Anonymisation de réseaux sociaux

Mohammed Ghesmoune

Anonymisation de réseaux sociaux

Mohammed Ghesmoune

Sous la direction de :

Sophie Pinchinat, Sébastien Gambs

Master Recherche en Informatique
Université de Rennes 1
05 Juin 2012

Résumé

Les réseaux sociaux ont été utilisés comme objet d'étude dans de nombreux domaines tels que la sociologie, l'épidémiologie ou encore le marketing viral. Ainsi, il est possible en analysant certaines propriétés structurelles d'un graphe social, telles que le degré des nœuds ou encore le diamètre du graphe, d'en déduire des informations sur la dynamique des individus qui composent ce graphe. Néanmoins, publier directement un graphe social en enlevant simplement le nom des personnes qui ont contribué à ce graphe soulève d'importants problèmes de vie privée. En particulier, certaines attaques par inférence sur le graphe publié peuvent conduire à dé-anonymiser certains nœuds, apprendre l'existence d'une relation sociale entre deux nœuds ou encore à utiliser la structure du graphe elle-même pour déduire la valeur de certains attributs sensibles. Une des manière d'anonymiser un graphe consiste à généraliser certains groupes de nœuds en un super-nœud et plusieurs liens en un méta-lien. Cependant, cette méthode d'anonymisation peut avoir un impact important sur l'utilité résultante qui peut être extraite du graphe généralisé. Dans ce travail de recherche, nous proposons de développer une technique d'anonymisation de graphe social guidée par l'utilité qui cherche à atteindre un équilibre entre les garanties offertes en terme de vie privée et l'utilité résultante du graphe généralisé.

Table des matières

1 Introduction

Un réseau social tel que Facebook, Google+, Myspace, etc., peut être représenté sous la forme d'un graphe social où les nœuds représentent les utilisateurs et les arêtes modélisent les liens sociaux entre individus (amitié, échange financier, inimitié, ...). Chaque utilisateur possède un profil représentant ses relations et interactions dans le réseau, ce profil peut décrire un certain nombre d'informations personnelles sur cette personne (par exemple, son âge, sa localisation, son affiliation politique, ...). Les utilisateurs d'un réseau social ont tendance à cacher leurs interactions et à ne pas partager ses informations qu'avec certains de leurs "amis". Afin de permettre des analyses utiles sur les données d'un réseau social, on anonymise généralement le graphe social avant de le publier, en supprimant les noms des utilisateurs (par exemple, le nom, le numéro de sécurité social, ...), et cela pour préserver la vie privée des membres du réseau.

Malgré cette procédure d'anonymisation, il est parfois possible d'apprendre des liens sociaux entre utilisateurs, de ré-identifier des nœuds du graphe à travers une attaque par inférence. Cette dernière consiste à déduire des informations personnelles, des attributs d'un profil privé, cachées par des individus au moyen d'informations auxiliaires publiques. Ainsi, plusieurs approches d'anonymisation d'un graphe social ont été développés. La généralisation d'un graphe social est reconnu comme une technique, d'anonymisation d'un graphe social, proposant un degré élevé de garantie en termes de préservation de la vie privée (proportionnelement avec la taille minimale des clusters). Cependant, l'utilité d'un tel graphe social publié n'atteint souvent pas un niveau acceptable au niveau pratique. Les analystes d'un graphe social considèrent des propriétés structurelles afin d'étudier l'influence et le pouvoir des nœuds, faire de la marketing virale ou étudier les modèles de propagation de l'information et de la maladie.

Comme exemple d'anonymisation et d'attaque par inférence, nous citons le cas historique, la compétition mise en place par Netflix qui est proposé comme un défi à la communauté d'apprentissage machine pour améliorer la précision de son système de recommandation de films. Chaque client (ligne) donne son score à propos d'un film (colonne). Bien qu'aucun identifiant n'ait été utilisé, une dé-anonymisation a été possible pour un nombre important d'enregistrements à l'aide d'une attaque par inférence utilisant Internet Movie DataBase (IMDB) comme informations auxiliaires [1].

Le reste de ce rapport est organisé comme suit. Tout d'abord, la Section 2 décrit des notions fondamentales nécessaires pour mieux comprendre le reste de ce rapport. Ensuite, la Section 3 détaille quelques algorithmes et méthodes d'attaques par inférence sur un graphe social. Puis, la Section 4 présente un état de l'art sur les approches d'anonymisation d'un graphe social. Après, la Section 5 présente notre algorithme d'anonymisation d'un graphe social guidé par l'utilité qu'on veut préserver du graphe publié. Enfin, la Section 6 conclut ce rapport et propose des perspectives de recherche pour la poursuite de ce projet.

2 Préliminaires

Dans cette section, nous définissons des notions fondamentales telle que l'utilité d'un graphe social, puis nous présentons les difficultés du problème

de l'anonymisation dans les graphes sociaux ainsi que les différents types de mesures d'utilité considérées par les analystes d'un graphe social.

2.1 Modélisation d'un réseau social

Dans ce rapport, on modélise un réseau social comme étant un graphe $G = (V, E)$ enrichi par un ensemble d'attributs A, où V représente l'ensemble des nœuds et tel que chaque nœud correspond à un individu, et E un ensemble d'arêtes tel que chaque arête représente une relation sociale (amitié, intérêts communs, relations sexuelles, échanges financiers, inimitié, etc.) entre deux individus. L'ensemble d'attributs A est tel que pour tout nœud dans V on peut trouver des attributs comme par exemple le nom, le numéro de téléphone, l'âge, etc., et pour chaque arête dans E, on peut la caractériser par un attribut comme par exemple le type de la relation.

2.2 Notions fondamentales

Dans cette partie, nous allons aborder la définition de certaines notions fondamentales dans le domaine de l'anonymisation d'un graphe social et qui sont essentielles pour mieux comprendre le reste de ce rapport telles que les notions d'adversaire, d'attaque par inférence, de connaissance d'un adversaire, d'utilité d'un graphe social, et de k-anonymité.

Définition 2.1 (Adversaire)**.** *Un* adversaire*, dans le contexte des réseaux sociaux, représente toute personne capable de causer un bris de vie privée des utilisateurs en accédant à la totalité ou une partie du graphe social. Par exemple, l'adversaire pourrait être un fournisseur d'un réseau social, un gouvernement, ou encore des chercheurs.*

Définition 2.2 (Attaque par inférence)**.** *Une* attaque par inférence *sur un graphe social anonymisé a pour but de causer un bris de vie privée sur des individus en divulguant certaines informations personnelles non présentes explicitement dans le graphe social rendu public. Cela pourrait être par exemple le fait d'inférer l'affiliation politique d'une personne ou encore le fait qu'elle partage un lien social avec un autre individu.*

On peut distinguer trois catégories de bris de vie privée dans les graphes sociaux [2] :

- *La divulgation d'identité* : l'identité (nom, numéro de sécurité social, ...) de l'individu qui est associé au nœud est révélée.
- *La divulgation de lien* : une relation sociale sensible entre deux individus est divulguée.
- *La divulgation d'attribut* : les données sensibles associées avec un nœud sont dévoilées.

Définition 2.3 (Connaissance d'un adversaire)**.** *Une* connaissance *représente potentiellement toute information auxiliaire sur les utilisateurs du réseau social que possède un adversaire et qui peut l'aider pendant l'attaque par inférence (Définition 2.2), cette connaissance peut être individuelle ou collective.*

Par exemple, l'adversaire peut savoir que Bob a 30 amis, ou qu'il participe au groupe *Yucatan*, ou bien que son âge est de 18 ans.

Définition 2.4 (Utilité d'un graphe social)**.** *L'*utilité d'un graphe social *quantifie à quel point les données issues d'un graphe social (y compris sa structure topologique) sont utilisables dans l'analyse des réseaux sociaux. Une mesure d'utilité est souvent fortement dépendante du domaine d'application considérée. En général, plus le graphe anonymisé partage de l'information et se rapproche du graphe d'origine, plus la mesure d'utilité de ce graphe est importante.*

Définition 2.5 (Anonymisation naïve)**.** *L'*anonymisation naïve *d'un graphe* $G = (V, E)$ *consiste à produire un graphe isomorphe* $G' = (V', E')$*, défini par une bijection aléatoire* $f : V \to V'$*, et tel que les arêtes de* G' *sont* $E' = \{(f(x), f(x'))|(x, x') \in E\}$*.*

Autrement dit, l'anonymisation naïve d'un graphe social consiste à supprimer les noms des utilisateurs et à les remplacer par des pseudonymes. Cette procédure est souvent insuffisante pour préserver la vie privée des individus présents dans un graphe social. Par exemple, si un adversaire sait que Bob a 30 amis, et que dans le graphe social publié il existe un seul nœud de degré 30, l'adversaire peut conclure sans aucun doute que ce nœud correspond à Bob.

Définition 2.6 (k-anonymité)**.** *Soit un graphe social* $G = (V, E)$*, et une connaissance auxiliaire* Q *(Définition 2.3) que possède un adversaire,* G *satisfait la condition de* k-anonymité *par rapport à* Q*, si et seulement si pour tout nœud* $v \in V$*, il existe au moins* $(k-1)$ *autres nœuds dans* V *possèdent la même caractéristique que celle du nœud* v *par rapport à* Q*, c'est-à-dire que au moins* k *nœuds sont indistinguables vis-à-vis de l'adversaire [3, 4].*

Par exemple, si la connaissance d'un adversaire représente le nombre d'amis que possède un individu v (c'est-à-dire le degré, d_v, du nœud v), la condition de k-anonymité exige que le graphe rendu public ait au moins $(k-1)$ autres nœuds ayant le même degré d_v.

2.3 Sources et partage de données

Une partie considérable des attaques sur les réseaux sociaux s'appuie sur le graphe social publié et sur des connaissances auxiliaires que l'adversaire peut acquérir et exploiter. Ces connaissances peuvent avoir été récupérées à partir de plusieurs sources de données :

- *Les graphes sociaux des appels téléphoniques* peuvent être utilisés pour détecter des activités illicites telles que des fraudes [5] et pour détecter des problèmes de sécurité nationale. Ces graphes sociaux contiennent des millions de nœuds.
- *Les sociologistes, épidémiologistes et professionnels de la santé* collectent des données correspondantes à différents types de graphes sociaux (famille, liens sociaux et localisations) pour étudier les risques de la propagation de maladies [1]. Il est possible que l'adversaire puisse accéder à ces données suite à une publication d'informations.
- Pour les réseaux sociaux en ligne, les données peuvent être collectées par *crawling* via une API ou par *screen-scrapping*. Par exemple, Mislove, Marcon, Gummadi, Druschel et Bhattacharjee ont collecté les données de réseaux sociaux tels que Flickr, YouTube, LiveJournal et Orkut [6].
- *Les opérateurs des réseaux sociaux en ligne* partagent parfois leurs graphes sociaux avec des opérateurs publicitaires pour leur permettre un meilleur

1. The National Longitudinal Study of Adolescent Health. http ://www.cpc.unc.edu/projects/addhealth, 2008.

ciblage de la publicité [7]. Par exemple, Facebook mentionne explicitement dans la politique de confidentialité que les profils des utilisateurs peuvent être partagés pour fins de publicités personnalisées tout en prétendant anonymiser ces données.
- *Des applications tierces* peuvent être installées sur certains réseaux sociaux et servir à collecter des informations sur les utilisateurs de ces systèmes. Ainsi, l'application Top-Friends qui donne une note de proximité aux amis, développée par la société Slide, a été suspendue car elle ne respecte pas les choix définis par l'utilisateur en termes de protection de ses données sur Facebook [2].
- *L'agrégation des informations* à partir de plusieurs réseaux sociaux, facilités par des projets comme OpenID [3] et le projet *graphe social* [8], présentent potentiellement une menace pour la vie privée des individus de ces réseaux, une fois que ces données seront publiées.

2.4 Différence entre l'anonymisation dans les graphes sociaux et les bases de données

Bien que le respect de la vie privée dans la publication de bases de données relationnelles a été largement étudié et plusieurs modèles importants de respect de la vie privée tels que k-anonymité [9], l-diversité [10] ainsi que de nombreux algorithmes efficaces ont été proposés, la plupart des méthodes d'anonymisation existantes peuvent traiter uniquement des données relationnelles. Ces méthodes ne peuvent pas être appliquées directement aux données de graphes sociaux. En effet, l'anonymisation des graphes sociaux est plus difficile que celle pour les bases de données.

Premièrement, il est plus difficile de modéliser les connaissances auxiliaires des adversaires sur un graphe social que sur une base de données. Dans le contexte des bases de données, il est souvent supposé qu'un ensemble d'attributs pouvant jouer le rôle de quasi-identificateurs est utilisé pour associer des données provenant de plusieurs bases de données, et les attaques par inférence consistent souvent en la ré-identification des personnes à partir de cet ensemble d'attributs quasi-identificateurs. Cependant, dans un graphe social de nombreuses informations peuvent être utilisées pour identifier les individus, tels que les étiquettes des nœuds et d'arêtes, le degré ou le voisinage d'un nœud, les sous-graphes induits, ainsi que leurs combinaisons. Ainsi, l'anonymisation dans les graphes sociaux est beaucoup plus complexe que dans le cas des bases de données.

Deuxièmement, le calcul de l'information perdue en anonymisant les données d'un graphe social est plus difficile que celle dans l'anonymisation de bases de données. En général, la perte d'information dans une table anonymisée peut être calculée en utilisant la somme de la perte d'information de différents tuples (les enregistrements de la table). Etant donné un tuple dans la table d'origine et le tuple correspondant dans la table anonymisée, on peut calculer la distance entre les deux tuples pour quantifier l'information perdue au niveau d'un tuple. Cependant, un graphe social est constitué d'un ensemble de nœuds et un ensemble d'arêtes. Il est difficile de comparer deux graphes sociaux en comparant les nœuds et les arêtes individuellement. Deux graphes sociaux ayant le même nombre de nœuds et le même nombre d'arêtes peuvent

2. Facebook. Facebook's privacy policy. http ://www.new.facebook.com/policy.php, 2012.
3. OpenID. http ://openid.net, 2008.

être très différents du point de vue de la perspective des propriétés structurelles telles que la connectivité, l'intermédiarité, et le diamètre. Ainsi, il peut y avoir plusieurs façons pertinentes de définir les mesures de perte d'information ainsi que la qualité d'anonymisation.

Troisièmement, la difficulté de mettre au point des méthodes d'anonymisation des données de graphes sociaux en comparaison des méthodes d'anonymisation de données relationnelles. Les approches du type *diviser pour régner* sont largement appliquées dans l'anonymisation de données relationnelles en raison du fait que les tuples d'une table relationnelle sont séparables durant la phase d'anonymisation. En d'autres termes, l'anonymisation d'un groupe de tuples n'affecte pas les autres tuples de la table. Cependant, l'anonymisation d'un graphe social est beaucoup plus complexe car changer les étiquettes des nœuds et des arêtes peut affecter les voisinages d'autres nœuds, et la suppression et l'ajout de nœuds et d'arêtes peut affecter d'autres nœuds et arêtes ainsi que les propriétés d'un graphe.

2.5 Mesures d'utilité

L'objectif principal derrière la publication d'un graphe social est de pouvoir permettre de faire des analyses sur la structure du graphe. En général, la qualité des analyses qui peuvent être faites à partir du graphe publié sont quantifiées par une mesure d'utilité. Jusqu'à présent, trois types d'utilités ont été considérées dans la littérature [11] :

- **Propriétés structurelles (topologiques)**. L'une des plus importantes applications des données d'un graphe social publié est d'analyser les propriétés structurelles de son graphe. Pour comprendre et utiliser les informations contenues dans un graphe social, les chercheurs ont développé diverses mesures pour indiquer la structure et les caractéristiques d'un graphe de différents points de vue. Cela inclut des propriétés telles que la séquence de degrés, le diamètre d'un graphe, le cœfficient de clustering, etc.
- **Propriétés spectrales**. Le spectre d'un graphe peut être défini comme l'ensemble des valeurs propres de la matrice d'adjacence (ou autres matrices telle que la matrice Laplacienne) représentant le graphe étudié. Le spectre d'un graphe est fortement corrélé avec de nombreuses propriétés structurelles d'un graphe.
- **Requêtes d'agrégation d'un graphe social**. Une requête d'agrégation sur un graphe social calcule l'agrégat de certaines propriétés sur certains chemins ou sous-graphes satisfaisant des conditions d'une requête. Par exemple, la requête pourrait être "quelle est la distance moyenne d'un médecin à un enseignant dans un graphe social ?".

