MASTER EN SCIENCES MATHÉMATIQUES

Théorie des réseaux : le problème de prédiction de liens manquants

BACH, Sebastian

Award date:
2013

Awarding institution:
Universite de Namur

MASTER EN MATHÉMATIQUES

Théorie des réseaux : le problème de prédiction de liens manquants

Sebastian Bach

2013

Facultés universitaires Notre-Dame de la Paix

UNIVERSITE DE NAMUR

Faculté des Sciences

THEORIE DES RESEAUX : LE PROBLEME DE PREDICTION DE LIENS MANQUANTS

Mémoire présenté pour l'obtention

du grade académique de master en «Sciences mathématiques, à finalité spécialisée»

Sebastian BACH

Janvier 2013

Que tous ceux qui m'ont aidé dans l'élaboration de ce travail trouvent ici l'expression de ma gratitude.

Je remercie tout spécialement Renaud Lambiotte, qui a bien voulu diriger ce mémoire et qui m'a inspiré dans nos nombreuses conversations fructueuses. Merci aussi pour la relecture de ce mémoire ainsi que la vérification de l'orthographe.

Je remercie aussi Johan Barthélemy qui m'a aidé à modifier les codes en C++ et qui a corrigé les fautes d'orthographes et de syntaxes de ce mémoire.

Merci à mes deux bons potes Max Rinck et Sven Bocken qui m'ont toujours changé les idées quand je croulais sous la charge de travail pour ce mémoire.

Je remercie finalement toute ma famille qui m'a toujours soutenu au cours de l'élaboration de ce mémoire, particulièrement mon père qui a relu de nombreuses pages de rédaction mathématique sans en comprendre le contenu.

Théorie des réseaux : Le problème de prédiction de liens manquants

Résumé

Le but de ce mémoire est la description du problème des liens manquants dans la théorie des réseaux ainsi que le développement et le test de cinq méthodes de prédiction de liens manquants, à savoir la méthode aléatoire, la méthode du produit des degrés, la méthode des triangles, la méthode des communautés et la méthode hiérarchique. Nous nous concentrerons surtout sur une méthode basée sur le principe de la hiérarchie dans un réseau et dont le développement est particulièrement intéressant à cause de ses fondements mathématiques : nous commencerons par la construction d'une fonction de vraisemblance à deux paramètres que nous maximiserons afin de trouver les valeurs optimales d'un de ces paramètres. L'optimisation du deuxième paramètre sera effectuée en considérant une chaine de Markov que nous modéliserons par un algorithme de la classe *Markov Chain Monte Carlo*. Nous ajouterons à cet algorithme la règle de *Metropolis-Hastings* pour optimaliser les pas considérés par l'algorithme.
Nous appliquerons ces méthodes à des réseaux réels et nous comparerons l'efficacité de ces méthodes en utilisant la mesure *Area Under Curve* (AUC). Grâce à un exemple nous montrerons comment la prédiction de liens manquants peut être appliquée dans la vie réelle, et plus particulièrement dans les stratégies de marketing d'une entreprise.

Network theory : The missing link problem

Abstract

In this thesis, we focus on the problem of detecting missing links in networks. We consider five different types of methods : a random method, used as a baseline, a method based on the product of node degrees, one closing open triangles, one based on the presence of communities in the network and a hierarchical method. We focus mainly on the latter because of its interesting mathematical formalism : it is based on a two-parameters likelihood-function that is maximized in order to uncover the best hierarchical representation of the data. The optimization is performed by using a *Markov Chain Monte Carlo* algorithm in the space of trees. The five methods are tested on five real world networks and their efficiency compared using the *Area Under Curve* statistic (AUC). Finally, we illustrate how to apply one of these methods to target products in a marketing strategy.

Table des matières

Introduction

L'étude des réseaux est devenue une branche très importante de la recherche scientifique actuelle. À cause de la montée de l'Internet et l'existence d'ordinateurs très puissants, nous avons la possibilité de collecter et d'analyser les données de réseaux complexes. Il s'agit d'une branche scientifique interdisciplinaire combinant des théories mathématiques, physiques, informatiques, biologiques, sociologiques,.... Les champs d'applications de cette théorie semblent inépuisables : dans [17], les auteurs donnent des exemples de la modélisation de phénomènes biologiques par la théorie des réseaux. Un exemple d'application de la théorie des réseaux dans la sociologie est donné dans [1] où les auteurs modélisent les relations entre les membres d'une communauté. Enfin, la théorie des réseaux peut même être appliquée pour proposer une manière alternative d'évaluer un championnat de sport, tel que Park et Newman le font dans l'article [26]. Un sous-domaine de la théorie des réseaux particulièrement intéressant est celui des réseaux sociaux. Les nœuds d'un réseau social correspondent à des personnes d'un groupe ou d'autres éléments incorporés dans un contexte social et les arêtes sont les interactions entre ces entités. Un exemple simple et concret d'un tel réseau social est celui dont les nœuds sont les employés d'une entreprise et un lien entre deux nœuds indique que ces deux employés travaillent sur un même projet. Les réseaux sociaux sont donc des outils très puissants afin de représenter et de modéliser les interactions dans un groupe social.

Un autre domaine d'application de la théorie des réseaux sociaux est la lutte contre la criminalité. Jusqu'il y à peu, en criminologie, les applications mathématiques se basaient soit sur un point de vue statistique (voir [14, 32]), soit sur un point de vue de systèmes dynamiques ([22, 27]). Depuis les attaques du 11 septembre 2001, l'étude des réseaux est devenu une méthode centrale dans l'analyse scientifique des cellules terroristes (voir [31] et plus particulièrement [6]). Notons dans ce cadre surtout le travail de Vadis Krebs, qui, dans [16], analyse de manière exhaustive le réseau des terroristes des attaques du 11/09 à New York.

Comme Rhodes le précise dans [28], l'analyse des réseaux sociaux joue un rôle très important quant à l'organisation et la représentation de données correspondant à des groupes criminels et terroristes. Mais la collecte de données est la base de chaque analyse d'un réseau criminel et elle constitue désormais l'étape la plus difficile à effectuer. Quand les agents de police désirent observer une cellule terroriste, ils doivent d'abord déterminer l'ensemble des acteurs de ce réseau, et, après, définir la notion de lien entre deux acteurs (par exemple une communication téléphonique suffit-elle déjà pour obtenir un lien significatif entre deux criminels ?). Une fois ces premières étapes effectuées, les agents doivent lancer des enquêtes afin de récolter une grande masse d'informations. Ce processus nécessite une démarche qui peut se révéler relativement longue, ce qui est bien évidemment un grand inconvénient, car souvent les affaires criminelles doivent être résolues le plus vite possible afin de limiter leur action néfaste sur la société.

Cependant, les criminels, adversaires des agents de police, utilisent également toutes les possibilités des nouvelles technologies et font tout afin d'échapper aux enquêtes des agents. Ils détruiront leurs traces, ils essayeront de mettre les agents sur une mauvaise piste, etc. Il est donc évident que l'ensemble des données récoltées par les policiers soit souvent incomplet, que ce soit en termes d'acteurs ou en termes d'interactions entre ces acteurs. Dans le cadre de ce travail, nous nous intéresserons au deuxième cas, i.e. le manque d'informations sur les interactions entre les acteurs d'un réseau social. Ce problème est connu dans la littérature sous le nom de *problème de liens manquants*.

Le problème de liens manquants est un sujet central de la recherche non seulement dans le cadre la lutte contre la criminalité, mais pour l'ensemble de la théorie des réseaux sociaux. On s'intéresse souvent à la dynamique d'un réseau par rapport aux arêtes. Dans les réseaux sociaux, non seulement des nouveaux nœuds apparaissent (si nous reconsidérons l'exemple précité du réseau de l'entreprise, il y a des employés qui, au cours du temps, quittent l'entreprise et d'autres qui y seront embauchés), mais aussi les interactions entre les personnes changent (des projets se terminent, d'autres commenceront) et il serait intéressant pour les chefs d'entreprise de savoir estimer l'état de ce réseau à un instant ultérieur.
Pour revenir une dernière fois au domaine de la lutte contre le terrorisme, comme Hasan et al. le précisent dans [13], l'objectif de l'analyse de réseaux terroristes est de conjecturer que certains individus (c'est-à-dire des terroristes) travaillent ensemble mais leurs interactions ne peuvent pas être détectées, puisqu'ils font tout afin de cacher leurs connections. Autrement dit, dans le cadre de réseaux terroristes, on utilise les méthodes de prédiction de liens manquants non seulement pour pouvoir prédire le réseau dans le futur, mais également pour trouver des liens cachés entre les individus. Ainsi, la prédiction de liens manquants aide les agents de

police ou les militaires dans leur travail, puisque ces méthodes indiquent des paires de nœuds, c'est-à-dire des terroristes dont il est très probables qu'ils interagissent ensemble et, dès lors, les agents peuvent concentrer leurs enquêtes sur ces individus.

D'une manière très générale et plus formelle, nous pouvons décrire le problème de liens manquants de la façon suivante. À partir d'un réseau observé G à n nœuds fixés et m arêtes à l'instant t_0, nous voulons prédire l'état du réseau à l'instant t_1. Dans ce mémoire, nous examinerons des méthodes dont le but est de prédire avec une probabilité la plus grande possible des liens qui s'ajouteront au réseau de l'instant t_0 à l'instant t_1. Au delà, les méthodes de prédiction de liens manquants peuvent aussi être appliquées pour trouver des liens du passé qui n'ont pas encore été détectés, comme c'est par exemple le cas pour les réseaux terroristes.

Ce mémoire est structuré en cinq chapitres qui commenceront tous par quelques mots introductifs.
Dans le premier chapitre, nous présenterons quatre méthodes de prédiction de liens manquants dont les codes écrits en *Matlab* se trouvent à l'*Annexe B*.
Le deuxième chapitre sera consacré à une cinquième méthode de prédiction de liens manquants, appelée la méthode hiérarchique. Il s'agit d'une méthode développée par Clauset et al. dans [9] et dont les codes sont accessibles gratuitement sur le site Internet d'Aaron Clauset [1]. Les fondements mathématiques de cette méthode sont particulièrement intéressants, car il passe de la construction d'une fonction de vraisemblance à la maximisation de celle-ci afin de trouver les valeurs maximales d'un des deux paramètres de la fonction de vraisemblance. Les valeurs optimales du deuxième paramètre seront trouvées en définissant une chaîne de Markov que nous modéliserons par un algorithme de classe *Markov Chain Monte Carlo* auquel nous ajouterons la règle de *Metropolis-Hastings*. Nous présenterons cette méthode de manière très détaillée et exhaustive dans ce deuxième chapitre qui forme ainsi le cœur de ce travail.
Dans le troisième chapitre, nous appliquerons les cinq méthodes à cinq différents réseaux sociaux réels. Nous comparerons les résultats de ces méthodes avec la statistique AUC que nous présenterons au début de ce chapitre. De plus, nous ne calculerons non seulement les valeurs d'AUC pour les méthodes, mais également les écarts-types de ces valeurs, ce qui n'a pas été fait dans [9] ni, à notre connaissance, dans la plupart des articles sur la prédiction de liens manquants. L'analyse des écarts-types est primordiale pour déterminer si une méthode est significativement meilleure qu'une autre. Nous montrerons dans la dernière section de ce chapitre une application possible d'une méthode de prédiction de liens manquants à un réseau réel. Cette application montre comment la prédiction de liens man-

1. `http://tuvalu.santafe.edu/~aaronc/`

quants peut être utile dans le cadre des stratégies de marketing d'une entreprise. Nous clôturerons ce mémoire par une conclusion générale, dans laquelle nous résumerons les points principaux de ce travail ainsi que les résultats trouvés.

Remarquons finalement que, dans ce travail, nous analyserons l'efficacité des méthodes de prédiction de liens manquants en termes de la mesure AUC. Nous n'effectuerons pas une étude de complexité des algorithmes et nous ne centrerons pas ce travail sur l'implémentation de ces méthodes. Les méthodes ont été implémentées en *Matlab* d'un point de vue pragmatique, sans optimiser les codes au point de vue de complexité ou de stockage des données.

Chapitre 1

Présentation de quelques méthodes de prédiction de liens manquants

Un exemple très populaire dans la littérature de réseaux sociaux sur lesquels on a testé des méthodes de prédiction de liens manquants est le réseau dont les nœuds sont les chercheurs d'un même domaine et dont une arête entre deux nœuds indique que ces deux auteurs ont publié un document ensemble (*co-authorship*). Dans [20], Liben-Nowell et Kleinberg considèrent ce réseau afin de montrer et de comparer l'efficacité de différentes méthodes de prédiction de liens manquants. Cet article a largement influencé ce mémoire puisque les auteurs présentent le problème de la prédiction de liens manquants et appliquent deux des méthodes que nous utiliserons dans le cadre de ce travail, à savoir la méthode des triangles et la méthode du produit des degrés.

De manière générale, les méthodes de prédiction de liens manquants fonctionnent de la même façon : on cherche les paires de nœuds qui sont "très proches", "très semblables". En d'autres mots, les méthodes calculent, pour chaque paire de nœuds non connectés dans le réseau à l'instant t_0, un score indiquant la probabilité d'existence d'un lien entre cette paire de nœuds à l'instant t_1. Les méthodes établiront donc une liste de paires de nœuds ordonnées de manière décroissante par rapport au score correspondant. Il est clair que les notions de proximité et de ressemblance sont à spécifier, ce qui sera le but de ce chapitre. En effet, les méthodes de prédiction de liens manquant diffèrent entre elles dans la manière de définir cette proximité entre nœuds.

En pratique, afin de comparer les méthodes, nous considérerons un réseau G duquel nous enlèverons un ensemble d'arêtes $\{k_i\}$. Appelons le nouveau graphe G'. Nous appliquerons nos méthodes de prédiction de liens manquants à ce graphe G' et nous examinerons si elles permettent de retrouver les arêtes enlevées $\{k_i\}$.

La première méthode que nous présenterons dans ce chapitre est une méthode complètement aléatoire. Elle choisit des paires de nœuds non connectés dans le graphe observé de manière aléatoire. On s'attend à ce que cette méthode soit la moins efficace, puisque toutes les paires de nœuds non-connectés ont la même probabilité d'être connectées à l'instant t_1. Pour qu'une méthode soit efficace, elle se doit d'être significativement meilleure que cette méthode aléatoire. Cette dernière est néanmoins utile puisqu'elle nous servira à calibrer la performance des autres méthodes.
La deuxième méthode que nous présenterons est basée sur le produit de degrés de deux nœuds. Il s'agit d'une méthode que Clauset et al. et Liben-Nowel et al. utilisent dans leurs travaux [20, 9].
La troisième méthode est une méthode bien connue dans la littérature. Le score entre deux nœuds non-connectés sera le nombre de triangles que l'arête entre cette paire de nœuds ferme (ce qui revient au nombre de voisins communs comme nous le verrons dans cette partie). Nous appellerons cette méthode la méthode des triangles.
La dernière méthode présentée dans ce chapitre s'appelle la méthode des communautés. Nous définirons le concept de communautés entre nœuds et expliquerons comment celles-ci peuvent aider à prédire des liens manquants.

1.1 La méthode aléatoire

Cette méthode trouve son inspiration dans la théorie des graphes aléatoires. Dans cette section, nous introduirons d'abord certains concepts de la théorie des graphes et présenterons les graphes d'Erdös-Rényi. Ensuite, nous présenterons la méthode aléatoire dans le cadre de la prédiction de liens manquants.

1.1.1 Le principe de base : les graphes d'Erdös-Rényi

Cette partie est principalement basée sur [24]. Le concept des graphes aléatoires a été développé pour répondre aux questions suivantes : Quels sont les effets attendus d'une propriété d'un graphe (par exemple son nombre de liens) sur la structure du réseau entier (par exemple son diamètre) ? Si nous connaissons par exemple la distribution des degrés de liaisons des nœuds d'un réseau, que pouvons nous dire sur le réseau tout entier ?

Défintions

Un *graphe aléatoire* est un modèle de réseau pour lequel les valeurs de certains paramètres sont fixées et le graphe est complètement aléatoire dans les autres paramètres. L'exemple le plus simple d'un graphe aléatoire est le réseau dans lequel nous fixons simplement le nombre de nœuds n et le nombre de liens m. Dès lors, nous considérons n nœuds et nous plaçons m liens de manière aléatoire entre ces nœuds. Autrement dit, nous considérons de manière aléatoire m paires de nœuds parmi l'ensemble de toutes les paires de nœuds possibles et nous y ajouterons un lien dans le graphe. En général, nous ne considérons que des graphes simples, i.e. des graphes dans lesquels le point de départ et le point d'arrivée d'une arête ne coïncident jamais et sans liens multiples entre deux nœuds quelconques. Cependant, la définition de graphe aléatoire peut aisément être élargie à des graphes plus complexes. Un graphe aléatoire pour lequel nous avons fixé le nombre de nœuds n et le nombre de liens m est noté $G(n, m)$.

Une autre manière d'interpréter les graphes aléatoires est de choisir aléatoirement un graphe parmi l'ensemble de tous les graphes à n nœuds et m liens. En général, le modèle du graphe aléatoire n'est pas défini comme un seul réseau généré aléatoirement, mais comme un *ensemble* de réseaux, ou encore comme une distribution de probabilité d'observer les réseaux possibles. Dès lors, le modèle $G(n, m)$ est défini comme une distribution de probabilité $P(G)$ sur tous les graphes G pour laquelle $P(G) = \frac{1}{\Omega}$ pour des graphes simples avec n nœuds et m liens et $P(G) = 0$ pour tous les autres graphes, où Ω est le nombre total de tels graphes simples.

Certaines propriétés des graphes aléatoires $G(n, m)$ sont facile à démontrer. Il est évident que le nombre moyen de liens est m et que le degré moyen d'un nœud (c'est-à-dire le nombre d'arêtes partant de ce nœud) est $\langle d \rangle = \frac{2m}{n}$. Cependant, d'autres propriétés sont plus difficiles à dériver, ce qui a motivé les mathématiciens à développer un modèle similaire, mais plus facile à manipuler. Dans ce modèle, nous ne fixons pas le nombre mais la *probabilité* de liens entre des nœuds. De nouveau, nous considérons n nœuds, mais nous plaçons un lien entre chaque paire de nœuds avec une probabilité p. Nous supposerons que la probabilité de connexion entre une paire de nœuds est indépendante des autres probabilités de connexions. Dans ce réseau, le nombre de liens n'est pas fixé. En effet, il est possible qu'il n'y ait soit aucun lien dans tout le graphe soit autant de liens que paires de nœuds, mais, pour la plupart des valeurs de p, ces événements sont très peu probables. Il est clair que le nombre moyen de liens du graphe augmente avec la probabilité p. De nouveau, la définition de ces graphes n'est pas en terme d'un seul réseau, mais en termes d'une distribution de probabilité sur tous les réseaux possibles.
En particulier, $G(n, p)$ est l'ensemble des réseaux de n nœuds dans lequel chaque

graphe simple G apparait avec une probabilité

$$P(G) = p^m(1-p)^{\binom{n}{2}-m} \tag{1.1.1}$$

où m est le nombre de liens dans le graphe. Les graphes non-simples ont une probabilité de 0. L'équation 1.1.1 est intuitive. Exactement m paires sont connectées avec une probabilité p, ils en restent $\binom{n}{2}-m$ [1] qui ne sont pas connectés. Le nombre $\binom{n}{2}$ est le nombre de paires possibles parmi n nœuds.

Les graphes aléatoires $G(n,p)$ sont aussi appelés *graphes d'Erdös-Rényi* d'après les deux mathématiciens Paul Erdös et Alfréd Rényi rappelant leurs contributions dans cette branche.
Dans la section suivante, nous examinerons quelques propriétés des graphes d'Erdös-Rényi.

Nombre moyen de liens et degré moyen

Calculons d'abord le nombre moyen de liens dans un graphe d'Erdös-Rényi. Le nombre exact de liens ne peut pas être calculé, mais nous pouvons déterminer sa moyenne ou sa valeur attendue. La probabilité de tracer un graphe à n nœuds et m liens est donnée par

$$P(m) = \binom{\binom{n}{2}}{m} p^m(1-p)^{\binom{n}{2}-m} \tag{1.1.2}$$

Dans l'équation 1.1.2, le nombre $\binom{\binom{n}{2}}{m}$ est le nombre de possibilités de sélectionner m paires parmi toutes les $\binom{n}{2}$ paires possibles. L'équation 1.1.2 est exactement la formule d'une distribution binomiale. Dès lors, la valeur moyenne est donnée par

$$\langle m \rangle = \binom{n}{2} p \tag{1.1.3}$$

Donc, si nous connaissons p et n, nous pouvons calculer le nombre moyen de liens de $G(n,p)$. De nouveau, l'équation 1.1.3 est intuitive : le nombre de liens moyens est égal à la probabilité de liens entre n'importe quelle paire de nœuds multipliée par le nombre de paires possibles.
Utilisons ce résultat afin de calculer la valeur du degré moyen d'un graphe d'Erdös-Rényi. Comme nous l'avons vu précédemment, le degré moyen d'un tel graphe à exactement m liens est donné par $\langle d \rangle = \frac{2m}{n}$. Dès lors, le degré moyen de $G(n,p)$ est donné par

$$\langle d \rangle = \frac{2\langle m \rangle}{n} = \frac{2}{n}\binom{n}{2} p = (n-1)p \tag{1.1.4}$$

1. Rappelons que $\binom{a}{b} = \frac{a!}{b! \times (a-b)!}$.

La méthode aléatoire de prédiction de liens manquants se place dans ce schéma théorique, comme nous le verrons dans la section suivante.

1.1.2 La méthode aléatoire

Considérons comme expliqué plus haut un réseau à n nœuds et m arêtes $G(n, m)$. Le nombre de paires de nœuds non-connectés est dès lors le nombre de paires de nœuds possibles moins le nombre de paires qui sont connectées :

$$\binom{n}{2} - m$$

et donc

$$\frac{n \times (n-1)}{2} - m$$

Si nous enlevons α arêtes $\{k_i\}_{i=1}^{\alpha}$ des m arêtes de $G(n, m)$, nous obtenons le graphe $G'(n, m-\alpha)$ et le nombre de paires de nœuds non-connectés vaut dès lors

$$\frac{n \times (n-1)}{2} - (m - \alpha).$$

La méthode aléatoire de prédiction de liens manquants est définie comme suit. On considère le réseau $G'(n, m-\alpha)$. Comme lors de la construction d'un graphe d'Erdös-Rényi, on attribue une probabilité p uniforme d'existence d'un lien entre chaque paire de nœuds non connectés. Précisément, nous considérons que les $m-\alpha$ arêtes de G' sont fixées et nous ajoutons un lien à chacun des $\frac{n\times(n-1)}{2} - (m-\alpha)$ paires de nœuds non-connectés de G' avec une probabilité proportionnelle au nombre de liens qu'on souhaite ajouter. Ainsi, la méthode proposera une liste de paires de nœuds choisies aléatoirement.

Au point de vue algorithmique, cette méthode est implémentée de manière légèrement différente. La méthode choisit aléatoirement un certain nombre de paires de nœuds parmi l'ensemble des $\frac{n\times(n-1)}{2} - (m-\alpha)$ paires de nœuds non-connectés dans $G'(n, m-\alpha)$ et mettra ces paires de nœuds dans la liste des arêtes prédites. Le nombre de paires de nœuds que nous choisissons aléatoirement dépend du nombre de prédictions que nous souhaitons faire. Puisque toutes les paires de nœuds non-connectés sont équiprobables à être choisies par cette méthode, il n'y a aucune raison que la méthode aléatoire choisisse les arêtes enlevées $\{k_i\}_{i=1}^{\alpha}$ plutôt que d'autres, et on s'attend à ce que son utilité soit limitée.

En résumé, cette méthode attribue le même score à toutes les paires de nœuds non-connectés. Le but des méthodes que nous présenterons par la suite sera d'améliorer ce concept en différenciant les arêtes par des critères de qualité.

1.2 La méthode du produit des degrés

Comme la méthode aléatoire est basée sur les graphes d'Erdös-Rényi, la méthode que nous présenterons dans cette partie est basée sur un principe semblable : *le modèle de configuration*. Nous présenterons ce principe dans la partie suivante et nous en déduirons une méthode de prédiction de liens manquants.

1.2.1 Le principe de base : le modèle de configuration

Définitions

De nouveau, cette partie est principalement basée sur [24]. Contrairement aux graphes d'Erdös-Rényi qui sont un couple composé du nombre de nœuds n et de la probabilité d'existence de lien p, le modèle de configuration caractérise un graphe donné par le nombre de nœuds n et une séquence de n *degrés de nœuds*. Rappelons que le *degré* d_i du nœud i est le nombre de liens reliant ce nœud. Le nombre d'arêtes est donc également déterminé par

$$m = \frac{1}{2} \sum_i d_i.$$

À partir de ce couple $G(n, \{d_i\}_{i=1}^n)$, nous construirons un graphe aléatoire de la manière suivante : nous associons à chaque nœud i d_i "stubs" comme indiqué sur la figure 1.1. En total, il y en a $\sum_i d_i = 2m$ stubs. Après, nous choisissons deux stubs de manière aléatoire et nous créons une arête entre ces deux stubs, comme indiqué sur la figure 1.1. Ensuite, nous considérons deux autres stubs parmi les $2m - 2$ restants, et ainsi de suite. Le résultat sera un réseau dont chaque nœud i est de degré exactement d_i. Il s'agit bien d'un graphe aléatoire, parce que chaque paire de stubs est choisie de manière aléatoire. Toutes les arêtes sont donc équiprobables. Remarquons qu'il y a quelques problèmes qui se posent si nous générons des réseaux comme décrit plus haut : Il faut par exemple que le nombre de stubs soit un nombre pair. En plus, rien n'empêche une arête de relier un nœud à lui-même. Nous négligerons ces problèmes dans ce travail, puisque nous ne considérerons que des graphes simples. Le lecteur intéressé trouvera des informations plus détaillées dans [24].

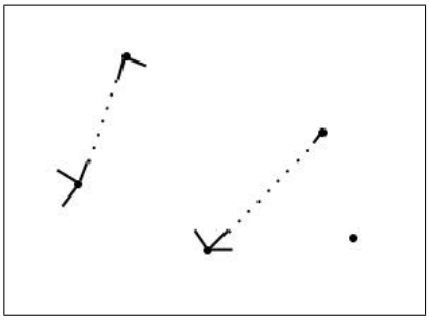

FIGURE 1.1 – Le modèle de configuration. Chaque nœud admet un certain nombre fixe de stubs dont deux paires sont liées par une arête dans cet exemple.

Probabilité d'existence d'une arête dans le modèle de configuration

Calculons dans cette partie la probabilité p_{ij} d'existence d'une arête entre les nœuds i et j. Il est clair que lorsqu'un des nœuds i ou j est de degré 0, la probabilité d'une arête entre ces nœuds vaudra $p_{ij} = 0$. Supposons dès lors que $d_i \times d_j > 0$. Considérons n'importe quel stub partant de i. Calculons la probabilité que ce stub soit connecté à un des stubs de j. En total, il y en a $2m$ stubs, donc $2m - 1$ en soustrayant le stub de i considéré. Parmi ces $2m - 1$ nœuds, exactement d_j partent de j. Puisque chaque stub dans le graphe a la même probabilité d'être connecté à n'importe quel autre stub, la probabilité que notre stub considéré de i soit connecté à n'importe quel stub de j est donnée par $\frac{d_j}{2m-1}$. Mais puisqu'il y a d_j stubs autour du nœud j, la probabilité d'existence d'une arête entre i et j est donnée par l'équation 1.2.1.

$$p_{ij} = \frac{d_i \times d_j}{2m - 1}. \tag{1.2.1}$$

Nous remarquons dès lors que deux nœuds dont le produit des degrés est élevé sont plus probable d'être liés par une arête que deux nœuds d'un produit des degrés plus faibles.

1.2.2 La méthode du produit des degrés

La méthode du produit des degrés s'inspire du modèle de configuration, en particulier du fait que la probabilité d'existence d'une arête entre i et j est donnée par 1.2.1.
Il s'agit en fait d'une méthode bien connue dans la littérature. Elle est basée sur le voisinage d'un nœud, dont nous parlerons encore dans la description de la méthode suivante. Indiquons l'ensemble des nœuds voisins à i par $\Gamma(i)$ qui est donc l'ensemble des nœuds liés par une arête à i. Dès lors, le produit des degrés de deux

nœuds i et j vaut $|\Gamma(i)| \times |\Gamma(j)|$ [2], c'est-à-dire le nombre de voisins de i multiplié par le nombre de voisins de j. Mitzenmacher a montré dans [21] que le produit des degrés est un bon outil pour simuler l'évolution d'un réseau. Newman a aussi montré dans [23] que la probabilité que deux auteurs i et j publient un article ensemble est corrélée au produit des nombres de collaborateurs de i et de j. Se basant sur cette observation, Liben-Nowel et al. utilisent dans [20] une méthode de prédiction de liens manquants basée sur le produit des degrés du réseau des chercheurs dont nous avons parlé dans l'introduction de ce chapitre. La motivation de cette méthode est que la probabilité qu'une nouvelle arête concernant le nœud i est proportionnelle à $|\Gamma(i)|$, c'est-à-dire au nombre de voisins de i.

Cette méthode attribuera donc à n'importe quelle paire de nœuds non-connectés i et j un score qui sera le produit des degrés de ces nœuds :

$$score(i, j) = |\Gamma(i)| \times |\Gamma(j)|.$$

Les liens prédits seront choisis par la règle suivante : plus grand le produit des degrés d'une paire de nœuds, plus grande sera la probabilité d'existence d'un lien entre cette paire de nœuds (ce qui correspond bien à la probabilité d'existence d'un lien calculée dans l'équation 1.2.1).
Dans [20], les auteurs ont montré qu'il s'agit d'une méthode dont les prédictions sont meilleures que celles de la méthode aléatoire. Nous présenterons dans le troisième chapitre ce qu'on entend par "meilleures prédictions" et nous vérifierons leurs résultats en les appliquant à d'autres données.

1.3 La méthode des triangles

Dans cette partie, nous présenterons une méthode basée sur un principe très populaire dans la littérature : les triangles dans les réseaux. Un triangle dans un réseau est une structure dans laquelle trois nœuds sont liés l'un à l'autre par trois arêtes, comme nous le voyons sur la figure 1.2. En sciences sociales, il a été montré dans [19] que, si dans un réseau social, l'individu A est lié aux individus B et C, une arête entre B et C permet aux trois individus de détecter mutuellement les comportements fautifs des autres. La surveillance mutuelle augmentera dès lors la confiance entre les individus connectés, ce qui réduit la compétition entre les individus et facilite l'échange d'informations. Selon cette hypothèse, la formation des triangles augmente le flux d'informations dans un réseau social.

2. Dans ce travail, nous noterons le nombre d'éléments d'un ensemble E par $|E|$.

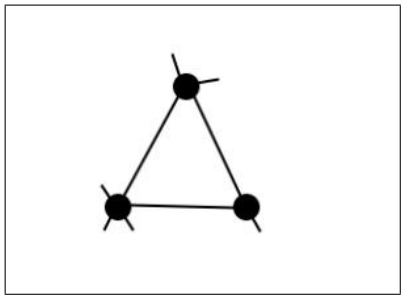

FIGURE 1.2 – Exemple d'un triangle dans un graphe

Nous présenterons tout d'abord le principe de base de cette méthode de prédiction de liens manquants, et, ensuite, nous la décrirons en détail et nous citerons quelques résultats antérieurs de cette méthode dans la littérature.

1.3.1 Le principe de base : les triangles dans un réseau

La fermeture de triangles joue un rôle prépondérant dans l'évolution des réseaux sociaux. Dans [1], les auteurs modélisent la dynamique d'un réseau entre des personnes d'une même communauté par les triangles : si un individu est ami avec deux autres, il est très probable que ces deux individus développent également une relation amicale. Notre méthode de prédiction de liens manquants sera basée sur ce principe : Si nous trouvons deux nœuds non-connectés i et j qui sont liés tous les deux à un même troisième nœud k, nous supposons qu'une arête entre i et j est très probable. Un exemple pour ce principe est donné à la figure 1.3.

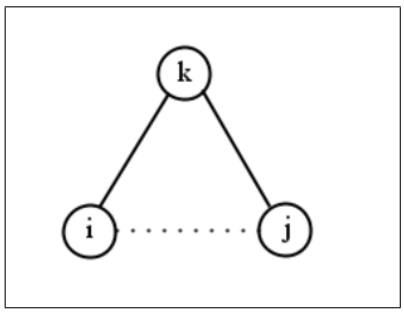

FIGURE 1.3 – Fermeture du triangle en ajoutant un lien entre i et j

Remarquons que cette méthode revient à trouver les paires de nœuds du réseau qui sont reliés par un chemin de longueur deux. Nous pouvons évidemment généraliser cette méthode et considérer aussi les paires de nœuds qui sont reliés par un chemin de longueur 3, 4,... Il s'agit d'une méthode très populaire dans la

littérature appelée la méthode des plus courts chemins. Des paires de nœuds qui sont liés par un court chemin reçoivent un score plus grand que ceux qui sont liés par un chemin plus long. C'est la raison pour laquelle, en général, le score que cette méthode associe aux paires de nœuds est la valeur négative de la longueur du plus court chemin entre ces nœuds pour assurer que les chemins les plus courts sont préférés.

Cependant, dans ce mémoire, nous n'allons pas appliquer la méthode des plus courts chemins, mais nous ajouterons une amélioration à la méthode des triangles (c'est-à-dire la méthode des plus courts chemins de longueur 2) que nous avons présentée ci-dessus.

1.3.2 La méthode des triangles

Le score que notre méthode améliorée associe aux deux nœuds i et j sera donné par le nombre de triangles qui sont fermés par l'ajout d'une arête entre i et j. Dès lors, comme on le voit sur la figure 1.4, l'arête entre i et j obtiendra un score de $score(i, j) = 4$, pendant que l'arête entre i' et j' obtiendra un score de $score(i', j') = 2$.

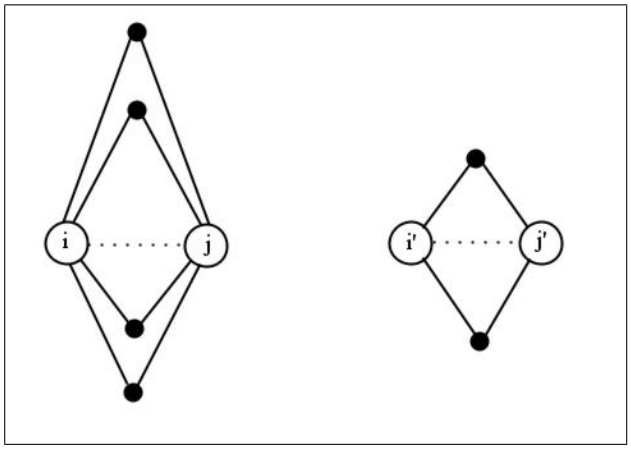

FIGURE 1.4 – Amélioration de la méthode des triangles

La méthode des triangles ordonnera les nouvelles arêtes en fonction des nombres des triangles qu'ils ferment. Cependant, ce principe est lié à une autre propriété des nœuds d'un graphe : *les voisins.*

Nous en avons déjà parlé dans la section sur la méthode du produit des degrés, mais rappelons que le voisinage d'un nœud i se note $\Gamma(i)$ et est l'ensemble de tous

les nœuds qui sont directement liés à i par une arête, comme nous le voyons sur la figure 1.5.

