Algorithmes de graphes pour la découverte de la topologie d'un réseau énergétique par la connaissance de ses flots

Joseph Ehounou

université PARIS-SACLAY

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
université PARIS-SACLAY

NNT : 2018SACLV056

Thèse de doctorat

Algorithmes de graphes pour la découverte de la topologie d'un réseau énergétique par la connaissance de ses flots

Thèse de doctorat de l'Université Paris-Saclay
préparée à Université de Versailles-Saint-Quentin-en-Yvelines

Ecole doctorale n°27 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Programmation : Modèles, Algorithmes, Langages, Architecture

Thèse présentée et soutenue à Versailles, le 02/10/2018, par

WILFRIED JOSEPH EHOUNOU

Composition du Jury :

Danièle GARDY Professeur, Université de Versailles (DAVID)	Présidente
Pascal BARTHOMÉ Professeur, INSA centre Val de Loire (LIFO)	Rapporteur
Christian LAFOREST Profésseur, Université Clermont Auvergne (LIMOS-ISIMA)	Rapporteur
Sandrine VIAL Maitre de conférences, Université de Versailles (DAVID)	Examinatrice
Dominique BARTH Professeur, Université de Versailles (DAVID)	Directeur de thèse
Arnaud DE MOISSAC Ingénieur, DCBRAIN	Encadrant

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Algorithme de graphes pour la découverte de la topologie d'un réseau énergétique par la connaissance de ses flots

Mots clés: graphes, découverte de topologie, flots, line-graphes, détection de changements, séries temporelles.

Résumé: Dans les réseaux énergétiques, la connaissance des équipements, leurs emplacements et leurs fonctions sont les prérequis à l'exploitation de l'infrastucture. En effet, tout opérateur dispose d'une carte appelée *schéma synoptique* indiquant les connexions entres les équipements. À partir de cette carte, sont prises des décisions pour un fonctionnement optimal du réseau. Ce schéma synoptique peut être erroné parce que des opérations de maintenance sur le réseau n'auraient pas été retranscrites ou mal saisies. Et cela peut entrainer des coûts supplémentaires d'exploitation du réseau énergétique.
Nous considérons le réseau électrique d'un data center. Ce réseau est composé d'une topologie physique modélisée par un *DAG sans circuit* et de mesures électriques sur ces arcs. La particularité de ce réseau est que les mesures contiennent des erreurs et la topologie est inconnue c'est-à-dire les arcs sont connus et les extrémités de ces arcs sont inconnues. Dans le cas où ces mesures sont correctes alors la corrélation des arcs induit la matrice d'adjacence du *line-graphe* de notre *DAG*. Un line-graphe est un graphe dans lequel chaque sommet et son voisinage peuvent être partitionnés par une ou deux cliques et chaque arête est couverte par une clique. Cependant, avec la présence des erreurs de mesures, nous avons un graphe avec des arêtes en plus ou en moins qui n'est pas nécessairement un line-graphe. Si ce graphe est un line-graphe alors il n'est pas le line-graphe de notre DAG. Notre problème est de découvrir cette topologie en se basant sur ces mesures électriques.
Nous débutons par une étude bibliographique des corrélations de mesures possibles afin de déterminer celle qui est pertinente pour notre problème. Ensuite nous proposons deux algorithmes pour résoudre ce problème. Le premier algorithme est *l'algorithme de couverture* et il détermine l'ensemble des cliques qui couvre chaque sommet de notre graphe. Le second algorithme est *l'algorithme de correction*. Il ajoute et supprime des arêtes au voisinage d'un sommet non couvert de telle sorte que son voisinage soit partitionné en une ou deux cliques. Enfin, nous évaluons les performances de nos algorithmes en vérifiant le nombre d'arêtes corrigées et la capacité à retourner le graphe le plus proche du line-graphe de notre DAG.

Title: Algorithm of graphs for topology discovery for a energy network from flot knowledges

Keywords: graphs, topology discovery, flow, linegraphs, change detection, time series.

Abstract: In energy network, the knowledge of equipments, their locations and their functions are the important informations for the distributor service operator. In fact, each operator has a network plan often named *synoptic schema*. That schema shows the interconnexion between equipments in the network. From this schema, some management decisions have taken for ensuring an optimal performance of a network. Sometimes, a synoptic schema has some mistakes because the maintenance operations, such as changed the connexion between equipments or replaced equipments, have not been updated or have been written with errors. And these mistakes increase exploitation cost in the energy network.
We consider an electric network of a datacenter. This network consists of physical topology modelised by a DAG without circuit and measurements are on the edges of a DAG. The main point of the network is that measurements are some mistakes and the topology is unknown i.e we know edges but the nodes of edges are unknown. When measurements are correct then the correlations between pairwise edges provide the adjacency matrix of the linegraph of undirected graph of the DAG. A linegraph is a graph in which each node and the neighbor are partitionned by one or deux cliques. However, with the mistakes in measurements, the obtained graph is not a linegraph because it contains more or less edges. If the obtained graph is a linegraph then it is a linegraph of the other DAG. Our problem is to discovery the topology of the DAG with some mistakes in measurements.
We start by the state of art in the measurement correlations in order to choose the good method for our problem. Then, we propose two algorithms to resolve our problem. The first algorithm is the cover algorithm and it returns the set of cliques in the graph. The second algorithm is a correction algorithm which adds or deletes edges in the graph for getting a nearest linegraph of the DAG. In the last, we evaluate the performances of the algorithms by checking the number of edges corrected and the ability to return a nearest linegraph of the DAG.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Remerciements

Par ces mots s'achève la rédaction de ce manuscrit. Je voudrais apporter ma reconnaissance à toutes ces personnes qui m'ont accompagnées durant cette période en particulier à ma famille, aux collègues de DCbrain SAS et aux membres de l'équipe DAVID.

Un grand merci à ma famille pour leur immense soutien. Ils ont été patients, compréhensifs et encourageants. Je pense à mon père pour ses encouragements, ma mère pour son reconfort, ma tante Félicité pour ses conseils et son hospitalité, mes frères et soeurs Grace, Prisca, Serge, Maruis, Ismael et Erven.
La rencontre de ces personnes a été décisive dans mes choix de vie. Mario de son Petionville, merci pour tes conseils éclairés de la vie. Dimitri l'électrochoc de mon quotidien, merci pour ta vision optimiste. Alice merci pour les relectures, tu m'as été d'une grande aide. Cher Henri-Joel, l'ami qui est devenu un frère, merci d'avoir cru j'y suis arrivé. Je t'écouterai plus souvent. Adrienne merci pour ses prières et Cécile pour son amour.

De Earthgrid à DCbrain SAS, l'aventure a débuté par un stage. De 3 personnes au début, elle est devenue une startup de dimension nationale sous la direction des bretons Arnaud et François. Arnaud, je retiens de toi ton esprit de management et aussi ta manière de simplifier un problème. François, je retiens de toi ton coup de main sur les graphes et ses astuces de programmation. À Maxime, merci pour son assistance et ses conseils en python pour l'implémentation des algorithmes. Et Damien pour son aide et ses explications sur les séries temporelles.

Je voudrai particulièrement exprimer ma gratitude à mon encadreur Dominique. Ses décisions, ses recommendations et sa disponibilité sont le fruit de ce manuscrit. Vous n'êtes pas un professeur pour rien.

À TOUS CEUX QUE JE N'AI PAS CITÉS. Le silence n'est pas synonyme d'oubli; mes pensées vont vers vous ;

L'enseignement que je retiens de ce travail :
Tout est possible à condition de faire le premier pas et de ne pas baisser les bras.

Table des Matières

Liste des Algorithmes

Liste des figures

Liste des Tableaux

Chapitre 1

Introduction Générale

Un réseau de distribution énergétique est un ensemble d'équipements énergétiques interconnectés permettant d'acheminer l'énergie des centres de production aux consommateurs. Le réseau est composé d'une infrastructure de transport et d'une infrastructure de distribution de l'énergie. Ces infrastructures sont spécifiques au type d'énergie et sont gérées chacune par un gestionnaire de réseau ou Distribution Source Operator. Son rôle est de maintenir la continuité et la qualité de l'approvisionnement physique des clients c'est-à-dire assurer l'entretien et le développement du réseau. En effet, le gestionnaire de transport achemine, au niveau national, l'énergie depuis son lieu de production à son lieu de consommation à travers le réseau de distribution. Quant au gestionnaire de distribution, il distribue l'énergie aux consommateurs au niveau régional.

Un exemple de réseau énergétique géré localement par un gestionnaire de réseau est un centre de données ou data center. Le data center a la particularité de contenir un réseau thermique, électrique et informatique. Le réseau électrique alimente les équipements des autres réseaux. La connaissance de son infrastructure est indispensable pour le gestionnaire afin de fournir la puissance nécessaire au fonctionnement d'un équipement et d'éviter les surcoûts de production d'électricité parce que l'électricité ne se stocke pas. Pour ce faire, des sondes sont installées dans le réseau électrique afin de collecter les consommations des équipements dans le temps. Les mesures temporelles des consommations sont intégrées dans un outil de supervision de ce réseau. Cet outil permet d'avoir l'état de tous les équipements à un instant donné, de connaître la topologie de fonctionnement du réseau et aussi la topologie générale du réseau c'est-à-dire l'ensemble des équipements arrêtés et en fonctionnement.

Malheureusement, l'état du réseau affiché par l'outil de supervision est souvent erroné. En effet, les sondes de certains équipements présentes dans le réseau physique ne sont pas répertoriées dans la

base de données de l'outil et leurs mesures ne figurent nulle part dans l'outil de supervision. Cela donne l'impression que ces équipements sont en panne et les équipements rattachés à ces derniers sont alimentés par d'autres équipements qui fonctionnent en surcapacité. En outre, les sondes de certains équipements sont interchangées avec d'autres. Les puissances sont remplacées par des intensités et aussi la puissance d'un équipement est remplacée par celle d'un autre équipement. Ces erreurs humaines impliquent le non respect de la loi de conservation des noeuds et aussi une maintenance complexe parce que nous ignorons où sont situées les sondes interchangées. Ces problèmes proviennent d'une faible communication entre les différents métiers dans le data center et aussi de la mise à jour erronée ou tardive des maintenances dans la base de données de l'outil. Ceux-ci entrainent des prévisions élevées des consommations électriques, des coûts de maintenances exorbitants et enfin la topologie connue n'est pas la topologie réelle.

Afin de réduire les erreurs humaines dans la découverte du réseau, nous proposons des méthodes pour déterminer la topologie du réseau électrique en se basant uniquement sur les mesures temporelles collectées. Des travaux similaires de découverte de topologie ont été réalisés par l'entreprise *DCbrain SAS*. En effet, elle a reconstruit la topologie du réseau électrique de la ville de Paris du gestionnaire *Enedis* à partir des incidents matériels dans ce réseau. Elle a supposé qu'un équipement défectueux impacte le fonctionnement les autres équipements qui lui sont rattachés. Cela implique que l'incident se propage dans le réseau à travers les équipements interconnectés.
Nous allons considérer, dans notre cas, que

- Toutes les mesures sont connues malgré des valeurs absentes dans certaines séries de mesures et aussi les équipements rattachés à ces mesures sont incorrects.
- Un équipement ne s'alimente pas lui-même mais peut alimenter plusieurs autres équipements.
- Le réseau électrique fonctionne en monophasé et en triphasé.
- Le réseau électrique est alimenté à un seul gestionnaire de réseau mais contient plusieurs sources d'énergie (les groupes électrogènes, les onduleurs).
- Les pertes par Effet Joule au cours du transport de l'électricité sont négligeables.

Nous débutons dans le chapitre 1 par la modélisation du réseau électrique comme un réseau de flots dans lequel les flots sont les mesures électriques et le réseau est un DAG sans circuit. Dans le DAG, les sommets sont des équipements et les arcs sont des câbles. Les mesures sont décrites par des séries temporelles et une nouvelle loi de conservation [2] est proposée en adéquation avec les lois physiques.

Dans le chapitre 2, nous présentons les analyses effectuées avec les séries temporelles puis nous indiquons que la comparaison de séries temporelles est adaptée à notre problème. Ensuite nous présentons l'ensemble de méthodes de comparaison de séries temporelles et nous retenons la distance de Pearson comme la méthode de calcul des coefficients de similarité entre les mesures. Enfin nous commentons les résultats obtenus avec cette méthode.

Dans le chapitre 3, les coefficients de similarité forment une matrice dite *matrice de corrélation* et le graphe associé à cette matrice est dit *graphe de corrélation*. Cette matrice de corrélation est la matrice d'adjacence du line-graphe du graphe non orienté sous-jacent au DAG du réseau électrique à condition que cette matrice ne contienne aucune case erronée. Nous montrons que le line-graphe admet un partitionnement unique en cliques à l'exception de situations d'ambiguïtés décrites dans ce chapitre. Ce partitionnement est appelé la *couverture de corrélation*. Ensuite, nous présentons deux algorithmes (couverture et correction) qui proposent le partitionnement du graphe de corrélation s'il n'y a aucune case erronée, sinon qui retourne le line-graphe le plus proche du graphe de corrélation en cas de cases erronées. Enfin, nous décrivons la construction du graphe non orienté sous-jacent au DAG du réseau électrique puis nous orientons les arêtes de ce graphe.

Dans le chapitre 4, nous évaluons les performances de nos algorithmes sur des graphes générés à partir de trois expérimentations. La première expérimentation consiste à modifier des cases dans le graphe de corrélation. La seconde expérimentation consiste à attribuer les coefficients de corrélation entre les arcs en se basant sur la distribution des valeurs des cases de la matrice du graphe de corrélation. La dernière expérimentation se réalise sur des graphes ayant plus de deux couvertures de corrélation. Nous évaluons le nombre de cases corrigées après l'exécution de nos algorithmes.

Chapitre 2

Réseaux de flots et mesures physiques

Les réseaux énergétiques ont pour rôle de fournir une énergie à des entités consommatrices sans interruption de service. Ces réseaux fonctionnent en courant continu et en courant alternatif.
Notre étude est limitée au courant alternatif parce que la demande d'électricité des équipements varie constamment et la quantité d'électricité est connue pour chaque équipement de ce réseau. Notre objectif est de regrouper les équipements qui ont la même source d'alimentation. Ces sources d'alimentation subissent régulièrement des maintenances et le schéma électrique n'est pas mis à jour, ce qui entraine des problèmes dans le système de surveillance de ce réseau et aussi dans la planification de nouvelles maintenances.
Dans ce réseau électrique, nous connaissons les liens et les mesures présentes sur ces liens mais nous ignorons les extrémités de ces liens. Notre problème est d'identifier ces extrémités à partir des mesures électriques. Nous faisons de la *découverte de topologie*. Le modèle proposé a été établi notamment à partir de la description du réseau électrique d'un data center d'un opérateur téléphonique appelé *Champlan*.
Notre chapitre comprend quatre parties. La première partie montre la relation entre un réseau électrique et un graphe de flots. La seconde partie modélise le réseau de flots et la troisième partie présente les travaux existants sur la découverte de topologie. La dernière partie présente l'approche retenue pour résoudre notre problème.

2.1 Réseau électrique d'un data center : un graphe de flots

2.1.1 Topologie du réseau électrique du data center

Un data center ou centre de données est un site physique regroupant des installations informatiques interconnectées (serveurs, réseau informatique, système de sauvegarde) et une infrastructure énergétique adéquate (un système de distribution électrique, un commutateur électrique, des réserves d'énergie, des générateurs dédiés à la sauvegarde de données, un système de ventilation et de refroidissement).
Le réseau électrique est le coeur de data center car il alimente à la fois les infrastructures informatique et énergétique. Il se compose de quatres entités :

- **Les sources** : ces équipements sont les points d'entrée de l'électricité dans le data center et sont directement rattachés au gestionnaire de réseau régional ou national. Ils sont de deux types :
  - Ceux qui alimentent le réseau en cas de dysfonctionnement du gestionnaire de réseau. Ce sont les accumulateurs, les groupes électrogènes.
  - Ceux qui transforment l'électricité reçue du gestionnaire pour des puissances utilisables dans le data center. Ce sont les transformateurs basse tension communement appelés *Transformateur Général Basse Tension (TGBT).*

  Généralement, ces deux types d'équipements ne fonctionnent pas concomitamment.

- **Les tableaux** : aussi appelés tableaux de répartition, ce sont des équipements passifs dont la fonction est celle de communateur. Ils représentent l'organe central de l'installation dans la mesure où ils regroupent tous les circuits électriques et systèmes de protection vers les baies de serveurs. Une baie est une armoire contenant plusieurs serveurs. Chaque baie possède un disjoncteur sur ce tableau afin d'interrompre l'alimentation en cas de danger. Les tableaux sont considérés comme des équipements passifs car l'électricité qui traverse ces équipements a une perte négligeable. Ces pertes sont les *pertes par Effet Joule.* Un exemple de tableau est présenté dans la figure à gauche du tableau 2.1.

- **Les baies (ou racks) de serveurs** : les baies distribuent la puissance nécessaire au fonctionnement de chaque serveur qui lui est rattaché. La baie a un rôle de multiprise pour tous les serveurs. Dans le système de supervision électrique, les baies sont les consommateurs de l'électricité. Elles sont des équipements actifs dans le réseau électrique. La figure à droite du tableau 2.1 est un exemple de baie de serveurs.

- **Les câbles** : les câbles ont pour rôle de rattacher les trois entités précédemment citées afin de transporter l'électricité vers les baies de serveurs. Ils sont caractérisés par les résistances que nous considérons constantes.

Tableau 2.1: Composants électriques d'un data center. À gauche : tableaux électriques Okken PCC, à Droite : baie de serveurs (source: news.pixelistes.com).

2.1.2 Réseau électrique : un graphe de flots

L'électricité, acheminée par le gestionnaire de réseau, arrive aux transformateurs basse tension qui généralement fonctionnent en mode triphasé. Ces transformateurs vont convertir la puissance *HTA* reçue (20KVA chez Enedis) en une puissance *BT* (400 KVA) et cette puissance est envoyée sur chaque phase. Une phase est un canal de transport de courant et le courant d'une phase est exprimé en fonction du sinus et d'un décalage de $2\pi/3$ par rapport au courant d'une autre phase. Chaque phase transporte cette énergie aux divers tableaux. Chaque tableau peut être rattaché à deux phases pour éviter les micro-coupures d'électricité. Pendant ces micro-coupures, les accumulateurs et les groupes électrogènes prennent le relai dans le but d'éviter une interruption de service. Les tableaux sont rattachés aux phases et aux baies par des câbles électriques. Chaque équipement mesure la quantité d'électricité qui le traverse. Les câbles sont unidirectionnels. Aucun équipement ne s'alimente lui-même et les équipements de même nature ne sont pas rattachés entre eux. Par exemple, il n'existe aucun câble entre des tableaux et aucune baie n'alimente une autre baie. L'électricité suit un sens : des sources vers les baies. Chaque équipement mesure la quantité d'électricité qui le traverse. Par convention avec les équipes métiers du réseau, nous avons décidé que ces mesures sont portées par les câbles incidents entrants dans chaque équipement.

Notre réseau électrique se modélise avec un réseau de flots dont le graphe est un graphe orienté sans circuit *Directed Acyclic Graph (DAG)* dans lequel chaque sommet représente un équipement, un arc représente des câbles électriques et qu'aucun équipement ne s'alimente soi-même (absence de circuits dans le réseau).

Conclusion : le réseau électrique d'un data center comprend des sources, des tableaux, des baies de serveurs et ils sont tous reliés par des câbles électriques. Les équipements de ce réseau ne s'alimentent pas soi-même et les équipements de même nature ne possèdent pas de câbles entre eux. Les câbles sont undirectionnels et l'électricité a toujours le même sens : de la source aux baies. Nous en concluons que la topologie du data center est un *graphe orienté sans circuit* induit par le sens de la circulation du courant électrique sur ces liens. Ainsi ce graphe est la topologie du réseau électrique.

2.2 Modélisation du réseau électrique du data center

2.2.1 Description du graphe de flots

Soit $G = (V, A, CAP)$ le graphe orienté sans circuit modélisant le réseau électrique. Chaque sommet de G est soit une source, soit un tableau ou soit un serveur. L'ensemble des sommets V est composé des équipements sources V_S, intermédiaires ou passifs V_I et charges ou serveurs V_C. Il est une union disjointe, deux à deux, des partitions V_C, V_I et V_S de V de cardinalité n dans laquelle :

- Les sommets V_S sont des sommets de degré entrant nul $d^- = 0$.
- Les sommets V_I sont des sommets de degrés entrant et sortant non nuls $d^- \neq 0, d^+ \neq 0$.
- Les sommets V_C sont des sommets de degré sortant nul $d^+ = 0$.

$$V = V_S \cup V_I \cup V_C \text{ et } V_S \cap V_I = \emptyset \text{ et } V_S \cap V_C = \emptyset \text{ et } V_C \cap V_I = \emptyset$$

L'ensemble des arcs A modélise m câbles électriques. Chaque arc a un flot, une capacité et une résistance considérée constante. La capacité de chaque arc est fonction de la grandeur associée à cet arc.
Par convention avec les équipes métiers du réseau, nous avons décidé que les arcs incidents entrants dans chaque sommet du graphe portent les mesures des grandeurs physiques.

2.2.2 Grandeurs, flots et contraintes physiques

Nous présentons les grandeurs physiques dans le réseau électrique puis définissons la capacité d'un arc et enfin décrivons les flots selon chaque grandeur pour un arc donné.

2.2.2.1 Grandeurs physiques

Le réseau électrique a deux modes de fonctionnement : le mode triphasé regroupant les grandeurs $U_{12}, U_{23}, U_{13}, I_{12}, I_{23}, I_{13}$ et le mode monophasé regroupant les grandeurs I, U. Les autres grandeurs sont communes aux deux systèmes (les grandeurs $P, Q, S, cos\phi$). Les symboles $U_{12}, U_{23}, U_{13}, U$ sont des tensions, $I_{12}, I_{23}, I_{13}, I$ des intensités, P, Q, S des puissances actives, réactives, apparentes respectivement et $cos\phi$ ou FP le facteur de puissance.
Une grandeur physique sur un arc est une caractéristique physique mesurable selon une unité de mesure. Selon le phénomène physique pris en compte, les grandeurs physiques sont prédéfinies. Dans le cas de l'électricité, les grandeurs physiques forment l'ensemble **GP** de cardinalité finie défini comme suit :

$$GP = \{I, I_1, I_2, I_3, U, U_{12}, U_{23}, U_{13}, P, Q, S, FP\} \tag{2.1}$$

avec le facteur de puissance FP qui désigne le déphasage entre l'intensité (I) et la tension (U).
Ces deux modes (triphasé et monophasé) peuvent fonctionner dans le même réseau. Cela implique qu'il n'existe qu'un seul sous-ensemble de grandeurs sur un arc, soit des grandeurs monophasées soit des grandeurs triphasées. On note GP^{a_i} l'ensemble des grandeurs sur un arc a_i.

$$\forall a_i \in A,\ \ GP^{a_i} \subset GP \tag{2.2}$$

On distingue deux types de grandeurs :

- Grandeurs à différentiel de potentiel : les tensions. On les note $gp_{ddp} \in \{U_{12}, U_{23}, U_{31}, U\}$.
- Grandeurs à effet calorique : l'intensité, la puissance active et réactive. On les note $gp_{cal} \in \{P, I_1, I_2, I_3, Q\}$.

2.2.2.2 Capacité d'un arc

Soit un arc $a_i \in A$ et $GP^{a_i} \in GP$ l'ensemble des grandeurs physiques associées à l'arc a_i. La capacité d'un arc a_i est une fonction Cap_{a_i} qui, pour chaque grandeur physique GP^{a_i} associe une valeur réelle positive $\mathbb{R}^+$.

$$Cap_{a_i} : GP^{a_i} \rightarrow \mathbb{R}^+ \tag{2.3}$$

Le vecteur CAP contient les capacités pour chaque grandeur et chaque arc.

$$CAP = (Cap_a[x])_{a \in A, x \in GP^a} \tag{2.4}$$

2.2.2.3 Description d'un flot physique

Les mesures physiques sont des valeurs de ces grandeurs. Le vecteur de mesures gp_a^x de norme $T^{a,x}$ (c'est-à-dire le nombre de valeurs associées à une grandeur dans une série temporelle) est une série de mesures associée à l'arc a et à la grandeur $x \in GP^a$ dont le i^{ieme} élément est $gp_a^x[i]$. Le vecteur gp_a^x associé à la grandeur $x \in GP^a$ et à l'arc $a \in A$ est défini comme suit :

$$gp_a^x = (gp_a^x[t])_{0<t<T^{a,x}} \tag{2.5}$$

avec $gp_a^x[t] \in \mathbb{R}^+$ et $t \in \mathbb{N}^+$.

Remarque 1. *Soient $gp_a^x[t]$ le t^{ieme} élément de la série temporelle de la grandeur $x \in GP^a$ et a, a' deux arcs distincts.*

- *$gp_a^x[t]$ et $gp_a^y[t]$, pour $x, y \in GP^a$ sont prises aux mêmes instants.*
- *pour $x \in GP^a \cap GP^{a'}$, $gp_a^x[t]$ et $gp_{a'}^x[t]$ sont prises à des instants différents.*

Certaines valeurs de gp_a^x sont indéfinies ou erronées dans certains cas.

2.2.2.4 Contraintes sur les flots

Un flot $gp_a^x[t]$ est admissible s'il respecte, pour chaque arc $a \in A$ traversé, la contrainte ci-dessous:

$$0 \leq gp_a^x[t] \leq Cap_a[x] \tag{2.6}$$

avec Cap_a la capacité de l'arc a pour la grandeur $x \in GP^a$.
Un flot est une fonction qui prend en entrées un arc a, une grandeur $x \in GP^a$, un vecteur gp_a^x associé à la grandeur x de l'arc a, un facteur de puissance $cos\phi$ ou FP associé à l'arc a et retourne un vecteur défini comme suit :

$$flo(a, x, gp_a^x) = f(a, x, r, cos\phi, gp_a^x) = \begin{cases} \frac{gp_a^x}{r \times cos\phi}, x \in gp_{ddp} \\ gp_a^x, x \in gp_{cal} \end{cases} \tag{2.7}$$

avec r la résistance du câble, FP ou cos ϕ le facteur de puissance.
Une valeur $flo_t(a, x, gp_a^x)$ de $flo(a, x, gp_a^x)$ s'obtient à un indice $t < T^{a,x}$ donné et se définit comme suit

$$flo_t(a, x, gp_a^x) = f(a, x, r, cos\phi, gp_a^x) = \begin{cases} \frac{gp_a^x(t)}{r \times cos\phi}, x \in gp_{ddp} \\ gp_a^x(t), x \in gp_{cal} \end{cases} \tag{2.8}$$

Soient $a \in A$ un arc et $x \in GP^a$ une grandeur liée à l'arc a. L'ensemble des arcs incidents à a ayant la même extrémité initiale que a est noté $succ(a)$ et l'ensemble des arcs incidents à a ayant la même extrémité finale que a est noté $pred(a)$. Tous les éléments de $succ(a)$ et $pred(a)$ ont les mêmes grandeurs physiques.
La fonction flo doit respecter la contrainte de la loi de conservation R [2]. La loi de conservation R ne s'applique qu'avec les grandeurs à effet calorique $gp_{cal} \in GP$ et se définit comme suit :

$$\sum_{a_j \in pred(a)} flo_t(a_j, x, gp^x_{a_j}) = \sum_{a_k \in succ(a)} flo_t(a_k, x, gp^x_{a_k}) + \epsilon \tag{2.9}$$

avec ϵ les pertes par Effet Joule. Cette équation est la loi de conservation ou de Kirchhoff.

2.2.2.5 Description de Verif-correl

La fonction $Verif - correl$ détermine le sous-ensemble d'arcs entrants et sortants d'un sommet du réseau électrique en se basant sur les mesures physiques et les lois de conservation R définies dans le paragraphe 2.2.2.4.
Soit $S \subset A$ l'ensemble fini d'arcs incidents à un sommet $v \in V$ et $x \in GP$ une grandeur physique telle que chaque arc $a \in S$ a un flot gp^x_a. Nous partitionnons S en deux sous-ensembles S_1 et S_2 tels que $S_1 \cap S_2 = \emptyset$.
La fonction $Verif - correl$ est booléenne, prend en paramètres S_1, S_2 et une grandeur x. Elle retourne 1 si :

- S_1 est l'ensemble des arcs entrants du sommet v.
- S_2 est l'ensemble des arcs sortants du sommet v.

Si les lois R ne sont pas vérifiées alors $Verif - correl$ retourne 0.

$$Verif - correl(S_1, S_2, x) = 1 \Leftrightarrow \sum_{a_i \in S_1} gp^x_{a_i}(t) - \sum_{a_j \in S_2} gp^x_{a_j}(t) \leq \epsilon$$

En effet, considérons t un instant de temps et $diff(t) = \sum_{a_i \in S_1} gp^x_{a_i}(t) - \sum_{a_j \in S_2} gp^x_{a_j}(t)$ la différence entre les flots entrants et sortants à un instant t. La différence $diff(t)$ doit toujours être inférieure aux pertes par Effet Joule ϵ quel que soit l'instant t.
Nous décidons que la fonction $Verif - Correl$ retourne 0 si le nombre de différences $diff(t)$ supérieure à ϵ est supérieur à un seuil (choisi à 10%).

Nous considérons, dans la suite du rapport, que la *décision de* $Verif - correl$ *est toujours exacte*. Cela signifie que si $Verif - correl(S1, S2, x) = 0$ alors les arcs de S ne partagent pas un sommet de V. Cependant, dans la pratique, nous n'utilisons pas cette fonction pour déterminer la topologie car pour un sommet v de la topologie ayant un ensemble d'arcs S_v, nous devons tester de l'ordre de $2^{|S_v|}$ bipartitions dans le pire des cas pour obtenir la bonne bipartition et cela est impossible pour S_v très grand.

Conclusion : le réseau électrique est modélisé par un graphe $G = (V, A, CAP)$. L'ensemble des sommets V est modélisé par des équipements sources V_S, intermédiaires V_I et serveurs V_C. Les sous-ensembles V_S, V_I, V_C sont disjoints deux à deux. Chaque arc a contient des grandeurs physiques GP^a. Nous en avons denombré 8 regroupées en grandeurs à différentiel de potentiel $gp_{ddp} = \{U_{12}, U_{23}, U_{31}, U\}$ et en grandeurs à effet calorifique $gp_{cal} = \{I_1, I_2, I_3, P\}$. Pour chaque grandeur physique $x \in GP^a$ associée à l'arc a, une capacité Cap_a et les mesures gp_a^x lui sont associées. Le flot de mesures flo est un vecteur de mesures qui dépend de l'arc a, de la grandeur physique x et de la mesure physique gp_a^x. Une valeur de flot flo_t vérifie la loi de conservation R et est définie comme suit :

$$flo_t(a, x, gp_a^x) = f[a, x, r, cos\phi, gp_a^x, t] = \begin{cases} \frac{gp_a^x[t]}{r \times cos\phi}, gp \in \{U, U_{12}, U_{23}, U_{13}\} \\ gp_a^x[t], gp \in \{P, Q, I, I_1, I_2, I_3\} \end{cases} \tag{2.10}$$

Nous avons défini la fonction $Verif - correl$, qui étant donnée deux ensembles d'arcs S_1 et S_2, affirme si ces arcs concourent en un sommet $v \in V$ en attribuant S_1 à l'ensemble d'arcs entrants et S_2 à l'ensemble d'arcs sortants du sommet v. Nous considérons que la réponse de $Verif - correl$ est toujours exacte.

2.3 Problème de découverte de topologie électrique

Notre problème est de découvrir la topologie d'un réseau électrique dont on ignore les sommets et on ne connait que les flots dans chaque arc.

1. **Données** :
   Nous avons donc

   - Un ensemble d'arcs distincts 2 à 2 du graphe G dont les extrémités **initiales** et **finales** sont inconnues.
     $A = \{a_1, ..., a_m\}$

- À chaque arc sont associées des séries de mesures $M(a_i)$.
  $\forall a \in A$, $M(a) = (gp_a^x)_{x \in GP^a}$
- Chaque série de mesures gp_a^x est de norme $T^{a,x}$ vérifiant la remarque de la section 1, $x \in GP^a \subset GP$
- $\forall t < T^{a,x}$, $gp_a^x[t]$ respecte les règles dans R.

2. **Objectif** :
   Déterminer la topologie du graphe G à partir des flots des arcs $M(a_i)$ et des règles R c'est-à-dire $\forall a_i$, déterminer n_1, n_2 tels que $a_i = (n_1, n_2)$.

3. **Approche** :
   Notre objectif est de déduire la topologie du réseau électrique représentée sous la forme d'un graphe de flots. Notre approche se subdivise en deux étapes :

   (a) La première étape consiste à la recherche d'arcs ayant des extrémités communes. Pour ce faire, nous calculons la similarité entre les paires de mesures d'arcs pour chaque grandeur dans le but de déterminer les arcs corrélés puis nous construirons la matrice de corrélation.

   (b) La seconde étape est la construction du graphe à partir de la matrice de corrélation.

2.4 État de l'art sur la découverte de topologie

Notre problème de découverte de topologie est d'identifier la topologie du réseau à partir des mesures c'est-à-dire les extrémités communes aux arcs dans le réseau.
Il existe un grand intérêt à la découverte de topologie notamment dans les systèmes distribués avec le déploiement de nouvelles générations de capteurs qui permettent de collecter des mesures selon différentes granularités (millisecondes, secondes, minutes, quart d'heure, etc). La communauté scientifique examine très peu comment interpréter ces mesures et l'impact de celles-ci dans le réseau. En effet, beaucoup de travaux sont orientés sur la découverte de la topologie au moyen de protocoles réseau. Des sondes sont propagées dans le réseau omettant la présence de capteurs dans les réseaux. D'autres travaux cherchent à prédire la topologie du réseau informatique à partir des lois statistiques et des modèles de files d'attentes.
Nous regroupons ces travaux en troix axes. Les deux premiers axes font principalement de la métrologie et cela leur permet de déduire la topologie en connaissant les caractéristiques de chaque lien et de chaque nœud du système. Le dernier axe porte sur la reconstruction de la topologie par les séries temporelles.

2.4.1 Découverte des topologies par des sondes

La découverte de topologie est un sujet important dans les réseaux informatiques. En effet, dans ces réseaux, on connait la topologie physique et les nœuds mais on ignore l'état des nœuds. On recherche alors la topologie fonctionnelle du réseau (l'interconnexion entre les nœuds) selon l'état des nœuds. Il existe de nombreux outils pour sonder et faciliter les tâches d'administration de ces réseaux. Ces outils permettent aux administrateurs réseau de manager efficacement et de découvrir le réseau. Nous citons HP OpenView [3] capable de localiser une erreur et d'envoyer des notifications de plusieurs évènements. Ces évènements incluent principalement des pertes dans le réseau et les informations sur les caractéristiques des liens. L'inconvénient de ces outils est leur coût et ils ne sont pas abordables pour les petites et moyennes organisations. La découverte de réseaux informatiques s'effectue aussi avec des protocoles réseaux dont les plus connus sont ICMP [4] et SNMP [5]. Divers algorithmes ont été proposés. Nous pouvons citer l'algorithme de *Narayan et al.* [6] qui réalise la découverte de topologie et de services pour des réseaux hétérogènes en se servant du système netInventory [7]. Cet algorithme est basé sur deux hypothèses : (i) chaque domaine doit avoir un seul sous-réseau et ($2i$) les tables de routage sont complètes. Pour la découverte de topologie, le système *netInventory* énumère la liste des adresse IP, envoie des messages ECHO ou ICMP pour déterminer si un nœud est actif. Dans le cas où PING est désactivé, netInventory se sert de ipRouteTable et ipNetToMediaTable dans les routeurs pour connaitre l'état d'un nœud.
De même, l'algorithme proposé par *Kuangyu Qin* [8] est basé sur *SNMP* et suppose que l'administrateur réseau est connecté à l'interface de routage et envoie des paquets de découverte aux routeurs. La station de gestion débute la découverte par la lecture des tables de routage. En utilisant les MIB (Management Information Base) et les agents SNMP, cet algorithme élimine les équipements redondants et génère une topologie efficiente même quand il existe des VLAN dans le réseau.
Un autre algorithme, proposé par *Bilal Saeed, TarekSheltami et Elhadi Shakshuki* [9], est basé sur netInventory et accélère la découverte de la topologie sans générer un trafic supplémentaire. Cet algorithme, nomme TDA (Topology Discovery Algorithm), se sert de la programmation parallèle pour gérer toutes les requêtes de découverte des interconnexions réseaux sur la plateforme Android. Une étude bibliographique, effectuée par *Ahmed et al.* [10], fournit les différentes techniques et algorithmes pour la découverte de topologie de réseaux informatiques. Les auteurs déclarent que la plupart des algorithmes de découverte de topologie physique de réseau sont basés sur le protocole SNMP. En d'autres termes, ils supposent que tous les nœuds de ce réseau sont repertoriés et activés au moment de la découverte.

Conclusion : cette méthode est contraire à notre sujet de recherche dans lequel nous ignorions les nœuds de notre réseau.

2.4.2 Tomographie des réseaux

La tomographie de réseaux est une méthode qui étudie les caractéristiques internes d'un réseau (c'est-à-dire les liens et nœuds ON/OFF, la bande passante, la congestion du réseau) en utilisant les mesures point-à-point (entre nœuds) obtenues à partir des sondes placées dans ce réseau et en supposant que le réseau est modélisable (on peut définir un modèle avec l'estimateur du maximum de vraisemblance ou l'inférence bayésienne). Il fait aussi la prédiction de la topologie du réseau.
Cette méthode est utilisée en l'absence de système de supervision fiable pour identifier les caractéristiques des liens et faire un diagnostic de ce dernier (quel nœud/lien est indisponible, congestionné ou ajouté) car il est impossible de contrôler les flux et l'état des équipements. Les problèmes, résolus par la tomographie des réseaux, sont regroupés en 3 catégories :

2.4.2.1 L'estimation des paramètres (caractéristiques) d'un lien

L'article de *Ghita et al.* [11] se propose de découvrir la congestion des liens dits "corrélés" dans un réseau informatique. Un lien entre deux nœuds du réseau est une connexion logique au niveau de la couche 3 du protocole TCP/IP et deux liens sont corrélés s'ils appartiennent au même domaine ou sous-réseau. Pour réaliser cet algorithme, il considère que la topologie du réseau et le degré de corrélation entre les liens sont connus et qu'il existe un trafic unicast entre les nœuds dans le réseau (ce trafic est désigné par chemin). Il énonce quatre hypothèses pour l'expérimentation de l'algorithme : (i) l'ensemble des chemins reste inchangé durant chaque simulation; ($2i$) chaque chemin est congestionné si au moins un lien du chemin est congestionné; ($3i$) le comportement de congestion de chaque lien pendant chaque simulation est modélisé par un processus aléatoire stationnaire; ($4i$) deux ensembles de corrélations disjoints ne doivent pas être traversés par les mêmes chemins.
Différentes méthodes ont été proposées et validées mais elles diffèrent de l'algorithme *Ghita et al.* [11] par les caractéristiques des liens fournis. En effet, les méthodes initiales se basent sur les corrélations temporelles, chacune parfaitement corrélée et envoyée par des packets multicast [12, 13, 14, 15]. Tous les taux de pertes des liens sont statistiquement identifiables dans une topologie en arbre [16]. Cependant le multicast n'est pas largement déployé et les groupes de paquets unicast exigent un développement subtantiel et un coût d'administration élevé. D'où il est moins aisé de sélectionner les corrélations temporelles.
L'ensemble des méthodes qui suivent [17, 18, 19, 20] utilisent seulement des mesures unicast point-à-point (c'est-à-dire des mesures sur les liens) dans le simple but d'identifier les congestions de liens.
Les méthodes booléennes de tomographie de réseaux considèrent des hypothèses supplémentaires [18, 17] pour identifier les liens congestionnés en trouvant le plus petit ensemble de liens qui peut être expliqué par ces mesures. Ces hypothèses sont : (i) les liens sont indépendants; ($2i$) les liens sont

congestionnés équiprobablement; ($3i$) le nombre de liens congestionnés est faible. Toutes les précédentes méthodes [11, 17, 18, 19, 20] se basent sur l'indépendance des liens c'est-à-dire l'hypothèse (i).

