liens intra-communautés, 625 liens inter-communauté et seulement 5 liens chevauchant.

Pour chaque graphe généré, nous testons les partitions TA et TB mais aussi la partition LC trouvée par *link clustering* [ABL10] et les partition EX trouvées par les méthodes de Evans *et al.* [EL09]. Ces algorithmes optimisent respectivement *Partition Density*, *Evans1*, *Evans2* et *Evans3*. Ces partitions sont ensuite évaluées par les fonctions de qualité *Partition Density*,

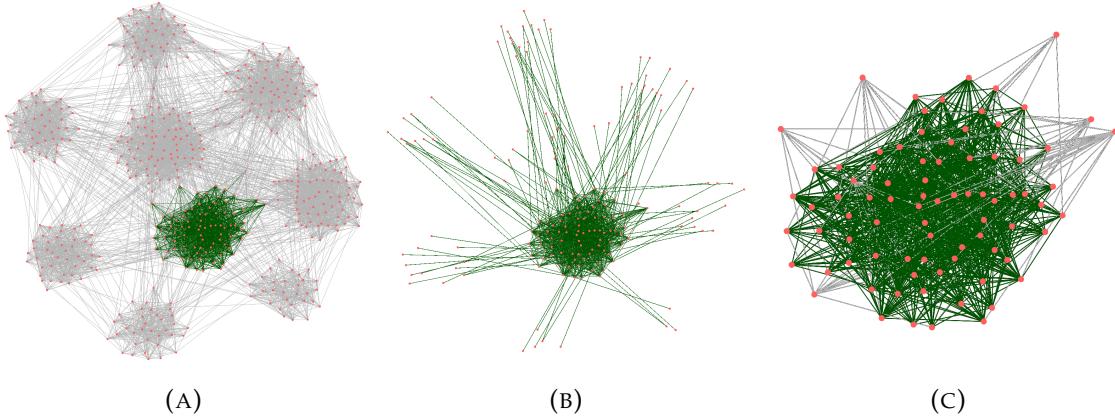


FIGURE 5.11 – Exemple de graphe généré par le LFR en (A) avec une communauté de la vérité de terrain mise en avant en vert. En (B) zoom sur une groupe détecté par $E2$ dont les liens sont en verts. En (C) zoom sur une groupe détecté par LC dont les liens sont en verts.

Evans1, *Evans2*, *Evans3* et *Expected Node*. Une illustration d'un graphe généré et des exemples de groupes capturés par LC et $E2$ sont présentés dans la figure 5.11.

Dans LC , TA , TB et EX , il y a 720, 650, 70 et 11 groupes en moyenne. Afin d'observer la ressemblance de ces partitions, nous avons utilisé la NMI [DDGDA05]. Il apparaît que les partitions TA et TB sont les plus proches. Nous remarquons les trois fonctions de qualité $EvansX$ et les partitions respectives EX sont similaires. Ensuite, nous remarquons que les partition EX diffèrent de TA et de TB uniquement sur les liens inter-communautés. En effet si ils ne sont pas pris en compte lors de la comparaison, alors EX , TA et TB sont équivalentes. Il semble en effet que les liens inter-communautés soient arbitrairement distribués entre les plus grosses communautés adjacentes, ce qui est visible dans la figure 5.11b. Enfin, la partition LC , bien que proche de TA et TB , est un peu plus différente. Ces 720 groupes semblent plus petit mais aussi plus denses que ceux de TA ou TB . En particulier, les liens intra-communautés peuvent être séparés en plusieurs groupes, comme dans la figure 5.11c. Les quatre partitions sont donc différentes et mettent en avant différentes caractéristiques.

Nous procédons maintenant à l'évaluation de ces partitions par les différentes fonctions de qualités. Comme le processus de génération de graphe est aléatoire, les évaluations présentées dans la figure 5.12 représentent 30 générations. On remarque tout d'abord que ni TA ni TB n'a la meilleure évaluation selon $EvansX$ (figures 5.12c à 5.12e) ou *Partition Density* (figure 5.12a) même si TA et TB représentent nos vérités de terrain. Dans le cas de $EvansX$, c'est les partitions EX qui obtiennent les meilleures évaluations. Cela prouve l'efficacité de l'algorithme pour optimiser $EvansX$ mais remet en cause la pertinence des critères EX pour mettre en avant la vérité de terrain. Notre fonction *Expected Nodes* se comporte différemment des 4 autres. Tout d'abord, c'est la vérité de terrain TA qui obtient la meilleure évaluation puis il s'agit de TB ou LC selon les générations. Notre mesure semble donc bien mettre en avant la vérité de terrain générée. De plus, *Expected Nodes* évalue différemment les partitions TA et TB contrairement à *Partition Density* et $EvansX$. C'est un point important car, dans la partition TB , les liens inter-communautés peuvent donner lieu à des groupes non-connexes dans TB , voir figure 5.10. Ce phénomène est très pénalisé par *Expected Nodes*. Enfin selon *Expected Nodes*, les partitions EX ont une mauvaise partition et cela est également dû aux liens inter-communautés. En effet en les fusionnant à une communauté adjacente, cela augmente fortement le nombre de noeuds interne ce qui fait baisser Q_{in} .

Avec ce test, nous mettons en évidence les limites des fonctions de qualités existantes. Les fonctions de qualité $EvansX$ ne peuvent pas permettre de capturer les vérités de terrain générées.

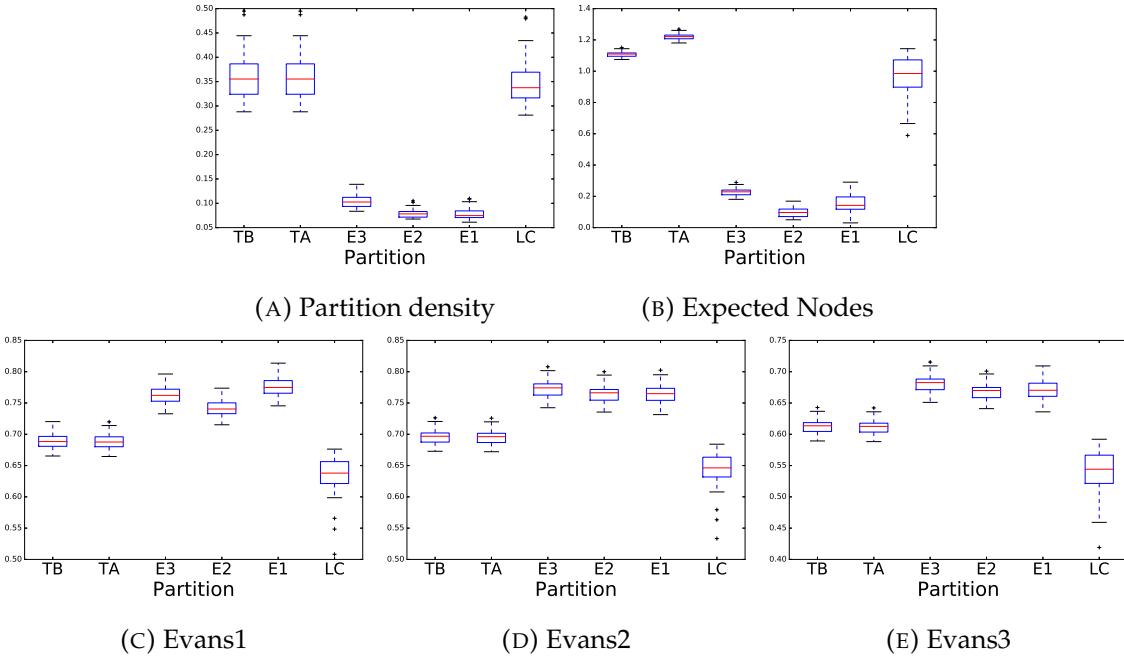


FIGURE 5.12 – Boîte à moustache des évaluations des trois fonctions de qualité pour les différentes partitions de liens. La boîte représente le premier et troisième quartile ainsi que la médiane. Les moustaches s'étendent sur 1.5 fois l'écart interquartile. Les croix sont les points au delà des moustaches.

par LFR. En effet, l'algorithme de Louvain est capable de trouver des partition ayant une évaluation plus élevée que la vérité de terrain. La *Partition Density*, quant à elle, souffre surtout de son incapacité à différencier clairement des partitions différentes. Cela se matérialise par des évaluations très similaires pour les vérités de terrain et la partition *LC*. Il en résulte que la *Partition Density* est sensible aux faibles perturbations. Lors de ces tests, *Expected Nodes* ne semble pas souffrir de ces défauts. Pour ces raisons, nous pensons que *Expected Nodes* est une mesure qui permet de bien évaluer les partitions de liens.