Les propriétés structurelles d'un graphe social seront abordées en détails dans la Section 5.

3 Etat de l'art des attaques par inférence d'un graphe social

Il existe dans la littérature trois types de bris de la vie privée des individus présents dans un graphe social : (1) la divulgation d'identité qui consiste à révéler l'identité de l'individu qui est associé au nœud, (2) la divulgation d'attribut où les données sensibles associées avec un nœud sont dévoilées et (3)

la divulgation de lien où une relation sociale sensible entre deux individus est divulguée. Nous allons présenter et détailler les deux premiers types d'attaques en décrivant des techniques permettant de faire réussir de telles attaques.

3.1 Ré-identification de nœuds

Dans cette partie, nous allons présenter quelques types de connaissances que l'adversaire peut acquérir en les modélisant formellement sous forme de requêtes, puis nous détaillerons quelques algorithmes pouvant être appliqués par l'adversaire pour pouvoir ré-identifier des individus présents dans un graphe social rendu public.

3.1.1 Connaissances d'un adversaire

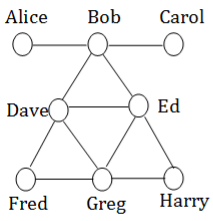
Dans [12], les auteurs ont étudié deux classes de requêtes décrivant les connaissances externes disponibles à un adversaire pour réussir une attaque sur un graphe social anonymisé. Ils modélisent les connaissances auxiliaires d'un adversaire comme l'accès à une source qui fournit des réponses à une requête de connaissance restreinte Q concernant un seul nœud cible dans le graphe original.

a. **Requêtes de raffinement de nœuds**

Ces requêtes de connaissance, H_i, décrivent la structure locale du graphe autour du nœud cible d'une manière qui permet un raffinement itératif de nœuds à travers ces requêtes. Ainsi, la requête $H_0(x)$ retourne l'étiquette d'un nœud x. Or, les noms des nœuds sont cachés (anonymisés), donc $H_0 = \varepsilon$, $H_1(x)$ retourne le degré d'un nœud x, $H_2(x)$ retourne les degrés de chaque voisin d'un nœud x. $H_i(x)$ peut être défini récursivement comme :

$$H_i(x) = \{H_{i-1}(y) \mid (x, y) \in E\}.$$

Exemple 3.1 (Calcul de H_0, H_1 et H_2)**.** *La Figure 1 illustre le calcul de H_0, H_1 et H_2 de chaque nœud du graphe 1(a). Ainsi, H_0 vaut uniformément ε, $H_1(Bob) = \{\varepsilon, \varepsilon, \varepsilon, \varepsilon\}$, abrégé dans le tableau simplement comme 4. En utilisant cette abréviation, $H_2(Bob) = \{1, 1, 4, 4\}$ qui représente les degrés des voisins de Bob.*



(a) graphe

Nœud	H_0	H_1	H_2
Alice	ε	1	{4}
Bob	ε	4	{1,1,4,4}
Carol	ε	1	{4}
Dave	ε	4	{2,4,4,4}
Ed	ε	4	{2,4,4,4}
Fred	ε	2	{4, 4}
Greg	ε	4	{2, 2, 4, 4}
Harry	ε	2	{4, 4}

(b) Raffinement de nœuds

Relation d'équivalence	Classes d'équivalence
$\equiv_{H_0}$	{A,B,C,D,E,F,G,H}
$\equiv_{H_1}$	{A,C}{B,D,E,G}{F,H}
$\equiv_{H_2}$	{A,C}{B}{D,E}{G}{F,H}
$\equiv_{H_A}$	{A,C}{B}{D,E}{G}{F,H}

(c) Classes d'équivalence

Figure 1 – (a) un graphe social ; (b) connaissances auxiliaires consistant en requêtes de raffinement de nœuds calculées pour chaque individu dans le graphe ; (c) les classes d'équivalences implicites par le raffinement de nœuds.

Définition 3.1 (Relation d'équivalence)**.** *Deux nœuds x et y dans un graphe sont* équivalents *relativement à H_i, notée $x \equiv_{H_i} y$, si et seulement si $H_i(x) = H_i(y)$.*

Par exemple, en se basant sur le tableau 1(c), Bob et Dave sont équivalents relativement à la requête H_1 et donc ne peuvent pas être distingués par un adversaire qui ne possède que la la connaissance auxiliaire correspondante à la requête H_1. Cependant, relativement à H_2, ils appartiennent à deux classes d'équivalence différentes et donc ils sont distinguables si l'adversaire peut poser la requête de connaissance H_2, c'est-à-dire il y aura une atteinte de la vie privée des individus Bob et Dave, en les ré-identifiant dans leur graphe social publié, par un adversaire possédant la requête H_2.

Théorème 3.1 (Babai et Kucera [13]). *Soit $G = (V, E)$ un graphe aléatoire sur n nœuds avec probabilité d'arête $p = \frac{1}{2}$. La probabilité qu'il existe deux nœuds distincts $x, y \in V$ tel que $x \equiv_{H_3} y$ est moins que 2^{-cn}, pour une certaine constante $c > 0$.*

Ce théorème démontre que la connaissance de $H_3(x)$ par l'adversaire lui permet de ré-identifier avec une forte probabilité un nœud cible x.

b. **Requêtes de connaissance de sous-graphe**

Les requêtes de raffinement de nœuds pré-supposent une connaissance complète sur les nœuds adjacents du nœud cible. Cependant, en pratique, l'adversaire peut souvent obtenir seulement une liste partielle des voisins du nœud cible. Un autre inconvénient est que les requêtes H décrivent arbitrairement des sous-graphes importants centrés autour de x si ce nœud est fortement connecté (l'adversaire apprend alors un sous-graphe dans G).

Une autre possibilité est de considérer une classe de requêtes qui confirment l'existence d'un sous-graphe autour du nœud cible. La puissance descriptive d'une telle requête est proportionnelle au nombre d'arêtes dans le sous-graphe décrit. Par exploration du voisinage du nœud cible x, l'adversaire apprend l'existence d'un sous-graphe représentant une connaissance partielle sur la structure autour de x. L'existence de ce sous-graphe peut être exprimée par une requête et l'octroi d'une réponse à cette requête modélise la connaissance d'un adversaire.

Exemple 3.2 (Sous-graphes autour d'un nœud). *La Figure 2 montre trois sous-graphes centrés autour de Bob, le premier confirme que Bob a (au moins) trois voisins, le deuxième décrit un arbre de nœuds proches de Bob, et le troisième établit les nœuds à proximité de Bob dans un sous-graphe.*

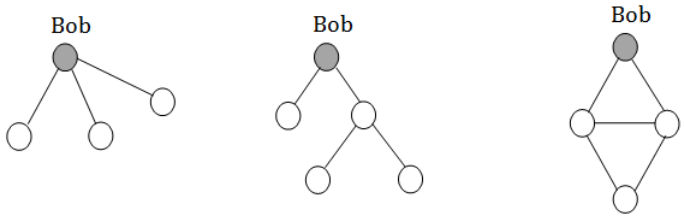


Figure 2 – Trois instances de l'information partielle autour de Bob qui peut être décrite sous forme de requêtes de connaissance de sous-graphe, tirée de [12].

Les requêtes ci-dessus représentent un ensemble de connaissances structurelles qui peuvent être accessibles à un adversaire connaissant des descriptions complètes ou partielles sur les voisinages locaux d'un nœud. Les requêtes de raffinement de nœuds fournissent des informations complètes sur le degré d'un nœud, alors qu'une requête de sous-graphe ne peut pas

exprimer une connaissance H_i. En effet, les requêtes de sous-graphes sont existentielles et ne peuvent pas confirmer les contraintes exactes sur les degrés ou l'absence d'une arête dans le graphe. Souvent, il est difficile pour un adversaire d'acquérir une description structurelle complète et détaillée d'une requête de raffinement de nœuds de haut-niveau, d'où la sémantique des requêtes de sous-graphe pour modéliser la capacité d'un adversaire réaliste.

3.1.2 Attaque passive

Lors d'une attaque passive, l'adversaire n'influe pas directement sur le réseau social, il n'est donc qu'un observateur qui essaie de comprendre sa structure. Différents types d'attaques passives sont possibles. Ainsi, Backstrom, Dwork et Kleinberg [14] considèrent une coalition entre plusieurs adversaires passifs qui sont des voisins dans le graphe G pour compromettre la vie privée d'autres voisins. Ainsi, les voisins connectés à un unique sous-ensemble de la coalition seront reconnus sans aucune ambiguité une fois la coalition identifiée. Après la libération de G, la coalition applique le même algorithme basé sur l'attaque par marche aléatoire (décrit plus loin dans la section 3.1.3) pour trouver les nœuds cibles dans le graphe anonymisé, et donc inférer l'existence de liens entre ces nœuds. Cette implémentation d'attaque ne cible pas un ensemble de nœuds particuliers, mais plutôt un ensemble de voisins de la coalition (on suppose que les attaquants connaissent les noms de leurs voisins).

Dans [15], les auteurs ont proposé un algorithme de ré-identification de nœuds en faisant une correspondance entre un graphe auxiliaire, dont les nœuds sont connus, et le graphe anonymisé, sous l'hypothèse qu'une fraction importante des nœuds cibles appartient aux deux graphes. Cet algorithme fonctionne en deux étapes :

- **Identification initiale** : L'adversaire identifie un petit nombre de nœuds "germes" qui sont présents dans le graphe auxiliaire et le graphe anonymisé. On pourra par exemple, appliquer les attaques décrites dans la Section 3.1.3 pour cela.
- **Propagation** : L'algorithme de propagation prend deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, et établit une correspondance déterministe 1 à 1 entre les nœuds des deux graphes. Ainsi, la correspondance des nœuds construite durant la première étape sera étendue en utilisant la topologie du graphe.

  À chaque itération, l'algorithme choisit arbitrairement un nœud $u \in V_1$ non encore associé et calcule un *score* pour chaque nœud non associé $v \in V_2$ égal au nombre de voisins de u ayant été associés aux voisins de v. Si l'*excentricité* de l'ensemble de ces scores est supérieure à un seuil, on ajoute une association entre u et le nœud v tel que $score(u, v) = \max(scores(u))$. Plus précisément, l'excentricité est une métrique définie dans [1] dans le contexte de désanonymisation de bases de données. Elle mesure l'éloignement d'un élément quelconque appartenant à un ensemble X par rapport au reste de cet ensemble, et est définie par :

$$Ecc(X) = \frac{\max_1(X) - \max_2(X)}{\sigma(X)}.$$

  où $\max_1(X)$ et $\max_2(X)$ dénotent la plus grande valeur et la deuxième plus grande valeur de l'ensemble X et σ dénote l'écart type.

Le lecteur curieux peut trouver le pseudo-code décrivant l'algorithme de propagation de manière plus détaillée dans [15]. La complexité de cet algorithme est : $O\left((|E_1| + |E_2|)\, d_1 d_2\right)$, où d_1 (respectivement d_2) est une borne sur les degrés des nœuds de V_1 (respectivement V_2). L'exécution de cet algorithme sur le graphe anonymisé de Twitter (comme graphe cible) et le graphe social de Flickr (comme connaissances auxiliaires), montre que 30.8% d'associations entre nœuds sont corrects, alors que 12.1% de nœuds identifiés sont incorrects, et 57% ne sont pas identifiés [15].

3.1.3 Attaque active

Lors d'une attaque active, l'adversaire cherche à modifier et influencer la structure du graphe avant qu'il soit rendu public. La version la plus simple consiste à créer autant de nœuds x_i que de nœuds cibles w_i en liant chaque x_i à w_i pour tout $i = 1, ..., k$. Puis, on ajoute un arc entre chaque paire (x_i, x_j) avec probabilité $\frac{1}{2}$. Par exemple, on tire à pile-ou-face, si on voit "pile" on crée l'arc (x_i, x_j), alors que si le résultat est "face" on ne le crée pas. On répète cette opération pour chaque paire (x_i, x_j) pour tout $i, j = 1, ..., k$. Ce sous-graphe, $G[W] = H$, est construit avant que le graphe social soit publié. En pratique, la construction du sous-graphe H réussit si :

- Toute copie de H qu'on trouve dans G est celle que l'adversaire a construite.
- H peut être trouvé de manière rapide et efficace étant donné G.
- Une fois H trouvé, on peut correctement étiqueter ses nœuds comme $x_1, ..., x_k$ et ainsi trouver $w_1, ..., w_k$.

La section suivante présente deux attaques actives différentes.

a. **Attaque par marche aléatoire**

La version complète de la construction du sous-graphe H est obtenue en prenant un nombre de nœuds cibles ($|W| = b$) supérieur à k. On construit b comme sous-ensembles $N_i \subseteq X$, puis on lie le nœud w_i à tous les nœuds de N_i, de façon à ce que w_i est le seul nœud du graphe $G - H$ attaché précisément aux nœuds de N_i. La recherche du chemin $x_1, ..., x_k$ conduit à l'identification de H. L'arbre de recherche T représente le déroulement de la recherche. Chaque nœud dans T correspond à un nœud dans le graphe G. On construit T de manière à ce que chaque chemin dans T corresponde à un chemin dans G. Pour un point intermédiaire dans la construction de T, on prend chaque nœud feuille α (soit l l'ordre du nœud α dans le chemin partant de la racine de T) et on lui ajoute un nouveau fils β si ce dernier a le même degré et la même structure que le nœud x_{l+1}. Finalement s'il existe dans T un seul chemin de longueur k, alors celui-ci doit correspondre à H. Une fois H trouvé, les nœuds cibles sont facilement identifiables.

b. **Attaque basée sur la coupe**

Dans cette attaque, le sous-graphe H est construit en ajoutant seulement $b = O\sqrt{\log(n)}$ nœuds (on peut remarquer que $O(\log(n))$ nœuds sont nécessaires pour la première attaque, où n est le nombre de nœuds présents dans le graphe G) qui sont fortement connectés entre eux et faiblement liés avec le reste du graphe. Un danger de cette attaque est qu'elle peut être détectée par les opérateurs des réseaux sociaux. Après la divulgation de G, on procède à la recherche du sous-graphe H comme suit [14] :

- On calcule l'arbre de Gomory-Hu [16], T, du graphe G. Il s'agit d'un arbre avec arêtes pondérés sur l'ensemble de nœuds V du graphe G, tel

que pour tout $u, v \in V$, la valeur de la coupe minimum $u - v$ dans G est égale au poids minimum de l'arête sur le chemin de $u - v$ dans l'arbre T.
- On supprime toutes les arêtes de poids au plus b de l'arbre T, produisant une forêt T'. Pour trouver l'ensemble de nœuds X qu'on a construit, on cherche toutes les composantes (ensembles de nœuds $S_1, ..., S_r$) dans T' de même taille X. Les auteurs prouvent qu'avec une forte probabilité un seul sous ensemble Si est isomorphe au sous-graphe $H = G[X]$.
- Tant que H n'a pas d'automorphes, en connaissant S_i on peut identifier les nœuds $x_1, ..., x_b$ qu'on a lié aux nœuds cibles $w_1, ..., w_b$ respectivement. Ainsi, l'identification des nœuds cible se fait sans ambiguïté.

L'attaque par marche aléatoire possède un algorithme de recouvrement rapide, donc il est applicable de manière efficace même sur des réseaux sociaux ayant des millions d'utilisateurs, alors que l'utilisation de l'arbre de Gomory-Hu rend l'algorithme de recouvrement de l'attaque par coupe plus coûteux en termes de temps d'exécution. En créant k nouveaux nœuds par l'adversaire, l'attaque par marche aléatoire peut compromettre $\Theta(k^2)$ utilisateurs, alors que l'attaque par coupe peut compromettre seulement $O(k)$ utilisateurs. Expérimentalement, la création de 7 nouveaux nœuds dans le réseau social LiveJournal (qui possède 4.4 millions de nœuds et 77 millions d'arêtes) par l'attaquant, l'algorithme d'attaque par marche aléatoire arrive à révéler une moyenne de 70 nœuds cibles, et ainsi $\binom{70}{2} = 2415$ liens entre eux [14]. Une attaque hybride dite semi-passive est possible, dans laquelle les attaquants, formant la coalition, ne créent pas de nouveaux nœuds mais plutôt de nouveaux liens vers les nœuds cibles.