2.4.2.2 La prédiction de topologie

L'objectif de cette méthode est d'identifier l'arbre de la topologie connectant un serveur aux autres machines du réseau. L'idée est d'utiliser une fonction croissante (ou monotone) du nombre de liens partagés entre deux nœuds ou le maximum de vraisemblance pour trouver l'arbre.
L'article de *Coates and al.* [21] suppose que les mesures de la couche 2 (protocole TCP/IP) des machines clientes sont assez fiables pour découvrir le réseau. Les mesures sont collectées à l'aide de l'outil Traceroute. Ensuite l'auteur définit un modèle basé sur les chaines de Markov de Monte Carlo pour déterminer les caractéristiques des liens du réseau et déduire les topologies probables. Enfin il propose un critère global du seuil maximum pour l'identification de topologie contrairement aux autres travaux [22, 23] qui emploient des stratégies semi-optimales de fusion des liens.

2.4.2.3 La densité du trafic entre émetteur/recepteur

Un problème de tomographie de réseaux qui retient notre attention est l'estimation de la matrice de trafic qui prévoit le volume de flots entre les nœuds point-à-point à partir des mesures [24, 25]. En effet, les variables inconnues sont les volumes de flots et nous suivons une certaine loi de probabilité.
Les corrélations de flots ont été étudiées par *Singal et Michailidis* [26] et ils montrent que, sous certaines classes de dépendances, les moments d'ordre n de ces volumes de flots sont identifiables à partir des mesures de liens pour $n \geq 2$. Cette approche diffère de 2 aspects par rapport à l'estimation des caractéristiques de liens. Premièrement, l'estimation des caractéristiques s'intéresse aux mesures point-à-point c'est-à-dire sur les liens tandis que le calcul du trafic point-à-point doit être estimé dans la densité de trafic [24, 26, 25]. Deuxièmement, l'estimation des caractéristiques utilise les variables booléennes et la connaissance des valeurs des variables des lois de distributions n'est pas nécessaire comme cela se fait dans la densité de trafic. La connaissance des valeurs des paramètres engendre la restriction de l'extension des résultats théoriques [26].

Conclusion : la tomographie de réseaux n'est pas appropriée pour notre sujet de recherche car nos mesures d'arcs déterminent les caractéristiques de ces arcs et les nœuds sont inconnus empêchant la découverte de réseau qui est l'étape nécessaire pour débuter la tomographie de réseaux.

2.4.3 Reconstruction de la topologie par les séries temporelles

Le brevet de *Chaudhary et al.* [27] décrit comment trouver le graphe induit par un réseau de canalisations de fluides en se servant des mesures de capteurs de différentes stations qui émettent des fluides. Il considère le réseau comme un arbre dans lequel la racine est une station comprenant un compresseur. La présence du compresseur induit un délai de livraison entre la station source et les stations de livraison (délai d'acheminement du fluide entre les nœuds du réseau). Le délai est dû au temps mis par le fluide pour atteindre la station de livraison.
Le traitement des mesures [28, 29, 30] porte sur la recherche de pics, la suppression d'anomalies (observations non conformes à un motif dans le dataset de données) et le lissage des données. Les données obtenues identifient la relation d'adjacences entre les nœuds. Le traitement des retards temporels déterminent la distance entre des nœuds du réseau. L'analyse des séries temporelles par paires détermine la causalité entre les capteurs associés. La causalité est la recherche d'évènements identiques observés dans deux séries de mesures. En d'autres termes, la causalité consiste à savoir si les évènements, observés dans une série de mesures, se reproduisent dans une autre série. Un modèle de causalité basé sur les séries temporelles des stations est défini comme une régression multiple avec un modèle de Granger. À partir de ce modèle, on calcule les différents retards $X(t)$ et leurs coefficients. On définit également un modèle de pénalisation basé sur une régression LASSO afin de filtrer les relations de causalité et obtenir un graphe creux (sparse). Le graphe de causalité est alors un arbre.
Les auteurs *Chaudhary et al.* proposent un script s'exécutant récursivement sur les nœuds du réseau en choisissant, à chaque étape, les nœuds qui ne sont pas des puits. En se servant du graphe de causalité (corrélation) et des retards de propagation, il détermine les voisins du nœud X_i et supprime les nœuds voisins de X_i ayant une causalité entre eux car le réseau est un arbre.
La méthode décrite ici est la construction du graphe itérativement à partir des sous-arbres du graphe. En effet, le nombre de sous-arbres est l'ordre dans lequel on a supprimé les nœuds puits. Cette méthode est réalisable en $O(n^2)$ car chaque sommet est traité une fois et la recherche de son voisinage est en $O(n)$ avec n le nombre de sommets.
Cette méthode est inadaptée pour un DAG car sa complexité est en $O(n^n)$. Elle ne s'exécute que sur des arbres.

Conclusion : l'étude bibliographique effectuée sur la découverte de topologie montre que la recherche sur ce sujet est très active dans le domaine des réseaux informatiques, précisement dans la détection de nœuds/liens congestionnés. Toutefois, dans le domaine énergétique, les rares travaux réalisés sur ce sujet mettent l'accent sur la découverte de topologie par la reconstruction des sous-graphes en supposant que les nœuds du réseau sont connus, que certains liens sont absents, que les mesures sont influencées

par les incidents et que des erreurs peuvent être présentes sur nos données. Ces travaux s'avèrent moins pertinents pour notre problématique parce que les résolutions proposées s'appuyent sur des hypothèses qui sont différentes des nôtres. En effet, nous connaissons que les liens et les mesures qui circulent sur ces liens. Mais nous ignorons exactement les extrémités de ces liens. Par ailleurs, ces mesures suivent des lois physiques qui impliquent la propagation d'évènements dans ces réseaux. Notre problématique est alors de déterminer les extrémités partagées entre les liens grâce aux lois et aux mesures physiques et aussi à la théorie des graphes.

2.5 Conclusion du chapitre 2

Dans ce chapitre, nous avons montré que le réseau électrique se modélise par un graphe de flots dont les sommets sont des équipements, les arcs sont les câbles électriques unidirectionnels et les flots par les mesures des équipements. Par convention, ces mesures sont portées par les arcs incidents entrants dans chaque sommet du graphe. Chaque mesure est associée à une grandeur physique et à chaque grandeur est définie une capacité sur l'arc. Nous avons redéfini la loi de conservation R en fonction de nos mesures, de nos grandeurs et nos capacités.

Dans ce graphe, nous connaissons les liens et les mesures qui circulent sur ces liens. Mais nous ignorons exactement les extrémités de ces liens. Par ailleurs, ces mesures suivent des lois physiques qui impliquent la propagation d'évènements dans ces réseaux. Notre problème est alors de déterminer les extrémités partagées entre les liens grâce aux lois R, aux mesures physiques et aussi à la théorie des graphes.

Une étude bibliographique a été réalisée sur la découverte de topologie. Les travaux concernent principalement la reconstruction de topologie dans le domaine informatique à partir de sondes et de protocoles de réseau. Ces travaux s'accentuent sur la supervision de réseau, l'administration du réseau et aussi sur la recherche des caractéristiques des éléments du réseau. Dans le domaine énergétique, nous avons trouvé un brevet qui reconstruit des sous-graphes du réseau à partir de la propagation des incidents. Tous ces travaux supposent que nous connaissons l'état de tous les éléments du réseau. Ce qui est contraire à notre problématique.

Nous avons proposé deux approches pour résoudre notre problème. La première approche consiste à déterminer la corrélation entre les arcs à partir des mesures et des règles de flots puis à construire une matrice de corrélation. La seconde approche découvre le réseau en effectuant certaines transformations sur la matrice de corrélation.

Chapitre 3

Mesures : Des Séries Temporelles

Les mesures sur les arcs forment des séries temporelles. Certains arcs partagent des équipements que nous souhaitons déterminer à partir des séries temporelles. Nous supposons que les séries temporelles associées à ces arcs ont les mêmes comportements au cours du temps. En d'autres termes, toute variation dans une série est visible dans une autre série. Nous disons que ces arcs sont *corrélés* et la valeur liée à cette relation entre les arcs est désignée par *coefficient de similarité*.
Dans le chapitre précédent, nous avons modélisé les mesures sur les arcs par des séries temporelles et la particularité de ces séries est la présence de valeurs érronées et manquantes.
Le problème est de savoir s'il existe une méthode de calcul du coefficient de similarité qui tienne compte des erreurs dans les séries temporelles et qui vérifie l'hypothèse sous-jacente.
Pour ce faire, nous procédons comme suit : la première partie énonce les analyses sur les séries temporelles. La seconde partie présente les différentes méthodes de calcul des coefficients de similarité entre les séries. Et enfin la dernière partie sélectionne la méthode de calcul du *coefficient de similarité* puis analyse les performances de cette méthode sur les données réelles d'un *sous-réseau du data center Champlan*.

3.1 Séries temporelles

Définition 1. *Une série temporelle est une suite chronologique de valeurs réelles x_t à des instants de temps régulièrement espacés.*

$$(x_t)_{t\in\Theta} \tag{3.1}$$

avec Θ l'ensemble discret et fini des espaces de temps de dimension n.

L'intervalle de temps entre deux mesures successives dépend de la série. Il peut s'agir d'une minute,

d'une semaine, d'un jour, etc. Généralement, les séries temporelles sont utilisées pour comprendre les mécanismes qui produisent ces observations. Ces mécanismes sont associés au temps et permettent de faire les analyses suivantes :

- La modélisation : la représentation de la série sous la forme d'une fonction du temps.
- La prévision : prédire les données futures à partir de valeurs prédécentes.
- La détection de rupture : la série change-t-elle significativement à un instant t.
- La comparaison : déterminer la relation existante entre une série observée et d'autres séries candidates.

Nous décrivons brièvement les modèles d'analyses sur des séries temporelles puis nous présentons l'objectif recherché par l'analyse de ces séries dans notre étude.

3.1.1 La modélisation et la prévison d'une série temporelle

Un modèle est une image simplifiée de la réalité qui vise à traduire le fonctionnement d'un phénomène et permet de mieux les comprendre. Nous distinguons deux types de modèles :

- Les modèles déterministes : ils utilisent les éléments de la statistique descriptive et suppose que l'observation de la série à la date t est une fonction du temps t et d'une variable ϵ_t:

$$x_t = f(t, \epsilon_t).$$

  La variable ϵ_t est le résidu ou l'erreur du modèle et elle représente la différence entre la réalité et le modèle proposé.
  Les deux modèles les plus utilisés sont les suivants :

  - Le modèle additif : c'est la décomposition de la série en trois termes

$$x_t = Z_t + S_t + Q_t$$

    où Z_t est la tendance, S_t la périodicité et Q_t les composantes (erreurs) identiquement distribuées. Z_t, S_t sont aussi déterministes.
  - Le modèle multiplicatif : la variable x_t est le produit de la tendance et d'une composante périodique :

$$x_t = Z_t(1 + S_t)(1 + Q_t).$$

Toutefois, l'application d'un logarithme nous permet de revenir au modèle additif.

$$y(t) = log(x_t) = log(Z_t) + log(1 + S_t) + log(1 + Q_t).$$

- Les modèles stochastiques : ils font l'hypothèse que les résidus ϵ_t ne sont pas indépendants et qu'il est possible de prévoir les résidus en partie. L'avantage réside dans la réduction de l'imprécision de la prévision des termes futurs de la série temporelle. La variable ϵ_t devient une fonction des valeurs du passé et d'un terme d'erreur η_t

  $$\epsilon_t = (\epsilon_{t-1}, \epsilon_{t-2}, ..., \eta_t).$$

  Nous pouvons citer, comme exemple de modèles couramment utilisés, les modèles SARIMA, ARIMA et ARMA. Dans ces modèles, la modélisation porte sur la forme du processus (ϵ_t). Un exemple de modélisation est le modèle autorégressif linéaire d'ordre 2 avec des coefficients autorégressifs a_1, a_2 définis par

  $$\epsilon_t = a_1 x_{t-1} + a_2 x_{t-2} + \eta_t,$$

  où η_t est un bruit blanc.

Un exemple de série temporelle et de ses différentes composantes sont présentés dans la figure 3.1. La série temporelle est la donnée du trafic prise chaque mois des routes françaises de 1989 à 1996. La première ligne est la répresentation de la série temporelle et la seconde est celle de la tendance. Quant à la troisième et la dernière ligne, elles représentent respectivement la figure de périodicité et les résidus. La période de la série est de 12.

Les deux types de modèles ci-dessus induisent des techniques de prévision bien particulières. Schématiquement, ils séparent la tendance de la saisonnalité éventuelle. Puis ils cherchent à les modéliser et à les estimer. Enfin ils les éliminent de la série : ces deux opérations sont nommées la *détendancialisation* et la *désaisonnalisation* de la série. Une fois ces composantes éliminées, on obtient la série aléatoire ϵ_t :

- Pour les modèles déterministes, cette série est considérée comme décorrélée et il n'y a plus rien à faire.

- Pour les modèles stochastiques, on obtient une série stationnaire (ce qui signifie que les observations successives de la série sont identiquement distribuées mais pas nécessairement indépendantes) qu'il s'agit de modéliser.

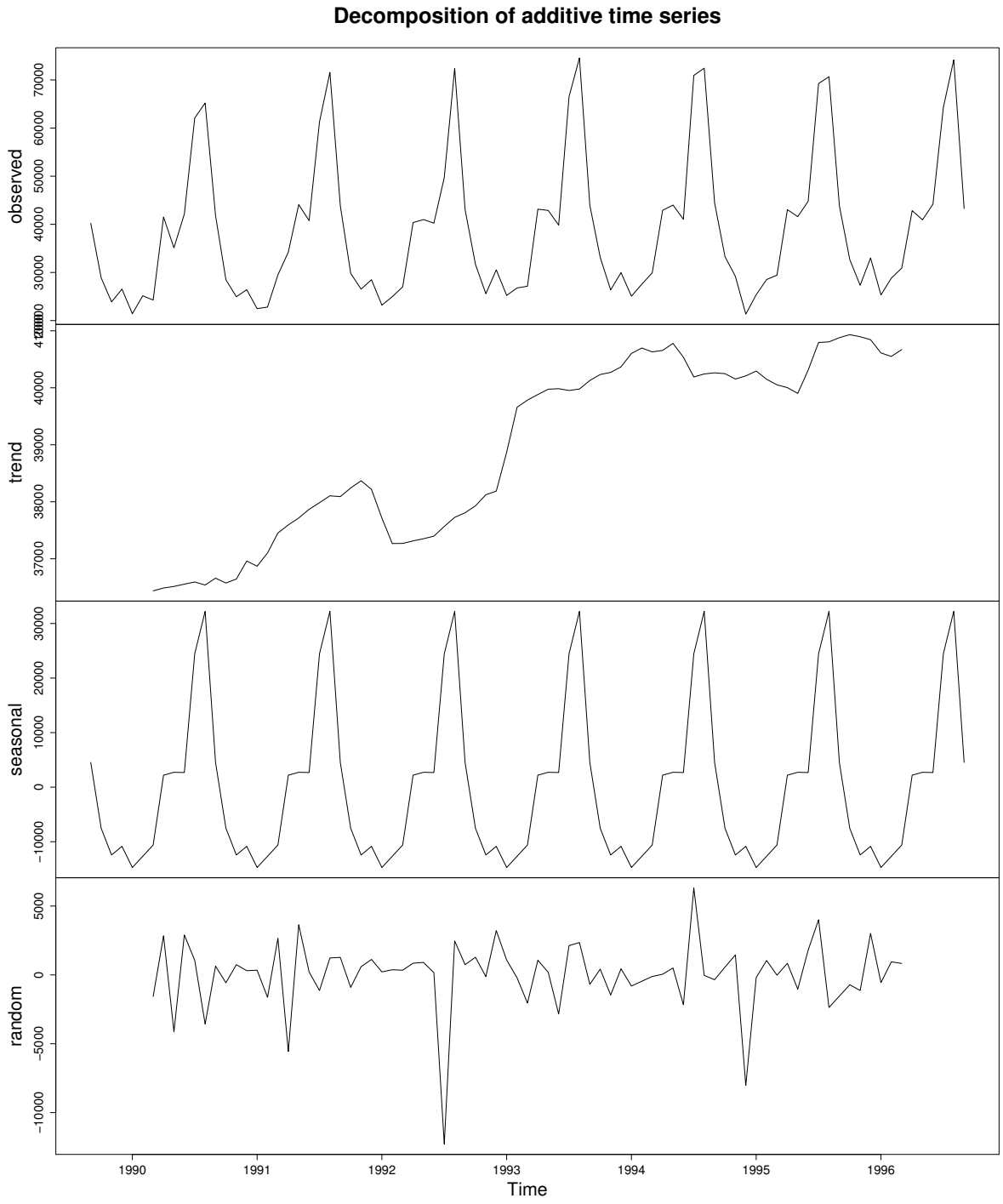

Figure 3.1: Décomposition de la série temporelle representant les valeurs mensuelles du trafic routier de l'autoroute $A7$ de 09/1989 à 09/1996. Représentation des composantes de la série temporelle.

3.1.2 La détection de rupture

La détection de rupture consiste à déceler la présence d'un ou de plusieurs pics dans la série temporelle et à les localiser dans la série temporelle. Déterminer l'existence d'une rupture est d'autant plus difficile que cette dernière n'est pas forcément caractérisée par un décalage de grande amplitude entre x_t et x_{t+1} par rapport à la dispersion des observations. Un enjeu de la détection est donc d'être sensible aux faibles variations tout en garantissant une certaine robustesse au bruit. La thèse de master de *Flore Harlé* [31] propose une méthode de détection de changements univariée et multivariée à la médiane des segments, le segment étant une subdivision de la série temporelle. Les modèles proposés sont exprimés par des fonctions de vraisemblance afin d'illustrer l'augmentation de la difficulté lors de l'ajout de nouvelles inconnues.

3.1.3 La comparaison de séries temporelles

Si deux séries sont observées, nous pouvons nous demander quelle influence elles exercent l'une sur l'autre. Par exemple, étant donnée deux séries X_t et Y_t, nous vérifions s'il existe par exemple des relations du type $Y_t = a_1 \times X_{t+1} + a_3 \times X_{t+3}$.
Ici, deux questions se posent : tout d'abord, la question de la causalité c'est-à-dire quelle variable (ici (X_t)) va expliquer l'autre (ici (Y_t)), ce qui conduit à la deuxième question, celle du décalage temporel: si une influence de (X_t) sur (Y_t) existe, avec quel délai et pendant combien de temps la variable explicative (X_t) influence-t-elle la variable expliquée (Y_t) ?

Nous allons considérer un réseau de flots modélisé par un graphe G et les arcs portent des mesures. Les mesures sont modélisées par des séries temporelles. Dans notre cas d'étude, nous cherchons à comparer des séries temporelles en supposant que les variations dans une série sont observables dans une autre série. Pour ce faire, nous rappelons que le coefficient de similarité est une valeur indiquant la relation existante entre deux arcs et nous définissons la corrélation entre des arcs comme suit :

Définition 2. *Corrélation entre arcs*
Soit corr le coefficient de similarité entre les séries x et y défini de $\mathbb{R}^n \times \mathbb{R}^n \to [0,1]$ et les arcs A et B contenant respectivement les séries x et y.
Deux arcs A et B sont corrélés si et seulement si le coefficient de similarité entre les séries x et y est contenu dans $[0.5, 1]$. ($corr(x,y) \in [0.5,1]$)

On dit alors que A et B sont *fortement corrélés* si le coefficient de similarité est contenu dans l'intervalle $[0.7, 1]$ ($corr(x,y) \in [0.7,1]$) et *faiblement corrélés* si $corr(x,y)$ appartient à l'intervalle

$[0.5, 0.7[$ $(corr(x, y) \in [0.5, 0.7[)$.

Notre objectif est de trouver la méthode qui calcule au mieux la corrélation entre des arcs en tenant compte des caractéristiques de nos données (valeurs manquantes et erronées à certains instants t dans les séries temporelles des arcs).

3.2 État de l'art des méthodes de similarité

Les différentes méthodes de calcul des coefficients de similarité se basent sur les séries temporelles associées aux arcs du réseau. Nous allons présenter les principales méthodes de calcul de similarité qui sont regroupées en 3 familles et qui sont détaillées dans cette section.

3.2.1 Similarité sur les séries entières

Les séries entières sont considérées comme des vecteurs et comparées avec une distance qui utilise toutes les valeurs des séries. Cette distance calcule la similarité entre ces deux séries. La similarité entre deux séries entières est excellente s'il existe des caractéristiques discriminatoires identiques entre ces séries sur l'axe du temps. Ces caractéristiques peuvent être identifiées aux mêmes instants de temps ou à des instants décalés mais constants dans le temps.
Par exemple, considérons le dataset *FiftyWord* [32] dans lequel les données proviennent de la base de donnée UCR [33] et décrivent les contours des mots écris par Georges Washington dans sa bibliothèque privée. Dans la figure 3.2, nous distinguons 4 séries regroupées en 2 classes. Les deux séries du haut (en noir) identifient la classe 30 et les deux séries du bas (en vert) identifient la classe 50. Le motif commun est observable en alignant les séries.

Les approches basées sur les vecteurs sont pertinentes quand il existe un décalage de temps entre les pics et les creux des séries, comme c'est le cas avec les deux courbes du bas dans la figure 3.2.
Les méthodes telles que *Time Warp Edit*, *Move-Split-Merge*, *Longest Common Subsequence* et *distance de Pearson* sont des distances de *mesures élastiques* qui se servent de l'approche vectorielle. Les distances de *mesures élastiques* sont les meilleures approches pour traiter les problèmes des séries entières à savoir la détection de pics, de décalage et de creux entre les séries.

3.2.1.1 Dynamic Time Warping DTW

Dynamic time warping (DTW) [34] est une technique pour trouver l'aligmenent optimal entre deux séquences dépendant du temps sous certaines contraintes (figure 3.3). Les séries temporelles sont

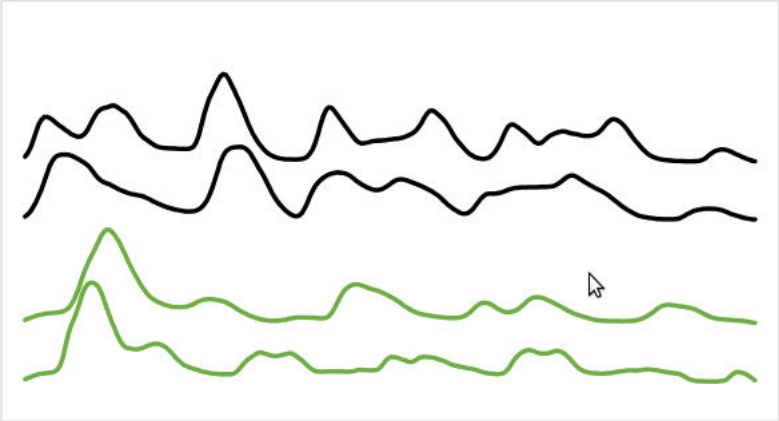

Figure 3.2: Détection du motif commun en alignant les séries. Les deux séries du haut représentent la classe 30 et les deux séries du bas représentent la classe 50.

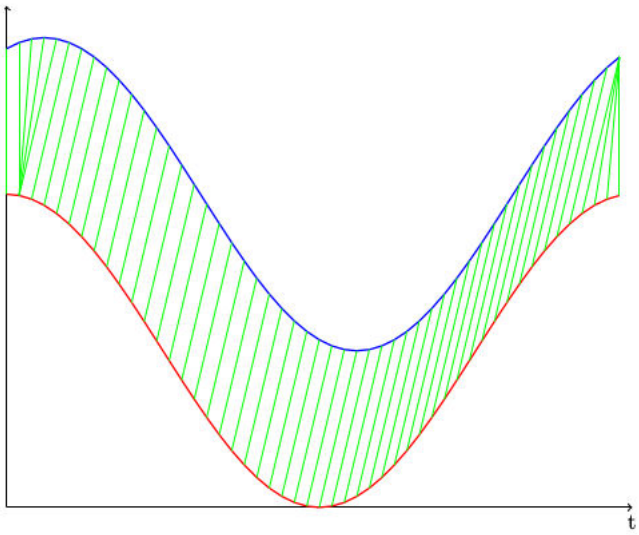

Figure 3.3: Deux séquences en dimension 1 alignées avec Dynamic Time Warping. Les coordonnées de la séquence du haut et de celle du bas correspondent respectivement à $cos(t)$ et à $cos(t+\alpha)$. Pour des questions de visualisation, la séquence du dessus a été décalée vers le haut lors du tracé. [1]

déformées par une transformation non-linéaire de la variable temporelle, pour déterminer une mesure de leur similarité, indépendamment de certaines transformations non-linéaires du temps.
Supposons que nous souhaitons mesurer la similarité entre deux séries $A = (a_1, \cdots, a_m)$ et $B = (b_1, \cdots, b_m)$. Soit $M(A, B)$ la matrice de distances entre A et B où $M_{i,j} = (a_i - b_j)^2$. Un chemin d'alignement *Dynamic Time Warping (DTW)* est une méthode inspirée de la distance de *Levenshtein*, également appelé *distance d'édition (ED)*. À l'origine, *DTW* était appliqué dans le domaine de la reconnaissance vocale et permet de trouver l'alignement global optimal entre deux séquences, c'est-à-

dire faire correspondre chaque élément de chaque séquence à au moins un élément de l'autre séquence en minimisant les coûts d'association. Le coût d'une association correspond à la distance entre les deux éléments, classiquement une norme L_p [35]. La figure 3.3 représente un exemple d'alignement opéré par *DTW*. Il illustre l'alignement de deux sinusoïdes légèrement déphasées. Le résultat numérique fourni par *DTW* correspond à la somme des hauteurs des "barreaux" formés par les associations. Les extrémités des alignements de la figure 3.3 montrent que *DTW* est capable de réaligner correctement une séquence par rapport à une autre, et parvient ainsi à saisir des similarités que la distance euclidienne ne peut extraire.
La distance *Dynamic Time Warping* est définie récursivement par :

$$D(A_i, B_j) = \delta(A_i, B_j) + min \begin{cases} D(A_{i-1}, B_{j-1}), \\ D(A_i, B_{j-1}), \\ D(A_{i-1}, B_j)) \end{cases}$$

où A_i représente la sous-séquence $(a_1, \cdots, a_i)$. Le coût de l'alignement optimal est alors donné par $D(A_{|A|}, B_{|B|})$. Le principe de programmation dynamique peut alors se résoudre par un arbre en partant des feuilles en supposant que le problème principal peut être symbolisé par la racine et les sous-problèmes par des nœuds appartenant aux différents sous-arbres. La fonction DTW peut alors être mémoïsée : les différents appels peuvent être retenus afin de ne pas calculer deux fois la fonction appelée avec les mêmes paramètres. Aussi est-il habituel, comme l'arbre contient $|A|.|B|$ nœuds différents, de stocker ces différents résultats intermédiaires dans une matrice $|A| \times |B|$. Le calcul de *DTW* consiste alors à trouver le chemin de coût minimum dans la matrice, ce qui s'exécute avec une complexité en temps et en mémoire de $\Theta(|A| \times |B|)$.

3.2.1.2 Time Warp Edit TWE

La distance *Time Warp Edit* (TWE) est une mesure de distance pour des séries temporelles discrètes. En comparaison avec les distances telles *Dynamic Time Warping (DTW)* [34] et *longest common subsequence (LCS)* [36], la distance *TWE* est une métrique proposée par *P.F. Marteau* en 2009. La distance *TWE* a quatre propriétés :

- Traite le décalage temporelle local avec des performances élevées.
- Satisfait l'inégalité triangulaire c'est-à-dire $AB \leq AC + CB$ avec les distancesAB, AC, CB des cotés d'un triangle quelconque.
- Comprend le paramètre de rigidité ν qui contrôle l'élasticité de la métrique.

- Utilise la différence de temps pour comparer les segments de séries temporelles comme des coûts de correspondances locales.

Le paramètre ν est important dans l'algorithme de *TWE* parce qu'il rend plus flexible l'identification de motifs (matching) entre les séries temporelles. La preuve de son efficacité a été prouvée dans [37].

L'algorithme de *TWE* introduit trois opérations : $delete_A$, $delete_B$ et *match* pour l'édition de deux séries discrètes A et B. L'édition d'une série temporelle consiste à modifier cette série en sous-ensembles de même cardinalité à partir d'une des trois opérations et chaque sous-ensemble est désigné par *séquence*. La similarité entre les séries A et B est le coût minimum de séquences nécessaire pour transformer A en B. En se basant sur ces trois opérations, l'algorithme de *TWE* calcule le coût d'une séquence à chaque opération pour toutes les paires de séries avec l'équation 3.2. Dans cette équation, U désigne l'ensemble des séries finies $U = \{A_1^p \mid p \in \mathbb{N}\}$ où A_1^p est une série avec des temps discrets variant de 1 à p, A_1^0 est la série nulle de longueur nulle et a'_i désigne le i^{ieme} élément de la série A. Nous considérons alors que $a'_i \in S \times T$ où $S \subset R^d$ avec $d \geq 1$ intègre un espace multidimensionnel de variables et $T \subset R$ intègre la variable temporelle. Ainsi, nous pouvons écrire $a'_i = (a_i, t_{a_i})$, où $a_i \in S$ et $t_{a_i} \in T$ avec la condition que $t_{a_i} > t_{a_j}$ quand $i > j$ (le temps est strictement croissant dans la séquence des éléments). La pénalité notée λ est constante. La similarité entre les séries A et B est calculée récursivement par

$$\delta_{\lambda,\nu}(A_1^p, B_1^q) = min \begin{cases} \delta_{\lambda,\nu}(A_1^{p-1}, B_1^q) + \Gamma(a'_p \to \Lambda) & delete_A, \\ \delta_{\lambda,\nu}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \to b'_q) & match, \\ \delta_{\lambda,\nu}(A_1^{p-1}, B_1^q) + \Gamma(\Lambda \to a'_p) & delete_B, \end{cases} \tag{3.2}$$

avec

$$\begin{aligned} \Gamma(a'_p \to \Lambda) &= d(a'_p, a_{p-1}) + \lambda \\ \Gamma(a'_p \to b'_q) &= d(a'_p, b'_q) + d(a'_{p-1}, b'_{q-1}) \\ \Gamma(\lambda \to b'_q) &= d(b'_q, b'_{q-1}) + \lambda. \end{aligned}$$

La récursivité est initialisée par

$$\begin{cases} \delta_{\lambda,\nu}(A_1^0, B_1^0) = 0, \\ \delta_{\lambda,\nu}(A_1^0, B_1^j) = \infty \ pour \ j \geq 1, \\ \delta_{\lambda,\nu}(A_1^0, B_0^j) = \infty \ pour \ j \geq 1, \end{cases}$$

avec $a'_0 = b'_0 = 0$ par convention.

L'algorithme *TWE* introduit la rigidité, constante positive ν, dans la définition de $\delta_{\lambda,\nu}$ en choisissant $d(a', b') = d_{LP}(a, b) + \nu \times d_{LP}(t_a, t_b)$ qui caractérise la rigidité de $\delta_{\lambda,\nu}$. Si $\nu = 0$ alors $\delta_{\lambda,\nu}$ est la distance

de S et non de $S \times T$. Dans cette équation, d_{LP} est la métrique norme L_p [35]. L'expression finale de $\delta_{\lambda,\nu}$ est présentée par l'équation 3.3.

$$\delta_{\lambda,\nu}(A_1^p, B_1^q) = min \begin{cases} \delta_{\lambda,\nu}(A_1^{p-1}, B_1^q) + \Gamma(a'_p \to \Lambda) & delete_A, \\ \delta_{\lambda,\nu}(A_1^{p-1}, B_1^{q-1}) + \Gamma(a'_p \to b'_q) & match, \\ \delta_{\lambda,\nu}(A_1^{p-1}, B_1^q) + \Gamma(\Lambda \to a'_p) & delete_B, \end{cases} \tag{3.3}$$

avec

$$\begin{aligned} \Gamma(a'_p \to \Lambda) &= d_{LP}(a'_p, a_{p-1}) + \nu.(t_{a_p} - t_{a_{p-1}}) + \lambda \\ \Gamma(a'_p \to b'_q) &= d_{LP}(a'_p, b'_q) + d_{LP}(a'_{p-1}, b'_{q-1}) + \nu.(|t_{a_p} - t_{b_q}| + |t_{a_{p-1}} - t_{b_{q-1}}|) \\ \Gamma(\lambda \to b'_q) &= d_{LP}(b'_q, b'_{q-1}) + \nu.(t_{b_q} - t_{b_{q-1}}) + \lambda \end{aligned}$$

3.2.1.3 Move-Split-Merge MSM

Move Split Merge (MSM) est une distance qui est basée sur le coût de transformation d'une série temporelle en une autre série en utilisant une séquence d'opérations de *move*, *split* et *merge*. *MSM* a l'avantage d'être une métrique et être invariant au choix de l'origine de la série. En effet, soit $X = (x_1, \cdots, x_m)$ une série temporelle dans laquelle x_i est un réel. Une translation de X par t, où t est aussi un réel, est une transformation qui ajoute t à chaque élément de X et produit la série $X + t = (x_1 + t, \cdots, x_m + t)$. Si la distance D est invariante au choix de l'origine alors $D(X, Y) = D(X+t, Y+t)$. *MSM* est invariante au choix de l'origine parce que toute transformation S qui convertit X et Y convertit aussi $X + t$ en $Y + t$. Quant à la métrique, elle permet l'utilisation d'un large nombre d'outils pour indexer, regrouper et visualiser des séries dans des espaces vectoriels arbitraires.
Nous présentons un algorithme de complexité quadratique qui calcule la similarité entre deux séries. Soient $X = (x_1, \cdots, x_m)$ et $Y = (y_1, \cdots, y_n)$ deux séries temporelles. L'algorithme 1 décrit la méthode dynamique de calcul de la distance entre X et Y. Pour chaque couple (i, j) tel que $1 \leq i \leq m$ et $1 \leq j \leq n$, nous définissons $Cost(i, j)$ la distance *MSN* entre les i premiers éléments de X et les j premiers éléments de Y. Ainsi, la distance *MSN* entre X et Y est simplement $Cost(X, Y)$. Comme indiqué dans l'algorithme 1, $Cost(i, j)$ est calculé récursivement en se basant sur $Cost(i, j-1)$, $Cost(i-1, j)$ et $Cost(i-1, j-1)$.

$$Cost(i,j) = min \begin{cases} Cost(i-1, j-1) + |x_i - y_j|, \\ Cost(i-1, j) + C(x_i, x_{i-1}, y_j), \quad \text{avec} \\ Cost(i, j-1) + C(y_j, x_i, y_{j-1}) \end{cases}$$

$$C(x_i, x_{i-1}, y_j) = \begin{cases} c \ si \ x_{i-1} \leq x_i \leq y_j \ ou \ x_{i-1} \geq x_i \geq y_j \\ c + min(|x_i - x_{i-1}|, |x_i - y_j|) \ sinon \end{cases}$$

Algorithme 1 MSM(X,Y)

Require: série $X = (x_1, \cdots, x_m)$
Require: série $Y = (y_1, \cdots, y_n)$
$Cost(1,1) = |x_1 - y_1|$
for $i = 2, \cdots, m$ **do**
    $Cost(i,1) = Cost(i-1,1) + C(x_i, x_{i-1}, y_1)$
end for
for $j = 2, \cdots, n$ **do**
    $Cost(1,j) = Cost(1,j-1) + C(y_j, x_1, y_{j-1})$
end for
for $i = 2, \cdots, m$ **do**
    for $j = 2, \cdots, n$ **do**

$$Cost(i,j) = min \begin{cases} Cost(i-1,j-1) + |a_i - b_j|, \\ Cost(i-1,j) + C(a_i, a_{i-1}, b_j), \\ Cost(i,j-1) + C(b_j, a_i, b_{j-1}) \end{cases}$$

    end for
end for
return la distance MSM $D(X,Y)$ est $Cost(m,n)$.

3.2.1.4 Distance de Pearson

Dans la comparaison de séries temporelles, nous avons toujours constaté que la normalisation dans le calcul d'une distance euclidienne entre des séries donne de meilleurs résultats. Nous allons montrer la relation existante entre la distance euclienne et le coefficient de Pearson.
Soient deux séries temporelles A et B composées de T éléments $A = (a_1, \cdots, a_T)$ *et* $B = (b_1, \cdots, B_T)$. La distance euclidienne entre A et B est définie par l'équation 3.4

$$d_E = \sum_{t=1}^{T} (a_i - b_i)^2 \tag{3.4}$$

Elle est une métrique et les séries A et B sont identiques si cette distance est égale à 0. Pour les analyses de séries, il est recommandé de normaliser les séries pour éviter les variations d'échelles. En ce qui concerne le coefficient de Pearson, il mesure la corrélation ϱ entre deux variables aléatoires X et Y comme indiqué dans l'équation 3.5.

$$\varrho_{X,Y} = \frac{E[X - \mu_X][Y - \mu_Y]}{\sigma_X . \sigma_Y} \tag{3.5}$$

où μ_X est la moyenne et σ_X est l'écart-type de X. Le coefficient $|\varrho| = 1$ est égal à 1 si les variables X et Y sont parfaitement corrélées et 0 si X et Y sont non corrélées.
Dans le but d'utiliser le coefficient de Pearson comme une distance de séries temporelles, nous introduisons la *distance de Pearson* en générant de petites valeurs de distances pour des séries similaires. Elle est définie par l'équation 3.6.

$$d_P(A,B) = 1 - \varrho_{A,B} = 1 - \frac{\frac{1}{T}\sum_{t=1}^{T}(a_t - \mu_A)(b_t - \mu_B)}{\sigma_A.\sigma_B} \tag{3.6}$$

avec $0 \leq d_P(A,B) \leq 2$. Nous obtenons une parfaite correspondance ($d_P = 0$) pour les séries A et B s'il existe des nombres $\alpha, \beta \in \mathbb{R}$ avec $\beta > 0$ tels que $a_i = \alpha + \beta * b_i$. Nous exprimons d_E en fonction de d_P.

$$d_E(A,B) = \sum_{t=1}^{T}(a_i - b_i)^2 = \sum_{t=1}^{T}(a_i - 0)^2 - 2\sum_{t=1}^{T}(a_i.b_i) + \sum_{t=1}^{T}(b_i - 0)^2 \tag{3.7}$$

Les termes $\sum_{t=1}^{T}(a_i - 0)^2$ et $\sum_{t=1}^{T}(b_i - 0)^2$ correspondent à l'écart-type des séries A et B en supposant que la moyenne de ces séries est nulle $\mu_A = \mu_B = 0$ et leurs écarts-types sont égaux à 1 ($\sigma_A = \sigma_B = 1$). L'équation précédente 3.7 devient

$$d_E(A_{norm}, B_{norm}) = 2.T(1 - \frac{\frac{1}{T}\sum_{t=1}^{T}(a_{i,norm} - 0)(b_{i,norm} - 0)}{1.1}) = 2.T.d_P(A_{norm}, B_{norm})$$

Ainsi, la distance euclidienne de deux séries normées est égale à la distance de Pearson multipliée par un facteur constant $2T$. L'équivalence entre ces deux distances est pertinente parce que certains algorithmes utilisent la distance euclidienne pour faire de la classification (cas de k-means). Dans l'article [38], l'auteur utilise la classification *k-means* de séries avec la distance de Pearson et il en conclut que cette classification donne les mêmes résultats que le *k-means* avec la distance euclidienne.

3.2.1.5 Longest Common Subsequence

La distance *longest common subsequence (LCSS)* est basée sur la reconnaissance de motifs dans les problèmes *(LCSS)* . Dans ces problèmes, elle recherche la plus longue séquence qui est commune aux deux séries discrètes en utilisant la distance *edit distance (ED)*. Cette approche a été élargie aux séries continues en définissant la variable de seuil ϵ qui indique la différence entre une paire de valeurs. Cette différence détermine s'il existe une similarité entre ces séries. *LCSS* trouve un alignement optimal entre deux séries en insérant des écarts pour déterminer le nombre maximum de paires correspondantes. La distance *LCSS* entre deux séries A et B peut être calculée à partir de l'algorithme 2.

Algorithme 2 LCSS(A,B)

```
Soit L une matrice initialisée à 0 de dimension (m + 1) × (m + 1)
for i ← m a 1 do
    for j ← m a 1 do
        L_{i,j} = L_{i+1,j+1}
        if a_i = b_j then
            L_{i,j} ← L_{i,j} + 1
        else if L_{i,j+1} > L_{i,j} then
            L_{i,j} ← L_{i,j+1}
        else if L_{i+1,j} > L_{i,j} then
            L_{i,j} ← L_{i+1,j}
        end if
    end for
end for
return L_{1,1}
```

La distance *LCSS* entre les séries A et B est

$$d_{LCSS}(A, B) = 1 - \frac{LCSS(A, B)}{m}$$

Conclusion : plusieurs distances ont été présentées et leur point commun est l'utilisation de toute la série pour le calcul de la similarité. Parmi ces distances, nous distinguons les distances qui sont des métriques *TWE, MSM et la distance de Pearson* et celles qui ne le sont pas *LCSS et DTW*.