5.4 Calcul et optimisation

Jusqu'à maintenant, nous avons évalué *Expected Nodes* sans nous attacher ni à son calcul ni à son optimisation. Nous discutons maintenant de la complexité de calcul d'*Expected Nodes* pour une partition donnée. Le calcul de la qualité interne 5.5 nécessite d'évaluer la probabilité qu'un nœud soit tiré. Ce calcul de probabilité nécessite d'évaluer pour un nœud u : $1 - \frac{\binom{2|E|-d(u)}{2|L|}}{\binom{2|E|}{2|L|}}$, ce qui peut être assez couteux. Ce calcul correspond à une loi hypergéométrique. Or sous certaines conditions, une loi hypergéométrique peut être approché par une loi binomiale ce qui simplifie le calcul à : $1 - (1 - \frac{|L|}{|E|})^{|L|}$ ce qui est plus rapide. Lors de nos tests, le biais induit par cette approximation reste assez faible, de l'ordre de 0.01% en moyenne. Ce changement de calcul est équivalent à considérer un tirage avec remise au lieu de tirage sans remise. De plus cette probabilité ne dépend que du degré du nœud et du nombre de liens dans le groupe. Donc, tout les nœuds ayant le même degré donnent lieu à la même probabilité. Si l'on considère que l'évaluation de la probabilité pour un nœud peut se faire en $O(1)$ alors, pour un groupe

donné, il est possible de calculer sa qualité en $O(|\{d_G(v)\}_{v \in V}|)$. Cette formulation est très efficace lorsque beaucoup de noeuds ont le même degré. Enfin comme la qualité d'un groupe ne dépend que de sa taille, on peut calculer la qualité d'une partition \mathcal{L} en $O(\{|L_i|\}_{L_i \in \mathcal{L}})$.

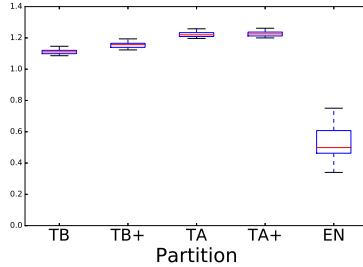
Il est donc assez rapide de calculer Q_{in} . En revanche, la situation est complètement différente pour Q_{ext} . Le processus de calcul est similaire. On peut également appliquer l'approximation de la loi hypergéométrique par une loi binomiale mais deux noeuds de même degré ne vont plus forcément avoir la même probabilité d'être tirés. En effet, Q_{ext} n'utilise pas le graphe initial mais le graphe $G \setminus L_i$. Pour chaque noeud, il faut évaluer son degré dans ce nouveau graphe. Le calcul de Q_{ext} pour un groupe se fait donc en $O(n)$. Pour l'évaluation d'une partition, il n'est pas non plus possible de considérer comme équivalents des groupes de même taille. Le coût pour évaluer une partition est donc en $O(|\mathcal{L}|n)$. Le code pour évaluer une partition de liens d'un graphe avec *Expected Nodes* mais aussi avec *Partition Density*, *Evans1*, *Evans2* et *Evans3* est disponible en ligne : <https://github.com/ksadofr/ExpectedNodes>.

Malgré ce coût élevé, nous avons développé un premier algorithme d'optimisation glouton de *Expected Node*. Le principe de fonctionnement est le suivant. Chaque lien est initialement dans son propre groupe. Puis à chaque itération, on considère deux types de modification de la solution courante. Soit la meilleure fusion de deux communautés soit le meilleur changement de communauté d'un seul lien. On fusionne ou change de communauté un lien si cela améliore la qualité de la partition. Les fusions considérées sont les fusions entre des communautés adjacentes. Les changements de liens se font également que avec une communauté adjacente. On continue de modifier la solution courante tant qu'elle est améliorable par un de ces mouvements. Malheureusement cette approche souffre de deux problèmes majeurs. Tout d'abord le calcul du gain est couteux et rend impossible l'étude de grands jeux de données. Ensuite dans nos tests dans les graphes générés par LFR, il semble que cette méthode reste bloquée sur des optimum locaux bien plus faibles que la vérité de terrain, voir la figure 5.13a où la qualité de la partition trouvée par notre algorithme, noté *EN*, est présentée. Il faudrait donc tester d'autres heuristiques d'optimisation mais aussi travailler sur une méthode de calcul du gain plus optimisée.

Malgré ces limitations, nous avons tout de même tiré parti de cet algorithme naïf afin de tester si une partition donnée peut être améliorée. En effet même si l'algorithme n'est pas adapté pour trouver une partition ayant une qualité élevée en partant de zéro, il peut modifier une partition donnée pour l'améliorer. Nous avons donc utilisé les partitions *TA* et *TB* comme partition de départ de l'algorithme et nous avons observé si l'algorithme est capable d'améliorer les vérités de terrain. Les changements qu'apportent notre algorithme se portent principalement sur les liens inter-communautés. Dans le cas de *TA*, chaque lien inter-communauté constitue une communauté. Or, il se peut que plusieurs liens inter-communautés soient connectés aux mêmes noeuds. Dans ce genre de situation, notre algorithme fusionne ces liens dans une communauté augmentant légèrement la qualité globale. Après optimisation, nous constatons que les partitions *TA* et *TB* peuvent être légèrement améliorées mais qu'elles sont très proches du maximum local lorsque l'on considère notre algorithme d'optimisation, voir la figure 5.13a où les partitions *TA+* et *TB+* sont les versions améliorées de *TA* et *TB*.

5.5 Conclusion

Nous considérons de nouveaux critères pour l'évaluation des partitions de liens tenant compte de la répartition des noeuds internes et externes d'un groupe. À partir de ces critères, nous définissons une mesure de qualité, *Expected Nodes*, basée sur la différence entre le nombre de noeuds induits par un groupe de lien et le nombre de noeuds attendu dans un modèle nul. Ce processus est similaire à celui qui a donné lieu à la modularité. En plus de s'inspirer de la modularité, *Expected Nodes* ne prends pas uniquement en compte la répartition de ses noeuds



(A) Expected nodes

FIGURE 5.13 – Boite à moustache des évaluations selon *Expected Nodes* pour différentes partitions. *TA* et *TB* sont les vérités de terrains, *EN* est la partition trouvée par notre algorithme et enfin *TA+* et *TB+* sont les versions améliorées par notre algorithme de *TA* et *TB*.

induits mais également celle de n œuds et liens adjacents. Ce changement permet d'avoir une évaluation plus fine d'une communauté. Pour montrer la pertinence de cette nouvelle fonction de qualité, nous évaluons quatre fonctions de qualité de la littérature. Nous montrons que, sur nos jeux de tests, *Expected Nodes* semble être la plus à même de capturer la vérité de terrain et de différencier des partitions différentes. Malgré le coût de calcul élevé d'*Expected Nodes*, nous avons implémenté un premier algorithme d'optimisation agglomératif. Cet algorithme n'est, pour l'instant, pas capable d'obtenir des partitions avec des scores élevés. Cependant il nous a permis de vérifier que les vérités de terrain générée par LFR sont des quasi-optimum locaux selon *Expected Nodes*. Bien qu'il soit possible d'améliorer légèrement les vérités de terrains, la structure de l'optimum local est très proche de la partition initiale.

5.5.1 Perspective

Les perspectives autour d'*Expected Nodes* sont nombreuses et portent sur deux points : d'une part le calcul et l'optimisation d'*Expected Nodes* et d'autre part les cas d'applications possibles.