Backstrom, Dwork et Kleinberg dans leur travail [14] sont considérés comme des pionniers du domaine de la préservation de la vie privée des utilisateurs d'un réseau social. Même si leurs travaux démontrent un risque sérieux en termes de bris de la vie privée des utilisateurs d'un réseau social, les attaques actives sont difficiles à adapter sur une grande échelle pour les raisons suivantes :

1. Elles sont limitées aux réseaux sociaux en ligne.
2. L'attaquant a peu de contrôle sur les arcs entrant vers les nœuds qu'il crée (les attaques actives sont faciles à détecter).
3. La troisième limitation des attaques actives est le fait que de nombreux opérateurs de réseaux sociaux comme Facebook ont besoin de faire un lien mutuel avant que l'information est mise à disposition sous toute forme. Par conséquent, en supposant que les utilisateurs réels n'ont pas un lien vers les utilisateurs factices, les liens à partir de faux nœuds vers les vrais nœuds ne se présentent pas dans le réseau.

L'attaques passive est réaliste, mais là encore, ne fonctionne que sur une petite échelle (ils ne peuvent compromettre que la vie privée de certains des utilisateurs qui sont déjà leurs amis). Cependant, le problème de la conception de techniques qui pourraient protéger la vie privée des utilisateurs d'un réseau social n'a pas été abordé dans [14].

3.2 Inférence d'attributs

Dans [17], les auteurs ont démontré qu'il est possible d'apprendre des informations personnelles même sur des utilisateurs ayant des profils privés à travers sa connaissance du graphe social causant ainsi un bris de la vie privée

des individus, en publiant, dans la plupart des réseaux sociaux, les appartenances aux groupes des utilisateurs.

Un *attribut sensible* est une donnée à caractère personnel, tel que l'âge, l'affiliation politique, la localisation. Les auteurs de [17] considèrent que les utilisateurs n'ont que des attributs sensibles. Un attribut sensible prend une des valeurs possibles parmi un ensemble $\{a_1, ..., a_m\}$. Le profil d'un utilisateur est associé à un identifiant unique avec lequel l'utilisateur forme des liens et participe dans des groupes. Chaque profil est composé d'au moins un attribut sensible qui peut être soit observé, soit caché. Dans un profil publique la valeur de son attribut sensible est observée, alors que dans un profil privé est un profil dans lequel la valeur de l'attribut sensible est caché. L'ensemble des utilisateurs (nœuds) ayant des profils privés représente l'ensemble sensible, dénoté V_s, alors que le reste des nœuds, V_o, représente l'ensemble observé. Le problème d'inférence des attributs sensibles est d'inférer les valeurs d'attributs sensibles cachées, $V_s.A$, en se basant sur les valeurs d'attributs sensibles observées, les liens entre les nœuds, et les appartenances aux groupes dans le graphe G.

a. **Attaques utilisant la distribution globale des attributs**
   En l'absence d'information sur les liens sociaux entre individus et leur appartenance à des groupes, la principale information disponible est la distribution globale des attributs sensibles dans les profils publics. Dans ce cas, l'adversaire applique un modèle d'inférence simple, BASIC, pour prédire les valeurs des attributs sensibles des profils privés. La probabilité d'une valeur sensible est estimée par :

$$P_{BASIC}(v_s.a = a_i; G) = P(v_s.a = a_i | V_o.A) = \frac{(|V_o.a_i|)}{(|V_o|)},$$

   tel que : $P_{BASIC}(v_s.a = a_i; G)$ est la probabilité que, la valeur de l'attribut sensible a égale a_i, du nœud $v_s \in V_s$ suivant le modèle BASIC et la partie observée du graphe G, $|V_o.a_i|$ est le nombre de profils publics ayant a_i pour valeur de l'attribut sensible, et $|V_o|$ est le nombre total de profils publics. Suivant ce modèle, l'adversaire choisit la valeur la plus probable d'après la distribution globale des attributs.

b. **Attaques se basant sur les liens sociaux**
   Les attaques par inférence se basant sur les liens sociaux tirent parti de la propriété d'auto-corrélation, qui mesure la corrélation entre attributs de profils qui sont liés. Par exemple l'auto-corrélation entre des personnes qui sont "amis" est souvent forte car ils partagent souvent des caractéristiques communes.

   – **Modèle ami-agrégat (AGG)**
     Ce modèle prend en considération la distribution des attributs sensibles parmi les amis de la personne dont on veut inférer les attributs privés. En se basant sur ce modèle, la probabilité d'une valeur d'un attribut sensible est estimée par :

$$P_{AGG}(v_s.a = a_i; G) = P(v_s.a = a_i | V_o.A, E) = \frac{(|V'_o.a_i|)}{(|V'_o|)},$$

     tel que :$V'_o = \{v_s \in V_o | \exists (v_s, v_o) \in E\}$ et $V'_o.a_i = \{v_o \in V'_o | v_o.a = a_i\}$.
     Suivant ce modèle, l'adversaire choisit la valeur la plus probable d'après la distribution globale des attributs de ses amis. Un défaut de cette méthode est celui quand les amis de la personne sont très divers alors leur distribution va être aussi très diverse.

– **Modèle de classification collective (CC)**
La classification collective vise à apprendre et déduire les attributs d'objets reliés entre eux. Dans notre contexte, elle fait appel non seulement des profils publics mais aussi les valeurs déduites des profils privés connectés. Les auteurs utilisent l'algorithme de classification itérative (ICA) [18], comme algorithme d'inférence approximative. Sans rentrer dans les détails, ICA commence à attacher une étiquette à chaque profil privé en se basant sur les étiquettes des amis ayant un profil public, puis il réaffecte itérativement des étiquettes en considérant les étiquettes des amis de profils à la fois publics et privés. L'affectation est basée sur un classificateur local qui prend des étiquettes de classe des amis comme entrées et essaye de prédire en sortie une nouvelle étiquette d'un nœud privé. Par exemple, un classificateur simple pourrait attribuer une étiquette en se basant sur l'étiquette majoritaire parmi les étiquettes des amis.
– **Modèle se basant sur les liens (LINK)**
Une autre approche se basant sur les liens consiste à "lisser" les données en tenant compte de la matrice d'adjacence du graphe social. Chaque utilisateur possède une liste de caractéristiques binaires de la taille du réseau, et chaque caractéristique a une valeur de 1 si l'utilisateur est ami avec la personne qui correspond à cette fonctionnalité, et 0 sinon. Chaque utilisateur a également une étiquette de classe qui est considéré connue si son profil est public, et inconnue s'il est privé. Les instances avec des profils publics sont les données d'entraînement d'un classificateur "traditionnel" d'apprentissage machine tels qu'un prédicteur de Bayes naïf. Le modèle appris peut alors être utilisé pour prédire les étiquettes de profils privés.
– **Attaque basée sur les clusters (BLOCK)**
L'idée fondamentale derrière cette attaque est que les utilisateurs forment des catégories naturelles (clusters), et leurs interactions peuvent être expliquées par les clusters auxquels ils appartiennent. Si des valeurs potentielles d'un attribut sensible séparent les différents utilisateurs en clusters, sur la base des interactions observées d'un utilisateur de profil privé avec les profils publics des utilisateurs, on peut prédire le cluster le plus probable auquel appartient l'utilisateur et ainsi lui associer la valeur majoritaire de l'attribut recherché.

c. **Attaques se basant sur les groupes**

En plus des informations de liens sociaux, les réseaux sociaux offrent une structure plus complexe où chaque utilisateur peut appartenir à un ou plusieurs groupes. Si un utilisateur appartient à un seul groupe, il sera évident d'inférer une étiquette en utilisant un agrégat par exemple le mode des étiquettes de son groupe. Le problème d'inférence des attributs sensibles devient plus difficile si l'utilisateur participe à plusieurs groupes. Deux modèles ont été proposés pour utiliser l'appartenance des utilisateurs à plusieurs groupes :

– **Modèle groupe-lien (CLIQUE)**
Dans ce modèle, on suppose que les membres d'un groupe soient des "amis", créant ainsi un lien de similarité entre les utilisateurs qui ont au moins un groupe en commun. Cette représentation des données nous permet d'appliquer n'importe lequel des modèles basés sur les liens décrits précédemment. Une limite de ce modèle est qu'il ne tient pas compte de la force de la relation entre deux personnes, comme par exemple, le nombre

de groupes en commun.

– **Modèle de classification basé sur les groupes (GROUP)**
L'approche de classification fondée sur les groupes comporte trois étapes. Durant la première étape, l'algorithme sélectionne les groupes qui sont pertinents à la tâche de classification de nœuds. Pendant la deuxième étape, l'algorithme apprend une fonction globale f (par exemple un classificateur) qui prend les groupes pertinents d'un nœud comme caractéristiques d'entrées et retourne la valeur de l'attribut sensible. Cette étape utilise uniquement les nœuds de l'ensemble des attributs observés dont les attributs sensibles sont connus. Lors de la troisième étape, le classificateur retourne l'attribut sensible prédit pour chaque profil privé.

d. **Attaques combinant les liens et les groupes**
Il est possible de construire une méthode qui utilise à la fois les liens et les groupes pour prédire les attributs sensibles des utilisateurs. La méthode simple proposée combine les deux modèles LINK et GROUP.

Les auteurs de [17] ont appliqué ces méthodes sur un ensemble de données issues de quatre réseaux sociaux (Flicker, Facebook, Dogster, BibSonomy) en supposant que 50% des profils sont privés. Les méthodes basées sur les liens sociaux n'ont pas produit des résultats importants, alors que les méthodes basées sur les groupes ont donné des précisions, d'inférences d'attributs de profils privés, importantes (par exemple, la méthode GROUP a donnée une précision d'inférence d'attributs privés égale à 63.5% sur les données du réseau social Flicker, alors que la méthode utilisant la distribution globale des attributs a donné 27.7%).

4 Etat de l'art sur les approches d'anonymisation

Pour se protéger contre différentes attaques sur un graphe social, les chercheurs ont développé de nombreux modèles de protection de la vie privée et différentes méthodes d'anonymisation d'un graphe social ont été conçues. Semblable à la conception des méthodes d'anonymisation pour les bases de données, la conception des méthodes d'anonymisation d'un graphe social doivent également prendre en compte les modèles d'attaque ainsi que l'utilité des données publié. Nous présentons dans cette section un état de l'art sur les approches d'anonymisation des données d'un réseau social.

4.1 Anonymisation de degrés

Liu et Terzi [19] proposent une approche d'anonymisation d'un graphe social contre un adversaire possédant les degrés de quelques nœuds, présents dans le graphe publié, comme connaissances auxiliaires. Ainsi, si cet adversaire sait que Bob a 30 amis et que dans le graphe publié il existe un seul nœud de degré 30, ce nœud sera aucun doute Bob. Ainsi, leur approche consiste à rendre chaque valeur de degré apparait au moins k fois dans le graphe rendu public. L'algorithme proposé s'exécute en deux étapes :

a. Soient $d = (d(1), ..., d(|V|))$, une séquence non croissante des degrés d'un graphe $G = (V, E)$, et k un entier, construire une autre séquence de degrés d' où chaque degré apparait au moins k fois (la condition de k-anonymité) et tel que le coût d'anonymisation de degrés $D\mathrm{A}(d', d) = \sum_i |d'(i) - d(i)|$ soit minimal (où $d(i)$ est le degré du nœud x_i). Pour résoudre cela, les auteurs de [19] utilisent un algorithme de type programmation dynamique.

b. Soit $G = (V, E)$ un graphe et d' une séquence de degrés, construire un nouveau graphe $G_a = (V, E_a)$ tel que la séquence des degrés du graphe G_a est égale à d' (c'est-à-dire, $d_{G_a} = d'$), et que la différence structurelle entre G_a et G qui est la différence symétrique entre leurs ensembles d'arêtes, $\Delta(G_a, G) = |E_a \setminus E| + |E \setminus E_a|$, soit minimale. Pour cela, deux algorithmes sont conçus :
   - Algorithme *ConstructGraph* : cet algorithme prend en entrée la séquence désirée de degrés d et rend en sortie un graphe consistant avec cette séquence de degrés si un tel graphe existe, sinon il retourne "Non". À chaque itération, on choisit arbitrairement un nœud v et on ajoute des arêtes de v à $d(v)$ nœuds ayant de plus grandes valeurs de degrés restants, où $d(v)$ est le degré restant du nœud v. Dans ce cas, les degrés restants de ces $d(v)$ nœuds sont décrémentés de 1.
   - Algorithme *GreedySwap* : cet algorithme est une heuristique gloutonne qui étant donnés G'_0 (le graphe obtenu en appliquant l'algorithme *ConstructGraph*), et le graphe original G, il transforme G'_0 à $G'(V, E')$ ayant pour séquence de degrés $d_{G'} = d = d_{G'_0}$ de telle sorte que $\Delta(G', G)$ soit minimale. A chaque étape i, le graphe $G'_{i-1}(V, E'_{i-1})$ est transformé en un graphe $G'_i(V, E'_i)$ tel que $d_{G'_0} = d_{G_{i-1}} = d_{G'_i} = d$ et $\Delta(G'_i, G) < \Delta(G'_{i-1}, G)$. La transformation est effectuée en utilisant une opération d'échange (*swap*, en anglais) valide comme la montre la Figure 3.

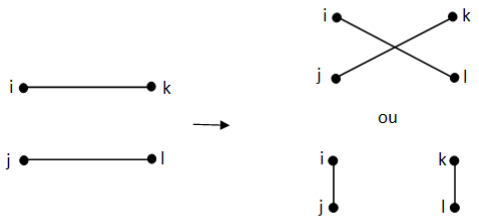


Figure 3 – Opération d'échange.

4.2 Anonymisation de voisinage

Dans [20], Zhou et Pei proposent une méthode d'anonymisation d'un graphe social par anonymisation de voisinage. Cette méthode satisfait la condition de k-anonymité et qui est consacrée contre les attaques se basant sur le voisinage d'un nœud pour lui ré-identifier dans un graphe social rendu public. On considère que les nœuds sont étiquetés (portent des attributs). Avant de présenter leur algorithme, nous commençons par définir quelques notions fondamentales.

4.2.1 Composante de voisinage

On définit le voisinage d'un nœud u par le sous-graphe induit sur les nœuds voisins du nœud u et est écrit $voisinage_G(u) = G(V_u)$ où $V_u = \{v|(u, v) \in E(G)\}$. Une composante de voisinage d'un nœud u est un sous-graphe connexe maximal composé de nœuds voisins au nœud u. La Figure 4 montre le voisinage d'un nœud u, $voisinage_G(u)$, qui contient trois composantes de voisinage C_1, C_2 et C_3.

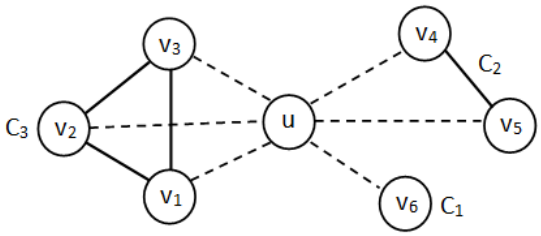


Figure 4 – Voisinage et composantes de voisinage (les arêtes pointillées sont juste pour illustration), tirée de [20].