3.2.2 Similarité sur les séquences

Nous présentons, dans cette section, les distances qui transforment au préalable les séries pour comparer leurs caractéristiques. Ces caractéristiques sont appelées des *features*.
Pour mesurer la similarité entre des séries temporelles, des méthodes proposent de représenter les données en sous-séquences différentes, chacune formant une classe. Cette transformation est désignée par *shapelets*. Un shapelet considère que les sous-séquences sont indépendantes. Le calcul de similarité entre des séries se produit localement entre les séquences de même phase au moyen d'une métrique. Généralement la métrique utilisée est la distance euclidienne. D'abord, les shapelets généralisent l'algorithme des k plus proche voisins (*k-means*) largement utilisé dans les arbres de décision pour améliorer les classifications [39]. Ensuite, ils sont interprétables et donnent une idée de la différence entre deux classes [40]. Enfin, ils peuvent être aussi plus précis que d'autres méthodes concurrentes [41, 40].
Hills et al. [42] propose l'algorithme 3 de transformation de séries en *shapelets* en retournant les k

premiers shapelets dans une seule exécution. L'algorithme 3 se décrit comme suit : Soient w_i une sous-série d'une série temporelle A avec $i \leq k$ et W_l l'ensemble de taille l de séries w_i. La distance de *shapelet* $sDist(S,T)$ entre un shapelet S et une série T est la distance euclidienne minimum entre S et $w_i \in W_l$.

$$sDist(S,T) = min_{w_i \in W_l}(dist(S,w_i))$$

Le meilleur shapelet a une distance $sDist$ faible pour les instances d'une classe et des distances $sDist$ élevées pour les instances des autres classes.

Nous considérons w_i comme un *shapelet candidat*. L'ensemble de valeurs de $sDist$ pour chaque candidat est trouvé en utilisant la fonction *findDistances* et est évalué par la procédure *assessCandidate* au moyen de la mesure $f - statistic$. Les k meilleurs shapelets sont retournés après la suppression des sous-séries candidats par la fonction *removeSelfSimilar*. Nous nous servons de la procédure d'estimation de longueur, décrite dans [43], pour trouver les valeurs appropriées à utiliser comme les longueurs maximales et minimales de shapelets. Nous générons un maximum de $k = 10n$ shapelets où n est la taille de la série initiale.

Nous transformons la série initiale en utilisant les meilleurs shapelets comme des features où $sDist(S_i, T_j)$ désigne l'élément i dans l'instance j de la série transformée, S_i est le i^{ieme} shapelet et T_j est la j^{ieme} instance dans la série initiale. La complexité de l'algorithme est de $\mathcal{O}(n * m^2)$ avec n le nombre de séries temporelles et m la plus longue série temporelle [44].

Algorithme 3 ShapeletSelection(T,min,max,k)

```
1: kShapelets ← ∅
2: for all T_i in T do
3:     shapelets ← ∅
4:     for l ← min to max do
5:         W_{i,l} ← generateCandidates(T_i, l)
6:         for all subsequence S ∈ W_{i,l} do
7:             D_S ← findDistances(S, T)
8:             quality ← assessCandidate(S, D_S)
9:             shapelets.add(S, quality)
10:        end for
11:    end for
12:    sortByQuality(shapelets)
13:    removeSelfSimilar(shapelets)
14:    kShapelets ← merge(k, kShapelets, shapelets)
15: end for
16: return kShapelets.
```

Conclusion : les shapelets transforment la série en un sous-ensembles de séquences avec la fonction *generateCandidate*. Chaque séquence d'une série est nommée *shapelet candidat* et ce *shapelet candidat* est comparé avec les autres shapelets candidat de la même série à partir à la *distance de shapelet*. Une fois les *shapelets candidats* de chaque série sélectionnée avec l'algorithme *ShapeletSelection*, on les compare avec la distance euclidienne. Cette méthode est inefficace pour des séries de grande taille.

3.2.3 Similarité par aggrégation des caractéristiques descriptives

Les modèles de classification de séries temporelles basés sur des caractéristiques descriptives supposent d'extraire un ensemble de caractères qu'on espère être représentatif de la forme générale d'une série temporelle. Le plus communément, ces caractères sont quantifiées pour former des "sacs de mots" (BoW pour "Bag of Words"). Dans la récupération d'information, l'approche *BoW* d'estimation de la fréquence des mots en ignorant leur localisation est très commune. L'idée est d'estimer la fréquence d'occurences des caractères des séries puis d'utiliser ces fréquences comme des "features" pour faire de la classification.
Les approches suivantes diffèrent uniquement par les caractéristiques extraites. En effet, l'approche *Bag Of Pattern (BOP)* [45] convertit la série temporelle en une série discrète grâce à la méthode *Symbolic Aggregate approXimation (SAX)* [46]. Il crée un ensemble de mots *SAX* pour chaque série par l'application d'une fenêtre glissante, puis se sert de la fréquence des mots dans la série comme sa nouvelle caractéristique. *Baydoyan et al.* [47] décrit l'approche *bag-of-features* qui combine les caractéristiques de fréquences et d'intervalles. L'algorithme appelé *time series based on bag-of-features representation (TSBF)* implique la séparation entre la création de features et les étapes de classification. La création de features implique la génération d'intervalles aléatoires et les features représentent, généralement, la moyenne, la variance et la pente sur un intervalle. Le début et la fin d'un intervalle sont incluses dans les features.

3.2.4 Conclusion sur les méthodes de similarité

Dans cette partie, nous avons énuméré les différentes méthodes (distances) que nous pouvons utiliser pour calculer la similarité entre des séries. Nous avons catégorisé les distances en deux groupes : celles qui n'apportent aucune modification aux séries et celles qui transforment les séries avant de débuter l'analyse. La transformation des séries se fait de deux manières. La première consiste à diviser la série en séquences et à supposer que chaque séquence est indépendante des autres. Quant à la seconde, elle consiste à remplacer la série par certaines caractéristiques descriptives telles que la moyenne, l'écart-type, l'encodage de la série par des mots. Parmi celles qui ne modifient pas les séries, nous distinguons

certaines qui ont la propriété de métriques. Cette propriété est importante pour choisir la méthode de calcul de la similarité entre des séries.

3.3 Méthode de similarité entre des mesures électriques : distance de Pearson

3.3.1 Choix de la distance

Nos séries temporelles ont la particularité d'être des mesures électriques. Ces séries sont associées aux arcs entrants dans des équipements. Pour certaines grandeurs comme la puissance ou l'intensité, les mesures se propagent dans l'ensemble du réseau en suivant la loi de conservation [2]. Cela implique que toute variation de consommation électrique des équipements apparait dans les courbes des séries temporelles comme des pics. Cela signifie que la consommation en puissance d'un équipement baisse quand les équipements qu'il alimente sont à l'arrêt ou augmente quand ces équipements se mettent en marche. Nous désignons la courbe d'une série temporelle par le *profil de consommation* d'un arc. Ainsi, nous supposons que les arcs rattachés aux mêmes équipements ont les mêmes profils de consommation. Néammoins, des arcs appartenant à la chaine de propagation de l'électricité (c'est-à-dire tous les arcs par lesquels l'électricité transite) n'ont pas les mêmes profils car certains équipements sont rattachés à deux sources d'énergies et les pics s'attenuent durant la propagation. *La distance de similarité entre des paires de séries calcule le coefficient de similarité qui compare les profils de consommation des arcs*. Des arcs ont les mêmes profils si et seulement si le coefficient de similarité est le plus élevé (c'est-à-dire 1) et ont des profils différents si leur coefficient est le plus faible (c'est-à-dire 0).

Le choix de notre méthode de calcul de similarité dépend des données que nous avons collectées. En effet, dans ces données, certains arcs n'ont pas de mesures associées à des grandeurs physiques. Chaque valeur dans une série est la moyenne des valeurs sur un intervalle de 10 minutes. Il existe aussi des valeurs manquantes dans certaines séries de mesures. Nous avons attribué à ces valeurs manquantes la moyenne des valeurs à leur voisinage. Cette interpolation pose un problème dans le calcul de similarité avec les *shapelets*. En effet, étant donné le shapelet candidat contenant des valeurs extrapolées, la correspondance entre le shapelet candidat et toutes autres séquences est fortement dégradée à cause des valeurs extrapolées qui augmentent la distance euclidienne entre elles. Ensuite, il s'exécute lentement sur de grands ensembles de données. Enfin le shapelet candidat est de longueur quelconque et sa détermination passe par la génération de toutes les shapelets possibles. Ce qui conduit à une complexité de $\mathcal{O}(m^2)$, avec m la taille de la longue série temporelle. Cela rend le calcul irréalisable pour

notre ensemble de données de dimension $30 * 4320$ avec 30 le nombre de séries temporelles et 4320 le taille d'une série temporelle.

Par ailleurs, les méthodes par aggrégation des caractéristiques descriptives fournissent également des résultats mitigés. Prenons l'exemple de méthodes de similarité avec *Symbolic Aggregate approXimation*(SAX). Elle consiste à subdiviser chaque série en M séquences de taille identique puis à encoder chaque séquence par une lettre alphabétique, chaque lettre étant choisie dans un alphabet de lettres prédéfinies. La transformation d'une séquence en une lettre s'obtient grâce à une représentation *PAA (Piecewise Aggregate Approximation)* et à une table de correspondance entre l'alphabet. La représentation *PAA* [48] d'une séquence est la moyenne des valeurs de la séquence. La table de correspondance contient la liste ordonnée des points appelés *breakpoints* dont chaque valeur est une division arbritraire de la distribution gaussienne en zones équiprobables [49] et un alphabet. Et à chaque point de la liste *breakpoints*, une lettre lui est associée. La transformation de cette représentation en lettres a une complexité de $\mathcal{O}(mM)$ [49] avec M est le nombre de séquences de la série et m la taille de la série. Le principal inconvénient provient de l'erreur produite lors de transformation. L'encodage considéré par *SAX* est celui qui minimise cette erreur. Ainsi, une série peut avoir deux encodages différents si nous changeons l'origine des séquences. L'encodage n'est donc pas unique. De même, il est difficile de détecter les variations dans la série avec l'encodage *SAX* car toute variation faible mais continue dans le temps a le même encodage. Par contre, une variation forte dans la série ne modifie pas l'encodage dans le *breakpoint* puisque la distance *PAA* est la moyenne des valeurs dans la séquence.

Enfin, les méthodes de similarité, dont le résultat est le moins impacté par les valeurs extrapolées et les profils de consommation, sont celles qui utilisent l'intégralité de séries temporelles. À cet effet, la fenêtre glissante de 6 (correspondant à une heure) permet d'abord d'attribuer des valeurs aux instants t ayant des valeurs manquantes puis de supprimer des petites variations qui peuvent pénaliser nos similarités et enfin de mettre en évidence les fortes variations dans la série. Parmi les distances énumérées dans la section 3.2.1, nous avons décidé de choisir la *distance de Pearson* comme méthode de calcul du coefficient de similarité entre les séries temporelles pour les raisons suivantes :

- La distance de Pearson est une métrique alors que *DTW* ne l'est pas. Toutefois, ces distances ne respectent pas l'inégalité triangulaire.

- La classification par la méthode de *k-means* avec la distance euclidienne produit les mêmes résultats que celle avec la distance de Pearson [38]. *k-means* est une classification de réference dans les bases de données *UCR* [50]

- Elle est de complexité *linéaire* alors que la complexité de la distance *MSM* est quadratique.

- Elle ne traite pas de décalage entre les séries comme *TWE*. Le traitement du décalage entre les séries n'est pas nécessaire puisque les séries sont moyennées sur 10 minutes et les possibles valeurs décalées ont déjà été moyennées. De même, nous ne sommes pas parvenus à trouver une valeur de rigidité ν correcte pour *TWE* à cause des problèmes dans le dataset de *Champlan* (valeurs manquantes et incorrectes).

- La distance de Pearson nécessite que les séries soient normalisées. Les coefficients de Pearson désignent les similarités entre des paires de séries normalisées et ils appartiennent à l'intervalle $[0, 1]$. Si le coefficient est égal à 1 alors il existe une forte similarité entre les séries. Cependant, il n'existe pas de similarité si le coefficient est égal à 0.

Conclusion : La distance de Pearson est la mieux adaptée aux calculs des coefficients de similarité entre des séries parce qu'elle est une métrique, de complexité linéaire, détecte les variations simultanées (faibles et fortes) entre des paires de séries. Toutefois, elle ne respecte pas l'inégalité triangulaire et nécessite que les données soient normalisées. Elle est donc une bonne métrique pour comparer les profils de consommation.

3.3.2 Résultats sur des données réelles

Nous présentons les résultats obtenus avec la distance de Pearson.
Les grandeurs physiques collectées proviennent du *data center Champlan* et sont au nombre de 10. Elles sont regroupées dans les systèmes triphasé et monophasé. Dans les séries de mesures provenant des arcs en triphasé, nous remarquons qu'il y a toujours des mesures sur une phase et les autres phases ne contiennent aucune mesure. Il n'y a aucune mesure sur deux phases simultanément et lorsqu'il existe des mesures, les i premières mesures sont sur la phase 1, les $n - i$ mesures sont sur la phase 2. Les autres grandeurs ne contiennent des mesures soit dans les i premiers instants de temps, soit dans les $n - i$ derniers instants de temps. Il est alors difficile d'exploiter ces mesures partielles. Quant au système monophasé, 20% des mesures des puissances réactives Q et apparentes S dans une série temporelle sont manquantes, 25% des mesures ne correspondent pas aux formules de calculs théoriques. Le nombre de valeurs à considérer dans les séries des grandeurs n'est pas assez significative pour calculer les similarités parce que la distance de Pearson est sensible à la dimension de la série à cause de sa relation avec le coefficient de Pearson. Il est difficile de trouver de possibles comportements identiques à partir des hypothèses de corrélation avec des séries de grandes tailles et autant de valeurs absentes.
Nous considérons un sous-réseau électrique de *Champlan* dans lequel

- La grandeur sélectionnée possède des valeurs quelque soit le système (triphasé ou monophasé). La grandeur qui respecte cette règle est la *puissance* P.

- Les séries temporelles correspondent à un mois de mesures. Dans chaque série, nous avons en moyenne 15% de valeurs manquantes que nous remplacons par la valeur moyenne de chaque série.

- L'application de la loi des noeuds est possible.

- Aucun équipement n'est alimenté par un onduleur. La présence d'un onduleur modifie le profil de consommation d'un arc car l'onduleur recrée le signal en attribuant des puissances différentes. Cela implique que nous n'avons pas des mêmes variations.

Un exemple de ce sous-réseau est illustré dans la figure 3.4 dans lequel nous avons 31 équipements. Les sources sont identifiées par *TGBT* (TGBT1, TGBT2, TGBT4) et GF(1,2) désigne le groupe froid qui gère la climatisation. Les tableaux sont *DD205, DD206, DD105, DD106, DD108, MSC3, R486, R481, CVC1 et CVC2*. Les baies sont *R491, R488, R484A, R484B, R042, R483, R487, R492, R490, R493, R494*. Les onduleurs sont indiqués par *OND* (1,2,RG). Les équipements *TGBT* sont alimentés par une source externe au data center, le fournisseur d'électricité régional *Enedis*.

Nous allons calculer la corrélation entre les arcs du *sous-réseau de Champlan* en supposant que *deux arcs incidents à un équipement sont fortement corrélés et deux arcs non corrélés ne possèdent aucun équipement en commun*. Chaque arc est identifié par deux équipements, celui qui fournit l'électricité (x) et celui qui en consomme (y). Nous désignons par convention $x- > y$, l'arc entre x et y. Par exemple, l'arc entre $TGBT1$ et $R481$ est noté $TGBT1- > R481$.
Soient trois arcs $x- > y$, $y- > z$ et $t- > u$. Le coefficient de similarité $corr(x- > y, y- > z) = 1$ signifie que les arcs $x- > y$ et $y- > z$ ont les mêmes profils de consommation, partagent le sommet y et sont fortement corrélés. De même, le coefficient $corr(x- > y, t- > u) = 0$ indique que les arcs $x- > y$ et $t- > u$ ne sont pas corrélés c'est-à-dire qu'ils n'ont pas de sommet en commun et que leurs profils de consommation sont différents.

Nous disposons de la topologie électrique réelle de *Champlan*. Nous comparons les coefficients de similarité calculés par rapport aux arcs qui sont incidents dans la topologie de *Champlan*. Pour ce faire, nous présentons, dans la figure 3.5, les distributions des arcs incidents et non incidents en fonction des coefficients de similarité. Les arcs non incidents sont désignés par *cases_0* et les arcs incidents par *cases_1*. La distribution des arcs non incidents est asymétrique vers la gauche. Cela est normal puisque nous avons supposé que les arcs non incidents ont des coefficients de similarité qui sont proche de 0.

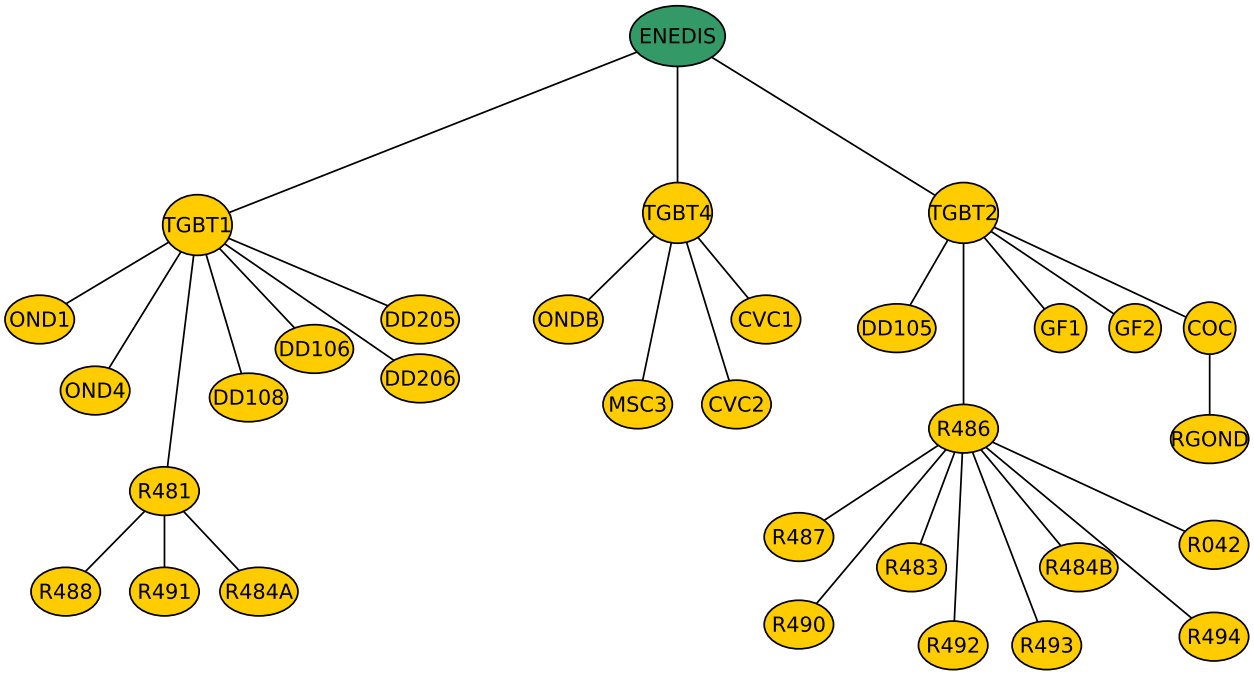

Figure 3.4: Le sous-réseau de Champlan étudié : Les sources sont *TGBT1, TGBT2, TGBT4*. *GF*(1,2) désigne le groupe froid qui gère la climatisation. Les tableaux sont *DD205, DD206, DD105, DD106, DD108, MSC3, R486, R481, CVC1 et CVC2*. Les baies sont *R491, R488, R484A, R484B, R042, R483, R487, R492, R490, R493, R494*. Les onduleurs sont indiqués par *OND1,OND2,RGOND*. Les équipements *TGBT* sont alimentés par une source externe au data center, le fournisseur d'électricité régional *Enedis*.

Cependant, nous avons quatre paires d'arcs qui ont des coefficients $corr(x- > y, y- > z) = 1$. La figure 3.6 présente les profils de consommation de ces quatre paires d'arcs. Nous constatons que :

- Les arcs $R486- > R487$ et $R481- > R488$ ont des profils opposés et en appliquant la valeur absolue sur les valeurs de chaque série, les profils se superposent. Il n'y a aucune corrélation entre cette paire d'arcs.
- Les arcs $TGBT1- > DD205$ et $TGBT4- > MSC3$ ont leurs profils qui se superposent et la corrélation de Pearson entre ces séries est alors nulle.
- Les paires d'arcs $TGBT2- > GF2$ et $TGBT1- > DD106$ ont des courbes qui ont plusieurs points d'intersection. À ces points, le coefficient de similarité est nul et est proche de 0 au voisinage de ces points. La corrélation de Pearson entre ces séries est alors nulle.

Cela implique que le coefficient de similarité est égal à $corr(x- > y, y- > z) = 1$ entre ces paires d'arcs $x- > y, y- > z$ par l'équation 3.6 alors que ces arcs n'ont aucun sommet en commun dans le graphe

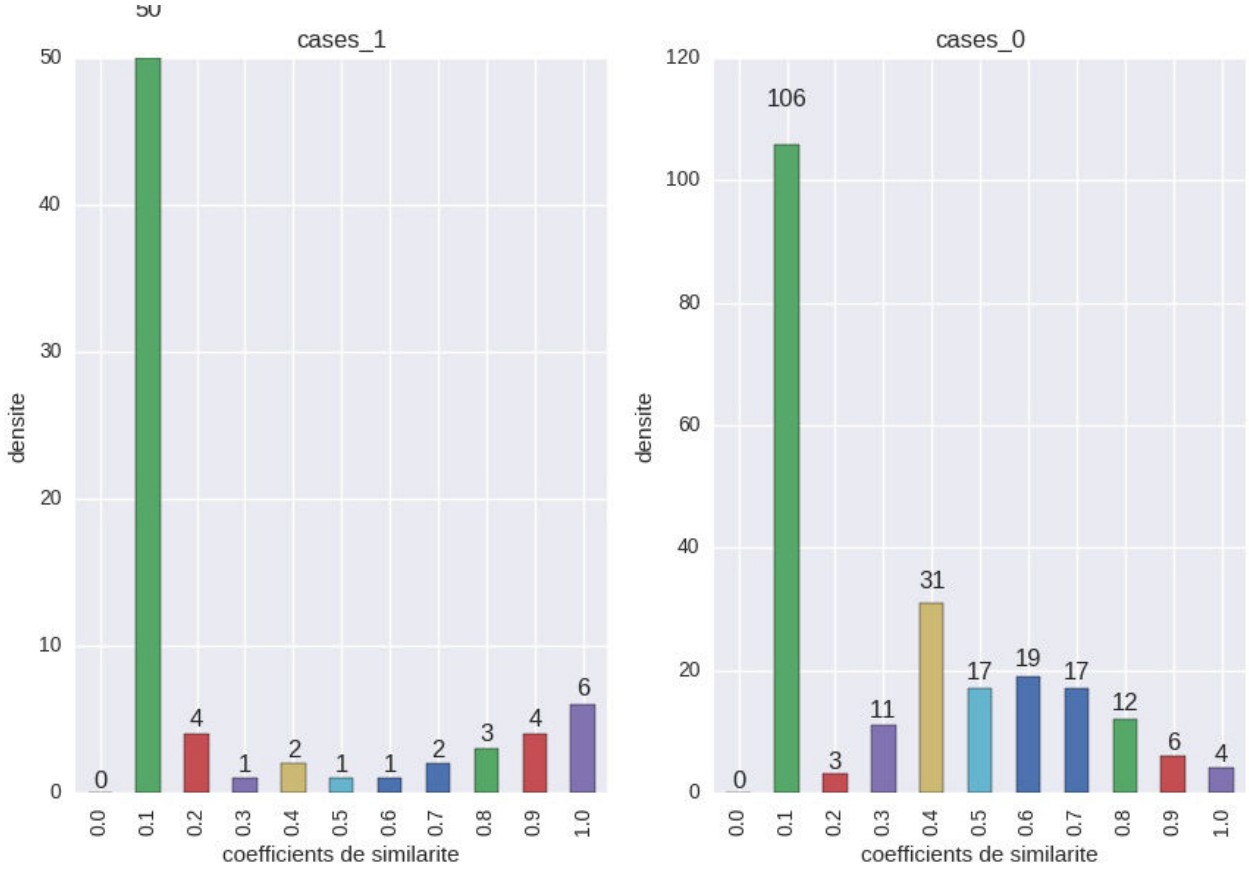

Figure 3.5: Distribution des coefficients des similarités selon les arcs qui sont incidents. *cases_1* désigne les arcs incidents et *cases_0* désigne les arcs non incidents dans le *sous-réseau de Champlan*. Le coefficient de similarité 0.1 indique toutes les valeurs dans l'intervalle $[0.1, 0.2[$.

de la figure 3.4. Ces erreurs de corrélation sont introduites par la méthode de calcul des coefficients.

En ce qui concerne la distribution des coefficients de similarité des arcs incidents (*cases_1*), elle est aussi asymétrique à gauche avec un pic pour les coefficients appartenant à l'intervalle $[0.1, 0.2[$. Pour comprendre cette distribution, nous représentons les profils de consommation de certaines paires d'arcs incidents ayant leur coefficient de similarité appartenant à l'intervalle $[0.1, 0.2[$ dans la figure 3.7. Dans ces représentations, il y a toujours une courbe constante et sa présence est due à l'impossibilité de collecter des données à cause d'une panne. Cette série contient alors des valeurs nulles. Il y a aussi la fourniture d'énergie dans les branches. En effet, la source ne fournit que la quantité d'énergie demandée par un équipement. Les séries des arcs sont différentes et la distance de Pearson est la plus faible ($corr(x- > y, y- > z) = 0.1$). Dans ce cas, les erreurs sont introduites par les données et le fonctionnement du réseau.

Dans les cas d'erreurs de similarité énoncés plus haut, nous ne pouvons pas les éviter pendant le cal-

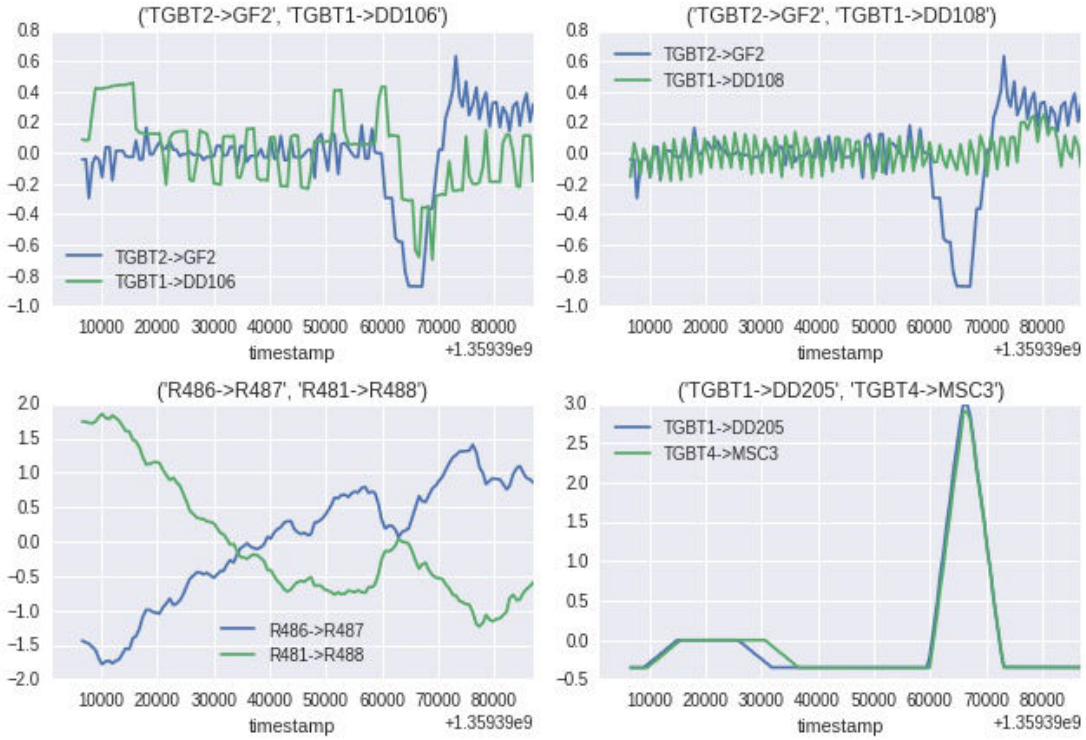

Figure 3.6: Profils de consommation des paires d'arcs n'ayant aucun équipement en commun. En haut à gauche, nous avons les courbes des arcs $TGBT2- > GF2$, $TGBT1- > DD106$. En haut à droite, les courbes des arcs $TGBT2- > GF2$, $TGBT1- > DD108$. En bas à gauche, les courbes des arcs $R486- > R487$, $R481- > R488$. En bas à droite, les courbes des arcs $TGBT1- > DD205$, $TGBT4- > MSC3$

cul des coefficients parce que le mécanisme de récupération des données est défaillant et l'arrêt de la consommation d'électricité d'une branche est masqué par la mise en service de plusieurs serveurs dans une baie appartenant à une autre branche.

Les coefficients de similarité sont regroupés dans une matrice symétrique carrée de dimension $N \times N$ avec N le nombre d'arcs dans le *sous-réseau de Champlan*. Les lignes et les colonnes de la matrice sont les arcs du sous-réseau. Chaque case de la matrice contient un coefficient de similarité entre deux arcs. Les cases de la diagonale de la matrice contiennent les coefficients d'un arc avec lui-même et sont égales à 0. Cette matrice est appelée *matrice de corrélation* et se note M_c. Sur cette matrice, différentes valeurs de seuils $s \in [0, 1]$ sont testées dans le but de déterminer la matrice d'adjacence $M_{c,s}$ du *sous-réseau de Champlan*. Des arcs $x- > y$ et $x- > z$ dont leur coefficient $corr(x- > y, x- > z) \geq s$ ont leur case $M_{c,s}[x- > y, x- > z] = 1$ sinon $M_{c,s}[x- > y, x- > z] = 0$ si $corr(x- > y, x- > z) < s$. La figure 3.8 présente les distributions des relations d'incidences entre les arcs après l'application des

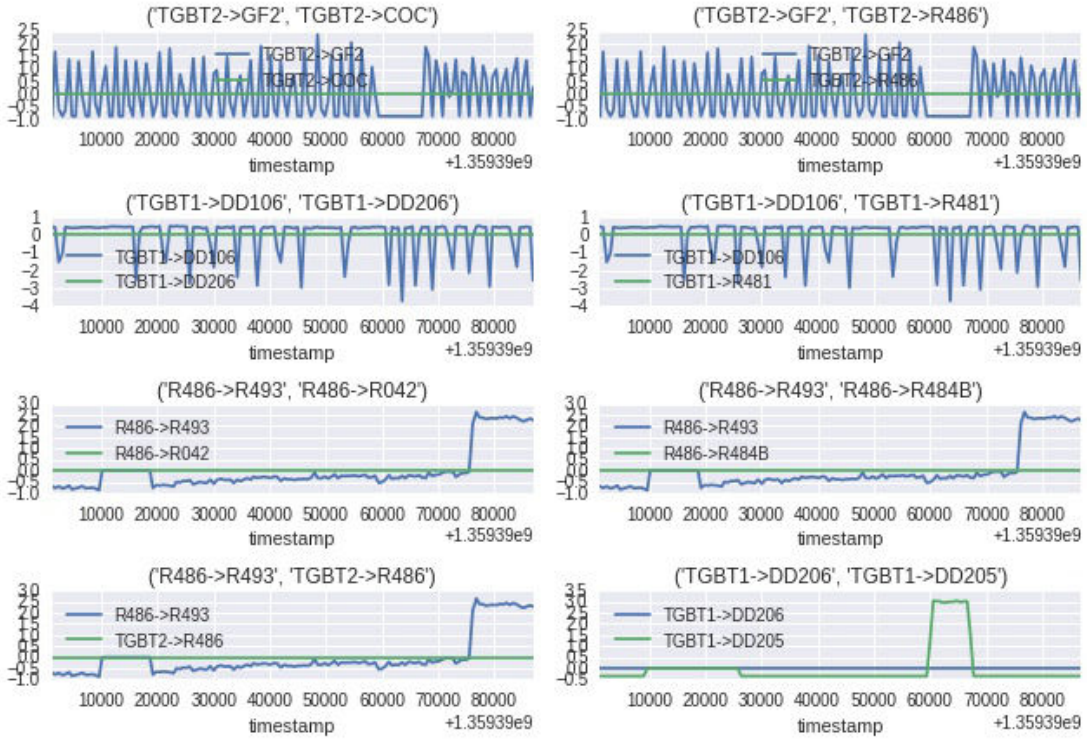

Figure 3.7: Profils de consommation des paires d'arcs ayant un équipement en commun. les arcs $TGBT2- > GF2$ et $TGBT2- > COC$ partagent l'équipement $TGBT2$.

seuils sur la matrice de corrélation. Pour un seuil donné, les relations d'incidences sont regroupées en 4 catégories :

- Les incidences *vraies positives* : il existe un sommet en commun entre les arcs et la case associée de la paire d'arcs $[x- > y, x- > z]$ est $M_{c,s}(x- > y, x- > z) = 1$.
- Les incidences *vraies négatives* : il n'existe aucun sommet en commun entre les arcs et la case associée de la paire d'arcs $[x- > y, x- > z]$ est $M_{c,s}(x- > y, x- > z) = 0$.
- Les incidences *fausses positives* : il n'existe aucun sommet en commun entre les arcs et la case associée de la paire d'arcs $[x- > y, x- > z]$ est $M_{c,s}(x- > y, x- > z) = 1$.
- Les incidences *fausses négatives* : il existe un sommet en commun entre les arcs et la case associée de la paire d'arcs $[x- > y, x- > z]$ est $M_{c,s}(x- > y, x- > z) = 0$.

Les incidences *fausses positives* et *fausses négatives* constituent des *erreurs d'incidences* dans la matrice d'adjacence $M_{c,s}$. Nous considérons aussi certaines incidences *vraies positives* et *vraies négatives* comme les *erreurs d'incidences*. Chaque graphique affiche les erreurs d'incidences associées à un seuil. Par

exemple, dans le graphique *VraiPositive_ErreurAdjacence* de la figure 3.8 nous avons 17 paires d'arcs partageant un sommet pour un seuil $s = 0.4$. De même, nous avons 57 paires d'arcs n'ayant aucun sommet en commun pour un seuil $s = 0.4$ dans le graphique *fauxNegatives_ErreurAdjacence* de la figure 3.8. Nous observons qu'il n'existe aucun seuil qui maximise les nombres d'incidences *vraies positives* et *vraies négatives* et qui minimise les nombres d'incidences *fausses positives* et *fausses négatives*. Et cela est due à l'introduction des erreurs des coefficients de similarité dans la *matrice de corrélation* $matE$.

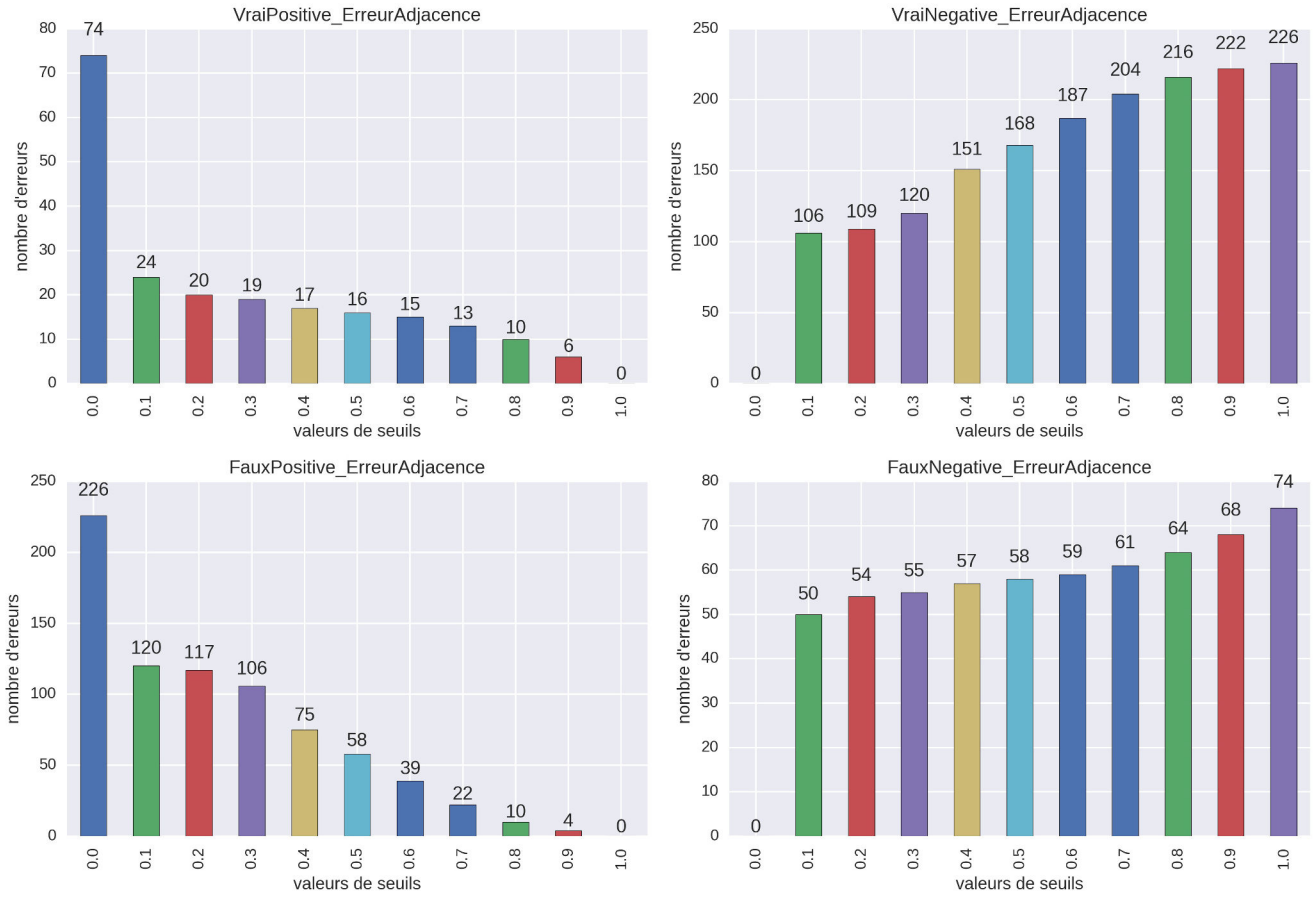

Figure 3.8: Distributions des relations d'incidences entre les arcs après l'application de seuils. On distingue 4 relations d'incidences entre les arcs : incidences *fausses positives* (graphique en bas à gauche),*fausses négatives* (graphique en bas à droite), *vraies positives* (graphique en haut à gauche) et *vraies négatives* (graphique en haut à droite).

Conclusion : dans cette section, nous avons limité notre étude sur un *sous-réseau de Champlan* dans lequel les équipements ne sont pas alimentés par un onduleur. Ce choix fut préféré à cause de notre hypothèse qui stipule que les variations d'électricité se propagent dans le réseau. Nous avons calculé les coefficients de similarité *corr* avec la grandeur P parce que c'est la seule grandeur qui fournit des

mesures en monophasé et en triphasée. Certains coefficients de similarité sont erronés à cause des données, de la méthode de calcul et du mécanisme de fonctionnement du réseau de Champlan. Ces coefficients forment la *matrice de corrélation* M_c carrée et symétrique dans laquelle sont testés des seuils. Si M_c ne contient aucune erreur de similarité alors il existe un seuil qui détermine la matrice d'adjacence de la topologie du sous-réseau de Champlan. Malheureusement, nous ne sommes pas parvenus à déterminer la bonne valeur de seuil et aussi à obtenir les coefficients de similarité qui reflètent les relations d'adjacences des arcs.