Amélioration de l'optimisation d'*Expected Nodes*

Considérer plus de modifications Il semble que les mouvements envisagés pour modifier une solution courante ne soient pas suffisant pour échapper à des maximums locaux qui sont bien inférieurs à la qualité des vérités de terrain. Il faudrait donc envisager de nouveaux. Inclure la possibilité de diviser une communauté en deux ne semble pas envisageable car il existe trop de nombreuses coupes possibles et le coût de calcul associé est trop coûteux. Il semble, par contre, prometteur de considérer des mouvements associés à un nœud. En effet au lieu de changer un à un les liens d'une communauté, il devrait être possible d'intégrer en un seul mouvement tous les liens adjacents qui sont reliés à un nœud adjacent donné. Pour l'illustrer ce mouvement, reprenons l'exemple d'une clique incluse dans une autre clique plus grande. La solution finale devrait être la clique maximale. Si l'on ne considère que l'ajout de lien un à un³, alors il n'est pas possible d'atteindre la clique maximale. L'algorithme est bloqué car l'ajout d'un unique lien détériore la solution courante. Si on considère l'ajout de tous les liens adjacents à un nœud adjacent donné, alors, après l'ajout, le groupe considéré est toujours une clique et la qualité interne ne diminue pas mais en plus on peut espérer faire diminuer la pénalisation de la qualité externe. De cette manière, il devrait être possible d'éviter des maximums locaux. Plus globalement, les mouvements faisant intervenir plusieurs liens d'un seul coup semblent intéressants.

3. Dans le cas où la fusion de communauté n'est pas une option viable.

Diminuer le coût de calcul du gain Ici, il ne s'agit pas d'améliorer la performance de l'algorithme en terme de qualité obtenue par la partition trouvée par l'algorithme mais de le rendre plus rapide. Dans sa version courante, l'algorithme est trop couteux pour traiter des graphes ayant plus de quelques dizaines de milliers de liens. La force de l'algorithme de Louvain réside dans l'existence d'une expression analytique du gain de fusion de deux communautés qui est rapide à évaluer. Dans notre algorithme, la lenteur est justement due au calcul du gain qui est très coûteux. En effet, nous ne calculons pas directement le gain d'un mouvement mais la différence entre la qualité courante et la qualité résultant du mouvement. C'est une différence importante car, dans notre cas, il faut calculer de zéro la qualité des communautés impactées. Une manière de résoudre ce problème serait de ne pas partir de zéro mais de calculer localement les changements ayant un impact. Prenons le cas de la fusion de communautés A et B . Cela est simple pour la qualité interne car elle ne dépend que du nombre de noeuds et liens internes. Le nouveau nombre de liens, après fusion, est simplement la somme des liens des communautés initiales. Le nouveau nombre de noeuds internes est la taille de l'union des noeuds internes. La situation est plus complexe pour la qualité externe. Le nouveau nombre de liens adjacents n'est pas simplement une somme. De même, l'identité et le degré des nouveaux noeuds adjacents dans $G \setminus (A \cup B)$ ne sont pas triviales à calculer. Il faudrait des parcours locaux sur $(A \cup B)$ et son voisinage pour arriver à mettre à jour ces données nécessaires au calcul de Q_{ext} . Si ce parcours est fait suffisamment soigneusement, il est sûrement possible de gagner en terme de vitesse d'exécution.

Changer la stratégie d'optimisation Une autre source potentielle de gain en vitesse de l'algorithme est la stratégie d'optimisation. Lors d'une itération, nous appliquons la meilleure fusion de communauté ou le meilleur changement de communauté d'un lien. Appliquer le meilleur mouvement peut améliorer la convergence de l'algorithme vers l'optimum local mais cela est coûteux car lors d'une itération tous les gains doivent être calculés. C'est problématique car c'est justement le calcul du gain qui est très coûteux. Une solution face à ce problème est d'utiliser une variante stochastique de l'algorithme. Dans cette variante, ce n'est pas le meilleur changement qui est appliqué mais le premier qui mène à une amélioration de la qualité. Pour ce faire, il est nécessaire de considérer un ordre aléatoire des mouvements possibles et d'appliquer le premier dont le gain est positif puis de recommencer. Avec ce changement, on teste beaucoup moins de mouvements avant d'en appliquer un. Cependant, il est possible que plus de mouvements soient nécessaires avant d'arriver à un optimum local. Par ailleurs avec ce changement, l'algorithme n'est plus déterministe car il dépend de l'ordre d'évaluation des mouvements. Il faudrait donc être très prudent vis-à-vis de ses aspects, lors du test de cette technique.

Cas d'utilisation d'*Expected Nodes*

Nous avons parlé de moyens possibles pour trouver une méthode d'optimisation efficace d'*Expected Nodes*. Il est également nécessaire de revenir sur l'utilisation d'*Expected Nodes*. Nous avons testé *Expected Nodes* sur un algorithme largement utilisé dans la littérature et les résultats sont encourageants. Il y a cependant une différence majeure dans la manière dont nous avons testé *Expected Nodes* vis-à-vis des autres fonctions de qualités. Il n'existe pas, pour l'instant, d'algorithme efficace pour trouver une partition maximisant *Expected Nodes*. C'est pourquoi nous testons des partitions trouvées empiriquement par d'autres moyens. Cependant, il existe peut-être une partition que nous n'avons pas testé qui aurait une bien meilleure qualité que les vérités de terrains. L'amélioration de l'algorithme d'optimisation et le test d'autres partitions accroiraient la pertinence des partitions testées et ainsi le bien fondé d'*Expected Nodes*. De manière analogue, il serait intéressant d'évaluer *Expected Nodes* sur des jeux de données ayant une réelle vérité de terrain sur les liens, que ce soit des données réelles ou simulées. Plus globalement, il

serait intéressant de comprendre ce qu'il est possible de capturer et représenter comme structure avec une partition de liens.

Nous avons parlé jusque maintenant de l'application d'*Expected Nodes* sur des graphes non pondérés et non dirigés. La question de la pondération est délicate à prendre en compte pour *Expected Nodes*. En effet, notre méthode repose sur des tirages aléatoires de demi-liens. La notion de tirage est intrinsèquement lié à des valeurs entières. En effet, il n'existe pas à ma connaissance de pendant continu à la loi hypergéométrique ; on ne tire pas aléatoirement 0.6 demi-liens parmi 14.3 demi liens possible. L'application d'*Expected Nodes* à un graphe pondéré semble donc compromise. Cependant, il devrait être trivialement possible de considérer des multigraphes. Dans les multigraphes, deux nœuds peuvent être relié par plus qu'un lien. Les notions de tirages aléatoires peuvent s'adapter à ce genre graphe assez facilement. Il y a toute fois une différence majeur lorsque l'on considère les multigraphes par rapport aux graphes classiques : les contraintes sur la partition résultat. Dans le cas des multigraphes, il serait sûrement intéressant d'ajouter la contrainte que chaque communauté ne doit contenir qu'une seule occurrence d'un lien donné. Ainsi s'il existe deux liens entre u et v , alors ces deux liens devront être dans deux communautés différentes.

Chapitre 6

Vers des fonctions de qualité dans les flots de liens

Sommaire

6.1 Définition	79
6.2 Générateur de flots de liens avec structure communautaire	79

Génération de : [GDA⁺15, KPV14, PGPSV12] snapshot generateur [SBPS13, VGB14] [MSPS15] [SBB10]

6.1 Définition

6.2 Générateur de flots de liens avec structure communautaire

Chapitre 7

Conclusion

Annexe A

Détection de groupes pertinents dans les flots de liens

A.1 Rollernet dataset

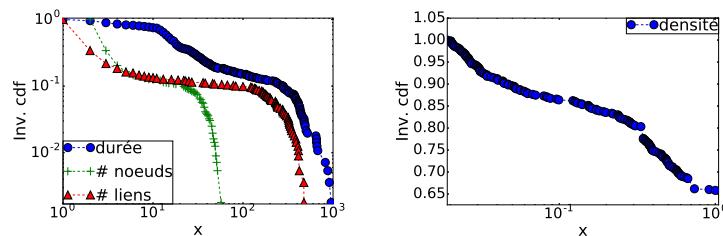


FIGURE A.1 – Distribution cumulative inverse du nombre de liens, de noeuds et de la durée en (A) et de la densité en (B) pour les candidats trouvés par la méthode de Louvain dans le jeu de données Rollernet.