Remarque 4.1 (Différence entre voisinage et degré d'un nœud)**.** *La Figue 4 montre la différence entre le degré et le voisinage d'un nœud. Ainsi, un adversaire possédant le voisinage d'un nœud u comme connaissances auxiliaires apprend, en plus du degré du nœud u, l'existence de liens entre ses voisins.*

4.2.2 Coût d'anonymisation

Soit u un nœud étiqueté par l_1 (par exemple l_1="enseignant d'école primaire") tel que l_1 est un nœud feuille dans la hiérarchie des étiquettes. Supposant que l_1 est généralisé à l_2 (par exemple l_2="enseignant"), on note $taille(l_2)$ le nombre de descendants de l_2 dans la hiérarchie des étiquettes, et $taille(*)$ est le nombre total des feuilles dans la hiérarchie des étiquettes. On définit la *Pénalité Normalisée de la Certitude* par :

$$PNC(l_2) = \frac{taille(l_2)}{taille(*)}.$$

Soit $G = (V, E)$ un graphe social, et $G' = (V', E')$ le graphe anonymisé publié. On suppose que pas de faux nœuds sont ajoutés au graphe publié. Ainsi, il existe une bijection $\mathcal{A} : V \to V'$. Soient u_1 et $u_2 \in V$ deux nœuds, supposant que leurs voisinages, $voisinage_G(u_1)$ et $voisinage_G(u_2)$, sont généralisés à $voisinage_{G'}(\mathcal{A}(u_1))$ et $voisinage_{G'}(\mathcal{A}(u_2))$, tel que $voisinage_{G'}(\mathcal{A}(u_1))$ et $voisinage_{G'}(\mathcal{A}(u_2))$ sont isomorphes. Soit $H = voisinage_G(u_1) \cup voisinage_G(u_2)$, et $H' = voisinage_{G'}(\mathcal{A}(u_1)) \cup voisinage_{G'}(\mathcal{A}(u_2))$. Le coût d'anonymisation est défini comme :

Coût$(u, v) = \alpha.\sum_{v' \in H'} PNC(v') + \beta.|\{(v_1, v_2)|(v_1, v_2) \notin E(H), (\mathcal{A}(v_1), \mathcal{A}(v_2)) \in E(H')\}| + \gamma.(|V(H')| - |V(H)|)$.

tel que α, β et γ sont des poids spécifiés par la personne en charge de l'anonymisation du graphe. Littérairement, le coût d'anonymisation consiste en trois parties : la première partie mesure la perte d'information en généralisant les étiquettes des nœuds. La deuxième partie mesure la perte d'information en ajoutant des arêtes. Enfin, la dernière partie comptabilise le nombre de nœuds qui sont reliés aux voisinages anonymisés pour atteindre le k-anonymité.

4.2.3 Anonymisation de voisinage

La Figure 5 montre la procédure d'anonymisation des voisinages de deux nœuds u et v où les nœuds sont représentés sous la forme (*id*, *étiquette*). La composante de voisinage $C_2(u)$ correspond exactement à $C_3(v)$. $C_1(u)$ et $C_1(v)$

sont correspondus en ajoutant un nouveau nœud w_1, existant dans le graphe social à anonymiser, à la composante $C_1(v)$. La sélection du nœud à ajouter est basée sur l'optimisation du coût d'anonymisation. De la même façon on anonymise $C_3(u)$ et $C_2(v)$ par l'ajout d'un nouveau nœud w_2 à la composante $C_3(u)$.

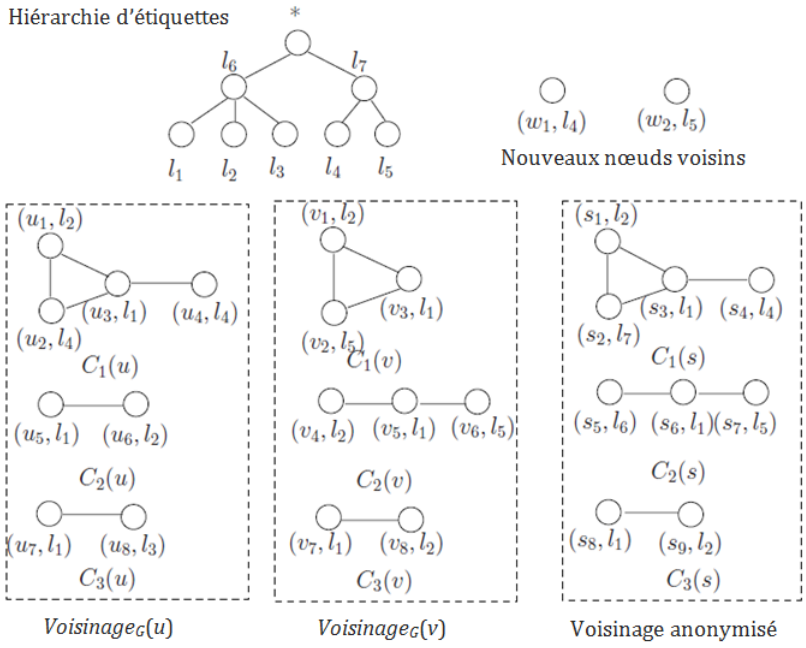


Figure 5 – Anonymisation de deux voisinages, adaptée de [20].

4.2.4 Algorithme d'anonymisation

L'algorithme d'anonymisation prend en entrée un graphe social, un paramètre d'anonymisation k, et trois paramètres α, β et γ qui servent à calculer la fonction de coût. La sortie de l'algorithme est un graphe anonymisé satisfaisant la condition de k-anonymité de façon à ce que les voisins d'au moins k nœuds sont isomorphes. L'exécution de l'algorithme se déroule de la manière suivante.

D'abord, on marque tous les nœuds comme "*non-anonymisé*", puis on ordonne une liste, *ListeNœud*, contenant les nœuds marqués "*non-anonymisé*" suivant la taille de leur voisinage par ordre décroissant. Puis, itérativement on choisit le premier nœud, "*NœudGerme*", dans la liste *ListeNœud* et on sélectionne $(k-1)$ autres nœuds, "*EnsembleCandidat*", de la tête de la liste *ListeNœud* qui produisent un plus petit coût d'anonymisation. Ensuite, le nœud *NœudGerme* et les nœuds de *EnsembleCandidat*=$\{u_1, ..., u_m\}$ sont anonymisés à tour de rôle suivant la méthode d'anonymisation de deux voisinages citée ci-dessus. Chaque nœud u_i anonymisé sera marqué "*anonymisé*" et ajouté à *NœudGerme*. Durant la procédure d'anonymisation d'un groupe de nœuds, des changements peuvent avoir lieu à d'autres nœuds qui sont déjà marqués "*anonymisé*" dans d'autres groupes (par exemple, l'ajout d'une arête entre un nœud anonymisé et un nœud à être anonymisé en se basant sur la correspondance de nœuds). Pour préserver la condition de k-anonymité pour ces derniers nœuds, on applique les mêmes changements à chacun des autres $(k-1)$ nœuds ayant des voisinages isomorphes. Une fois que les k nœuds sont subis de chan-

gements, ils sont marqués "*non-anonymisé*" et réinsérés à la liste *ListeNœud*. L'algorithme d'anonymisation continue jusqu'à ce que tous les nœuds soient marqués "*anonymisé*".

4.3 k-automorphisme

Dans [21], Zou, Chen et Özsu ont adopté une hypothèse plus générale : l'adversaire peut savoir tout sous-graphe autour d'un certain nœud u. Si un tel sous-graphe pourrait être identifié dans le graphe publié avec une forte probabilité, l'utilisateur a un risque élevé de divulgation d'identité. Les auteurs ont cherché à construire un graphe $\tilde{G}$ à partir du graphe d'origine G tel que pour tout sous-graphe $X \subset G$, $\tilde{G}$ contient au moins k sous-graphes isomorphes à X.

Définition 4.1 (Graphe isomorphe et graphe automorphe)**.** *Soient deux graphes $G_1 = (V_1, E_1)$ et $G_2 = (V_2, E_2)$, G_1 est dit* graphe isomorphe *à G_2 si et seulement si, il existe une fonction bijective $f : V_1 \to V_2$ tel que pour toute arête $(u, v) \in E_1$, il existe une arête $(f(u), f(v)) \in E_2$. Si G_1 est isomorphe à lui-même sous une fonction f alors G_1 est dit* graphe automorphe *et f est dite fonction automorphe.*

Définition 4.2 (Graphe k-automorphe)**.** *Un graphe $G = (V, E)$ est k-*automorphe *si et seulement si 1) il existe $(k-1)$ fonctions automorphes $f_1, ..., f_{k-1}$ dans G, et 2) pour tout nœud u dans G, $f_i(u) \neq f_j(u)$ $(i \neq j)$.*

Si le graphe publié $\tilde{G}$ est un graphe k-automorphe, lorsque l'adversaire essaie de ré-identifier un nœud u à travers un sous-graphe, il sera toujours au moins k différents sous-graphes dans $\tilde{G}$ qui correspondent à sa requête sous-graphe. Pour générer un graphe k-automorphe, les auteurs de [21] ont proposé un algorithme qui pourrait être résumé en ces étapes :

1 Partitionner le graphe G en différents groupes de sous-graphes $\{U_i\}$, tel que chaque groupe U_i contient $k_i \geq k$ sous-graphes $\{P_{i1}, ..., P_{ik_i}\}$ où chaque deux sous-graphes ne partagent aucun nœud ni arête.

2 Pour chaque U_i, rendre $P_{ij} \in U_i$ isomorphes entre-eux en ajoutant des arêtes. Ainsi, il existe une fonction $f^i_{s,t}(.)$ en vertu de laquelle P_{is} est isomorphe à P_{it}.

3 Pour chaque arête (u, v) à travers deux sous-graphes, c'est-à-dire $u \in P_{ij}$ et $v \in P_{st}$ $(P_{ij} \neq P_{st})$, ajouter une arête $\left(f^{(i)}_{j,\pi_j(r)}(u), f^{(s)}_{t,\pi_t(r)}(v)\right)$, où $\pi_j(r) = (j + r) \mod k$, $r = 1, ..., k-1$.

Après la modification, pour chaque nœud $u \in P_{ij}$, on définit $f_r(.)$ comme $f_r(u) = f^{(i)}_{j,\pi_j(r)}(u), r = 1, ..., k-1$. Ainsi, $f_r(u), r = 1, ..., k-1$ sont des fonctions automorphes dans $\tilde{G}$, et pour tout $s \neq t$, $f_s(u) \neq f_t(u)$, garantissant ainsi le k-automorphisme.

Pour mieux préserver l'utilité du graphe publié, cet algorithme devrait introduire le minimum nombre de faux arêtes, ce qui implique que les sous-graphes au sein d'un groupe U_i devraient être très similaires entre-eux (de sorte que l'étape 2 introduit seulement un petit nombre d'arêtes), et il y a quelques arêtes à travers des sous-graphes différents (de sorte que l'étape 3 n'ajouterait pas de nombreuses arêtes). Cela dépend de la façon dont le graphe est partitionné. Si G est divisé en petit nombre de sous-graphes, il y aura moins d'arêtes à ajouter à travers les sous-graphes. Cependant, moins de sous-graphes

implique que la taille de chaque sous-graphe est grande, et ainsi plus d'arêtes au sein de chaque sous-graphe doivent être ajoutés à l'étape 2. Les auteurs ont prouvé que trouver la solution optimale est NP-complet, mais ils ont proposé une heuristique gloutonne pour atteindre l'objectif [21].

4.4 Généralisation d'un graphe social

Pour préserver la vie privée des individus présents dans un graphe social, plusieurs techniques d'anonymisation modifient la structure du graphe social en ajoutant et/ou supprimant des nœuds et/ou des arêtes. Attaquant le problème sous un angle différent de ces approches, la technique de généralisation d'un graphe social consiste à regrouper plusieurs nœuds en une même partition appelée *super-nœud* ou *cluster*, et regrouper plusieurs arêtes en une *super-arête*. Nous allons présenter deux algorithmes d'anonymisation d'un graphe social par la technique de généralisation, *GraphGen* et *SaNGreeA*.

4.4.1 Algorithme GraphGen

La technique d'anonymisation d'un graphe social en le généralisant, proposée dans [22] par Hay, Miklau, Jensen, Towsley, et Weis, consiste à partitionner les nœuds d'un graphe en un ensemble de super-nœuds, puis publier le nombre de nœuds dans chaque partition ainsi que la densité d'arêtes existantes dans et entre les partitions. Si chaque super-nœud contient au moins k nœuds on dit que le graphe généralisé satisfait la condition de k-anonymité. On note cet algorithme de généralisation d'un graphe social par GraphGen, qui prend en entrée un graphe social naïvement anonymisé $G = (V, E)$, et le paramètre de protection de la vie privée k, et rend en sortie un graphe généralisé $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ qui satisfait la condition de k-anonymité et qui maximise la vraisemblance $1/(|[\![\mathcal{G}]\!]|)$, tel que

$$|[\![\mathcal{G}]\!]| = \prod_{X \in \mathcal{V}} \binom{\frac{1}{2}|X|(|X|-1)}{m_X} \prod_{X \in \mathcal{V}} \binom{|X||Y|}{m_{X,Y}},$$

et est le nombre de graphes, dont ses nœuds appartiennent à V, consistants avec $\mathcal{G}$, et $m_{X,Y}$ est le nombre d'arêtes entre les nœuds, du graphe d'origine, résumés par les super-nœuds X et Y. Leur algorithme recherche le partitionnement approximatif optimal en se basant sur le principe de la technique "*recuit simulé*" [23]. Pour trouver le graphe généralisé qui maximise la fonction de vraisemblance, l'algorithme GraphGen commence avec un graphe généralisé contenant une seule partition (super-nœud) regroupant tous les nœuds de V, puis il propose un changement d'état, en divisant une partition, fusionnant deux partitions, ou déplaçant un nœud d'une partition à une autre. La proposition du changement d'état d'un graphe généralisé à un autre nouveau graphe généralisé est évaluée en se basant sur la différence de vraisemblance résultant. La proposition est toujours acceptée si elle améliore la vraisemblance et acceptée avec une certaine probabilité si elle diminue la vraisemblance. La probabilité d'acceptation commence avec une grande valeur et diminue rapidement jusqu'à ce qu'un changement est accepté seulement s'il améliore la vraisemblance. La recherche est terminée quand il y aura moins de 0.02% de propositions acceptées (cette constante est souvent utilisée dans la conception des algorithmes de type "*recuit simulé*" [23]).

4.4.2 Algorithme SaNGreeA

On considère un graphe social $G = (N, E)$ où N est l'ensemble des nœuds et E l'ensemble des arêtes. Chaque nœud est décrit par un ensemble d'attributs. Ces attributs peuvent être classés comme suit :

- $I_1, I_2, ..., I_m$ ensemble d'attributs *identifiants* tels que *Nom* et *NSS*.
- $Q_1, Q_2, ..., Q_q$ ensemble d'attributs *quasi-identifiants* tels que *CodeZip*, *Sexe* et *Age* qui peuvent être connus par un adversaire.
- $S_1, S_2, ..., S_r$ ensemble d'attributs *sensibles* tels que *Maladie* et *Revenu* qui sont supposés à être inconnus par un adversaire.

Dans la procédure d'anonymisation d'un graphe social, il existe deux aspects à considérer. Les données associées aux nœuds d'un graphe social (attributs identifiants, quasi-identifiants et sensibles) et l'information structurelle que porte ce graphe concernant les relations entre les nœuds. Le but de l'anonymisation d'un graphe social est de rendre chaque groupe de k nœuds similaires et donc indistinguables, par généralisation. Ceci se fait en ces trois opérations. On remplace la valeur actuelle d'un attribut quasi-identifiant par une valeur moins spécifique. On attache à chaque super-nœud une information structurelle $(|cl|, |E_{cl}|)$ où $|cl|$ est le nombre de nœuds regroupés dans cl et $|E_{cl}|$ est le nombre d'arêtes existant entre ces nœuds. Les arêtes entre les nœuds généralisés par deux super-nœuds différents sont généralisées par une super-arête portant le nombre d'arêtes généralisées.