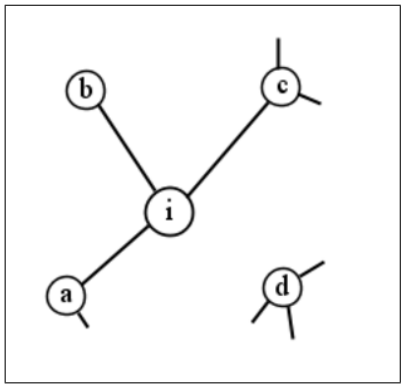

FIGURE 1.5 – Le voisinage du nœud i : $\Gamma(i) = \{a, b, c\}$

La méthode des triangles que nous venons de présenter est équivalente à une méthode connue dans la littérature, à savoir une méthode basée sur le nombre de voisins communs. En effet, s'il existe n différents triangles entre les nœuds i et j, alors les nœuds i et j ont n voisins en commun, c'est-à-dire $|\Gamma(i, j)| = n$. Comme pour la méthode du produit des degrés, Liben-Nowell et al. ont appliqué cette méthode dans [20] au réseau des auteurs d'un même domaine. Cette méthode est également motivée par une observation de Newman [23], qui a montré une corrélation entre le nombre de voisins communs de deux auteurs à l'instant t_0 et la probabilité que ces deux auteurs publient un document ensemble dans le futur. Dans [9, 20], les auteurs montrent que cette méthode est très efficace et produit des résultats jusqu'à 40 fois meilleurs que la méthode aléatoire. Dans le chapitre 3, nous étendrons ces résultats à d'autres jeux de données.

1.4 La méthode des communautés

Cette quatrième méthode, qui sera la dernière que nous présenterons dans ce chapitre, se base sur les communautés dans les réseaux. L'idée d'envisager les communautés dans les réseaux dans le cadre de la prédiction de liens manquants n'a pas encore été explorée de manière détaillée. Un travail à citer est celui de Soundarajan et al. qui ajoutent des informations sur les communautés dans les réseaux à la méthode des voisins communs dans [30]. Cependant, dans ce mémoire, nous allons construire une nouvelle méthode de prédiction de liens manquants qui sera basée sur un algorithme de recherche de communautés appelé la méthode de *Louvain*. Cet algorithme est expliqué dans [5] et modifié dans [18]. Dans la partie

suivante, nous présenterons d'abord brièvement cet algorithme *Louvain* et, ensuite, nous présenterons la méthode de prédiction basée sur les communautés.

1.4.1 Le principe de base : les communautés dans les réseaux

Définitions

Les individus d'une société ont une tendance naturelle à former des communautés, et même des communautés de communautés, etc (voir dans ce contexte aussi le chapitre prochain sur la méthode hiérarchique). D'après [11, 5], une communauté dans un graphe est un ensemble de nœuds tel qu'il existe beaucoup de liens entre eux, tandis qu'il existe relativement moins de liens entre des paires de nœuds qui appartiennent à différentes communautés. La figure 1.6 illustre bien cette idée de communautés de nœuds (cet image provient de [11]).

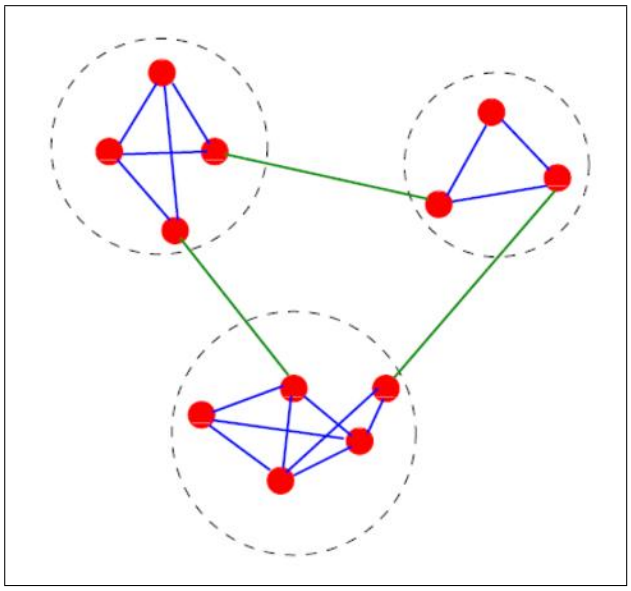

FIGURE 1.6 – Exemple de communautés dans un graphe ([11])

Puisque la théorie des réseaux devient de plus en plus importante et les réseaux observés de plus en plus grands (Google lie à titre d'exemple plus que 10^9 sites web), il est de plus en plus important de structurer le graphe en communautés afin de représenter ce graphe et d'en tirer des conclusions sur la topologie de celui-ci.

La méthode Louvain

Dans cette section, nous présentons une méthode pour découvrir des communautés dans des grands graphes, appelée la méthode *Louvain*. Cette partie est

principalement basée sur [5, 18]. Remarquons tout d'abord que l'algorithme que nous présentons dans cette partie considère des graphes pondérés, c'est-à-dire des graphes dont les arêtes ont un certain poids.
Le but d'un algorithme de détection de communautés est de trouver la meilleure partition des nœuds de ce réseau en communautés. Une mesure très populaire pour mesurer la qualité d'une partition en communautés est la modularité Q. D'après [19], la modularité d'une partition mesure si les arêtes sont plus abondantes à l'intérieur des communautés que dans le cas d'une partition aléatoire, c'est-à-dire

$$Q = (\text{fraction de liens dans une communauté})$$

$$- (\text{fraction attendue de ces liens}) .$$

Dès lors, si nous considérons la partition $\mathcal{P} = \{C_i\}$, la matrice d'adjacence A d'un réseau et $m = \frac{1}{2}\sum_{i,j} A_{ij}$ le nombre d'arêtes du réseau, alors l'équation de la modularité devient

$$Q = \frac{1}{2m} \sum_{C \in \mathcal{P}} \sum_{i,j \in C} \left[A_{ij} - P_{ij} \right], \tag{1.4.1}$$

où la deuxième somme $(i, j \in C)$ assure qu'on considère uniquement les nœuds d'une même communauté C et considère dès lors les liens à l'intérieur d'une communauté. La valeur P_{ij} est le poids attendu entre i et j. Il est évident que, afin de connaître cette matrice P, nous devons admettre une hypothèse sur la distribution de liens. L'hypothèse choisie dans la plupart des travaux est

$$P_{ij} = \frac{d_i d_j}{2m}, \tag{1.4.2}$$

où d_i est, de nouveau, le degré du nœud i, mais remarquons qu'ici, dans le cas d'un graphe pondéré, d_i n'est pas seulement le nombre d'arêtes partantes de i, mais *la somme des poids* des arêtes partantes de i. Le modèle 1.4.2 tient en compte le poids des nœuds et préserve dès lors la distribution des degrés. Ce modèle étant choisi, l'équation de la modularité 1.4.1 devient

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(C_i, C_j), \tag{1.4.3}$$

où δ est le symbole de Kronecker et C_i est la classe à laquelle le nœud i appartient.

L'algorithme *Louvain* considère cette mesure comme une fonction à optimiser, c'est-à-dire il cherche, pour un réseau donné, la partition en communautés qui maximise l'équation 1.4.3.

L'algorithme est composé en deux phases :

1. Au pas initial, chaque nœud forme une communauté. Le nombre de communautés est donc équivalent au nombre de nœuds et les communautés sont formées par des singletons. Ensuite, l'algorithme envisage pour chaque nœud i tous les voisins j de i et il évalue le gain de modularité si on place i dans la communauté d'un de ces voisins. Si au moins un de ces déplacements admet un gain de modularité positif, le déplacement admettant le plus grand gain possible sera effectué.
   L'algorithme répète ce procédé jusqu'au moment où on ne gagne plus rien en modularité, c'est-à-dire lorsqu'on a atteint un maximum local. A cet instant, aucun changement individuel n'améliore la modification.
   Avant de passer à la deuxième phase de l'algorithme, notons que, d'après [5], le gain de modularité, si on place i dans la communauté C, est très facile à calculer par la formule suivante :

$$\Delta Q = \left[\frac{\sum_{in}+d_{i,in}}{2m} - \left(\frac{\sum_{tot}+d_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{d_i}{2m}\right)^2\right],$$

   où $\sum_{in}$ est le nombre d'arêtes à l'intérieur de C, $\sum_{tot}$ est le nombre d'arêtes partant des nœuds de C (donc les liens à l'intérieur de C ainsi que ceux qui partent de nœuds de C vers des nœuds d'autres communautés), d_i est le degré de i et $d_{i,in}$ est le nombre de liens partant de i vers un nœud de C.
2. Ensuite, l'algorithme construit un nouveau réseau dont les nœuds sont les communautés de la phase 1, comme il est illustré sur la figure 1.7 (cet image provient de [5]). L'arête entre deux communautés C_1 et C_2 sera pondéré par le nombre d'arêtes entre les nœuds de C_1 et C_2, lorsqu'il en existe au moins un.

L'algorithme répètera ces deux phases jusqu'à obtenir un maximum de modularité. Puisque le nombre de méta-communautés diminue d'une étape à l'autre, le temps d'exécution de cet algorithme est très raisonnable, comme les auteurs le précisent dans [5]. La phase la plus coûteuse en terme de temps d'exécution de l'algorithme est la première phase de la première itération, i.e. quand toutes les communautés sont des singletons, comme les auteurs l'écrivent dans [5].

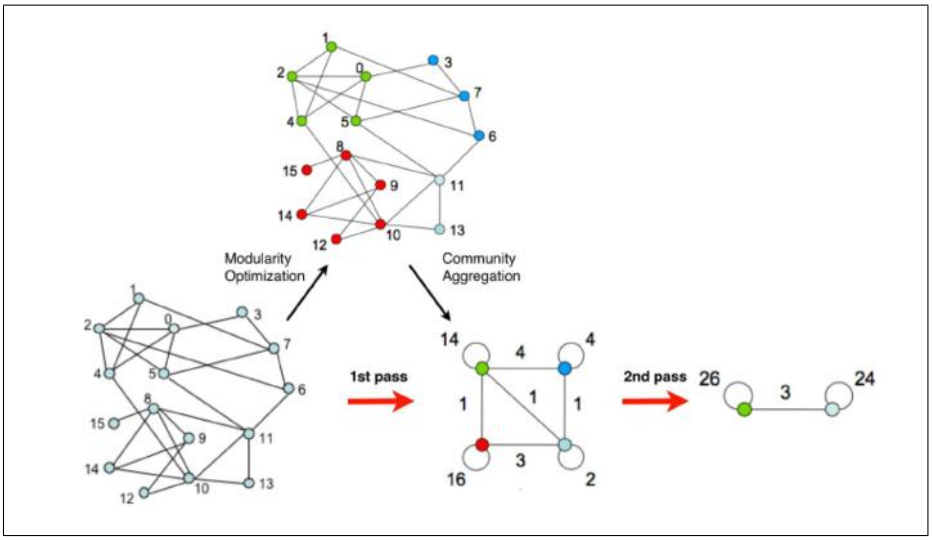

FIGURE 1.7 – Fonctionnement de l'algorithme *Louvain* ([5])

Dans ce mémoire, nous appliquerons une version légèrement modifiée de cet algorithme [18], qui permet de contrôler la taille des communautés en généralisant la notion de modularité :

$$Q_t = \frac{1}{2m} \sum_{i,j} \left[tA_{ij} - \frac{d_i d_j}{2m} \right] \tag{1.4.4}$$

Il est immédiat qu'un choix de $t = 1$ revient à l'équation 1.4.3. On peut montrer que t joue le rôle d'un paramètre de résolution : de petites valeurs de t donnent de petites communautés, et de grandes valeurs de t de grandes communautés.

Pourquoi des communautés ?

Pour cette méthode de prédiction de liens manquants, nous admettrons l'hypothèse de l'existence naturelle de communautés dans un réseau, c'est-à-dire l'existence naturelle de groupes de nœuds qui sont fortement connectés. Avant d'aller plus loin, il est intéressant de se demander l'origine de telles communautés [18]. Cette question a été étudiée par de nombreux chercheurs. Une approche intéressante a été émise par Simon [29], qui nous raconte l'histoire de deux horlogers, *Hora* et *Tempus* :

The watches the men made consisted of about 1,000 parts each. Tempus had so constructed his that if he had one partly assembled and had to put it down to answer the phone say it immediately fell to pieces and had to be reassembled from the elements. The better the customers liked his watches, the more they phoned

him, the more difficult it became for him to find enough uninterrupted time to finish a watch. The watches that Hora made were no less complex than those of Tempus. But he had designed them so that he could put together subassemblies of about ten elements each. Ten of these subassemblies, again, could be put together into a larger subassembly; and a system of ten of the latter subassemblies constituted the whole watch. Hence, when Hora had to put down a partly assembled watch in order to answer the phone, he lost only a small part of his work, and he assembled his watches in only a fraction of the man-hours it took Tempus.

Cette parabole montre que, dans la vie réelle, l'existence de communautés facilite l'émergence de systèmes complexes dans des situations instables. Dans la même veine, Kashtan et al. [15] montrent que les réseaux qui se partitionnent en communautés sont optimales pour effectuer des tâches dans un environnement changeant.

1.4.2 La méthode des communautés

L'idée de notre méthode de prédiction de liens manquants basée sur les communautés est la suivante : nous conjecturons qu'une arête entre une paire de nœuds non-connectés appartenant à la même communauté est plus probable qu'une arête entre deux nœuds qui ne se trouvent pas dans la même communauté. En pratique, pour la méthode des communautés, nous allons tout d'abord appliquer l'algorithme *Louvain* au réseau pour obtenir les communautés des nœuds. Ensuite, nous allons construire la liste des paires de nœuds qui se trouvent dans une même communauté et qui ne sont pas connectés. Cette liste contiendra dès lors les liens prédits par la méthode. Contrairement aux deux autres méthodes que nous avons vues dans les sections précédentes, à savoir les méthodes des triangles et la méthode du produit des degrés, toutes les arêtes de cette liste sont équiprobables, il n'y a donc plus de métrique classant les paires de nœuds. Si nous souhaitons faire moins de prédictions que proposées par la liste, nous en choisirons au hasard. Cependant, cette méthode se distingue de la méthode aléatoire qui, elle aussi, n'attribue aucun score aux arêtes. Nous conjecturons que les arêtes choisies par la méthode des communautés seront des meilleures prédictions, puisque les paires de nœuds considérées proviennent d'une même communauté.
Comme déjà dit plus haut, il s'agit d'une méthode qui n'a pas encore été explorée dans la littérature. Il sera dès lors intéressant de voir la performance de cette méthode-ci comparée aux autres méthodes connues dans la littérature. Finalement, nous testerons la méthode pour différents choix du paramètre ajouté à l'algorithme dans [18].

Chapitre 2

Présentation de la méthode hiérarchique

Dans le chapitre précédent, nous avons présenté quatre méthodes de prédiction de liens manquants dont une méthode aléatoire et trois méthodes basées sur différentes propriétés d'un réseau, à savoir le produit des degrés des nœuds, les triangles et les communautés. Dans ce chapitre, nous introduirons une cinquième méthode, appelée la méthode hiérarchique. Elle a été présentée pour la première fois par Aaron Clauset dans [9]. Puisqu'il s'agit de la méthode la plus sophistiquée et puisque ses fondements mathématiques sont particulièrement intéressants, nous y consacrons un chapitre entier.

Cette méthode se base sur la *hiérarchie* dans un réseau, concept que nous définirons dans une des parties suivantes. Elle modélise un réseau observé par ce qu'on appelle les *graphes aléatoires hiérarchiques* et calcule une fonction de vraisemblance pour ces graphes aléatoires hiérarchiques à partir de ce réseau observé. Ensuite, la méthode cherche le graphe aléatoire hiérarchique qui maximise cette fonction de vraisemblance et produira, à partir de ce "meilleur" graphe aléatoire hiérarchique, des prédictions d'arêtes en associant à chaque arête un score correspondant à la valeur de la fonction de vraisemblance.
La procédé de la maximisation d'une fonction de vraisemblance est une démarche assez connue dans la littérature. Nous trouvons des exemples dans les travaux de Newman et al. et de Ball et al. ([2, 25]).

Dans ce chapitre, nous procéderons de la manière suivante. Nous présenterons tout d'abord le principe de base en déterminant les points centraux du développement de cette méthode. Ensuite, nous aborderons dans les sections suivantes chacun de ces points d'une manière plus détaillée.

2.1 Le principe de base

Comme déjà dit précédemment, cette méthode est basée sur la notion de *hiérarchie* dans un réseau. D'après [8], un *réseau hiérarchique* est un réseau qui se sépare naturellement en plusieurs groupes de nœuds, et ces groupes se séparent en sous-groupes jusqu'à obtenir le niveau des singletons, c'est-à-dire des groupes constitués d'un seul nœud. Nous remarquons que les notions d'hiérarchie et de communautés sont reliées. En effet, les communautés constituent un niveau dans la hiérarchie. Au niveau hiérarchique suivant, on considérera des communautés de communautés, etc.
Considérons un réseau supposé hiérarchique G avec n nœuds. Nous représenterons la hiérarchie de G par un arbre hiérarchique (dendrogramme) D dont les "feuilles" sont les nœuds du graphe G et dont les $n-1$ nœuds internes indiquent la relation hiérarchique entre les feuilles.

Soit r_i un des $n-1$ nœuds internes du dendrogramme. À chaque nœud interne r_i correspond deux groupes de nœuds de G, déterminés par les *sous-arbres à gauche* et *à droite* descendants de r_i, comme nous le voyons dans l'exemple de la figure 2.1. Nous appellerons *le plus bas ancêtre commun* de deux nœuds a et b le nœud interne le plus proche reliant la paire (a, b). Il est évident qu'à chaque paire de nœuds possible correspond un seul ancêtre commun le plus bas. Nous associerons à chaque nœud interne r_i une certaine probabilité p_{r_i} exprimant la probabilité de lien entre les paires de nœuds formées par les nœuds des sous-arbres à gauche et à droite de r_i, la méthode pour calculer les p_{r_i} sera expliquée dans une section ultérieure.
La probabilité que les nœuds a et b soient connectés est notée p_{ab} et est donnée par $p_{ab} = p_r$, où r est le plus bas ancêtre commun de a et b.
Par convention, nous construirons le dendrogramme tel que les valeurs des p_{r_i} vont s'accroitre du bas vers le haut.

Nous pouvons faire le lien avec les graphes aléatoires présentés dans le chapitre précédent. Les graphes aléatoires sont déterminés par un nombre de liens et la probabilité constante d'existence de liens entre toutes les paires de nœuds possibles. Le principe de cette méthode est semblable, cependant nous ne considérons pas des graphes aléatoires, mais des graphes aléatoires *hiérarchiques*. Ceux-ci sont déterminés par un dendrogramme (y compris donc le nombre de nœuds du réseau comme étant le nombre de feuilles) et d'un ensemble de probabilités $\{p_r\}$. Nous noterons un tel graphe aléatoire hiérarchique par $(D, \{p_r\})$.

Illustrons ces nouvelles définitions par un exemple. Considérons un graphe G à 8 nœuds et un dendrogramme D possible.

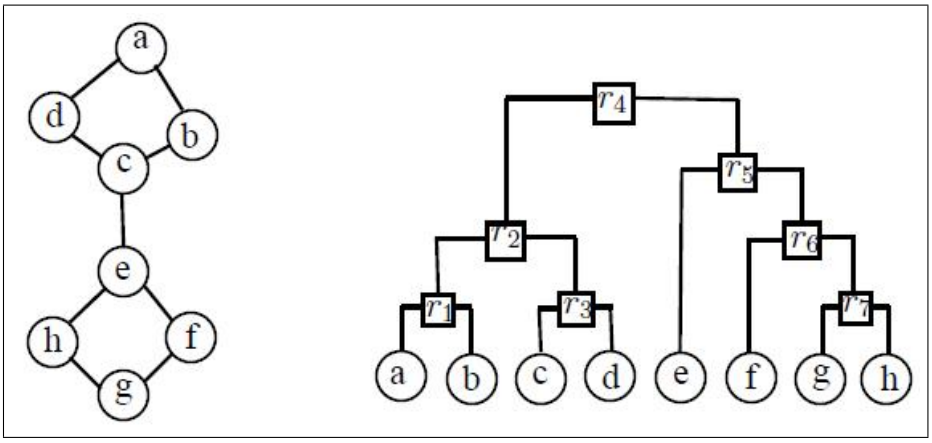

FIGURE 2.1 – Graphe et son dendrogramme associé

Par convention, il faut que $p_{r_4} \leq p_{r_5} \leq p_{r_2} \leq p_{r_1}$. Remarquons que si $\forall i \neq j \; p_{r_i} = p_{r_j}$, nous retrouvons un graphe aléatoire d'Erdös-Rényi présenté dans le premier chapitre.
Le tableau suivant illustre la notion du *plus bas ancêtre commun* pour quelques paires de nœuds.

Paire de nœuds	Plus bas ancêtre commun
(a, b)	r_1
(a, c)	r_2
(c, d)	r_3
(a, e)	r_4
(e, g)	r_5
(f, h)	r_6
(g, h)	r_7

Pour terminer l'analyse de cet exemple, considérons le nœud interne r_5. Le nœud déterminé par le *sous-arbres à gauche de* r_5 est le nœud e et ceux déterminés par le *sous-arbres à droite de* r_5 sont les nœuds f, g et h. Donc, la probabilité d'existence d'un lien entre la paire des nœuds (e, h) est $p_{eh} = p_{r_5}$.

Nous avons jusqu'à présent défini le principe de hiérarchie d'un réseau et les graphes aléatoires hiérarchiques. Dans ce chapitre, nous modéliserons les réseaux par des graphes aléatoires hiérarchiques et nous ferons, à partir de ce modèle hiérarchique, des prédictions d'arêtes. Cependant, une question qui se pose est la suivante. Comment peut-on trouver le/les meilleur(s) graphe(s) aléatoire(s) pour un réseau donné ? Il nous faut donc une mesure qui indique la qualité de chaque

modèle hiérarchique. Cette mesure sera donnée par une fonction de vraisemblance, et dont la construction sera **le premier point central** du développement de cette méthode. Puisque nous nous intéressons aux meilleures modèles hiérarchiques en terme de qualité d'ajustement, nous appliquerons ensuite la méthode de maximisation de vraisemblance afin de trouver tout d'abord l'ensemble de probabilités $\{p_r\}$ maximisant la fonction de vraisemblance. La détermination de cet ensemble de probabilités sera **le deuxième point central** du développement de la méthode hiérarchique. La détermination du dendrogramme maximisant cette fonction sera **le troisième point central**. Ce troisième point sera le point le plus complexe.
Comme pour les autres méthodes de prédiction présentées jusqu'ici, nous terminerons ce chapitre avec un résumé du fonctionnement de cette méthode.
Finalement, notons que nous essayerons, au cours de ce chapitre, d'être le plus clair possible. C'est la raison pour laquelle, chaque fois que nous présenterons un nouveau résultat, nous essayerons de l'illustrer par un exemple afin de rendre la méthode hiérarchique plus compréhensible.

2.2 La fonction de vraisemblance

Notre but est de modéliser un graphe quelconque par un graphe hiérarchique aléatoire. Pour ce faire, nous développerons des méthodes pour trouver le dendrogramme et l'ensemble de probabilités qui ont la plus grande probabilité de correspondre au réseau observé, c'est-à-dire de trouver le dendrogramme qui explique le mieux la hiérarchie dans nos observations ainsi que l'ensemble de probabilités qui correspond le plus à la distribution d'arêtes entre les nœuds. Nous mesurerons cette qualité des graphes aléatoires hiérarchiques par une fonction de vraisemblance.

Soit O le réseau observé. Notre but est de déterminer le graphe hiérarchique aléatoire $(D, \{p_r\})$ tel que la probabilité $P\left((D, \{p_r\})|O\right)$ est maximale. Autrement dit, nous cherchons un dendrogramme D et un ensemble de probabilités $\{p_r\}$ tel que $(D, \{p_r\})$ soit le graphe hiérarchique aléatoire ayant le plus de chance d'expliquer la structure hiérarchique du réseau observé.

Supposons que les graphes hiérarchiques aléatoires sont tous équiprobables *a priori*, c'est-à-dire $P\left((D, \{p_r\})_i\right) = P\left((D, \{p_r\})_j\right), \forall i \neq j$. Si $\mathcal{L}$ désigne la fonction de vraisemblance, nous obtenons, par le théorème de Bayes, le résultat suivant.

$$\begin{aligned} P\left((D, \{p_r\})|O\right) &= \frac{P\left(O|(D, \{p_r\})\right) P\left(D, \{p_r\}\right)}{P\left(O\right)} \\ &\sim P\left(O|(D, \{p_r\})\right) = \mathcal{L}(D, p_r). \end{aligned}$$

Donc, en supposant que tous les graphes hiérarchiques aléatoires sont *a priori*

équiprobables (ce qui rend le terme $P(D, \{p_r\})$ constant), nous obtenons que la probabilité qu'un modèle donné $(D, \{p_r\})$ soit la bonne explication des observations est proportionnelle à la probabilité *a posteriori* $P(O|(D, \{p_r\}))$ ou à sa fonction de vraisemblance avec laquelle le modèle génère le réseau observé. Grâce à cette observation, nous ne devons plus calculer des probabilités afin de déterminer la qualité des graphes hiérarchiques aléatoires mais les fonctions de vraisemblance. Un autre avantage est que nous pouvons utiliser la méthode du Maximum de Vraisemblance afin de trouver les "meilleures" graphes hiérarchiques aléatoires.

Établissons l'expression de la fonction de vraisemblance pour un graphe aléatoire hiérarchique.
Soit E_r le nombre de liens de G qui relient des nœuds admettant r comme plus bas ancêtre commun, L_r et R_r le nombre de feuilles du sous-arbre respectivement à gauche et du sous-arbre à droite de r. Alors, la vraisemblance de (D, p_r) est donnée par

$$\mathcal{L}(D, p_r) = \prod_{r \in D} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r}, \tag{2.2.1}$$

où nous adoptons la convention $0^0 = 1$.
Afin de mieux comprendre cette fonction de vraisemblance, regardons de plus près ce qui se passe à chaque nœud interne r. Le nombre de paires (a, b) possibles tel que a (resp. b) soit une feuille du sous-arbre à gauche (resp. à droite) de r vaut $L_r R_r$. Entre ces $L_r R_r$ paires, il y en a E_r qui sont connectées avec une probabilité p_r en tenant compte de nos observations. D'où le terme $p_r^{E_r}$ dans l'équation 2.2.1. Puisqu'il y en a E_r connexions entre des paires des nœuds a et b ayant r comme ancêtre le plus bas commun, les $L_r R_r - E_r$ paires de nœuds restants ne sont pas connectés. C'est la raison pour laquelle il faut multiplier par le terme $(1 - p_r)^{L_r R_r - E_r}$. Il s'agit d'une fonction de vraisemblance à deux paramètres, à savoir le dendrogramme D et l'ensemble des probabilités $\{p_r\}$. Nous maximiserons et illustrerons la fonction de vraisemblance dans les deux sections suivantes.

2.3 L'ensemble p_r maximisant $\mathcal{L}$

Puisqu'il est, comme nous le verrons par la suite, plus facile de calculer l'ensemble de probabilités maximisant la fonction de vraisemblance et puisque nous aurons besoin de cet ensemble maximal lors de la détermination du meilleur dendrogramme, nous commençons par la recherche de l'ensemble de probabilités maximisant la fonction de vraisemblance. Il s'agit d'un principe bien connu en mathématiques : nous avons déterminé une fonction de vraisemblance à deux paramètres,

et, afin de maximiser cette fonction, nous annulons les dérivées partielles de la fonction de vraisemblance par rapport aux paramètres.

Au lieu de calculer le maximum de la fonction de vraisemblance $\mathcal{L}$, nous calculerons le maximum du logarithme de la vraisemblance, ce qui sera plus facile à déterminer. Ensuite, nous annulerons la dérivée par rapport à p_r de la log-vraisemblance afin de trouver les valeurs $\overline{p}_r$ qui maximisent $log\mathcal{L}$. Il est claire que les valeurs maximales de la log-vraisemblance maximisent aussi la fonction de vraisemblance, puisque la fonction logarithmique est une fonction monotone. Remarquons que, dans toute cette section, la fonction de vraisemblance est calculée pour n'importe quel dendrogramme D. En partant de 2.2.1, nous obtenons

$$\begin{aligned}\mathcal{L}(D, p_r) &= \prod_{r\in D} p_r^{E_r}(1-p_r)^{L_rR_r-E_r} \\ &= p_r^{\sum_{r\in D} E_r}(1-p_r)^{\sum_{r\in D} L_rR_r-E_r}. \end{aligned} \tag{2.3.1}$$

En prenant le logarithme de l'équation 2.3.1, elle devient

$$\log \mathcal{L}(D, p_r) = \sum_{r\in D} E_r \log(p_r) + \left(\sum_{r\in D} L_rR_r - E_r\right)\log(1-p_r). \tag{2.3.2}$$

En dérivant cette équation 2.3.2 par rapport à p_r, tous les termes des sommes dans l'équation 2.3.2 s'annuleront sauf le terme correspondant à p_r.

$$\frac{\partial}{\partial p_r}\log \mathcal{L}(D, p_r) = \frac{E_r}{p_r} - \frac{L_rR_r - E_r}{1-p_r}. \tag{2.3.3}$$

Enfin, afin de trouver les valeurs $\overline{p}_r$ qui maximisent la fonction de vraisemblance, annulons l'équation 2.3.3.

$$\begin{aligned} \frac{E_r}{p_r} - \frac{L_rR_r - E_r}{1-p_r} &= 0 \\ \Leftrightarrow \frac{E_r(1-p_r) - (L_rR_r - E_r)p_r}{p_r(1-p_r)} &= 0 \\ \Leftrightarrow E_r(1-p_r) - (L_rR_r - E_r)p_r &= 0 \\ \Leftrightarrow \overline{p_r} = \frac{E_r}{L_rR_r}. \end{aligned} \tag{2.3.4}$$

Ce résultat est intuitif, car il s'agit du nombre de liens possibles entre les paires de nœuds (E_r) divisé par le nombre de paires possibles (L_rR_r).

Puisque nous disposons maintenant d'une expression analytique des probabilités maximisant la fonction de vraisemblance, nous pouvons déterminer l'expression analytique de cette fonction pour ces valeurs maximales.

$$\begin{aligned}\mathcal{L}(D) &= \prod_{r\in D}\left[\overline{p}_r^{E_r}(1-\overline{p}_r)^{L_rR_r-E_r}\right]^{\frac{L_rR_r}{L_rR_r}} \\ &= \prod_{r\in D}\left[\overline{p}_r^{\frac{E_r}{L_rR_r}}(1-\overline{p}_r)^{\frac{L_rR_r-E_r}{L_rR_r}}\right]^{L_rR_r} \\ &= \prod_{r\in D}\left[\overline{p}_r^{\overline{p}_r}(1-\overline{p}_r)^{1-\overline{p}_r}\right]^{L_rR_r}. \end{aligned} \tag{2.3.5}$$

L'équation 2.3.5 est l'expression de la fonction de vraisemblance calculée pour l'ensemble de probabilités maximisant cette fonction. Elle ne dépend donc plus que d'un seul paramètre, à savoir le dendrogramme. Analysons cette expression 2.3.5 de manière plus détaillée.

En calculant le logarithme de l'équation 2.3.5, nous obtenons le résultat suivant.

$$\log\mathcal{L}(D) = -\sum_{r\in D} L_rR_rh(\overline{p}_r), \tag{2.3.6}$$

où $h(p) = -p\log p-(1-p)\log(1-p)$ est appelée la fonction d'entropie de Gibbs-Shannon ([9]). Nous déduisons de la figure 2.2 que chaque terme $-L_rR_rh(\overline{p}_r)$ est maximal quand $\overline{p}_r$ est proche de 0 ou 1. En effet, si la valeur de $\overline{p}_r$ pour un nœud interne quelconque r est proche de 0 ou proche de 1, cela signifie que l'existence de connexion entre les nœuds i et j ayant r comme plus bas ancêtre commun est très probable (proche de 1) ou très peu probable (proche de 0). La fonction de vraisemblance est minimale si $\overline{p}_r = 0.5$. En effet, dans ce cas, la probabilité d'existence d'un lien entre i et j équivaut à la probabilité de non-existence de lien entre les deux nœuds. Dans le cas maximal, nous pouvons prédire avec un grand niveau de certitude la (non-)existence d'un lien entre i et j, tandis que dans le cas minimal, il y a une chance sur deux qu'il existe un lien ou non.
En d'autre mots, les dendrogrammes d'une très grande vraisemblance sont ceux qui partitionnent les nœuds en groupe parmi lesquels les connexions sont soit très rares, soit très fréquentes.

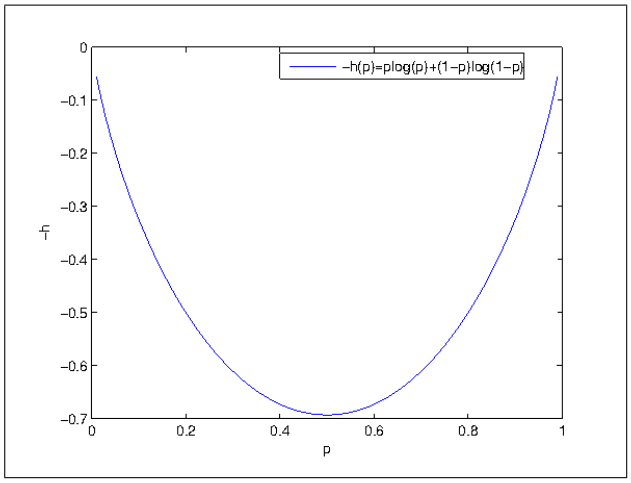

FIGURE 2.2 – Fonction d'entropie en fonction de $\overline{p}_r$

Illustrons les concepts que nous venons de voir par un exemple simple. Supposons que nous observons le réseau G de 6 nœuds $\{a, b, c, d, e, f\}$ et de 7 liens de la figure 2.3. La figure 2.4 affiche deux dendrogrammes possibles pour ce réseau. Nous calculerons par la suite les valeurs correspondantes de la fonction de vraisemblance afin de décider lequel des deux dendrogramme représente le mieux la hiérarchie du réseau G.