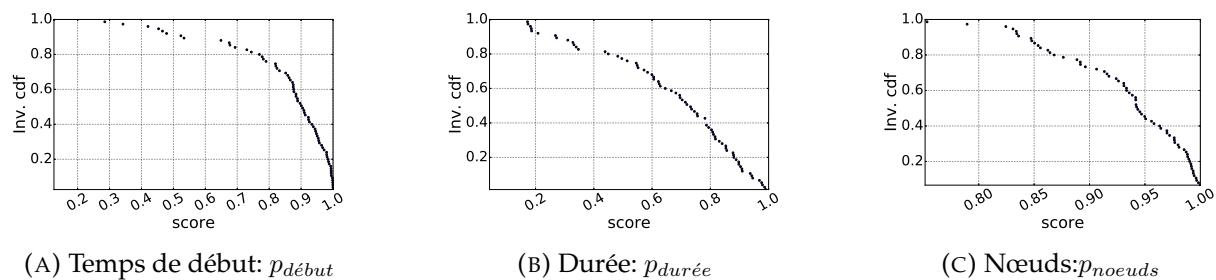


FIGURE A.2 – Distribution cumulative inverse des scores pour chaque voisinage dans le jeu de données Rollernet.

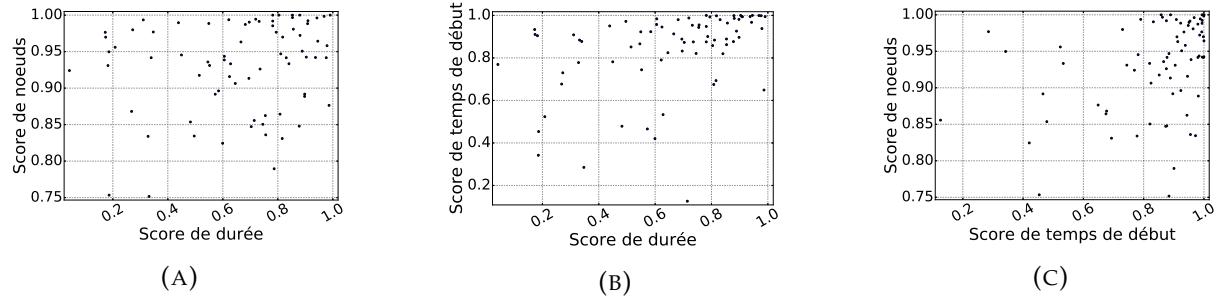


FIGURE A.3 – Corrélations des scores selon le voisinage dans le jeu de données Rollernet.

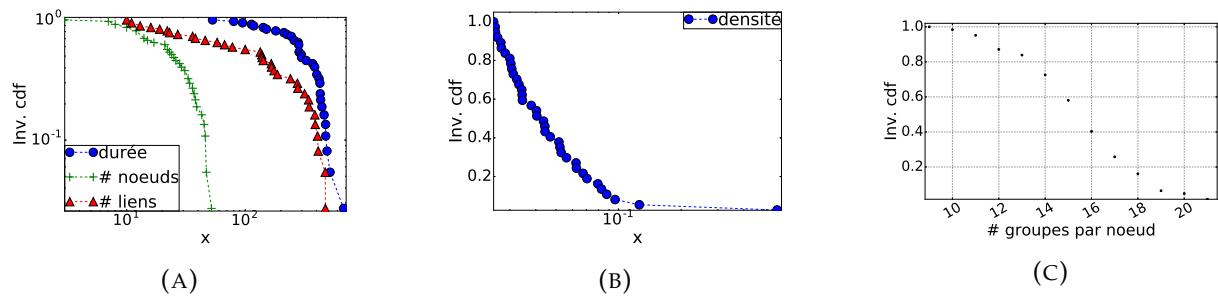


FIGURE A.4 – Distributions cumulatives inverses du nombre de liens, de noeuds et de la durée en (A), de la densité en (B), du nombre de groupes par nœud en (C) pour les groupes capturés par notre méthode dans les données Rollernet.

A.2 Baboon dataset

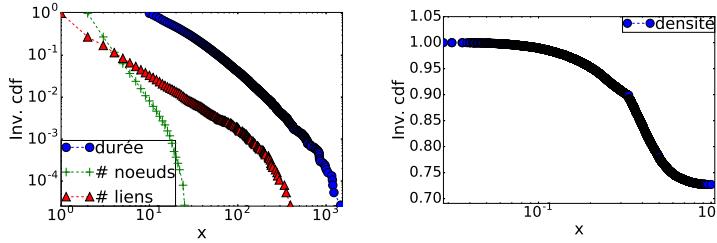


FIGURE A.5 – Distribution cumulative inverse du nombre de liens, de noeuds et de la durée en (A) et de la densité en (B) pour les candidats trouvés par la méthode de Louvain dans le jeu de données Babouins.

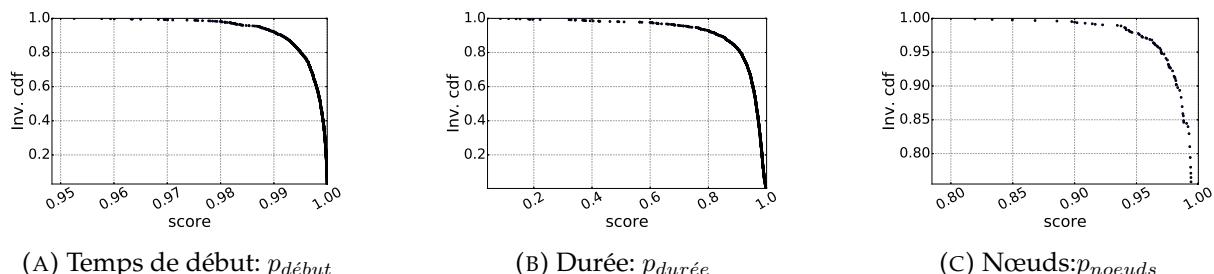


FIGURE A.6 – Distribution cumulative inverse des scores pour chaque voisinage dans le jeu de données Babouins.

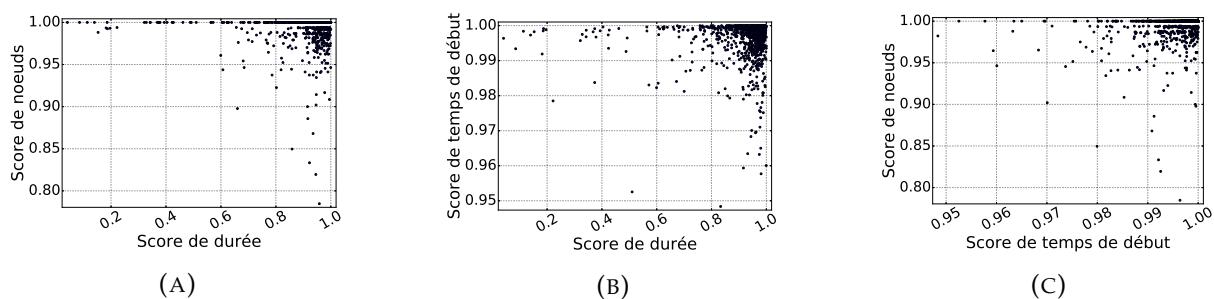


FIGURE A.7 – Corrélations des scores selon le voisinage dans le jeu de données Babouins.