3.4 Conclusion du chapitre 3

Ce chapitre se subdivise en quatres parties. La première partie présente les domaines dans lesquels l'analyse des séries temporelles est importante. Ensuite, nous exposons notre problème qui consiste à comparer deux séries temporelles en supposant que les variations dans une série sont reproduites dans l'autre série. Pour résoudre notre problème, nous décidons de nous servir des méthodes de classification de séries temporelles. Dans la seconde partie, nous détaillons les méthodes qui se regroupent en trois catégories : *similarité sur les séries entières, similarité sur les parties significatives, similarité par aggrégation des caractéristiques descriptives*. Chaque catégorie décrit des distances de similarité. Les avantages et les inconvénents de chaque catégorie sont décrites dans la troisième partie. Ainsi, en analysant nos données et en se basant sur notre hypothèse, nous avons montré que les méthodes sur les séries entières sont adaptées à notre problème. En effet, notre hypothèse stipule que deux arcs partageant un équipement ont les mêmes profils de consommation et leur coefficient de similarité est proche de 1. Dans le cas contraire, leur coefficient de similarité tend vers 0 et les profils de consommation ont des courbes différentes. Nous avons alors choisi la *distance de Pearson* comme la méthode de similarité parce qu'elle est une métrique, de complexité linéaire, ne traite pas le décalage temporel et enfin retourne des valeurs appartenant à l'intervalle $[0, 1]$. Nous avons appliqué cette distance sur un *sous-réseau du data center Champlan* parce que ce sous-réseau ne possède aucun équipement alimenté par un onduleur. Ensuite, la seule grandeur qui contient des valeurs en monophasé et triphasé est la grandeur P. Les coefficients de similarité entre les arcs obtenus avec la grandeur P contiennent des erreurs de similarité. Une erreur de similarité est un coefficient proche de 1 alors que les arcs ne concourent pas en un équipement et vice-versa. Ces coefficients forment la *matrice de corrélation* qui appliquée à un seuil propose la matrice d'adjacence du sous-réseau de Champlan. Ceci est vérifié à condition que les coefficients ne contiennent aucune erreur. Enfin, nous avons montré qu'il est difficile de déterminer la bonne valeur de seuil en présence des coefficients de similarité erronés.

Chapitre 4

Line-graphes

Dans le chapitre précédent, nous avons déterminé la matrice de corrélation M_c du graphe G d'un réseau électrique. Cette matrice peut contenir des cases erronées. Une case erronée $M_c[i, j]$ est un coefficient de corrélation proche de 1 (resp. de 0) entre les arcs i et j alors que ces arcs ne partagent aucune extrémité (resp. ces arcs ont une extrémité commune).
Nous considérons une matrice M de dimension identique à celle de M_c telle que, pour toute valeur de seuil $s \in [0, 1]$ et toute paire d'arcs (i, j), $M[i, j] = 1$ si et seulement si $M_c[i, j] \geq s$. La matrice M est la matrice d'adjacence d'un graphe non orienté G_c dit *graphe de corrélation*. Cette matrice peut également contenir des cases erronées. Une case erronée $M[i, j] = 1$ désigne la présence d'arêtes dans G_c alors qu'il n'existe aucune arête entre les sommets i et j dans le line-graphe du graphe non-orienté sous-jacent au DAG G. De même, l'absence d'une arête entre i et j dans G_c alors qu'elle est présente dans le line-grahe du graphe non-orienté sous-jacent à G est aussi une case erronée $M[i, j] = 0$.
S'il n'existe aucune case erronée dans la matrice M, alors G_c est le line-graphe de graphe non-orienté sous-jacent au DAG G et le line-graphe de G est isomorphe à G_c. Notre but est de déduire le DAG G à partir de G_c en deux étapes :

- Déterminer si G_c est un line-graphe. Si c'est le cas, déduire le graphe dont G_c est le line-graphe.
- Dans le cas où G_c n'est pas un line-graphe, proposer un algorithme qui modifie G_c de telle sorte qu'il devient un line-graphe et ensuite déduire le graphe dont le graphe G_c modifié est le line-graphe.

Nous désignons ce problème par *Proxi-Line*.
Nous allons, dans un premier temps, décrire les caractéristiques d'un line-graphe et le problème *Proxi-Line*. Ensuite nous présentons les algorithmes qui traitent ce problème et enfin nous expliquons la reconstruction de la topologie à partir du line-graphe découvert par nos algorithmes.

4.1 Line-graphes : caractéristiques et propriétés

Dans la théorie des graphes, un line-graphe est aussi appelé un *graphe adjoint* et ce terme est introduit par l'article de Harary et Norman [51]. Le line-graphe d'un graphe non orienté G est un graphe qui représente la relation d'incidence entre les arêtes de G. Nous allons définir formellement le line-graphe d'un graphe et présenter ses propriétés.

4.1.1 Caractéristiques d'un line-graphe

Définition 3. *Soit $G = (V, E)$ un graphe non orienté.*
Le line-graphe de G est un graphe non-orienté $L(G) = (V', E')$ dans lequel $V' = E$ et une paire $[a, a']$ de sommets de $L(G)$ est une arête si et seulement si a et a' ont une extrémité commune dans G.
Le graphe G est appelé le graphe racine *de $L(G)$.*

Si G est un DAG alors $L(G)$ est le line-graphe du graphe non-orienté sous-jacent à G. Étant donné que G a n sommets et m arcs (sans arcs symétriques), le graphe $L(G)$ a m sommets et $|E'| = \sum_{i=1}^{n} d_i(d_i - 1)/2$ arêtes avec d_i le degré de chaque sommet i de G.

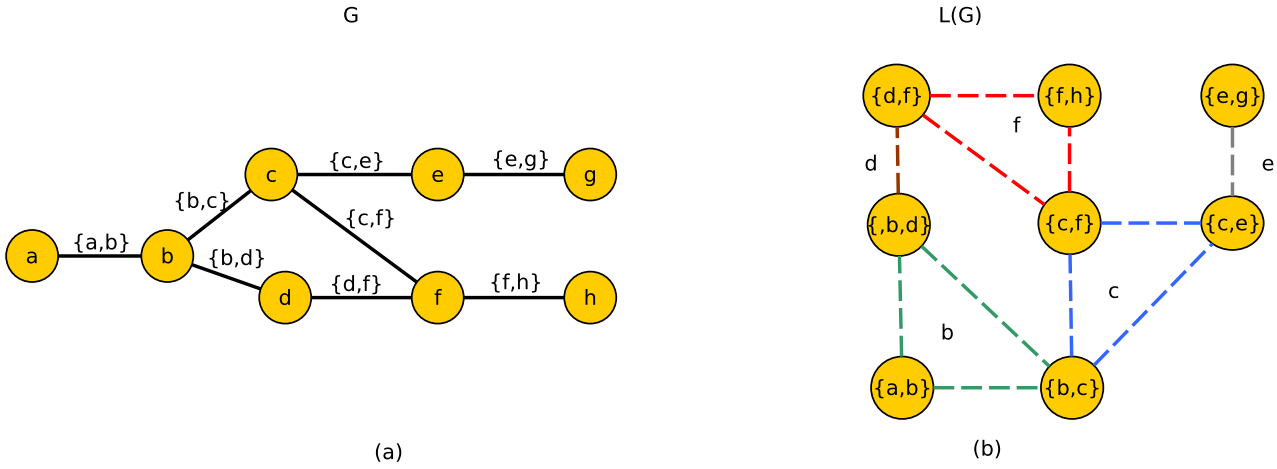

Figure 4.1: Le graphe G et son line-graphe $L(G)$.

La figure 4.1(a) présente le graphe $G = (V, E)$ dans lequel l'ensemble V contient 8 sommets $V = \{a, b, c, d, e, f, g, h\}$ et l'ensemble E contient 8 arêtes
$E = \{\{a, b\}, \{b, c\}, \{b, d\}, \{c, f\}, \{d, f\}, \{f, h\}, \{c, e\}, \{e, g\}\}$. Chaque arête de E devient un sommet de $L(G)$ dans la figure 4.1(b). Lorsque deux arêtes de E ont une extrémité commune alors leurs sommets respectifs dans $L(G)$ sont adjacents. Par exemple, dans la figure 4.1(b), les sommets $\{b, d\}$ et $\{d, f\}$ sont liés par une arête à cause du sommet $d \in V$. Nous construisons ainsi le graphe $L(G)$ qui contient

8 sommets et 11 arêtes. Nous constatons qu'un sommet de G correspond à une clique dans $L(G)$. En effet, les sommets de la clique $\{\{a,b\},\{b,c\},\{b,d\}\}$ de taille 3 dans $L(G)$ concourent à un point b de degré 3 qui est un sommet de $G[V]$. Le sommet b de G identifie la clique $\{\{a,b\},\{b,c\},\{b,d\}\}$ dans $L(G)$. Le graphe $L(G)$ est le line-graphe de G et G est le *graphe racine*.

La notion de *line-graphe* a été introduite par *Whitney* [52] en se basant sur la notion d'isomorphisme. Il montre que si deux line-graphes sont isomorphes et connexes alors leurs graphes racines sont aussi isomorphes à l'exception des graphes triangle K_3 et étoile $K_{1,3}$.

Proposition 1. *[53] Le graphe étoile $K_{1,3}$ n'est pas un* line-graphe.

Preuve 1. Supposons que $K_{1,3}$ est le line-graphe de H ($K_{1,3} = L(H)$). Alors H est un graphe connexe de quatre arêtes. Tous les graphes connexes de quatres arêtes sont représentés dans la figure 4.2. Comme $L(C_4) = C_4$ et $L(K_{1,3} + e) = K_4 + e$ (voir figure 4.2), $L(H)$ ne peut être que l'un des trois arbres P_4, $K_{3,2}$ et $K_{1,4}$. Ce qui est contraire à notre hypothèse de départ ($K_{1,3} = L(H)$).

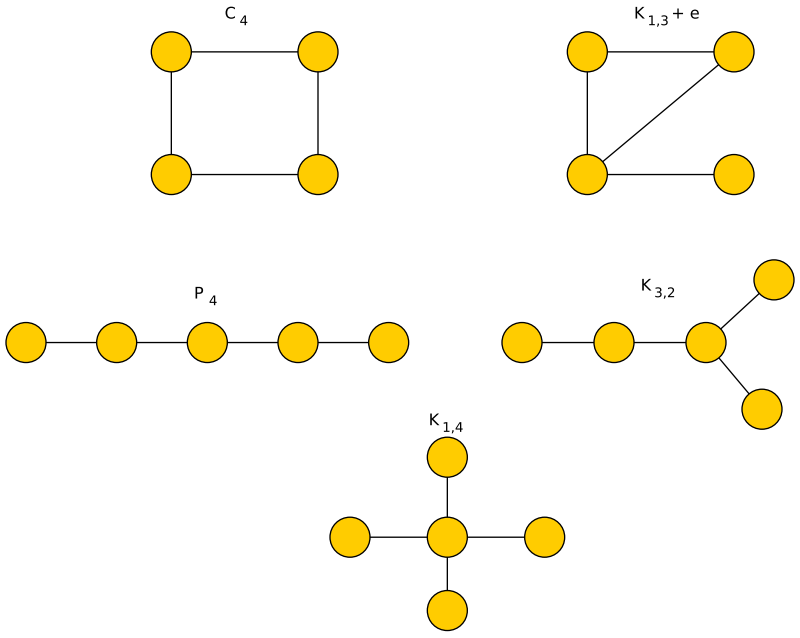

Figure 4.2: Les graphes racines possibles de $K_{1,3}$ de quatres arêtes.

Le graphe étoile ($K_{1,3}$) a un rôle important dans la caractérisation des line-graphes. La première caractéristique provient des travaux de *Krausz* [54] et elle est relative au partitionnement du line-graphe en sous-graphes. La seconde caractéristique, formulée par *Van Rooij et Wilf* [55], décrit la structure de base d'un graphe pour être un line-graphe. Et enfin, la dernière caractéristique présentée

par *Beineke[56] et Hemminger* a déterminé les neufs sous-graphes ne pouvant pas être les sous-graphes induits de line-graphes.

Théorème 2. *[53] Soit H un graphe. Les affirmations suivantes sont équivalentes.*

(a) H est un line-graphe.

(b) Les arêtes de H peuvent être partitionnées en sous-graphes complets appelés cliques *tel qu'aucun sommet n'est contenu dans plus de deux sous-graphes.*

(c) H ne contient pas $K_{1,3}$ comme sous-graphe et si deux triangles ont une arête commune alors le sous-graphe induit est K_4.

(d) Aucun des neufs sous-graphes de la figure 4.3 ne peut être un sous-graphe du line-graphe H.

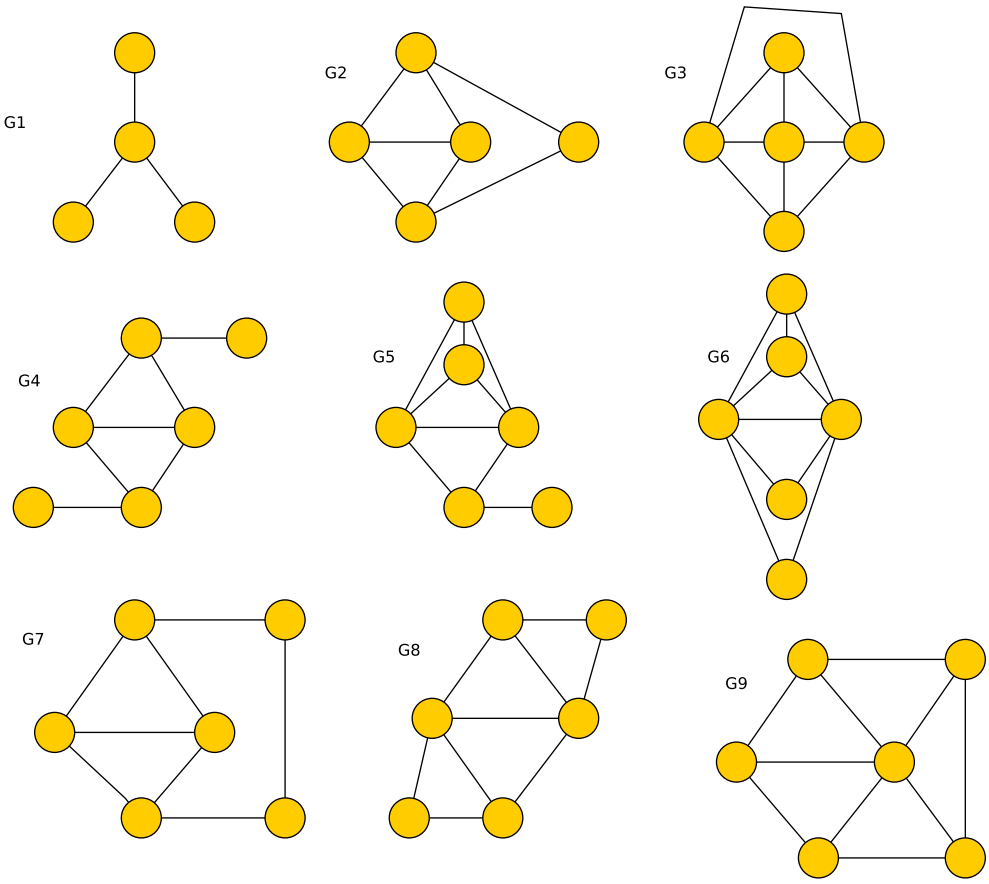

Figure 4.3: Les 9 sous-graphes interdits dans un line-graphe.

Conclusion : soit H un graphe et G un graphe non-orienté. Le graphe H est un line-graphe de G si le théorème 2 est respecté. Les graphes H et $L(G)$ sont isomorphes et le graphe racine de H est $L^{-1}(H)$.

Si H est un line-graphe d'un graphe non orienté G, alors le graphe H admet un partitionnement en cliques et chaque clique correspond à un sommet de G. L'ensemble de cliques est appelé une *couverture de corrélation* et est noté $\mathcal{CC}(G)$.

4.1.2 Line-graphes ambigus

Soient G un graphe non orienté et H l'unique line-graphe de G. D'après le théorème 2(b), les arêtes de H se partitionnent en cliques telles que chaque clique correspond à un sommet de G.
Si le graphe de correction G_c est sans erreur, alors il est un line-graphe et son partitionnement en cliques forme une *couverture de corrélation* notée $\mathcal{CC}(G_c)$. Il a été montré qu'un line-graphe a un unique graphe racine à isomorphisme près [52] à l'exception du graphe K_3. Cet isomorphisme conduit à plusieurs *couvertures de corrélation* pour un même line-graphe. Il est alors nécessaire de déterminer quels line-graphes admettent différentes *couvertures de corrélation*. En d'autres termes, le line-graphe G_c a-t-il plusieurs graphes racines qui sont isomorphes? Pour répondre à cette question, nous définissons la notion d'*ambiguïté*.

Définition 4. *Soient G un graphe non orienté et $L(G)$ le line-graphe de G.*
Une ambiguïté dans $L(G)$ est un sous-graphe isomorphe à l'un des graphes de la figure 4.4. Le sommet X est appelé le **point d'ambiguïté**.

Il a été montré que deux line-graphes isomorphes ont leurs graphes racine isomorphes à l'exception du graphe triangle [52]. Les graphes de la figure 4.4 sont isomorphes mais leurs graphes racines ne sont pas isomorphes. Cela implique que leurs couvertures de corrélation sont différentes. D'où la présence de sommets *ambigus*.

Lemme 3. *Soient $G = (V, E)$ un line-graphe et u un sommet de G.*
Si G admet deux couvertures de corrélation, alors il existe au moins un sommet u de G tel que $G[\{u\} \cup \Gamma_G(u)]$ est une ambiguïté dans laquelle u est le point ambiguïté.

Preuve 2. Considérons deux couvertures de corrélation $\mathcal{CC}(G)$ et $\mathcal{CC}'(G)$ de G. Il existe au moins un sommet $v \in V[G]$ qui n'est pas couvert par la (ou les) même(s) clique(s) dans $\mathcal{CC}(G)$ et $\mathcal{CC}'(G)$. Soient deux cliques c_1 et c_2 (potentiellement vide) partitionnant $\{v\} \cup \Gamma_G(v)$ dans $\mathcal{CC}(G)$. Considérons deux autres cliques c_3 et c_4 différentes de c_1 et c_2 partitionnant également $\{v\} \cup \Gamma_G(v)$ dans $\mathcal{CC}(G)$.
Notons $c_{i,j}$ l'intersection de c_i et c_j pour tout $i \in \{1, 2\}$ et $j \in \{3, 4\}$. Chaque sommet $w \in c_{i,j}$ est couvert par au plus deux cliques de G dans $\mathcal{CC}(G)$, dont la clique c_i. Puisque c_j est une clique alors ce sommet w est voisin de tous les sommets de $c_{i',j}$, pour $i' \neq i$. Les arêtes entre ces sommets sont dans

c'_i, donc chaque arête $[w, z]$ pour tout sommet $z \in c_{i',j}$ forme une clique dans le réseau de flots. Ainsi, le cardinal de chaque ensemble $c_{i,j}$ est égal à 1.
Appelons $v_{i,j}$ le seul sommet présent dans $c_{i,j}$. Il est possible d'avoir $v_{1,3} = v_{1,4}$ ou $v_{2,3} = v_{2,4}$. Si les deux égalités sont vérifiées, le sommet v est alors couvert non pas par deux cliques mais par une seule de cardinalité 3. Ainsi, les seuls cas possibles sont alors résumés par la figure 4.5. Le sommet v est bien le point d'une ambiguïté isomorphe à $G[\{u\} \cup \Gamma_G(u)]$. □

Nous déduisons le corollaire suivant :

Corollaire 4. *Soit G un line-graphe.*
Si G admet deux couvertures de corrélation différentes, alors il est isomorphe à l'un des graphes de la figure 4.5.

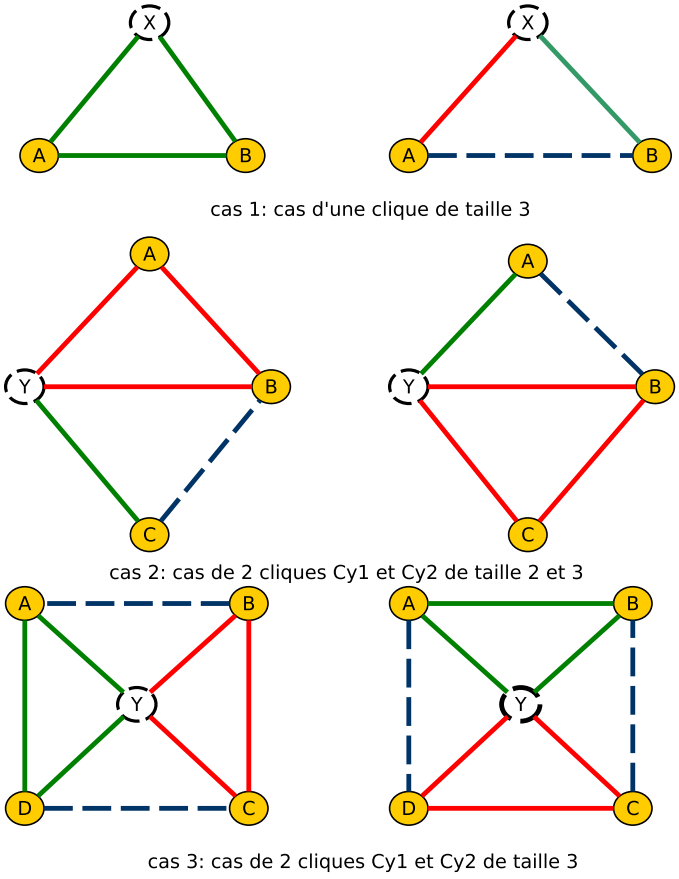

Figure 4.4: Configurations possibles d'une ambiguïté au sommet X.

En effet, si $G[\{u\} \cup \Gamma_G(u)]$ a une ambiguïté, chaque arête, qui n'est pas liée au point d'ambiguïté, doit être une arête d'une et une seule autre ambiguïté de G. Et chaque sommet d'une ambiguïté, qui

n'est pas un point d'ambiguïté, doit appartenir à une et une seule autre ambiguïté de G dont il n'est pas non plus le point d'ambiguïté. De plus, chaque arête, n'étant pas couverte par les deux configurations de cliques possibles dans une ambiguïté (les arêtes en pointillées dans la figure 4.4), doit être dans une autre ambiguïté à laquelle elle appartient. Ces contraintes font que si un graphe contient une ambiguïté induite, alors il ne peut être que dans un cas de la figure 4.5.

Définition 5. *Soient G un graphe, u un sommet de G et $\Gamma_G(u)$ les sommets voisins de u.*
Une partition de $\Gamma_G(u)$ en deux cliques C_{u1}, C_{u2} est **cohérente** *si et seulement si chaque sommet v de C_{u1} (resp. C_{u2}) a au plus un voisin dans C_{u2} (resp. C_{u1}).*

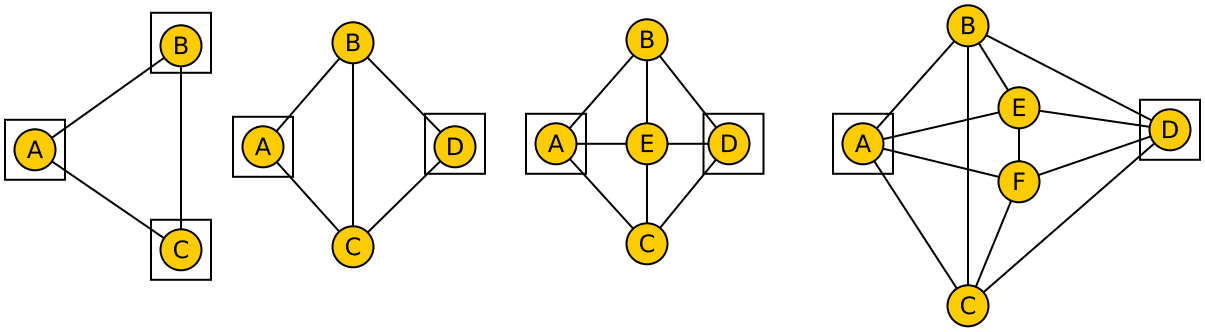

Figure 4.5: Les graphes possibles de deux couvertures de corrélation avec les points d'ambiguïtés encadrés.

Conclusion : nous avons montré que la *couverture de corrélation* d'un line-graphe est unique à l'exception des situations ambiguïtés. Les cas d'ambiguïtés sont présentés dans la figure 4.5.

4.2 Formulation du Problème *Proxi-Line*

Soient G un graphe non orienté d'un DAG, G_c un graphe de corrélation de G et M la matrice d'adjacence de G_c.
Notre problème est de déterminer G à partir de G_c. Pour ce faire, nous décidons de nous servir de la *couverture de corrélation*. On a trois cas :

- Soit G_c est isomorphe à $L(G)$. Nous trouverons la couverture de corrélation unique qui donne G.
- Soit G_c est un line-graphe non isomorphe à $L(G)$. Modifier la matrice d'un line-graphe peut en effet le transformer en un autre line-graphe. Ce cas arrive rarement notamment lorsqu'il y a peu d'arêtes erronées dans G_c.

- Soit G_c n'est pas un line-graphe. Dans ce cas, l'idée est de corriger G_c en ajoutant ou supprimant le minimum d'arêtes.

Nous resolvons le 3^{ieme} cas en introduisant le problème suivant :

4.2.1 Problème

Étant donné un graphe G' qui a des arêtes en plus ou en moins par rapport à un autre graphe G de même ensemble de sommets.

Définition 6. *Soient G et G' deux graphes non orientés ayant le même ensemble de sommets.*
La distance de Hamming entre les graphes G et G' notée $DH(G, G')$ est le nombre d'arêtes présentes dans G et pas G' et inversement.

Une distance de Hamming égale à k ($k \in \mathbb{N}$) signifie qu'il existe k cases différentes entre les matrices d'adjacence des graphes G et G'.

Définition 7. *Soit G un graphe non orienté.*
On appelle distance line *de G, notée $DL(G)$, la plus petite distance de Hamming entre G et G', un line-graphe ayant le même ensemble de sommets que G.*

Nous considérons le problème suivant.
Problème Proxi-Line
Données : Un graphe $G = (V, E)$, un entier k.
Question : $DL(G) \leq k$?

Conjecture 5. *Proxi-Line est NP-complet.*

Ce problème généralise le problème *NP-complet* défini et montré dans [57], c'est-à-dire étant donnés un graphe G et un entier k, le problème de savoir s'il existe un line-graphe G' qui est un sous-graphe couvrant de G tel que $dH(G, G') \leq k$ (c'est-à-dire, le problème Proxi-Line dans lequel seule la suppression d'arêtes est autorisée) utilise une solution de programmation linéaire en nombres entiers dans [58]. Récemment, il a été montré au sein du laboratoire DAVID que l'opération d'ajout d'arêtes uniquement dans le graphe de corrélation est aussi *NP-complet*.

4.3 Algorithmes de découverte de topologie

Le problème considéré ici est, étant donné un graphe, de déterminer s'il est un line-graphe et dans ce cas donner le graphe racine. Nous débutons par l'état de l'art des algorithmes de couverture en cliques puis présentons nos algorithmes tout en spécifiant leurs particularités par rapport aux méthodes existantes.

4.3.1 Recherche de couverture en cliques

Différents travaux ont été réalisés sur la découverte de *couverture en cliques* dans les line-graphes. Parmi lesquelles, nous citons l'algorithme de *Roussopoulos* [59] qui utilise une propriété des line-graphes provenant des travaux de *Krausz* [54]. Il affirme que le graphe G est un line-graphe si ses arêtes peuvent être partitionnées en cliques de telle sorte qu'aucun sommet ne soit couvert par plus de deux cliques. L'algorithme proposé détecte si G est un line-graphe et il fournit, en plus, son graphe racine en temps linéaire $O(max(\{m, n\}))$, avec n et m les nombres respectifs de sommets et d'arêtes.
Un autre algorithme, proposé par *Klauss Simon* et *Daniele Degiorgi* [60], est une simplication du problème de reconnaissance de line-graphes. Basé sur la preuve de *Oystein Ore* du théorème de *Whitney* [52], il stipule que deux line-graphes connexes avec plus de quatre sommets sont isomorphes si et seulement si leurs graphes racines sont aussi isomorphes et que ces graphes doivent être différents de $K_{1,3}$ et K_3. Il détermine en un temps linéaire une couverture étant mise à jour sommet après sommet. L'inconvénient de cette méthode est le traitement de sommets appartenant à des cliques ayant déjà été découverts parce qu'il applique le partitionnement sur la liste des sommets obtenus.
L'algorithme de *Philippe Lehot* [61] a une complexité en $O(n) + m$ avec n le nombre de sommets de G et m le nombre d'arêtes de $L(G)$. Il recherche les 9 sous-graphes de la figure 4.3 et il utilise le théorème de *Van Rooij et Wilf* [55] qui énonce qu'un graphe G est un line-graphe si G ne contient pas de sous-graphe induit $K_{1,3}$ et si deux graphes triangles *impairs* ont une arête commune, alors le sous-graphe induit par ces sommets est une clique K_4. Rappelons qu'un graphe triangle (c'est-à-dire un cycle de longueur 3) $\{a_1, a_2, a_3\} \subseteq V(L(G))$ est *impair* s'il existe un sommet $e \in V(G)$ adjacent à au moins un des sommets $\{a_1, a_2, a_3\}$. Ce triangle est *pair* si chaque sommet de ce triangle est adjacent à 0 ou 2 autres sommets. Cet algorithme est détaillé dans la section 4.3.2.
Tous les algorithmes existants ne retournent aucun résultat lorsque le graphe de corrélation G_c possède des cases erronées c'est-à-dire qu'il n'est pas un line-graphe. Cependant, la méthode proposée par *Halldórsson and al.* [58] utilise un algorithme génétique pour corriger un graphe de corrélation pour en obtenir un line-graphe. En effet, il propose une méthode de découverte de la généalogie de population en se basant sur les haplotypes partagés dans les génomes des individus. Un haplotype est un groupe d'allèles dans un organisme qui est transmis ensemble par un parent. Il modélise un graphe dit *Clark*

Consistency [62] dans lequel les arêtes sont les haplotypes (ils sont uniques dans les génomes) et les sommets sont les individus. Cette méthode recherche le graphe racine induit par le graphe *Clark Consistency* si celui-ci est un line-graphe. Dans le cas où le graphe *Clark Consistency graph* n'est pas un line-graphe, l'algorithme suppose qu'il existe des sommets en plus dans le *graphe Clark Consistency*, va les supprimer afin que le graphe devienne un line-graphe et enfin retourner le graphe racine. Le graphe *Clark Consistency (CC)* a été proposé dans la méthode d'identification d'haplotypes par *Andrew Clark*. En effet, *Andrew Clark* considère un ensemble de génomes d'individus qui ont des haplotypes homozygotes et hétérozygotes. Il suppose que deux génomes n'ont pas les mêmes paires d'haplotypes. Les sommets du graphe CC sont les génomes des individus et une arête de graphe CC entre deux génomes existe s'ils partagent le même haplotype (homozygote ou hétérozygote). Les arêtes de ce graphe sont formées par des individus partageant les mêmes haplotypes. Le problème de découverte de line-graphes étant *NP-Complet*, la solution proposée réalise un algorithme de suppression de sommets et d'arêtes. L'algorithme de suppression de sommets est une 6-approximation alors que celui des arêtes est de complexité $O(n * m)$ avec m le nombre de sommets et n le nombre d'arêtes. Dans le cas où des suppressions sont effectuées, le line-graphe fourni est le plus proche possible du line-graphe de l'arbre généalogique. La particularité de la solution est l'absence d'arêtes ajoutées dans le line-graphe et cela implique que cette solution est inapplicable dans notre problème où il existe des arêtes inconnues dans notre graphe de corrélation. En plus, l'ensemble de nos sommets dans le graphe de corrélation est connu contrairement à l'algorithme de *Halldórsson et al.* qui suppose que les sommets doivent être supprimés pour atteindre un line-graphe.
Nous nous basons sur l'algorithme de *Lehot* parce qu'il s'exécute en un temps linéaire en effectuant un traitement sommet par sommet pour la reconnaissance de sous-graphes complets. Ce traitement permet de sélectionner les cliques existantes et les sommets, n'appartenant à aucune clique, qui nécessitent une modification de leur voisinage.

4.3.2 Algorithme de couverture

L'algorithme de *couverture* que nous proposons est une amélioration de celui de *Lehot*. Ainsi, nous présentons brièvement le principe de l'algorithme de couverture en cliques de *Lehot* [61].
Soient H et G deux graphes. Nous supposons que H est le line-graphe de G ($H = L(G)$). Le but de cet algorithme est d'identifier le graphe racine $L^{-1}(H)$ de H. L'algorithme va construire G au fur et à mesure en identifiant les cliques dans H. Les arêtes et les sommets de H et G peuvent avoir au cours de l'exécution plusieurs états :

- Sommet "bien-défini" : un sommet découvert de G tel que la clique correspondante dans H a été trouvée et identifiée.

- Sommet "à moitié-nommé" : un sommet de H tel que l'arête correspondante dans G a une extrémité "bien-définie".

- Sommet "pleinement-nommé" : un sommet de H tel que l'arête correspondante dans G a des extrémités "bien-définies".

- Sommet "basique" : un sommet de H est une arête de G. Ces sommets sont notés $x - y$ dans H avec x et y des sommets découverts de G.

- Sommet "partagé" : sommet de H partageant une arête avec des sommets "basiques" adjacents. Ce sommet est une extrémité commune entre des arêtes de G incidentes.

L'idée de cet algorithme est de déterminer une couverture de corrélation de H en détectant, selon 3 cas [61] dans H, les sommets partagés adjacents à un sommet "basique" qui forment une clique. Nous illustrons le fonctionnement de l'algorithme avec l'exemple suivant illustré par la figure 4.6. L'algorithme sélectionne deux sommets "basiques" $1-2, 2-3$ et l'ensemble X des sommets adjacents aux sommets "basiques". Si $X = \emptyset$ alors il n'existe pas de sommet partagé dans G et les sommets $1-2, 2-3$ sont étiquetés "à moitié-nommé" (figure 4.6(a)). Si $X = \{x\}$ alors le sommet x est un sommet partagé dans H si $x = 2-4$ car le triangle $\{x, 1-2, 2-3\}$ est impair. Dans le cas où $x = 1-3$, le triangle $\{x, 1-2, 2-3\}$ est pair et aucun sommet découvert dans G n'est incident aux sommets du triangle $\{x, 1-2, 2-3\}$ (figure 4.6(b)). Le sommet x dans H est étiqueté "pleinement-nommé" et les autres sommets $1-2, 2-3$ dans H sont étiquetés "à moitié-nommé". Si $X = \{x, y\}$, il n'existe aucun sommet partagé dans H si x et y sont adjacents dans H. Dans le cas où ils ne sont pas adjacents alors ils forment deux triangles avec $1-2, 2-3$. Si $x = 2-4$ et $y = 1-3$ alors le triangle $\{2-4, 1-2, 2-3\}$ est impair et x est le sommet partagé dans H (figure 4.6(c)). Enfin pour $|X| = |\{a, b, c, \cdots\}| = 3$, le sommet b dans H est étiqueté sommet partagé si a est adjacent à b sinon le sommet a devient le sommet partagé. La dernière étape sélectionne aléatoirement un sommet "à moitié-nommé" dans H qui est adjacent à un sommet "pleinement-nommé" dans H. Si ce sommet n'est pas déjà couvert par une clique alors il est "pleinement-nommé" et il est un sommet partagé dans H. À la fin de cette étape, tous les sommets sont étiquetés à "pleinement-nommé" dans H et ils deviennent des sommets partagés dans H.

L'algorithme s'exécute en $O(m) + m'$ avec m le nombre d'arêtes dans G et m' le nombre d'arêtes dans $L(G)$. Il retourne la liste de cliques découvertes dans H dans laquelle chaque clique correspond à un sommet de G. Cette liste de cliques est appelée *couverture de corrélation* ($\mathcal{CC}$). Malheureusement, si H n'est pas un line-graphe alors il ne retourne pas de couverture partielle.

4.3.2.1 Description de l'algorithme de couverture

Nous proposons l'algorithme de *couverture* (voir algorithme 4) en lien avec celui de *Lehot* qui couvre autant que possible les sommets du graphe de corrélation G_c par une ou deux cliques. Notre algorithme retourne la *couverture de corrélation* ($\mathcal{CC}$) de G_c si la matrice d'adjacence de G_c ne contient aucune case erronée sinon il renvoie une *couverture de corrélation partielle* de G_c.

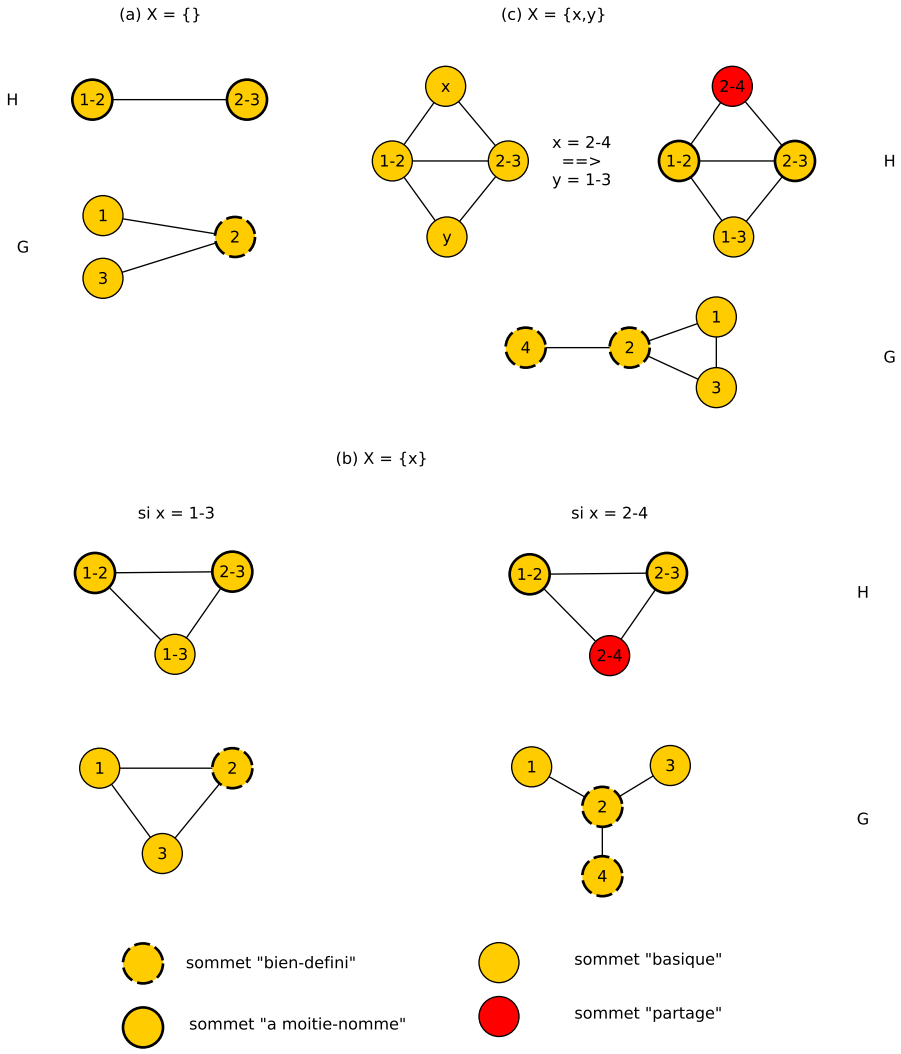

Figure 4.6: Identification des sommets partagés dans le graphe H et nommage des sommets de G.

Soient $G_c = (V, E)$ un graphe de corrélation et $Cliq(v)$ l'état de chaque sommet v de G_c. L'algorithme de *couverture* va construire une couverture de corrélation $\mathcal{CC}(G_c)$ de G_c en ajoutant des cliques découvertes dans $\mathcal{CC}(G_c)$. Une clique est un ensemble de sommets qui induit un sous-graphe complet. Si un sommet appartient à une clique alors il est couvert par cette clique. De même, si deux

sommets u et v appartiennent à une même clique, alors la clique couvre l'arête $[x, y]$. Initialement $\mathcal{CC}(G_c)$ est vide et chaque sommet de $v \in V$ a un état $Cliq(v) = 0$.
À chaque étape de l'algorithme, chaque sommet v a 5 états possibles :

- $Cliq(v) = 0$: le sommet v n'est couvert par aucune clique. Il correspond à un sommet "basique" dans l'algorithme de *Lehot*.
- $Cliq(v) = 1$: le sommet v est couvert par une clique ou deux cliques. Dans le cas où il est couvert par deux cliques, l'intersection de ces cliques donne le sommet v. Ce sommet est étiqueté "pleinement-nommé" dans l'algorithme de *Lehot*.
- $Cliq(v) = 2$: le sommet v est couvert par une clique et peut être couvert par une seconde clique. Ce sommet est "bien-nommé" dans l'algorithme de *Lehot*.
- $Cliq(v) = 3$: le sommet v est un sommet ambigu. L'algorithme doit identifier la clique à laquelle il appartient pour qu'elle devienne un sommet partagé dans l'algorithme de *Lehot*.
- $Cliq(v) = -1$: le sommet v est couvert par plus de deux cliques. Il est contenu dans l'ensemble $\mathcal{C}$ et doit être corrigé par l'algorithme de correction.