Définition 4.3 (Perte d'information de généralisation)**.** *Soit cl un super-nœud et $QI = (N_1, N_2, ..., N_s, C_1, C_2, ..., C_t)$ l'ensemble des attributs quasi-identifiants numériques et catégorielles.* la perte d'information causée par la généralisation *des attributs quasi-identifiants du super-nœud cl est :*

$$GIL(cl) = |cl|. \left(\sum_{j=1}^{s} \frac{size(gen(cl)[N_j]))}{size(min_{x \in N}(x[N_j]), max_{x \in N}(x[N_j]))} + \sum_{j=1}^{t} \frac{height(\Lambda(gen(cl)[C_j]))}{height(H_{C_j})} \right),$$

où :

- $gen(cl)[N_j]$ = l'intervalle $[min\left\{x^1[N_j], ..., x^u[N_j]\right\}, max\left\{x^1[N_j], ..., x^u[N_j]\right\}]$
- $size([i_1, i_2])$ = la taille de l'intervalle $[i_1, i_2]$, c'est-à-dire $(i_2 - i_1)$
- $gen(cl)[C_j]$ = le plus petit ancêtre commun de $\left\{x^1[C_j], ..., x^u[C_j]\right\}$ dans la hiérarchie d'étiquettes.
- $\Lambda(w)$ est la sous-hiérarchie d'étiquettes de racine w.
- $height(H_{C_j})$ dénote la hauteur de la hiérarchie d'arbre H_{C_j}

Définition 4.4 (Perte d'information de généralisation normalisée)**.** *Il s'agit de la somme des pertes d'information de généralisation pour chaque super-nœud dans l'ensemble de super-nœuds S :*

$NGIL(G, S) = \frac{GIL(G,S)}{n.(s+t)}$. *(où n est le nombre de nœuds et $(s+t)$ est le nombre d'attributs quasi-identifiants).*

Définition 4.5 (Distance entre deux nœuds)**.** *Soit $B_i = (b_1^i, ..., b_n^i)$ le vecteur de voisinage d'un nœud x^i, où $b_j^i = 1$ s'il existe une arête (x^i, x^j), et O sinon. La* distance entre deux nœuds x^i *et x^j est définit de la manière suivante :*

$dist(x^i, x^j) = \frac{|\left\{l | l=1..n \wedge l \neq i,j; b_l^i \neq b_l^j\right\}|}{n-2}$.

Définition 4.6 (Distance entre un nœud et un super-nœud)**.** *La* distance entre un nœud x et un super-nœud cl *est la distance moyenne entre x et chaque nœud dans cl.*

L'algorithme *SaNGreeA (Social Network Greedy Anonymization)* est un algorithme d'anonymisation d'un graphe social en le généralisant par groupes de nœuds ou *super-nœuds* (*clusters* en anglais), les nœuds d'un même groupe doivent être aussi similaires que possible. Cette similarité est calculée, à la fois, en termes des valeurs des attributs quasi-identifiants de chaque nœud, et en termes de sa structure de voisinage [24]. Cette approche gloutonne tente de minimiser la perte d'information causée par la généralisation des valeurs des attributs et la perte d'information structurelle pour le graphe social généré qui satisfait la condition de k-anonymité. Le pseudo-code de l'algorithme peut-être donné comme suit (sa complexité temporelle est $O(n^2)$) :

Algorithme 1: SaNGreeA

Entrées : $G = (V, E)$, un graphe à généraliser
k, cardinalité minimale des clusters
α, β, deux paramètres définis par l'utilisateur
Sortie : $S = \{cl_1, cl_2, ..., cl_v\}$, un ensemble de clusters assurant le k-anoymité

$S \leftarrow \emptyset$
$i \leftarrow 1$
répéter
  $x_{seed} \leftarrow$ un nœud de V de degré maximal
  $cl_i \leftarrow x_{seed}$
  $V \leftarrow V - x_{seed}$
  répéter
    $x^* = \text{argmin}_{x \in V}(\alpha.NGIL(G, S) + \beta.dist(x, cl_i)$ $cl_i = cl_i \cup x^*$
    $V \leftarrow V - x^*$
  jusqu'à *(cl_i a k éléments) ou ($V = \emptyset$)*;
  si $|cl_i| < k$ **alors**
    DisperseCluster(S, cl_i)
  sinon
    $S = S \cup cl_i$
    $i \leftarrow i + 1$
  finsi
jusqu'à $V = \emptyset$;
Retourner S

Algorithme 2: DisperseCluster

Entrées : $S = \{cl_1, cl_2, ..., cl_v\}$, un ensemble de clusters
cl, un cluster
Sortie : $S = \{cl_1, cl_2, ..., cl_v\}$, un ensemble de clusters

pour *tout* $x \in cl$ **faire**
  bestCluster = null
  infoLoss $\leftarrow \infty$
  pour *tout* $cl_j \in S$ **faire**
    si $\alpha.NGIL(G_1,S_1) + \beta.dist(x,cl_j) < infoLoss$ **alors**
      infoLoss = α.NGIL(G_1,S_1) + β.dist(x,cl_j)
      bestCluster $\leftarrow cl_j$
    finsi
  fin
  bestCluster $\leftarrow$ bestCluster $\cup\{x\}$
fin

4.5 Anonymisation par perturbation aléatoire

Le brouillage par perturbation aléatoire consiste en deux étapes – suppression d'arêtes suivie par ajout d'arêtes. Une manière de réaliser cela est que le propriétaire des données sélectionne un entier h, puis, il choisit aléatoirement un sous ensemble de h arêtes et il les supprime. Dans la deuxième étape le propriétaire des données choisit aléatoirement h paires de nœuds non connectés et il ajoute des arêtes entre eux. Nous considérons des perturbations aléatoires qui utilisent une séquence de tirages de Bernoulli. Durant la première étape le propriétaire des données choisit une probabilité $p \in [0, 1]$. Puis, pour chaque arête e, on la garde avec une probabilité p. Lors de la deuxième étape le propriétaire des données doit choisir une autre probabilité q, puis, il ajoute une arête entre deux nœuds non encore connectés avec une probabilité q. Pour garantir que le nombre d'arêtes attendu dans le graphe perturbé égal au nombre d'arêtes existant dans le graphe initial, la probabilité q devrait être choisie de telle sorte que : $pm + q.\left(\binom{n}{2} - m\right) = m$.

Dans [25], les auteurs ont défini deux notions de protection de la vie privée. La première protège contre des adversaires qui essayent de réidentifier un individu spécifique dans le graphe perturbé, alors que la deuxième protège contre des adversaires qui ne ciblent pas une personne spécifique.

4.5.1 k-brouillage

On suppose que l'adversaire sache la méthode de perturbation et la valeur de p. Le but de l'adversaire est localiser l'image du nœud $v \in V$ dans l'ensemble des nœuds du graphe perturbé U. Pour cela, l'adversaire peut associer une probabilité, $X_v(u)$, à chaque nœud u dans U comme étant l'image recherchée du v dans V.

Définition 4.7 (k-brouillage)**.** *Soit $v \in V$ et soit $p(v) = (p_1, ..., p_n)$ dénote la distribution de probabilité X_v, on définit l'*entropie *par :*

$$H(p(v)) = \sum_i p_i \log\left(\frac{1}{p_i}\right).$$

Un graphe perturbé satisfait k-brouillage *si pour chaque nœud v dans V, l'entropie de la variable aléatoire X_v sur U est au moins* $\log k$.

Les auteurs de [25] prétendent que la condition de k-candidat anonymité, $X_v(u) \leq \frac{1}{k}$, ne mesure pas correctement la quantité d'incertitude que l'adversaire possède par rapport à l'identification correcte d'un nœud cible. Par exemple, la définition de k-candidat anonymité ne différencie pas entre les deux situations suivantes :

(1) $X_v(u_1) = X_v(u_2) = \frac{1}{2}$, et
$X_v(u_1) = 0$ pour tout $3 \leq i \leq n$;

(2) $X_v(u_1) = \frac{1}{2}$,
$X_v(u_i) = \frac{1}{2t}$ pour tout $2 \leq i \leq t + 1$, et
$X_v(u_i) = 0$ pour tout $t + 2 \leq i \leq n$.

Les deux cas respectent 2-candidat anonymité (la probabilité maximale dans les deux cas est $\frac{1}{2}$). Cependant, il est évident que dans le premier cas, où il y a deux suspects, la quantité d'incertitude est plus petite que celle dans le deuxième cas, où il y a $t + 1$ suspects.

Selon [25], la façon correcte pour mesurer l'incertitude est l'entropie. En appliquant la définition de k-brouillage aux deux cas, on trouve que le le premier cas satisfait 2-brouillage, alors que le deuxième cas satisfait $(2\sqrt{t})$-brouillage.

4.5.2 k- préimage brouillage

Cette notion de protection de la vie privé lutte contre un adversaire intéressé par la ré-identification de n'importe quel individu présent dans le graphe rendu public.

Définition 4.8 (k-préimage brouillage)**.** *Soit $G = (V, E)$ et $G_p = (U, E_p)$ un graphe social et son graphe perturbé. Pour chaque $u \in U$ soit X_u dénote la variable aléatoire correspondante définie sur V, c'est-à-dire $X_u(v)$ est la probabilité que v est la préimage de u dans G. Le graphe perturbé G_p satisfait* k-préimage brouillage *si pour chaque nœud $u \in U$, l'entropie de la variable aléatoire X_u sur V est au moins* $\log k$.

4.5.3 k-degré brouillage

Soit $v \in V$ un nœud cible dans V, supposant que l'adversaire sait son degré $d(v) = a$. Soit $u \in U$ un nœud candidat dans U dont le degré $d(u) = b$. La probabilité, $f(v, u)$, qu'un nœud avec un degré $d(v)$ a été convertie à un nœud avec un degré $d(u)$, est égale à la probabilité conditionnelle suivante : $f(u, v) = Pr(d(u) = b | d(v) = a, v \mapsto u)$, c'est-à-dire, soit v un nœud de degré a et son image dans G_p est u, $f(u, v)$ est la probabilité que le degré du nœud u est b. Dans le cas de l'approche consistant à supprimer des arêtes de manière aléatoire (*randomization by sparsification*, en anglais), $b \sim B(a, p)$ tel que $B(a, p)$ est la distribution Binomiale sur a expérimentations et de probabilité de succès p. Dans le cas de l'approche de perturbation aléatoire, $b \sim B(a, p) + B(n - 1 - a, q(p))$.

4.6 Étude critique des approches d'anonymisation

La plupart des méthodes d'anonymisation d'un graphe social citées ci-dessus pré-supposent un modèle d'adversaire et conçoivent des algorithmes d'anonymisation d'un graphe social basés sur cette hypothèse. Ainsi, Liu et Terzi [19] supposent un adversaire possédant le degré d'un nœud comme connaissance auxiliaire. Cependant, il n'existe aucune motivation claire pour justifier cette restriction, car un adversaire pourrait avoir tout un sous-graphe autour d'un nœud cible et pas seulement son degré. Zhou et Pei [20] considèrent une supposition plus forte avec la prise en compte des attributs modélisée par une hiérarchie d'étiquettes. Un adversaire peut avoir le voisinage d'un nœud comme connaissance auxiliaire. Cependant, les expériences sont réalisées sur un graphe non orienté avec un degré moyen de 4 (un ordre de grandeur plus faible que dans de véritables réseaux sociaux) et en appliquant leur algorithme, il faudra accroître le nombre d'arêtes de 6%. Le nombre d'arêtes à ajouter et le temps de calcul (le temps d'exécution de l'algorithme) sont susceptibles d'augmenter fortement avec le degré moyen des nœuds.

La technique de généralisation d'un graphe social demande à ce que les k nœuds d'un même cluster soient équivalents, c'est-à-dire qu'il doit exister des automorphismes dans le graphe publié qui établissent une correspondance entre chacun des k nœuds. C'est une exigence structurelle très forte, qui n'est atteinte que contre des adversaires strictement restreints : dans un modèle,

l'attaquant ne dispose que d'informations sur les séquences des degrés autour de son nœud cible ; dans un autre, la connaissance partielle de la structure dans le voisinage de la cible. Dans cette technique d'anonymisation, nous avons présenté deux algorithmes, *GraphGen* et *SaNGreeA*. L'algorithme *SaNGreeA* englobe *GraphGen* par le fait qu'il considère les attributs des nœuds alors que *GraphGen* ne prend en considération que la structure du graphe social. Généralement, le graphe social anonymisé par la technique de généralisation n'atteint pas un niveau acceptable en termes de mesures d'utilité.

Les mêmes inconvénients de la technique de généralisation peuvent être rapporter en parlant de l'algorithme de k-automorphisme. La différence entre les algorithmes de généralisation et l'algorithme de k-automorphisme est que les algorithmes de généralisation entre dans la catégorie des méthodes basées sur l'approche de clustering des nœuds alors que le k-automorphisme entre dans la catégorie des méthodes basées sur la modification des arêtes.

L'approche basée sur la perturbation aléatoire du graphe social considère un adversaire possédant le degré d'un nœud comme connaissance auxiliaire. Ainsi, ce que nous avons dit à propos de cette supposition peut être redit. De plus, la technique de perturbation du graphe peut modifier considérablement les propriétés structurelles du graphe, et donc son utilité potentielle.

La condition de k-anonymité est souvent rencontrée dans la plupart des approches d'anonymisation d'un graphe social. Cependant, cette notion souffre de quelques lacunes :

1. La première limite est qu'elle ne garantie pas contre la divulgation d'attributs : imaginons le cas où tous les nœuds d'un même groupe de k nœuds partagent la même valeur pour un attribut sensible, même si l'adversaire ne peut pas localiser son nœud cible u, il connait de façon certaine la valeur de son attribut sensible.
2. La deuxième limite pourrait être une question de faible utilité. En effet, l'utilité d'un graphe social diminue inversement avec le nombre k de la condition de k-anonymité.

Pour pallier la première limite de cette approche, plusieurs modèles et notions ont été proposés tels que la l-diversité [26] et la p-sensitive [27]. Nous proposons dans notre travail de recherche de nous occuper de la deuxième défaillance, ainsi de concevoir un algorithme d'anonymisation orienté par l'utilité tout en préservant la vie privée des individus présents dans le graphe social publié.

5 Généralisation guidée par l'utilité

Dans ce travail de recherche, nous visons à améliorer la technique d'anonymisation d'un graphe social par généralisation. Bien que cette technique d'anonymisation produise un niveau élevé (proportionnel au nombre k de la condition de k-anonymité) de garantie en termes de protection de la vie privée des individus présents dans un graphe social, l'utilité d'un tel graphe social publié n'atteint souvent pas un niveau acceptable au niveau pratique [2]. Dans [22] (voir aussi la version journal [28]), Hay, Miklau, Jensen, Towsley, et Weis proposent un algorithme de généralisation d'un graphe social. Son critère pour accepter ou ne pas accepter un partitionnement candidat est basé sur le nombre de graphes candidats pouvant être échantillonnés à partir d'un graphe généralisé (le partitionnement candidat). Cet algorithme cherche à minimiser

le nombre de graphes candidats pouvant avoir conduit au graphe généralisé. Cependant, cet algorithme ne garantit rien par rapport à l'utilité résultante de la généralisation en termes de préservation des différentes propriétés structurelles.

Comme en général, une analyse spécifique d'un graphe social s'intéresse à une propriété structurelle bien précise du graphe publié, nous proposons de concevoir un algorithme général d'anonymisation d'un graphe social par la technique de généralisation qui est guidé par le niveau d'utilité qu'on souhaite préserver dans le graphe publié. Ainsi, notre algorithme fournit un graphe social généralisé préservant le mieux possible la propriété structurelle considérée. Les analystes d'un graphe social doivent cibler le graphe social publié qui préserve le mieux la (ou les) propriété(s) structurelle(s) qu'ils souhaitent étudier.

Dans l'étude d'un graphe généralisé, l'analyste peut échantillonner un graphe candidat à partir du graphe généralisé rendu public, ensuite il effectue une analyse standard sur le graphe échantillonné. À ce stade, il est possible que l'analyse de deux graphes candidats différents puissent donner des résultats très divergents. Afin d'éviter cela et de générer des résultats d'analyse plus proches quelque soit le graphe échantillonné, notre algorithme doit l'assurer en calculant des bornes possibles sur la valeur de cette propriété structurelle qui soient les plus précises possibles.