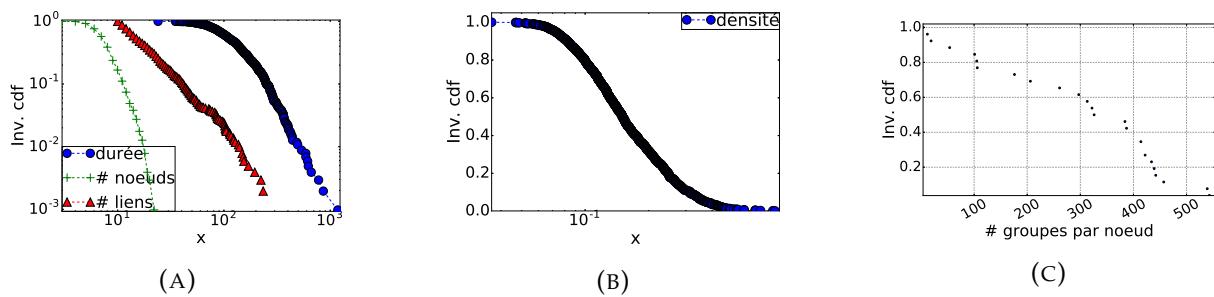


FIGURE A.8 – Distributions cumulatives inverses du nombre de liens, de noeuds et de la durée en (A), de la densité en (B), du nombre de groupes par noeud en (C) pour les groupes capturés par notre méthode dans les données Babouins.

A.3 Reality mining dataset

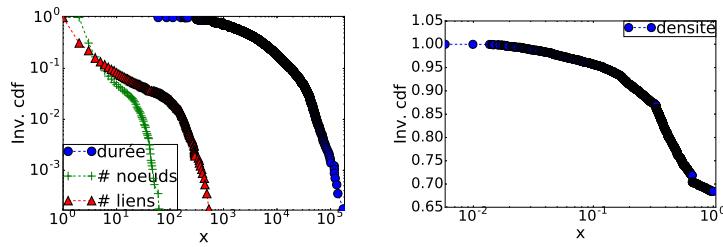


FIGURE A.9 – Distribution cumulative inverse du nombre de liens, de noeuds et de la durée en (A) et de la densité en (B) pour les candidats trouvés par la méthode de Louvain dans le jeu de données Reality Mining.

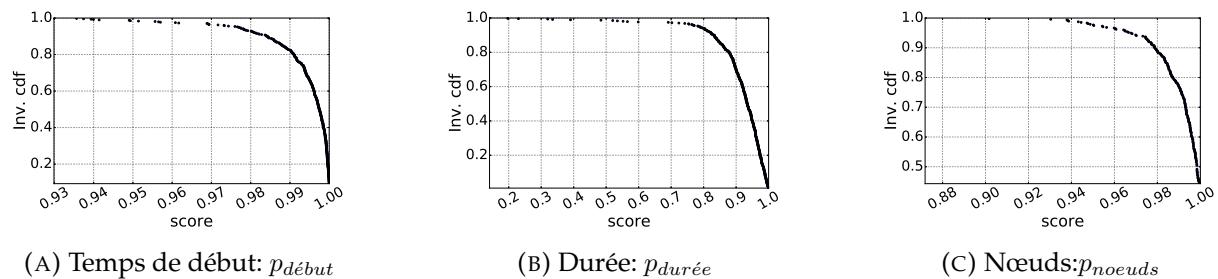


FIGURE A.10 – Distribution cumulative inverse des scores pour chaque voisinage dans le jeu de données Reality Mining.

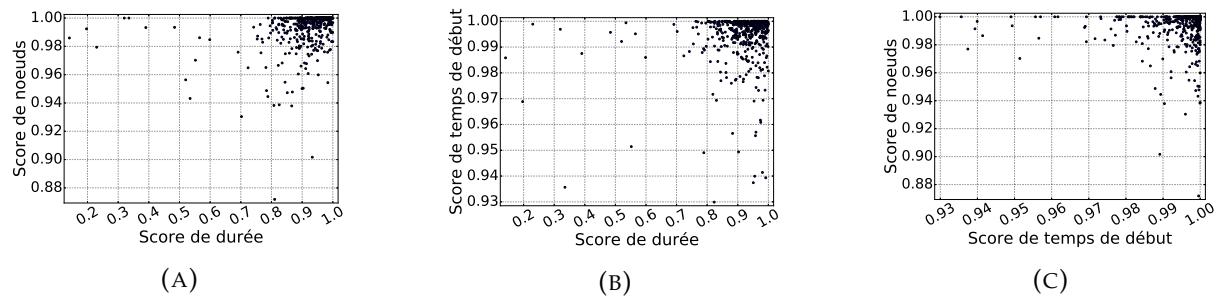


FIGURE A.11 – Corrélations des scores selon le voisinage dans le jeu de données Reality Mining.

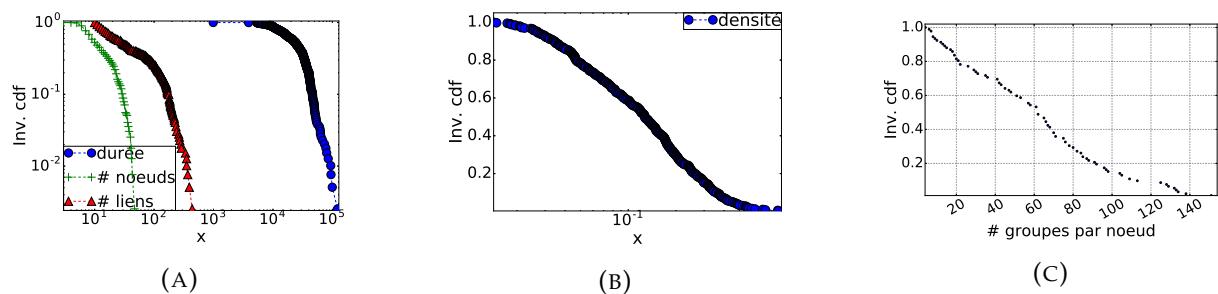


FIGURE A.12 – Distributions cumulatives inverses du nombre de liens, de noeuds et de la durée en (A), de la densité en (B), du nombre de groupes par noeud en (C) pour les groupes capturés par notre méthode dans les données Reality Mining.

Bibliographie

- [ABFX08] Edoardo Airoldi, David Blei, Stephen Fienberg, and Eric Xing. Mixed Membership Stochastic Blockmodels. *J. Mach. Learn. Res.*, 9 :1981 – 2014, 2008.
- [ABL10] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307) :761–764, aug 2010.
- [AG10] Thomas Aynaud and Jean-Loup Guillaume. Static community detection algorithms for evolving networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 508–514, 2010.
- [AGL14] Alice Albano, Jean Loup Guillaume, and Benedicte Le Grand. On the use of intrinsic time scale for dynamic community detection and visualization in social networks. In *Proceedings - International Conference on Research Challenges in Information Science*, 2014.
- [AM11] Rodrigo Aldecoa and Ignacio Marín. Deciphering Network Community Structure by Surprise. *PLoS ONE*, 6(9) :e24195, sep 2011.
- [aMGH11] a.F. McDaid, Derek Greene, and Neil Hurley. Normalized Mutual Information to evaluate overlapping community finding algorithms. *Arxiv preprint arXiv:1110.2515*, pages 1–3, 2011.
- [APU09] Sitaram Asur, Srinivasan Parthasarathy, and Duygu Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs, 2009.
- [BBC⁺14] S. Boccaletti, G. Bianconi, R. Criado, C.I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1) :1–122, 2014.
- [BBC⁺15] Oana Denisa Balalau, Francesco Bonchi, T-H. Hubert Chan, Francesco Gullo, and Mauro Sozio. Finding Subgraphs with Maximum Total Density and Limited Overlap. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*, pages 379–388, New York, New York, USA, feb 2015. ACM Press.
- [BC78] Edward A Bender and E.Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3) :296–307, may 1978.
- [BCS15] Sanghamitra Bandyopadhyay, Garisha Chowdhary, and Debarka Sengupta. FOCS : Fast Overlapped Community Search. *IEEE Transactions on Knowledge and Data Engineering*, PP(99) :1–1, 2015.
- [BDG⁺07] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On Finding Graph Clusterings with Maximum Modularity. In *Graph-Theoretic Concepts in Computer Science*, volume 4769, pages 121–132. 2007.
- [BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, oct 2008.