Nous choisissons un sommet v de degré minimum qui n'appartient à aucune clique ou qui est un sommet ambigu. S'il existe une partition cohérente (voir définition 5) de ce sommet et de son voisinage $\{v\} \cup \Gamma_{G_c}(v)$ en deux cliques C_1, C_2, alors ces deux cliques sont contenues dans la couverture de corrélation $\mathcal{CC}(G_c)$ en cours de construction. Les sommets v et u (avec $u \in \Gamma_{G_c}(v)$), appartenant à C_1 ou C_2, ont leur état modifié à chaque étape de l'algorithme de la manière suivante :

- $Cliq(v) = 1$ si son état précédent est égal à 0 et la clique C_2 est vide.
- $Cliq(v) = 3$ si son état précédent est égal à 0 et la clique C_2 est non vide.
- $Cliq(v) = 2$ si son état précédent est différent de 0.
- $Cliq(u) = 1$ si son état précédent est égal à 0 et l'ensemble des arêtes incidentes à u est vide.
- $Cliq(u) = 2$ si son état précédent est égal à 3 et l'ensemble des arêtes incidentes à u est vide.
- $Cliq(u) = 3$ si son état précédent est égal à 0 et l'ensemble des arêtes incidentes à u est non vide.
- $Cliq(u) = -1$ si son état précédent est égal à 3 et l'ensemble des arêtes incidentes à u est non vide.

Dans le cas où il n'existe aucune partition cohérente (voir définition 5) au sommet v, son état est à $Cliq(v) = -1$.

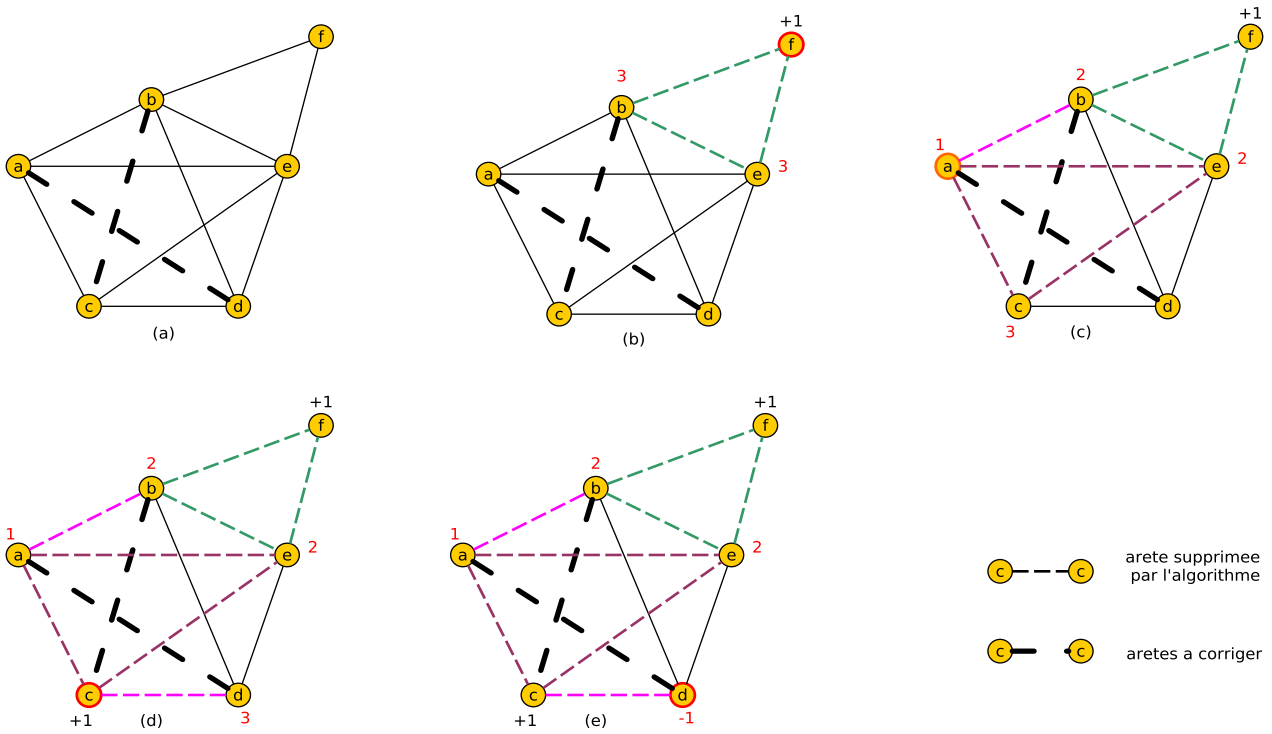

Figure 4.7: Les différentes étapes de la couverture en cliques du graphe G. Les arêtes de même couleur appartiennent à la même clique. Les arêtes (b, c) et (a, d) sont supprimées du graphe avant l'exécution de l'algorithme de *couverture*.

L'algorithme de recherche de couverture de corrélation $\mathcal{CC}(G_c)$ (voir algorithme 4) considère, tant qu'il en existe, un sommet v dont les arêtes incidentes sont couvertes par une seule clique. Il affecte à ce sommet l'état $Cliq(v) = 1$, sauvegarde cette clique dans $\mathcal{CC}(G_c)$ puis supprime ces arêtes incidentes dans le graphe G_c. Les voisins de ce sommet passent à l'état 2.
Si au cours de l'exécution un tel sommet v n'existe pas alors l'algorithme de *couverture* considère un sommet u dont son voisinage peut être couvert par deux cliques et qui n'a pas été précédemment couvert par une clique de $\mathcal{CC}(G_c)$. Si cette partition en deux cliques est unique, l'algorithme affecte $Cliq(u) = 1$ à ce sommet et supprime les arêtes. Dans le cas où ce sommet appartient à un des cas de la figure 4.4 (cas d'un sommet encadré), nous avons montré que ces graphes sont les seuls line-graphes pour lesquels deux partitions possibles existent. Nous utilisons la fonction de décision $Verif - correl$ (section 2.2.2.5) pour lever l'ambiguïté.
Un état est affecté aux autres sommets v de la clique couvrant v selon l'un des trois cas. Dans le premier

cas, l'état actuel appartient à $Cliq(v) \in \{2,3\}$ si les sommets v ont des arêtes incidentes non encore couvertes par une clique et l'état précédent est $Cliq(v) \in \{3,0\}$. Dans le second cas, $Cliq(v) = 1$ est attribué aux sommets v si l'ensemble des arêtes incidentes est vide. Enfin, dans le dernier cas, l'algorithme affecte $Cliq(v) = -1$ à ces sommets si leur état précédent est $Cliq(v) \in \{2,3\}$ et leurs arêtes incidentes ne forment pas une clique.
À la fin de l'exécution de cet algorithme, tous les sommets v dont l'état courant est $Cliq(v) = -1$ n'ont pas été couverts. Ces sommets appartiennent à l'ensemble $\mathcal{C}$ des sommets à corriger par l'algorithme de correction.

La figure 4.7 détaille l'algorithme de *couverture* (algorithme 4) sur un graphe $G = (V, E)$ dans lequel nous avons supprimé deux arêtes. L'objectif de la suppression d'arêtes dans la figure 4.7(a) est d'obtenir des sommets à corriger à la fin de la couverture. Nous sélectionnons le sommet f car il est de degré minimum. Il forme une clique avec son voisinage alors la clique $\{f, b, e\}$ est ajoutée à l'ensemble $\mathcal{CC}(G)$ puis les arêtes $(f,b),(b,e),(f,e)$ sont supprimées de E. L'état de f est $Cliq(f) = 1$ et les sommets b et e ont $Cliq(b) = Cliq(e) = 3$ (voir figure 4.7(c)).
Le second sommet traité est a car son état est $Cliq(a) = 0$. Il existe deux partitions cohérentes $\{a, b\}$ et $\{a, e, c\}$ au voisinage de a. Ces deux partitions sont ajoutées à $\mathcal{CC}(G)$ et les arêtes de ces cliques sont supprimées de E. L'algorithme attribue

- Au sommet a, l'état $Cliq(a) = 1$,
- Aux sommets b et e, les états $Cliq(b) = Cliq(e) = 2$ car leur état prédécent était $Cliq(b) = Cliq(e) = 3$ et ces sommets ont encore un voisin,
- Au sommet c, l'état $Cliq(b) = 3$.

On traite les autres sommets (c) de la même manière jusqu'à ce qu'on sélectionne le sommet d. Ce sommet a deux partitions $\{d, b\}$ et $\{d, e\}$ non cohérentes (voir définition 5) parce que la fonction $Verif - correl$ (section 2.2.2.5) appliquée à ces partitions retourne 0. Ce sommet est donc à corriger $\mathcal{C} = \{d\}$ (voir figure 4.7(e)).

Si le graphe $G_c = (V_c, E_c)$ est un graphe de corrélation alors l'algorithme de couverture en détermine la couverture de corrélation $\mathcal{CC}(G_c)$. En effet, si G_c est un line-graphe, par récurrence sur l'ensemble des sommets et à chaque étape, il existe un sommet non encore couvert qui :

- Soit est couvert par une clique appartenant à $\mathcal{CC}(G_c)$ et son voisinage restant peut être convert par une nouvelle clique.

- Soit n'est couvert par aucune clique de $\mathcal{CC}(G_c)$ et son voisinage restant peut être couvert par une ou deux nouvelles cliques.

- Soit est dans une ambiguïté alors on a recours à la fonction $Verif - correl$ (section 2.2.2.5) pour déterminer les bonnes partitions de ce sommet.

4.3.2.2 Complexité de l'algorithme de couverture

Si G_c est un line-graphe, le sommet choisi u (s'il existe) à la ligne 4 de l'algorithme 4 est à l'état $Cliq(u) = 0$ et le sommet u n'est pas un sommet ambigu. Dans le cas contraire, il est à l'état $Cliq(u) = 3$. Chaque sélection de sommets conduit à une unique et correcte partition et aussi à une seule couverture de corrélation. Nous montrons par induction sur l'ensemble des sommets qu'à chaque étape de l'algorithme 4, il y a un sommet :

- Non couvert par aucune clique et certains de ses voisins peuvent être couverts par 1 ou 2 nouvelles cliques ($Cliq(u) = 0$, voir graphe (b) de la figure 4.7).

- Couvert par une clique déjà dans la couverture de corrélation $\mathcal{CC}$ et ses voisins non couverts peuvent être couverts par une nouvelle clique ($Cliq(u) = 3$, voir graphe (c) de la figure 4.7).

L'algorithme de couverture détermine la couverture de corrélation si G_c est véritablement un line-graphe.

En revanche, si le graphe G_c n'est pas un line-graphe alors, à certaines étapes de l'exécution, deux différents méthodes de couvrir le sommet choisi par des cliques peuvent se présenter. Dans certains cas, nous choisissons aléatoirement une des méthodes (cela peut avoir un impact sur l'algorithme de correction). Notons que nous réalisons les mêmes opérations si le graphe G_c est non connexe, même si des composantes connexes sont isomorphes aux graphes de la figure 4.3. Le line-graphe obtenu est toujours un graphe connexe.

Concernant la complexité, déterminer si un sommet u de G_c est couvert par 1 ou 2 cliques a une complexité de $O(\Delta(G_c)^2)$ (en déterminant si le nombre chromatique du graphe complémentaire est 1 ou 2). Alors la complexité de l'algorithme de couverture est dans le pire des cas $O(n \times \Delta(G_c)^2)$ avec n le nombre de sommets. Rappelons que l'algorithme de Lehot [61] a une complexité de $O(n \times \Delta(G_c))$. Cependant, il ne fournit pas de couverture de corrélation partielle lorsque G_c n'est pas un line-graphe.

Conclusion : si le graphe G_c est un line-graphe, tous ses sommets v sont labellisés à $Cliq(v) = 1$ et l'algorithme de *couverture* trouve une partition du voisinage d'un sommet en une ou deux cliques

de façon unique (voir les lemmes précédents). Une fois ce sommet et ses arêtes incidentes supprimées, le graphe restant est toujours un line-graphe, et la propriété se propage. Ainsi G_c qui possède des sommets v aux états $Cliq(v) = -1$ n'est pas un line-graphe. Nous proposons l'algorithme de correction qui retourne le line-graphe le plus proche de G_c.

Algorithme 4 Couverture

if G_c est isomorphe à un graphe double (voir figure 4.5) **then**
    le traiter avec $Verif - correl$ ([1])
else
    while il existe un sommet u t.q $Cliq(u) \in \{0, 3\}$ **do**
        choisir un sommet u de degré minimum
        if $\{u\} \cup \Gamma_{G_c}(u)$ peut être couvert par deux cliques C_1 et C_2 cohérentes, C_1 maximale et $C_2 = \emptyset$ si $Cliq(u) = 3$ ([2]) **then**
            if $Cliq(u) = 0$ et $C_2 \neq \emptyset$ **then**
                $Cliq(u) = 3$
            else
                if $Cliq(u) = 0$ et $C_2 = \emptyset$ **then**
                    $Cliq(u) = 1$
                else
                    $Cliq(u) = 2$
                end if
            end if
            $\epsilon_u = E(G_c[C_1]) \cup E(G_c[C_2])$
            for $w \in \Gamma_{G_c}(u)$ **do**
                $\alpha(w) = card\{[w, x] \in E - \epsilon_u\}$
                if $\alpha_w > 0$ **then**
                    if $Cliq(w) = 0$ **then**
                        $Cliq(w) = 3$
                    else
                        if $Cliq(w) = 3$ **then**
                            $Cliq(w) = -1$
                        end if
                    end if
                else
                    if $Cliq(w) = 0$ **then**
                        $Cliq(w) = 1$
                    else
                        if $Cliq(w) = 3$ **then**
                            $Cliq(w) = 2$
                        end if
                    end if
                end if
            end for
            $E = E - \epsilon_u$
        else
            $Cliq(u) = -1$
        end if
    end while
end if

[1] : chaque graphe de la figure 4.5 admet deux couvertures de corrélation, souvent isomorphes, mais une seule de ces couvertures de corrélation peut correspondre au DAG du réseau électrique sous-jacent. Dans ce cas, on utilise la fonction $Verif - correl$ afin de déterminer la couverture de corrélation la plus probable.

[2] : le sommet u choisi (s'il existe) ne sera pas prioritairement un sommet tel que $Cliq(u) = 0$ et u est un point d'ambiguïté. Si lors d'une étape, seul un tel choix est possible et qu'il n'y a aucun sommet u tel que $Cliq(u) = -1$, c'est que chaque sommet du graphe initial G_c est un point d'ambiguïté. Dans ce cas, G_c est une union de composantes connexes isomorphes à un des graphes de la figure 4.5. Dans ce cas, n'importe quel choix conduit à une couverture de corrélation correcte.

4.3.3 Algorithme de correction

Nous l'avons vu, si $G_c = (V_c, E_c)$ n'est pas un line-graphe, certains sommets ne peuvent pas être converts par 1 ou 2 cliques. Dans l'algorithme de couverture, ces sommets v sont labellisés par $Cliq(v) = -1$. L'ensemble des cliques $\mathcal{CC}(G_c)$ ne contient alors que des cliques dans lesquelles les sommets v labellisées à $Cliq(v) = 1$ sont couverts par 1 ou 2 cliques. Les sommets v dont l'état $Cliq(v) = -1$ appartiennent à l'ensemble $\mathcal{C}$ des sommets à corriger et ce sont ces sommets qui sont traités par l'algorithme suivant.

Nous proposons l'*algorithme de correction* qui va modifier l'ensemble initial E_c par l'ajout et la suppression d'arêtes dans le but d'obtenir un *line-graphe*. Dans cet algorithme, nous traitons un sommet de $\mathcal{C}$ après l'autre sachant que chaque sommet peut modifier $\mathcal{C}$. Soit z_i le i^{ieme} sommet traité dans $\mathcal{C}$. Certaines expériences réalisées dans le chapitre 5 montrent que le choix des sommets à traiter, à chaque étape de correction, peut avoir une influence sur le line-graphe fourni parce que la correction modifie le voisinage des sommets. Nous notons alors E_c^i l'ensemble des arêtes de G_c après le traitement du $(i-1)^{ieme}$ sommet de $\mathcal{C}$ et $\mathcal{CC}^i(G_c)$ l'ensemble des cliques de G_c à l'étape i. Ainsi $E_c^1 = E_c$ et $\mathcal{CC} = \mathcal{CC}(G_c) = \mathcal{CC}^1(G_c)$. Nous notons $\mathcal{CC}$ pour désigner $\mathcal{CC}(G_c)$ dans la suite de cette section.

Soient z_i le i^{ieme} sommet et $\mathcal{CC}(z_i) = \{C_1, \cdots, C_k\}$ l'ensemble des cliques maximales de $\mathcal{CC}^i$ de taille supérieure ou égale à 3 auxquelles le sommet z_i appartient. Notons que, par définition et par construction, chaque paire de cliques dans $\mathcal{CC}(z_i)$ n'a que z_i comme sommet commun et que $S(z_i)$ est l'union des voisins v de z_i dans des cliques $\{v, z_i\} \in \mathcal{CC}^i$ de taille 2 et des voisins v de z_i tels que l'arête $[z_i, v]$ n'est couverte par aucune clique de $\mathcal{CC}^i$.

$$C(z_i) = \{C_i, i \in [1, k] \mid |C_i| \geq 3 \ \& \ C_i \in \mathcal{CC}^i\} \tag{4.1}$$

$$S(z_i) = \{v \in \Gamma_{G_c}(z_i) \mid \{v, z_i\} \in \mathcal{CC}^i\} \cup \{v \in \Gamma_{G_c}(z_i) \mid \nexists C \in \mathcal{CC}^i, [z_i, v] \in E_c(C)\} \quad (4.2)$$

Définition 8. *Soient $\mathcal{CC}^i$ la couverture de corrélation après le traitement des $(i-1)^{ieme}$ sommets de $\mathcal{C}$ et $\mathcal{CC}^i(z_i)$ l'ensemble des cliques contenant le i^{ieme} sommet z_i.*
Deux cliques C et C' de $\mathcal{CC}^i(z_i)$ sont **contractables** *si aucune arête $[u, v]$ de E_c^i telle que $u \in C$ et $v \in C'$ n'est couverte par une clique (autre que u, v) dans $\mathcal{CC}^i$. Un ensemble de cliques de $\mathcal{CC}^i$ est contractable si tous les cliques sont deux à deux contractables.*

Dans la figure 4.8(a), les paires de cliques $(C3, C4)$, $(C2, C3)$ sont contractables car il n'y a aucune arête entre les sommets 5 et 6 dans la première paire et dans la seconde paire, les sommets 3 et 4 n'ont aucune arête entre eux. Cependant, la paire $(C4, C6)$ n'est pas contractable car l'arête $[z_i, 10]$ est couverte par la clique $C5$. De même, la clique $C1$ n'appartenant pas à $\mathcal{CC}^i(z_i)$ entraine que les cliques $C1$ et $C2$ ne sont pas contractables.

Définition 9. *Soient $\mathcal{CC}^i$ la couverture de corrélation après le traitement des $(i-1)^{ieme}$ sommets de $\mathcal{C}$ et $\mathcal{CC}^i(z_i)$ l'ensemble des cliques contenant le i^{ieme} sommet z_i.*
Une clique $C \in \mathcal{CC}^i$ est **voisine** *de z_i si $C \notin \mathcal{CC}^i(z_i)$ et $card(C \cap S(z_i)) \geq 1$. La dépendance d'une clique C voisine de z_i est l'ensemble $D_{z_i}(C) \subset \mathcal{CC}^i(z_i)$ tel que $C' \in D_{z_i}(C)$ si et seulement si $C' \cap C \cap \Gamma_{G_c}(z_i) \neq \emptyset$.*
Une clique C est **augmentante** *pour le sommet z_i si et seulement si elle est voisine de z_i et $D_{z_i}(C)$ est vide ou $D_{z_i}(C) \cup \{C\}$ est contractable.*

$$voisine(z_i) = \{C \in \mathcal{CC}^i \mid C \notin C(z_i) \ \& \ card(C \cap S(z_i)) \geq 1\} \quad (4.3)$$

$$D_{z_i}(C) = \{C' \in C(z_i) \mid C' \cap C \cap \Gamma_{G_c}(z_i) \neq \emptyset\} \quad (4.4)$$

On appelle **augmentation** du sommet z_i l'union d'une clique augmentante C pour z_i et d'une contraction de cliques de $D_{z_i}(C)$.

Dans notre exemple, considérons $\overline{C}(z_i) = \{C1, C6\}$ les cliques n'appartenant pas à $\mathcal{CC}^i(z_i)$ et $S(z_i) = \{10, 1\}$. L'ensemble des cliques voisines à z_i est $voisine(z_i) = \{C1, C6\}$ parce que l'intersection de $C1$ et $S(z_i)$ donne un sommet $\{1\}$ et celle de $C6$ et $S(z_i)$ donne un sommet $\{10\}$ ($C1 \cap S(z_i) = \{1, 2, 11\} \cap \{10, 1\} = \{1\}$, $C6 \cap S(z_i) = \{8, 9, 10\} \cap \{10, 1\} = \{10\}$). Par ailleurs, la dépendance de la clique $C1$ est $D_{z_i}(C1) = C2$ ($C1 \cap C2 \cap \Gamma_{G_c}(z_i) = \{2\}$) et celle de $C6$ est $D_{z_i}(C6) = C4$ ($C6 \cap C4 \cap \Gamma_{G_c}(z_i) = \{8\}$). Nous en déduisons que la clique $C1$ est *augmentante* car $C1$ est contractable avec $C2$ et est voisine de z_i. De même, la clique $C6$ est *augmentante* car $C6$ est voisine de z_i et puisque l'arête $[z_i, 10]$ forme la clique $C5$, la paire $(C6, C4)$ est contractable. Une *augmentation* de z_i est soit $\{z_i\} \cup C1 \cup C2$ ou soit $\{z_i\} \cup C4 \cup C6$.

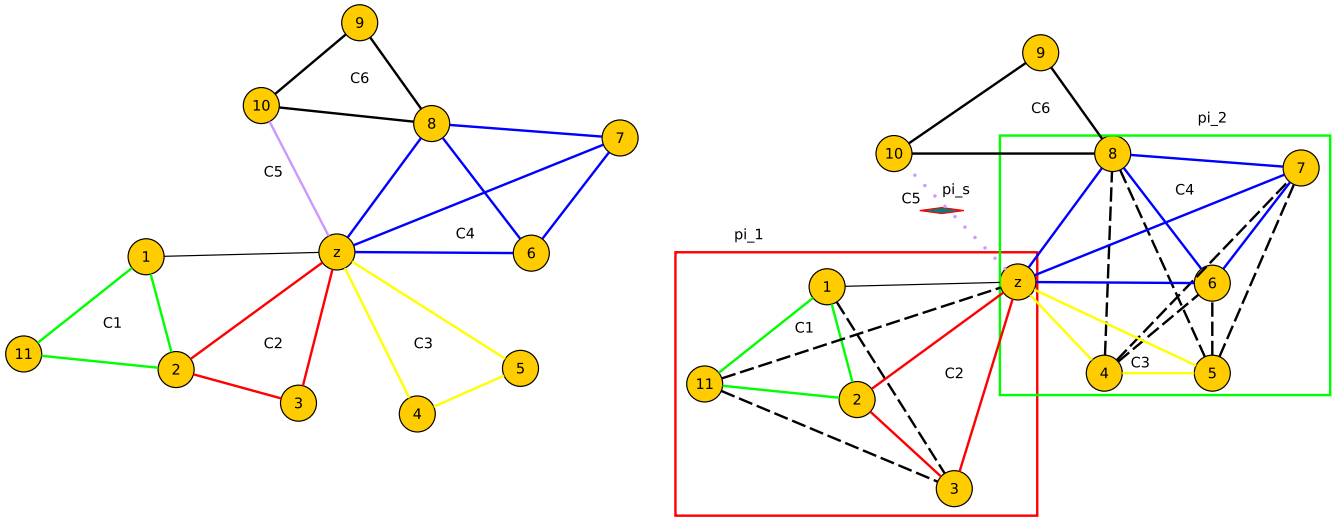

Figure 4.8: (a) Le sommet z et son voisinage avec les cliques qui le couvrent , (b) un exemple de compression de cliques : les sommets à l'intérieur des rectangles rouges et verts forment les nouvelles cliques couvrant z. $\pi_1 = \{1, 2, 3, z, 11\}$, $\pi_2 = \{z, 4, 5, 6, 7, 8\}$, $\pi_s = \{10\}$

Soient π_1, π_2 une bipartition du voisinage du sommet z_i en cliques et π_s un ensemble de sommets dont l'arête formée par un de ces sommets et le sommet z_i sont à retirer du graphe G_c.

Définition 10. *Soient $\mathcal{CC}^i$ la couverture de corrélation après le traitement des $(i-1)^{ieme}$ sommets de $\mathcal{C}$ et $\mathcal{CC}^i(z_i)$ l'ensemble des cliques contenant le i^{ieme} sommet z_i.*
On appelle **compression** *du sommet z_i un triplet (π_1, π_2, π_s) défini par :*

- *π_1 (resp. π_2) peut être d'une des formes suivantes :*

  1. *L'union de z_i, d'un sous-ensemble C_1 (resp. C_2) de cliques de $\mathcal{CC}^i(z_i)$ telle que toute paire (C, C') de C_1 (resp. C_2) est contractable et d'un sous-ensemble S_1 (resp. S_2) de sommets $v \in S(z_i)$ n'appartenant à aucune clique de C_1 (resp. C_2) tel que*

  $$\forall v \in S_1,\ \forall x \in C_1, \nexists C' \in \mathcal{CC}^i \text{ t.q. } card(C') > 2 \ \text{ et } \ \{v, x\} \subset C'$$

  (ce qui fait que {v,x} peut être une clique de $\mathcal{CC}^i$).

  2. *Une augmentation du sommet z_i.*

- *L'intersection entre π_1 et π_2 est réduite $\{z_i\}$ ($\pi_1 \cap \pi_2 = \{z_i\}$),*

- *$\pi_s = \Gamma_{G_c}(z_i) - ((\pi_1 \cap \Gamma_{G_c}(z_i)) \cup (\pi_2 \cap \Gamma_{G_c}(z_i)))$ tel que l'ensemble des arêtes $\{[z_i, v] \in E_c^i : v \in \pi_s\}$ n'est pas déconnectant.*

- *Le triplet* $\pi_1 \cap \Gamma_{G_c}(z_i)$, $\pi_2 \cap \Gamma_{G_c}(z_i)$, $\pi_s \cap \Gamma_{G_c}(z_i)$ *est une 3-partition de* $\Gamma_{G_c}(z_i)$.

Il existe toujours une telle compression, ne serait-ce que $\pi_1 = \{z_i\} \cup C_i \in C(z_i)$, $\pi_2 = \emptyset$, $\pi_s = \Gamma_{G_c}(z_i) - (\Gamma_{G_c}(z_i) \cup C_i)$ si $\mathcal{CC}^i(z_i)$ n'est pas vide. Sinon, $\pi_1 = \{z_i\} \cup \{v \in \Gamma_{G_c}(z_i)\}$, $\pi_2 = \emptyset$, $\pi_s = \Gamma_{G_c}(z_i) - \{v\}$ est aussi une compression. Un exemple de compression est aussi donné dans la figure 4.8. Le coût $c(T)$ d'une compression π_1, π_2, π_s est défini par :

$$c(T) = |\{\{u,v\} \in \pi_1 :\ [u,v] \notin E_c^i\}| + |\{\{u,v\} \in \pi_2 :\ [u,v] \notin E_c^i\}| +\ |\pi_s|$$

L'exemple de la compression donnée dans la figure 4.8(b) est $\pi_1 = C1 \cup C2$ (une augmentation), $\pi_2 = C3 \cup C4$ (ces deux cliques étant contractables), et $\pi_s = \{10\}$. Les cliques $C1$ et $C2$ sont compressées en ajoutant les arêtes $[1,3]$ et $[z_i, 11]$. De même, les cliques $C3$ et $C4$ sont compressées en ajoutant les arêtes $[4,8]$, $[5,8]$, $[7,4]$, $[7,5]$, $[6,4]$ et $[6,5]$. La clique $C5$ est supprimée afin que z_i ne soit pas couvert par trois cliques. Le coût de cette compression est 10, 10 étant le nombre d'arêtes en pointillées plus l'arête supprimée $[10, z_i]$.

Soit $c(T)$ le coût minimum d'une compression T de z_i. Le but est de modifier G_c afin que z_i puisse être couvert par une ou deux cliques issues de π_1 et π_2. Pour cela, le coût de cette modification $c(T)$ tient compte des arêtes à ajouter (liées à π_1 et π_2) et à supprimer (liées à π_s).

$$c(T) = \sum_{\{u,v\} \subseteq \pi_1 : [u,v] \notin E_c^i} \phi^+(u,v) + \sum_{\{u,v\} \subseteq \pi_2 : [u,v] \notin E_c^i} \phi^-(u,v) + \sum_{v \in \pi_s} \phi^-(u,v) \tag{4.5}$$

Avec ϕ^+ le coût de l'opération *ajouter une arête* et ϕ^- le coût de l'opération *ajouter une arête*. Nous évaluons les performances des différents couples de fonctions ϕ^+ et ϕ^- dans le chapitre 5.

Ainsi, **appliquer une compression** $T = \pi_1, \pi_2, \pi_s$ consiste à ajouter dans E_c^i les arêtes définies par les ensembles de paires $\{\{u,v\} \in \pi_1 :\ [u,v] \notin E_c^i\}$ (qui seront couvertes par la clique π_1) et $\{\{u,v\} \in \pi_2 :\ [u,v] \notin E_c^i\}$ (qui seront couvertes par la clique π_2) et à supprimer les arêtes $\{[z_i,v] \in E_c^i :\ v \in \pi_S\}$. Dès lors, le sommet z_i appartient aux deux cliques π_1 et π_2. On procède alors aux mises à jour suivantes pour obtenir $\mathcal{CC}^{i+1}$ et E_M^{i+1} :

- Supprimer toutes les cliques $\mathcal{CC}^i(z_i)$ couvertes par π_1 dans $\mathcal{CC}^i$.
- Supprimer toutes les cliques $\mathcal{CC}^i(z_i)$ couvertes par π_2 dans $\mathcal{CC}^i$.
- Supprimer toutes les cliques de cardinalité 2 couvertes par π_1 et π_2 dans $\mathcal{CC}^i$.
- Ajouter π_1 et π_2 dans $\mathcal{CC}^i$, supprimer de E_c^{i+1} toutes les arêtes $\{[z_i,v] \in E_c^i :\ v \in \pi_s\}$.

- Affecter $Cliq(z)$ à 1 (si π_1 ou π_2 est vide) ou 2 (sinon).

Cette procédure a les propriétés suivantes :

Propriété 1. *Considérons l'application d'une compression.*
Soit $\mathcal{CC}^{i+1}$ l'ensemble obtenu à partir de $\mathcal{CC}^i$ après la mise à jour selon cette application.

- *Tout sommet de G_c couvert par une ou deux cliques dans $\mathcal{CC}^i$ le reste dans $\mathcal{CC}^{i+1}$.*
- *Toute arête couverte par une et une seule clique dans $\mathcal{CC}^i$ et qui n'est pas supprimée le reste dans $\mathcal{CC}^{i+1}$.*
- *Le sommet z_i est couvert par une ou deux cliques dans $\mathcal{CC}^{i+1}$ (le nombre de sommets ainsi couverts augmente de 1 par rapport à celui dans $\mathcal{CC}^i$).*

Ainsi, pour chaque sommet z_i, on considère une compression de coût minimum c_m^i et on l'applique. La propriété ci-dessus garantit qu'à la fin du processus, on obtient un graphe de corrélation $G_c^t = (V_c, E_c^t)$ dont l'ensemble $\mathcal{CC}^i$ modifié est une couverture de corrélation. Considérons la distance de correction $DC(G_c^0, G_c^t) = |(E_c^0 \cup E_c^t) - (E_c^0 \cap E_c^t)|$ qui est le nombre de cases modifiées dans la matrice d'adjacence du graphe G_c. La distance-line vérifie

$$DL(G_c^0, G_c^t) \leq DC(G_c^0, G_c^t)$$

Notons que lors d'une étape $j > 1$, le sommet z_j et son voisinage se retrouvent être couvert par une ou deux cliques suite au traitement des $j-1$ sommets précédents, aucune compression ne lui est appliquée (on considère la compression identité) et donc $c_m^j = 0$.

4.3.4 Complexité des algorithmes

L'algorithme de correction traite au plus une fois chaque sommet du graphe. La complexité de traitement de chaque sommet est exponentielle en fonction du degré de chaque sommet et des cliques auxquelles il appartient, la encore en fonction de son degré en taille et en nombre. L'algorithme global (couverture et correction) est donc pseudo-polynomial en fonction du degré du graphe.

Nous déterminons une conjecture sur le comportement de l'algorithme. Étant donné un graphe de départ, une exécution de l'algorithme est un ordre dans lequel seront traités les sommets dans l'algorithme de couverture, puis la sélection des sommets z_i à traiter dans $\mathcal{C}$.
Considérons un graphe de corrélation G_c n'étant pas isomorphe à un graphe de la figure 4.5. On dira que G_c est non-ambigu.

Deux arêtes $[u, v]$ et $[u', v']$ de G_c seront dit **clique-independantes** si et seulement si il n'existe pas de cliques C dans la couverture de corrélation de G_c telle que $C \cap \{u, v\} \cap \{u', v'\} \neq \emptyset$

Conjecture 6. *Si $G' = (V, E')$ est un graphe obtenu en supprimant un ensemble d'arêtes deux à deux clique-independantes d'un graphe de corrélation non-ambigu $G_c = (V, E_c)$, alors il existe une exécution de l'algorithme qui transforme G' en G_c.*

4.3.5 Conclusion de la description des algorithmes

Dans cette section, nous décrivons deux algorithmes. Le premier algorithme est *l'algorithme de couverture* qui attribue un état à un sommet du graphe de corrélation en fonction des cliques qui le couvrent. L'ensemble de cliques est la couverture de corrélation $\mathcal{CC}$. La particularité de la couverture de corrélation est que chaque sommet appartient à 1 ou 2 cliques. Lorsqu'un sommet n'est pas couvert par 1 ou 2 cliques, cela signifie que le graphe de corrélation n'est pas un line-graphe et ces sommets sont regroupés dans l'ensemble $\mathcal{C}$ de sommets à corriger.

Le second algorithme est *l'algorithme de correction*. Il consiste à ajouter ou à supprimer des arêtes au voisinage d'un sommet $u \in \mathcal{C}$ afin que la partition de ce sommet et son voisinage forment deux cliques. Pour assurer ces opérations d'ajout et de suppression, il utilise une phase d'augmentation et de compression. En effet, la phase d'augmentation détermine les cliques de $\mathcal{CC}$ contenant u, les cliques de $\mathcal{CC}$ dont u partage une arête avec un sommet de la clique (cliques voisines), les cliques contractables (cliques de $\mathcal{CC}$ dont l'intersection retourne le sommet u) et les cliques dépendantes (cliques contenant u dans lesquelles un des sommets partagent une arête avec une clique de la couverture de corrélation $\mathcal{CC}$). Puis elle effectue le produit cartésien de ces ensembles de cliques. Chaque élément de ce produit est noté π_1 ou π_2. Quant à la phase de compression, elle sélectionne deux éléments du produit cartésien qu'elle note π_1 et π_2, puis elle ajoute des arêtes à π_1 et π_2 dans l'ensemble des arêtes initiales du graphe de correction pour en faire des cliques. Elle crée aussi l'ensemble π_s des arêtes à supprimer pour que le sommet u ne soit couvert que par deux cliques. Les cliques π_1 et π_2 sont ajoutées à la couverture de corrélation $\mathcal{CC}$.
Avec la découverte de la couverture de corrélation $\mathcal{CC}$, nous allons construire le graphe racine de ce line-graphe dans la section suivante.

4.4 Détermination de la topologie du réseau énergétique

Soient $\mathcal{CC}(G_c)$, l'ensemble des cliques du line-graphe G_c et le graphe non orienté $G' = (V', E')$ sous-jacent du DAG G.

Nous considérons que chaque clique de $\mathcal{CC}(G_c)$ est un sommet dans G'. Si l'intersection de deux cliques $c_1, c_2 \in \mathcal{CC}(G_c)$ retourne l'arête a_i alors nous ajoutons l'arête a_i dans E' entre les sommets $c_1, c_2 \in G'$. Dans le cas où un sommet de G_c n'appartient qu'à une seule clique $c \in \mathcal{CC}(G_c)$, nous ajoutons un nouveau sommet (noté ext_c) dans V' puis nous ajoutons une arête $[ext\_c, c]$ dans E'. Nous obtenons le graphe G' non orienté connexe. Nous utilisons la figure 4.9 pour illustrer la construction de G'. La couverture de corrélation de G_c est $\mathcal{CC}(G_c) = [c_1 = \{\{a,b\},\{b,c\},\{b,d\}\}, c_2 = \{\{d,f\},\{f,h\},\{c,f\}\}, c_3 = \{\{b,c\},\{c,e\},\{c,f\}\}, c_4 = \{\{c,e\},\{e,g\}\}, c_5 = \{\{b,d\},\{d,f\}\}]$. Les sommets du G' sont $V' = \{c_1, c_2, c_3, c_4, c_5\}$. L'intersection des cliques ci-dessous est non vide et le sommet d'intersection est l'identifiant d'une arête dans G'.

$$c_1 \cap c_3 = \{b,c\},\ c_1 \cap c_5 = \{b,d\},\ c_3 \cap c_2 = \{c,f\},\ c_3 \cap c_4 = \{c,e\},\ c_2 \cap c_5 = \{d,f\}$$

Pour les sommets de G_c couverts par une seule clique, nous créons le sommet ext_x dans G' avec x le nom d'une clique de $\mathcal{CC}(G_c)$ puis nous ajoutons une arête entre le sommet ext_x et le sommet de G' correspondant à la clique. Par exemple le sommet $\{a,b\}$ de G_c est couvert par la clique c_1. Nous relions ext_c_1 de G' avec le sommet c_1 de G'. Nous répétons la même opération pour les sommets $\{f,h\}$, $\{e,g\}$ de G_c. Le graphe G' (figure 4.9(c)) est ainsi construit et est isomorphe au graphe G de la figure 4.9(a).

4.4.1 Orientation du graphe G'

Nous supposons que le graphe non orienté $G' = (V', E')$ sous-jacent au DAG G a déjà été obtenu. Nous orientons les arêtes de G' afin de découvrir le DAG cible.