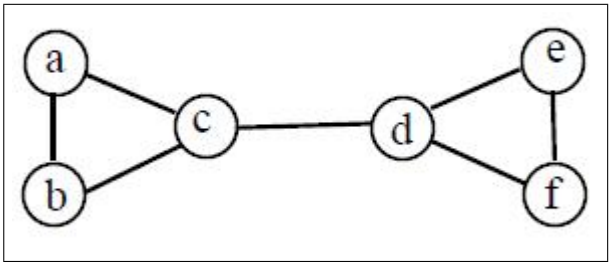

FIGURE 2.3 – Exemple d'un réseau

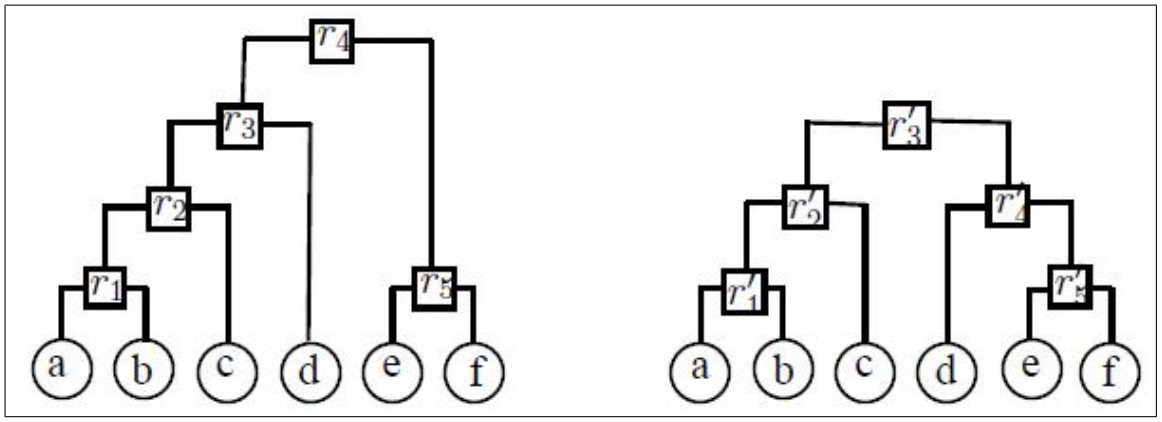

FIGURE 2.4 – Deux dendrogrammes possibles

Le tableau suivant donne les valeurs de E_r, L_r, R_r et $\overline{p}_r$ pour le dendrogramme à gauche de la figure 2.4. Rappelons que $\overline{p}_r = \frac{E_r}{L_r R_r}$.

noeud interne	E_r	L_r	R_r	$\overline{p}_r$
r_1	1	1	1	1
r_2	2	2	1	1
r_3	1	3	1	$\frac{1}{3}$
r_4	2	4	2	$\frac{1}{4}$
r_5	1	1	1	1

Considérons par exemple le nœud r_3. L'ensemble des nœuds engendrés par le sous-arbre à gauche de r_3 est $\{a, b, c\}$, celui engendré par le sous-arbre à droite est $\{d\}$. Nous en déduisons que $L_{r_3} = 3$ et $R_{r_3} = 1$. De plus, nous voyons que le nombre de paires possibles est égale à $L_{r_3} R_{r_3} = 3$ (à savoir les paires (a, d), (b, d), (c, d)). Pour déterminer la valeur de E_{r_3}, regardons le réseau observé de la figure 2.4. E_r est le nombre de liens entre l'ensemble des nœuds du sous-arbre à gauche et celui du sous-arbre à droite de r, ou autrement dit, le nombre de paires de nœuds pour lesquels il existe une connexion. Dans cet exemple, nous observons une seule connexion entre les paires des nœuds des sous-arbres à gauche et à droite, à savoir le lien entre la paire (c, d). D'où $E_{r_3} = 1$ et nous en déduisons que $p_{r_3} = \frac{1}{3\times 1} = \frac{1}{3}$. En procédant de la même manière, nous calculons les valeurs des $p_{r'}$. Les dendrogramme avec les valeurs des probabilités maximisant les fonctions de vraisemblances correspondantes sont présentés dans la figure 2.5.

Calculons les fonctions de vraisemblance pour chaque dendrogramme (rappelons que nous avons pris la convention que $0^0 = 1$).
Pour le premier dendrogramme, nous obtenons, en appliquant l'équation 2.3.5 à ce dendrogramme, l'équation suivante.

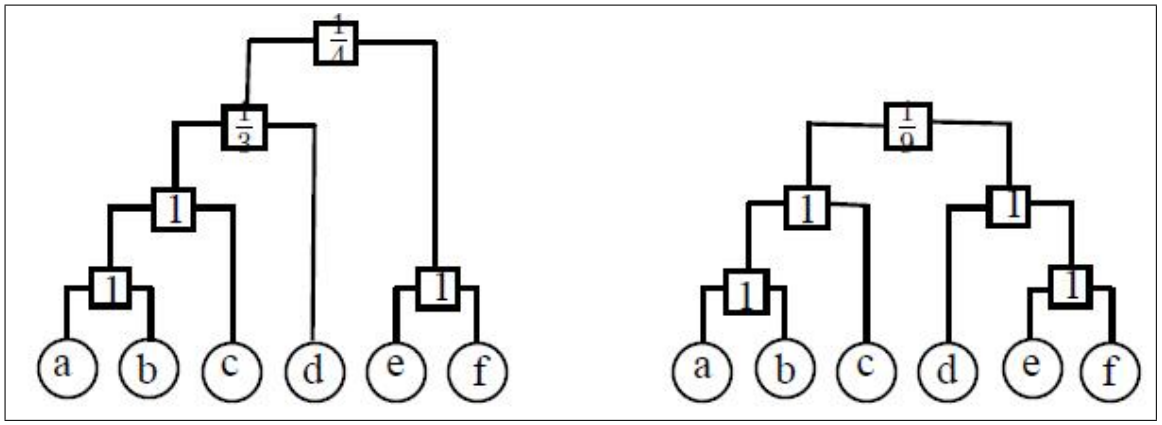

FIGURE 2.5 – Dendrogrammes avec p_r optimales

$$\begin{aligned}\mathcal{L}(D_1) &= (1^1 \times 0^0)(1^1 \times 0^0)^2 \left(\left(\frac{1}{3}\right)^{\frac{1}{3}} \times \left(\frac{2}{3}\right)^{\frac{2}{3}}\right)^3 \left(\left(\frac{1}{4}\right)^{\frac{1}{4}} \times \left(\frac{3}{4}\right)^{\frac{3}{4}}\right)^8 (1^1 \times 0^0) \\ &= 0.00165\ldots\end{aligned}$$

De la même façon, nous trouvons pour le deuxième dendrogramme

$$\begin{aligned}\mathcal{L}(D_2) &= \left(\frac{1}{9} \times \left(\frac{8}{9}\right)^8\right) \\ &= 0.0433\ldots\end{aligned}$$

Nous en déduisons que le deuxième dendrogramme est beaucoup plus probable, ce qui nous parait assez intuitif. En effet, le premier dendrogramme sépare le nœud d au niveau $\frac{1}{4}$ des nœuds e et f, alors que le deuxième le sépare au niveau 1 des deux autres. Cela veut dire que, d'après le premier dendrogramme, la probabilité d'existence d'un lien entre e et d est égale à $\frac{1}{4}$, mais d'après le deuxième dendrogramme, cette probabilité est égale à 1.
Un autre exemple qui conforte le choix du deuxième dendrogramme est la paire de nœuds (a, e). D'après le premier dendrogramme, ces nœuds sont connectés avec une probabilité de $\frac{1}{4}$, ce qui est beaucoup plus grand que la probabilité associée à cette paire de nœuds par le deuxième dendrogramme, à savoir $\frac{1}{9}$.
Dès lors, pour cet exemple simple, le dendrogramme à gauche de la figure 2.5 a le plus de chances d'expliquer les observations que l'autre dendrogramme.

Nous venons de trouver analytiquement l'expression de l'ensemble de probabilités maximisant la fonction de vraisemblance pour un réseau - c'est-à-dire pour

une hiérarchie - fixé. Dans la section suivante, nous montrerons comment trouver le "meilleur" dendrogramme, i.e. le dendrogramme le plus probable ayant engendrer le réseau observé.

2.4 Le dendrogramme D maximisant $\mathcal{L}$

Trouver le dendrogramme D qui maximise la fonction de vraisemblance est une tâche plus difficile que la détermination des valeurs maximales des $\{p_r\}$, puisque nous ne disposons pas d'expressions analytiques pour les dendrogrammes. C'est la raison pour laquelle la démarche à suivre afin de trouver le(s) dendrogramme(s) expliquant le mieux la hiérarchie d'un réseau observé diffère de la détermination de l'ensemble des probabilités maximisant la fonction de vraisemblance : Nous construirons tout d'abord une chaine de Markov parcourant l'espace de tous les dendrogrammes possibles et, ensuite, nous établirons une règle afin de ne pas devoir comparer tous les dendrogrammes entre eux, mais en sélectionner directement les meilleurs.

2.4.1 Les chaines de Markov

Les définitions de cette section sont reprises de [3], lecture recommandée pour des explications plus approfondies. Nous nous limitons dans ce travail aux explications nécessaires pour comprendre la suite.
Une chaine de Markov est un processus stochastique à temps et à espace d'états discrets. Considérons un processus stochastique qui produit la séquence des variables aléatoires suivantes :

$$\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, \ldots\}.$$

Supposons que X_7 ne dépend que de X_6, X_6 ne dépend que de X_5, X_5 ne dépend que de X_4 etc. En général, si $\forall i, j, n$

$$P(X_{n+1} = j | X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j | X_n = i_n), \quad (2.4.1)$$

alors nous appelons ce processus une chaine de Markov.

La probabilité conditionnelle $P(X_{n+1} = i_{n+1} | X_n = i_n)$ signifie la probabilité qu'un procès se trouvant en i_n à l'instant n se trouvera en i_{n+1} à l'instant suivant $n + 1$. Cette probabilité est appelée *probabilité de transition.*
L'hypothèse 2.4.1 est *l'hypothèse de Markov.* Il s'agit d'une hypothèse qui est vérifiée dans beaucoup d'événements de la vie quotidienne et les chaînes de Markov

offrent un bon outil de modélisation de ces événements. Pour plus de détails, voir [4].

Dans notre cas, les états de la chaine de Markov sont des dendrogrammes. Nous devons définir une transition d'un dendrogramme à l'autre. Remarquons que, sans perdre de généralité, à chaque nœud interne r sont associés trois sous-arbres, à savoir ses deux enfants et son ancêtre. Pour chaque nœud interne, il y a trois configurations possibles, comme nous pouvons le voir sur la figure 2.6. Notons que, en général, les triangles désignent ici des sous-arbres et pas des nœuds.

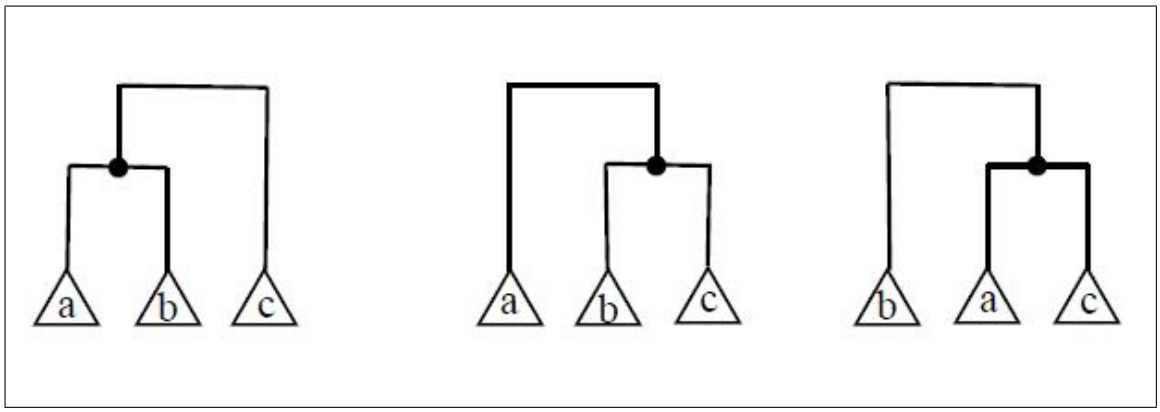

FIGURE 2.6 – Configurations possibles pour un nœud interne

Dès lors, dans la chaine de Markov, nous passerons d'un dendrogramme à l'autre en choisissant tout d'abord de manière aléatoire un nœud interne et en choisissant après une des deux configurations possibles pour ce nœud.

La chaine de Markov que nous venons de développer est un concept abstrait parcourant l'espace entier de tous les dendrogrammes possibles. Nous la modéliserons par un algorithme de la classe *Markov Chain Monte Carlo* auquel nous ajoutons la règle de *Metropolis-Hastings* afin d'assurer que l'algorithme ne favorise que les dendrogrammes d'une valeur de vraisemblance élevée. Nous expliquerons ces deux derniers principes dans les sections suivantes.

2.4.2 Markov Chain Monte Carlo

En général, un algorithme Monte Carlo est un algorithme aléatoire d'une complexité faible qui peut fournir une valeur fausse avec une probabilité faible. Si nous appliquons cet algorithme plusieurs fois, la probabilité que le résultat fourni soit

faux devient très petite [1].

Illustrons cette démarche par un exemple très célèbre : l'algorithme de Monte Carlo pour la détermination du nombre π.
L'idée de base est très simple : Nous considérons un carré dont les côtés sont de longueur 1 et nous traçons un quart d'un cercle de rayon 1 dans ce carré, comme dans la figure 2.7 [2].

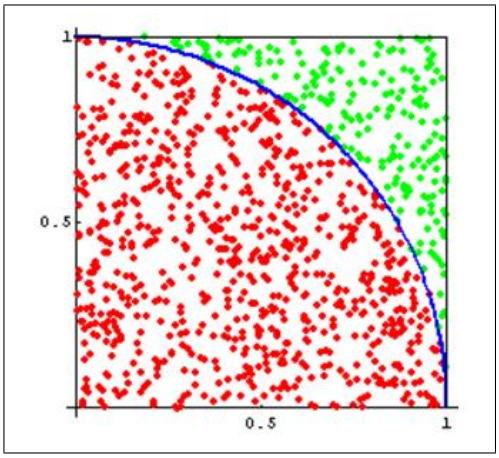

FIGURE 2.7 – Illustration de l'algorithme d'approximation de π.

Lançons des billes dans ce carré et calculons le rapport entre le nombre de billes se trouvant dans le quart du cercle et le nombre de billes totales. En théorie, nous avons que

$$\frac{\text{Aire}(1/4 \text{ cerle})}{\text{Aire}(\text{carré})} = \frac{\frac{r^2\pi}{4}}{r^2} = \frac{\pi}{4}.$$

Donc, si nous lancions un nombre infini de billes, le rapport obtenu vaudrait exactement $\frac{\pi}{4}$.

Nous comprenons bien le caractère de Monte Carlo de cet algorithme : à n'importe quelle étape, le rapport peut être différent ou même loin de $\frac{\pi}{4}$, mais si nous appliquons l'algorithme plusieurs fois (c'est-à-dire si nous lançons plus de billes), le rapport se rapprochera de $\frac{\pi}{4}$.

1. `http://www.uni-protokolle.de/Lexikon/Monte-Carlo_Algorithmus.html`, page consulté le 1 mai 2012
2. `http://math.fullerton.edu/mathews/n2003/montecarlopimod.html`, page consulté le 1 mai 2012

Comme nous l'avons vu, afin de trouver un dendrogramme acceptable, nous construirons une chaine de Markov de dendrogrammes avec les transitions définies plus haut. Afin de construire un algorithme de Monte Carlo, nous devons assurer que la probabilité de générer un dendrogramme fautif (c'est-à-dire un dendrogramme n'expliquant pas bien la structure hiérarchique dans les données) est faible. Autrement dit, nous devons ajouter un critère assurant la convergence de l'algorithme vers un dendrogramme acceptable. Ce critère sera une règle de décision appelée la règle de *Metropolis-Hastings*.

2.4.3 Règle de Metropolis-Hastings

Cette section est basée sur [7]. Soient deux dendrogrammes D et D' tels que D' est obtenue à partir de D après une étape de notre chaine de Markov. Autrement dit, D' est obtenu en modifiant un seul nœud interne de D en appliquant la transition décrite dans une autre partie de ce rapport. La règle d'acceptance du nouveau dendrogramme D' est appelée la règle de Metropolis-Hastings :

- Si

$$\Delta \log \mathcal{L} = \log \mathcal{L}(D') - \log \mathcal{L}(D) \geq 0,$$

  nous accepterons la transition $D \to D'$
- Sinon nous accepterons la transition avec une probabilité

$$\exp\left(\Delta \log \mathcal{L}\right) = \frac{\mathcal{L}(D')}{\mathcal{L}(D)}$$

- Si nous n'acceptons pas la nouvelle transition, nous gardons le dendrogramme D.

La première règle stipule que nous acceptons le nouveau dendrogramme D' s'il apporte un gain de vraisemblance par rapport au dendrogramme D. Donc, cette première règle nous assure que le nouveau dendrogramme considéré est au moins aussi probable pour la structure hiérarchique de notre réseau observé que l'ancien. La deuxième règle assure que nous ne négligeons pas les dendrogrammes d'une vraisemblance plus faible. Cependant, si la différence des vraisemblances de D et de D' est très grande (i.e. $\mathcal{L}(D) >> \mathcal{L}(D')$), alors la probabilité d'accepter D' est très faible.

Une première approche serait de chercher le meilleur dendrogramme, c'est-a-dire d'appliquer l'algorithme beaucoup de fois et de considérer le dernier dendrogramme comme étant le meilleur. Cependant, dans [9], les auteurs ont choisi d'envisager plusieurs bons dendrogrammes à la place d'en choisir un seul. En effet,

par malheur, le dernier dendrogramme de l'algorithme pourrait être moins bon que le précédent. En considérant plusieurs dendrogrammes produits par l'algorithme, nous évitons ce problème.

Le grand avantage de cette règle de Metropolis-Hastings est que, à l'équilibre, le nombre de fois que chaque dendrogramme a été parcouru par l'algorithme est proportionnel à sa vraisemblance. En effet, examinons ce qui se passe d'une transition à l'autre. Tout d'abord, nous remarquons que l'ensemble de tous les dendrogrammes possibles pour un réseau donné est un *ensemble ergodique*[3]. Dès lors, si nous appliquons l'algorithme un grand nombre de fois, nous pouvons être assurés d'avoir parcouru tout dendrogramme de l'espace des dendrogrammes possibles, et cela indépendamment du dendrogramme de départ.

Sur la figure 2.8, nous avons deux états (dendrogrammes) possibles i et j, et la probabilité que, pendant l'exécution de l'algorithme, on considère le dendrogramme i (resp. j), est donnée par P_i (resp. P_j).

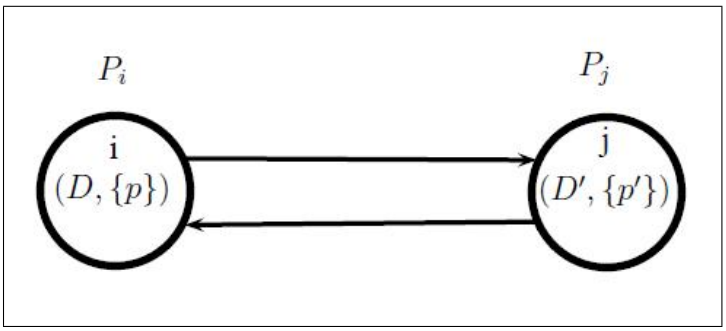

FIGURE 2.8 – Transition d'un état i vers j et vice-versa

Supposons sans perdre de généralité que $\mathcal{L}_j < \mathcal{L}_i$. Dès lors, si nous partons de j, la transition vers i se fera avec une probabilité 1, étant donné que $\mathcal{L}_j < \mathcal{L}_i$, $\Delta \log \mathcal{L} = \mathcal{L}_i - \mathcal{L}_j \geq 0$. Si nous partons de i, la transition vers j se fera avec une probabilité égale à $\frac{\mathcal{L}_j}{\mathcal{L}_i}$.

À l'état d'équilibre, il faut que la balance détaillée soit respectée, c'est-à-dire que, pour n'importe quelle paire d'états i et j, le nombre de transitions de i à j est égal au nombre de transitions de j à i. Le flux de probabilité entre i et j est donc égal au flux de probabilité entre j et i. Illustrons cette situation d'équilibre par un marcheur aléatoire qui déambule d'un état à l'autre suivant les transitions.

3. C'est-à-dire de n'importe quel dendrogramme de départ, nous pouvons arriver à n'importe quel autre dendrogramme.

Lorsque le processus stochastique a atteint l'équilibre, le nombre de fois que le marcheur aléatoire fait un pas de i vers j est égal au nombre de fois qu'un tel pas se fait de j vers i.

La probabilité de passer de i à j est la probabilité de se trouver en i multipliée par la probabilité de faire la transition vers j, c'est-à-dire

$$P(i) \times P(i \to j) = P(i) \times \frac{\mathcal{L}_j}{\mathcal{L}_i},$$

et, de manière similaire, la probabilité de passer de j à i vaut

$$P(j) \times P(j \to i) = P(j) \times 1.$$

Dès lors, à l'équilibre, il faut que ces deux flux de probabilités soient équivalents, c'est-à-dire

$$P_i \times \frac{\mathcal{L}_j}{\mathcal{L}_i} = P_j.$$

Les seules solutions de cette équations sont

$$P_i \sim \mathcal{L}_i$$
$$P_j \sim \mathcal{L}_j.$$

Dès lors, la règle de *Metropolis-Hastings* implique que la probabilité d'observer un dendrogramme est proportionnelle à sa vraisemblance.

Avant de terminer le développement de la méthode hiérarchique, illustrons les derniers concepts par un exemple. Reconsidérons les deux dendrogrammes de l'exemple précédent.

Appelons le dendrogramme à gauche i et le dendrogramme à droite j. Nous avons vu que $\mathcal{L}_i = 0.00165\ldots$ et $\mathcal{L}_j = 0.0433\ldots$. Dès lors, si nous partons de i, la transition $i \to j$ sera acceptée avec une probabilité 1. Si nous partons de j, la transition $j \to i$ sera acceptée avec une probabilité égale à

$$\frac{\mathcal{L}_i}{\mathcal{L}_j} \approx 0.038,$$

ce qui est très faible.

Donc, dans cet exemple, nous remarquons qu'il est très improbable d'accepter les dendrogrammes ayant une vraisemblance faible. Sur notre exemple, à l'état d'équilibre, nous sommes assurés que l'algorithme aura envisagé presque 40 fois

plus souvent le dendrogramme j que le dendrogramme i, et dès lors il est très peu probable que nous considérerons le dendrogramme j. Il est évident que nous pouvons envisager des dendrogrammes encore "plus mauvais" que le dendrogramme i, qui auront donc une probabilité d'être considérés par l'algorithme encore plus faible.

Finalement, nous connaissons tous les outils nécessaires pour développer la méthode hiérarchique de prédiction de liens manquants.

2.5 Algorithme de prédiction de liens manquants

Dans les parties précédentes, nous avons vu tous les concepts théoriques de la méthode hiérarchique. Rappelons encore une fois que nous n'avons pas implémenté cette méthode nous-mêmes. Les codes de cette méthode sont mis à notre disposition gratuitement par Aaron Clauset sur son site Internet[4]. Nous nous limitons dans le cadre de ce mémoire à la compréhension et à l'application de cette méthode.

La méthode de prédiction combine les concepts abordés précédemment de la manière suivante.

1. Choisir un graphe hiérarchique aléatoire comme état initial de la chaine de Markov.
2. Appliquer l'algorithme MCMC jusqu'à l'équilibre est atteint.
3. Sélectionner des dendrogrammes construits par la chaine de Markov par une règle arbitraire.
4. Pour chaque paire de nœuds i et j qui ne sont pas connectés dans le graphe donné, calculer la probabilité moyenne $< p_{ij} >$ en calculant la moyenne des probabilités p_{ij} de tous les dendrogramme D.
5. Ordonner les paires (i, j) dans l'ordre décroissant des probabilités moyennes $< p_{ij} >$. Prédire l'existence d'un lien entre les paires placées le plus haut dans ce tableau.

Dès lors, dans le deuxième point, l'algorithme crée beaucoup de dendrogrammes possibles qui expliquent bien la hiérarchie observée grâce à la règle de Metropolis-Hastings. Nous ne considérons pas tous ces dendrogrammes à cause de leur nombre élevé, c'est pourquoi, dans le troisième point, nous sélectionnons de manière arbitraire un sous-ensemble de dendrogrammes.

4. `http://tuvalu.santafe.edu/~aaronc/`

Comme toutes les méthodes non-aléatoires, excepté la méthode des communautés, cette méthode associe un score aux arêtes prédites qui sera la probabilité moyenne de leur existence. Il sera intéressant de vérifier les résultats de Clauset ([9]) sur le même réseau, mais aussi d'appliquer sa méthode hiérarchique à d'autres jeux de données et de la comparer avec les méthodes présentées dans le chapitre précédent. Le chapitre suivant résume les résultats de ces expériences.

Chapitre 3

Application des méthodes de prédiction de liens manquants

Ce chapitre contiendra la partie applicative de ce mémoire. Nous appliquerons les méthodes de prédiction de liens manquants que nous avons présentées dans les chapitres précédentes à cinq réseaux sociaux de la vie réelle.
Nous commencerons ce chapitre par une brève introduction de la statistique utilisée pour mesurer la qualité des prédictions, à savoir la statistique appelée *AUC*. Il suit une partie dans laquelle nous présentons les cinq réseaux sociaux et nous étudierons quelques propriétés de ceux-ci. Ensuite, dans les cinq parties suivantes, nous présenterons les résultat des applications des méthodes de prédiction de liens manquants pour chaque jeu de données. Nous présenterons, dans chaque partie, d'abord les résultats pour la méthode des communautés avec différents choix du paramètre et, ensuite, nous présenterons les résultats de toutes les méthodes. Puis, dans une section récapitulative, nous résumerons les résultats trouvés dans les sections précédentes. Nous clôturerons ce chapitre par une application qui montre l'utilité des méthodes de prédiction de liens manquants dans la vie réelle.

3.1 La statistique *AUC*

Cette partie est principalement basée sur [10]. Afin de savoir comparer les méthodes de prédiction de liens manquants, nous avons décidé d'utiliser la statistique *AUC* qui est une mesure intuitive et bien compréhensible. Nous présenterons tout d'abord les concepts de *true positive* ainsi que *false positive*, *true negative* et *false negative*. Ensuite, nous définirons la *true positive rate* et la *false positive rate* qui nous permettent de tracer le graphe de *ROC* et, finalement, nous définirons la mesure *AUC*.

3.1.1 Les *TP*, *FP*, *TN* et *FN*

Supposons que nous envisageons un réseau G et nous en enlevons 10% des arêtes pour construire le nouveau réseau G'. Sur ce réseau réduit, nous appliquons une des méthodes de prédiction de liens manquants présentées dans les chapitres précédents. La méthode nous fournira un nombre souhaité par l'utilisateur de prédictions de liens pour ce réseau. Les concepts de *TP*, *FP*, *TN* et *FN* définissent la nature de ces arêtes prédites. Une arête prédite par la méthode de prédiction est un

- *TP* (*true positive*) s'il s'agit d'une arête enlevée du réseau de départ, c'est donc une bonne prédiction,
- *FP* (*false positive*) s'il s'agit d'une arête qui n'est pas présente dans le réseau de départ, il s'agit dès lors d'une mauvaise prédiction.

Les *TN* et *FN* sont définis de la manière suivante :

- *TN* (*true negative*) est le nombre d'arêtes pas présentes dans le réseau de départ et qui ne sont pas prédites par la méthode,
- *FN* (*false negative*) est le nombre d'arêtes non-prédites par la méthode mais qui sont enlevées du réseau de départ.

Nous résumons ces définitions dans le tableau suivant en tout sachant que la méthode de prédiction propose toujours des arêtes non-présentes dans G'. Dès lors, une arête proposée par la méthode est soit présente dans G, dans ce cas il s'agit d'une des arêtes enlevées de G, ou soit pas présente dans G, et dans ce cas il s'agit d'une mauvaise prédiction.

	arête enlevée	**arête pas présente dans** G
arête prédite	TP	FP
arête pas prédite	FN	TN

3.1.2 Les *TPR* et *FPR*

Nous définissons la *TPR* (*true positive rate*) comme la fraction des bonnes prédictions parmi toutes les arêtes enlevées du réseau, c'est-à-dire

$$TPR = \frac{TP}{TP + FN}. \tag{3.1.1}$$

La *FPR* (*false positive rate*) par contre est la fraction des mauvaises prédictions parmi toutes les arêtes non-présentes dans le réseau de départ et non-enlevées de celui ci, et donc

$$FPR = \frac{FP}{FP + TN}. \tag{3.1.2}$$

3.1.3 La *ROC*

L'espace *ROC* (*receiver operating characteristic*) est un espace de dimension 2 et est définie par $(FPR \times TPR)$, où FPR et TPR désignent des vecteurs dont les composantes ont été calculées par les équations 3.1.2 et 3.1.1 respectivement. La courbe de *ROC* affiche donc la TPR en fonction de la FPR. Concrétisons ce concept par un exemple théorique.

Envisageons un réseau G duquel nous enlevons 10% des arêtes. Appliquons la méthode aléatoire à ce réseau G' pour différents nombres de prédictions d'arêtes et enregistrons les valeurs des TPR et FPR dans deux vecteurs différents. Ensuite, nous traçons le graphe des valeurs des TPR en fonction des FPR. Puisque, comme déjà dit dans la partie correspondante à la méthode aléatoire dans le chapitre 2, toutes les paires de nœuds non-connectés sont équiprobables, la fraction des mauvaises prédictions est équivalente à la fraction des bonnes prédictions et nous obtenons dès lors en moyenne une droite diagonale comme indiqué sur la figure 3.1.
Si nous appliquons par contre une méthode plus sophistiquée qui fournit des meilleurs résultats, la valeur correspondante de la *TPR* sera plus grande que la valeur correspondante de la *FPR*, et cette mesure se trouvera dès lors au-dessus de la droite diagonale correspondante à la méthode aléatoire, comme le point a l'indique sur la figure 3.1. Le point b sur le même graphe est le résultat pour une méthode qui fournit des résultats encore moins bons que la méthode aléatoire.

Concrètement, afin de tester l'efficacité d'un algorithme de liens manquants, nous procédons de la manière suivante.

- Nous enlevons des liens d'un réseau de départ.
- Dans une boucle, nous varions le nombre de liens prédits par une des méthodes. Pour chaque nombre de prédictions, nous calculons les *TP*, *FN*, *FP* et les *TN*.
- Nous enregistrons les valeurs de la *TPR* et *FPR* dans un vecteur.
- En dehors de la boucle, nous traçons le vecteur des *TPR* en fonction du vecteur contenant les *FPR*.

Nous obtiendrons par exemple une courbe de *ROC* comme sur la figure 3.1.

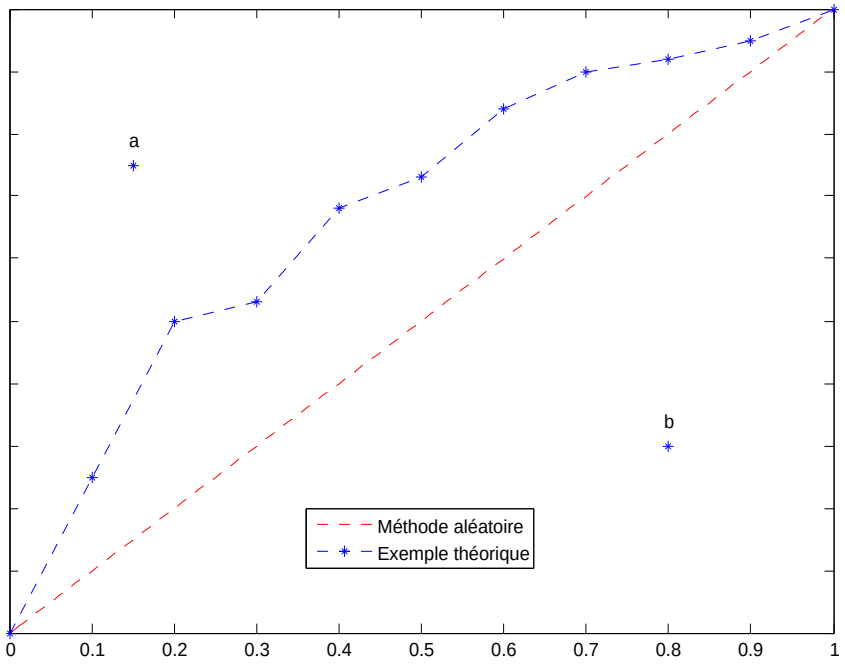

FIGURE 3.1 – Exemple de *ROC*

A partir d'un graphe de *ROC* comme par exemple celui de la figure 3.1, nous calculerons la valeur de la mesure de qualité des méthodes de prédiction que nous présenterons dans la prochaine partie.

3.1.4 La mesure *AUC*

La mesure *AUC* (*area under curve*) est définie, comme le nom l'indique, comme l'aire de la surface délimitée par les deux axes et la courbe de *ROC*, comme il est indiqué sur la figure 3.2 pour notre exemple théorique.

Il est immédiat que la méthode aléatoire admet une valeur d'*AUC* de 0.5. Une méthode produisant des meilleures résultats que la méthode aléatoire aura donc des valeurs d'*AUC* supérieures à 0.5, pendant qu'une méthode qui produit des résultats moins efficaces que la méthode aléatoire admet des valeurs d'*AUC* inférieures à 0.5.

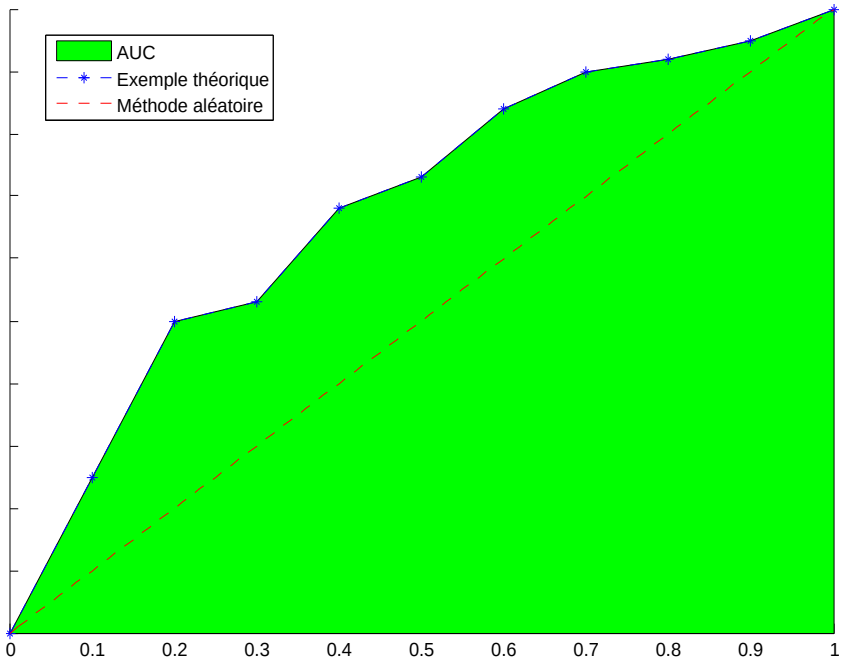

FIGURE 3.2 – Exemple d'*AUC*

Cependant, afin de savoir calculer numériquement la valeur de l'aire sous la courbe, nous avons besoin d'une expression analytique de cette courbe. Dans le cadre de ce travail, nous choisirons d'approximer la courbe de *ROC* par la droite de régression, c'est-à-dire par un polynôme de degré 1. Il est clair que des polynômes de degrés supérieurs augmenteraient la précision de l'approximation et mèneraient donc à des valeurs d'*AUC* plus précises, mais, puisque le but de ce mémoire est la comparaison des différentes méthodes de prédiction de liens manquants, la droite de régression répond suffisamment à nos exigences.