- [BKN11] Brian Ball, Brian Karrer, and M. E. J. Newman. Efficient and principled method for detecting communities in networks. *Physical Review E*, 84(3) :036103, sep 2011.
- [BMS11] Petko Bogdanov, Misael Mongiovì, and Ambuj K. Singh. Mining Heavy Subgraphs in Time-Evolving Networks. In *2011 IEEE 11th International Conference on Data Mining*, pages 81–90. IEEE, dec 2011.
- [BP15] Vladimir Batagelj and Selena Praprotnik. An algebraic approach to temporal network analysis based on temporal quantities. may 2015.
- [BPW⁺13] Danielle S. Bassett, Mason A. Porter, Nicholas F. Wymbs, Scott T. Grafton, Jean M. Carlson, and Peter J. Mucha. Robust Detection of Dynamic Community Structure in Networks. *Chaos : An Interdisciplinary Journal of Nonlinear Science*, 23(1) :013142, 2013.
- [BPW⁺16] Marya Bazzi, Mason a. Porter, Stacy Williams, Mark McDonald, Daniel J. Fenn, and Sam D. Howison. Community detection in temporal multilayer networks, and its application to correlation networks. *Multiscale Modeling & Simulation*, 14(1) :1–41, 2016.
- [Bre74] Breiger, Ronald. The Duality of Persons and Groups. *Social Forces*, 53(2) :181–190, 1974.
- [CA14] Rémy Cazabet and Frédéric Amblard. Dynamic Community Detection. In *Encyclopedia of Social Network Analysis and Mining*, pages 404–414. Springer New York, New York, NY, 2014.
- [CAH10] R. Cazabet, F. Amblard, and C. Hanachi. Detection of Overlapping Communities in Dynamical Social Networks. *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 2010.
- [CBW13] RajmondaSulo Caceres and Tanya Berger-Wolf. Temporal Scale of Dynamic Networks. In Petter Holme and Jari Saramäki, editors, *Temporal Networks, Understanding Complex Systems*, pages 65–94. Springer Berlin Heidelberg, 2013.
- [CFQS11] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6811 LNCS, pages 346–359, 2011.
- [CGP11] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5) :512–546, oct 2011.
- [CKT06] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, page 554, New York, New York, USA, aug 2006. ACM Press.
- [CKU13] Yudong Chen, Vikas Kawadia, and Rahul Urgaonkar. Detecting Overlapping Temporal Community Structure in Time-Evolving Networks. *arXiv preprint arXiv:1303.7226*, page 12, mar 2013.
- [CKW15] Margaret C. Crofoot, Roland W. Kays, and Martin Wikelski. Data from : Shared decision-making drives collective movement in wild baboons, 2015.
- [CLR16] Marco Corneli, Pierre Latouche, and Fabrice Rossi. Exact ICL maximization in a non-stationary temporal extension of the stochastic block model for dynamic networks. *Neurocomputing*, mar 2016.

- [CSG16] Mário Cordeiro, Rui Portocarrero Sarmento, and João Gama. Dynamic community detection in evolving networks using locality modularity optimization. *Social Network Analysis and Mining*, 6(1) :15, mar 2016.
- [DDDGA05] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 2005(09) :10, 2005.
- [DDG⁺15] Armando Di Nardo, Michele Di Natale, Carlo Giudicianni, Dino Musmarra, Giovanni Francesco Santonastaso, and Antonietta Simone. Water Distribution System Clustering and Partitioning Based on Social Network Algorithms. *Procedia Engineering*, 119 :196–205, 2015.
- [DDGDA05] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 2005(09) :P09008–P09008, sep 2005.
- [DGG12] Maximilien Danisch, Jean-Loup Guillaume, and Bénédicte Le Grand. Towards multi-ego-centred communities : a node similarity approach. *International Journal of Web Based Communities*, mar 2012.
- [DLAR14] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in social systems. *arXiv preprint*, aug 2014.
- [DLCA07] Remi Dorat, Matthieu Latapy, Bernard Conein, and Nicolas Auray. Multi-level analysis of an interaction network between individuals in a mailing-list. *Ann Telecommun*, 62 :325–349, 2007.
- [DM04] Luca Donetti and Miguel A. Muñoz. Detecting network communities : a new systematic and efficient algorithm. *Journal of Statistical Mechanics : Theory and Experiment*, 10(2004) :8, 2004.
- [dRSKvdH14] Marcel A de Reus, Victor M Saenger, René S Kahn, and Martijn P van den Heuvel. An edge-centric perspective on the human connectome : link communities in the brain. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 369(1653) :20130527, oct 2014.
- [DSRC⁺13] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A. Porter, Sergio Gómez, and Alex Arenas. Mathematical Formulation of Multilayer Networks. *Physical Review X*, 3(4) :041022, dec 2013.
- [DVR16] Simon De Ridder, Benjamin Vandermarkiere, and Jan Ryckebusch. Detection and localization of change points in temporal networks with the aid of stochastic block models. *arXiv preprint arXiv* :1602.00661, 2016.
- [DYB10] J-C Delvenne, S N Yaliraki, and M Barahona. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences of the United States of America*, 107(29) :12755–60, jul 2010.
- [EL09] T. S. Evans and Renaud Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80(1) :016105, jul 2009.
- [ELS15] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient Densest Subgraph Computation in Evolving Graphs. In *Proceedings of the 24th International Conference on World Wide Web*, pages 300–310. International World Wide Web Conferences Steering Committee, may 2015.
- [EPL09] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring Social Network Structure using Mobile Phone Data. *Pnas*, 106(usually 1) :15274–15278, 2009.

- [ER59] P Erdős and a Rényi. On random graphs. *Publicationes Mathematicae*, 6 :290–297, 1959.
- [ER11] Alcides Viamontes Esquivel and Martin Rosvall. Compression of Flow Can Reveal Overlapping-Module Organization in Networks. *Physical Review X*, 1 :1–11, 2011.
- [FB07] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1) :36–41, 2007.
- [FB14] Julie Fournet and Alain Barrat. Contact patterns among high school students. *PloS one*, 9(9) :e107878, jan 2014.
- [FCF11] Adrien Friggeri, Guillaume Chelius, and Eric Fleury. Triangles to capture social cohesion. In *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*, pages 257–265, 2011.
- [For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5) :75–174, feb 2010.
- [FW15] Damien R Farine and Hal Whitehead. Constructing, conducting, and interpreting animal social network analysis. *The Journal of animal ecology*, jul 2015.
- [GB13] 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, sep 2013.
- [GDA⁺15] Clara Granell, Richard K. Darst, Alex Arenas, Santo Fortunato, and Sergio Gómez. A benchmark model to assess community structure in evolving networks. page 11, jan 2015.
- [GdMC10] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4) :046106, apr 2010.
- [GPBC15] Laetitia Gauvin, André Panisson, Alain Barrat, and Ciro Cattuto. Revealing latent factors of temporal networks for mesoscale intervention in epidemic spread. jan 2015.
- [GPC14] Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks : a non-negative tensor factorization approach. *PloS one*, 9(1) :e86028, jan 2014.
- [Gre10] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12, 2010.
- [HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1) :193–218, 1985.
- [HBG14] S Harenberg, G Bello, and L Gjeltema. Community detection in large-scale networks : a survey and empirical evaluation. *Wiley*, 2014.
- [HBP15] De-Shuang Huang, Vitoantonio Bevilacqua, and Prashan Premaratne, editors. *Intelligent Computing Theories and Methodologies*, volume 9225 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2015.
- [HDF14] Darko Hric, Richard K. Darst, and Santo Fortunato. Community detection in networks : Structural communities versus ground truth. *Physical Review E*, 90(6) :062805, 2014.