Soient un sommet v, son voisinage $N(v)$ et une bipartition $p(v)$, $s(v)$ de v.
Si $Verif - correl(p(v), s(v)) = 1$ alors la partie $p(v)$ est l'ensemble $p(v)$ des arcs entrants de v et la partie $s(v)$ est l'ensemble $s(v)$ des arcs sortants de v. Sinon la bipartition n'est pas la bonne et nous testons une autre bipartition. Nous testons ainsi dans l'ordre de $2^{d(v)}$ bipartitions avec $d(v) = |N(v)|$ dans le pire des cas,.
Soit une séquence $S = v_1, v_2, \cdots, v_k$ de sommets de G telle que chaque arête est incidente à $\{v_1, \cdots, v_k\}$ et que $\{v_1, \cdots, v_{k-1}\}$ n'a pas cette propriété.
Soit d_i le nombre d'arêtes liant v_i à un sommet de $V - \{v_1, \cdots, v_{i-1}\}$.
Nous allons prendre dans l'ordre chaque sommet de la séquence S et nous allons traiter ses arêtes pour trouver une bipartition correcte. Si une arête a déjà été traitée pour un sommet, elle n'est plus prise en compte dans les arêtes incidentes des sommets suivants dans la séquence S. Le traitement du sommet

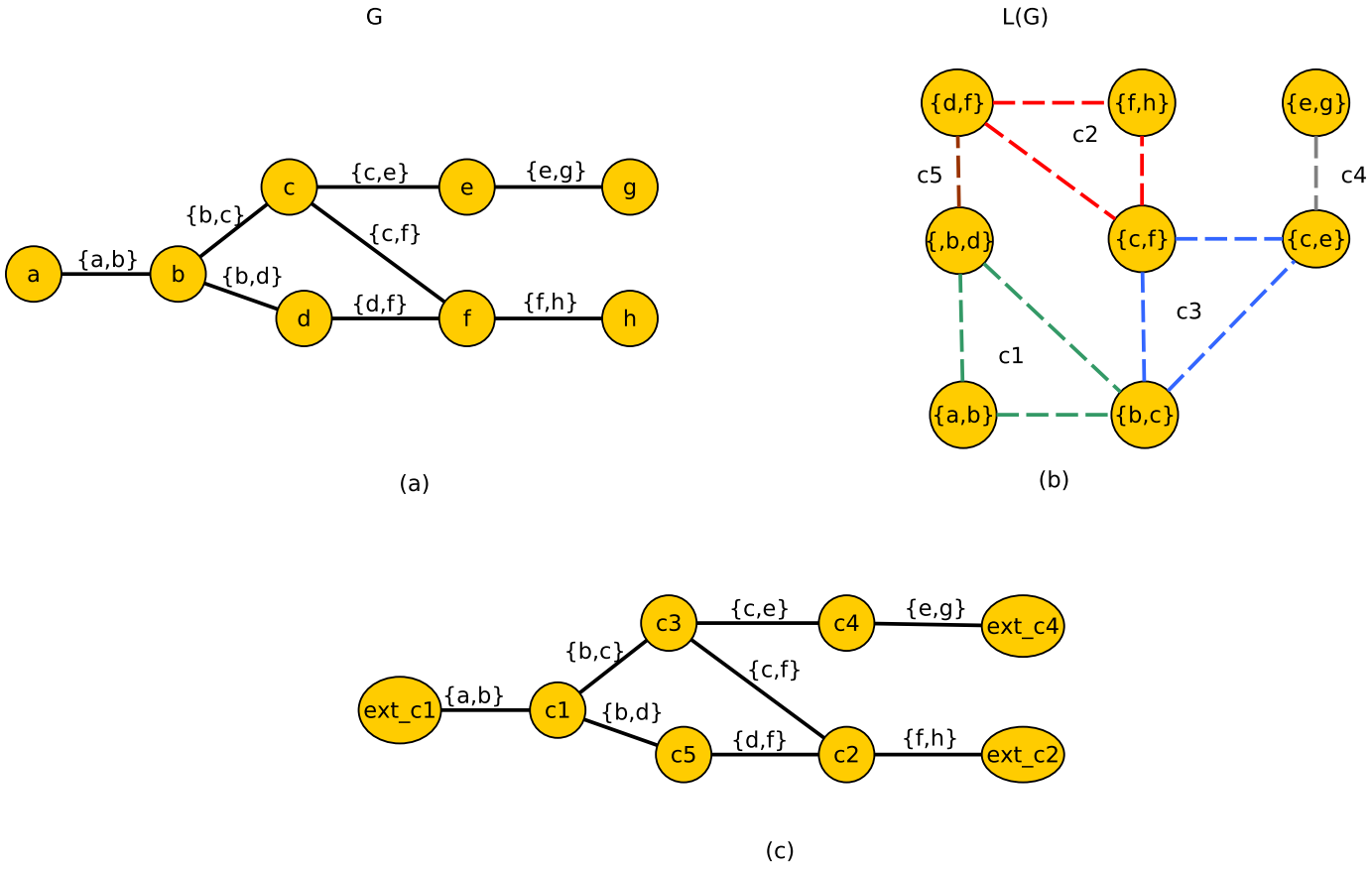

Figure 4.9: Construction de la topologie non orienté de G_c. (a) : réseau initial modélisé par G. (b) : line graphe de G. (c) : graphe G' reconstruit.

v_i nécessite 2^{d_i} opérations et le nombre total d'opérations est alors

$$2^{d_1} + 2^{d_2} + \cdots + 2^{d_k}$$

Notre problème est de trouver une telle liste telle que cette somme est minimum. L'heuristique est, à chaque étape, de choisir le sommet tel que le nombre d'arêtes incidentes non encore traitées est minimum. Par exemple, on commence par le sommet v_1 comme sommet de degrée minimum. À chaque étape, on prend le sommet v_i tel que son degré d_{v_i} est minimum.

Cependant, cette heuristique n'est pas optimale et voici un contre-exemple illustré par la figure 4.10.
Soit le graphe H composé de 3 cliques K_5 et d'un sommet v ayant une arête incidente avec un seul sommet dans chaque clique K_5. Le sommet v est de degré minimum. Nous considérons 2 séquences

de sommets différents pour dénombrer les bipartitions. La première séquence suit l'heuristique et la seconde séquence débute par un sommet de K_5 de degré 4.
Soit $c(S)$ la somme des bipartitions possibles. En considérant l'heuristique, nous débutons par v. Puis le sommet suivant de degré minimum est un sommet de K_5. On traite tous les sommets de cette clique K_5 avant de passer à une autre clique K_5. Le nombre de bipartitions traitées avec l'heuristique est

$$c(S_H) = 2^3 + 3(2^4 + 2^3 + 2^2 + 2^1) = 98$$

En considérant la seconde séquence, on choisit le sommet de K_5 de degré minimum. On traite ce sommet en 2^4 bipartitions. Le sommet suivant est encore dans cette clique mais le nombre de bipartitions baisse à 2^3. Après les deux sommets traités, le nombre de bipartitions est $2^4 + 2^3$. On répète le traitement des sommets jusqu'à ce que tous les sommets soient traités avant de passer à une autre clique K_5. Dans cette clique, on reprend à nouveau la seconde séquence. Le nombre de bipartitions traitées avec la seconde séquence est

$$c(S_2) = 3(2^4 + 2^3 + 2^2 + 2^1) = 96$$

Nous remarquons que le nombre de bipartitions avec la seconde séquence est minimum. Cet exemple confirme que la solution de l'heuristique n'est pas minimale car il existe une autre séquence de choix de ces sommets (ici la seconde séquence) qui minimise le nombre de bipartitions.

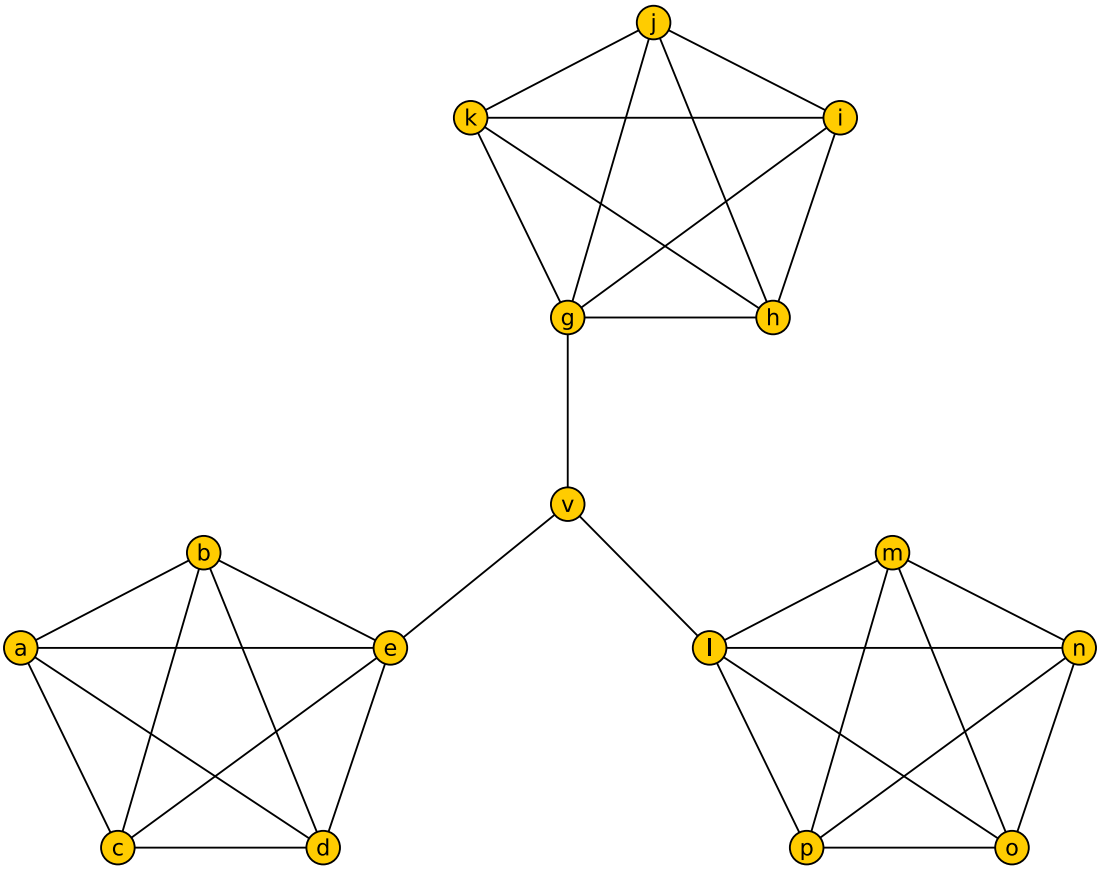

Figure 4.10: Un contre-exemple de l'heuristique choisie pour l'orientation des arêtes. On choisit un sommet de degré minimum dans une clique k_5. Ensuite, on traite tous les sommets de la clique et on se sert du degré minimum pour choisir les sommets de la séquence. Puis on passe à une clique et on reprend le traitement jusqu'à ce qu'il ne reste plus d'arêtes dans une seule clique.

Conclusion : l'algorithme d'orientation que nous avons proposé choisit le sommet de degré minimum à chaque étape de l'algorithme et cette solution n'est pas optimale. Nous conjecturons que la séquence S de traitement des sommets est $NP - complet$.

4.5 Conclusion du chapitre 4

Nous avons décrit la relation existante entre la matrice de corrélation et la notion de line-graphe. En effet, la matrice de corrélation appliquée à une valeur de seuil donne la matrice d'adjacence d'un line-graphe si elle ne contient aucune erreur de corrélation. Dans le cas où elle possède des cases erronées, il est impossible de déterminer la *couverture de corrélation* du graphe de corrélation. Nous avons alors défini le problème *Proxi-Line* dont l'objectif est de trouver le line-graphe qui a le moins d'arêtes

différentes avec le graphe de corrélation.

Ensuite, nous avons présenté les propriétés d'un line-graphe et les travaux existants dans la découverte de couverture en cliques sur des line-graphes. Nous avons retenu l'algorithme de Lehot [60] comme la base de notre algorithme de couverture parce qu'il attribue des états à tous les sommets à chaque étape de l'algorithme. Cette opération permet de connaitre les sommets non couverts des sommets déjà couverts.
Dans le but de résoudre le problème *Proxi-Line*, nous proposons ainsi deux algorithmes.
Le premier algorithme est *l'algorithme de couverture* et il couvre tous les sommets par une ou deux cliques à partir des états des sommets. Nous avons distingué trois types d'états. En effet, un sommet couvert par une clique et qui possède des arêtes incidentes est à l'état 2. Un sommet couvert par une clique ou deux cliques et qui n'a aucune arête incidente est à l'état 1. Un sommet couvert par deux cliques ayant des arêtes incidentes ou un sommet ayant des arêtes incidentes qui ne forment pas une clique est à l'état -1. L'ensemble des sommets v à l'état -1 est l'ensemble $\mathcal{C}$ de sommets à corriger. À la fin de l'algorithme, il retourne les cliques découvertes (couverture de corrélation $\mathcal{CC}$).
Le second algorithme proposé est *l'algorithme de correction*. Il se base sur l'ensemble $\mathcal{C}$ et les cliques découvertes pendant l'algorithme de couverture. En effet, cet algorithme sélectionne chaque sommet $u \in \mathcal{C}$ et le corrige en procédant par une phase d'augmentation et de compression. La phase d'augmentation détermine les cliques de $\mathcal{CC}$ contenant u, les cliques de $\mathcal{CC}$ dont u partage une arête avec un sommet de la clique (cliques voisines), les cliques contractables (cliques de $\mathcal{CC}$ dont l'intersection retourne le sommet u) et les cliques dépendantes (cliques contenant u dans lesquelles un des sommets partagent une arête avec une clique de la couverture de corrélation $\mathcal{CC}$). Puis elle effectue le produit cartésien de ces ensembles de cliques. Chaque élément de ce produit est noté π_1 ou π_2. Quant à la phase de compression, elle sélectionne deux éléments du produit cartésien qu'elle note π_1 et π_2, puis elle ajoute des arêtes de π_1 et π_2 dans l'ensemble des arêtes initiales du graphe de correction pour en faire des cliques. Elle crée aussi l'ensemble π_s des arêtes à supprimer afin que le sommet u ne soit couvert que par deux cliques. Les cliques π_1 et π_2 sont ajoutées à la couverture de corrélation $\mathcal{CC}$. La complexité des algorithmes de couverture et de correction est *pseudo-polynomiale* en fonction du degré du graphe de corrélation.

Enfin, la dernière section présente la construction de la topologie du graphe racine et son orientation. Pour la construction de la topologie, nous nous servons principalement de la couverture de corrélation. En effet, chaque clique de cette couverture de corrélation est un sommet dans le graphe racine. Si l'intersection de deux cliques est non vide alors il existe une arête entre les sommets correspondant à ces cliques dans le graphe racine. Concernant l'orientation des arêtes, nous nous servons de la fonction

$Verif-correl$ (section 2.2.2.5) qui teste tous les bipartions possibles afin de trouvant les arcs incidents entrants et sortants d'un sommet du graphe racine. Nous avons montré qu'il existe un ordre de sommets qui réduit le nombre de bipartitions à tester. Toutefois cette solution n'est pas optimale.

Chapitre 5

Évaluation des performances des algorithmes

Dans ce chapitre, nous générons des topologies de réseaux électriques qui sont des DAG sans circuits. À partir de ces topologies, nous construisons leurs line-graphes que nous modifions selon deux approches. La première approche consiste à changer les valeurs de k cases choisies aléatoirement. La seconde approche construit une matrice associée au line-graphe du DAG dont chaque case contient une valeur de probabilité puis applique une valeur de seuil sur cette matrice pour en déduire une matrice d'adjacence. De ce fait, cette matrice d'adjacence contient des cases modifiées qui seront désignées par *erreurs*.
Notre objectif est d'évaluer les performances de notre couple d'algorithmes sur ces line-graphes modifiés c'est-à-dire la capacité de nos algorithmes à corriger les *erreurs*. Pour ce faire, nous divisons ce chapitre en quatre parties. La première partie décrit la génération de graphes électriques (les DAG) et la construction de leurs line-graphes associés. Ensuite, la seconde partie présente les différentes étapes de la modification des k cases des line-graphes, le protocole d'expérimentation et l'analyse des résultats. La troisième partie analyse les performances des algorithmes sur des line-graphes modifiés par la deuxième approche. Enfin, dans la dernière partie, nous analysons ces performances sur des graphes dit *grilles bouclées*. Dans ces graphes, chaque sommet est couvert par plus de deux cliques.

5.1 Génération de graphes électriques

La topologie du réseau électrique est représentée par un graphe orienté sans circuit G. Les câbles électriques sont unidirectionnels et les équipements sont toujours alimentés par une source. Ce qui implique que le courant se propage dans une direction et cette direction indique l'orientation des arcs d'un *DAG* (*Directed Acyclic Graph*). Nous allons décrire comment nous générons le graphe G.

Considérons un graphe non orienté $G = (V, E)$ dans lequel V est l'ensemble de n sommets, E l'ensemble des m arêtes et α son degré moyen choisi. La probabilité d'existence d'une arête entre deux sommets est de $\frac{\alpha}{n}$. Afin de générer un tel graphe après avoir choisi n et α, nous sélectionnons deux sommets x et y de V et nous générons une valeur p_{xy} qui suit une loi de probabilité uniforme. Si p_{xy} est supérieure à la probabilité d'existence d'une arête alors nous ajoutons l'arête $[x, y]$ à E.
Si G n'est pas connexe, nous choisissons aléatoirement un sommet dans chaque composante connexe et nous ajoutons une arête entre ces sommets. Nous obtenons alors m arêtes.
Afin d'orienter G comme un DAG, nous sélectionnons de façon aléatoire quatre sommets de degré minimum pour les définir comme les sources de notre tri topologique. Nous effectuons ce tri avec un parcours en largeur *Breast First Search (BFS)* dans le graphe G. Chaque sommet x obtient un ordre topologique D_x et l'arête e_{xy} devient soit l'arc a_{xy} si $D_x < D_y$ soit l'arc a_{yx} si $D_x > D_y$. Les arcs a_{xy} forment l'ensemble A des arcs de G. Nous en déduisons que $G = (V, A)$ est orienté et son line-graphe LG est construit à partir de la définition 3.
Nous notons M_G la matrice d'adjacence de G et M_{LG} la matrice d'adjacence du line-graphe LG.

5.2 Expérimentation 1 : modification de k cases de la matrice du line-graphe

5.2.1 Sélection de k cases et génération de la matrice $M_{k,p}$

Nous modifions k cases de la matrice d'adjacence M_{LG}. Ces cases sont choisies de manière aléatoire. Afin de contrôler la proportion des cases à modifier dont la valeur initiale est 0 ou 1, nous introduisons la probabilité p.
Soit donc $p \in [0, 1]$ la variable qui désigne la proportion de cases à 0 sélectionnées. La proportion de cases à 1 est donc $1 - p$. Par exemple, $p = 0.5$ signifie que 50% des cases sélectionnées sont des cases à 0 et 50% des autres cases sélectionnées sont des cases à 1. De même, les k cases sont des cases à 1 si $p = 0$ et elles sont des cases à 0 si $p = 1$. Avec la repartition p, nous calculons les nombres n_0 de cases à 0 et n_1 de cases à 1. Ces cases sont à modifier dans M_{LG}. Ensuite nous sélectionnons uniformement n_0 cases à 0 et n_1 cases à 1 dans la matrice M_{LG}. Les cases à 0 sont changées en 1 et les cases à 1 sont changées en 0. La nouvelle matrice d'adjacence $M_{k,p}$ contient quatre types de cases :

- Si $M_{k,p}[i, j] = M_{LG}[i, j] = 0$ alors $M_{k,p}[i, j]$ est dit *vrai négatif.*
- Si $M_{k,p}[i, j] = M_{LG}[i, j] = 1$ alors $M_{k,p}[i, j]$ est dit *vrai positif.*
- Si $M_{k,p}[i, j] = 0$ et $M_{LG}[i, j] = 1$ alors $M_{k,p}[i, j]$ est dit *faux négatif.*

- Si $M_{k,p}[i,j] = 1$ et $M_{LG}[i,j] = 0$ alors $M_{k,p}[i,j]$ est dit *faux positif.*

La matrice $M_{k,p}$ est la matrice d'adjacence du graphe $G_{k,p}$ et ce graphe a le même ensemble de sommets que LG mais leur ensemble d'arêtes diffère de k arêtes. Généralement, $G_{k,p}$ n'est pas un line-graphe. Toutefois, s'il est un line-graphe alors il est impossible que $G_{k,p}$ soit le line-graphe de G.

5.2.2 Protocole d'expérimentation sur les graphes $G_{k,p}$

L'application de nos algorithmes de découverte et de correction débute par la génération de 500 topologies électriques G de 30 sommets, chacune ayant un degré maximal moyen de $\Delta(G) = 5$. Nous construisons 500 line-graphes LG de 150 sommets et 470 arêtes, en moyenne.
Nous avons donc introduit trois paramètres k, p, α_{max}:

1. Le paramètre k désigne le nombre de cases modifiées dans la matrice M_{LG}. Dans notre étude, $k \in \{1, \cdots, 9\}$.

2. Le paramètre p désigne la proportion de cases à 0 sélectionnées parmi les k cases. Dans notre étude, $p \in \{0.1, \cdots, 0.9\}$.

3. Le paramètre α_{max} désigne le nombre de graphes générés pour un couple (k, p) de valeurs données.

Nous choisissons $\alpha_{max} = 5$ pour des temps de calculs réalistes. Chaque graphe généré est identifié par une valeur de $\alpha \in \{1, \cdots, \alpha_{max}\}$, est noté $G_{k,p,\alpha}$ et sa matrice d'adjacence est $M_{k,p,\alpha}$. La valeur α désigne l'indice de modification de k cases dans la matrice M_{LG} pour une valeur de p donnée. Elle est utilisée pour faire varier les k cases choisies dans un graphe. Nous appliquons notre couple d'algorithmes sur une matrice $M_{k,p,\alpha}$ et nous obtenons la matrice $M'_{k,p,\alpha}$ qui est la matrice d'adjacence du line-graphe $LG_{k,p,\alpha}$.

Pour comparer les arêtes entre les graphes $LG_{k,p,\alpha}$ et $G_{k,p,\alpha}$, nous calculons la distance de correction (section 4.3.3) notée $DC_{k,p,\alpha}$. De même, le nombre d'arêtes différentes entre les graphes $LG_{k,p,\alpha}$ et LG définit la distance de Hamming $DH_{k,p,\alpha}$. Nous définissons par les variables $moy\_DH_{k,p}$ et $moy\_DC_{k,p}$, les moyennes respectives des distances de Hamming (notée $DH_{k,p,\alpha}$) et des distances de correction (notée $DC_{k,p,\alpha}$) pour une valeur donnée de k et pour tout $\alpha \in \{1, \cdots, \alpha_{max}\}$.

$$moy\_DH_{k,p} = \sum_{\alpha=1}^{\alpha_{max}} DH_{k,p,\alpha} \quad ; moy\_DC_{k,p} = \sum_{\alpha=1}^{\alpha_{max}} DC_{k,p,\alpha} \tag{5.1}$$

Les différentes étapes de l'expérimentation sont résumées dans la figure 5.1. Les étapes sont représentées par des graphes et les phases de modification de ces graphes sont désignées par les flèches

unidirectionnelles. Quant aux flèches bidirectionnelles (en rouge), elles indiquent le calcul de distances (de Hamming et de correction).

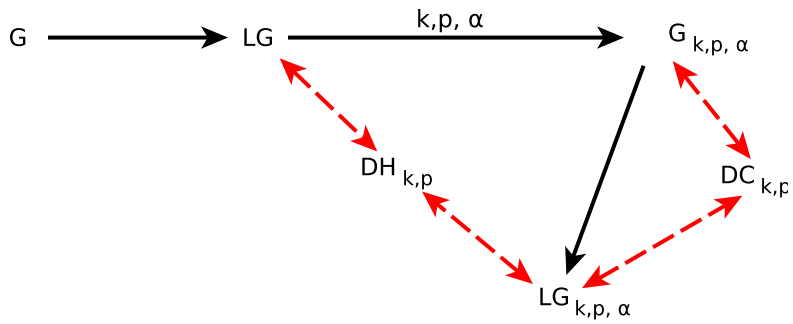

Figure 5.1: Étapes de l'expérimentation :
1) On génère le graphe G et son line-graphe LG;
2) On modifie k cases la α^{ieme} fois selon la repartition p pour obtenir le graphe $G_{k,p,\alpha}$;
3) On applique les algorithmes de couverture et de correction pour avoir un line-graphe $LG_{k,p,\alpha}$. $LG_{k,p,\alpha}$ et $G_{k,p,\alpha}$ diffèrent de $DC_{k,p,\alpha}$ arêtes. $LG_{k,p,\alpha}$ a $DH_{k,p,\alpha}$ cases modifiées par rapport à LG.

Soit $\mathcal{C}$ l'ensemble des sommets n'étant couverts par aucune clique après l'algorithme de couverture. La correction de la matrice $M_{k,p,\alpha}$ est nécessaire s'il existe des sommets appartenant à $\mathcal{C}$.
Nous distinguons deux modes d'exécution de notre algorithme de correction:

1. Mode avec remise :
   1. Exécution algorithme de couverture, **return** $\mathcal{C}$
   2. **Tant que** $\mathcal{C}$ n'est pas vide
   3. Correction d'un sommet de $\mathcal{C}$
   4. Exécution algorithme de couverture, **return** $\mathcal{C}$

2. Mode sans remise :
   1. Exécution algorithme de couverture, **return** $\mathcal{C}$
   2. **Tant que** $\mathcal{C}$ n'est pas vide
   3. Correction d'un sommet de $\mathcal{C}$
   4. Mise à jour du sommet de $\mathcal{C}$

À chaque étape 3 dans les deux modes, un sommet de $\mathcal{C}$ est choisi selon :

(a) *Degré minimum* : le sommet de degré minimum est sélectionné.

(b) *Coût minimum* : le sommet de coût de compression minimum est sélectionné. Le coût de compression est la somme des coûts de chaque case modifiée.

(c) *Aléatoire* : le sommet est sélectionné aléatoirement parmi les sommets de $\mathcal{C}$.

La correction de chaque sommet de $\mathcal{C}$ implique l'ajout et la suppression des arêtes du graphe. Nous souhaitons orienter les décisions de l'algorithme de correction de telle sorte qu'il ajoute ou supprime uniquement des arêtes ou qu'il réalise les deux opérations. Nous priorisons une opération en attribuant des coûts différents à la modification d'arêtes. Nous distinguons trois types d'opérations que nous appelons *fonctions de coût* :

(i) *Unitaire* : l'ajout et la suppression d'une arête ont un même coût c'est-à-dire 1.

(ii) *Ajout* : l'ajout d'une arête a un coût de 1 et la suppression a un coût de 2.

(iii) *Suppression* : l'ajout d'une arête a un coût de 2 et la suppression a un coût de 1.

Nous rappelons que la distance de correction n'est pas la somme de toutes les modifications d'arêtes réalisées. En effet, une arête supprimée et ajoutée plusieurs fois (pour différents sommets de $\mathcal{C}$) n'est comptabilisée qu'une fois et son coût est appliqué selon la fonction de coût.

Étant donnée que nous avons 3 fonctions de coût, 2 modes de correction et 3 sélections possibles des sommets, nous nous retrouvons avec 18 approches de correction de sommets et il est fastidieux de les interpréter sur une même figure. Ainsi nous nous limitons à la fonction de coût *unitaire* dans un premier temps et nous considérons les approches de correction suivantes : $1a$, $1b$, $2a$, $2b$, $2c$. La lecture de l'approche de correction ($1a$) est le suivant : 1) avec remise et a) le degré minimum. L'approche ($1c$) n'est d'aucune utilité car l'algorithme de correction ne parvient pas à fournir un line-graphe. En effet, l'ensemble $\mathcal{C}$ croit linéairement à chaque étape de correction et la correction devient une boucle infinie. Le tableau 5.1 résume les approches de correction retenues dans l'analyse des performances de l'algorithme de correction.

Nous recherchons l'approche qui traite le problème *Proxi-Line* c'est-à-dire le mode qui majore la distance line de $G_{k,p}$ par la distance de correction entre $LG_{k,p}$ et $G_{k,p}$. Une fois le meilleur mode trouvé, nous comparons les fonctions de coût i, $2i$ et $3i$ avec ce mode pour trouver l'influence de la fonction de coût sur les distances de correction.

Tableau 5.1: Tableau récapitulatif des approches de corrections

Mode	choix sommets	fonction de coût
Sans remise	degré minimum	unitaire ajout suppression
	coût minimum	unitaire ajout suppression
	aléatoire	unitaire ajout suppression
Avec remise	degré minimum	unitaire ajout suppression
	coût minimum	unitaire ajout suppression

5.2.3 Analyses des résultats

Nous débutons l'interprétation de nos résultats par l'analyse des distributions des distances de Hamming avec l'approche de correction *aléatoire sans remise* ($2c$) et la fonction de coût *unitaire* ($3i$). Ensuite, nous expliquons le choix de l'approche ($2c$) pour la correction des sommets. Nous présentons également le meilleur compromis dans la repartition des k cases modifiées et la relation existante entre les distances de correction et de Hamming. Enfin, nous montrons que l'approche ($2c$) fournit des distributions de distances de correction et de Hamming identiques, quelles que soient la fonction de coût $(i), (2i), (3i)$ et la repartition k choisies.

5.2.3.1 Interprétation du mode de correction *aléatoire sans remise*

Nous supposons que $p = 0.5$ c'est-à-dire qu'il y a autant de cases *fausses négatives* que de cases *fausses positives* dans la matrice $M_{k,p}$.

Nous représentons les distributions des distances de correction et de Hamming, la fonction de repartition de la corrélation entre ces distances et la fonction cumulative de la distance de Hamming. La distribution des distances de correction indique la proportion de graphes $LG_{k,p,\alpha}$ qui ont le même ensemble d'arêtes que les graphes $G_{k,p,\alpha}$. En ce qui concerne la distribution des distances de Hamming, elle indique la proportion de graphes $LG_{k,p,\alpha}$ qui ont le même ensemble d'arêtes que les graphes LG. La corrélation entre les distances de correction et de Hamming, notée *correlation_DC_DH*, est calculée

avec la formule 5.3. Sa fonction de repartition $F_k(x)$ indique le nombre de corrélations inférieures à une valeur de corrélation x donnée. Quant à la fonction cumulative de la distance de Hamming, elle montre l'évolution du nombre de cases modifiées de la matrice $M'_{k,p}$ en fonction du nombre de line-graphes construits LG.

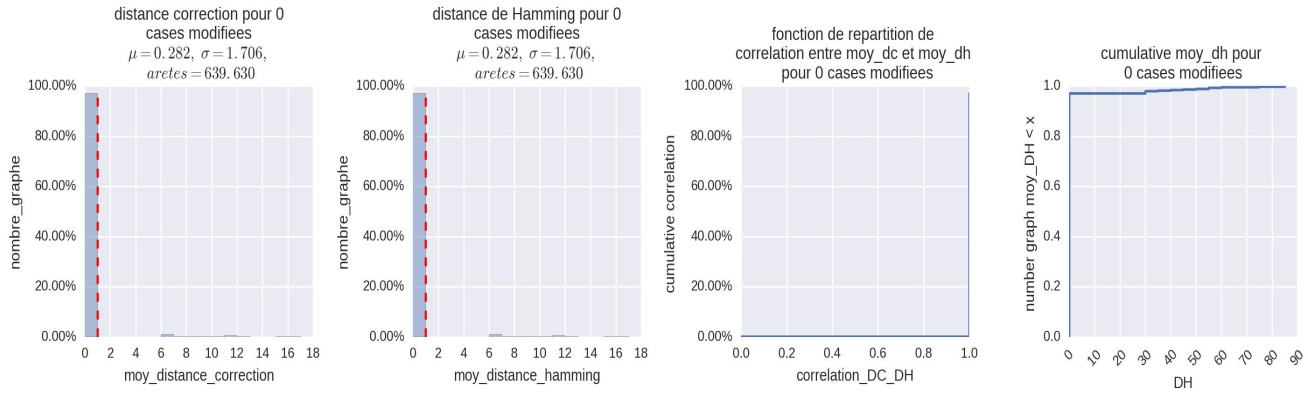

Figure 5.2: Approche de correction aléatoire sans remise à coût unitaire pour $k = 0$ case modifiée. La première colonne représente la distribution des distances de correction $moy\_DC_{0,0.5}$. La seconde colonne est la distribution des distances de Hamming $moy\_DH_{0,0.5}$. La troisième colonne est la fonction de repartition de la corrélation entre les distances de correction et de Hamming avec en abscisse la corrélation entre ces distances (correlation_DC_DH). La quatrième colonne est la fonction cumulative des distances de Hamming.

Les figures 5.2 et 5.3 présentent les courbes respectives pour $k = 0$ et $k \in \{1, 2, 5, 9\}$ cases modifiées. La colonne 1 indique la distribution des distances de correction, la colonne 2 est la colonne de la distribution des distances de Hamming, la colonne 3 est associée à la fonction de repartition de la corrélation entre les distances de correction et de Hamming avec en abscisse la corrélation entre ces distances (*correlation_DC_DH)*. Et la colonne 4 est celle de la fonction cumulative des distances de Hamming. Dans les colonnes 1 et 2, les distributions se divisent en deux zones :

- Zone *gauche* ou *améliorante* : elle correspond aux batonnets de l'intervalle $[0, k]$. Cet intervalle améliore l'ensemble $E'_{k,p,\alpha}$ des arêtes du graphe $LG_{k,p,\alpha}$ pour que $E'_{k,p,\alpha}$ soit identique à l'ensemble E_{LG} des arêtes du graphe LG. Si $moy\_DH_{k,p} \to 0$ alors $LG_{k,p,\alpha}$ est très proche de LG. Si $moy\_DH_{k,p} \to k$ alors les matrices des graphes $LG_{k,p,\alpha}$ et LG diffèrent de k cases et ces cases sont les k cases modifiées dans LG.

- Zone *droite* ou *dégradante* : elle correspond aux batonnets de l'intervalle $]k, +\infty[$. Cet intervalle détériore l'ensemble $E'_{k,p,\alpha}$ des arêtes du graphe $LG_{k,p,\alpha}$. Ainsi, le line-graphe $LG_{k,p,\alpha}$ s'éloigne de LG quand $moy\_DH_{k,p} \to +\infty$.

Ces deux zones sont séparées par une droite en pointillée d'équation $y = k$. Cette droite désigne le nombre de cases modifiées dans le line-graphe LG.

Pour $k = 0$ case modifiée, nous vérifions que nos algorithmes sont cohérents c'est-à-dire que la phase de correction est inutile. En effet, nous avons 100% de graphes $G_{k,p,\alpha}, LG_{k,p,\alpha}, LG$ qui ont les mêmes ensembles d'arêtes et cela implique que $moy\_DH_{0,0.5} = moy\_DC_{0,0.5} = 0$. D'où le seul batonnet dans les colonnes 1 et 2. Par ailleurs, la fonction de repartition de la corrélation et la fonction cumulative des distances de Hamming sont définies par les équations 5.2 (a) et (b) respectivement.

$$F_k(x_1) = \begin{cases} 0 & si \quad x_1 < 1 \\ 100 & si \quad x_1 = 1 \end{cases} (a) \qquad y^0_{cumulDH}(x) = 1 \quad si \quad x \in \mathbb{N}(b) \tag{5.2}$$

avec x_1 la corrélation entre les distances et x le nombre d'arêtes modifiées. Les valeurs des distances de Hamming sont égales à 0 donc sa fonction cumulative $y^k_{cumulDH}$ vaut 1. L'équation 5.2(a) s'interprète comme suit : $F_k(x) = 100\%$ des line-graphes ont leurs distances de correction et de Hamming corrélées $(x = 1)$.

Pour $k \in \{1, 2\}$, le pic des histogrammes se localise dans la zone *améliorante* des colonnes 1 et 2 de la figure 5.3 et son pourcentage est supérieur à 50%. Les autres batonnets sont dans la zone *dégradante* et leur pic a un pourcentage inférieur à 10% en moyenne. Dans la colonne 1 de la figure 5.3, le pic correspond aux k cases modifiées du graphe $G_{k,p,\alpha}$ et son pourcentage est identique à celui du pic de la colonne 2. Le pic de la colonne 2, correspondant à $moy\_DH_{k,0.5} = 0$, signifie que LG et $LG_{k,p,\alpha}$ ont le même ensemble d'arêtes. Ainsi, les $k \leq 2$ cases modifiées sont supprimées de la matrice $M'_{k,p,\alpha}$ lorsque $moy\_DC_{k,0.5} \leq k$. Cependant, les distances de correction et de Hamming ont approximativement les mêmes valeurs lorsque $moy\_DC_{k,0.5} > k$ ($moy\_DC_{k,0.5}$ est dans la partie *dégradante* de la figure 5.3). Cela s'explique par le fait que les distances $moy\_DC_{k,0.5}$ et $moy\_DH_{k,0.5}$ sont corrélées. Nous détaillons la notion de corrélation de distances dans la section 5.2.3.4. Ainsi, les distances de correction et de Hamming sont corrélées dans $\eta_k = 5\%$ des line-graphes $LG_{k,p,\alpha}$ et le line-graphe $LG_{k,p,\alpha}$ devient le line-graphe initial LG si nous corrigons les DC cases modifiées du graphe $LG_{k,p,\alpha}$. La variable η_k est la proportion de line-graphes dont les distances de correction et de Hamming sont fortement corrélées.

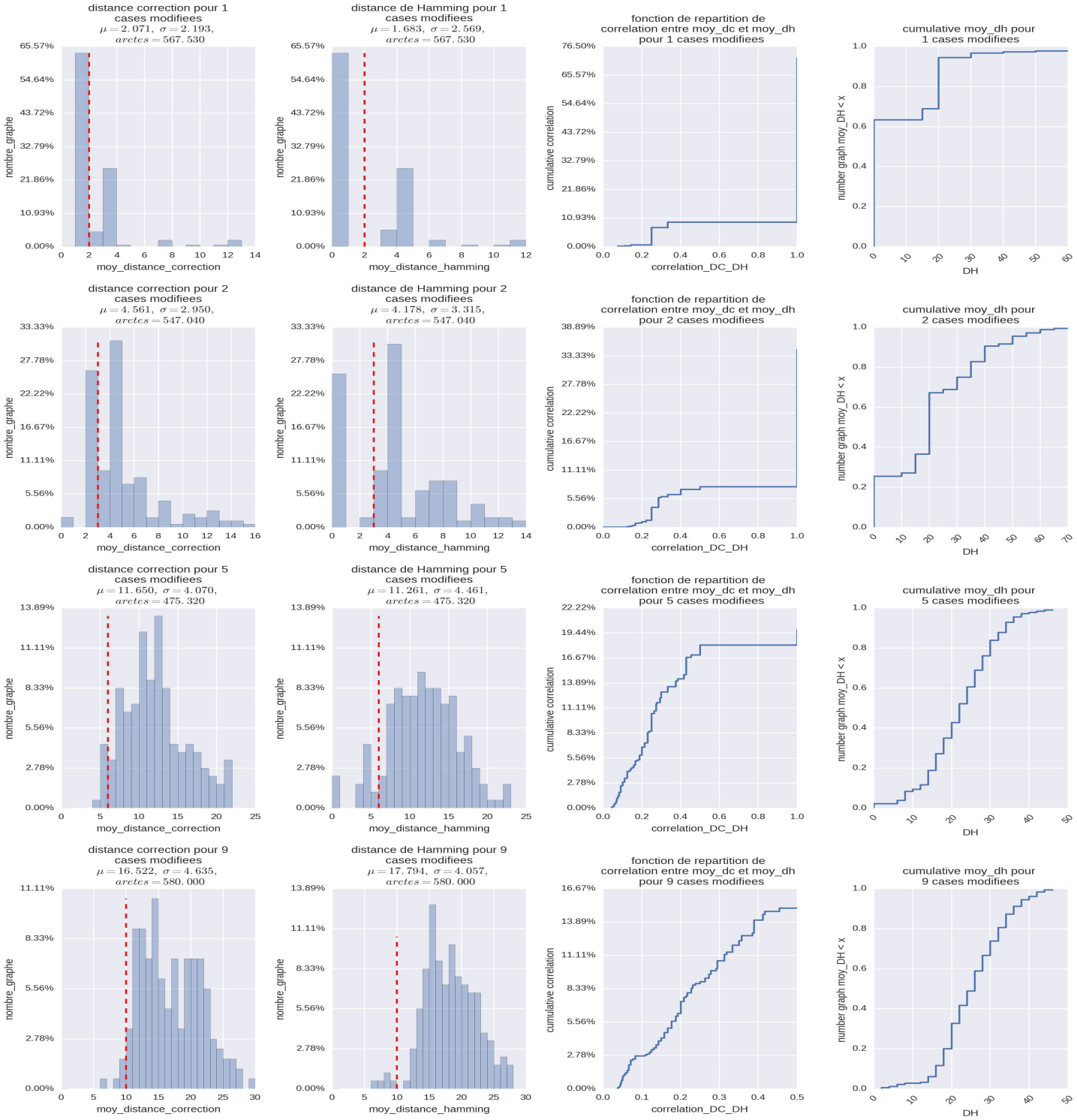

Figure 5.3: Approche de correction aléatoire sans remise à coût unitaire pour $k = \{1, 2, 5, 9\}$ cases modifiées : La première colonne représente la distribution des distances de correction $moy\_DC_{k,0.5}$. La seconde colonne est la distribution des distances de Hamming $moy\_DH_{k,0.5}$. La troisième colonne est la fonction de repartition de la corrélation entre les distances de correction et de Hamming avec en abscisse la corrélation entre ces distances (correlation_DC_DH). La quatrième colonne est la fonction cumulative des distances de Hamming. La première ligne est associée à $k = 1$ case modifiée, la seconde ligne à $k = 2$ cases modifiées, la troisième ligne à 5 cases modifiées et enfin la dernière à 9 cases modifiées.