3.1.5 Approche algorithmique au calcul des valeurs d'*AUC*

Nous pouvons résumer le fonctionnement de nos méthodes de prédiction de liens manquants comme suit.

- Nous enlevons des liens d'un réseau de départ.
- Dans une boucle, nous varions le nombre de liens prédits par une des méthodes. Pour chaque nombre de prédictions, nous calculons les *TP*, *FN*, *FP* et les *TN*.
- Nous enregistrons les valeurs de la *TPR* et *FPR* dans un vecteur.

- En dehors de la boucle, nous traçons le vecteur des *TPR* en fonction du vecteur contenant les *FPR*.
- Nous calculons les coefficients du polynôme approximant le mieux les points de l'espace *ROC*.
- Nous calculons l'aire délimitée par les axes et la droite de régression calculée dans le pas précédent.

Finalement, comme déjà dit dans l'introduction, nous souhaitons non-seulement analyser les valeurs pures d'*AUC*, mais aussi les écarts-types de celles-ci. Avant de passer aux applications des méthodes de prédiction de liens manquants, expliquons la démarche que nous avons suivie afin d'obtenir ces valeurs.

Nous nous sommes orientés à la démarche la plus populaire dans le cadre de l'évaluation de la qualité des méthodes de prédiction de liens manquants. Nous sommes partis d'un réseau observé duquel nous avons enlevés 10% de ses arêtes et nous avons ainsi obtenu un nouveau réseau. Ce nouveau réseau admettra donc le même nombre de nœuds que le réseau de départ, mais il contient 10% d'arêtes en moins que celui-ci. C'est la raison pour laquelle nous dirons que *la fraction de liens observés* pour ce nouveau réseau vaut 0.9. Il s'agit d'une expression qui reviendra très souvent dans la suite de ce travail. Le nombre d'arêtes de ces réseaux modifiés sera toujours donné en terme de la fraction de liens observés. Ensuite, nous enlevons de nouveau 10% d'arêtes (les 10% correspondent au réseau observé et non au réseau à fraction de liens observés de 0.9) de ce réseau à fraction de liens observés 0.9 et nous obtenons ainsi un nouveau réseau à fraction de liens observés 0.8. Nous procédons de la même manière jusqu'à arriver au réseau à fraction de liens observés 0.1. Nous appellerons ces 9 réseaux une *série de réseaux consécutifs*, puisque l'un des réseaux est toujours équivalent à son prédécesseur, excepté un nombre fixe d'arêtes que nous lui ont retirées. Il s'agit de la même procédure que Clauset a suivi dans [9] afin de montrer l'efficacité de sa méthode hiérarchique.

Cependant, une question se pose. Quid si nous avons enlevé, par malchance, des arêtes particulièrement importante pour la structure de ce réseau ? Il est bien possible que le retirement de certaines arêtes change plus fortement la topologie du réseau entier que d'autres arêtes. Afin de répondre à cette question, nous n'appliquerons les méthodes de prédiction de liens manquants non seulement à une seule série de réseaux consécutifs, mais à 10 séries différentes. Pour chacune des 10 séries de réseaux, nous répèterons alors 9 fois la procédure présentée ci-dessus et nous obtiendrons une matrice de dimensions 10×9, chaque ligne contenant les valeurs d'*AUC* pour une série de réseaux, chaque colonne contenant les valeurs d'*AUC* de chaque série de réseaux pour une fraction fixée de liens observés. Ensuite, nous calculons les moyennes ainsi que les écarts-types des valeurs d'*AUC* pour chaque

fraction de liens observés (c'est-à-dire, en mots simples, nous calculons la moyenne et l'écart-type de chaque colonne de la matrice obtenue). Nous obtenons dès lors, pour chaque méthode de prédiction, un vecteur à 9 éléments qui sont les valeurs moyennes ainsi que les écarts-types d'AUC que nous traçons sur un graphe en fonction de la fraction des arêtes observées.
Il est clair que, en général, si la fraction de liens observés augmente, c'est-à-dire si nous avons plus d'informations sur ce réseau en termes d'arêtes, les prédictions seront meilleures. C'est la raison pour laquelle nous nous attendons pour les méthodes de prédiction non-aléatoire à une courbe d'AUC croissante. La méthode aléatoire quant à elle devrait montrer un graphe d'AUC horizontal à valeur 0.5.

Clôturons cette section sur l'AUC par une remarque générale. Dans le cadre de ce mémoire, nous avons choisi la mesure AUC pour évaluer la qualité des méthodes de prédiction de liens manquants. Il existe d'autres mesures de la qualité de telles méthodes, comme par exemple le critère $F1$. Contrairement à la mesure AUC qui est basée sur les TPR et FPR, le critère $F1$ combine deux valeurs appelées *recall* et *precision* qui sont définies de la manière suivante :

$$precision = \frac{TP}{TP + FP}, \tag{3.1.3}$$

$$recall = \frac{TP}{TP + FN}. \tag{3.1.4}$$

La valeur de *precision* est la fraction de bonnes prédictions parmi toutes les arêtes prédites par la méthode et la valeur de *recall* est la fraction des bonnes prédictions parmi les arêtes enlevées. Le critère $F1$ combine ces deux métriques de la manière suivante :

$$F1 = 2 \times \frac{recall \times precision}{recall + precision} = \frac{2 \times TP}{2 \times TP + FN + FP}. \tag{3.1.5}$$

La mesure $F1$ est un critère très connu dans la littérature et est par exemple appliquée dans [13] afin de montrer la qualité des différentes méthodes de prédiction de liens manquants. Cependant, dans le cadre de ce mémoire, nous avons choisi pour deux raisons la mesure AUC. Premièrement, il s'agit d'une mesure très intuitive et facile à comprendre. Deuxièmement, Clauset utilise cette méthode dans [9] afin de montrer l'efficacité de la méthode hiérarchique. Puisque cette méthode forme le cœur de ce mémoire, nous avons choisi la même mesure que Clauset. En plus, cela nous permet de comparer facilement nos résultats avec les résultats trouvés dans [9].

La mesure de qualité des méthodes de prédictions étant définie, nous passons par la suite à la partie applicative de ce mémoire. Nous présenterons tout d'abord dans une première section les cinq réseaux auxquels nous avons appliqué les méthodes de prédictions de liens manquants. Les sections suivantes de ce chapitre seront consacrées à l'analyse de nos résultats.

3.2 Présentation des jeux de données

Nous avons appliqué les cinq méthodes de prédictions de liens manquants à cinq différents jeux de données réels que nous présenterons dans cette section. Dans la liste suivante, nous donnons une brève description des jeux de données ainsi que leurs abréviations dans les codes des méthodes de prédiction de liens manquants qui se trouvent dans l'*Annexe B* de ce mémoire. Des informations plus détaillées sur les structures internes de chacun des cinq réseaux seront données dans les parties suivantes.

- *greekterrorists* : Il s'agit du réseau terroriste grec "Revolutionary Organisation November 17", aussi connu sous le nom *politexnio* ou sous l'abréviation *17N* qui a lancé dans les années septantes 103 attaques sur des cibles américaines, britanniques et grecques. Un nœud représente un terroriste et une arête entre deux nœuds signifie que ces deux terroristes ont travaillé ensemble dans le cadre d'une activité terroriste. Ce jeu de données a été analysé par Rhodes dans [28]. Nous remercions Tim Evans qui nous a envoyé les données.
- *karate* : Il s'agit d'un réseau très connu dans la littérature qui contient des informations sur des membres d'un club de karaté d'une université américaine dans les années septantes. Une arête entre deux nœuds signifie que ces deux individus sont amis en dehors des activités associées au club de karaté. Ce jeu de données est disponible gratuitement sur Internet et provient de [33].
- *terrorists0911* : Ce réseau est le réseau terroriste des attaques du 11 septembre sur le Word Trade Center à New York. Une arête entre deux nœuds indique qu'une relation entre ces deux terroristes a été prouvée (en terme d'échange d'informations, de cohabitation, de formation dans le même camp,...). Ce jeu de données a été construit par Krebs dans [16] et a été mis à notre disposition par Aaron Clauset.
- *polbooks* : Ce réseau contient des informations sur des livres politiques vendus par le libraire en ligne *Amazon*. Une arête entre deux nœuds signifie que ces deux livres sont fréquemment achetés par le même client. Il s'agit d'un réseau non-publié disponible sur le site Internet de V. Krebs [1].

1. `http://www.orgnet.com/VKbio.html`

– *football* : Ce dernier réseau modélise un championnat de la première division de football américain entre des équipes universitaires. Une arête entre deux nœuds signifie que ces équipes se sont rencontrées dans un match au cours de la saison. Ce qui est intéressant à ce réseau est qu'il admet, par construction, différentes communautés, parce que le championnat de football américain est formé de différentes divisions, chaque division contenant 8 à 12 équipes. Des matchs entre des équipes appartenant à la même division sont plus fréquents que des matchs entre des équipes appartenant à différentes divisions (en moyenne : 7 matchs par équipe contre des équipes de la même division, 4 matchs par équipe contre des équipes d'une autre division). Cependant, lorsque deux équipes très proches géographiquement appartiennent à deux communautés différentes, des matchs entre ces deux équipes seront plus fréquents que des matchs entre des équipes appartenant à différentes communautés et situées très loin géographiquement l'une de l'autre. Ce jeu de données a été récoltée par Girvan dans [12].

Dans la suite, nous passons à une interprétation plus détaillée de ces réseaux. Nous analyserons les propriétés suviantes des réseaux : le nombre de nœuds, le nombre d'arêtes, le degré moyen, le nombre de triangles présent dans le réseau, le coefficient de clustering et la valeur de modularité. Le *coefficient de clustering* est une mesure de la transitivité d'un réseau. Définissons cette notion de manière plus détaillée, puisque nous n'en avons pas encore parlé dans le cadre de ce mémoire.
En général, la transitivité d'une relation " $\circ$ " est exprimée par le concept suivant : si $a \circ b$ et $b \circ c$, alors $a \circ c$. Dans la théorie des réseaux, la transitivité se traduit par le fait que s'il y a une arête entre a et b et s'il y en a une arête entre b et c, alors il y en a aussi une entre a et c. Dans la théorie des réseau sociaux, on pourrait interpréter cette propriété par "l'ami de mon ami est mon ami". Il est immédiat que cette notion de transitivité dans les réseaux se traduit par la fermeture des triangles.
Cependant, un réseau complètement transitive n'est pas intéressant, puisqu'il s'agirait d'un réseau pour lequel il existerait un sous-réseau dans lequel chaque nœud serait connecté à chaque autre nœud. C'est pourquoi nous nous intéressons plutôt à la *transitivité partielle*. Dans un réseau à transitivité partielle, le fait d'une arête entre a et b et d'une arête entre b et c ne garantit pas l'existence d'une arête entre a et c, mais cette arête sera plus probable qu'une arête aléatoire. En terme de réseau social, l'ami de mon ami n'est pas nécessairement un ami à moi, mais il est beaucoup plus probable que j'aurai une liaison amicale avec cette personne qu'avec une personne aléatoire. Nous voyons donc que cette notion a des rapports avec la méthode des triangles présentée dans le chapitre 2.
Nous mesurons le niveau de transitivité partielle par le *coefficient de clustering* que nous définissons comme suite. S'il y a une arête entre a et b et une arête entre

b et c, nous disons qu'il y a un chemin de longueur 2 appelé abc. Si, en plus, il y a une arête entre a et c, nous dirons que le chemin abc est fermé. Le *coefficient de clustering* est la fraction entre le nombre des chemins **fermés** de longueur 2 et le nombre total de chemins de longueur 2, c'est-à-dire

$$C = \frac{\text{nombre de chemins fermés de longueur 2}}{\text{nombre de chemins de longueur 2}}.$$

Si $C = 1$, alors le réseau est parfaitement transitive et chaque chemin de longueur 2 est fermé, et si $C = 0$, aucun chemin de longueur 2 est fermé.
La dernière propriété des réseaux que nous afficherons pour chaque jeu de données est la modularité que nous avons défini dans la partie sur la méthode des communautés. Une grande valeur de modularité indique dès lors une partition très marquée en communautés.

Le réseau des terroristes grecs

Le réseau des terroristes grecs est affiché sur la figure 3.3. Pour cette visualisation, nous avons utilisé le logiciel libre *Geophi*. Chaque nœud dans le réseau remplace un terroriste, et à chaque terroriste correspond un label. Le lecteur intéressé trouve la liste des noms des terroristes et les labels correspondants dans l'*Annexe C*.

Le tableau suivant résume quelques propriétés de ce réseau.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
greekterrorists	22	63	5.727	73	0.532	0.289

Il s'agit dès lors d'un petit réseau à un petit nombre de nœuds à degré moyen relativement faible. Le coefficient de clustering est relativement proche de 0.5, ce qui nous indique un niveau de transitivité moyen. La valeur faible de modularité nous indique une partition pas très marquée en communautés. Cependant, notons que des petits réseaux ont tendance à avoir une faible modularité : plus le réseau est grand, plus de grandes valeurs de modularité peuvent être observées. Finalement, la distribution des degrés est affichée sur la figure 3.4 qui nous donne une impression sur l'hétérogénéité de la distribution des degrés. Il semble que les degrés ne soient pas distribués uniformément parmi l'ensemble des nœuds.

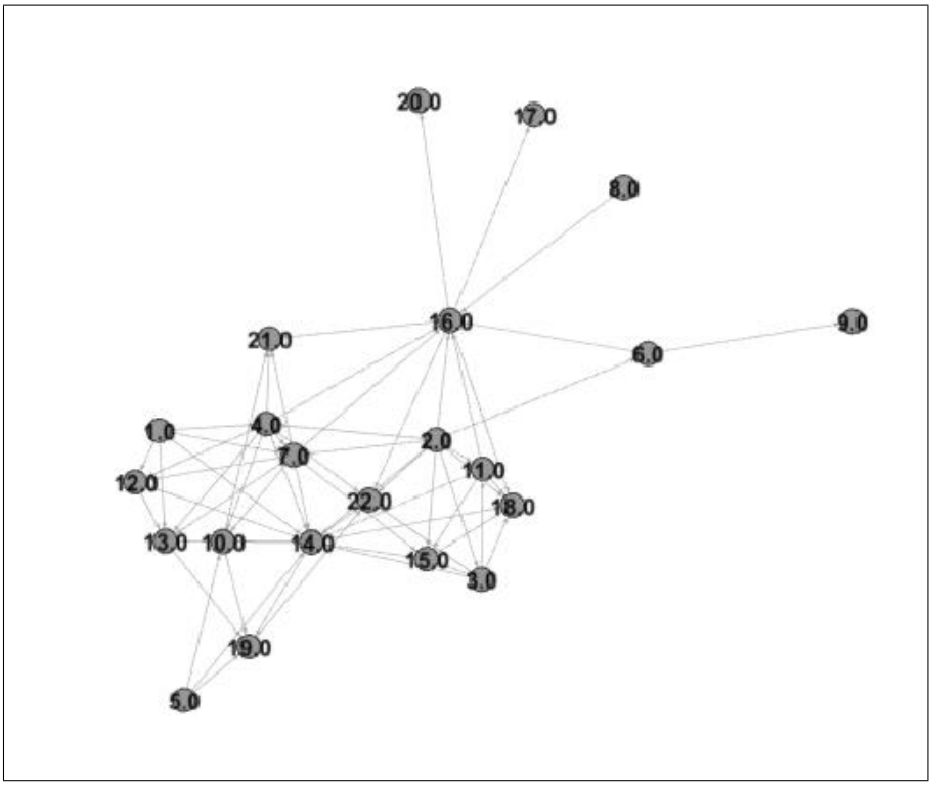

FIGURE 3.3 – Représentation du réseau des terroristes grecs

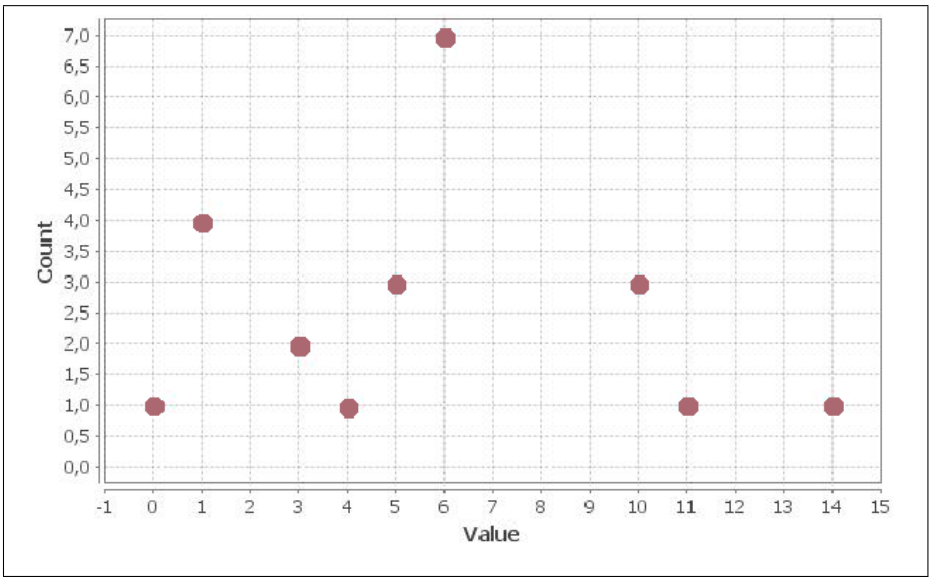

FIGURE 3.4 – Distribution des degrés du réseau des terroristes grecs

Le réseau du club de karaté de Zachary

Le réseau karaté est représenté sur la figure 3.5. Le tableau suivant affiche quelques propriétés de ce réseau.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
karate	34	78	4.588	45	0.571	0.415

Ce réseau est donc composé de 34 nœuds et 78 arêtes, le nombre de triangles est plus petit que celui du réseau des terroristes grecs, mais le niveau de transitivité est presque équivalent.
La figure 3.6 montre la distribution des degrés pour ce réseau. Nous observons qu'il y a 11 des 22 nœuds qui admettent comme degré la valeur 2, les 11 nœuds restants admettent comme degré des valeurs entre 1 et 17. Il s'agit dès lors d'une distribution non-hétérogène.

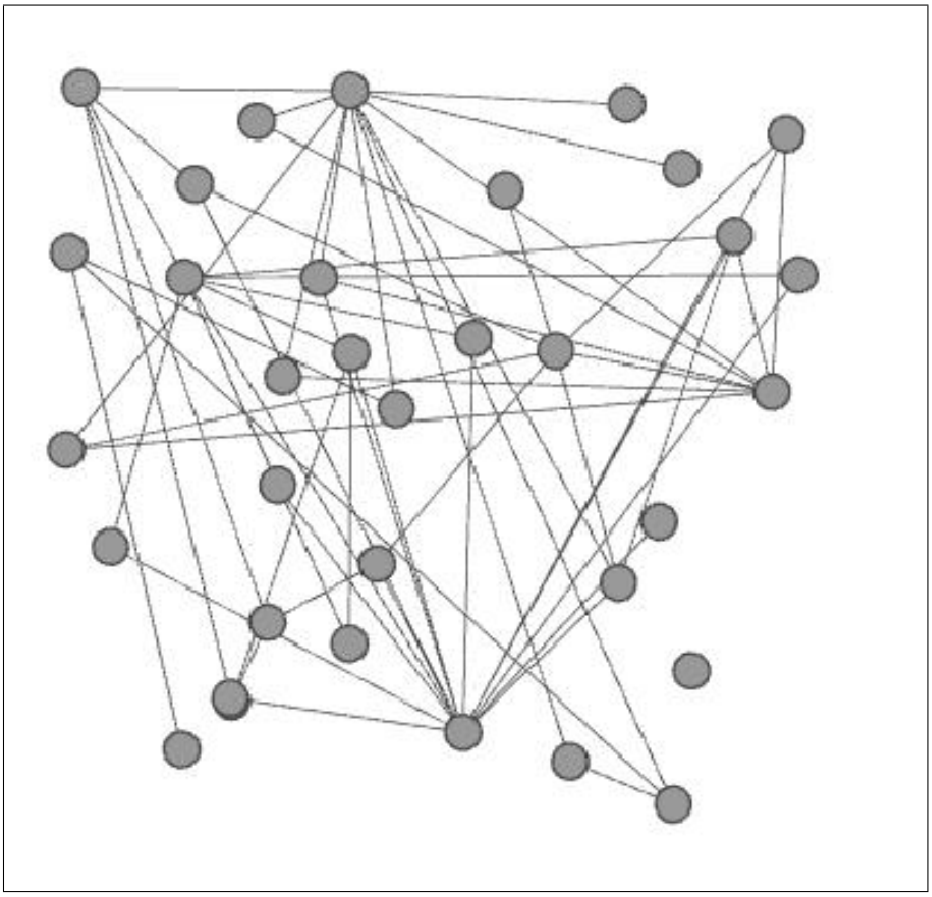

FIGURE 3.5 – Représentation du réseau karaté

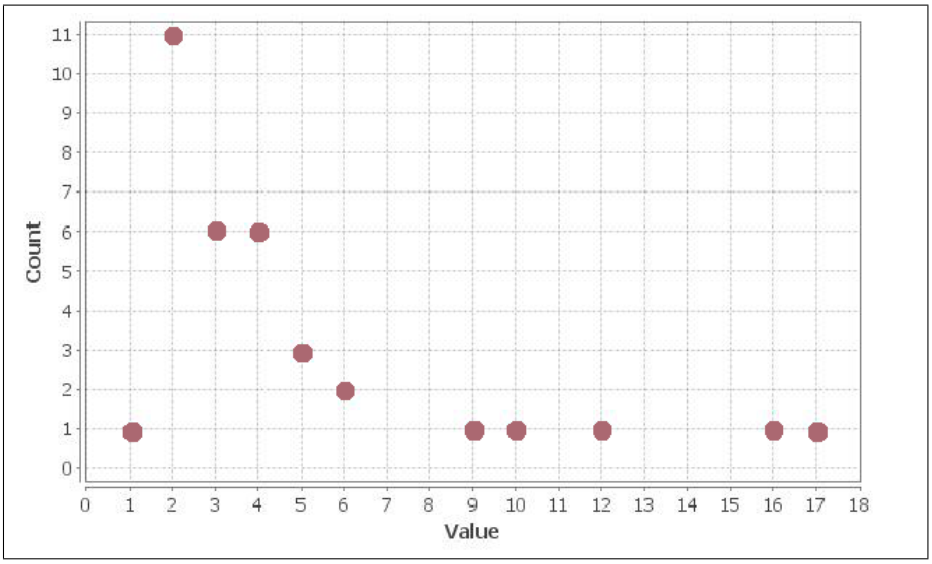

FIGURE 3.6 – Distribution des degrés du réseau karaté

Le réseau des terroristes des attaques du 11/09

La représentation de ce réseau n'est plus intéressant puisque les nombres de nœuds et des arêtes deviennent trop grands. Le tableau suivant résume quelques propriétés de ce réseau.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
terrorists0911	62	152	4.903	131	0.486	0.532

Ce graphe admet donc beaucoup plus d'arêtes que les deux précédents. Le coefficient de clustering est plus petit que pour les autres méthodes, ce qui indique un réseau de niveau de transitivité plus faible. La valeur de la modularité par contre est plus grande que pour les deux réseaux précédents, ce qui nous indique une partition plus marquée en communautés, mais il faut, de nouveau, tenir compte de la plus grande taille de ce réseau.
La figure 3.7 nous montre la distribution des degrés. Nous remarquons que la distribution des degrés est beaucoup moins hétérogène que pour les autres réseaux.

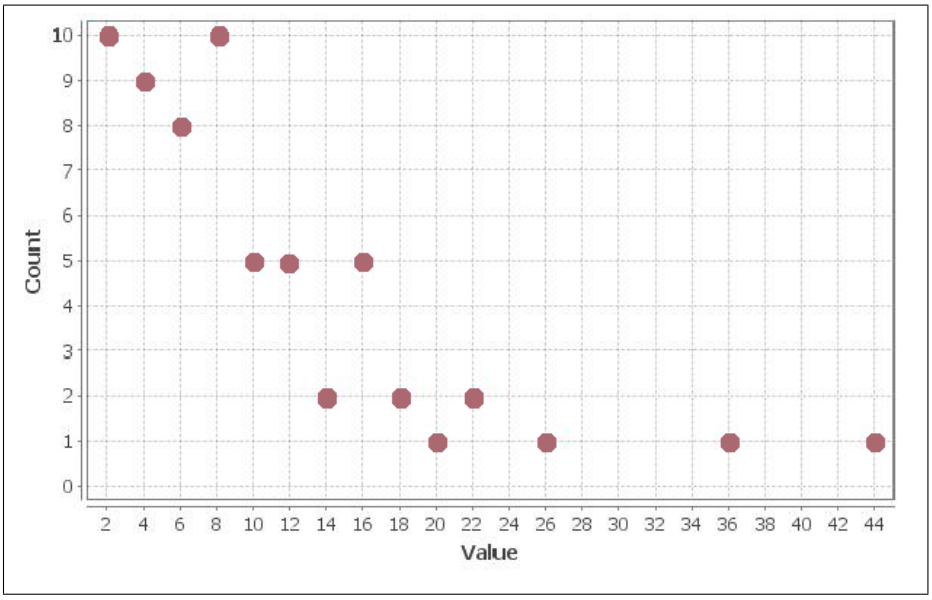

FIGURE 3.7 – Distribution des degrés du réseau des terroristes des attaques du 11/09

Le réseau des livres politiques sur *Amazon*

Les propriétés de ce réseau sont indiquées dans le tableau suivant.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
polbooks	105	441	8.400	560	0.488	0.519

Il s'agit dès lors d'un réseau de 105 nœuds et 441 arêtes. Le nombre de triangles est très élevé, mais le coefficient de clustering est relativement petit, ce qui nous indique l'existence d'un grand nombre de chemins de longueur 2 non-fermés.
En tenant compte du grand nombre de nœuds présents dans ce réseau, la figure 3.8 nous indique que, comme pour le réseau des terroristes des attaques du 11/09, la distribution des degrés n'est pas hétérogène pour ce réseau.

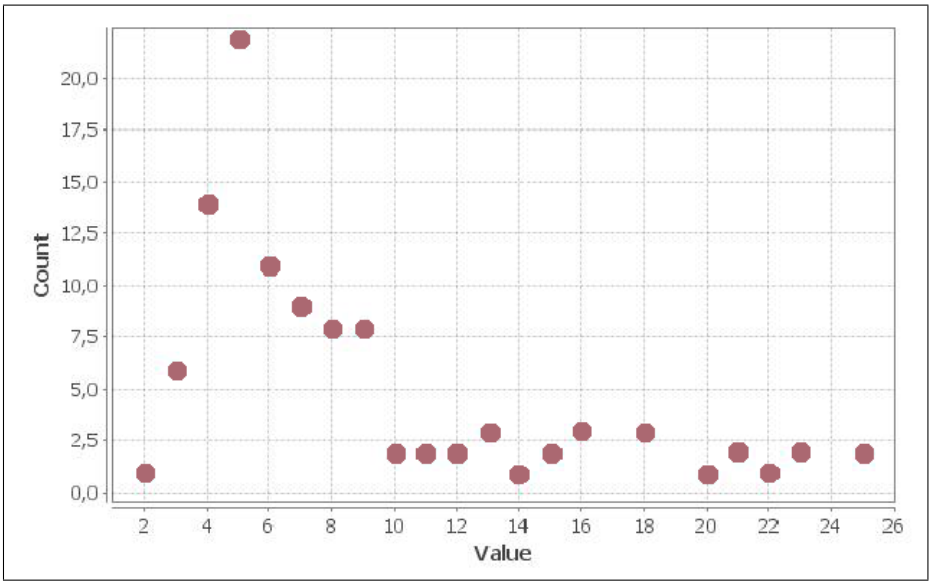

FIGURE 3.8 – Distribution des degrés du réseau des livres politiques

Le réseau du championnat de football américain

Les propriétés de ce dernier réseau sont résumées dans le tableau suivant.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
football	115	613	10.661	810	0.403	0.605

Il s'agit dès lors du réseau admettant le plus grand nombre de nœuds et d'arêtes. De nouveau, il y a beaucoup de triangles dans le graphe, mais le coefficient de clustering est relativement faible, ce qui est dû aux matchs entre les équipes appartenant à différentes divisions. Ces matchs forment beaucoup de chemins de longueur 2 qui ne sont pas fermés. Cependant, la valeur de la modularité est relativement élevée pour ce réseau, ce qui nous paraît logique à cause de la partition naturelle en communautés (c'est-à-dire en divisions) de ce réseau. La figure 3.9 nous indique que la distribution des degrés de ce réseau est beaucoup plus hétérogène que pour les autres réseaux, ce qui est, de nouveau, dû au caractère de championnat de ce réseau.

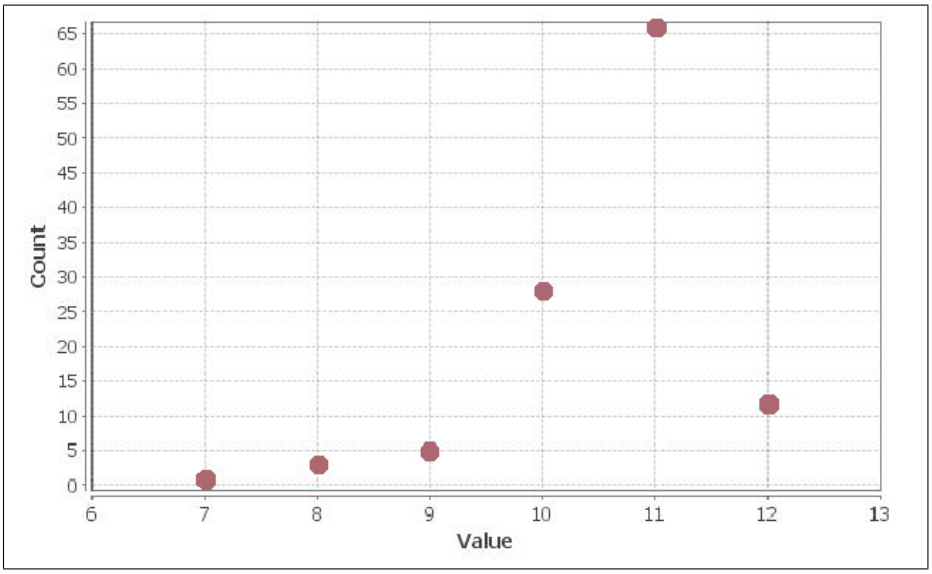

FIGURE 3.9 – Distribution des degrés du réseau du championnat de football américain

Tableau récapitulatif

Comparons maintenant les cinq réseaux présentés ci-dessus entre eux. Quelques propriétés de ceux-ci sont résumées dans le tableau suivant.

	nombre de nœuds	nombre d'arêtes	degré moyen	nombre de triangles	coefficient de clustering	modularité
greekterrorists	22	63	5.727	73	0.532	0.289
karaté	34	78	4.588	45	0.571	0.415
terrorists0911	62	152	4.903	131	0.486	0.532
polbooks	105	441	8.400	560	0.488	0.519
football	115	613	10.661	810	0.403	0.605

Il s'agit dès lors de cinq réseaux qui diffèrent dans leurs nombres de nœuds et d'arêtes. De plus, nous remarquons que les degrés moyens sont plus élevés pour les deux derniers réseaux, notamment pour le réseau du football américain. En ce qui concerne les coefficients de clustering, seulement les deux premiers réseaux admettent des valeurs élevées. Comme nous avons déjà remarqué plus haut, il semble que les deux derniers réseaux admettent un grand nombre de chemins de longueur 2 non-fermés. Finalement, les valeurs de la modularité nous indiquent que surtout le dernier réseau du football américain a une forte tendance naturelle à former des communautés, pendant que le réseau des livres politiques et celui des terroristes des attaques aux Twin Towers admettent des valeurs moyennes pour cette mesure. Finalement, il semble qu'une partition en communautés est moins marquée pour le réseau karaté et surtout pour le réseau des terroristes grecs.
Par la suite, nous présenterons les résultats de nos expériences sur les méthodes de prédictions de liens manquants.

3.3 Application au réseau des terroristes grecs

Dans les sections suivantes, nous procédons de la manière suivante pour représenter les résultats de nos expériences. Nous présenterons tout d'abord les trois graphes d'AUC pour la méthode des communautés pour trois valeurs différentes du paramètre t (0.2, 1.0 et 5.0). Parmi ces trois graphes, nous en choisirons le meilleure et nous le présenterons ensuite sur un même graphe avec les graphes d'AUC des quatre méthodes restantes que nous interpréterons. Avant de présenter les résultats de nos expériences pour le réseau des terroristes grecs, remarquons que, comme déjà dit précédemment, nous avons appliqué les méthodes de prédiction de liens manquants à 10 séries de 9 réseaux consécutifs, chaque série partant du réseau de départ des terroristes grecs. Ensuite, nous avons calculé la moyenne pour chaque fraction de liens observés ainsi que l'écart-type. Nous avons tracé sur le graphe d'AUC les moyennes ainsi que les barres d'erreur dont la longueur vaut deux fois l'écart-type. Puisque ces barres d'erreurs se recouvrent souvent, ce qui rend les graphes parfois embrouillés, nous n'affichons pas seulement les courbes d'AUC sur le même graphe, mais aussi la courbe d'AUC pour chaque méthode de prédiction dans un repère séparé.

3.3.1 Comparaison des différents choix de t pour la méthode des communautés

La figure 3.10 affiche les résultats de la méthode des communautés pour différents choix du paramètre t.