- [HK14] Dávid X Horváth and János Kertész. Spreading dynamics on networks : the role of burstiness, topology and non-stationarity. *New Journal of Physics*, 16(7) :073037, jul 2014.
- [HKKS04] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101 Suppl :5249–5253, apr 2004.
- [HKW14] Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. Clustering evolving networks. *arXiv preprint arXiv:1401.3516*, 2014.
- [HL14] Petter Holme and Fredrik Liljeros. Birth and death of links control disease spreading in empirical contact networks. *Scientific reports*, 4 :4999, jan 2014.
- [HLL83] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels : First steps. *Social Networks*, 5(2) :109–137, 1983.
- [Hol15a] Petter Holme. Information content of contact-pattern representations and predictability of epidemic outbreaks. mar 2015.
- [Hol15b] Petter Holme. Modern temporal network theory : a colloquium. *Eur. Phys. J. B*, 88(9) :234, 2015.
- [HS13] Petter Holme and Jari Saramäki, editors. *Temporal Networks*. Understanding Complex Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [HWW⁺13] Lan Huang, Guishen Wang, Yan Wang, Enrico Blanzieri, and Chao Su. Link Clustering with Extended Link Similarity and EQ Evaluation Division. *PLoS ONE*, 8(6) :e66005, jun 2013.
- [JBP⁺15] 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*, 91(1) :012821, jan 2015.
- [JPKK14] Hang-Hyun Jo, Juan I. Perotti, Kimmo Kaski, and János Kertész. Analytically Solvable Model of Spreading Dynamics with Non-Poissonian Processes. *Physical Review X*, 4(1) :011041, mar 2014.
- [KAB⁺14] Mikko Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3) :203–271, jul 2014.
- [Kan14] Rushed Kanawati. Seed-centric approaches for community detection in complex networks. In Gabriele Meiselwitz, editor, *6th international conference on Social Computing and Social Media*, volume LNCS 8531, pages 197–208. Springer International Publishing, 2014.
- [KBV15] J. Kalavathi, S. Balamurali, and M. Venkatesulu. An Efficient Evolutionary Approach for Identifying Evolving Groups in Dynamic Social Networks Using Genetic Modeling. *Procedia Computer Science*, 57 :428–437, 2015.
- [Kim14] Sungmin Kim. Community Detection in Directed Networks and its Application to Analysis of Social Networks. 2014.
- [KJ11] Youngdo Kim and Hawoong Jeong. Map equation for link communities. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 84(2) :026110, aug 2011.
- [KKB⁺12] Gautier Krings, Márton Karsai, Sebastian Bernhardsson, Vincent D Blondel, and Jari Saramäki. Effects of time window size and placement on the structure of an aggregated communication network. *EPJ Data Science*, 1(1) :4, may 2012.
- [KKBK12] Márton Karsai, Kimmo Kaski, Albert-László Barabási, and János Kertész. Universal features of correlated bursty behaviour. *Scientific reports*, 2 :397, jan 2012.

- [KKK⁺11] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2011(11) :P11005, nov 2011.
- [KKKS08] Jussi M Kumpula, Mikko Kivelä, Kimmo Kaski, and Jari Saramäki. Sequential algorithm for fast clique percolation. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 78 :026109, 2008.
- [KKKS13] Lauri Kovanen, Kimmo Kaski, János Kertész, and Jari Saramäki. Temporal motifs reveal homophily, gender-specific patterns and group talk in mobile communication networks. page 8, feb 2013.
- [KKP⁺11] Márton Karsai, Mikko Kivelä, R. K. Pan, K. Kaski, J. Kertész, A.-L. Barabási, and Jari Saramäki. Small but slow world : How network topology and burstiness slow down spreading. *Physical Review E*, 83(2) :025102, feb 2011.
- [KN11] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1) :016107, jan 2011.
- [KPV14] Márton Karsai, Nicola Perra, and Alessandro Vespignani. Time varying networks and the weakness of strong ties. *Scientific reports*, 4 :4001, jan 2014.
- [KR15] Tatsuro Kawamoto and Martin Rosvall. Estimating the resolution limit of the map equation in community detection. *Physical Review E*, 91(1) :012809, jan 2015.
- [Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565, jul 1978.
- [LB62] C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1) :45–64, 1962.
- [LCZ⁺08] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet : A Framework for Analyzing Communities and Their Evolutions in Dynamic Networks. In *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 685, New York, New York, USA, apr 2008. ACM Press.
- [LF09a] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80 :016118, 2009.
- [LF09b] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms : a comparative analysis. Technical report, aug 2009.
- [LF11] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6) :066122, dec 2011.
- [LHB⁺12] Jingyong Li, Lan Huang, Tian Bai, Zhe Wang, and Hongsheng Chen. CDBIA : A dynamic community detection method based on incremental analysis. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 2224–2228. IEEE, may 2012.
- [LLDM08] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceeding of the 17th international conference on World Wide Web - WWW '08*, page 695, New York, New York, USA, apr 2008. ACM Press.
- [LLJT14] Shenghong Li, Hao Lou, Wen Jiang, and Junhua Tang. Detecting Community Structure via Synchronous Label Propagation. *Neurocomputing*, nov 2014.
- [LRK⁺14] Sungsu Lim, Seungwoo Ryu, Sejeong Kwon, Kyomin Jung, and Jae-Gil Lee. LinkSCAN* : Overlapping community detection using the link-space transformation. In *2014 IEEE 30th International Conference on Data Engineering*, pages 292–303. IEEE, mar 2014.

- [LRRF11] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. *PLoS one*, 6(4) :e18961, jan 2011.
- [LS99] D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755) :788–91, 1999.
- [LWP08] Feng Luo, James Wang, and Eric Promislow. Exploring Local Community Structures in Large Networks. *2006 IEEE/WIC/ACM International Conference on Web Intelligence WI 2006 Main Conference Proceedings WI06*, 6(4) :387–400, 2008.
- [LZW⁺13] Zhenping Li, Xiang-Sun Zhang, Rui-Sheng Wang, Hongwei Liu, and Shihua Zhang. Discovering link communities in complex networks by an integer programming model and a genetic algorithm. *PLoS one*, 8 :e83739, 2013.
- [MM15] Catherine Matias and Vincent Miele. Statistical clustering of temporal networks through a dynamic stochastic block model. *arXiv preprint arXiv :1506.07464*, jun 2015.
- [MRM⁺10] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. *Science*, 328(5980) :876–878, may 2010.
- [MRV15] C Matias, T Rebafka, and F Villers. Estimation and clustering in a semiparametric Poisson process stochastic block model for longitudinal networks. *arXiv preprint arXiv :1512.07075*, 2015.
- [MSCA09] R. D. Malmgren, D. B. Stouffer, A. S. L. O. Campanharo, and L. A. N. Amaral. On Universality in Human Correspondence Activity. *Science*, 325(5948) :1696–1700, sep 2009.
- [MSMA08] R Dean Malmgren, Daniel B Stouffer, Adilson E Motter, and Luís A N Amaral. A Poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences of the United States of America*, 105(47) :18153–8, nov 2008.
- [MSPS15] Antoine Moinet, Michele Starnini, and Romualdo Pastor-Satorras. Burstiness and Aging in Social Temporal Networks. *Physical Review Letters*, 114(10) :108701, mar 2015.
- [MT09] Marija Mitrović and Bosiljka Tadić. Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 80(2), 2009.
- [MT15] Clémence Magnien and Fabien Tarissan. Time Evolution of the Importance of Nodes in dynamic Networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 - ASONAM '15*, pages 1200–1207, New York, New York, USA, 2015. ACM Press.
- [MTR12] Bivas Mitra, Lionel Tabourier, and Camille Roth. Intrinsically dynamic network communities. *Computer Networks*, 56(3) :1041–1053, feb 2012.
- [MV13] FD Malliaros and M Vazirgiannis. Clustering and community detection in directed networks : A survey. *Physics Reports*, 2013.
- [New09] M. E. J. Newman. Random Graphs with Clustering. *Physical Review Letters*, 103(5) :058701, jul 2009.
- [New16] MEJ Newman. Community detection in networks : Modularity optimization and maximum likelihood are equivalent. *arXiv preprint arXiv :1606.02319*, 2016.
- [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, 2004.