Définition 5.1 (Graphe généralisé)**.** *On appelle* graphe généralisé *un graphe* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *muni de deux fonctions d'étiquetage :*

$$nodeLabels : \quad \mathcal{V} \quad \rightarrow \quad \mathbb{N} \times \mathbb{N}$$
$$edgeLabels : \quad \mathcal{E} \quad \rightarrow \quad \mathbb{N}$$

Définition 5.2 (Généralisation d'un graphe)**.** *Soit* $G = (V, E)$ *un graphe, une* généralisation *de* G *est un graphe généralisé* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *avec une bijection* $cluster_G : \mathcal{V} \rightarrow 2^V$ *telle que :*

- *si* $nodeLabels(u) = (m_1, m_2)$ *alors* $m_1 = |X|$ *et* $m_2 = |\{(x, x') \in E \mid x, x' \in X\}| = |E \cap (X \times X)|$*, où* $X = cluster_G(u)$*.*
- *si* $edgeLabels(u, u') = m$ *alors* $m = \frac{|E \cap \{(X \times Y) \cup (Y \times X)\}|}{2}$*, où* $X = cluster_G(u)$ *et* $Y = cluster_G(u')$*.*

et $\{cluster_G(u)\}_{u \in \mathcal{V}}$ *est une partition de* V*.*

Définition 5.3 (Graphe candidat)**.** *Soit* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *un graphe généralisé, on appelle* graphe candidat *tout graphe* $G = (V, E)$ *dont* $\mathcal{G}$ *est une généralisation et on le note* $G \trianglelefteq \mathcal{G}$*. On note l'ensemble de graphes candidats d'un graphe généralisé* $\mathcal{G}$ *par* $[\![\mathcal{G}]\!] = \{G \mid G \trianglelefteq \mathcal{G}\}$*.*

Par exemple, dans la Figure 6, on peut voir un graphe généralisé (à gauche) situé côte-à-côte avec un graphe candidat.

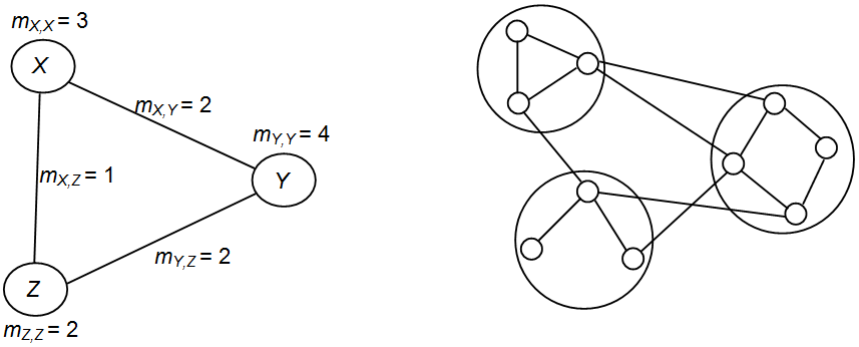


Figure 6 – Un graphe généralisé côte-à-côte avec un graphe candidat.

Soit $m_{X,X}$ représente le nombre d'arêtes entre les nœuds d'un cluster X, et $m_{X,Y}$ est le nombre d'arêtes reliant les nœuds d'un cluster X avec les nœuds d'un autre cluster Y. Nous rappelons que le nombre de graphes candidats est donné par la formule suivante :

$$|[\![\mathcal{G}]\!]| = \prod_{X\in\mathcal{V}} \binom{\frac{1}{2}|X|(|X|-1)}{m_{X,X}} \prod_{X\in\mathcal{V}} \binom{|X||Y|}{m_{X,Y}}. \tag{1}$$

Définition 5.4 (Mesure d'utilité)**.** *Soit $\mathbb{G}$ un ensemble de graphes. On appelle* mesure d'utilité *sur $\mathbb{G}$, une fonction $U : \mathbb{G} \longrightarrow \mathbb{R}$.*

Par exemple, le degré d'un nœud dans un graphe est une mesure d'utilité.

Définition 5.5 (Bornes d'une mesure d'utilité)**.** *Soit $\mathbf{G} \subseteq \mathbb{G}$, un ensemble fini de graphes. On appelle* bornes sur une mesure d'utilité U *(ou simplement* bornes d'utilité*) d'un ensemble fini de graphes $\mathbf{G}$, une paire de valeurs $(U_{\min}, U_{\max}) \in \mathbb{R}^2$ tel que $\forall\, G \in \mathbf{G}$:*

$$U_{\min} \leqslant U(G) \leqslant U_{\max}.$$

L'algorithme suivant calcule les bornes d'utilité ($U_{\min}$ et $U_{\max}$) sur une mesure d'utilité U d'un graphe généralisé $\mathcal{G}$:

Algorithme 3: Bornes$_U(\mathcal{G})$

Entrées : $\{G_1, G_2, ..., G_N\}$, un ensemble fini de graphes
U, une mesure d'utilité

Sortie : des bornes d'utilité de l'ensemble d'entrée

```
1 U_min ← U(G_1) ; U_max ← U(G_1)
2 pour i de 2 à N faire
3 |   util ← U(G_i)
4 |   U_min ← min(U_min, util)
5 |   U_max ← max(U_max, util)
6 fin
7 Retourner (U_min, U_max)
```

Par exemple, à partir du graphe généralisé $\mathcal{G}$ de la Figure 6 la paire de valeurs $(1, 5)$ sont des bornes d'utilité pour le degré d'un nœud dans un graphe $G \trianglelefteq \mathcal{G}$. Étant donnée une utilité $U : \mathbb{G} \longrightarrow \mathbb{R}$, et soit $\mathbf{G} \subseteq \mathbb{G}$ tel que $|\mathbf{G}| = N$. On note $util(n) \in \mathbb{R}$, une borne de calcul de $U(G)$ pour $G \in \mathbf{G}$ et $|G| = n$. Le coût de calcul de Bornes$_U(\mathbf{G})$ est en $O(N.\max_i util(|G_i|))$.

Remarque 5.1 (Complexité de Bornes_U)**.** *Soit $\mathcal{G}$ un graphe généralisé. Alors $\forall G, G' \in [\![\mathcal{G}]\!]$, $|G| = |G'|$; on note cette valeur $CardElem(\mathcal{G})$. Ainsi, $Bornes_U(\mathcal{G})$ se calcule en $O\big(|[\![\mathcal{G}]\!]|.util(CardElem(\mathcal{G}))\big)$, tel que $|[\![\mathcal{G}]\!]|$ est donné par l'équation* (1)*.*

Définition 5.6 (Précision de bornes d'utilité)**.** *La* précision *d'une paire de bornes d'utilité $(U_{\min}, U_{\max})$ est donnée par la différence entre ces deux valeurs :*

$$\text{Precision}(U_{\min}, U_{\max}) = U_{\max} - U_{\min}.$$

5.1 Propriétés structurelles et utilité

Dans le contexte de notre étude, nous nous intéressons aux mesures d'utilité reflétant des propriétés structurelles d'un graphe social $G = (V, E)$ que nous supposons désormais connexe. Nous considérons donc des mesures qui permettent de quantifier différents aspects structurels et des propriétés globales d'un graphe telles que la connectivité et la centralité. Ces caractéristiques sont généralement utilisées par des analystes, par exemple pour étudier l'influence, le pouvoir, l'engagement et les formes de communication dans les réseaux sociaux. Ci-après, nous définissons quelques propriétés structurelles et pour chacune nous donnons à titre d'exemple un type d'analyse qui se base sur cette propriété structurelle :

- L'**excentricité** d'un nœud est sa distance (on définit la distance entre deux nœuds comme le nombre de liens dans un plus court chemin entre ces deux nœuds) maximale à tous les autres nœuds, et est définie par :

  $$\varepsilon(v) = \max\{d(v, w) \mid w \in V\}.$$

  Notons que pour les graphes connexes $\varepsilon(v) < +\infty, \forall v \in V$.
- Le **rayon** d'un graphe est l'excentricité minimale de ses nœuds, c'est-à-dire la plus petite distance à laquelle puisse se trouver un nœud de tous les autres, et défini par :

  $$\mathit{rayon}(G) = \min\{\varepsilon(v) \mid v \in V\}.$$

  Le *centre* d'un graphe est formé de l'ensemble de ses nœuds d'excentricité minimale. Comme exemple d'utilisation pratique du rayon d'un graphe social dans l'épidémiologie, imaginons qu'une maladie se propage d'un individu à ces voisins au bout d'un jour. Soit J le jour où un individu est infecté, au jour $(J + n)$ si $n < \mathit{rayon}(G)$ alors on est sûr qu'il reste au moins $(\mathit{rayon}(G) - n)$ individus non encore infectés et on peut donc encore réagir.
- Le **diamètre** d'un graphe est l'excentricité maximale de tous les nœuds du graphe, et est défini par :

  $$\mathit{diametre}(G) = \max\{\varepsilon(v) \mid v \in V\}.$$

  Dans ce contexte, nous citons le problème du petit monde (*small world problem* en anglais) où les expériences sondaient la distribution des longueurs de chemins dans un réseau de connaissance en demandant aux participants de passer une lettre à un de leurs premiers noms de connaissances dans une tentative de se rendre à un individu cible (le résutat obtenu était, il suffit, en moyenne, six nœuds intermédiaires pour faire

arriver une lettre à un individu cible) [29]. Une telle expérience permet d'établir la théorie de "*six degrés de séparation*" [4].

- La **densité** d'un graphe G est la proportion des arêtes d'un graphe relativement au total de liens possibles, et est définie par :

$$Densite(G) = \frac{2|E|}{n(n-1)},$$

tel que n est le nombre de nœuds du graphe G, et $|E|$ est le nombre d'arêtes. Une densité plus élevée est le reflet d'un grand nombre de contacts moyen dans un graphe social.

- La **centralité de degré d'un nœud** v est le nombre de nœuds voisins à ce nœud. Pour comparer entre différents graphes, on utilise la centralité de degré normalisée :

$$C_D(v) = \frac{d}{n-1},$$

où d est le degré d'un nœud v, et n est le nombre de nœuds du graphe. Une centralité de degré plus élevée peut signifier qu'un individu a plus de connexions, indiquant ainsi un plus grand cercle social dans son réseau social.

- La **centralité de degré d'un graphe** est définie par :

$$C_D(G) = \frac{\sum_{i=1}^{n} [C_D(v^*) - C_D(i)]}{n-2} = \frac{\sum_{i=1}^{n} [deg(v^*) - deg(i)]}{(n-1)(n-2)},$$

où v^* est un nœud ayant une plus grande valeur de centralité de degré parmi tous les nœuds $\{1, ..., n\}$ du graphe G.

- La **centralité de proximité d'un nœud** v est l'inverse de la distance moyenne du nœud v à tous les autres nœuds :

$$C_P(v) = \frac{n-1}{\sum_{v' \in V \setminus \{v\}} g(v, v')},$$

tel que $g(v, v')$ est la distance géodésique entre v et v' où v' est un des autres nœuds du graphe (la distance géodésique est la longueur d'un plus court chemin entre deux nœuds). La centralité de proximité mesure combien un nœud est proche des autres nœuds et ainsi, par exemple, une propagation rapide d'information, de maladie ou de rumeur peut être mesurée. Ainsi, si un nœud est connecté à tous les autres nœuds alors sa centralité de proximité est égale à 1 et ce nœud sera considéré comme le voisin le plus proche à tous les autres nœuds. Ainsi dans le cadre de marketing ciblé, il pourrait être intéressant de cibler ce type de nœud pour avoir ensuite un impact important sur le réseau.

- La **centralité de proximité d'un graphe** est définie par :

$$C_P(G) = \frac{\sum_{i=1}^{n} [C_P(v^*) - C_P(i)]}{(n-1)(n-2)/(2n-3)},$$

où v^* est un nœud ayant une plus grande valeur de centralité de proximité parmi tous les nœuds d'un graphe G.

4. http ://en.wikipedia.org/wiki/Six degrees of separation

- La **centralité d'intermédiarité d'un nœud** v est la proportion du nombre de plus courts chemins passants par v par rapport au nombre total des plus courts chemins existants dans le graphe :

$$C_I(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

où σ_{st} est le nombre total de plus courts chemins entre les nœuds s et t, et $\sigma_{st}(v)$ est le nombre de plus courts chemins entre les nœuds s et t et qui passent par le nœud v. Plus la valeur de la centralité d'intermédiarité est élevée, plus ce nœud peut être important dans la perspective de la communication dans un réseau. Une autre expérience, considérant une communauté où une procédure de vaccination doit se dérouler, si un nœud v a une centralité d'intermédiarité élevée et qu'il est vacciné alors on coupe beaucoup de chemins de propagation de la maladie.

- La **centralité d'intermédiarité d'un graphe** est définie par :

$$C_I(G) = \frac{\sum_{i=1}^{n} [C_I(v^*) - C_I(i)]}{(n-1)},$$

où v^* est un nœud ayant la plus grande valeur de centralité d'intermédiarité parmi tous les nœuds d'un graphe G.

- La **centralité vecteur propre** est une mesure de l'importance d'un nœud en fonction de ses liens avec les nœuds importants. On considère le principal vecteur propre de la matrice d'adjacence représentant un graphe social. Ce vecteur propre est le seul vecteur propre de valeurs positives associé à la plus grande valeur propre de la matrice d'adjacence (le lecteur curieux pourra consulter [30] pour avoir la formulation et la démonstration du théorème de principal vecteur propre).
  L'algorithme PageRank calcule une variante de la mesure centralité vecteur propre [31].

- Le **cœfficient de clustering global** d'un graphe mesure le nombre de triangles transitifs existants proportionnel au nombre de triplets connectés :

$$C_G = \frac{3 * \Delta}{\Lambda},$$

tel que Δ est le nombre de triangles, et Λ est le nombre de triplets connectés. Dans le contexte de réseaux sociaux le cœfficient de clustering mesure la probabilité moyenne qu'un ami d'un ami est un ami.
Le **cœfficient de clustering local** d'un nœud v_i est défini comme :

$$C_i = \frac{|e_{jk}|}{k_i * (k_i - 1)},$$

où $|e_{jk}|$ est le nombre d'arêtes entre les paires de nœuds (v_j, v_k) voisins de v_i, et k_i est le nombre de nœuds voisins de v_i.

5.2 Généralisation et bornes de mesures d'utilité

Nous constatons que les algorithmes de généralisation d'un graphe social tel que celui proposé dans [22] ne donnent aucune garantie en termes de préservation des différentes mesures d'utilité. Nous proposons un algorithme dont le critère d'évaluation d'un partitionnement candidat est basé sur la précision des bornes d'utilité. Ainsi, en fixant une propriété structurelle (par exemple

le degré des nœuds, le diamètre d'un graphe, etc.), lors du partitionnement, notre algorithme calcule les bornes d'une mesure d'utilité donnée et décide d'accepter ou ne pas accepter un partitionnement candidat suivant le changement dans la précision (Définition 5.6) obtenue sur les bornes de la mesure d'utilité considérée.

Soient $G = (V, E)$ un graphe et $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ une généralisation de G. Dans la suite on confondra un nœud de $\mathcal{G}$ avec le cluster qui lui correspond. Pour calculer les bornes d'utilité du graphe $\mathcal{G}$, on procède comme suit. Pour chaque cluster $X \in \mathcal{V}$, on définit :

1. $U^l(X)$ comme étant la valeur locale de la mesure d'utilité U pour le sous-graphe induit par les nœuds de X.
2. $U^g(X)$ comme étant la valeur globale de la mesure d'utilité U pour le sous-graphe induit par les nœuds de X et ses voisins (c'est-à-dire, en considérant les arêtes entre clusters).

Nous détaillons ci-après des bornes d'utilité d'un graphe généralisé pour quelques propriétés structurelles.

5.2.1 Bornes pour le degré d'un nœud

Lemme 5.1 (Bornes sur le degré maximal et minimal)**.** *Soient $n \geq 1$ et $m \geq n-1$:*

1. $degre^l_{\max} = n - 1$
2. $degre^l_{\min} = \begin{cases} 1 & si\ m \leq \frac{(n-1)(n-2)}{2} + 1 \\ m - \frac{(n-1)(n-2)}{2} & sinon \end{cases}$

Alors pour tout graphe $G = (V, E)$ tel que $|V| = n$ et $|E| = m$, on a :

$$degre^l_{\min} \leq Degre(G) \leq degre^l_{\max}.$$

Démonstration. Étant donné n et m, on construit par récurrence sur (n, m) un graphe $G = (\{1, ..., n\}, E)$ tel que : $Degre(G) = degre^l_{\min}$.