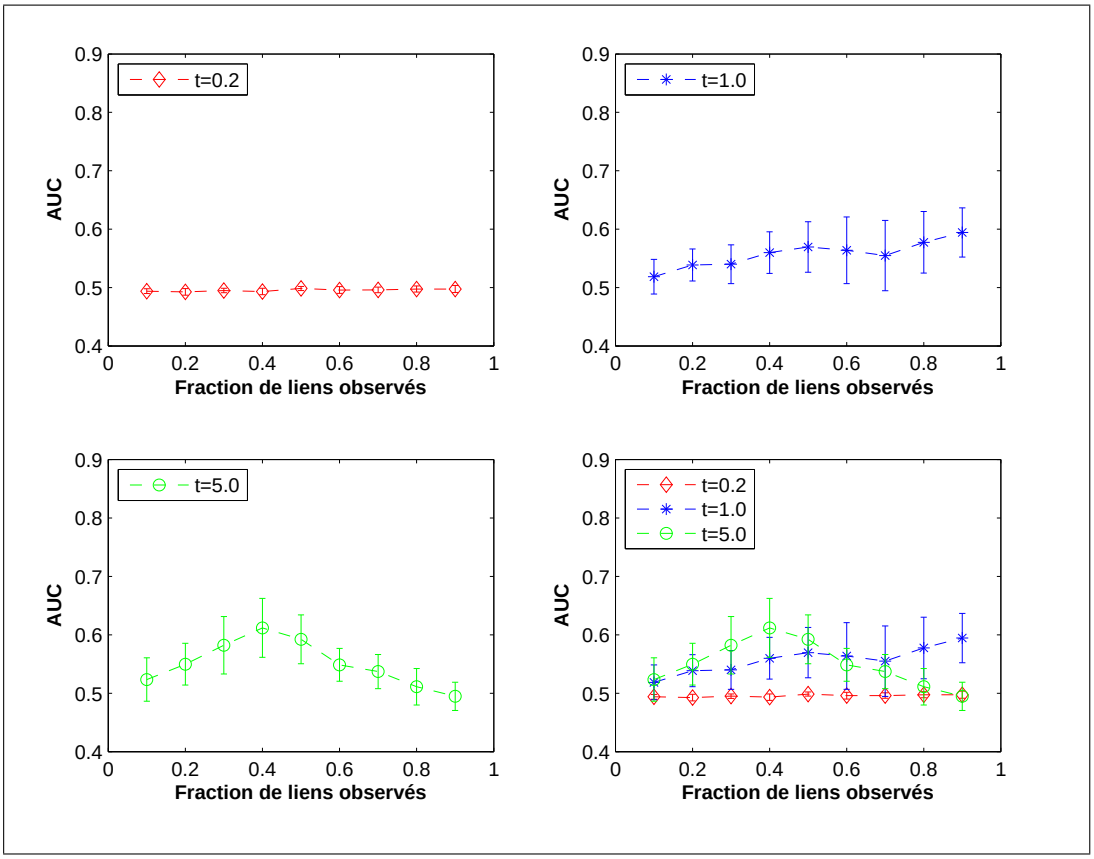

FIGURE 3.10 – Résultats de l'*AUC* pour la méthode des communautés pour le réseau des terroristes grecs

Tout d'abord, nous remarquons que la méthode des communautés avec un choix de $t = 0.2$ revient presque à la méthode de prédiction aléatoire. Comme nous avons expliqué dans la section sur la méthode des communautés, un choix de $t = 0.2$ provoque la formation de beaucoup de classes de petite taille en termes de nombre de nœuds. En effet, analysons à titre d'exemple un réseau pour lequel la fraction de liens observés vaut 0.9. Pour ce réseau, l'algorithme *Louvain* trouve la meilleure partition en une partition en 16 classes. Autrement dit, il propose une partition des 22 nœuds en 16 communautés. La liste exhaustive des 22 nœuds et leur appartenance aux communautés se trouve dans l'*Annexe A*. Parmi ces 16 communautés, il y en a 13 qui sont formées d'un seul nœud et 3 communautés contiennent 2, 3 et 4 nœuds respectivement. Il y a donc en total $\binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 10$ paires de nœuds qui se trouvent dans une même communauté. Nous avons vérifié que, pour ce jeu de données, trois de ces paires de nœuds se trouvant dans la même communauté sont non-connectés et sont donc l'objet d'une prédiction. La méthode

de prédiction de liens manquants basée sur les communautés prédira donc trois arêtes basées sur l'appartenance des nœuds à une même communauté, les autres arêtes seront prédites de manière aléatoire. Dans ce cas-ci, par malheur, les trois arêtes prédites par la méthode des communautés ne sont pas des arêtes enlevées, il s'agit donc de mauvaises prédictions. Nous avons expliqué le comportement de la méthode des communautés avec un choix de $t = 0.2$ par un exemple particulier d'un réseau admettant une fraction de liens observés relativement élevée. Cependant, les remarques faites ci-dessus sont valables pour toutes les autres séries de réseaux, ce qui explique le même comportement de cette méthode que la méthode aléatoire. Nous remarquons que la méthode trouve les meilleures résultats en termes d'AUC pour un choix de $t = 1.0$. La courbe d'AUC croît légèrement et indique une meilleure efficacité que la méthode aléatoire.
Le comportement de la courbe d'AUC de la méthode des communautés pour un choix de $t = 5.0$ est étrange à premier vue. Elle croît constamment jusqu'à arriver à une fraction de liens observé de 0.4 et elle décroît après. De nouveau, ce comportement se justifie en lançant un regard plus proche aux résultats de la méthode *Louvain*. Comparons à titre d'exemple les résultats de la méthode *Louvain* avec $t = 5.0$ pour une série de réseaux partant du réseau des terroristes grecs (ces fichiers se trouvent dans l'*Annexe A*). Nous remarquons que le nombre de classes varie entre les réseaux à fractions de liens observés entre 0.1 et 0.4 de 13, 8, 7 à 4 classes. Mais pour les réseaux pour lesquels la fraction de liens observés est supérieure à 0.4, la partition en communautés ne change presque plus (à l'exception d'une classe en moins à partir des réseaux à fraction de liens observés de 0.6), c'est-à-dire nous considérons la même partition pendant que le nombre de liens observés augmente. Autrement dit, la partition reste la même, mais le nombre de liens à prédire décroit (puisque la fraction de liens observés croît). Dès lors, puisque la partition ne change pas, mais le nombre de liens à prédire diminue, la qualité des prédictions diminue en terme d'AUC avec la fraction de liens observés à partir de 0.4. De nouveau, nous avons expliqué le comportement du graphe d'AUC à partir d'un exemple particulier, mais le raisonnement est le même pour les autres séries de réseaux.
Dès lors, la méthode des communautés avec $t = 1.0$ est la meilleure, puisque les résultats en termes d'AUC augmentent constamment pour cette méthode et les arêtes prédites ne sont jamais choisies aléatoirement.

3.3.2 Comparaison des différentes méthodes de prédiction de liens manquants.

Passons aux résultats des autres méthodes de prédiction pour le réseau des terroristes grecs. La figure 3.11 montre les graphes d'AUC des méthodes de pré-

diction de liens manquants un par un et la figure 3.12 les montre dans le même système d'axes.

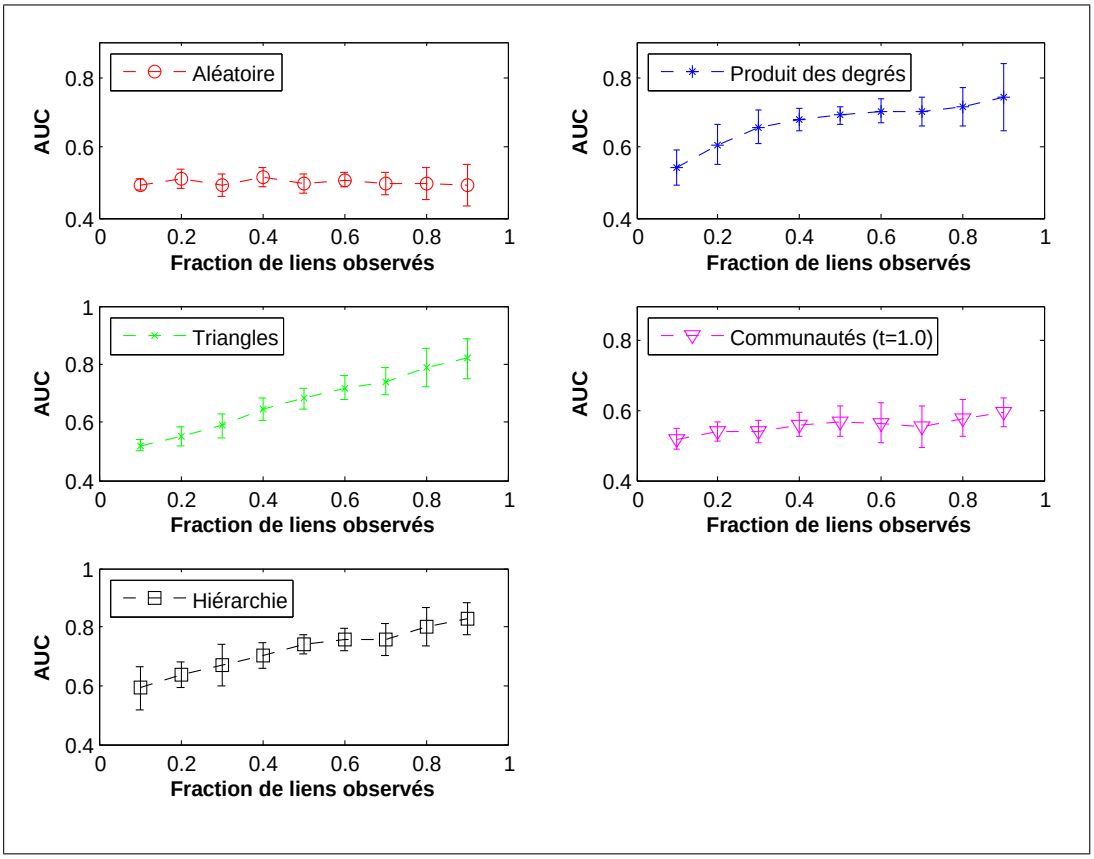

FIGURE 3.11 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau des terroristes grecs.

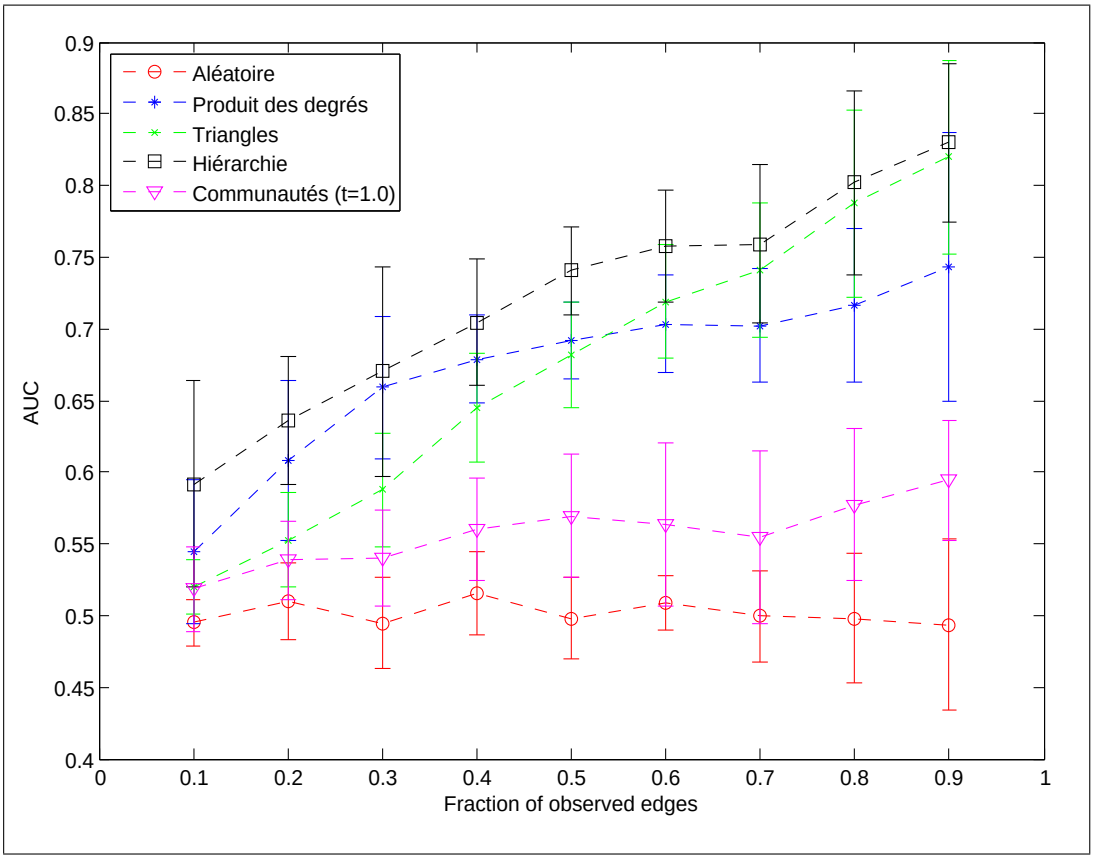

FIGURE 3.12 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau des terroristes grecs.

Nous tirons directement de la figure 3.12 que toutes les méthodes sont plus efficaces que la méthode aléatoire qui, comme nous nous attendons, admet une quasi-droite horizontale à valeur 0.5 comme courbe d'AUC. Cependant, nous remarquons aussi que la méthode des communautés est la moins efficace des 4 autres méthodes. En effet, toutes les 4 méthodes de prédiction non-aléatoire se divisent en deux classes : une classe est formée des méthodes plus efficaces pour ce jeu de données, à savoir la méthodes du produit des degrés, la méthode des triangles et la méthode hiérarchique, tandis que la deuxième classe est formée par la méthode des communautés. Si nous regardons d'une manière plus détaillée les méthodes plus efficaces, nous remarquons que pour les réseaux à fractions de liens observés de 0.1, 0.2 et 0.3, la méthode hiérarchique et la méthode du produit des degrés sont légèrement plus efficaces que la méthode des triangles. A partir de la fraction de liens observer de 0.5, la méthode des triangles devient en moyenne plus efficace que la méthode du produit des degrés, mais elle reste encore légèrement moins

efficace que la méthode hiérarchique. Pour les réseaux à fractions de liens observés de 0.8 et 0.9, la méthode hiérarchique et la méthode des triangles donnent presque les mêmes résultats et deviennent significativement plus efficaces que la méthode du produit des degrés.
En ce qui concerne les barres d'erreurs, à partir de la fraction de liens observés 0.7, les longueurs de ceux-ci de la méthode hiérarchique sont plus petites que les longueurs des barres d'erreur de la méthode des triangles et la méthode du produit des degrés. En effet, les valeurs d'AUC pour les réseaux d'une fraction de liens observé de 0.9 se trouvent dans l'intervalle $[0.5672; 0.8075]$ pour la méthode du produit des degrés et, pour la méthode des triangles, dans l'intervalle $[0.7043; 0.9093]$, pendant que les valeurs d'AUC de la méthode hiérarchique se trouvent pour les réseaux d'une fraction de liens observés de 0.9 dans l'intervalle $[0.7487; 0.8917]$. Cela montre que la méthode hiérarchique est beaucoup plus stable que surtout la méthode du produit des degrés. Autrement dit, les résultats de la méthode du produit des degrés dépendent beaucoup plus de la manière dont les liens ont été retirés du réseau puisque les valeurs d'AUC varient de 0.56 à 0.80 en fonction de la série de réseaux considérée. Les résultats de la méthode hiérarchique, quant à elle, ne varient que très peu, ce qui nous indique une indépendance de la structure des réseaux.

3.4 Application au réseau de karaté

3.4.1 Comparaison des différents choix de t pour la méthode des communautés

La figure 3.13 montre les résultats de la méthode des communautés pour différents choix de t pour le réseau de karaté.

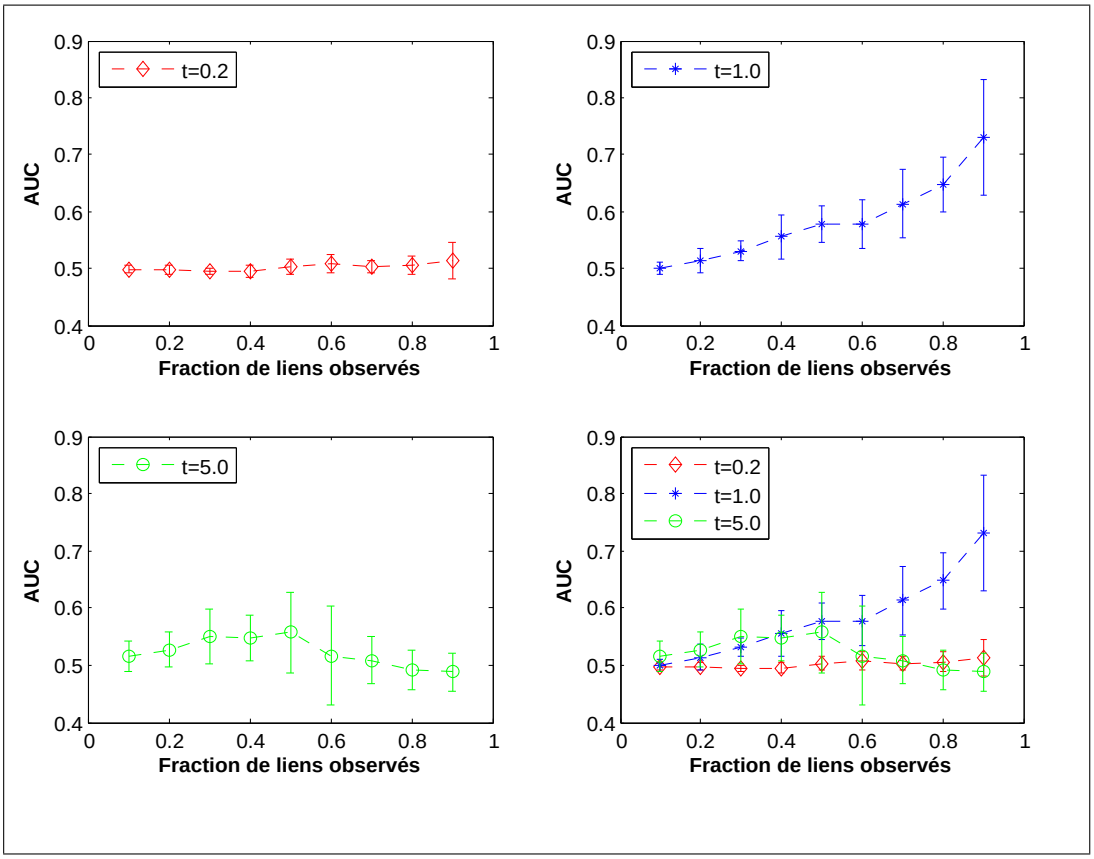

FIGURE 3.13 – Résultats de l'*AUC* pour la méthode des communautés pour le réseau karaté

Nous remarquons de nouveau que le choix de $t = 0.2$ revient presqu'à la méthode aléatoire. L'explication de ce raisonnement est la même que celle donnée dans le cadre de l'application au réseau des terroristes grecs : Puisque l'algorithme *Louvain* produit beaucoup de groupes d'un petit nombre d'éléments, beaucoup des groupes sont composés d'un seul nœud et les autres groupes sont formés par des nœuds qui sont très connectés entre-eux. C'est pourquoi la méthode de communautés doit choisir beaucoup d'arêtes prédites aléatoirement pour un choix de $t = 0.2$.
Le comportement de la courbe d'*AUC* pour un choix de $t = 5.0$ est également similaire à celui de la courbe d'*AUC* pour la méthode des communautés avec même valeur du paramètre appliquée au réseau des terroristes grecs. Le nombre de communautés décroît avec la fraction de liens observés, et, à partir d'une certaine fraction de liens observés, les partitions ne diffèrent plus significativement mais le nombre de liens à prédire diminue avec la fraction de liens observés. Dès lors, la

méthode choisit moins d'arêtes d'un ensemble qui ne change plus. C'est la raison pour laquelle la qualité des prédictions décroît à partir d'une certaine fraction de liens observés.
Pour ce jeu de données, la méthode des communautés donne des bon résultats pour un choix de $t = 1.0$. Nous choisirons donc cette valeur pour le paramètre t et nous comparons ce résultat aux graphes d'AUC des autres méthodes. Remarquons cependant que l'écart-type pour les réseaux à une fraction de liens observés de 0.9 est très élevé, à savoir 0.1012.

3.4.2 Comparaison des différentes méthodes de prédiction de liens manquants

Passons aux résultats des autres méthodes de prédiction de liens manquants, affichés dans les figures 3.14 et 3.15.

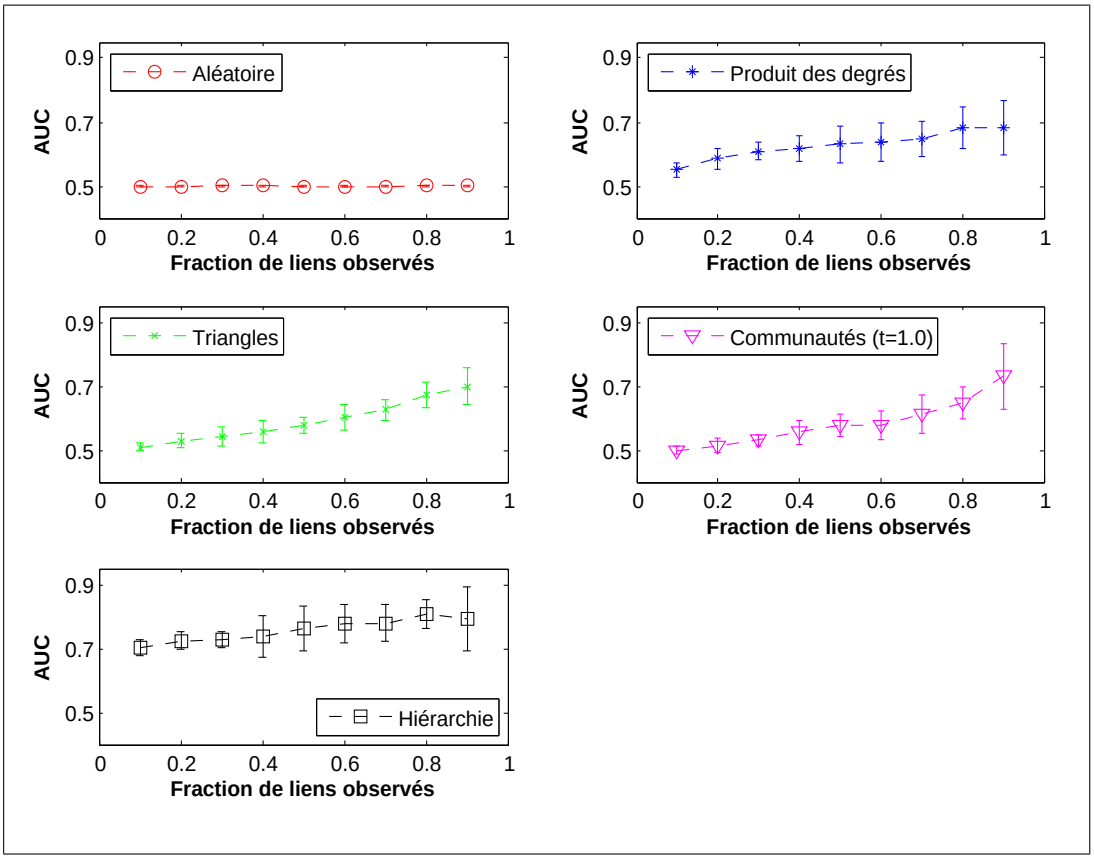

FIGURE 3.14 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau de karaté.

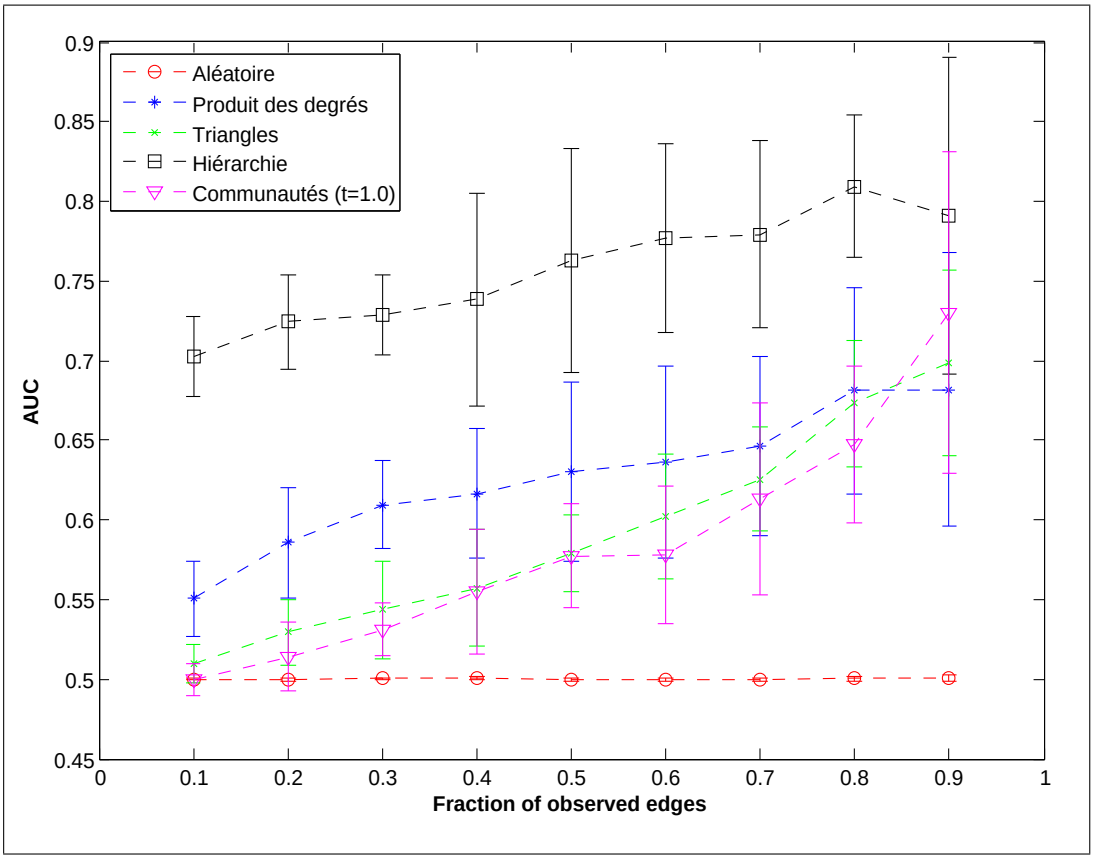

FIGURE 3.15 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau de karaté.

De nouveau, le graphe d'AUC de la méthode aléatoire est une droite horizontale de valeur 0.5. Toutes les autres méthodes donnent des résultats plus efficaces que la méthode aléatoire. De nouveau, nous pouvons grouper les méthodes en deux classes : la première classe est formée de la méthode du produit des degrés, de la méthode des triangles et de la méthode des communautés, la deuxième classe contient la méthode hiérarchique.
Nous remarquons que pour la plupart des mesures, la méthode du produit des degrés est un peu plus efficaces que la méthode des communautés et la méthode des triangles, qui donnent des résultats presque égaux en termes d'AUC. A partir de la fraction de liens observés de 0.6, l'efficacité en termes d'AUC est presque la même pour ces trois méthodes. Pour les réseaux à fraction de liens observés de 0.9, la méthode des communautés est la meilleure de ces trois, mais il faut tenir compte de la grande étendue de l'écart-type.
Pour ce jeu de données, la méthode hiérarchique est significativement meilleure

que les autres méthodes (sauf peut-être pour les réseaux d'une fraction de liens observés de 0.9 pour lesquels la méthode des communautés donne aussi des bons résultats mais il faut, une fois de plus, tenir compte de la grande étendue de l'écart-type associé).
Cependant, la méthode hiérarchique admet pour certaines fractions de liens observés des écarts-types relativement élevés, comme par exemple un écart-type de 0.06 pour la fraction de liens 0.4, 0.07 pour la fraction de liens 0.5 et 0.09 pour la fraction 0.9. Dès lors, l'efficacité de la méthode hiérarchique dépend à un certain niveau de la structure du réseau. Néanmoins, malgré ces barres d'erreur élargies, la méthode hiérarchique admet des meilleures valeurs d'AUC que les autres méthodes. La méthode des triangles quant, à elle, semble être la méthode la plus indépendante de la structure des réseaux, puisqu'elle admet les plus petites valeurs d'écart-type (le plus grand écart-type pour cette méthode appliquée au réseau de karaté étant 0.05).

3.5 Application au réseau des terroristes des attaques du 11/09

3.5.1 Comparaison des différents choix de t pour la méthode des communautés

La figure 3.16 affiche le résultats de la méthode des communautés pour différents choix du paramètre t.

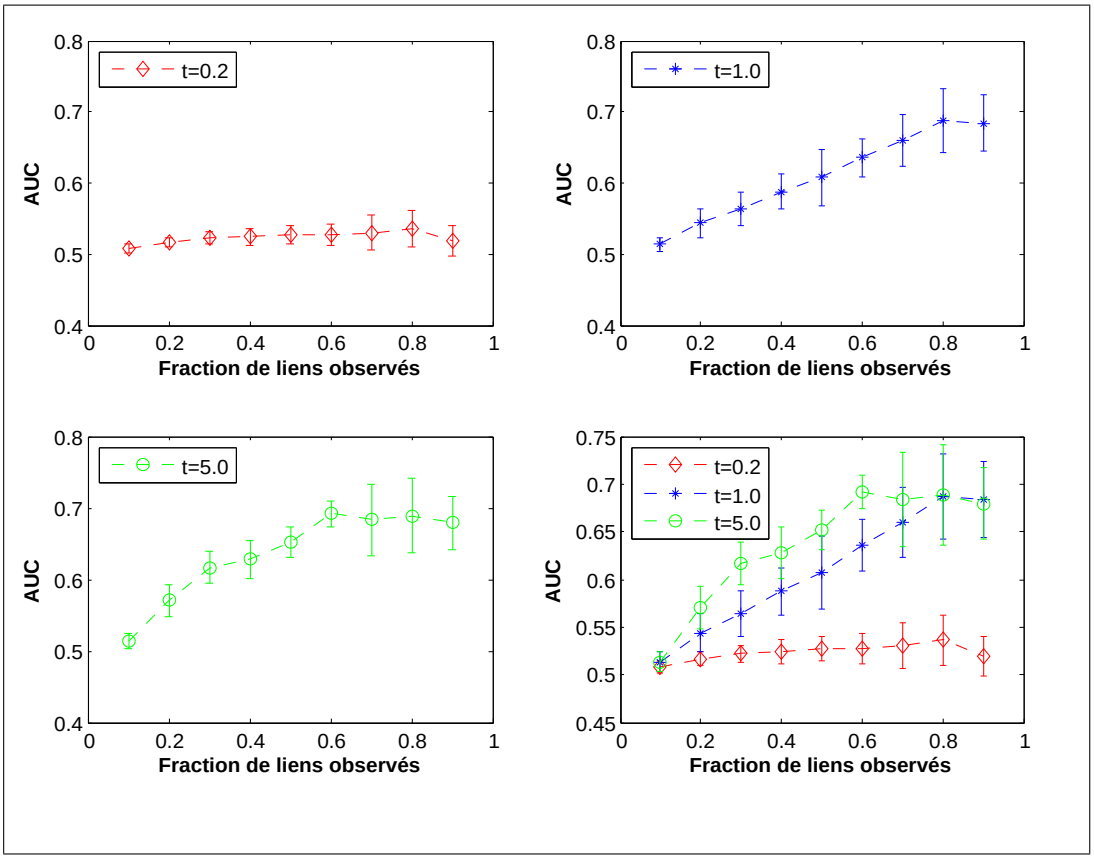

FIGURE 3.16 – Résultats de l'*AUC* pour la méthode des communautés pour le réseau des terroristes des attaques du 11/09.

Le comportement de la courbe d'*AUC* pour le choix de $t = 0.2$ est le même que pour les réseaux déjà traités ci-dessus. L'explication de ce comportement est la même et, c'est la raison pour laquelle nous ne la répèterons pas dans cette partie. Cependant, nous remarquons que les résultats de la méthode pour les choix du paramètre $t = 1.0$ et $t = 5.0$ sont presque identiques et montrent que ces choix de paramètre rendent la méthode des communautés efficaces. Puisque ce réseau admet un plus grand nombre de nœuds (62 nœuds), les communautés trouvées par la méthode *Louvain* avec $t = 5.0$ sont des communautés comptant plus d'éléments que pour les réseaux précédents. C'est la raison pour laquelle, dans chaque communauté, il y a plus de paires de nœuds non-connectés et donc plus de prédictions non-aléatoires (voir le tableau correspondant dans l'*Annexe A*). Pour tous les réseaux, la méthode *Louvain* avec $t = 5.0$ forme moins de communautés que la méthode avec $t = 1.0$. Il y a donc plus de nœuds dans les communautés formées avec $t = 5.0$ et la méthode choisit des arêtes entre un plus grand nombre

d'arêtes non-aléatoires. Il semble que, pour ce réseau, le choix de $t = 5.0$ mène à un bon compromis entre la taille et le nombre de communautés, c'est pourquoi nous comparerons la méthode des communautés avec $t = 5.0$ aux autres méthodes de prédiction de liens manquants.

3.5.2 Comparaison des différentes méthodes de prédiction de liens manquants

Les figures 3.17 et 3.18 montrent les courbes d'AUC pour les différentes méthodes de prédiction de liens manquants.