Pour $k = 5$, le pic se trouve toujours dans la zone *améliorante* des colonnes 1 et 2 de la figure 5.3 mais son pourcentage baisse significativement à 15.69% (voir la ligne 3). Le nombre de line-graphes dans la zone *dégradante* dans les colonnes 1 et 2 augmente tout comme les distances de correction et de Hamming qui atteignent jusqu'à 45 arêtes. La variable η_k est égale à 18.38% de line-graphes $LG_{k,p,\alpha}$ (voir ligne 3 de la colonne 3 de la figure 5.3). Cette augmentation provient de la baisse du pourcentage du pic de la zone *améliorante* au profit de la zone *dégradante* et la plupart des graphes appartenant à cette zone ont leurs distances de correction et de Hamming corrélées. Les 15.69% de line-graphes $LG_{k,p,\alpha}$ identiques à LG s'expliquent par le type de cases modifiées et l'emplacement des arêtes dans le graphe LG. En effet, ces cases modifiées sont des *fausses négatives* et ces arêtes supprimées n'appartiennent pas à des cliques voisines. Toutefois, quelque soit le type de cases modifiées, notre couple d'algorithmes ajoutent beaucoup d'arêtes pour obtenir le line-graphe $LG_{k,p,\alpha}$ lorsque les cases sont reparties avec $p = 0.5$. Par exemple, nous avons constaté, en moyenne, 10 à 20 arêtes différentes pour $k = 5$ cases modifiées dans nos expérimentations. Par ailleurs, nous remarquons qu'il existe des graphes dans lesquels la distance de correction est inférieure à k. Tel est le cas pour $k = 5$ ou nous avons $moy\_DC_{k,0.5} = 3$ et $nombre\_graphe = 0.14\%$ dans la colonne 1 de la figure 5.3. En effet, les cases modifiées sont des cases *fausses négatives*. Ces arêtes supprimées de LG appartiennent à la même clique et certaines arêtes sont ajoutées de telle sorte que la clique se partitionne en deux cliques.

Enfin, pour $k = 9$, la distribution des valeurs de distances est dans la zone *dégradante* et le pic est à $moy\_DC_{k,0.5} = 23$ cases modifiées avec un pourcentage de 6% line-graphes. En comparant les pourcentages des distances $moy\_DC_{k,0.5}$ et $moy\_DH_{k,0.5}$, nous constatons qu'ils ne sont pas identiques comme pour $k \leq 5$. En effet, certaines cases modifiées ne sont pas corrigées car l'algorithme de correction modifie énormément de cases qui sont différentes des k cases et ce taux croit quand $moy\_DC_{k,0.5}$ est élevé. Le taux de cases corrigées est de 53.38% en moyenne. La variable η_k passe à 22% de line-graphes $LG_{k,p,\alpha}$ parce que la correction a modifiée des cases différentes des k cases. Cependant les distances de correction et de Hamming restent toujours corrélées et 3% line-graphes $LG_{k,p,\alpha}$ ont le même ensemble d'arêtes que LG (voir ligne 4 de la colonne 3 de la figure 5.3). C'est pourquoi, les courbes de $F_k(x)$ et $y^k_{cumulDH}$ ont cette forme enrobée proche de la fonction sigmoide de paramètre $\lambda \leq -15$. Pour rappel, nous précisons que ces courbes tendent vers la fonction suivante $f_\lambda(x) = \frac{1}{1+e^{\lambda*(x-0.5)}}$.
Les cases modifiées $k \in \{1, \cdots, 9\}$ sont présentées dans les figures A.1 et A.2 de l'annexe A.

L'approche de correction ($2c$) nous montre que les distances de correction et de Hamming se dégradent quand le nombre k de cases modifiées augmentent. En fait, pour $k \leq 5$, le nombre de line-graphes $LG_{k,p,\alpha}$ identiques à LG est supérieur à $nombre\_graphe = 25\%$ avec des distances de correction $moy\_DC_{k,0.5} = k$. Pour des distances $moy\_DC_{k,0.5} = 2 \times k$, les k cases modifiées sont cor-

rigées. Des cases erronées sont ajoutées pendant l'exécution de l'algorithme de correction mais elles sont peu nombreuses. Nous pouvons considérer ces cases comme la précision de notre algorithme car ces cases indiquent le nombre de cases à modifier pour obtenir LG. Ainsi la distance de correction est majorée par le nombre de cases k modifiées. Cependant, au délà de $k > 5$, la distance de correction $moy\_DC_{k,0.5}$ double et moins de 50% des k cases sont corrigées. Les cases erronées proviennent de l'ajout et la suppression d'arêtes dans le line-graphe $LG'_{k,p,\alpha}$ étant donnée que la fonction de coût est *unitaire*. La distance de correction est majorée par $2 \times k$ en moyenne.

Conclusion : l'approche de correction *aléatoire sans remise* ($2c$) propose des line-graphes $LG_{k,p}$ identiques à LG quand $k \leq 5$. Dans ce cas, le problème *Proxi-Line* est traité car la distance line est majorée par k. Toutefois, le nombre de cases à corriger après l'exécution de notre couple d'algorithmes est faible lorsque la distance de correction est inférieure au double de k ($moy\_DC_{k,0.5} = 2 \times k$) avec $k > 5$. Cela conduit à majorer la distance line par le double des cases erronées. En outre, pour $k > 5$, le pourcentage η_k de corrélation entre $moy\_DC_{k,0.5}$ et $moy\_DH_{k,0.5}$ croit. Nous allons étudier l'évolution de η_k dans le paragraphe 5.2.3.4 mais nous commencons par la comparaison des différentes approches de correction pour en déduire celle qui minimise les distances de correction ou de Hamming.

5.2.3.2 Comparaison des modes de correction

Nous recherchons la meilleure approche de correction parmi les cinq énumérées dans le tableau 5.1. Pour ce faire, nous disposons des distributions des distances de correction et de Hamming, des fonctions de repartition de ces distributions et aussi des moyennes de distances de correction et de Hamming associées aux k cases modifiées. Les distributions des distances de correction et de Hamming sont obtenues avec $p = 0.5$ et la fonction de coût est *unitaire*. Les distributions de distances de chaque approche sont regroupées dans les colonnes 1 et 2 dans les figures en annexes A. Nous décidons d'utiliser la moyenne des distances de Hamming pour la comparaison d'approches de correction parce qu'il est facile de déterminer le nombre de cases modifiées étant donnée que nous connaissons les line-graphes LG et $LG_{k,p}$.

Soit $G^i_{k,p,\alpha}$ le i^{ieme} graphe généré contenant k cases modifiées la α^{ieme} fois avec la repartition $p = 0.5$ des k cases. Nous le notons $G^i_{k,\alpha}$ avec $0 \leq i \leq 500$ et $\alpha \leq \alpha_{max}$. Le line-graphe de $G^i_{k,\alpha}$ obtenu après l'algorithme de correction est noté $L(G^i_{k,\alpha})$.
Soit DH^i_k la distance de Hamming entre LG et $L(G^i_{k,\alpha})$.
La variable $moy\_DH^i_k$ est la moyenne de DH^i_k pour les α_{max} graphes $G^i_{k,\alpha}$ et la variable moy_DH_k est la moyenne de $moy\_DH^i_k$ pour 500 graphes contenant k cases modifiées. La figure 5.4 affiche les

courbes des différents approches de correction pour des distances de Hamming moyennées moy_DH_k en fonction des k cases modifiées. En ordonnée, nous avons le nombre moyen de cases différentes entre deux line-graphes.

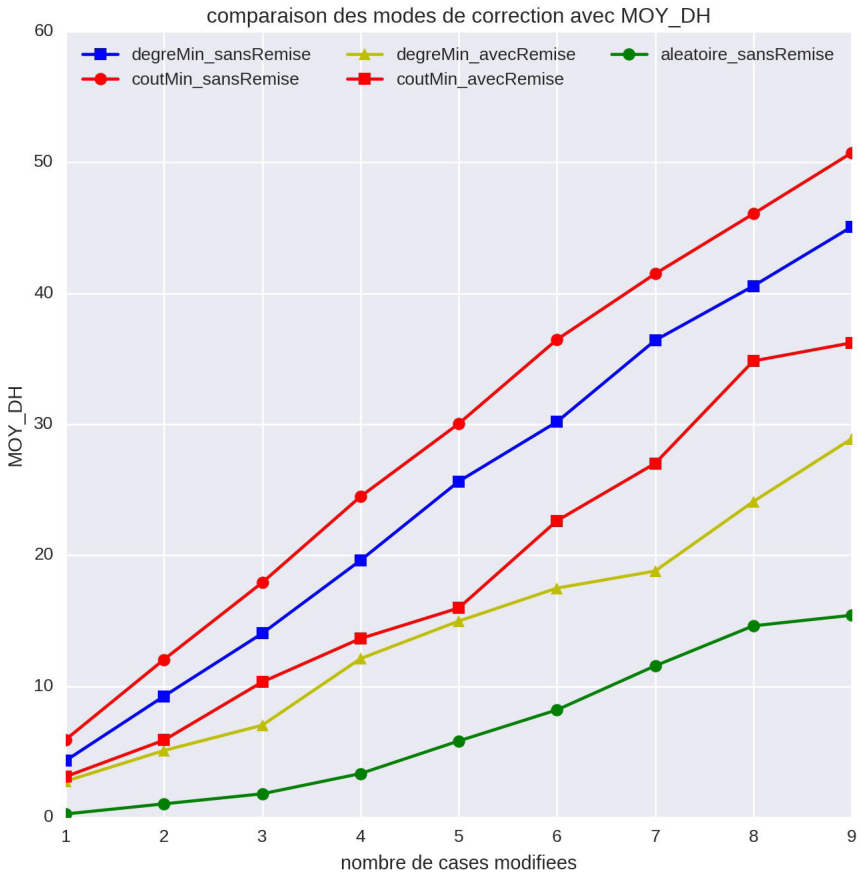

Figure 5.4: Comparaison des différentes approches de correction de sommets pour $k \in \{1, \cdots, 9\}$ cases modifiées. Les courbes en bleu carré : approche degré minimum sans remise (2a), rouge carrée : approche coût minimum avec remise (1b), rouge rond : approche coût minimum sans remise (2b), vert rond : approche aléatoire sans remise (2c) et jaune triangle : approche degré minimum avec remise (1a)

Considérons des courbes associées aux approches $(2b)$, $(2c)$ et $(1b)$. En choisissant les nombres de cases modifiées $k \in \{4, 8\}$, nous avons $moy\_DH_{k,p} \in \{4, 15\}$ cases pour l'approche $(2c)$, $moy\_DH_{k,p} \in \{13, 36\}$ cases pour l'approche $(1b)$ et $moy\_DH_{k,p} \in \{25, 46\}$ cases pour l'approche $(2b)$. Pour $k = 4$ cases modifiées, l'approche $(1b)$ modifie, en moyenne, 9 cases de plus que l'approche

($2c$). En revanche, ce nombre moyen de cases modifiées augmente à 21 cases quand $k = 8$. De même, l'approche ($2b$) modifie 12 cases de plus que l'approche ($1b$) pour $k = 4$ cases modifiées et 10 cases pour $k = 8$ cases. L'approche ($2c$) donne de meilleures résultats par rapport aux approches ($1b$) et ($2b$).

Conclusion : l'approche *aléatoire sans remise* propose de meilleurs résultats que les approches ($2a$), ($2b$), ($1a$) et ($1b$) parce que les distances de correction sont minimales pour toute valeur de k comme le montre la figure 5.4. Nous retenons, pour la suite, l'approche *aléatoire sans remise* comme l'approche de correction des sommets de $\mathcal{C}$, sommets n'appartenant à aucune couverture.

5.2.3.3 Influence des cases modifiées et de la fonction de coût

Nous mesurons l'influence des fonctions de coût sur nos distances de Hamming. Pour ce faire, nous appliquons les fonctions de coût *unitaire*, *ajout* et *suppression* (voir tableau 5.1) pour en déduire les valeurs de p qui sont favorables à l'algorithme de correction c'est-à-dire qui minimisent les distances de Hamming.

Nous considérons d'abord la fonction *unitaire*. La figure 5.5 représente l'évolution des distances de Hamming selon les différentes valeurs de p. Les courbes de p sont éloignées pour $k \leq 5$ et au délà de $k > 5$, les courbes se rapprochent. En effet, pour $k \leq 5$, 43.4% des cases *fausses négatives* en moyenne sont corrigées pour $p \in \{0.7, 0.9\}$ et pour $p = 0.8$, cette moyenne est de 45.5%. Quant aux cases *fausses positives*, seulement 10% des cases sont corrigées. Ces moyennes sont plus élevées que celles obtenues avec $p < 0.7$. En effet, seulement 19.32% des cases *fausses positives* et 38% des cases *fausses négatives* sont corrigées avec $p < 0.7$. L'algorithme de correction ajoute beaucoup d'arêtes pour $p \geq 0.7$ par rapport à $p < 0.7$ parce que le nombre de cases *fausses négatives* dans la matrice du graphe de corrélation est élevé pour $p < 0.7$ sachant que le coût d'ajout d'une arêtes est de 1.
De même, pour $k > 0.5$, 80% des cases erronées sont des cases *fausses négatives* après l'algorithme de correction. L'algorithme privilégie l'ajout à la suppression d'arêtes.
La fonction *unitaire* a peu d'influence sur les variations des distances de Hamming quelque soit les valeurs de p.

La figure 5.7 correspond à la fonction *suppression* et elle a le même comportement que la fonction *unitaire* parce que toutes les courbes divergent quand $k \leq 5$ et convergent quand $k > 5$. Ici nous remarquons aussi que nous avons en moyenne beaucoup de cases *fausses négatives* à la fin de l'algorithme de correction. Avec la fonction *suppression*, nous constatons que le nombre moyen de cases *fausses négatives* à la fin de l'algorithme de correction est largement supérieur au nombre de cases *fausses négatives* modifiées dans la matrice $M_{k,p,\alpha}$. Cependant, le nombre de ces cases *fausses négatives* (39.4%

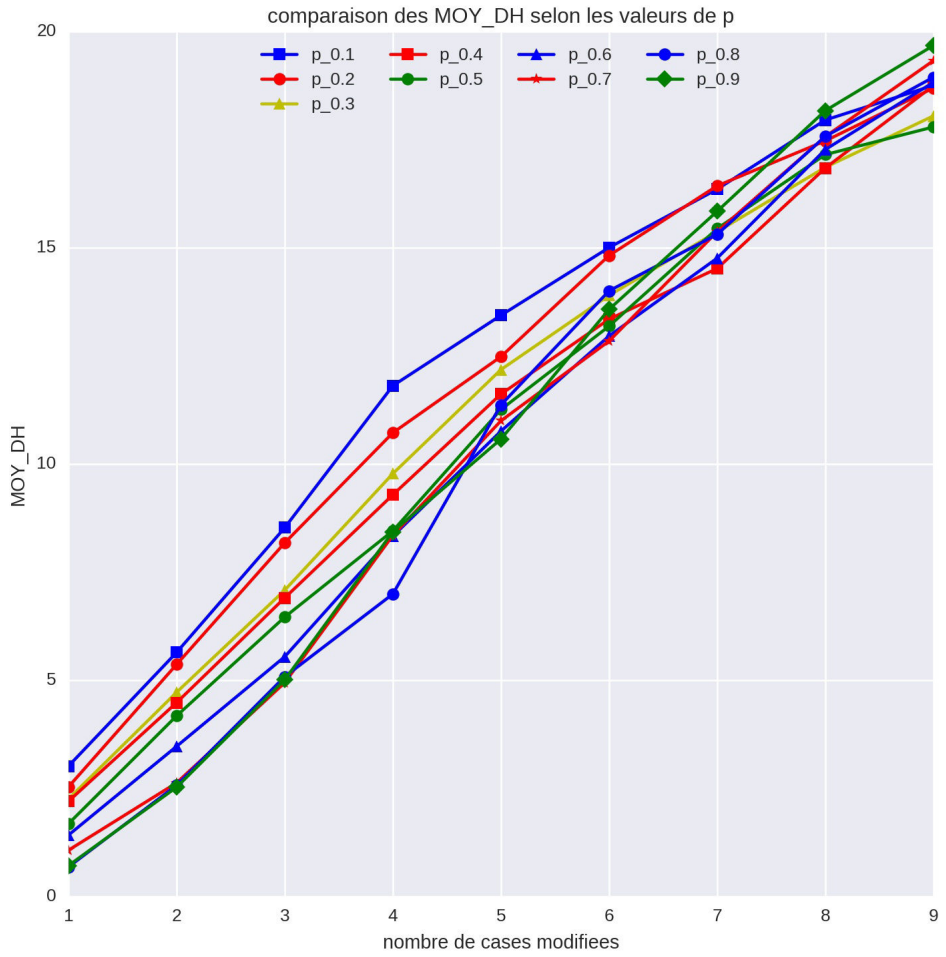

Figure 5.5: Comparaison des différentes repartitions des $k \in \{1, \cdots, 9\}$ cases fausses positives et fausses négatives avec l'approche *aléatoire sans remise* et la fonction de coût *unitaire*.

en moyenne) à la fin de l'algorithme de correction est inférieur au nombre de cases corrigés *fausses négatives* (47.7% en moyenne) avec la fonction *unitaire*. Cette baisse provient majoritairement de la position des sommets à corriger dans $\mathcal{C}$. En effet, il existe des chaines simples de longueur 2 entre certains sommets de $\mathcal{C}$. Une arête de chacune des chaines est supprimée afin que la compression de deux cliques fournisse une nouvelle clique de $\mathcal{CC}$. Cela fait que les arêtes incidentes ont leurs sommets de $\mathcal{CC}$ et ces arêtes sont supprimées une fois sur deux au minimum. La fonction *suppression* donne le même résultat que la fonction *unitaire*.

Dans la figure 5.6 associée à la fonction *ajout*, les courbes divergent quand k est croissant. En effet,

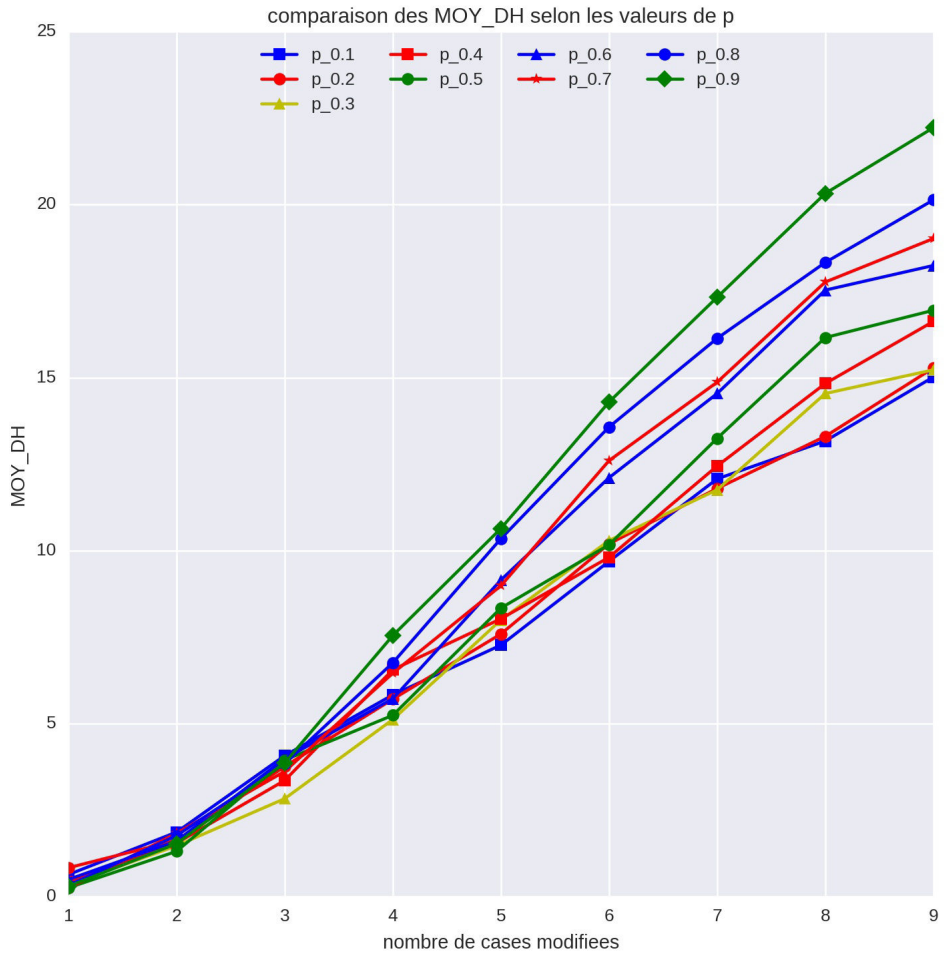

Figure 5.6: Comparaison des différentes repartitions des $k \in \{1, \cdots, 9\}$ cases fausses positives et fausses négatives avec l'approche *aléatoire sans remise* et la fonction de coût *ajout*.

des arêtes *fausses négatives* sont ajoutées à $G_{k,p,\alpha}$ pour $p \geq 0.4$ pendant la correction. Alors les distances $moy\_DH_{k,p}$ augmentent car le nombre de cases *fausses négatives* augmente et aussi plus de 40% des cases *fausses positives* ne sont pas supprimées. Par ailleurs, les courbes convergent vers 0 quand $k \leq 4$. La baisse des valeurs moyennes des distances de Hamming est le résultat de la correction des cases modifiées dans $M_{k,p}$. Néanmoins, l'algorithme corrige majoritairement des cases *fausses négatives* lorsque toutes les cases modifiées ne parviennent pas à être corrigées.

Conclusion : nous ne pouvons pas conclure que les valeurs de p ont une influence sur la correction

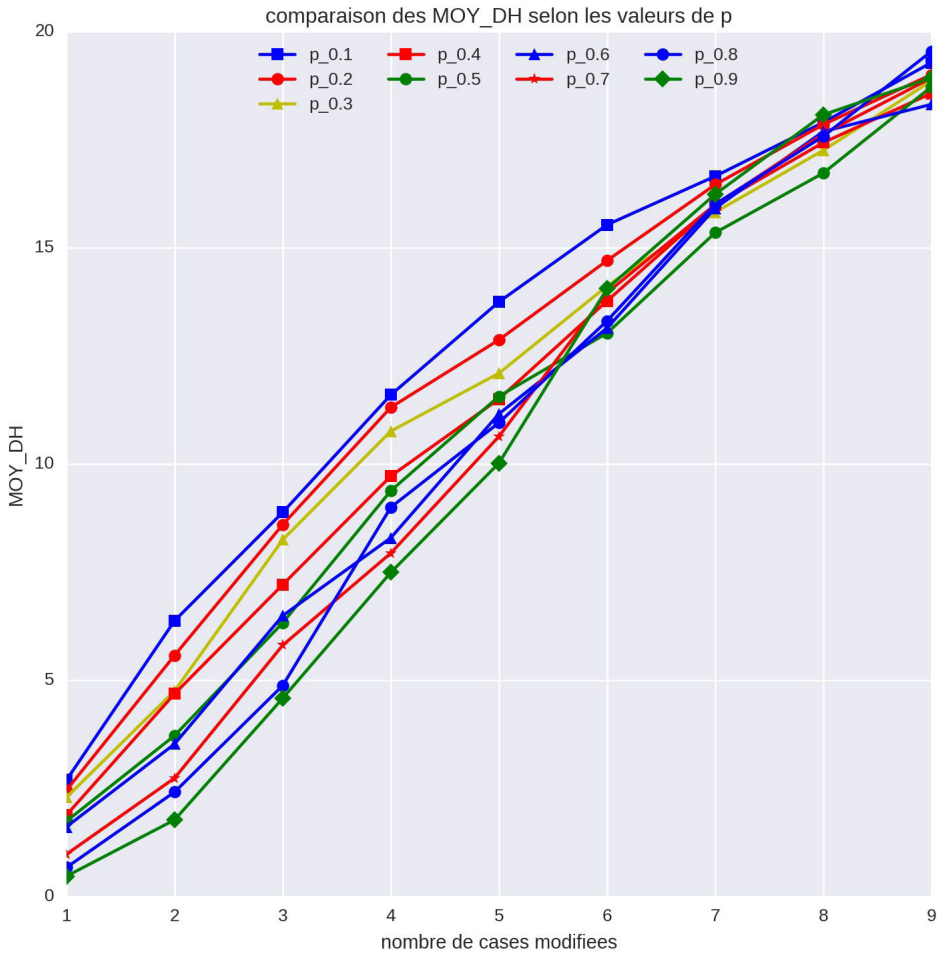

Figure 5.7: Comparaison des différentes repartitions des $k \in \{1, \cdots, 9\}$ cases fausses positives et fausses négatives avec l'approche *aléatoire sans remise* et la fonction de coût *suppression*.

des k cases modifiées car les fonctions de coût *unitaire* et *suppression* font converger l'une vers l'autre leurs distances de Hamming avec l'augmentation du nombre de cases modifiées. Toutefois, la fonction de coût *ajout* fait diverger nos courbes sans créer un écart significatif de distances entre ses courbes.

5.2.3.4 Relation entre la distance de Hamming et la distance de correction

Nous allons calculer les corrélations entre les distances de correction et de Hamming en se basant sur le graphe G, son line-graphe LG et le line-graphe $LG_{k,p,\alpha}$ obtenus par notre couple d'algorithmes. Notre objectif est de montrer la relation entre ces deux distances.

Considérons les distances de correction et de Hamming obtenues par l'approche *aléatoire sans remise*, la variable $p = 0.5$ et la fonction de coût *unitaire* (voir tableau 5.1).

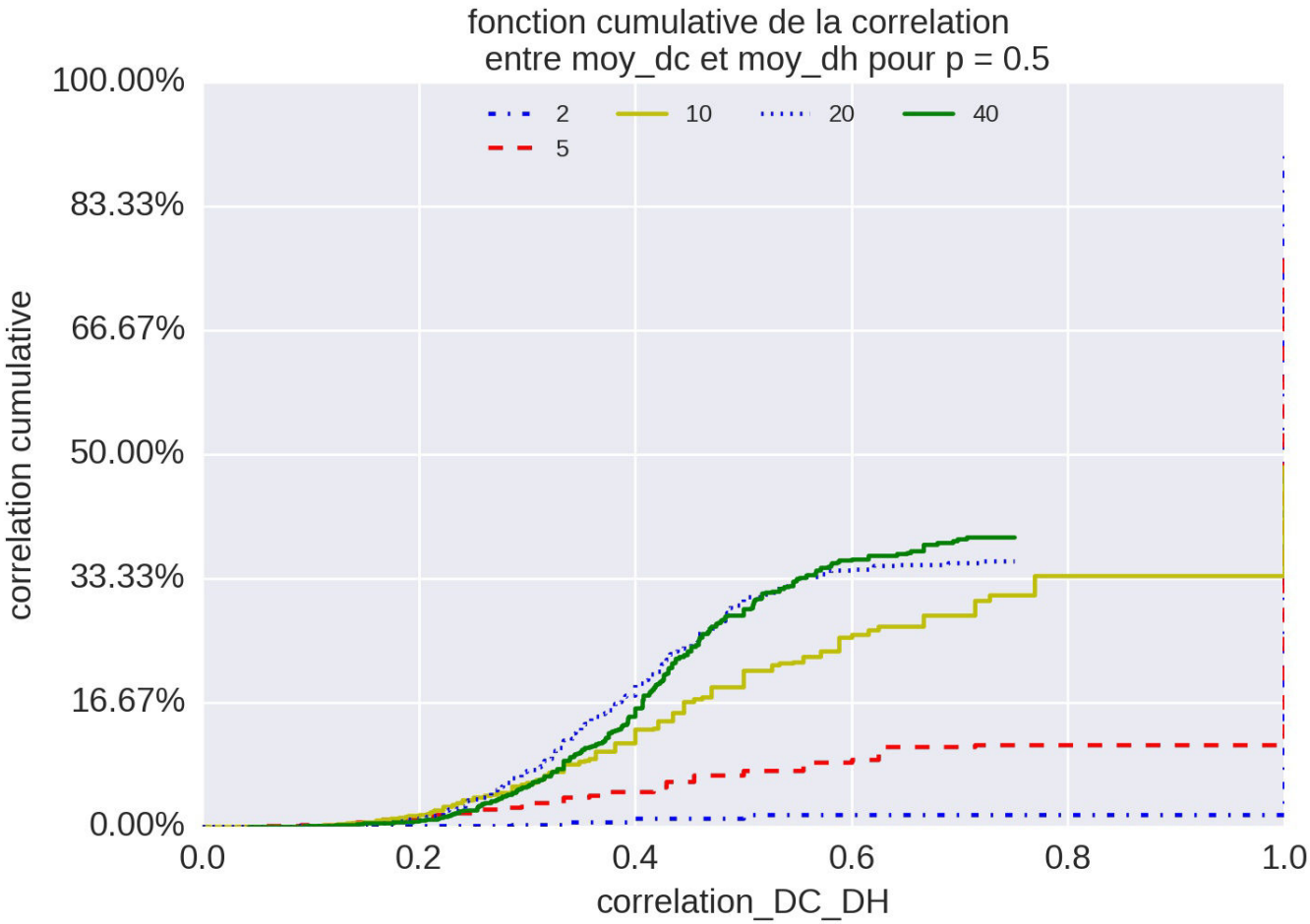

Figure 5.8: La corrélation de la distance de correction versus la distance de Hamming pour k cases modifiées et $p = 0.5$

Nous calculons la corrélation entre les distances de correction et de Hamming avec la formule 5.3.

$$corr_{k,p,\alpha} = \frac{|moy\_DC_{k,p,\alpha} - moy\_DH_{k,p,\alpha}|}{max(moy\_DC_{k,p,\alpha}, moy\_DH_{k,p,\alpha})}; corr_{k,p} = \sum_{\alpha=1}^{5} corr_{k,p,\alpha}; F_k(x,p) = P(corr_{k,p} < x) \quad (5.3)$$

avec $x \in [0,1]$ une valeur de corrélation et k le nombre de cases modifiées. La fonction $corr_{k,p,\alpha}$ retourne l'écart entre ces deux distances sous la forme de valeurs probabilistes. Ainsi $corr_{k,p,\alpha} = 1$ indique qu'il n'existe aucune corrélation entre les distances de correction et de Hamming c'est-à-dire que $moy\_DC_{k,p,\alpha} = k$ et $moy\_DH_{k,p,\alpha} = 0$. De même, $corr_{k,p} = 0$ indique que ces distances sont identiques c'est-à-dire $moy\_DC_{k,p,\alpha} = moy\_DH_{k,p,\alpha}$. En moyenne, $moy\_DH_{k,p,\alpha}$ tend vers k cases modifiées.

La figure 5.8 représente la fonction de repartition F_k dans laquelle nous avons, en abscisse, la corrélation entre les distances et, en ordonné, le pourcentage de graphes dont la corrélation moyenne $corr_{k,p}$ est inférieure à $x \in [0,1]$. Si $corr_{k,p,\alpha} = 0$ alors les matrices de $LG_{k,p,\alpha}$ et LG sont différentes de k cases quand $k < 6$ ($LG_{k,p,\alpha} \neq LG$). Si $corr_{k,p,\alpha} = 1$ alors le line-graphe $LG_{k,p,\alpha}$ est le line-graphe du graphe G ($LG_{k,p,\alpha} = LG$) et $F_k(1) \approx 0$. La corrélation $corr_{k,p} = 1$ est sa valeur maximale. Ce cas est illustré dans la figure 5.8 par les courbes de $k \in \{2,5\}$. Par exemple $F_5(1) \approx 10\%$ signifie que nous avons $70 - 10 = 60\%$ des line-graphes LG_k qui ont le même ensemble d'arêtes que les line-graphes LG (70% est le pourcentage de corrélations égales à 1 $|corr_{5,p} = 1| = 70\%$).
En revanche, une valeur de $F_k(1)$ très élevée signifie que le nombre x de $corr_{k,p} = 1$ est très faible. Ce nombre x implique une corrélation très forte entre les distances de correction et de Hamming. C'est l'observation faite avec les courbes de $k \in \{10, 20, 40\}$ de la figure 5.8 dans lesquelles nous avons une croissance continue en fonction de l'augmentation des valeurs de corrélations.
Nous subdivisons nos courbes en deux catégories:

- Celles pour lesquelles il y a une corrélation entre les distances de correction et de Hamming (courbes de $k \in \{10, 20, 40\}$).
- Celles pour lesquelles il y a aucune corrélation entre ces distances parce que $LG = LG_{k,p}$ (courbes de $k \in \{2,5\}$).

Conclusion : il existe de fortes corrélations entre les distances de correction et de Hamming lorsque $k \geq 10$. Dans ce cas, nous pouvons utiliser la distance de correction pour calculer les écarts de cases modifiées pendant l'algorithme de correction. Dans le cas où $k \leq 5$, les distances de correction inférieures ou égales à 5 cases proposent le line-graphe LG et ces distances entrainent une corrélation proche de 0. Pour $k \in \{6,7,8,9\}$, les valeurs de $corr_{k,p}$ avoisinent 0.5. Ces valeurs sont faibles et nous ne pouvons rien conclure. Ainsi, pour juger de la qualité de notre algorithme de correction en l'absence de la distance de Hamming, la distance de correction est alors une bonne métrique.

5.2.4 Conclusion de l'expérimention 1

Les performances de notre couple d'algorithmes ont été testées sur des graphes non orientés sans circuits qui représentent la topologie de réseaux électriques. Nous avons construit les line-graphes de ces graphes et avons modifié k cases dans les matrices des line-graphes. Les cases modifiées sont reparties en deux ensembles (cases *fausses positives* et *fausses négatives*) selon la variable $p \in [0,1]$. L'analyse des performances compare les distances de correction et de Hamming en fonction du nombre de cases modifiées selon 5 approches de correction et 3 fonctions de coût (voir tableau 5.1). Nous

concluons que l'approche *aléatoire sans remise* donne de meilleurs résultats lorsque le nombre k de cases modifiées est faible ($k \leq 5$). La distance line est alors majorée par la distance de correction. Le problème *Proxi-Line* a une solution mais elle n'est pas optimale. En revanche, au-délà de $k > 5$, il est alors difficile de déterminer une borne supérieure à la distance line car l'algorithme de correction modifie des cases n'étant pas contenues dans les k cases modifiées préalablement. Par ailleurs, nous avons montré que la distance de correction peut être utilisée comme une métrique pour connaitre le nombre de cases modifiées dans la matrice d'un graphe à condition que cette distance soit supérieure à 10 arêtes. Enfin, les fonctions de coût ont peu d'influences sur les distances de correction pendant la phase de correction.

5.3 Expérimentation 2 : Ajout de probabilité dans la matrice du line-graphe

Notre seconde expérimentation a le même but que la première sauf qu'ici la modification des cases est faite à partir de valeurs de seuils appliquées à des matrices de corrélation. Les valeurs de ces matrices sont définies selon la distribution des valeurs de corrélation d'un réseau réel, celui du data center *Champlan*.
Notre but est de déterminer la valeur de seuil qui minimise les distances de correction et de Hamming obtenues avec l'approche *aléatoire sans remise* (tableau 5.1).
Nous décrivons tout d'abord l'affectation des valeurs de corrélation aux cases de la matrice du line-graphe. Ensuite nous présentons les différentes étapes de notre expérimentation et enfin nous analysons les résultats obtenus.

5.3.1 Affectation de probabilités aux cases de la matrice M_{LG}

Soient la matrice M_{LG} d'adjacence du line-graphe LG du DAG non orienté de *Champlan* et M_c sa matrice de corrélation obtenue avec la *distance de Pearson*. La distribution de valeurs de M_c est représentée par la figure 5.9. Le graphique à gauche de la figure 5.9 est la distribution des valeurs de M_c selon les cases à 1 dans M_{LG} et le graphique à droite est celui des cases à 0 dans M_{LG}. Par exemple, la valeur de corrélation 0.1 désigne toutes les valeurs appartenant à $[0.1, 0.2[$.
La densité des corrélations est décroissante pour les cases à 0 quand les valeurs de corrélation tendent vers 1. De même, cette densité est croissante pour les cases à 1 quand les corrections tendent vers 1. La matrice M_c contient des cases erronées et ces cases ont une influence sur le calcul des valeurs de corrélation. C'est pourquoi les distributions ne sont pas linéairement croissantes ou décroissantes.

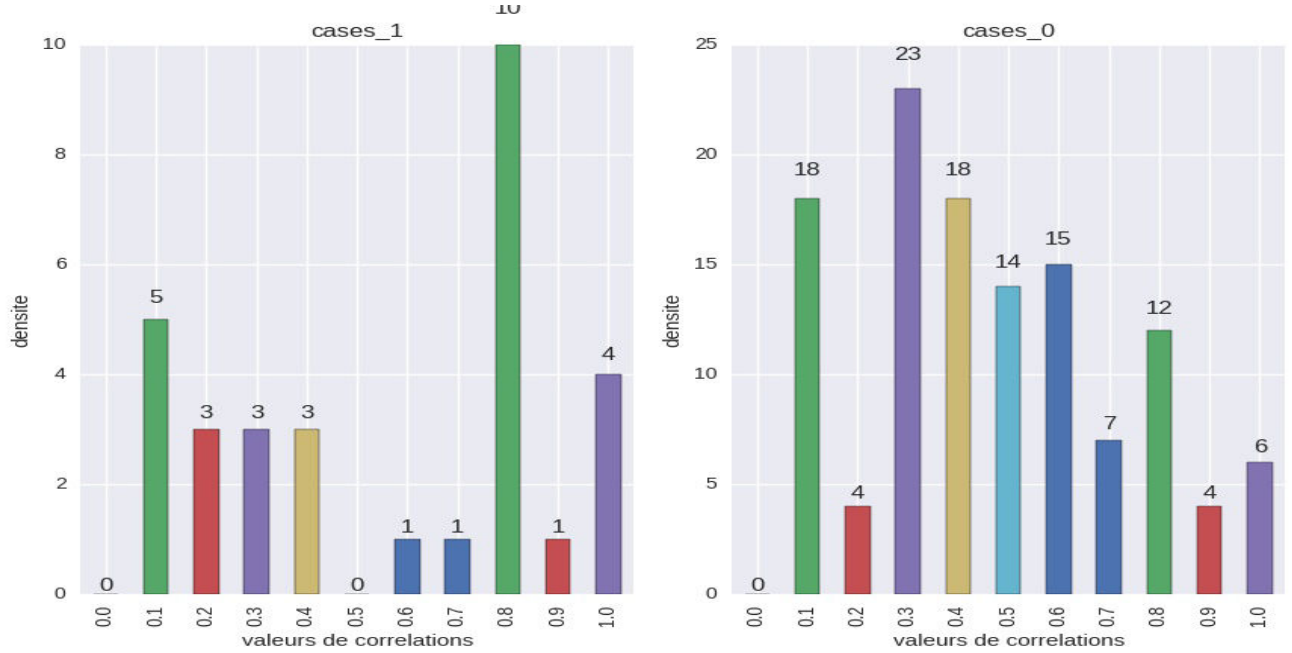

Figure 5.9: Distribution des valeurs de corrélation de M_c selon les cases à 0 (à gauche) et à 1 (à droite) de M_{LG}. La corrélation à 0.1 désigne les valeurs de corrélation comprises entre 0.10 et 0.199.

Nous décidons, avec nos constatations, que la génération des distributions des valeurs de corrélation des cases à 0 et 1 suivent des lois normales asymétriques.

Nous générons des valeurs de corrélation qui sont similaires à la distribution conjecturée des valeurs de corrélation du graphe de *Champlan*. Nous supposons que les valeurs de corrélation des cases à 0 de M_{LG} suivent la loi asymétrique de paramètre $\alpha = 5$ et les valeurs de corrélation des cases à 1 de M_{LG} suivent cette même loi avec $\alpha = -5$ comme illustré dans la figure 5.10.