- Cas de base $(1, 0)$: il n'y a qu'un seul graphe $G = (\{1\}, \emptyset)$ et $Degre(G) = 0$, or $degre^l_{\min} = 0$ et $degre^l_{\max} = 0$.
- Cas (n, m) : $n - 1 \leq m \leq \frac{n(n-1)}{2}$, sachant que la propriété est vraie pour tous (p, q) avec ($p < q$ et $q \leq m$) ou ($p = n$ et $q < m$).
  Soit $V' = \{1, ..., n-1\}$, par hypothèse d'induction sur $(n-1, \frac{(n-1)(n-2)}{2})$, construire $G' = (V', E')$ tel que $|E'| = \frac{(n-1)(n-2)}{2}$ et $1 \leq Degre_{\min}(G') \leq m - \frac{(n-1)(n-2)}{2}$. On ajoute $m - \frac{(n-1)(n-2)}{2}$ arêtes entre le nœud n et les nœuds de V'.
  Pour le degré maximal, cela est évident.

□

Soient X et Y deux super-nœuds (clusters). Ils définissent deux graphes $(X, m_{X,X})$ et $(Y, m_{Y,Y})$, on pose :

- $degre^g_{\min}(X) = degre^l_{\min}(X) + \sum_{Y \in \mathcal{V}}(1, m_{X,Y} - (|X| - 1)|Y|)$, c'est-à-dire :
  $degre^g_{\min}(X) = \max(1, m_{X,X} - \frac{(n-1)(n-2)}{2}) + \sum_{Y \in \mathcal{V}}(1, m_{X,Y} - (|X| - 1)|Y|)$.
- $degre^g_{\max}(X) = degre^l_{\max}(X) + \sum_{Y \in \mathcal{V}} \min(|Y|, m_{X,Y})$, c'est-à-dire :
  $degre^g_{\max}(X) = \min(|X| - 1, m_{X,X}) + \sum_{Y \in \mathcal{V}} \min(|Y|, m_{X,Y})$.

On peut calculer le degré minimal et maximal d'un graphe généralisé $\mathcal{G}$ comme suit. En posant, $degre_{\min}(\mathcal{G}) = \min_{X \in \mathcal{V}} degre^{g}_{\min}(X)$ et $degre_{\max}(\mathcal{G}) = \max_{X \in \mathcal{V}} degre^{g}_{\max}(X)$, on a :

$$Bornes_{degre}(\mathcal{G}) = (degre_{\min}(\mathcal{G}), degre_{\max}(\mathcal{G})).$$

5.2.2 Bornes sur le diamètre

Soit $\mathcal{G}$ un graphe généralisé et $G \in [\![\mathcal{G}]\!]$, l'intervalle de valeurs que peut prendre le diamètre de G est donné par le lemme suivant :

Lemme 5.2 (Bornes sur le diamètre)**.** *Soit $G = (V, E)$ un graphe qui se généralise en $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, le diamètre de G peut être borné comme suit :*

$diametre(\mathcal{G}) + \big(diametre(\mathcal{G}) + 1\big).\min_{X \in \mathcal{V}} diametre(X) \leq diametre(G) \leq diametre(\mathcal{G}) + \big(diametre(\mathcal{G}) + 1\big).\max_{X \in \mathcal{V}} diametre(X)$.

Soit $S = \{X_1, X_2, ...X_i\}$ une chaîne constituée de super-nœuds engendrés dans le calcul du diamètre d'un graphe généralisé $\mathcal{G}$, le diamètre d'un graphe candidat peut être borné de manière plus stricte comme suit :

$diametre(\mathcal{G}) + \sum_{X \in S} diametre(X) \leq diametre(G) \leq diametre(\mathcal{G}) + \sum_{X \in S} diametre(X)$.

Pour calculer le diamètre minimal que peut prendre un graphe, on procède à une construction d'un graphe sous forme étoile (cf. Figure 7). Un tel graphe a un diamètre égal à 2 si ce graphe n'est pas complet (cf. Figure 8 pour voir l'exemple d'un graphe complet), et 1 sinon.

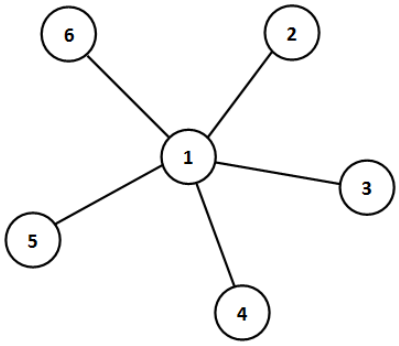


Figure 7 – Graphe sous forme étoile.

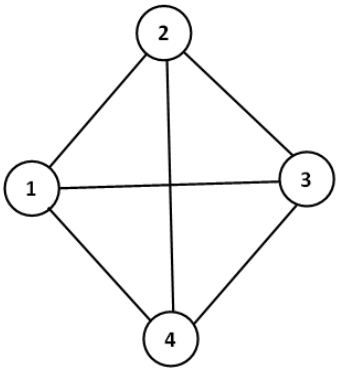


Figure 8 – Graphe complet.

Lemme 5.3 (Diamètre minimal)**.** *Soit un graphe $G = (V, E)$ tel que $|V| = n$ et $|E| = m$, le diamètre minimal que peut prendre G est :*

$$\text{Diametre}_{\min} = \begin{cases} 2 & si\ n-1 \leq m < \frac{n(n-1)}{2} \\ 1 & si\ m = \frac{n(n-1)}{2} \end{cases}$$

Démonstration. On construit un graphe connexe $G = (\{1, ..., n\}, E)$ comme suit :

On forme les $(n-1)$ arêtes $(1, i)\ \forall i \geq 2$. C'est une étoile. On ajoute les $m-(n-1)$ arêtes restantes, si ce graphe est complet alors son diamètre est de 1 sinon il est de 2. □

Pour calculer le diamètre maximal que peut prendre un cluster X (ou de manière général, un graphe), tel que $|X| = n$ (le nombre de nœuds dans un cluster X) et $|E_X| = m$ (le nombre d'arêtes dans un cluster X), on procède à la construction d'un graphe tel que, à chaque étape dans la construction, le graphe construit possède le diamètre maximal que peut prendre un graphe avec un tel nombre de nœuds et un tel nombre d'arêtes. Pour avoir un diamètre maximal, on commence à construire une chaîne, et s'il reste encore des arêtes à ajouter, on se permet de placer le maximum d'arêtes tel qu'à l'étape i on ne diminue le diamètre que de 1 par rapport à l'étape $(i-1)$. En se basant sur cette construction (voir Figure 9, une ligne en pointillés "Uni" est construite à l'étape 1, une ligne en pointillés "Tiret" est construite à l'étape 2, et une ligne en pointillés "Point carré" est construite à l'étape 3) on pourrait donner le diamètre maximal d'un graphe décrit par la conjecture suivante (à noter que $n > 5$) :

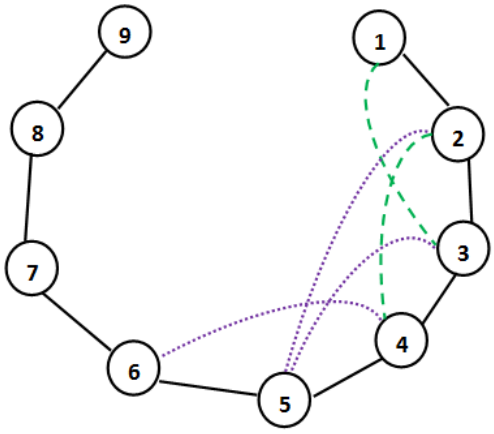


Figure 9 – Construction d'un graphe pour maximiser le diamètre.

Conjecture 5.1 (Bornes sur le diamètre maximal)**.** *Soit un graphe $G = (V, E)$ tel que $|V| = n$ et $|E| = m$, le diamètre maximal que peut prendre G est :*

$$\text{Diametre}_{\max} \leq \begin{cases} \frac{4n-m-5}{3} & si\ n \leq m \leq 2n \\ \frac{2n-5}{3} - \left(\frac{m-2n}{n-3}\right)^2 & si\ 2n < m \leq \frac{n(n-1)}{2} \end{cases}$$

5.2.3 Bornes sur le rayon

Soit $\mathcal{G}$ un graphe généralisé et $G \in [\![\mathcal{G}]\!]$, l'intervalle de valeurs que peut prendre le rayon de G se caractérise par le lemme suivant :

Lemme 5.4 (Bornes sur le rayon)**.** *Soit $G = (V, E)$ un graphe qui se généralise en $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, et soit $X \in \mathcal{V}$ un super-nœud, le rayon d'un graphe G peut être borné comme suit :*

$rayon(\mathcal{G}) + \big(rayon(\mathcal{G}) + 1\big).\min_{X\in\mathcal{V}} rayon(X) \leq rayon(G) \leq rayon(\mathcal{G}) + \big(rayon(\mathcal{G}) + 1\big).\max_{X\in\mathcal{V}} rayon(X)$.

Soit $S = \{X_1, X_2, ...X_i\}$ une chaîne constituée de super-nœuds engendrés dans le calcul du rayon d'un graphe généralisé $\mathcal{G}$, le rayon d'un graphe candidat peut être borné de manière plus stricte comme suit :

$rayon(\mathcal{G}) + \sum_{X\in S} rayon(X) \leq rayon(G) \leq rayon(\mathcal{G}) + \sum_{X\in S} rayon(X)$.

D'après l'inégalité suivante qui donne la relation entre le rayon et le diamètre d'un graphe G [32] :

$$rayon(G) \leqslant diametre(G) \leqslant 2.rayon(G).$$

$$\Rightarrow rayon(G) \leqslant diametre(G) \text{ et } \frac{1}{2}.diametre(G) \leqslant rayon(G).$$

Ainsi, le rayon d'un graphe candidat peut être calculé en fonction du diamètre comme suit :

$rayon(\mathcal{G}) + \frac{1}{2}.\sum_{X\in S} diametre(X) \leq rayon(\mathcal{G}) + \sum_{X\in S} rayon(X) \leq rayon(G) \leq rayon(\mathcal{G}) + \sum_{X\in S} rayon(X) \leq rayon(\mathcal{G}) + \sum_{X\in S} diametre(X)$.

5.2.4 Bornes sur d'autres propriétés structurelles

À cause de la difficulté à donner des bornes d'utilité pour des propriétés structurelles telles que la centralité intermidiarité, le cœfficient de clustering, ..., une alternative possible est de concevoir un algorithme qui étant donné un graphe généralisé $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, retourne la valeur minimale et la valeur maximale que peut prendre un graphe candidat. Un algorithme possible consisterait par exemple à énumèrer tous les graphes candidats, soit les éléments de l'ensemble $[\![\mathcal{G}]\!] = \{G = (V, E) \mid G \trianglelefteq \mathcal{G}\}$, et pour chaque graphe candidat à calculer son utilité. L'algorithme retourne la paire de bornes d'utilité $(U_{\min}, U_{\max})$ tel que décrit par l'Algorithme 3.

Comme la cardinalité de l'ensemble $[\![\mathcal{G}]\!]$ peut être assez importante (voir l'équation (1), on peut éviter de faire une énumération de tous les graphes candidats possibles en échantillonnant de manière aléatoire un certains nombre de graphes candidats. Cependant, l'application d'une telle solution peut conduire à une sous estimation de la précision des bornes d'utilité. Toute fois, on peut espérer que les échantillonnages aléatoires calculés par les analystes donnent lieu à des utilités à des bornes approximatives que nous aurons exhibées.

5.3 Algorithme de généralisation guidé par l'utilité

Nous décrivons maintenant notre algorithme de généralisation d'un graphe social guidé par l'utilité. Les entrées de cet algorithme sont, un graphe G, un paramètre k (pour la condition de k-anonymité), une mesure d'utilité U, qu'on souhaite préserver dans le graphe publié. La sortie est une généralisation de G dont les clusters sont de taille au moins k. La conception de l'algorithme de recherche est basée sur des techniques pour résoudre un problème lié à l'analyse des réseaux sociaux : *blockmodeling stochastique* [33] (*stochastic block-modeling*, en anglais). Le pseudo-code de cet algorithme est décrit par l'Algorithme 4.

Soumis à la condition de k-anonymité, qui exige que la taille de chaque cluster soit au moins k, et étant donnée une mesure d'utilité U qu'on souhaite

favoriser (voir Section 5.1), nous cherchons à générer un graphe généralisé basé sur des clusters de taille au moins k tel que la précision des bornes d'utilité soit la meilleur possible, de sorte que les valeurs d'utilité des graphes candidats soient les plus proches possibles.

L'algorithme recherche un bon partitionnement, proche de l'optimal, en se basant sur le principe de la technique "*recuit simulé*" [23]. Pour trouver le graphe généralisé qui minimise la précision des bornes d'utilité de la mesure U, l'algorithme commence avec un graphe généralisé ayant une seule partition (un seul super-nœud) regroupant tous les nœuds du graphe d'origine. Au moment de la recherche, chaque partitionnement *valide* (c'est-à-dire celui basé sur des clusters de taille au moins k) est un état (élément) dans l'espace de recherche, l'algorithme propose un changement d'état, soit en divisant une partition, soit en fusionnant deux partitions, soit en déplaçant un nœud vers une autre partition. La proposition d'un changement d'état d'un graphe généralisé $\mathcal{G}$ à un nouveau graphe généralisé $\mathcal{G}'$ est évaluée en se basant sur le changement dans la précision des bornes d'utilité de la propriété structurelle considérée. Une fonction $Bornes_U(\mathcal{G})$ retourne $U_{\min}$ la valeur minimale et $U_{\max}$ la valeur maximale que peut prendre n'importe quel graphe candidat du graphe généralisé $\mathcal{G}$ pour la propriété structurelle U, ces bornes sont calculées comme expliqué dans la Section 5.2. Le changement d'état proposé est toujours accepté si le graphe dérivé améliore la précision des bornes d'utilité, et il est accepté avec probabilité p s'il n'améliore pas la précision. La probabilité d'acceptation commence avec une valeur élevée p où $p = \mathrm{e}^{\Delta P/T}$ (tel que ΔP est la différence entre la précision d'utilité du partitionnement proposé et celle du partitionnement courant, et T est la température courante) et diminue lentement jusqu'à ce qu'un changement, en approchant de zéro, n'est accepté que s'il améliore la précision. On termine la recherche quand il y a moins de 0.02% de propositions sont acceptées (souvent cette constante est utilisée par des algorithmes de type "recuit simulé").

La fonction Successeurs$(\mathcal{G}, k)$, retourne un ensemble de graphes généralisés (valides) qui peuvent être dérivés à partir de $\mathcal{G}$ en faisant un seul changement local, tel que déplacement d'un nœud, partitionnement ou fusion d'un super-nœud dans $\mathcal{G}$.

- La procédure Partitionner(X,$\mathcal{G}$) partitionne un super-nœud X de cardinal au moins $2k$ de manière gloutonne : un nœud quelconque $u \in X$ est sélectionné pour être déplacé vers un nouveau super-nœud X'. On complète X' en ajoutant $(k-1)$ autre nœuds de X : on sélectionne dans X un nœud u' dont le déplacement vers X' optimise la précision des bornes d'utilité du graphe généralisé courant [5]. Cette approche gloutonne n'est cependant envisageable que pour un calcul de Bornes$_U$ efficace, par exemple parce qu'il suffit de ne considérer que le changement local du graphe généralisé, sans recalculer les nouvelles bornes "*from scratch*", comme c'est le cas pour le degré.
- La fonction DéplacerNœud$(u, X, Y, \mathcal{G})$ consiste à déplacer le nœud u du cluster X vers le cluster Y.
- La procédure FusionnerEtPartitionner$(X, Y, \mathcal{G})$ consiste à fusionner les clusters X et Y suivie de l'opération Partitionner$(X \cup Y, \mathcal{G})$.