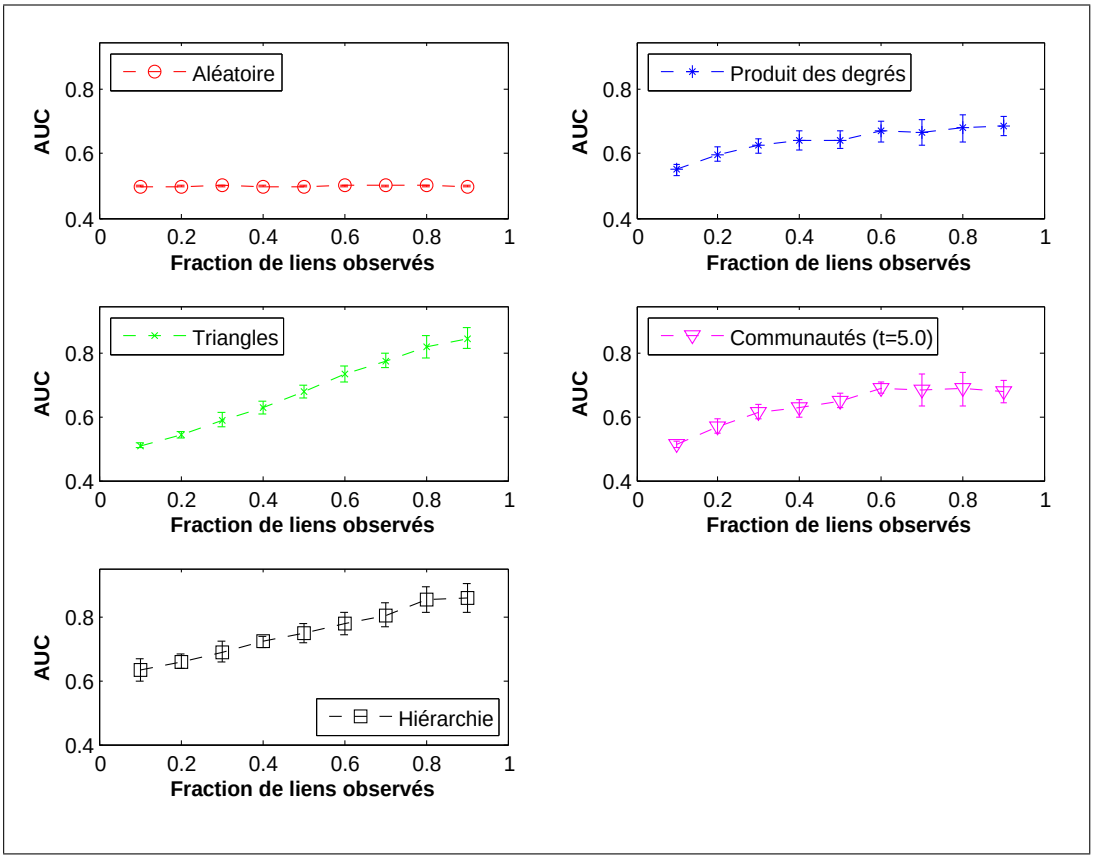

FIGURE 3.17 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau des terroristes des attaques du 11/09.

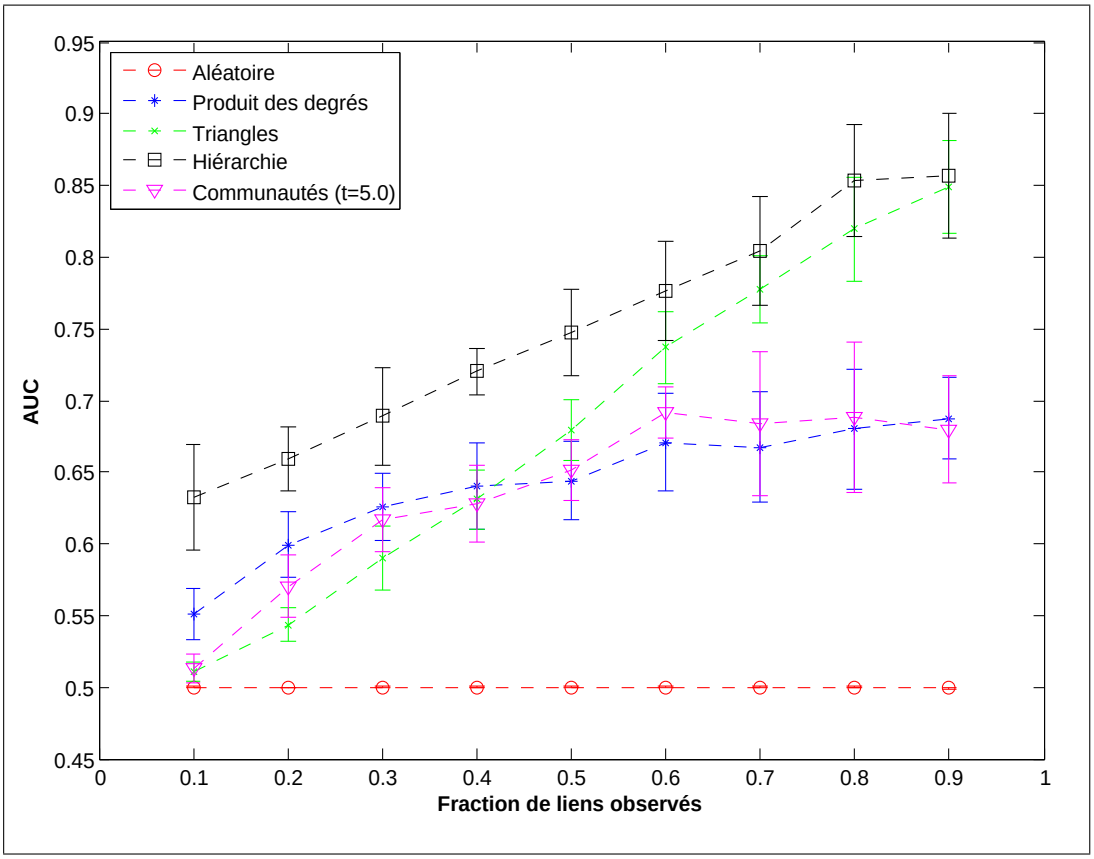

FIGURE 3.18 – Résultats de l'*AUC* pour les différentes méthodes de prédiction de liens manquants pour le réseau des terroristes des attaques du 11/09.

Une fois de plus, le graphe d'*AUC* de la méthode aléatoire est une ligne horizontale à valeur 0.5. Cette fois-ci, nous pouvons grouper les méthodes en trois classes : l'une est formée par la méthode hiérarchique, la deuxième est formée par la méthode des communautés et la méthode du produit des degrés pendant que la troisième classe est formée par la méthode des triangles.
La méthode des communautés et la méthode du produit des degrés sont très semblables en termes d'*AUC*. En moyenne, il semble que la méthode du produit des degrés soit mieux que la méthode des communautés pour les réseaux à fractions de liens observés inférieures, pendant que la méthode des communautés semble être, en général, meilleure que la méthode du produit des degrés pour des réseaux à fractions de liens observés supérieures à 0.5.
La méthode hiérarchique est, de nouveau, significativement la meilleure méthode, excepté la méthode des triangles à partir des fractions de liens observés supérieures à 0.5 ; la méthode hiérarchique reste quand même, en moyenne, la meilleure de ces

deux méthodes.
La courbe d'AUC de la méthode des triangles montre un comportement assez intéressant. Pour les petites fractions de liens observés, elle est, en moyenne, la méthode la moins efficace et elle est même significativement beaucoup moins efficaces que la méthode hiérarchique. A partir d'une fraction de liens observés de 0.4, elle devient significativement plus efficace que la méthode des communautés et la méthode du produit des degrés, et à partir d'une fraction de 0.7, la méthode des triangles devient même presque équivalente à la méthode hiérarchique en termes de valeurs d'AUC.
Remarquons que Clauset a appliqué la méthode des triangles, la méthode du produit des degrés et la méthode hiérarchique à ce même réseau des terroristes des attaques du 11 septembre dans [9] et notre étude confirme ses résultats. Cependant, il n'a appliqué cet algorithme qu'à une seule série de réseaux et il n'a donc pas publié les écarts-types des valeurs d'AUC. Si nous étudions les écarts-types de plus proche, nous remarquons d'une part les grands étendues des barres d'erreur de la méthode des communautés. De nouveau, cela veut dire que la méthode des communautés produit des résultats d'AUC qui diffèrent fortement en fonction de la structure du réseau. D'autre part, nous remarquons que l'étendue des barres d'erreur pour la méthode hiérarchique est, en moyenne, plus petite que pour le réseau de karaté (l'écart-type maximale vaut 0.04). Finalement, nous remarquons que, de nouveau, la méthode des triangles est la méthode la plus indépendante de la structure du réseau, puisque l'écart-type maximal est le plus petit pour ce réseau en valant 0.03.

3.6 Application au réseau des livres politiques d'*Amazon*

3.6.1 Comparaison des différents choix de t pour la méthode des communautés

La figure 3.19 affiche les résultats d'AUC pour la méthode des communautés avec différents choix du paramètre t.
Le résultat pour la méthode des communautés pour un choix de $t = 0.2$ est presque le même que celui pour les jeux de données analysés précédemment et ne nécessite aucune nouvelle explication. Nous remarquons que la méthode des communautés avec $t = 1.0$ est très efficace et que la courbe d'AUC croît de manière constante. La méthode avec $t = 5.0$ est aussi efficace surtout pour des réseaux de fractions de liens observés inférieures à 0.6. Pour toutes les fractions de liens observés supérieures à 0.6, le choix de $t = 1.0$ produit des résultats admettant des valeurs d'AUC plus grandes, mais admet des écarts-types plus élevés. Puisque les valeurs d'AUC de la méthode des communautés avec $t = 1.0$ sont presque partout moins bonnes

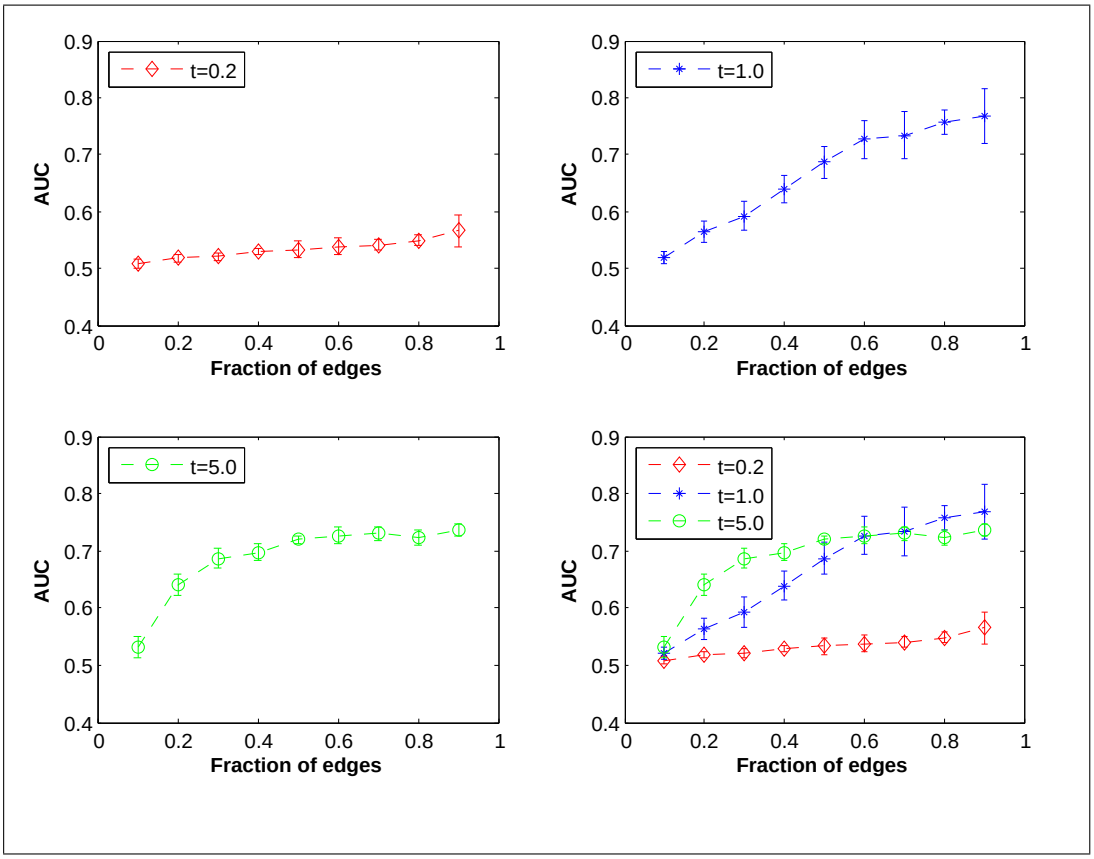

FIGURE 3.19 – Résultats de l'*AUC* pour la méthode des communautés pour le réseau des livres politiques.

qu'avec le choix de $t = 5.0$ et puisque, pour les fractions de liens observés pour lesquelles le choix de $t = 1.0$ donne des meilleures résultats que pour $t = 5.0$, les étendues des écart-types deviennent plus grandes, nous comparerons par la suite la méthode des communautés pour un choix de $t = 5.0$ aux autres méthodes de prédiction de liens manquants.

3.6.2 Comparaison des différentes méthodes de prédiction de liens manquants

Passons à la comparaison des différentes méthodes de prédiction de liens manquants. Les courbes d*AUC* de ces méthodes sont affichées dans les figures 3.20 et 3.21.

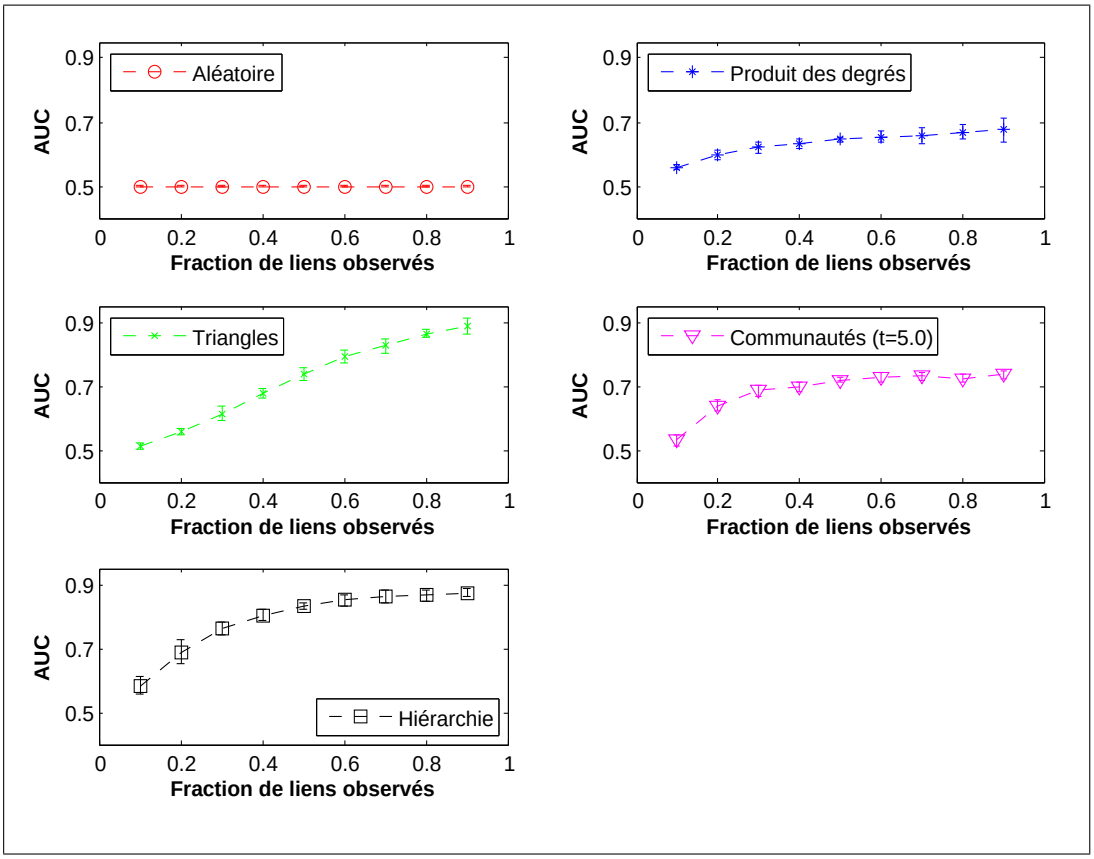

FIGURE 3.20 – Résultats de l'*AUC* pour les différentes méthodes de prédiction de liens manquants pour le réseau des livres politiques.

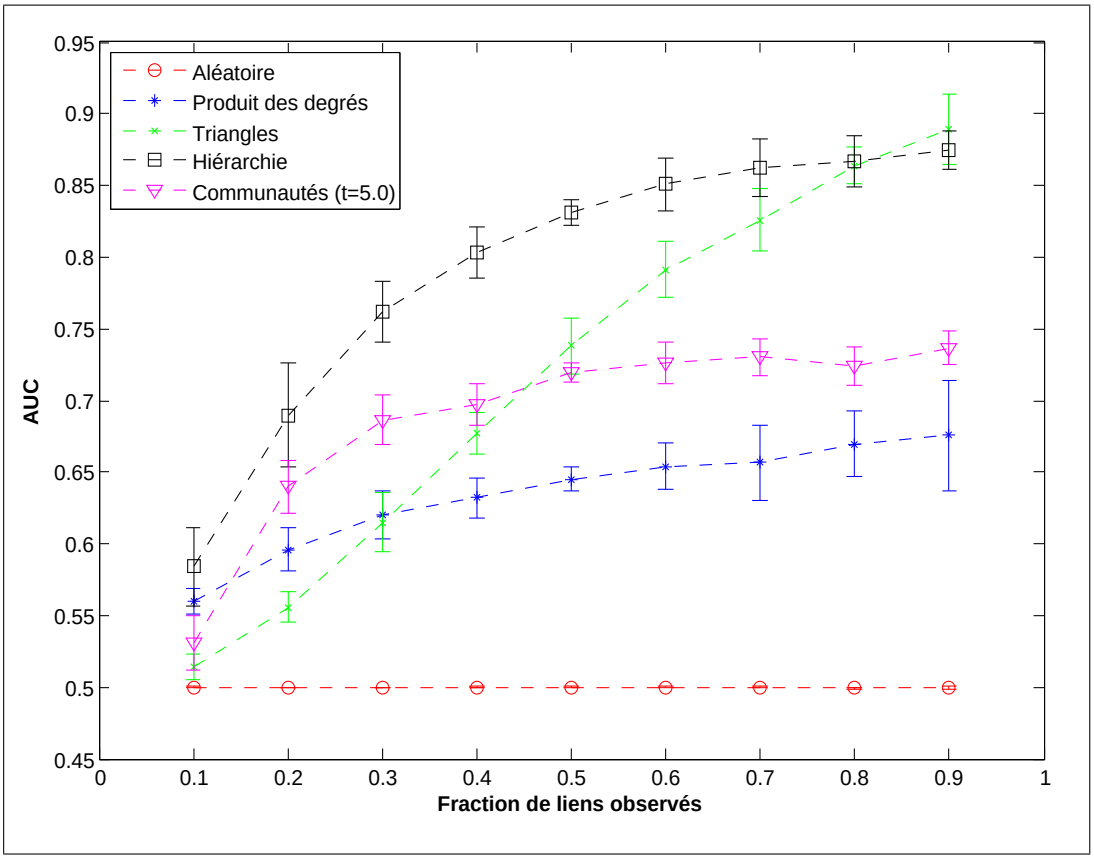

FIGURE 3.21 – Résultats de l'*AUC* pour les différentes méthodes de prédiction de liens manquants pour le réseau des livres politiques.

Tout comme pour les jeux de données traités précédemment, la courbe d'*AUC* de la méthode aléatoire est une droite horizontale coupant l'axe d'*AUC* en 0.5. Les comportements des courbes d'*AUC* des autres méthodes sont similaires à ceux qui résultent de l'application de ces méthodes au réseau des terroristes des attaques du 11/09. Cependant, cette fois-ci, la méthode des communautés est significativement meilleure que la méthode du produit des degrés. La courbe d'*AUC* de cette dernière méthode croît très légèrement, pendant que la courbe de la méthode des communautés croît fortement au début et se stabilise à partir des fractions de liens observés supérieures à 0.5.
La méthode des triangles est, comme pour le réseau des terroristes des attaques sur les Twin Towers, la méthode la moins efficace en termes d'*AUC* pour les réseaux à petites fractions de liens observés. A partir de la fraction de 0.4, cette méthode devient significativement plus efficace que la méthode du produit des degrés, à partir de 0.5 elle devient plus efficace que la méthode des communautés et à partir

de la fraction 0.8 même aussi efficace que la méthode hiérarchique, et même un peu plus efficace en moyenne.
La méthode hiérarchique, quant à elle, est significativement beaucoup plus efficaces que les autres méthodes pour les fractions de liens observés inférieures à 0.7. Les étendues des écarts-types des valeurs d'AUC sont relativement petites pour ce jeu de données. Cette fois-ci, la méthode des communautés admet le plus petit écart-type maximal (valeur de 0.01). Pour la méthode hiérarchique, les valeurs des écarts-types d'AUC restent raisonnables.

3.7 Application au réseau du championnat de football américain

3.7.1 Comparaison des différents choix de t pour la méthode des communautés

Les résultats d'AUC pour la méthode des communautés avec différents choix de t pour ce dernier réseau du championnat de football américain sont présentés sur la figure 3.22.

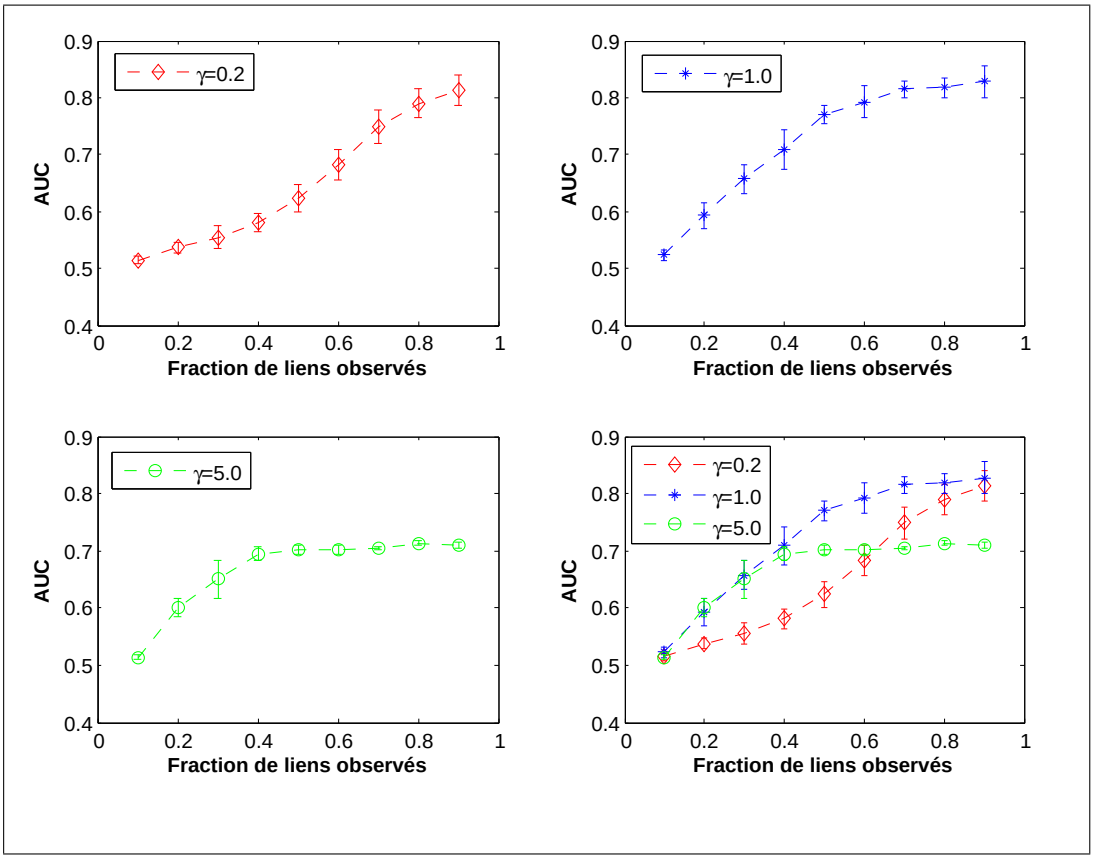

FIGURE 3.22 – Résultats de l'*AUC* pour la méthode des communautés pour le réseau du championnat de football américain.

Rappelons tout d'abord que ce réseau admet une structure naturelle de partition en communautés, à savoir la partition en différentes divisions. Nous remarquons directement que la méthode des communautés avec $t = 0.2$ est, pour la première fois, très efficace. En effet, nous déduisons du tableau de l'*Annexe A* que le nombre de classes est moins grand relative au nombre de nœuds. Autrement dit, pour ce réseau-ci, la méthode *Louvain* avec $t = 0.2$ propose une partition pour laquelle la relation entre le nombre et la taille des communautés est meilleure que pour la même méthode avec le même choix du paramètre appliquée aux autres réseaux. La courbe d'*AUC* croît avec la fraction des liens observés.
La courbe d'*AUC* de la méthode des communautés avec $t = 5.0$ montre le même comportement que pour le réseau des livres politiques. Elle croît avec la fraction de liens observés jusqu'à atteindre une valeur d'*AUC* maximale en 0.4.
Le meilleur choix du paramètre est dans ce cas $t = 1.0$, puisqu'il admet des valeurs d'*AUC* qui sont, en général, plus grandes que celles de la méthode des communau-

tés avec un autre choix de t. En plus, l'étendue des écarts-types reste raisonnable.

3.7.2 Comparaison des différentes méthodes de prédiction de liens manquants

Les deux figures 3.23 et 3.24 affichent les courbes d'AUC pour les méthodes de prédictions de liens manquants appliquées à ce dernier réseau.

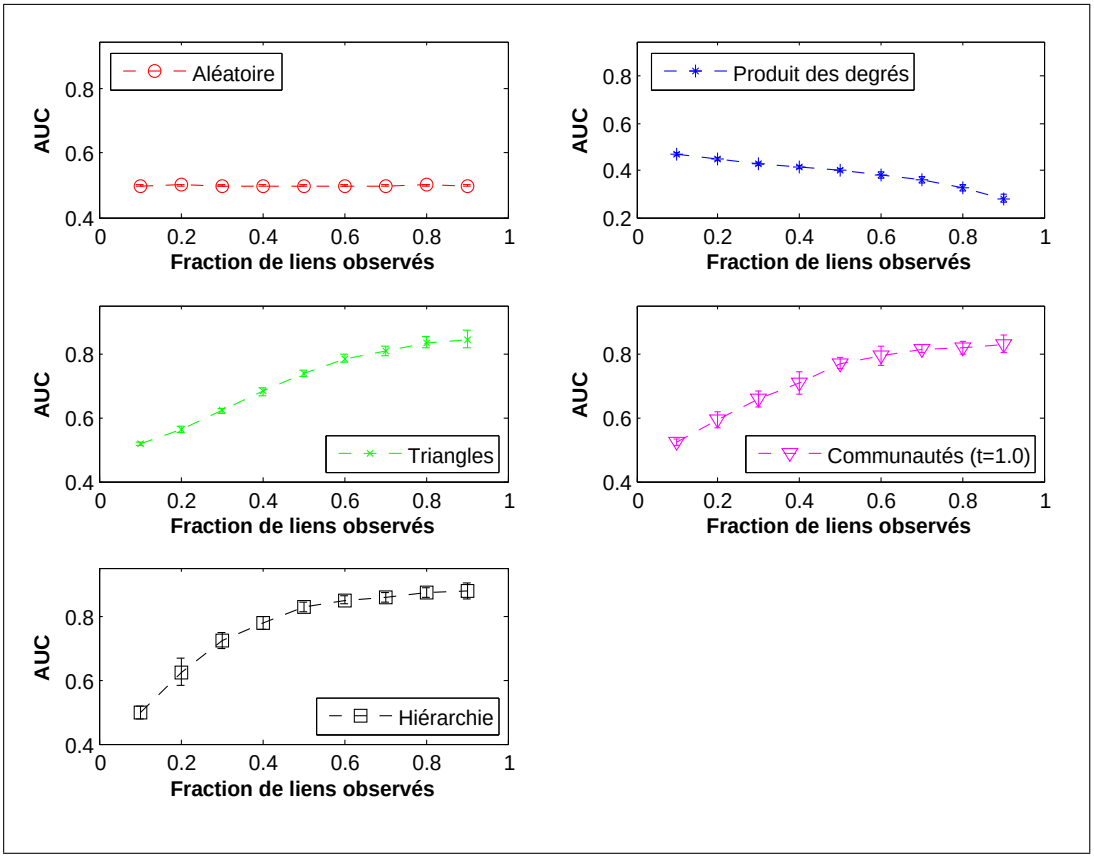

FIGURE 3.23 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau du championnat de football américain.

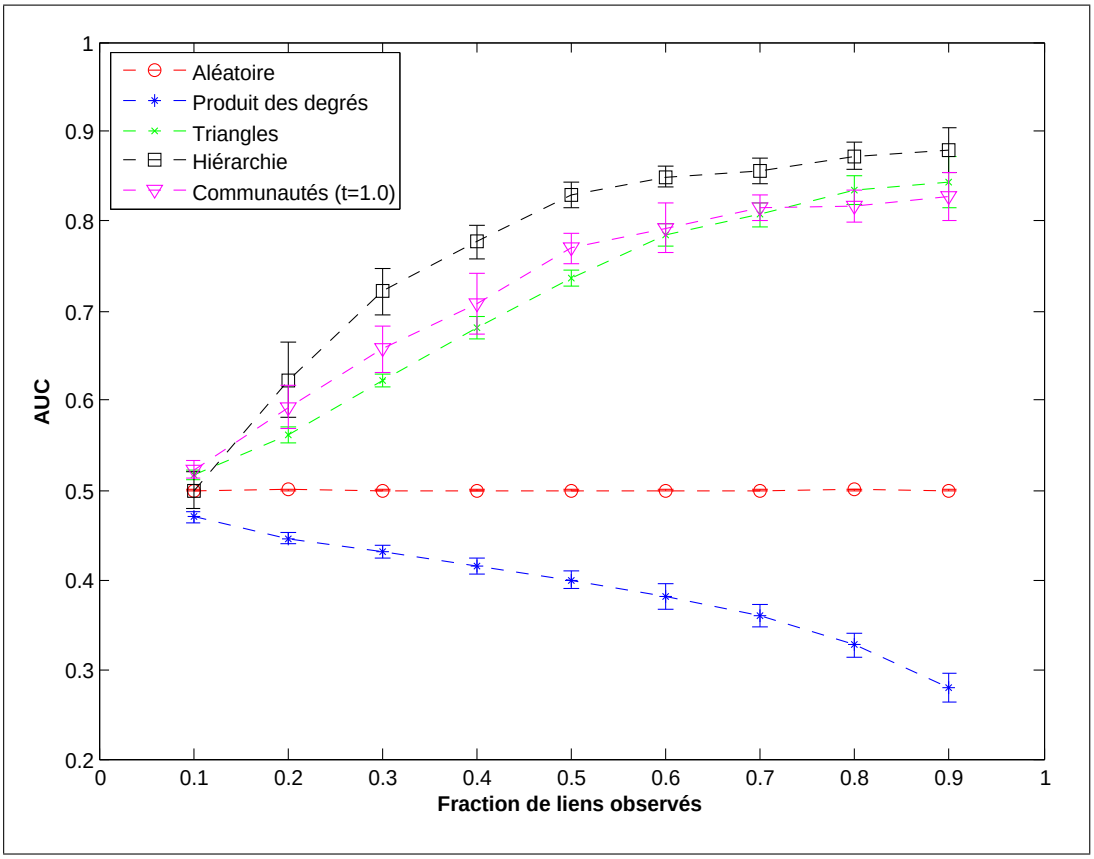

FIGURE 3.24 – Résultats de l'AUC pour les différentes méthodes de prédiction de liens manquants pour le réseau du championnat de football américain.

Parmi les 5 courbes d'AUC, il y en a une qui se distingue fondamentalement des autres : les prédictions de la méthode basée sur le produit des degrés des nœuds sont plus mauvaises que les arêtes prédites de la méthode aléatoire qui, quant à elle, admet comme courbe d'AUC de nouveau une droite horizontale qui coupe l'axe d'AUC en 0.5. Puisque le résultat associé est particulièrement intéressant, nous analyserons la méthode du produit des degrés plus tard dans cette section.
En ce qui concerne les autres méthodes, nous remarquons que toutes les 3 méthodes de prédiction non-aléatoire sont très efficaces pour ce jeu de données et ne diffèrent presque pas en termes d'AUC, sauf la méthode hiérarchique qui est significativement la meilleure parmi les méthodes de prédiction pour ce réseau. Remarquons que ce réseau du championnat de football américain est celui pour lequel la méthode des communautés fonctionne le mieux, ce qui est dû à la partition naturelle de ce réseau en divisions.
L'étendue de toutes les barres d'erreur est raisonnable.

Passons à la méthode du produit des degrés. La courbe d'AUC associée à cette méthode pour ce réseau décroît avec le nombre de liens observé. Autrement dit, si nous ajoutons de l'information (en termes d'arêtes) à notre réseau, la qualité des prédictions diminue, ce qui est contre-intuitif. Cependant, une analyse plus détaillée aide à comprendre le mécanisme qui mène à ce résultat.

Considérons un réseau artificiel d'un championnat de sport à deux divisions, chaque division contenant 5 équipes. Pour que chaque équipe rencontre chaque autre équipe dans un match, nous choisirons que le nombre d'arêtes à l'intérieur des divisions vaut 10. En plus, supposons que chaque équipe joue un match contre une équipe de l'autre division, le nombre d'arêtes d'une division à l'autre vaut donc 5. Ce réseau est illustré dans la figure 3.25. Même si les nombres d'équipes par divisions et donc les nombres de matchs par équipe dans le réseau de football sont légèrement différents, ce réseau artificiel illustre la structure du réseau de football. Puisque le nombre d'arêtes qui restent dans la même communauté est beaucoup plus grand que le nombre d'arêtes entre les deux divisions, il est beaucoup plus probable de sélectionner deux arêtes qui restent dans la même communauté que sélectionner au moins une arête qui quitte une communauté. En effet, si nous désignons par D l'événement de choisir une arête qui ne quitte pas une communauté, puisque le nombre d'arêtes qui restent dans une même communauté vaut 20 et le nombre d'arêtes quittant une communauté vaut 5, la probabilité d'en choisir 2 qui ne quittent pas une communauté est donnée par

$$P(D, D) = \frac{\binom{20}{2}}{\binom{25}{2}} = 0.6333\ldots$$

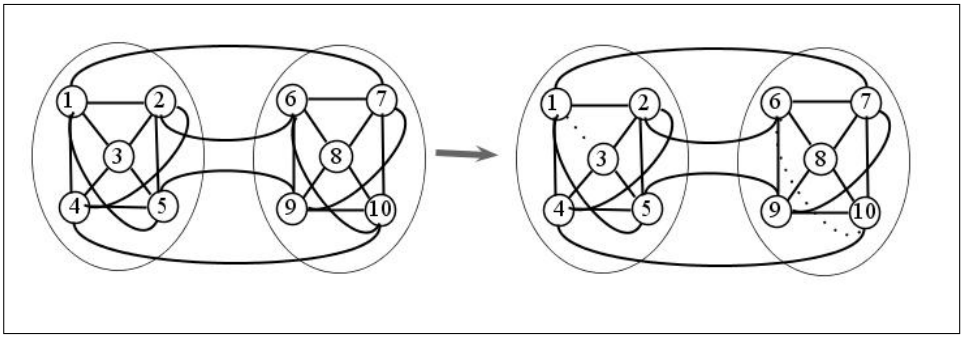

FIGURE 3.25 – Illustration de la méthode du produit des degrés.