- [NMCM09] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics : Theory and Experiment*, 2009(03) :P03024, mar 2009.
- [NS01] Krzysztof Nowicki and Tom a. B Snijders. Estimation and Prediction for Stochastic Blockstructures. *Journal of the American Statistical Association*, 96(455) :1077–1087, 2001.
- [PBV07] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136) :664–667, 2007.
- [PC13] M Planté and M Crampes. Survey on social community detection. *Social Media Retrieval*, 2013.
- [PC15] Leto Peel and Aaron Clauset. Detecting change points in the large-scale structure of evolving networks. In *29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1–11, 2015.
- [PDFV05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043) :814–818, jun 2005.
- [Pei15] Tiago P. Peixoto. Inferring the mesoscale structure of layered, edge-valued, and time-varying networks. *Physical Review E*, 92(4) :042807, oct 2015.
- [PGPSV12] Nicola Perra, B Gonçalves, R Pastor-Satorras, and A Vespignani. Activity driven modeling of time varying networks. *Scientific reports*, 2 :469, jan 2012.
- [PHK11] Daniel Cosmin Porumbel, Jin Kao Hao, and Pascale Kuntz. An efficient algorithm for computing the distance between close partitions. *Discrete Applied Mathematics*, 159(1) :53–59, jan 2011.
- [PJHS14] Juan Ignacio Perotti, Hang-Hyun Jo, Petter Holme, and Jari Saramäki. Temporal network sparsity and the slowing down of spreading. nov 2014.
- [PL05] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Computer and Information Sciences (ISCIS)*, 10 :284–293, 2005.
- [RAK07] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 76, 2007.
- [Ran71] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336) :846–850, 1971.
- [RB06] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 74(1), 2006.
- [RB08] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(4) :1118–23, jan 2008.
- [RB10] Martin Rosvall and Carl T Bergstrom. Mapping change in large networks. *PloS one*, 5(1) :e8694, jan 2010.
- [RB13] Luis Enrique Correa Rocha and Vincent D Blondel. Bursts of vertex activation and epidemics in evolving networks. *PLoS computational biology*, 9(3) :e1002974, jan 2013.
- [RPB13] Bruno Ribeiro, Nicola Perra, and Andrea Baronchelli. Quantifying the effect of temporal resolution on time-varying networks. *Scientific reports*, 3 :3006, jan 2013.

- [RTG14] Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. Discovering Dynamic Communities in Interaction Networks. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8725 of *Lecture Notes in Computer Science*, pages 678–693. Springer Berlin Heidelberg, 2014.
- [SBB10] Juliette Stehlé, Alain Barrat, and Ginestra Bianconi. Dynamical and bursty interactions in social networks. *Physical Review E*, 81(3) :035101, mar 2010.
- [SBBPS12] Michele Starnini, Andrea Baronchelli, Alain Barrat, and Romualdo Pastor-Satorras. Random walks on temporal networks. *Physical Review E*, 85(5) :056115, may 2012.
- [SBPS13] Michele Starnini, Andrea Baronchelli, and Romualdo Pastor-Satorras. Modeling Human Dynamics of Face-to-Face Interaction Networks. *Physical Review Letters*, 110(16) :168701, apr 2013.
- [SCCH09] Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. Detect overlapping and hierarchical community structure in networks. *Physica A : Statistical Mechanics and its Applications*, 388(8) :1706–1712, apr 2009.
- [SCF⁺13] Chuan Shi, Yanan Cai, Di Fu, Yuxiao Dong, and Bin Wu. A link clustering based overlapping community detection algorithm. In *Data and Knowledge Engineering*, volume 87, pages 394–404, 2013.
- [SFY07] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. GraphScope. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 687, New York, New York, USA, aug 2007. ACM Press.
- [SHZ⁺14] Heli Sun, Jianbin Huang, Xin Zhang, Jiao Liu, Dong Wang, Huailiang Liu, Jianhua Zou, and Qinbao Song. IncOrder : Incremental density-based community detection in dynamic networks. *Knowledge-Based Systems*, oct 2014.
- [SLX⁺14] J Shang, L Liu, F Xie, Z Chen, J Miao, and X Fang. A real-time detecting algorithm for tracking community structure of dynamic networks. *arXiv preprint arXiv* :, 2014.
- [SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8) :888–905, 2000.
- [SPFCC15] A. Strandburg-Peshkin, D. R. Farine, I. D. Couzin, and M. C. Crofoot. Shared decision-making drives collective movement in wild baboons. *Science*, 348(6241) :1358–1361, jun 2015.
- [SSA06] Sulayman Sowe, Ioannis Stamelos, and Lefteris Angelis. Identifying knowledge brokers that yield software engineering knowledge in OSS projects. *Information and Software Technology*, 48(11) :1025–1033, 2006.
- [SSTM15] Natalie Stanley, Saray Shai, Dane Taylor, and Peter J. Mucha. Clustering Network Layers With the Strata Multilayer Stochastic Block Model. jul 2015.
- [STM15] Leo Speidel, Taro Takaguchi, and Naoki Masuda. Community detection in directed acyclic graphs. *The European Physical Journal B*, 88(8) :203, aug 2015.
- [SVB⁺11] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean François Pinton, Marco Quaggiotto, Wouter van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhemps. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8), 2011.
- [SWP⁺14] Ingo Scholtes, Nicolas Wider, René Pfitzner, Antonios Garas, Claudio J Tessone, and Frank Schweitzer. Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks. *Nature communications*, 5 :5024, jan 2014.

- [TAD15] V. A. Traag, R. Aldecoa, and J. C. Delvenne. Detecting communities using asymptotical surprise. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 92(2), 2015.
- [TLB⁺09] P.-U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Dias de Amorim, and J. Whitbeck. The Accordion Phenomenon : Analysis, Characterization, and Impact on DTN Routing. In *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, pages 1116–1124. IEEE, apr 2009.
- [Tra15] V. A. Traag. Faster unfolding of communities : speeding up the Louvain algorithm. mar 2015.
- [VBB08] A Vespiagnani, M Bathelemy, and A Barrat. *Dynamical Processes on Complex Networks*. 2008.
- [VGB14] Christian L. Vestergaard, Mathieu Génois, and Alain Barrat. How memory generates heterogeneous dynamics in temporal networks. *Physical Review E*, 90(4) :042805, oct 2014.
- [VL14] Jordan Viard and Matthieu Latapy. Identifying roles in an IP network with temporal and structural density. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 801–806. IEEE, apr 2014.
- [WFAG12] Qinna Wang, Eric Fleury, Thomas Aynaud, and Jean-Loup Guillaume. Communities in evolving networks : definitions, detection and analysis techniques. *Dynamics of Time Varying Networks*, 2012.
- [WGD13] Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Overlapping community detection using seed set expansion. *CIKM '13*, pages 2099–2108, New York, NY, USA, 2013. ACM.
- [WLWT10] Zhihao Wu, Youfang Lin, Huaiyu Wan, and Shengfeng Tian. A fast and reasonable method for community detection with adjustable extent of overlapping. In *Proceedings of 2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2010*, pages 376–379, 2010.
- [WWL⁺14] Yi Bo Wang, Wen Jun Wang, Dong Liu, Xiao Liu, and Peng Fei Jiao. Using Prior Knowledge for Community Detection by Label Propagation Algorithm. In *Advanced Materials Research*, volume 1049-1050, pages 1566–1571, nov 2014.
- [WZF14] Klaus Wehmuth, Artur Ziviani, and Eric Fleury. A Unifying Model for Representing Time-Varying Graphs. feb 2014.
- [XH14] Kevin S. Xu and Alfred O. Hero. Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal on Selected Topics in Signal Processing*, 8(4) :552–562, 2014.
- [XKS13] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping community detection in networks. *ACM Computing Surveys*, 45(4) :1–35, aug 2013.
- [XSL11] Jierui Xie, Boleslaw K. Szymanski, and Xiaoming Liu. SLPA : Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 344–349, 2011.
- [YCZ⁺11] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Machine Learning*, 82(2) :157–189, 2011.
- [YL13] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale. In *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, page 587, New York, New York, USA, feb 2013. ACM Press.

- [YL15] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1) :181–213, jan 2015.
- [YWW13] L. Yu, B. Wu, and B. Wang. LBLP : link-clustering-based approach for overlapping community detection. *Tsinghua Science and Technology*, 18(4) :-, 2013.
- [Zha15] Pan Zhang. A revisit to evaluating accuracy of community detection using the normalized mutual information. jan 2015.