5. Notons que ce graphe généralisé intermédiaire n'est pas forcément valide.

Algorithme 4: Algorithme de généralisation d'un graphe social

Entrées : $G = (V, E)$, un graphe à généraliser
k, une cardinalité des clusters
U, une mesure d'utilité
Sortie : $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, une généralisation de G dont les clusters sont de taille au moins k

$\mathcal{G} \leftarrow$ Initialiser(G) {mettre tous les nœuds dans un même cluster}
P $\leftarrow$ Précision($\mathcal{G}$)
$T \leftarrow T_0$ {Initialisation de la température}
répéter
  $\mathcal{S} \leftarrow$ Successeurs($\mathcal{G}, k$)
  $\mathcal{G}' \leftarrow \left(\arg\min_{\mathcal{G}' \in \mathcal{S}} \text{Précision}_U(\mathcal{G}')\right)$
  P$'$ $\leftarrow$ Précision($\mathcal{G}'$)
  ΔP $\leftarrow$ P - P$'$
  P $\leftarrow$ P$'$
  si $\Delta P > 0$ **alors**
    $\mathcal{G} \leftarrow \mathcal{G}'$
  sinon
    $\mathcal{G} \leftarrow \mathcal{G}'$ avec probabilité $e^{\Delta P/T}$
  finsi
  $T \leftarrow$ décrémenter(T)
jusqu'à $\mathcal{G}$ *est mis-à-jour moins de* 0.02% *dans les dernières itérations*;
Retourner $\mathcal{G}$

Algorithme 5: Successeurs, fonction qui retourne un ensemble de graphes généralisés

Entrées : $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, une généralisation d'un graphe G
k, un entier non nul $\leq |G|$
Sortie : un ensemble généralisation valides de G

$S \leftarrow \emptyset$ {ensemble des successeurs}
$u \leftarrow$ choisir un nœud aléatoire, soit X son cluster
si $|X| > 2k$ **alors**
  $\mathcal{G}' \leftarrow$ Partitionner($X, \mathcal{G}$)
  $S \leftarrow S \cup \mathcal{G}'$
finsi
pour *tout* Y *tel que* X, Y *sont voisins ou ont un voisin commun* **faire**
  si $|X| > k$ **alors**
    $\mathcal{G}' \leftarrow$ DéplacerNœud($u, X, Y, \mathcal{G}$)
    $S \leftarrow S \cup \mathcal{G}'$
  finsi
  $\mathcal{G}' \leftarrow$ FusionnerEtPartitionner($X, Y, \mathcal{G}$)
  $S \leftarrow S \cup \mathcal{G}'$
fin
Retourner S

5.4 Garantie en termes d'anonymat/respect de la vie privée

Notre algorithme, présenté en Section 5.3, fait partie des algorithmes d'anonymisation d'un graphe social par généralisation. Cet technique assure que chaque cluster contient au moins k nœuds du graphe initial, autrement dit,

chaque k nœuds sont indistinguables du point de vue de l'adversaire. Un adversaire possédant quelques connaissances auxiliaires à propos des individus présents dans un graphe social rendu public commence son attaque en échantillonnant tous les graphes candidats G tel que $G \trianglelefteq \mathcal{G}$, ce qui rend son attaque plus difficile car cela lui demande de générer un grand nombre de graphes candidats.

Afin de préserver l'utilité d'un graphe social publié, les algorithmes de généralisation minimisent (de manière directe ou indirecte) le nombre de graphes candidats. Dans notre cas, le fait d'optimiser la précision des bornes d'utilité d'une propriété structurelle donnée, entraine une réduction de nombre de graphes candidats. Malgré cette réduction, le nombre de graphes candidats reste en pratique suffisamment élevé en prenant en compte le cas des graphes sociaux réels qui possèdent généralement un grand nombre de nœuds et d'arêtes. Généralement, la condition de k-anonymité assure un degré assez acceptable de protection de la vie privée (relativement au nombre k). Le nombre de graphes candidats d'un graphe généralisé $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ est donné par l'équation (1). Ainsi, $||\llbracket\mathcal{G}\rrbracket|| = 1$ si chaque facteur entrant dans la formule précédente (chaque combinaison $\binom{n}{p}$) est égal à 1, c'est-à-dire soit il n'y a pas de liens entre les nœuds d'un même cluster soit ce cluster forme un sous-graphe complet, et en même temps soit il n'y a pas de liens entre les nœuds de deux clusters soit il y a un lien entre chaque paire de nœuds de ces deux clusters. Ainsi, même si un graphe généralisé a un seul graphe candidat, il est difficile pour un adversaire de réussir une attaque par inférence, car quand $||\llbracket\mathcal{G}\rrbracket|| = 1$ tous les nœuds d'un même cluster ont la même caractéristique (le même degré) et donc ne sont distinguables par un adversaire possédant le degré des nœuds comme connaissance auxiliaire. Cependant, nous pouvons garantir que le nombre de graphes candidats soit supérieur à un seuil donné et cela pour offrir plus de garantie en termes de préservation de la vie privée des individus présents dans un graphe social. Pour réaliser cela, nous diminuons la probabilité pour laquelle notre algorithme termine sa recherche d'un graphe généralisé (le nombre de propositions de changement d'état non acceptées dans les dernière itérations) jusqu'à ce que le nombre de graphes candidats ($||\llbracket\mathcal{G}\rrbracket||$) soit supérieur à un seuil fixé.

5.5 Implémentation

Nous sommes en train d'implémenter notre algorithme, décrit dans la Section 5.3, en langage de programmation Python [6], sous l'environnement de développement PyDev [7] intégré sous Eclipse [8]. Nous utilisons la librairie NetworkX [9] pour la création et la manipulation de graphes.

Jusqu'à l'écriture de ce rapport, nous n'avions pas encore terminé l'implémentation de notre algorithme. Cependant, nous espérons d'ici fin du stage d'avoir de résultats d'exécution de cet algorithme en évaluant sa performance sur des données issues de graphes sociaux réels et en comparant ses résultats avec ceux obtenus des algorithmes de clustering de graphes notamment les deux algorithmes de généralisation de graphes sociaux, GraphGen et SaNGreeA, présentés en Section 4.4.1 et Section 4.4.2 respectivement.

6. http ://www.python.org/
7. http ://pydev.org/
8. http ://www.eclipse.org/
9. http ://networkx.lanl.gov/

Un algorithme de clustering de graphes prend en entrées un graphe et un nombre k représentant le taille minimale de chaque cluster, et rend en sortie un graphe composé de clusters chacun de taille au moins k. Notre algorithme pourrait être vu comme un algorithme de clustering de graphes à la différence qu'il prend en entrées, en plus du graphe à généraliser et du paramètre k, une mesure d'utilité U à préserver, et rend en sortie un graphe généralisé préservant le mieux la mesure d'utilité U, et tel que chaque cluster est de taille au moins k. Ainsi, pour pouvoir tester l'efficacité et la performance de notre algorithme, plusieurs sources fournissant des données de graphes sociaux réels peuvent être utilisées par exemple :

- *Stanford Network Analysis Project (SNAP)* [10] qui regroupe une collection de données de graphes issues de réseaux sociaux en ligne, réseaux de communication, collaboration entre auteurs, ...
- *The Social Computing Data Repository* [11] qui contient en particulier des graphes sociaux issues de réseaux sociaux tel que Flickr, Twitter, Friendster, ...

Afin d'effectuer leurs analyses, les chercheurs (analystes) de graphes sociaux commencent à échantillonner un graphe candidat à partir du graphe généralisé rendu public, puis ils réalisent une analyse standard sur le graphe échantillonné. Nous suivons le même principe à la différence qu'on échantillonne, non pas seulement un seul graphe candidat, mais plutôt un certains nombre de graphes candidats (soit par exemple 100) et cela pour confirmer notre hypothèse supposant que l'écart de valeurs de la mesure d'utilité considérée est assez petit en analysant plusieurs graphes candidats échantillonnés à partir d'un graphe généralisé par notre algorithme. Pour cela, nous avons déjà implémenté une procédure permettant d'échantillonner un graphe candidat à partir d'un graphe généralisé : à partir des cardinales des clusters on engendre tous les noeuds du graphe, puis on place aléatoirement les arêtes au sein d'un même cluster (en préservant la connexité) et entre les clusters. Pour chacun des graphes candidats échantillonnés et pour chacune des mesures d'utilité dont nous avons calculé les bornes (Section 5.2), nous calculons différentes propriétés structurelles (décrites en Section 5.1) et nous comparons la moyenne des valeurs d'une propriété structurelle donnée avec celle du graphe d'origine. Pour calculer la valeur d'une propriété structurelle d'un graphe donné, nous pouvons soit faire une implémentation "maison" soit utiliser des outils existants d'analyse de graphes, par exemple UCINET [12] et Pajek [13]. On répète toute cette procédure de généralisation, d'échantillonnage, et de mesures en variant la taille minimale des clusters, par exemple pour $k = 5, 10$, et 20.

6 Conclusion et perspectives

L'anonymisation d'un graphe social est un procédé nécessaire avant toute publication des données d'un réseau social afin de préserver la vie privée des individus présents dans ce réseau. Plusieurs travaux sont réalisés pour démontrer les risques d'atteinte à la vie privée suite à une publication d'un graphe social anonymisé naïvement. Ainsi, un adversaire peut révéler l'existence d'une relation sociale entre deux utilisateurs en créant un certains nombre de nœuds,

10. http ://snap.stanford.edu/
11. http ://socialcomputing.asu.edu/
12. https ://sites.google.com/site/ucinetsoftware/home
13. http ://vlado.fmf.uni-lj.si/pub/networks/pajek/

en ajoutant des liens vers les nœuds cibles, et en établissant un motif particulier qui aide à retrouver ces nœuds ajoutés, attachés avec les nœuds cibles, dans le graphe social publié. Un autre exemple serait, un adversaire qui sait que Alice possède 60 amis dont 3 ont 30 amis et qui peut ré-identifier Alice dans le graphe social. Un autre type d'attaque consiste à inférer des attributs des profils privés à partir des attributs des profils publics des individus en relations (que ce soit ses amis ou partagent au moins un groupe en commun) avec l'individu cible.

Afin de permettre des analyses utiles sur un graphe social tout en préservant la vie privée des utilisateurs de ce réseau social, plusieurs approches d'anonymisation d'un graphe social sont développées. Ainsi, ces méthodes d'anonymisation peuvent être regroupées en trois catégories. Le k-anonymité par modification d'arêtes qui consiste à modifier la structure d'un graphe social par une série d'ajout et de suppression d'arêtes de telle sorte que chaque nœud dans le graphe modifié est indistinguable d'au moins $(k-1)$ autre nœuds en termes de quelques motifs structurels, par exemple le degré d'un nœud, son voisinage, ou tout un sous-graphe autour d'un nœud. La perturbation aléatoire des arêtes d'un graphe social qui consiste à ajouter/supprimer des arêtes ou les échanger afin de préserver contre toute tentative de ré-identification dans une manière probabiliste. La généralisation d'un graphe social qui consiste à regrouper les nœuds d'un graphe social en clusters chacun de taille au moins k, puis résumer le nombre d'arêtes existant entre et à travers ces clusters par des super-arêtes.

Nous nous sommes focalisés dans ce travail de recherche à améliorer la technique de généralisation d'un graphe social. Il est reconnu que cette technique d'anonymisation propose un niveau élevé de garantie en termes de respect de la vie privée des individus présents dans un graphe social. Cependant, l'utilité d'un tel graphe social n'atteint pas en pratique un niveau acceptable. Pour ces deux raisons et du fait que plusieurs analyses d'un graphe social s'intéressent à étudier des propriétés structurelles du graphe social étudié, nous avons proposé de guider la façon dont on généralise un graphe social en prenant en considération la mesure d'utilité qu'on veut préserver du graphe publié. Pour cela, nous avons proposé des bornes d'utilité de différentes propriétés structurelles tel que le degré des nœuds, le diamètre et le rayon d'un graphe. Une borne d'utilité représente la plus petite (respectivement la plus grande) valeur d'une mesure d'utilité donnée que peut prendre n'importe quel graphe candidat d'un graphe généralisé. Lors de sa recherche, notre algorithme tente à minimiser l'écart entre les bornes d'utilité minimale et maximale dans l'objectif que la valeur de cette propriété structurelle pour n'importe quel graphe candidat du graphe généralisé se rapproche de celle du graphe d'origine.

Comme perspectives à court et à moyen terme pour mener à bien notre projet de recherche, nous espérons d'abord avoir de résultats d'exécution de notre algorithme afin de pouvoir le comparer expérimentalement avec d'autres algorithmes de généralisation de graphes sociaux. Nous envisageons aussi de concevoir des algorithmes qui calculent des bornes d'utilité pour des propriétés structurelles (par exemple, le cœfficient de clustering, la centralité d'intermidiarité, ...) où il est difficile de les donner sous formules mathématiques.

Le domaine de l'anonymisation de réseaux sociaux est un domaine de recherche et de développement très prometteur. Nous proposons comme perspectives à long terme de prendre en considération, dans notre algorithme de généralisation d'un graphe social, l'ensemble d'attributs décrivant les utili-

sateurs du réseau social à anonymiser. Pour cela, nous pouvons adopter la technique de généralisation des valeurs des attributs en utilisant une hiérarchie d'étiquettes pour chacun des attributs. Nous proposons aussi de prendre en compte les analyses de graphes sociaux n'intéressant aux requêtes d'agrégation d'un graphe social. Ainsi, une étude permettant d'évaluer laquelle des propriétés structurelles doit on considérer dans notre algorithme de généralisation guidé par l'utilité pour pouvoir répondre assez correctement à une requête d'agrégation.

Références

[1] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[2] Elena Zheleva and Lise Getoor. Privacy in social networks : A survey. In *Social Network Data Analytics*, pages 277–306. 2011.

[3] L. Sweeney et al. k-anonymity : A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5) :557–570, 2002.

[4] P. Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6) :1010–1027, 2001.

[5] Graham J. Wills. An interactive view for hierarchical clustering. In *INFOVIS*, pages 26–31, 1998.

[6] Alan Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Internet Measurement Comference*, pages 29–42, 2007.

[7] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[8] B. Fitzpatrick and D. Recordon. *Thoughts on the social graph*. Number 92. 2007.

[9] Bradley Malin and Latanya Sweeney. Inferring genotype from clinical phenotype through a knowledge based algorithm. In *Pacific Symposium on Biocomputing*, pages 41–52, 2002.

[10] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity : Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[11] Xintao Wu, Xiaowei Ying, Kun Liu, and Lei Chen. A survey of privacy-preservation of graphs and social networks. In *Managing and Mining Graph Data*, pages 421–453. 2010.

[12] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. Technical Report 07-19, University of Massachusetts Amherst, 2007.

[13] László Babai and Ludek Kucera. Canonical labelling of graphs in linear average time. In *FOCS*, pages 39–46, 1979.

[14] Lars Backstrom, Cynthia Dwork, and Jon M. Kleinberg. Wherefore art thou r3579x ? : anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190, 2007.

[15] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, pages 173–187, 2009.

[16] Gary William Flake, Robert Endre Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4) :385–408, 2003.

[17] Elena Zheleva and Lise Getoor. To join or not to join : the illusion of privacy in social networks with mixed public and private user profiles. In *WWW*, pages 531–540, 2009.

[18] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3) :93–106, 2008.

[19] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *SIGMOD Conference*, pages 93–106, 2008.

[20] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.

[21] Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism : A general framework for privacy preserving network publication. *PVLDB*, 2(1) :946–957, 2009.

[22] Michael Hay, Gerome Miklau, David Jensen, Donald F. Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *PVLDB*, 1(1) :102–114, 2008.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, third edition, December 2009.

[24] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *PinKDD*, pages 33–54, 2008.

[25] Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Identity obfuscation in graphs through the information theoretic lens. In *ICDE*, pages 924–935, 2011.

[26] Bin Zhou and Jian Pei. The *k*-anonymity and *l*-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl. Inf. Syst.*, 28(1) :47–77, 2011.

[27] Roy Ford, Traian Marius Truta, and Alina Campan. P-sensitive k-anonymity for social networks. In *DMIN*, pages 403–409, 2009.

[28] M. Hay, G. Miklau, D. Jensen, D. Towsley, and C. Li. Resisting structural re-identification in anonymized social networks. *The VLDB Journal*, 19(6) :797–823, 2010.

[29] S. Milgram. The small world problem. *Psychology Today*, 2 :60–67, 1967.

[30] Thomas L. Saaty. Decision-making with the ahp : Why is the principal eigenvector necessary. *European Journal of Operational Research*, 145(1) :85–91, 2003.

[31] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7) :107–117, 1998.

[32] Erich Prisner. Radius versus diameter in cocomparability and intersection graphs. *Discrete Mathematics*, 163(1-3) :109–117, 1997.

[33] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45 :167–256, 2003.