Il est donc très probable que nous retirons par exemple les paires de nœuds $(1, 3)$ et $(6, 10)$ du réseau de départ, comme il est indiqué sur la figure 3.25. Appliquons ensuite une fonction à ce réseau qui calcule toutes les paires de nœuds

non-connectés et qui les classe de manière décroissante en fonction de leur produit des degrés. La liste complète de paires de nœuds pour cet exemple se trouve dans l'*Annexe C*. Nous remarquons que les paires de nœuds qui ont le plus grand produit des degrés sont les paires de nœuds qui se trouvent dans des communautés différentes, à savoir les paires $(2, 7)$, $(2, 8)$, $(2, 9)$, $(4, 7)$, $(4, 8)$, $(4, 9)$, $(5, 7)$ et $(5, 8)$ qui admettent tous un produit des degrés de 25. Puisque le fait d'avoir enlevé les arêtes diminue fortement le produit des degrés des paires de nœuds relatives, ces paires de nœuds auront un produit des degrés beaucoup plus petits que d'autres paires de nœuds. Pour cet exemple-ci, les arêtes retirées $(1, 3)$ et $(6, 10)$ admettent un produit des degrés de 16. Dès lors, la méthode de prédiction de liens manquants basée sur le produit des degrés considérera ces arêtes comme des prédictions plus mauvaises et proposera dès lors beaucoup d'autres (mauvaises) arêtes avant de tenir aussi compte de ces arêtes d'un score plus petit.
Remarquons encore une fois qu'il ne s'agit que d'un exemple et pas d'un sous-réseau du réseau réel, mais cet exemple nous aide à comprendre le mécanisme qui rend la méthode de prédiction de liens manquants basée sur le produit des degrés moins efficaces.

3.8 Récapitulatif des applications des méthodes de prédiction de liens manquants

Dans cette avant-dernière section du chapitre applicatif de ce mémoire, nous récapitulons les résultats trouvés ci-dessus pour chaque jeu de données.

- *Le réseau des terroristes grecs.* Pour ce jeu de données, la méthode hiérarchique est, en général, la méthode la plus efficace. Il semble que pour des réseaux à petites fractions de liens observés, la méthode du produit des degrés est presque aussi efficace que la méthode hiérarchique, mais elle ne croît pas constamment avec la fraction de liens observés. En plus, elle admet, surtout pour des réseaux à grandes fractions de liens observés, des écarts-types très élevés. La méthode des communautés est la méthode la moins efficace parmi les méthodes non-aléatoires et admet des barres d'erreur d'une très grande étendue, ce qui pourrait être dû à la valeur faible de modalité du réseau de départ. La méthode des triangles est très efficace surtout pour les réseaux à fraction de liens observés supérieures, à savoir à partir de 0.7. Elle devient presque aussi efficace que la méthode hiérarchique, mais l'étendue des barres d'erreur devient de plus en plus grande.
- *Le réseau karaté.* Pour ce jeu de données, la méthode hiérarchique est très significativement la meilleure en termes d'AUC et les résultats surtout pour les réseaux à petites fractions de liens observés sont très remarquables. Ce-

pendant, on ne peut pas négliger la grande valeur d'écart-type de la valeur d'AUC correspondant aux réseaux à fraction de liens observés de 0.9. La méthode du produit des degrés est la méthode la plus efficaces après la méthode hiérarchique pour des réseaux à petites fractions de liens observés mais, à partir d'une certaine fraction, les autres méthodes la dépassent en termes de valeur d'AUC. La méthode des communautés est efficace pour les réseaux à une fraction de liens observés de 0.9, mais elle semble être dépendante de la structure du réseau, puisque l'écart-type correspondant à la valeur d'AUC pour les réseaux à cette fraction de liens observés est très grand. La méthode des triangles croît légèrement, mais de manière constante avec la fraction de liens observés.

- *Le réseau des terroristes des attaques du 11/09.* La méthode hiérarchique est significativement la méthode la plus efficace en termes d'AUC pour ce jeu de données pour les réseaux de fractions de liens observés inférieures à 0.6. Pour les réseaux à fractions de liens observés supérieures à 0.6, la méthode des triangles, qui est encore moins efficace que les deux autres méthodes restantes pour des fractions inférieures à 0.4, devient presque aussi efficace que la méthode hiérarchique et admet des valeurs d'écarts-types plus petites que cette méthode-ci. La méthode du produit des degrés est, de nouveau, relativement efficace pour des réseaux à fractions de liens inférieures et n'augmente plus en termes de valeur d'AUC à partir d'une certaine fraction. Les résultats d'AUC de la méthode des communautés ne diffèrent pas significativement des résultats de la méthode du produit des degrés.
- *Le réseau des livres politiques.* Une fois de plus, la méthode hiérarchique est la méthode la plus efficace en termes d'AUC pour ce jeu de données pour des réseaux à fraction de liens observés inférieures à 0.8. Pour chaque fraction de liens supérieure à 0.8, la méthode des triangles dépasse même la méthode hiérarchique en termes de valeur d'AUC et, en plus, admet des valeurs d'écart-types plus petites que la méthode hiérarchique. Cependant, il semble que cette méthode ne soit pas la plus efficace pour des réseaux à fraction de liens observés inférieures. La méthode du produit des degrés est, en général, la méthode la moins efficaces pour ce jeu de données, la courbe d'AUC associée croît légèrement. Finalement, la méthode des communautés est relativement efficace pour ce jeu de données. Les valeurs d'AUC croient fortement avec la fraction de liens observés et se stabilisent après une certaine fraction autour d'un maximum d'AUC.
- *Le réseau du championnat de football américain.* La méthode hiérarchique est la méthode la plus efficace pour ce jeu de données. La différence entre la méthode des triangles et la méthode des communautés n'est pas significative. Le fait que les prédictions de la méthode du produit des degrés sont moins efficaces que celles de la méthode aléatoire est très intéressant.

3.9 Application économique dans la vie réelle

Jusqu'ici, nous avons appliqué les méthodes de prédiction de liens manquants en essayant de retrouver des arêtes que nous avons enlevées des réseaux de départ. Cependant, il est clair que cette démarche n'a pas d'intérêt dans la vie réelle et ne sert donc qu'à la comparaison des méthodes. Dans cette courte section, nous montrerons comment les méthodes de prédiction de liens manquants pourraient être utilisé dans la vie réelle, plus précisément dans le domaine de stratégies de marketing.

Considérons le réseau complet des livres politiques d'*Amazon*. Pour rappel, ce réseau contient 105 livres et 441 arêtes entre ceux-ci, une arête signifiant que ces deux livres ont été fréquemment achetés par le même client. Pour ce réseau, le pouvoir de savoir prédire le réseau à un instant de temps ultérieur est particulièrement intéressant puisque l'offreur pourrait ainsi prévoir ses achats dans le futur et en adapter ses méthodes de marketing. Dans cette section, nous appliquerons la méthode hiérarchique, comme étant la méthode la plus efficace et la méthode principale de ce mémoire, 1000 fois à ce réseau observé et nous associerons aux arêtes comme score les valeurs moyennes de vraisemblance de celles-ci. Nous analyserons les arêtes prédites qui admettent un score supérieur à 0.5. Ces informations sur les arêtes pourraient être utilisées par *Amazon* afin d'améliorer leurs suggestions d'achat de livres.

Un autre avantage de ce réseau qui nous sert à évaluer la qualité de ces prédictions est que chaque livre appartient à une classe dépendant de l'idéologie principale présente dans ce livre : soit conservateur (c), libéral (l) ou neutre (n). Dès lors, par intuition, la méthode devrait créer des paires de livres appartenant à la même classe idéologique. Le tableau de la figure 3.26 affiche le résultat de la méthode hiérarchique aplliquée à ce réseau, le score correspondant ainsi que leur appartenance aux différentes classes. Nous avons choisi de n'afficher que les 25 livres qui admettent un score supérieur à 0.5.

Nous remarquons que les quatre premières paires de livres admettent des valeurs de vraisemblance très élevées, en se souvenant qu'il s'agit d'une moyenne de 1000 valeurs. Dès lors, il est très probable qu'il s'agit d'une bonne prédiction de lien.
Il est très remarquable que les paires de nœuds sont toujours composées de livres appartenant à la même classe idéologique, excepté la dernière paire de nœuds. Ce fait augmente la qualité et la crédibilité des prédictions. Donc, si un client achète un livre politique sur *Amazon*, le libraire pourrait ajouter à la liste des suggestions d'achats tous les livres formant des paires de nœuds avec ce livre acheté.

House of Bush, House of Saud (l)	American Dynasty (l)	0,74878652
American Dynasty (l)	Fanatics and Fools (l)	0,73885051
House of Bush, House of Saud (l)	Against All Enemies (l)	0,64809407
American Dynasty (l)	Weapons of Mass Deception (l)	0,64543711
A National Party No More (c)	Slander (c)	0,61267852
A National Party No More (c)	Hollywood Interrupted (c)	0,61251256
A National Party No More (c)	The Savage Nation (c)	0,6119615
A National Party No More (c)	The O'Reilly Factor (c)	0,60720471
A National Party No More (c)	Why Courage Matters (c)	0,59617176
A National Party No More (c)	Shut Up and Sing (c)	0,59392778
The Price of Loyalty (l)	American Dynasty (l)	0,57686875
Dereliction of Duty (c)	The O'Reilly Factor (c)	0,55741208
Bush Country (c)	Why Courage Matters (c)	0,52631589
The Price of Loyalty (l)	The New Pearl Harbor (l)	0,51558966
Off with Their Heads (c)	Bias (c)	0,51161468
Off with Their Heads (c)	Let Freedom Ring (c)	0,51159901
Off with Their Heads (c)	Spin Sisters (c)	0,51136651
Off with Their Heads (c)	The Official Handbook Vast... (c)	0,51134844
Off with Their Heads (c)	The Enemy Within (c)	0,51015169
Off with Their Heads (c)	Endgame (c)	0,50991154

FIGURE 3.26 – Paires de livres prédites par la méthode hiérarchique pour le réseau des livres politiques.

Nous avons dès lors montré comment les méthodes de prédiction de liens manquants pourraient être utilisées pour améliorer les stratégies de marketing. Cependant, il est clair que cette petite application montrée dans cette section est loin d'être exhaustive : On pourrait par exemple envisager une combinaison de différentes méthodes de prédiction de liens manquants afin d'augmenter le nombre ainsi que la qualité des prédictions. En plus, le réseau des livres politiques est un réseau *non-pondéré*, c'est-à-dire il n'existe pas de notion de poids pour les arêtes, qui serait dans ce cas le nombre de fois que les paires de livres sont achetées par le même client. Une adaptation de nos méthodes aux réseaux pondérés augmenterait la qualité des prédictions, mais cela sort du cadre de ce travail.

Conclusion et perspectives

Dans ce mémoire, nous avons présenté, analysé et comparé cinq méthodes de prédiction de liens manquants. Nous nous sommes surtout concentrés sur la présentation du développement de la méthode hiérarchique puisque ses fondements mathématiques sont particulièrement intéressants. Nous avons essayé d'être le plus clair possible en illustrant les concepts théoriques par des exemples.

Les applications à des réseaux réels ont tout d'abord montré que toutes les méthodes de prédiction de liens manquants sont plus efficaces que la méthode aléatoire, à l'exception de la méthode du produit des degrés pour le réseau du championnat de football, dont nous avons analysé le comportement de manière détaillée.
Nous avons remarqué que les méthodes des triangles et du produit des degrés sont des méthodes locales qui dépendent seulement du voisinage direct des nœuds, pendant que la méthode des communautés et la méthode hiérarchique sont des méthodes globales qui dépendent de tout le réseau entier.
En général, nous avons constaté que la méthode du produit des degrés est particulièrement efficace pour des réseaux à fraction de liens observés faibles et les valeurs d'AUC se stabilisent pour des réseaux à fractions de liens observés supérieures. Cependant, les résultats concernant le réseau des livres politiques nous indique que l'efficacité de cette méthode peut varier avec la topologie du réseau : cet exemple nous a indiqué que l'efficacité de cette méthode en termes de la mesure AUC diminue lorsque la valeur de modularité du réseau de départ croît.
La méthode des communautés, qui, à notre connaissance, n'a pas encore été utilisée sous cette forme, est moyennement efficace et semble bien fonctionner pour des graphes admettant une grande valeur de modularité. En effet, nous avons observé que l'efficacité de la méthode des communautés augmente avec les valeurs de modularité. En ce qui concerne le paramètre t que nous avons fait varier dans les applications, nous avons constaté qu'un choix trop petit pour ce paramètre réduit l'efficacité de la méthode. Cependant, nous avons trouvé des bons résultats pour $t = 1.0$ et $t = 5.0$.

Nous avons remarqué que la méthode des triangles n'est pas très efficace pour des réseaux à fractions de liens observés inférieures par rapport à toutes les autres méthodes. Cependant, pour des réseaux à fractions de liens observés supérieures à 0.7, elle devient très efficace et il s'agit, en général, de la seconde meilleure méthode considérée dans ce mémoire.

Ces résultats correspondent bien aux résultats des travaux antérieurs dans lesquels cette méthode a été appliquée, par exemple dans [9, 20]. De plus, nous avons remarqué que les écarts-types associés aux valeurs moyennes d'AUC sont toujours très faibles, ce qui indique une indépendance par rapport à la structure des réseaux. Puisque la méthode des triangles est très efficace pour tous les réseaux à fractions de liens observés supérieures, nos calculs ont suggéré que cette méthode ne dépend pas du coefficient de clustering du réseau de départ, ni du nombre de triangles présents dans celui-ci.
Un des points centraux de ce mémoire est l'étude de la méthode hiérarchique de Clauset. Nous avons remarqué que, en général, cette méthode est de loin la meilleure en termes d'AUC pour tous les réseaux considérés et pour n'importe quelle fraction de liens observés. Même si les écarts-types correspondants sont parfois élevés, la méthode produit toujours les meilleures prédictions. L'efficacité de cette méthode pour des réseaux à fraction de liens observés faibles est aussi très remarquable. Puisque cette méthode marche très bien dans de nombreuses systèmes, l'hypothèse du modèle, à savoir le caractère hiérarchique des réseaux, est validée.
Finalement, l'application sur le réseau des livres politiques a montré l'utilité de la prédiction de liens manquants dans la vie réelle. Les prédictions fournies par la méthode hiérarchique sont très probablement des liens manquants à cause de leurs grandes valeurs de vraisemblance et à cause du fait que la méthode prédit toujours une arête entre des paires de livres appartenant à la même classe idéologique.

Pour aller plus loin dans ce travail, il serait intéressant d'étudier de manière plus détaillée la variation des méthodes de prédiction de liens manquants en fonction de certaines propriétés, comme par exemple le coefficient de clustering. On pourrait envisager des réseaux artificiels dont on décide d'avance le coefficient de clustering et montrer ainsi la variation de l'efficacité des méthodes en fonction de cette propriété. Il serait aussi très intéressant d'analyser de manière plus détaillée la conjecture ci-dessus concernant la dépendance entre la méthode du produit des degrés et de la valeur de modularité.
Puisque le réseau le plus grand que nous avons considéré dans ce mémoire admet 115 nœuds et 613 arêtes, il serait aussi intéressant d'étudier l'efficacité des méthodes sur des réseaux de très grandes tailles, à savoir des milliers de nœuds (ou plus). Pour une telle analyse, les codes que nous avons ajoutés dans l'*Annexe B*

devraient être modifiés afin de réduire le temps de calcul et la manière dont on stocke les données.

Finalement, une piste de recherche particulièrement intéressante serait l'adaptation de ces méthodes de prédiction de liens manquants à des réseaux pondérés. Si nous reconsidérons l'exemple du réseau des livres politiques d'*Amazon*, un réseau pondéré donnerait des informations encore plus détaillée, ce qui améliorerait probablement la qualité des prédictions. De plus, puisque beaucoup des réseaux de la vie réelle sont pondérés, cette adaptation élargirait fortement le champ d'application des méthodes de prédictions de liens manquants. Le problème de prédiction de liens manquants deviendrait dès lors une prédiction de poids de liens dans ces réseaux pondérés.

Annexe A

Méthode des communautés : Tableaux relatifs aux explications des résultats

A.1 Réseau des terroristes grecs - $t = 0.2$

label du nœud	label de la communauté
1	1
3	2
4	3
2	4
7	5
6	6
9	6
5	7
10	7
19	7
11	8
12	9
13	10
14	11
15	12
8	13
16	13
17	13
20	13
18	14
21	15
22	16

A.2 Réseau des terroristes grecs - $t = 5.0$

	Fraction de liens observés								
Label des noeuds	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1	1	1	1	2	2	2	2	2	2
2	9	4	4	2	2	2	2	2	2
3	6	4	4	2	2	2	2	2	2
4	1	1	1	2	2	2	2	2	2
5	2	3	3	2	2	2	2	2	2
6	9	4	4	2	2	2	2	2	2
7	3	4	4	2	2	2	2	2	2
8	4	2	2	1	1	1	1	1	1
9	9	4	4	2	2	2	2	2	2
10	5	3	3	2	2	2	2	2	2
11	6	4	4	2	2	2	2	2	2
12	7	4	4	2	2	2	2	2	2
13	1	1	1	2	2	2	2	2	2
14	9	4	4	2	2	2	2	2	2
15	8	4	4	2	2	2	2	2	2
16	9	4	4	2	2	2	2	2	2
17	9	4	4	2	2	2	2	2	2
18	10	5	4	2	2	2	2	2	2
19	11	6	5	2	2	2	2	2	2
20	12	7	6	3	3	3	3	3	2
21	13	8	7	4	4	2	2	2	2
22	9	4	4	2	2	2	2	2	2

FIGURE A.1 – Partition en communautés en fonction des fractions de liens observés

A.3 Réseau des terroristes des attaques du 11/09 - $t = 1.0$ et $t = 5.0$

Les nombres de communautés en fonction des fractions de liens observés.

Fraction de liens observés	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$t = 1.0$	44	35	25	20	15	9	6	6	7
$t = 5.0$	44	32	23	16	12	6	3	3	3

A.4 Réseau du championnat de football américain - $t = 0.2$

Les nombres de communautés en fonction des fractions de liens observés.

Fraction de liens observés	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$t = 0.2$	60	34	26	26	19	20	16	13	14

Annexe B

Codes en *Matlab*

B.1 Fonction calculant des arêtes aléatoires

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cette fonction cherche de maniere aleatoire a partir d'une matrice
%d'adjacence un certain nombre de liens non-connectes
%
%Input  : A              : la matrice d'adjacence
%         nbr_predictions: le nombre de liens a predire
%Output : pairs          : la liste des liens predits
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function[pairs]=randomPairs(A,nbr_predictions)

nbr_vertex=length(A(:,1));
%le nombre de noeuds

v=1;
stop=1;
%variable de sortie

while stop<=nbr_predictions

  %Nous calculons d'abord deux entiers dans l'intervalle [1,nbr_vertex],
  %cad deux noeuds aleatoires.
    i=ceil(rand*(nbr_vertex));
    j=ceil(rand*(nbr_vertex));
```

```
    %Lorsque ces deux noeuds aleatoires sont differents l'un de l'autre et
    %s'il n'y a pas d'arete entre cette paire de noeuds, on la rajoute a
    %notre liste
    if(i~=j)&(A(i,j)==0)
        pairs(stop,1)=i;
        pairs(stop,2)=j;
        stop=stop+1;
        A(i,j)=1;
        A(j,i)=1;
        %Necessaire pourque l'algorithme ne predise pas deux fois la meme
        %paire de noeuds
    end
end
end
```

B.2 Programme calculant les valeurs d'AUC pour la méthode aléatoire

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ce programme calcule les valeurs d'AUC pour un reseau pour la methode de
%prediction aleatoire.
%
%La fonction random_pairs nous donnera la liste des aretes choisies
%aleatoirement et ce programme calculera les valeurs d'AUC correspondantes
%et calculera, tout a la fin, les moyennes et les ecarts types pour les
%fractions de liens observees pour les 10 series de graphes.
%
%Input : 10 series de 9 reseaux, en forme de matrice d'adjacence comme la
%        serie produite par le programme construction_lists ainsi que la
%        matrice d'adjacence du reseau complet, le reseau du depart
%Output: Une matrice appelee AUC (12x9). Chaque serie de reseaux sera
%        presentee sur une nouvelle ligne, chaque colonne sera une fraction
%        de liens observes.
%        Les lignes 11 et 12 seront composees de la moyenne et de l'ecart
%        type pour chaque colonne.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear all
close all
clc


%La variable reseau est une variable de choix indiquant le reseau avec
%lequel l'utilisateur souhaite travailler. Possibilites de choix ici:
%   1: Reseau des terroristes grecs
%   2: Reseau karate de Zachary
%   3: Reseau des terroristes des attaques du 09/11
%   4: Reseau des equipes americaines de football
%   5: Reseau des livres politiques
%
%NB: Les reseaux doivent se trouver dans le meme ordre que ce programme, et
%    doivent etre sous formes d'un fichier .txt comprennant la liste
%    des aretes du reseau. En plus, dans la structure switch dans le
%    programme principale, l'utilisateur doit indiquer le nombre de liens
%    qu'il souhaite enlever a chaque etape (ici: 10% par defaut) ainsi que
%    le nombre de noeuds du reseau


reseau=input(strcat('Avec quel reseau voulez-vous travailler? \n',...
    '1 pour le reseau des terroristes grecs \n',...
    '2 pour le reseau karate de Zachary \n',...
    '3 pour le reseau des terroristes des attaques du 09/11 \n',...
    '4 pour le reseau des livres politiques d''Amazon \n',...
    '5 pour le reseau des equipes americaines de football\n \n'));


switch reseau
    case {1}
        data=importdata('greek_matrix_obs.txt');
    case {2}
        data=importdata('karate_matrix_obs.txt');
    case {3}
        data=importdata('terrorist_matrix_obs.txt');
    case {4}
        data=importdata('polbooks_matrix_obs.txt');
    case {5}
```

```
        data=importdata('football_matrix_obs.txt');
end
%data contient la matrice d'adjacence du reseau observe


%-------------------------------------------------------------------------
%          Calcul des nombres de paires non-connectees dans data
TN=0;
for i=1:length(data)
    for j=i:length(data)
        if data(i,j)==0 && (i~=j)
            TN=TN+1;
        end
    end
end
%         Fin Calcul des nombres de paires non-connectees dans data
%-------------------------------------------------------------------------




for fichier=1:10
    for fraction_of_edges=1:9

        %-------------------------------------------------------------------
        %   Initialisation des variables en fonction du choix du reseau
        %
        switch reseau
            case {1}
                A=importdata(strcat('greek_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=6;
                nbr_vertex=22;


            case {2}
                A=importdata(strcat('karate_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=7;
```

```
            nbr_vertex=34;


        case {3}
            A=importdata(strcat('terrorist_',num2str(fichier),...
                '_matrix_0',num2str(fraction_of_edges),'.txt'));

            removed_links=15;
            nbr_vertex=62;


        case {4}
            A=importdata(strcat('polbooks_',num2str(fichier),...
                '_matrix_0',num2str(fraction_of_edges),'.txt'));

            removed_links=44;
            nbr_vertex=105;


        case {5}
            A=importdata(strcat('football_',num2str(fichier),...
                '_matrix_0',num2str(fraction_of_edges),'.txt'));

            removed_links=61;
            nbr_vertex=115;


    end

    %  End Initialisationdes variables en fonction du choix du reseau
    %-----------------------------------------------------------------

    links=(10-fraction_of_edges)*removed_links;
    %links est nombre des liens a predire! Le nombre de liens enleves
    %vaut (10-fraction_of_edges)*removed_links


    %-----------------------------------------------------------------
    %             Boucle sur le nombre de liens a predire
    %Les valeurs de TPR et FPR seront enregistrees dans un tableau
```

```
%nombre maximal de predictions a faire: le nombre de paires de
%noeuds non-connectes dans le reseau de depart plus le nombre de
%liens qu'on en a enleve
for l=2:1:(TN+links)
    nbr_predictions=l;
    network=A;
    TPR=0;
    FPR=0;
    pairs=randomPairs(network,nbr_predictions);
    %pairs contiens une liste de nbr_predictions paires de noeuds
    %non-connectes choisies de maniere aleatoire. Il s'agit de nos
    %predictions.

    %---------------------------------------------------------------
    %                    Calcul des true positiv
    TP=0;
    for i=1:nbr_predictions
        a=pairs(i,1);
        b=pairs(i,2);
        if data(a,b)==1
            TP=TP+1;
        end
    end
    %                   Fin Calcul des true positiv
    %---------------------------------------------------------------


    %Calcul des TPR et FPR:
    TPR=TP/(TP+(links-TP));

    FP=nbr_predictions-TP;
    FPR=(FP)/(FP+(TN-FP));


    array_TPR(l-1)=TPR;
    array_FPR(l-1)=FPR;
end



[p1 p2]=polyfit(array_FPR,array_TPR,1);
```

```
        %p1 et p2 seront les coefficients de la droite de regression
        x=[0:0.1:1];
        p=p1(1)*x+p1(2);

        %Calcul de l'aire sous la courbe:
        AUC(fichier,fraction_of_edges)=trapz(x,p);

    end
end

%Calcul des moyennes et ecarts-types:
for i=1:9
    AUC(11,i)=mean(AUC(1:10,i));
    AUC(12,i)=std(AUC(1:10,i));
end
```

B.3 Fonction calculant des arêtes en fonction du produit des degrés

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cette fonction calcule une liste d'aretes en fonction du produit des
%degres des noeuds.
%
%Input : A              : La matrice d'adjacence
%Output: list           : La liste composee de toutes les aretes et leur
%                         produit des degres
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



function[list]=degreeproductPairs(A)

nbr_nodes=length(A);
a=1;
degree_i=0;
degree_j=0;

for i=1:nbr_nodes
```

```
    for j=i:nbr_nodes

        if A(i,j)==0 && i~=j
            %-------------------------------------------------------------
            %            Calcul des degres des noeuds connectes
            list(a,1)=i;
            list(a,2)=j;
            for k=1:nbr_nodes
                if A(i,k)==1
                    degree_i=degree_i+1;
                end
                if A(j,k)==1
                    degree_j=degree_j+1;
                end
            end
            %                       Fin Calcul des degres
            %-------------------------------------------------------------
            list(a,3)=degree_i*degree_j;
            %Calcul du score

            degree_i=0;
            degree_j=0;
            a=a+1;
        end
    end
end

list=sortrows(list,-3);
%list sera une liste d'aretes decroissantes en fonction du produit des
%degres

end
```

B.4 Programme calculant les valeurs d'AUC pour la méthode du produit des degrés

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ce programme calcule les valeurs d'AUC pour un reseau pour la methode de
%prediction basee sur le produit des degres.
```

```
%
%La fonction degreeproduct_pairs nous donnera la liste des aretes avec un
%score en fonction du produit des degres des aretes et ce programme
%calculera les valeurs d'AUC correspondantes et calculera, tout a la fin,
%les moyennes et les ecarts types pour les fractions de liens observees
%pour les 10 series de graphes.
%
%Input : 10 series de 9 reseaux, en forme de matrice d'adjacence comme la
%        serie produite par le programme construction_lists ainsi que la
%        matrice d'adjacence du reseau complet, le reseau du depart
%Output: Une matrice appelee AUC (12x9). Chaque serie de reseaux sera
%        presentee sur une nouvelle ligne, chaque colonne sera une fraction
%        de liens observes.
%        Les lignes 11 et 12 seront composees de la moyenne et de l'ecart
%        type pour chaque colonne.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
close all
clc



%La variable reseau est une variable de choix indiquant le reseau avec
%lequel l'utilisateur souhaite travailler. Possibilites de choix ici:
%   1: Reseau des terroristes grecs
%   2: Reseau karate de Zachary
%   3: Reseau des terroristes des attaques du 09/11
%   4: Reseau des equipes americaines de football
%   5: Reseau des livres politiques
%
%NB: Les reseaux doivent se trouver dans le meme ordre que ce programme, et
%    doivent etre sous formes d'un fichier .txt comprennant la liste
%    des aretes du reseau. En plus, dans la structure switch dans le
%    programme principale, l'utilisateur doit indiquer le nombre de liens
%    qu'il souhaite enlever a chaque etape (ici: 10% par defaut) ainsi que
%    le nombre de noeuds du reseau


reseau=input(strcat('Avec quel reseau voulez-vous travailler? \n',...
```

```
    '1 pour le reseau des terroristes grecs \n',...
    '2 pour le reseau karate de Zachary \n',...
    '3 pour le reseau des terroristes des attaques du 09/11 \n',...
    '4 pour le reseau des equipes americaines de football \n',...
    '5 pour le reseau des livres politiques d''Amazon \n \n'));


switch reseau
    case {1}
        data=importdata('greek_matrix_obs.txt');
    case {2}
        data=importdata('karate_matrix_obs.txt');
    case {3}
        data=importdata('terrorist_matrix_obs.txt');
    case {4}
        data=importdata('polbooks_matrix_obs.txt');
    case {5}
        data=importdata('football_matrix_obs.txt');
end
%data contient la matrice d'adjacence du reseau observe et nous servira
%pour voir si la methode a detecte les aretes retirees


%-------------------------------------------------------------------------
%          Calcul des nombres de paires non-connectees dans data

TN=0;
for i=1:length(data)
    for j=i:length(data)
        if data(i,j)==0 && (i~=j)
            TN=TN+1;
        end
    end
end
%         Fin Calcul des nombres de paires non-connectees dans data
%-------------------------------------------------------------------------



for fichier=1:10
    for fraction_of_edges=1:9
```

```
%-----------------------------------------------------------------
%   Initialisation des variables en fonction du choix du reseau
%
switch reseau
    case {1}
        A=importdata(strcat('greek_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));

        removed_links=6;
        nbr_vertex=22;


    case {2}
        A=importdata(strcat('karate_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));

        removed_links=7;
        nbr_vertex=34;


    case {3}
        A=importdata(strcat('terrorist_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));

        removed_links=15;
        nbr_vertex=62;


    case {4}
        A=importdata(strcat('polbooks_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));

        removed_links=44;
        nbr_vertex=105;


    case {5}
        A=importdata(strcat('football_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));
```

```
        removed_links=61;
        nbr_vertex=115;


end

% End Initialisation des variables en fonction du choix du reseau
%-----------------------------------------------------------------

links=(10-fraction_of_edges)*removed_links;
%links est nombre des liens a predire! Le nombre de liens enleves
%vaut (10-fraction_of_edges)*removed_links

list=degreeproductPairs(A);
%list contient toutes les paires de noueds non-connectes du reseau,
%ordonnees de maniere decroissante en fonction de leur produit des
%degres


%-----------------------------------------------------------------
%             Boucle sur le nombre de liens a predire
%Les valeurs de TPR et FPR seront enregistrees dans un tableau

%nombre maximal de predictions a faire: le nombre de paires de
%noeuds non-connectes dans le reseau de depart plus le nombre de
%liens qu'on en a enleve
for l=2:(TN+links)
    nbr_predictions=l;

    pairs=fillup(list,nbr_predictions);
    %pairs contient les predictions pour ce nbr_predictions. La
    %fonction fillup choisi nbr_predictions paires parmi toutes les
    %paires de la variable list selon un certain critere. Cf fillup
    %pour plus de details

    TPR=0;
    FPR=0;

    %-------------------------------------------------------------
    %                   Calcul des true positiv
```

```
            TP=0;
            for i=1:nbr_predictions
                a=pairs(i,1);
                b=pairs(i,2);
                if data(a,b)==1
                    TP=TP+1;
                end
            end
            %                  Fin Calcul des true positiv
            %-------------------------------------------------------------


            %Calcul des TPR et FPR:

            TPR=TPR+TP/(TP+(links-TP));

            FP=nbr_predictions-TP;
            FPR=FPR+(FP)/(FP+(TN-FP));


            array_TPR(l-1)=TPR;
            array_FPR(l-1)=FPR;
        end


        [p1 p2]=polyfit(array_FPR,array_TPR,1);
        %p1 et p2 seront les coefficients de la droite de regression
        x=[0:0.1:1];
        p=p1(1)*x+p1(2);

        %Calcul de l'aire sous la courbe:
        AUC(fichier,fraction_of_edges)=trapz(x,p)

    end
end

%Calcul des moyennes et ecarts-types:
for i=1:9
    AUC(11,i)=mean(AUC(1:10,i));
    AUC(12,i)=std(AUC(1:10,i));
end
```

B.5 Fonction calculant des arêtes en fonction des nombres de triangles fermés

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cette fonction calcule une liste d'aretes qui ferment au moins un
%triangle. Nous calculons le carre de la matrice d'adjacence pour connaitre
%les noeuds lies par des chemins de longueur 2
%
%Input : A              : La matrice d'adjacence
%        nbr_predictions: Le nombre de predictions qu'on souhaite faire
%Output: B              : La matrice d'adjacence modifiee
%        list           : La liste composee de toutes les aretes fermant
%                         au moins un triangle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[B,list]=trianglePairs(A,nbr_predictions)

B=A;
%B sera la matrice d'adjacence A, mais nous ajoutons une arete a chaque
%arete fermant au moins un triangle

A2=B*B;

list=[0 0];
a=0;

for i=1:length(A2)
    A2(i,i)=-1;
    %Nous mettons -1 sur la diagonale de A^2 pour rassurer qu'on ne
    %considerera pas des aretes liant un noeuds a lui meme
end

matr_int=A2;

%La boucle est sur le nombre de triangles qu'une arete ferme, commencant
%par les aretes fermant le plus grand nombre de triangles
while max(max(matr_int))~=0
```

```
    max_actuel=max(max(matr_int));
    for i=1:length(matr_int)
        for j=i:length(matr_int)
            %-------------------------------------------------------------
            %   Boucle considerant les aretes fermant le meme nombre de
            %                           triangles
            if matr_int(i,j)==max_actuel
                %Si on a trouve une arete fermant le nombre considere de
                %triangles, nous l'ajoutons a la liste et nous changeons la
                %valeur de la matrice d'adjacence intermediare pour ne plus
                %considerer cette arete au cours de cette boucle

                matr_int(i,j)=-1;
                matr_int(j,i)=-1;
                if B(i,j)==0
                    %Si i et j ne sont pas connectes dans le reseau, on les
                    %ajoute a la liste des paires
                    list(a+1,1)=i;
                    list(a+1,2)=j;
                    B(i,j)=1;
                    B(j,i)=1;
                    %On change la valeur corresondante de la matrice
                    %d'adjacence pour ne plus considerer cette arete dans
                    %le cas d'une selection aleatoires d'aretes
                    %supplementaires

                    a=a+1;
                end
            end
            %                           End Boucle
            %-------------------------------------------------------------
        end
    end
end

%Nous ajoutons le score des paires de noeuds a la liste
for i=1:length(list(:,1))
    a=list(i,1);
    b=list(i,2);
    list(i,3)=A2(a,b);
end
```

```
end
```

B.6 Programme calculant les valeurs d'AUC pour la méthode des triangles

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ce programme calcule les valeurs d'AUC pour un reseau pour la methode de
%prediction basee sur les triangles.
%
%La fonction triangle_pairs nous donnera la liste des aretes avec un
%score en fonction du produit des degres des aretes et ce programme
%calculera les valeurs d'AUC correspondantes et calculera, tout a la fin,
%les moyennes et les ecarts types pour les fractions de liens observees
%pour les 10 series de graphes.
%
%Input : 10 series de 9 reseaux, en forme de matrice d'adjacence comme la
%        serie produite par le programme construction_lists ainsi que la
%        matrice d'adjacence du reseau complet, le reseau du depart
%Output: Une matrice appelee AUC (12x9). Chaque serie de reseaux sera
%        presentee sur une nouvelle ligne, chaque colonne sera une fraction
%        de liens observes.
%        Les lignes 11 et 12 seront composees de la moyenne et de l'ecart
%        type pour chaque colonne.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
close all
clc


%La variable reseau est une variable de choix indiquant le reseau avec
%lequel l'utilisateur souhaite travailler. Possibilites de choix ici:
%   1: Reseau des terroristes grecs
%   2: Reseau karate de Zachary
%   3: Reseau des terroristes des attaques du 09/11
%   4: Reseau des equipes americaines de football
%   5: Reseau des livres politiques
%
```

```
%NB: Les reseaux doivent se trouver dans le meme ordre que ce programme, et
%    doivent etre sous formes d'un fichier .txt comprennant la liste
%    des aretes du reseau. En plus, dans la structure switch dans le
%    programme principale, l'utilisateur doit indiquer le nombre de liens
%    qu'il souhaite enlever a chaque etape (ici: 10% par defaut) ainsi que
%    le nombre de noeuds du reseau


reseau=input(strcat('Avec quel reseau voulez-vous travailler? \n',...
    '1 pour le reseau des terroristes grecs \n',...
    '2 pour le reseau karate de Zachary \n',...
    '3 pour le reseau des terroristes des attaques du 09/11 \n',...
    '4 pour le reseau des equipes americaines de football \n',...
    '5 pour le reseau des livres politiques d''Amazon \n \n'));


switch reseau
    case {1}
        data=importdata('greek_matrix_obs.txt');
    case {2}
        data=importdata('karate_matrix_obs.txt');
    case {3}
        data=importdata('terrorist_matrix_obs.txt');
    case {4}
        data=importdata('polbooks_matrix_obs.txt');
    case {5}
        data=importdata('football_matrix_obs.txt');
end
%data contient la matrice d'adjacence du reseau observe et nous servira
%pour voir si la methode a detecte les aretes retirees


%-------------------------------------------------------------------------
%          Calcul des nombres de paires non-connectees dans data

TN=0;
for i=1:length(data)
    for j=i:length(data)
        if data(i,j)==0 && (i~=j)
            TN=TN+1;
        end
```

```
    end
end
%         Fin Calcul des nombres de paires non-connectees dans data
%-------------------------------------------------------------------------



for fichier=1:10
    for fraction_of_edges=1:9
        %-------------------------------------------------------------------
        %   Initialisation des variables en fonction du choix du reseau
        %
        switch reseau
            case {1}
                A=importdata(strcat('greek_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=6;
                nbr_vertex=22;


            case {2}
                A=importdata(strcat('karate_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=7;
                nbr_vertex=34;


            case {3}
                A=importdata(strcat('terrorist_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=15;
                nbr_vertex=62;


            case {4}
                A=importdata(strcat('polbooks_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));
```

```
            removed_links=44;
            nbr_vertex=105;


        case {5}
            A=importdata(strcat('football_',num2str(fichier),...
                '_matrix_0',num2str(fraction_of_edges),'.txt'));

            removed_links=61;
            nbr_vertex=115;


    end

    % End Initialisation des variables en fonction du choix du reseau
    %-----------------------------------------------------------------

    links=(10-fraction_of_edges)*removed_links;
    %links est nombre des liens a predire! Le nombre de liens enleves
    %vaut (10-fraction_of_edges)*removed_links

    %La variable pairs contiendra la liste ordonnee de toutes les
    %aretes fermant au moins un triangle, nous avons en plus besoin de
    %savoir de quelles aretes il s'agit, c'est pourquoi nous A
    %contiendra la matrice d'adjacence modifiee (les elements de la
    %matrice d'adjacene correspondants aux aretes qui ferment au moins
    %un triangle changent de 0 pour aucune arete a 1
    [A,pairs]=trianglePairs(A);


    %-----------------------------------------------------------------
    %             Boucle sur le nombre de liens a predire
    %Les valeurs de TPR et FPR seront enregistrees dans un tableau

    %nombre maximal de predictions a faire: le nombre de paires de
    %noeuds non-connectes dans le reseau de depart plus le nombre de
    %liens qu'on en a enleve
    for l=2:1:(TN+links)
        nbr_predictions=l;
        network=A;
```

```
TPR=0;
FPR=0;

a=0;

%-------------------------------------------------------------
%           Mise a jour de la liste des predictions
if length(pairs(:,1))==nbr_predictions
    %Si le nombre d'aretes qui ferment un triangle est egal au
    %nombre de predictions qu'on veux faire
    list=pairs;

elseif length(pairs(:,1))<nbr_predictions
    %Si moins d'aretes qui ferment des triangles que liens
    %a predire

    if pairs(1,1)==0
        %Si aucune arete ferme un triangle, on revient a la
        %methode aleatoire
        stop=1;
        while stop<=nbr_predictions
            i=ceil(rand*(nbr_nodes));
            j=ceil(rand*(nbr_nodes));
            if(i~=j)&(network(i,j)==0)
                list(stop,1)=i;
                list(stop,2)=j;
                stop=stop+1;
                network(i,j)=1;
                network(j,i)=1;
            end
        end

    else
        %Si moins d'aretes qui ferment au moins un triangle
        %que de predictions a faire

        %Nous mettons les aretes qui ferment au moins un
        %triangle dans la liste des predictions
        for i=1:length(pairs(:,1))
            list(i,1)=pairs(i,1);
            list(i,2)=pairs(i,2);
```

```
                a=a+1;
            end

            %Les autres aretes seront choisies aleatoirement
            %Il est donc important de sortir la matrice d'adjacence
            %modifiee dans triangles_pairs, pour qu'on ne considere
            %plus les aretes qui ferment deja un triangle
            while (length(list)~=nbr_predictions)
                i=ceil(rand*(nbr_nodes));
                j=ceil(rand*(nbr_nodes));
                if(i~=j)&&(network(i,j)==0)
                    list(a+1,1)=i;
                    list(a+1,2)=j;
                    a=a+1;
                    network(i,j)=1;
                    network(j,i)=1;
                end

            end
        end

    else
        %Si le nombre d'aretes qui ferment un triangle est plus
        %grand que le nombre de predictions a faire, nous
        %appliquons la fonction fillup pour choisir les bonnes
        %aretes

        list=fillup(pairs,nbr_predictions);
    end
    %         End Mise a jour de la liste des predictions
    %-----------------------------------------------------------------

    %-----------------------------------------------------------------
    %                   Calcul des true positiv
    TP=0;
    for i=1:nbr_predictions
        a=list(i,1);
        b=list(i,2);
        if data(a,b)==1
            TP=TP+1;
        end
```

```
            end
            %                   Fin Calcul des true positiv
            %-------------------------------------------------------------


            %Calcul des TPR et FPR

            TPR=TPR+TP/(TP+(links-TP));

            FP=nbr_predictions-TP;
            FPR=FPR+(FP)/(FP+(TN-FP));


            array_TPR(l-1)=TPR;
            array_FPR(l-1)=FPR;
        end


        [p1 p2]=polyfit(array_FPR,array_TPR,1);
        %p1 et p2 seront les coefficients de la droite de regression
        x=[0:0.1:1];
        p=p1(1)*x+p1(2);

        %Calcul de l'aire sous la courbe:
        AUC(fichier,fraction_of_edges)=trapz(x,p);
    end
end

%Calcul des moyennes et ecarts-types:
for i=1:9
    AUC(11,i)=mean(AUC(1:10,i));
    AUC(12,i)=std(AUC(1:10,i));
end
```

B.7 Programme calculant les valeurs d'AUC pour la méthode des communautés

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ce programme calcule les valeurs d'AUC pour un reseau pour la methode de
```

```
%prediction basee sur les communautes.
%
%Les communautes trouvees par la methode Louvain doivent se trouver dans le
%meme fichier que ce programme et doivent etre de la forme
%communities_nom-du-reseau_'nombre'_fraction-de-liens-observes_...
%...valeur-parametre.txt
%
%Input : 10 series de 9 listes de noeuds avec leur appartenance aux
%        communautes
%Output: Trois matrices appelees AUC_02, AUC_1, AUC_5 (12x9) en fonction de
%        la valeur du parametre de la methode. Chaque serie de reseaux sera
%        presentee sur une nouvelle ligne, chaque colonne sera une fraction
%        de liens observes.
%        Les lignes 11 et 12 seront composees de la moyenne et de l'ecart
%        type pour chaque colonne.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
clc


%La variable reseau est une variable de choix indiquant le reseau avec
%lequel l'utilisateur souhaite travailler. Possibilites de choix ici:
%   1: Reseau des terroristes grecs
%   2: Reseau karate de Zachary
%   3: Reseau des terroristes des attaques du 09/11
%   4: Reseau des equipes americaines de football
%   5: Reseau des livres politiques
%
%NB: Les reseaux doivent se trouver dans le meme ordre que ce programme, et
%    doivent etre sous formes d'un fichier .txt comprennant la liste
%    des aretes du reseau. En plus, dans la structure switch dans le
%    programme principale, l'utilisateur doit indiquer le nombre de liens
%    qu'il souhaite enlever a chaque etape (ici: 10% par defaut) ainsi que
%    le nombre de noeuds du reseau


reseau=input(strcat('Avec quel reseau voulez-vous travailler? \n',...
    '1 pour le reseau des terroristes grecs \n',...
    '2 pour le reseau karate de Zachary \n',...
```

```
    '3 pour le reseau des terroristes des attaques du 09/11 \n',...
    '4 pour le reseau des equipes americaines de football \n',...
    '5 pour le reseau des livres politiques d''Amazon \n \n'));


switch reseau
    case {1}
        data=importdata('greek_matrix_obs.txt');
    case {2}
        data=importdata('karate_matrix_obs.txt');
    case {3}
        data=importdata('terrorist_matrix_obs.txt');
    case {4}
        data=importdata('polbooks_matrix_obs.txt');
    case {5}
        data=importdata('football_matrix_obs.txt');
end
%data contient la matrice d'adjacence du reseau observe et nous servira
%pour voir si la methode a detecte les aretes retirees


%-------------------------------------------------------------------------
%          Calcul des nombres de paires non-connectees dans data

TN=0;
for i=1:length(data)
    for j=i:length(data)
        if data(i,j)==0 && (i~=j)
            TN=TN+1;
        end
    end
end
%         Fin Calcul des nombres de paires non-connectees dans data
%-------------------------------------------------------------------------


for fichier=1:10
    for fraction_of_edges=1:9
        for param=[0.2 1.0 5.0]
            AUC=0;
```

```
%-------------------------------------------------------------
% Initialisation des variables en fonction du choix du reseau
%
switch reseau
    case {1}
        switch param
            case{0.2}
                community=importdata(strcat(...
                    'communities_greek_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'_0.2'));


            case{1.0}
                community=importdata(strcat(...
                    'communities_greek_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'_1.0'));

            case{5.0}
                community=importdata(strcat(...
                    'communities_greek_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'_5.0'));

        end
        network=importdata(strcat('greek_',num2str(fichier),...
            '_matrix_0',num2str(fraction_of_edges),'.txt'));
        removed_links=6;
        nbr_vertex=22;

    case {2}
        switch param
            case{0.2}
                community=importdata(strcat(...
                    'communities_karate_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'_0.2'));


            case{1.0}
                community=importdata(strcat(...
                    'communities_karate_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'_1.0'));
```

```
                case{5.0}
                    community=importdata(strcat(...
                        'communities_karate_',num2str(fichier),...
                        '_',num2str(fraction_of_edges),'_5.0'));

            end
            network=importdata(strcat('karate_',...
                num2str(fichier),'_matrix_0',...
                num2str(fraction_of_edges),'.txt'));
            removed_links=7;
            nbr_vertex=34;


        case {3}
            switch param
                case{0.2}
                    community=importdata(strcat(...
                        'communities_terrorist_',...
                        num2str(fichier),'_',...
                        num2str(fraction_of_edges),'_0.2'));


                case{1.0}
                    community=importdata(strcat(...
                        'communities_terrorist_',...
                        num2str(fichier),'_',...
                        num2str(fraction_of_edges),'_1.0'));

                case{5.0}
                    community=importdata(strcat(...
                        'communities_terrorist_',...
                        num2str(fichier),'_',...
                        num2str(fraction_of_edges),'_5.0'));

            end
            network=importdata(strcat('terrorist_',...
                num2str(fichier),'_matrix_0',...
                num2str(fraction_of_edges),'.txt'));
            removed_links=15;
            nbr_vertex=62;
```

```
case {4}
    switch param
        case{0.2}
            community=importdata(strcat(...
                'communities_polbooks_',...
                num2str(fichier),'_',...
                num2str(fraction_of_edges),'_0.2'));


        case{1.0}
            community=importdata(strcat(...
                'communities_polbooks_',...
                num2str(fichier),'_',...
                num2str(fraction_of_edges),'_1.0'));

        case{5.0}
            community=importdata(strcat(...
                'communities_polbooks_',...
                num2str(fichier),'_',...
                num2str(fraction_of_edges),'_5.0'));

    end
    network=importdata(strcat('polbooks_',...
        num2str(fichier),'_matrix_0',...
        num2str(fraction_of_edges),'.txt'));
    removed_links=44;
    nbr_vertex=105;


case {5}
    switch param
        case{0.2}
            community=importdata(strcat(...
                'communities_football_',...
                num2str(fichier),'_',...
                num2str(fraction_of_edges),'_0.2'));


        case{1.0}
            community=importdata(strcat(...
```

```
                    'communities_football_',...
                    num2str(fichier),'_',...
                    num2str(fraction_of_edges),'_1.0'));

            case{5.0}
                community=importdata(strcat(...
                    'communities_football_',...
                    num2str(fichier),'_',...
                    num2str(fraction_of_edges),'_5.0'));

        end
        network=importdata(strcat('football_',...
            num2str(fichier),'_matrix_0',...
            num2str(fraction_of_edges),'.txt'));
        removed_links=61;
        nbr_vertex=115;


end

%                End Initialisation des variables
%-------------------------------------------------------------

links=(10-fraction_of_edges)*removed_links;
%links est nombre des liens a predire! Le nombre de liens
%enleves vaut (10-fraction_of_edges)*removed_links

community=community+1;
%Ce pas est necessaire puisque l'algorithme de detection des
%communautes commence la numerotation des noeuds a 0

community=sortrows(community,2);
x=1;
v=1;
A=network;

%-------------------------------------------------------------
%              Mise a jour de la liste des aretes
for i=1:length(community(:,1))

    act=community(i,2);
```

```
        a=community(i,1);

        %Nous envisageons toutes les paires de noeuds qui se
        %trouvent dans la meme communaute. S'ils ne sont pas lies
        %dans le graphe observe, nous les ajoutons a la liste des
        %paires de noueds.

        if i<length(community(:,1))

            for j=i:length(community(:,1))
                if community(j,2)==act
                    %Tant que le noeud se trouve dans la meme
                    %communaute
                    b=community(j,1);
                    if A(a,b)==0 && a~=b
                        list(v,1)=a;
                        list(v,2)=b;
                        v=v+1;
                        A(a,b)=1;
                        A(b,a)=1;
                    end
                end
            end
        end
    end
    network_int=A;


    %-------------------------------------------------------------
    %             Boucle sur le nombre de liens a predire
    %Les valeurs de TPR et FPR seront enregistrees dans un tableau

    %nombre maximal de predictions a faire: le nombre de paires de
    %noeuds non-connectes dans le reseau de depart plus le nombre
    %de liens qu'on en a enleve
    for l=2:1:(TN+links)

        nbr_predictions=l;
        B=network_int;

        pairs=communities_pairs(B,list,nbr_predictions);
```

```
    %communities_pairs nous fournira une liste de
    %nbr_predictions d'aretes reliant des noeuds d'une meme
    %communaute

    TPR=0;
    FPR=0;
    TP=0;

    %-----------------------------------------------------------
    %                    Calcul des true positiv

    for i=1:nbr_predictions
        a=pairs(i,1);
        b=pairs(i,2);
        if data(a,b)==1
            TP=TP+1;
        end
    end

    %                   Fin Calcul des true positiv
    %-----------------------------------------------------------

    %Calcul des TPR et FPR:

    TPR=TP/(TP+(links-TP));

    FP=nbr_predictions-TP;
    FPR=(FP)/(FP+(TN-FP));


    array_TPR(l-1)=TPR;
    array_FPR(l-1)=FPR;
end

[p1 p2]=polyfit(array_FPR,array_TPR,1);
%p1 et p2 seront les coefficients de la droite de regression
x=[0:0.1:1];
p=p1(1)*x+p1(2);

%Calcul de l'aire sous la courbe:
AUC=AUC+trapz(x,p);
```

```
            %end

            if param==0.2
                AUC_02(fichier,fraction_of_edges)=AUC;
            elseif param==1.0
                AUC_1(fichier,fraction_of_edges)=AUC;
            elseif param==5.0
                AUC_5(fichier,fraction_of_edges)=AUC;
            end
            AUC=0;


        end

    end
end

%Calcul des moyennes et ecarts-types:
for i=1:9
    AUC_02(11,i)=mean(AUC_02(1:10,i));
    AUC_02(12,i)=std(AUC_02(1:10,i));


    AUC_5(11,i)=mean(AUC_5(1:10,i));
    AUC_5(12,i)=std(AUC_5(1:10,i));

    AUC_1(11,i)=mean(AUC_1(1:10,i));
    AUC_1(12,i)=std(AUC_1(1:10,i));

end
```

B.8 Programme calculant les valeurs d'AUC pour la méthode hiérarchique

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Ce programme calcule les valeurs d'AUC pour un reseau pour la methode de
%prediction basee sur la hierarchie.
%
```

```
%Les rankings calcules par la methode hierarchique de Clauset doivent se
%trouver dans le meme fichier que ce programme et doivent etre de la forme
%nom-du-reseau_'nombre'_fraction-de-liens-observes_-ranked.wpairs
%
%Input : 10 series de 9 listes d'aretes calculees par la methode
%        hierarchqiue de Clauset
%Output: Une matrice appelee AUC (12x9). Chaque serie de reseaux sera
%        presentee sur une nouvelle ligne, chaque colonne sera une fraction
%        de liens observes.
%        Les lignes 11 et 12 seront composees de la moyenne et de l'ecart
%        type pour chaque colonne.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



clear all
close all
clc


%La variable reseau est une variable de choix indiquant le reseau avec
%lequel l'utilisateur souhaite travailler. Possibilites de choix ici:
%   1: Reseau des terroristes grecs
%   2: Reseau karate de Zachary
%   3: Reseau des terroristes des attaques du 09/11
%   4: Reseau des equipes americaines de football
%   5: Reseau des livres politiques
%
%NB: Les reseaux doivent se trouver dans le meme ordre que ce programme, et
%    doivent etre sous formes d'un fichier .txt comprennant la liste
%    des aretes du reseau. En plus, dans la structure switch dans le
%    programme principale, l'utilisateur doit indiquer le nombre de liens
%    qu'il souhaite enlever a chaque etape (ici: 10% par defaut) ainsi que
%    le nombre de noeuds du reseau


reseau=input(strcat('Avec quel reseau voulez-vous travailler? \n',...
    '1 pour le reseau des terroristes grecs \n',...
    '2 pour le reseau karate de Zachary \n',...
    '3 pour le reseau des terroristes des attaques du 09/11 \n',...
    '4 pour le reseau des equipes americaines de football \n',...
```

```
    '5 pour le reseau des livres politiques d''Amazon \n \n'));


switch reseau
    case {1}
        data=importdata('greek_matrix_obs.txt');
    case {2}
        data=importdata('karate_matrix_obs.txt');
    case {3}
        data=importdata('terrorist_matrix_obs.txt');
    case {4}
        data=importdata('polbooks_matrix_obs.txt');
    case {5}
        data=importdata('football_matrix_obs.txt');
end
%data contient la matrice d'adjacence du reseau observe et nous servira
%pour voir si la methode a detecte les aretes retirees


%-------------------------------------------------------------------------
%          Calcul des nombres de paires non-connectees dans data

TN=0;
for i=1:length(data)
    for j=i:length(data)
        if data(i,j)==0 && (i~=j)
            TN=TN+1;
        end
    end
end
%         Fin Calcul des nombres de paires non-connectees dans data
%-------------------------------------------------------------------------



for fichier=1:10
    for fraction_of_edges=1:9

        %-----------------------------------------------------------------
        %   Initialisation des variables en fonction du choix du reseau
        %
```

```
        switch reseau
            case {1}
                list=importdata(strcat('greek_',num2str(fichier),'_',...
                    num2str(fraction_of_edges),'-ranked.wpairs'));

                removed_links=6;
                nbr_vertex=22;


            case {2}

                list=importdata(strcat('karate_',num2str(fichier),'_',...
                    num2str(fraction_of_edges),'-ranked.wpairs'));
                A=importdata(strcat('karate_',num2str(fichier),...
                    '_matrix_0',num2str(fraction_of_edges),'.txt'));

                removed_links=7;
                nbr_vertex=34;


            case {3}

                list=importdata(strcat('terrorist_',num2str(fichier),...
                    '_',num2str(fraction_of_edges),'-ranked.wpairs'));

                removed_links=15;
                nbr_vertex=62;


            case {4}

                list=importdata(strcat('polbooks_',num2str(fichier),'_',...
                    num2str(fraction_of_edges),'-ranked.wpairs'));

                removed_links=44;
                nbr_vertex=105;


            case {5}

                list=importdata(strcat('football_',num2str(fichier),'_',...
```

```
            num2str(fraction_of_edges),'-ranked.wpairs'));

        removed_links=61;
        nbr_vertex=115;


end

% End Initialisation des variables en fonction du choix du reseau
%-----------------------------------------------------------------

links=(10-fraction_of_edges)*removed_links;
%links est nombre des liens a predire! Le nombre de liens enleves
%vaut (10-fraction_of_edges)*removed_links


%-----------------------------------------------------------------
%             Boucle sur le nombre de liens a predire
%Les valeurs de TPR et FPR seront enregistrees dans un tableau

%nombre maximal de predictions a faire: le nombre de paires de
%noeuds non-connectes dans le reseau de depart plus le nombre de
%liens qu'on en a enleve
for l=2:1:(TN+links)
    nbr_predictions=l;
    pairs=list(1:l,1:2);

    %-------------------------------------------------------------
    %                   Calcul des true positiv
    TP=0;
    for i=1:nbr_predictions
        a=pairs(i,1);
        b=pairs(i,2);
        if data(a,b)==1
            TP=TP+1;
        end
    end

    %                  Fin Calcul des true positiv
    %-------------------------------------------------------------
```

```
            %Calcul des TPR et FPR:

            TPR=TP/(TP+(links-TP));

            FP=nbr_predictions-TP;
            FPR=(FP)/(FP+(TN-FP));


            array_TPR(l-1)=TPR;
            array_FPR(l-1)=FPR;
        end

        [p1 p2]=polyfit(array_FPR,array_TPR,1);
        %p1 et p2 seront les coefficients de la droite de regression
        x=[0:0.1:1];
        p=p1(1)*x+p1(2);

        %Calcul de l'aire sous la courbe:
        AUC(fichier,fraction_of_edges)=trapz(x,p);
    end
end

%Calcul des moyennes et ecarts-types:
for i=1:9
    AUC(11,i)=mean(AUC(1:10,i));
    AUC(12,i)=std(AUC(1:10,i));
end
```

B.9 Fonction qui sélectionne par une règle particulière un certain nombre d'arêtes d'un ensemble

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cette fonction choisit nbrPredictions differentes paires de noeuds de la
%liste list. Elle considere differentes cas afin de respecter la generalite
%dans les calculs des valeurs de l'AUC.
%Si nous souhaitons par exemple choisir 7 paires de noeuds parmi une liste
%de 20 paires avec un certain score par paire (prenons a titre d'exemple le
%nombre de triangles qu'ils ferment). Supposons que les 2 premieres paires
```

```
%de la liste ferment 5 triangles, les 3 suivantes en ferment 4, les 8
%paires suivantes en ferment 3 et les 7 paires restantes en ferment 2.
%Alors cette fonction ajoutera a la liste des predictions les 5 aretes
%fermant 5 et 4 triangles. Ils en restent des lors 2 a choisir parmi les 8
%aretes qui ferment 2 triangles. Cette fonction choisira les 2 aretes
%parmi les 8 aretes restantes aleatoirement. Nous definirons dans le
%programme un score limite qui sera le premier score pour lequel la somme
%des nombres d'aretes de scores plus grands depasse le nombre de liens a
%predire (les aretes du score limite inclues)
%
%Input : list           : La liste des aretes avec un score par arete
%        nbrPredcictions: Le nombre de predictions a faire, c'est-a-dire le
%                         nombre de paires a choisir parmi celles dans list
%Output: result         : La liste des nbrPredcictions paires d'aretes
%                         choisies en fonction de leur score
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[result]=fillup(list,nbrPredictions)

list=sortrows(list,-3);
%La liste sera ordonnee de maniere decroissante en fonction du score des
%aretes

%Nous comptons d'abord le nombre de differentes aretes par valeur de score
x=0;
int=list(1,3);
%La valeur de int sera le score considere, donc le score de la premiere
%paire au debut

j=1;
%La variable j compte le nombre de differents scores presents dans la
%liste

result=[0 0 0];
%Par defaut, result ne contient rien

%-------------------------------------------------------------------------
%                  Comptage des nombres d'aretes par score
%On parcourt toute la liste et on compte les nombres d'aretes par
%valeur de score.
for i=1:length(list(:,1))
```

```
    if list(i,3)==int && i<length(list(:,1))
        x=x+1;
        %Tant que le score ne change pas, nous augmentons la variable
        %contant le nombre de fois que le score apparait d'une unite

    else
        %Si la valeur du score considere change, nous mettons a jour les
        %variables
        if i==length(list(:,1))
            %Si nous avons atteint la fin de la liste
            x=x+1;
            nbrTriangles(j)=x;
        else
            nbrTriangles(j)=x;
            j=j+1;
            x=1;
            int=list(i,3);
        end
    end
end
%               Fin Comptage des nombres d'aretes par score
%-------------------------------------------------------------------------


%Dans la suite, nous calculons le score limite. Pour notre exemple
%dans la description de ce programme, le score limite sera 3, puisque les 2
%aretes qui ferment 5 triangles plus les 3 qui en ferment 4 plus les 8 qui
%en ferment 3 depassent en terme de nombre le nombre de liens qu'on
%souhaite predire. Le score 3 est donc le premier score pour lequel la
%somme des aretes fermant PLUS de triangles depasse le nombre de
%predictions (la somme des aretes du score limite inclues)

i=1;
indice=nbrTriangles(i);
%nbrTriangles(1) est le nombre d'aretes du meilleur score
%nbrTriangles(2) est le nombre d'aretes du deuxieme meilleur score etc

while indice<nbrPredcictions
    indice=indice+nbrTriangles(i+1);
    i=i+1;
end
```

```
%indice est l'indice de la derniere arete du score limite

%-------------------------------------------------------------------------
%       Partie principale: on remplit result par des paires de list

listInt=list(1:indice,:);
%La variable intermediare listInt sera la liste de toutes les aretes
%admettant au moins la valeur du score limite (inclues). Nous coupons des
%lors la liste complete a la bonne place.

if indice == nbrPredictions
    %Si le nombre d'aretes dans listInt correspond au nombre d'aretes
    %qu'on souhaite predire
    result=listInt;
else
    %Sinon, le nombre de liens a predire est plus petit que la longueur de
    %la listInt

    coupure=length(listInt(:,1));
    %Nous cherchons les aretes de listInt que nous mettons dans result,
    %et apres, nous choisirons les aretes restantes dans listInt par
    %hasard.
    int=listInt(coupure,3);
    i=1;
    while listInt(i,3)==int && i<length(listInt(:,1))
        %Quand le score change, nous avons trouve la bonne place a couper
        %la liste listInt
        coupure=coupure-1;
        i=i+1;
    end

    if coupure==1
        %Dans ce cas: Toutes les aretes dans listInt auront le meme score
        %Nous choisirons des lors aleatoirement nbrPredcictions parmi ces
        %aretes

        i=1;
        while length(result(:,1))<nbrPredcictions
            a=ceil(rand*length(listInt(:,1)));
            if listInt(a,3)~=-1
                %Si l'arete n'a pas encore ete considere
```

```
                result(i,:)=listInt(a,:);
                i=i+1;
                listInt(a,3)=-1;
            end
        end
    else
        %Sinon nous mettons les aretes de 1 jusqu'a coupure dans une
        %deuxieme liste intermediare, les autres seront choisies
        %aleatoirement parmi les aretes restantes dans listInt
        listInt2=listInt(1:coupure,:);

        j=1;
        act=nbrTriangles(j);
        i=1;

        %Nous mettons les aretes du score strictement plus grand que le
        %score limite dans result:
        while act<nbrPredcictions
            result(i:act,:)=listInt2(i:act,:);
            j=j+1;
            i=act+1;
            act=act+nbrTriangles(j);
        end


        i=length(result(:,1));
        listInt3=listInt2((i+1):length(listInt2(:,1)),:);
        %Dans la troisieme liste intermediaire, nous mettons toutes les
        %aretes restantes, parmi lesquelles nous choisirons aleatoirement
        %quelques aretes

        %Choix aleatoire d'aretes parmi les aretes restantes
        while length(result(:,1))<nbrPredcictions
            random=ceil(rand*(length(listInt3(:,1))));
            if listInt3(random,3)~=-1
                result(i+1,:)=listInt3(random,:);
                i=i+1;
                listInt3(random,3)=-1;
            end
        end
    end
```

```
end
%                              End Partie principale
%-------------------------------------------------------------------------
end
```

Annexe C

Divers

C.1 Acteurs du réseau terroriste grec $17N$

label	Nom du terroriste
1	Yiannis Skandalis
2	Alexandros Giotopoulos
3	Anna'
4	Christodoulos Xiros
5	Constantinos Karatsolis
6	Constantinos Telios
7	Dimitris Koufontinas
8	Dionysis Georgiadis
9	Elias Gaglias
10	Iraklis Kostaris
11	Nikos Papanastasiou
12	Ojurk Hanuz
13	Patroclos Tselentis
14	Pavlos Serifis
15	'Sardanapalos'
16	Savas Xiros
17	Sotirios Kondylis
18	Theologos Psaradelis
19	Thomas Serifis
20	Vassilis Tzortzatos
21	Vassilis Xiros
22	Yiannis Serifis

C.2 Réseau du championnat de football - Liste exhaustive des arêtes du réseau artificiel

Nœud 1	Nœud 2	Produit des degrés
2	7	25
2	8	25
2	9	25
4	7	25
4	8	25
4	9	25
5	7	25
5	8	25
1	8	20
1	9	20
2	10	20
3	7	20
3	9	20
4	6	20
5	6	20
5	10	20
1	3	16
1	6	16
1	10	16
3	6	16
3	10	16
6	10	16

Bibliographie

[1] P. Antal, T. Krapivsky and S. Redner. Social balance on networks : The dynamics of friendship and enmity. *Physica D*, 224, 2006.

[2] Brian Ball and M. E. J. Newman. Friendship networks and social status. *CoRR*, abs/1205.6822, 2012.

[3] C. Bergstrom. Lecture 7 : Introduction to markov chains. *University of Wahsington.*, (Notes du cours Biology 497), 2006.

[4] C. Bergstrom. Lecture 8 : Analyzing markov chains. *University of Wahsington.*, (Notes du cours Biology 497), 2006.

[5] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 10, 2008.

[6] S. J. Brams, H. Mutlu, and S.L. Ramirez. Influence in terrorist networks : From undirected to directed graphs. *Studies in Conflict and Terrorism*, (29) :703—-718, 2006.

[7] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4) :327–335, 1995.

[8] A. Clauset, C. Moore, and M. E. J. Newman. Structural inference of hierarchies in networks. *Proceedings of the 23rd International Conference on Machine Learning*, (Pittsburgh, PA), 2006.

[9] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453 :98–101, 2008.

[10] C. Cortes and M. Mohri. Auc optimization vs. error rate minimization. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

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

[12] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc Natl Acad Sci U S A*, 99(12) :7821–7826, June 2002.

[13] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counter-terrorism and Security*, 2006.

[14] S. D. Johnson. A brief history of the analysis of crime concentration. *Euro. Jnl of Applied Mathematics*, (21) :349–370, 2010.

[15] N. Kashtan and U. Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences*, 102(39) :13773–13778, September 27 2005.

[16] V. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3) :43–52, 2002.

[17] C. K. Kwoh and P.Y. P.Y. Network analysis approach for biology. *Cell. Mol. Life Sci.*, (64) :1739–1751, 2007.

[18] R. Lambiotte. Multi-scale modularity and dynamics in complex networks. *WiOpt*, pages 546–553, 2010.

[19] R. Lambiotte and P. Panzarasa. Communities, knowledge creation and information diffusion. *Journal of Informetrics*, 3 :180–190, 2009.

[20] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7) :1019–1031, 2007.

[21] M. Mitzenmacher. A brief history of lognormal and power law distributions. *Proceedings of the Allerton Conference on Comunication, Control, and Computing*, pages 182–191, 2001.

[22] J. P. Nadal, M. B. Gordon, J. R. Iglesias, and V. Semeshenko. Modelling the individual and collective dynamics of the propensity to offend. *Euro. Jnl of Applied Mathematics*, (21) :421–440, 2010.

[23] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2), 2001.

[24] M. E. J. Newman. *Networks : An Introduction*. Oxford University Press, 2010.

[25] M. E. J. Newman and E. A. Leicht. Mixture models and exploratory analysis in networks. *physics/0611158*, 2006.

[26] J. Park and M. E. J. Newman. A network-based ranking system for american college football. *Journal of Statistical Mechanics*, (P10014) :1739—1751, 2005.

[27] A.B. Pitcher. Adding police to a mathematical model of burglary. *Euro. Jnl of Applied Mathematics*, (21) :401–419, 2010.

[28] C. J. Rhodes. *The Use of Open Source Intelligence in the Construction of Covert Social Networks*, volume 2 of *Lecture Notes in Social Networks*. Springer Vienna, 2011.

[29] H. Simon. The architecture of complexity. *Proc. of the American Philosophical Society*, 106(6) :467–482, December 1962.

[30] S. Soundarajan and J. Hopcroft. Using community information to improve the precision of link prediction methods. *Proceedings of the 21st international conference companion on Word Wide Web*, pages 607–608, 2012.

[31] C. Stohl and M. Stohl. Networks of terror : Theoretical assumptions and pragmatic consequences. *Communication Theory*, (17) :93–124, 2007.

[32] A. Tseloni, I. Ntzoufras, A. Nicolaou, and K. Pease. Concentration of personal and household crimes in england and wales. *Euro. Jnl of Applied Mathematics*, (21) :325–348, 2010.

[33] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33 :452–473, 1977.