Soient n_0 la cardinalité de l'ensemble des cases à 0 de M_{LG} et n_1 la cardinalité de l'ensemble des cases à 1 de M_{LG}. Pour obtenir des valeurs de corrélation de M_c, nous générons des valeurs réelles val appartenant à $[0, 1]$ selon la loi asymétrique de coefficient $\alpha = 5$. Nous divisons l'intervalle $[0, 1]$ en 10 sous-intervalles. Chaque sous-intervalle est noté $]x_i, x_{i+1}]$ avec x_i une valeur de corrélation et $i \in [0, 9]$. Nous calculons la densité de chaque sous-intervalle et l'ensemble des densités forme un histogramme qui suit une loi asymétrique. Nous calculons la fonction de repartition P de chaque densité de telle sorte que

$$\forall x_i, x_{i+1}, P(X \leq x_i) \leq P(X \leq x_{i+1}) \quad et \quad P(X \leq x_{10}) = 1.$$

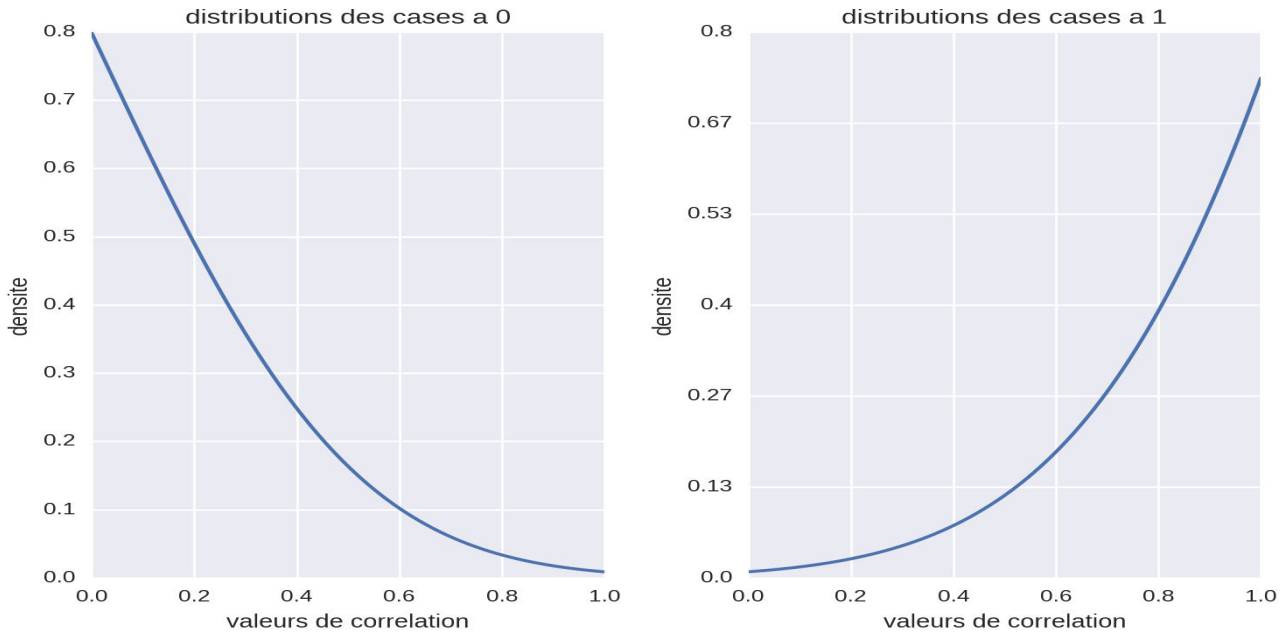

Figure 5.10: Distribution des cases à 0 et à 1. À gauche : loi asymétrique de coefficient d'asymétrie $\alpha = 5$ pour les cases à 0. À droite : loi asymétrique de coefficient d'asymétrie $\alpha = -5$ pour les cases à 1.

Pour chaque case à 0, nous tirons aléatoirement n_0 nombres réels compris entre 0 et 1 uniformement. Si un nombre appartient à $]P(X \leq x_i), P(X \leq x_{i+1})]$ alors sa valeur de corrélation est x_{i+1}. Nous répétons les mêmes étapes pour les cases à 1 en générant les valeurs *val* avec une loi asymétrique de coefficient $\alpha = -5$. Nous obtenons la matrice de corrélation M_c du line-graphe LG. Une distribution des valeurs de corrélation est représentée par la figure 5.11 dans laquelle nous avons 189 cases à 0 et 111 cases à 1 dans LG.

5.3.2 Génération du graphe de corrélation

Nous allons déterminer la matrice d'adjacence du graphe de corrélation à partir des valeurs de M_c. Soit $s = \{0.1, \cdots, 0.9\}$, une valeur de seuil choisie. Nous construisons la matrice M_s selon les règles suivantes :

- Si $M_c[i, j] \geq s$ alors $M_s[i, j] = 1$.
- Si $M_c[i, j] < s$ alors $M_s[i, j] = 0$.

La matrice M_s est la matrice d'adjacence du graphe G_s dit *graphe de corrélation*. Ces cases peuvent contenir des cases erronées. Ces cases erronées proviennent de la sélection du seuil s et de la génération

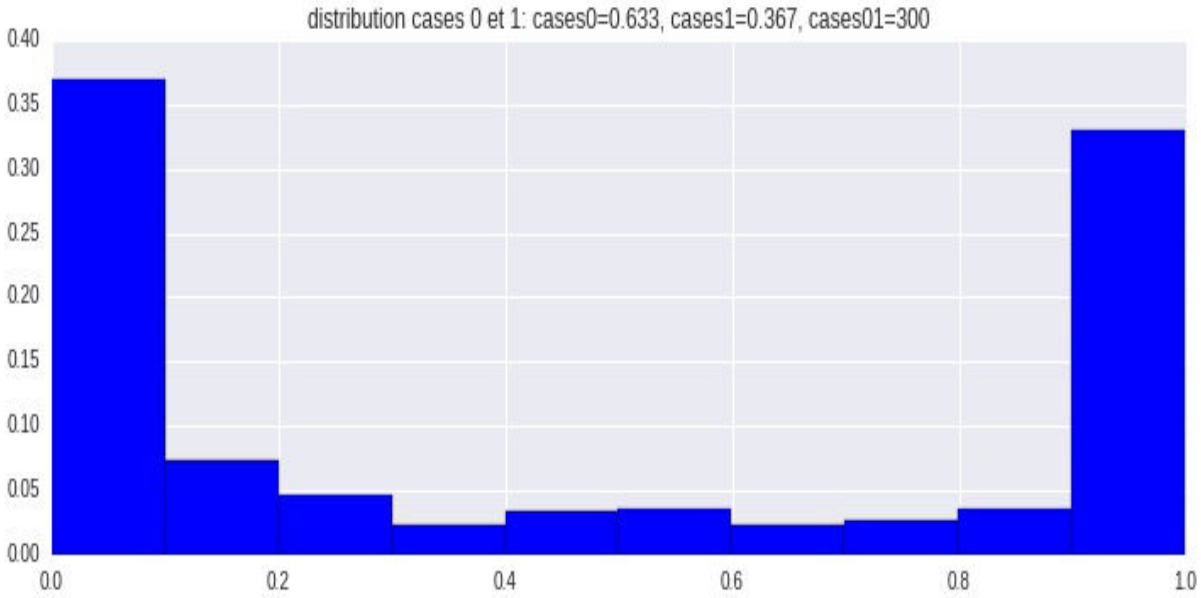

Figure 5.11: Un exemple de la distribution des valeurs de corrélations générées. 63% des cases sont des cases à 0 et 37% sont des cases à 1 dans LG.

de valeurs de corrélation pour les cases à 0 et à 1 du line-graphe LG. En effet ,

- Si $M_s[i,j] = M_{LG}[i,j] = 0$ alors $M_s[i,j]$ est dit *vrai négatif.*
- Si $M_s[i,j] = M_{LG}[i,j] = 1$ alors $M_s[i,j]$ est dit *vrai positif.*
- Si $M_s[i,j] = 0$ et $M_{LG}[i,j] = 1$ alors $M_s[i,j]$ est dit *faux négatif.*
- Si $M_s[i,j] = 1$ et $M_{LG}[i,j] = 0$ alors $M_s[i,j]$ est dit *faux positif.*

Un exemple de distributions des cases erronées selon les valeurs de seuils est présenté dans la figure 5.12. Par exemple, le graphe de corrélation G_s contient 17 cases *vrais positives*, 151 cases *vrais négatives*, 75 cases *fausses positives* et 57 cases *fausses négatives* pour un seuil $s = 0.4$.

5.3.3 Protocole d'expérimentation des graphes G_s de corrélation

Le but de notre couple d'algorithmes est de corriger les cases erronées dans M_s afin que la matrice proposée M'_s soit la matrice d'adjacence d'un line-graphe LG_s et que la distance de Hamming entre LG_s et LG soit minimale. Pour ce faire, nous recherchons la valeur du seuil s qui mininise la distance de Hamming entre LG_s et LG.

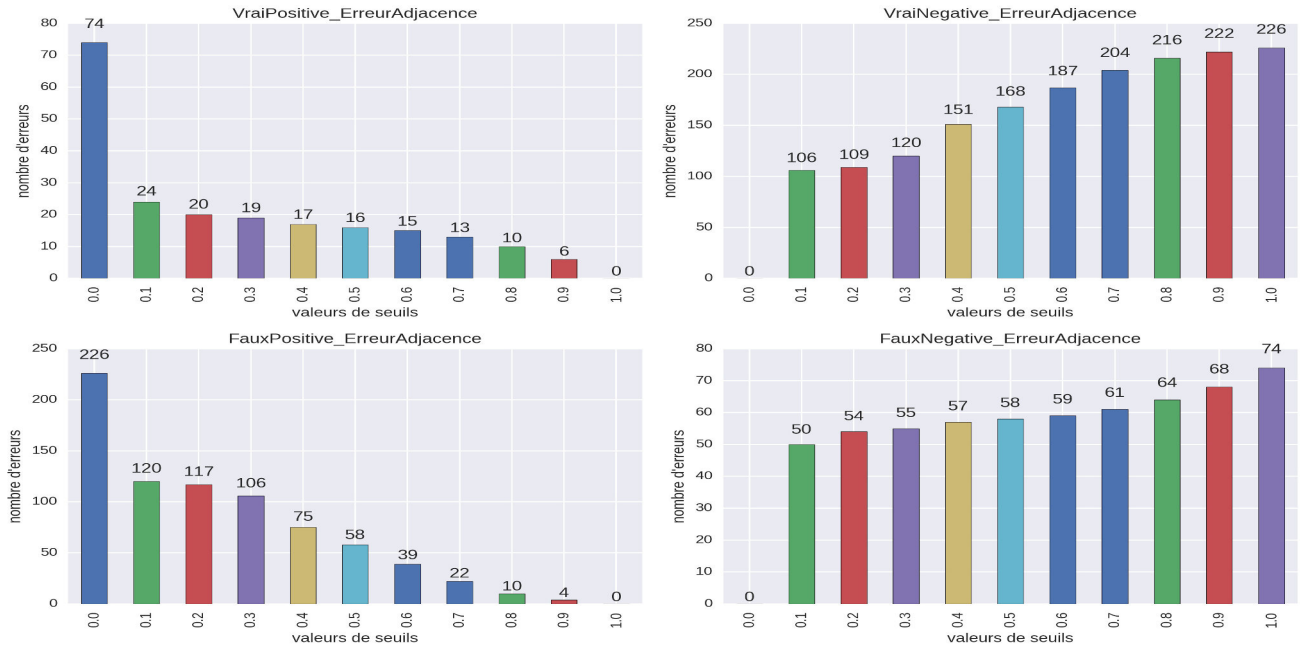

Figure 5.12: Distribution des valeurs de corrélation sur un graphe généré de 30 sommets et de degré maximal de 5. .

Nous générons les graphes dans les mêmes conditions que l'expérimentation 1 de la section 5.2 avec de petites modifications. D'abord, le nombre de graphes générés est de 150. Ensuite, nous construisons une matrice de corrélation dont les étapes sont décrites dans la section 5.3.1. Et enfin, l'ajout des cases erronées est réalisé à partir d'un seuil s dans la matrice de corrélation M_c comme cela est expliqué dans la section 5.3.2. Les étapes de l'expérimentation sont résumées dans la figure 5.13.

Nous considérons que l'approche *aléatoire sans remise* (tableau 5.1) pendant l'algorithme de correction. Mais les fonctions de coûts des arêtes utilisent les valeurs de corrélation comme suit:

(a) *Unitaire* : l'ajout et la suppression d'une arête ont un coût de 1.

(b) *Normale* : l'ajout d'une arête à la case $M_s[i, j]$ a un coût égal à sa valeur de corrélation $M_c[i, j]$ et la suppression d'une arête a un coût $1 - M_c[i, j]$.

(c) *Ajout* : l'ajout d'une arête à la case $M_s[i, j]$ a un coût $M_c[i, j]$ alors que la suppression à cette case a un coût $2 \times (1 - M_c[i, j])$.

(d) *Suppression* : la suppression d'une arête à la case $M_s[i, j]$ a un coût $1 - M_c[i, j]$ alors que l'ajout d'une arête à cette case a un coût $2 \times M_c[i, j]$.

Nous allons comparer les distances de Hamming et le pourcentage de cases corrigées pour en déduire la bonne valeur de seuil.

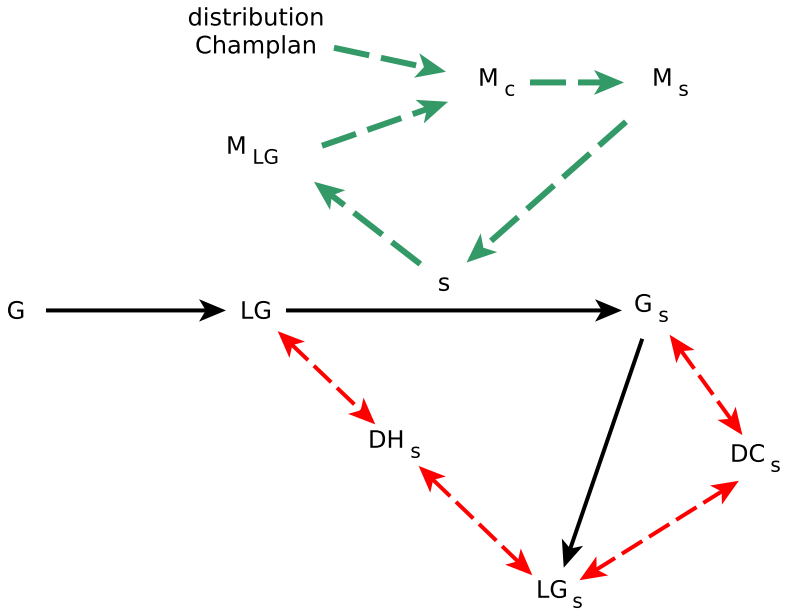

Figure 5.13: Étapes de l'expérimentation : 1) on génère le graphe G et son line-graphe LG, 2) on génère la matrice de corrélation M_c du line-graphe LG à partir de la distribution des valeurs de corrélation du graphe de Champlan puis on lui applique une valeur de seuil s pour obtenir le graphe G_s, 3) on applique les algorithmes de découverte et de correction pour avoir un line-graphe LG_s. LG_s et G_s diffèrent de DC_s arêtes. LG_s a DH_s cases modifiées par rapport à LG.

5.3.4 Analyses des résultats

Nous allons décrire l'évolution du pourcentage des cases corrigées en fonction de la valeur du seuil pour la fonction de coût *normale*. Ensuite nous déterminons l'influence de la valeur de seuil sur le nombre de cases corrigées et enfin nous recherchons la meilleure fonction de coût et l'influence des fonctions de coût sur l'évolution des distances de Hamming.

5.3.4.1 Évolution du pourcentage de cases corrigées

Pour mesurer le pourcentage de cases corrigés, nous allons procéder comme suit :

(a) Considérer les cases *fausses positives* dans la matrice M_s puis représenter le nombre de cases *fausses positives* pour chaque valeur de seuil. Le graphique (a) de la figure 5.14 correspond à la comparaison des seuils par rapport au nombre de cases *fausses positives* avant la correction. Nous remarquons qu'il n'y a aucune case *fausse positive* dans M_s pour $s \in \{0.8, 0.9\}$. Ce nombre croit quand le seuil s decroit ($s \to 0$).

(b) Considérer les cases *fausses négatives* dans la matrice M_s puis représenter le nombre de cases *fausses négatives* pour chaque valeur de seuil. Le graphique (b) de la figure 5.14 correspond à la comparaison des seuils par rapport au nombre de cases *fausses négatives* avant la correction. Le nombre des cases *fausses négatives* est nul pour $s \notin \{0.8, 0.9\}$.

(c) Considérer les cases *fausses positives* après la correction de G_s (matrice M'_s) puis représenter le nombre de cases *fausses positives* pour chaque valeur de seuil. Le graphique (c) de la figure 5.14 correspond à la comparaison des seuils par rapport au nombre de cases *fausses positives* après la correction. Le nombre de cases baisse quand $s \leq 0.7$ puis augmente $s > 0.7$.

(d) Considérer les cases *fausses négatives* après la correction de G_s (matrice M'_s) puis représenter le nombre de cases *fausses négatives* pour chaque valeur de seuil. Le graphique (d) de la figure 5.14 correspond à la comparaison des seuils par rapport au nombre de cases *fausses négatives* après la correction. Le nombre de cases varie peu quand $s < 0.4$ puis baisse quand $s = \{0.5, 0.6\}$ avant d'atteindre sa valeur minimum à $s = 0.7$. Il augmente $s > 0.7$.

(e) Représenter les distances de Hamming moyennes de chaque graphe pour chaque seuil. Le graphique (e) de la figure 5.14 correspond à la comparaison des seuils en fonction de la distance de Hamming de chaque seuil. La distance de Hamming baisse quand $s \to 0.7$ avec sa valeur minimum à $s = 0.7$ puis augmente quand $s > 0.7$.

Rappelons que les différents graphiques sont rangés par ordre croissant et les distances de Hamming sont obtenues à partir la fonction de coût *normale*.

Nous distinguons trois familles de seuils:

- Les seuils $s < 0.7$ qui baissent le nombre de cases *fausses positives* et augmentent celui des cases *fausses négatives* après l'algorithme de correction. Ils n'ont pas d'effets réels sur la distance de Hamming car il y a un transfert d'éléments de l'ensemble des cases *fausses positives* à celui des *fausses négatives* et vice-versa. À cet effet, on remarque qu'il y a 20% de cases *fausses négatives* après la correction alors qu'il n'en existait aucune case *fausse négative* avant la correction. Il en est de même avec les cases *fausses positives* dont le nombre diminue de 20% également pendant la correction. Ces seuils n'ont aucune influence sur les cases erronées.

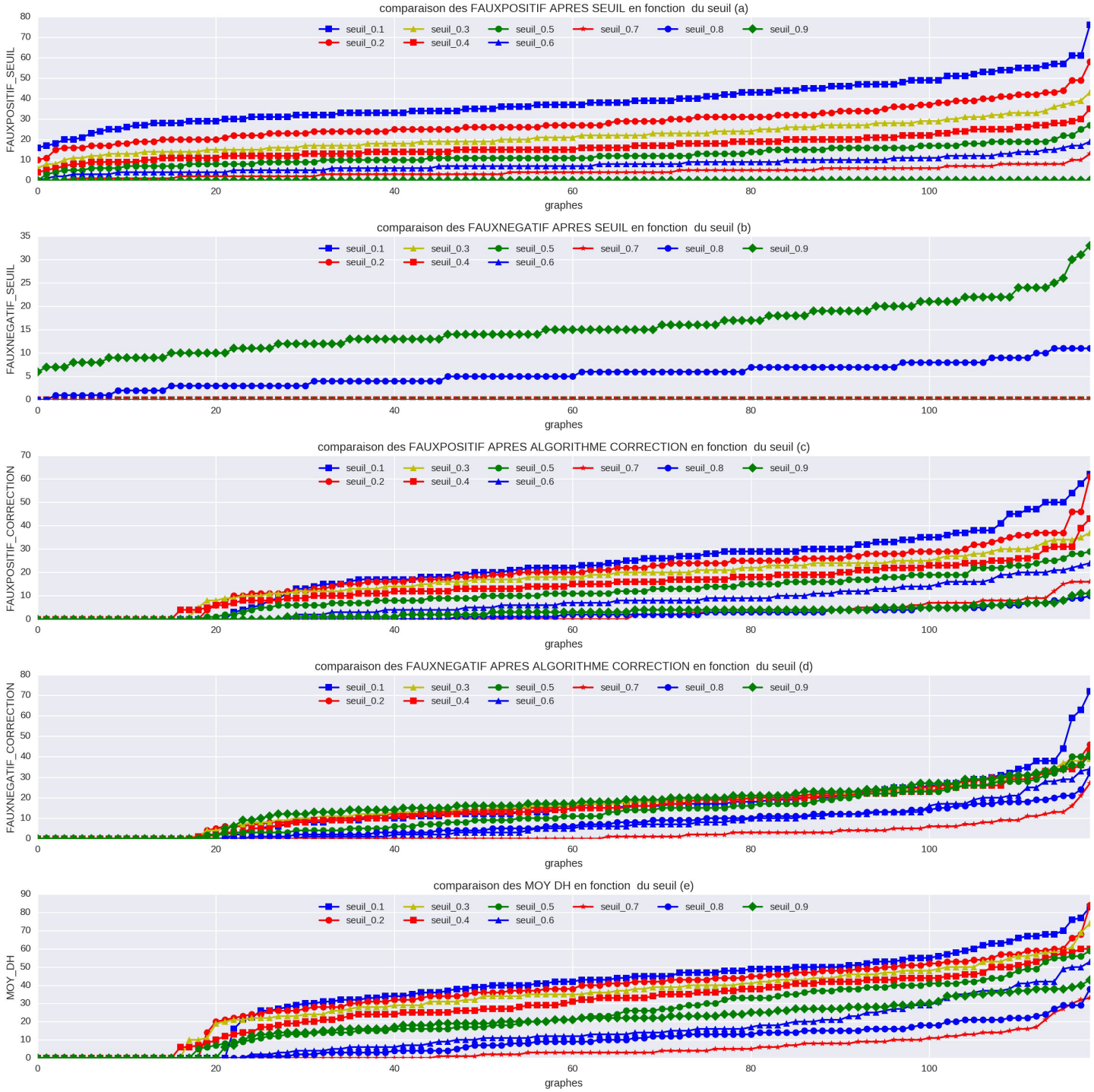

Figure 5.14: Choix du seuil : (a) cases *fausses positives* dans la matrice M_s; (b) cases *fausses négatives* dans la matrice M_s,(c) cases *fausses positives* dans la matrice M'_s; (d) cases *fausses négatives* dans la matrice M'_s; (e) comparaison des seuils selon *moy_DH*

.

- Les seuils $s > 0.7$ qui augmentent le nombre de cases *fausses positives* et baissent celui des cases *fausses négatives* après l'algorithme de correction. En effet, le nombre de cases *fausses positives* est nul dans M_s parce qu'il n'y a aucune case à 1 ayant une valeur inférieure à s dans la distribution. Dans M'_s, le nombre moyen de cases *fausses positives* est de 1.76 pour $s = 0.8$ et de 2.33 pour $s = 0.9$. Il y a l'ajout d'arêtes dans le graphe LG_s parce que le nombre d'arêtes à supprimer pour corriger un sommet est très élevé et cela implique que le coût de la correction par la suppression d'arêtes est très onéreux par rapport à celui de l'ajout d'arêtes. Ainsi à chaque sommet à corriger, l'algorithme ajoute plus d'arêtes qu'il en supprime et certaines arêtes ajoutées appartiennent à l'ensemble des arêtes de LG. Cela fait baisser les cases *fausses négatives* comme nous le constatons avec les chiffres suivants : le nombre moyen de ces cases passe de 6.7 à 2.5 pour $s = 0.8$ et de 17.7 à 12.5 pour $s = 0.9$.

- Le seuil $s = 0.7$ qui diminue le nombre de cases *fausses négatives* et *fausses positives*. En effet, le nombre moyen de cases erronées est faible < 10 avant la phase de correction et il est ≤ 5 après la correction. Nous remarquons aussi les cases erronées sont majoritairement des cases *fausses négatives* après la correction. La présence de ces cases provient du coût de modification des arêtes car la compression π_1, π_2, π_s de coût minimale nécessite la suppression d'arêtes existantes. En effet, la matrice M_s ne contient que des cases *fausses positives*. Pour trouver des bipartitions cohérentes (voir défintion 5) autour des sommets à corriger, il faut ajouter beaucoup d'arêtes pour chaque partition et cela fait croitre le coût de la correction. Nous avons remarqué aussi que la suppression de quelques arêtes permet d'obtenir des cliques qui correspondent à des sommets de G. L'algorithme préfère alors supprimer des arêtes car elles sont peu par rapport aux arêtes à ajouter et aussi le coût de la suppression d'arêtes est faible. La distance de Hamming obtenue avec ce seuil est minimale par rapport aux autres seuils. Ce seuil fournit alors une meilleure correction des sommets de $\mathcal{C}$.

Conclusion : nous pouvons conclure que la repartition des cases erronées avant la phase de correction a une influence sur l'algorithme de correction. Ainsi, le choix du seuil dans le bon intervalle permet de réduire le nombre de cases erronées et fournit une excellente correction des sommets de $\mathcal{C}$. Cependant, les seuils différents du bon seuil entraine que la fonction de coût n'a aucune influence sur les corrections parce que l'algorithme corrige peu de cases erronées mais ajoute aussi des cases *fausses positives* et *fausses négatives* dans la même proportion.

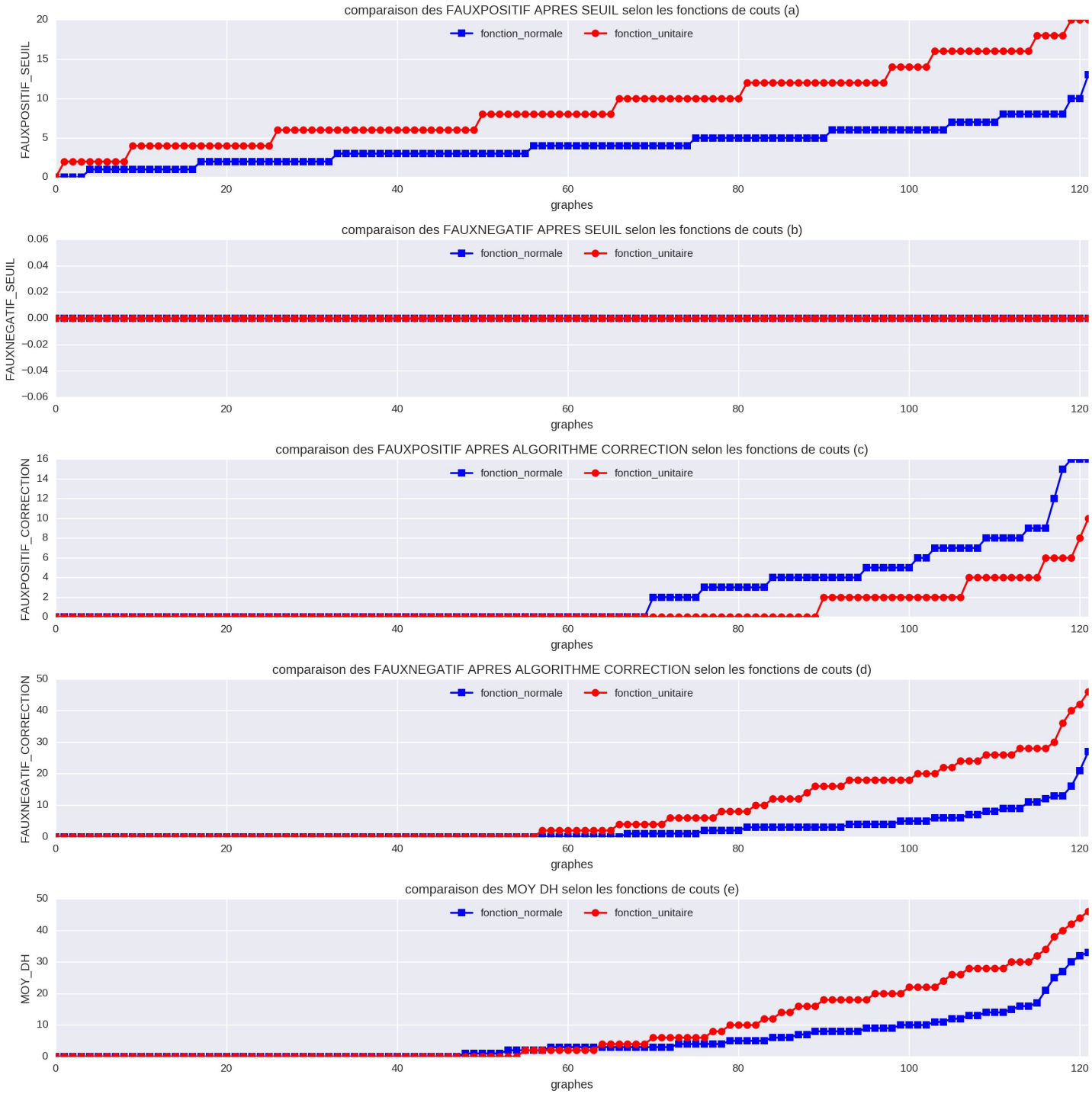

Figure 5.15: Comparaison entre les fonctions de coût *unitaire* et *normale*: (a) cases *fausses positives* dans la matrice M_s; (b) cases *fausses négatives* dans la matrice M_s; (c) cases *fausses positives* dans la matrice M'_s; (d) cases *fausses négatives* dans la matrice M'_s; (e) comparaison des fonctions de coût selon *moy_DH*.

5.3.4.2 Influence de la valeur du seuil

Les seuils inférieurs à 0.7 correspondent à une reduction des cases *fausses positives* et une augmentation de cases *fausses négatives* dans la matrice de correction. Dans ce cas, nous constatons que le nombre des cases *fausses positives* baisse énormement quand $s \to 0.7$. Ce phénomène s'explique par le coût de modifications des arêtes (normale) et le nombre faible de cases erronées au voisinage de 0.7. L'augmentation des cases *fausses négatives* provient du fonctionnement de notre algorithme de correction. L'algorithme doit ajouter des arêtes dans une partition π_1 *ou* π_2 au voisinage d'un sommet pour en faire une clique. Le nombre élevé de cases *fausses négatives* nécessite l'ajout de beaucoup d'arêtes.
Les seuils supérieurs à 0.7 correspondent à l'augmentation des cases *fausses positives* et la baisse de celles *fausses négatives*. La présence de cases *fausses positives* en grand nombre entraine l'algorithme de correction dans deux cas : l'ajout de peu d'arêtes et la suppression de beaucoup d'arêtes pour atteindre un line-graphe. Dans le premier cas, la distance de Hamming est faible mais le line-graphe est différent du line-graphe de G_s. Dans ce second cas, la distance de Hamming est très élevée et nous avons très peu de chance de retrouver le line-graphe de G_s.

Conclusion : le meilleur compromis de seuil est celui qui baisse les cases *fausses positives* et les cases *fausses négatives* après l'algorithme de correction. Le seuil capable d'atteindre ce résultat est dans l'intervalle $s =]0.6, 0.7]$. Dans la suite du chapitre, nous retenons $s = 0.7$. Avec ce seuil, les distances de Hamming sont aussi les plus faibles (graphique (e) figure 5.14).

5.3.4.3 Choix de la fonction de coût et impact sur les distances de Hamming

Nous rappelons que la fonction de coût est définie par la somme des coûts des cases à modifier pour corriger chaque sommet de $\mathcal{C}$ pendant l'algorithme de correction. Le coût de correction d'un sommet est le coût minimal de toutes les cases modifiées. Nous recherchons alors la fonction de coût qui minimise globalement le coût de correction des sommets de $\mathcal{C}$. Pour ce faire, nous comparons d'abord les fonctions de coût *unitaire* et *normale* parce que nous souhaitons savoir s'il est préférable d'utiliser les valeurs de corrélations dans les coûts des opérations. Puis nous comparons la meilleure des deux fonctions avec celles *ajout* et *suppression*. La figure 5.15 contient

- La comparaison des distances de Hamming entre les fonctions de coût *unitaire* et *normale* (graphique (e)).
- La comparaison du nombre de cases *fausses négatives* entre les deux fonctions de coût avant l'algorithme de correction (graphique (b)).

- La comparaison du nombre de cases *fausses négatives* entre les deux fonctions de coût après l'algorithme de correction (graphique (d)).

- La comparaison du nombre de cases *fausses positives* entre les deux fonctions de coût avant l'algorithme de correction (graphique (a)).

- La comparaison du nombre de cases *fausses positives* entre les deux fonctions de coût après l'algorithme de correction (graphique (a)).

Les distances de Hamming et les nombres de cases sont ordonnées par ordre croissant.

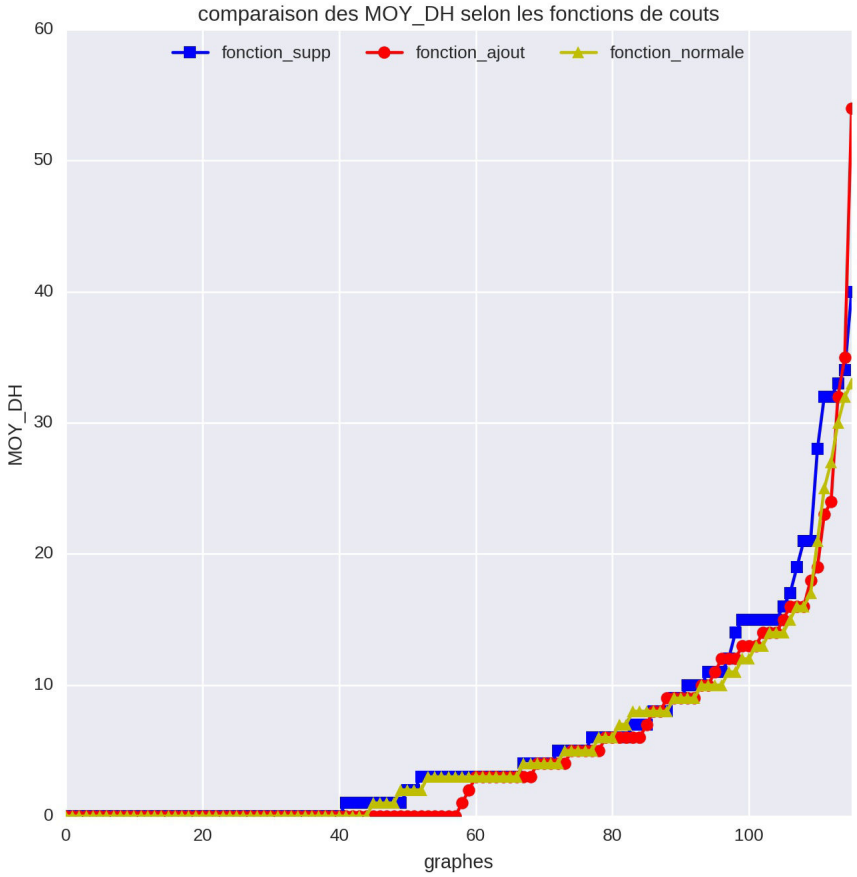

Figure 5.16: Comparaison entre les fonctions de coût *normale*, *ajout* et *suppression* : la fonction *ajout* est la courbe en rouge, la fonction *normale* est en jaune et la fonction *suppression est en bleu*

Nous constatons que la courbe de la fonction *unitaire* est au dessus de celle de la fonction *normale* dans le graphique (e) de la figure 5.15. Avec $s = 0.7$, l'ensemble de cases erronées sont des cases *fausses*

positives (graphique (c) de la figure 5.15). En appliquant les fonctions *unitaire* et *normale*, nous avons l'introduction de cases *fausses négatives* et le nombre de ces cases est plus important dans la fonction *unitaire*. Ce nombre fait croître la distance de Hamming moy_DH parce que la correction a réduit le nombre de cases *fausses positives* dans la fonction *normale* (voir les graphiques (a) et (c) de la figure 5.15). Ce qui explique les faibles distances moy_DH de la fonction *normale*. La fonction *normale* donne de meilleurs résultats et nous la choisissons dans le calcul des coûts de correction.

Par ailleurs, dans la figure 5.16, les courbes des fonctions *normale*, *ajout* et *suppression* sont entremelées et il ne se dégage aucun écart significatif entre elles. Il est donc difficile de juger l'influence d'une des fonctions sur la correction des sommets de $\mathcal{C}$.

Conclusion : prioriser l'ajout à la suppression et vice versa n'a aucune influence sur les distances de Hamming quand nous utilisons les valeurs de corrélations. Toutefois, il est préférable de considérer les corrélations dans le calcul du coût de la modification d'une case parce que cela améliore les distances moy_DH comme il est indiqué dans le graphique (e) de la figure 5.15.

5.3.5 Conclusion de l'expérimentation 2

Nous avons généré des valeurs de corrélations pour toutes les cases de la matrice M_{LG} en considérant la distribution des valeurs de corrélation du réseau électrique du data center *Champlan*. Les valeurs de corrélation suivent des lois normales asymétriques de paramètre $\alpha = 5$ pour les cases à 0 et de paramètre $\alpha = -5$ pour les cases à 1. La matrice de corrélation M_c est construite à partir de ces corrélations. Un ensemble $s \in S$ de seuils est appliqué à la matrice M_c pour la transformer en la matrice d'adjacence M_s du graphe G_s. La matrice M_s contient des cases *fausses négatives* et *fausses positives*. Nous cherchons à minimiser la distance de Hamming en corrigeant le maximum de cases erronées pendant l'algorithme de correction. Cela passe par la sélection adéquate du seuil et de la fonction de coût. Après l'exécution de notre couple d'algorithmes, nous avons déduit que le bon seuil est contenu dans l'intervalle $s =]0.6, 0.7]$ et que la fonction *normale* est la meilleure fonction de coût. L'utilisation du seuil s et de la fonction *normale* ne garantissent pas la suppression totale des cases erronées. Mais elles minimisent leur nombre de telle sorte qu'un expert du métier puisse effectuer les corrections manuellement qui conduisent au line-graphe LG recherché. Dans la section suivante, nous nous intéressons aux graphes dans lesquels tous les sommets ne peuvent être couverts par 1 ou 2 cliques. Ces graphes sont dits *grilles bouclées*.

5.4 Expérimentation 3 : algorithmes sur les grilles bouclées

Nous considérons des graphes dans lequels le voisinage d'un sommet peut être couvert par une ou deux cliques. L'exécution de l'algorithme de couverture sur chacun de ces graphes fournit une couverture vide. Cette famille de graphes est désignée graphes *grilles bouclées*. Après l'exécution de l'algorithme de couverture, tous les sommets de la *grille bouclée* sont dans l'ensemble $\mathcal{C}$ des sommets à corriger.
Dans cette section, nous évaluons les performances de nos algorithmes, particulièrement l'algorithme de correction en effectuant des opérations de *suppression* et d'*ajout* d'arêtes uniquement. Nous déterminons une borne supérieure de la distance line de ces graphes. Nous comparons cette borne avec les résultats obtenus par l'algorithme de correction avec l'approche *aléatoire sans remise* (voir tableau 5.1).
Nous débutons notre analyse par la définition et la construction d'une grille bouclée. Ensuite, nous décrivons le protocole d'expérimentation sur des grilles bouclées d'ordres différents. Enfin nous interprétons les résultats obtenus pour chaque opération.

5.4.1 Définition des grilles bouclées et les distances line théoriques

5.4.1.1 Définition de la grille bouclée $G_{k,k'}$

Chaque sommet de $G_{k,k'}$ est identifié par le couple (i, j) avec $0 \leq i < k$ et $0 \leq j < k'$. Le sommet (i, j) est adjacent au sommet :

- $(i, j + 1)$ si $j < k' - 2$
- $(i + 1, j)$ si $i < k - 2$
- $(i, j - 1)$ si $j > 0$
- $(i - 1, j)$ si $i > 0$

De plus, les sommets $(0, 0)$ et $(0, k - 1)$, $(0, 0)$ et $(0, k' - 1)$, $(k - 1, 0)$ et $(k - 1, k' - 1)$, $(0, k' - 1)$ et $(k - 1, k' - 1)$ sont adjacents. On remarque que tout graphe induit par un sommet et son voisinage forme un graphe étoile $K_{1,4}$. La figure 5.17 est un exemple de *grille bouclée* $G_{4,4}$. Ce graphe contient 16 sommets, 28 arêtes. Les sommets $(0, 0), (0, 1), (1, 1), (1, 0)$ forme une cellule et le graphe $G_{4,4}$ contient 10 cellules.

Définition 11. *Une cellule est un cycle de longueur* 4 *identifié par les sommets* (i, j), $(i, j+1)$, $(i+1, j)$ *et* $(i + 1, j + 1)$ *avec* $i < k - 1$ *et* $j < k' - 1$. *Nous notons une telle cellule* $C_{i,j}$.

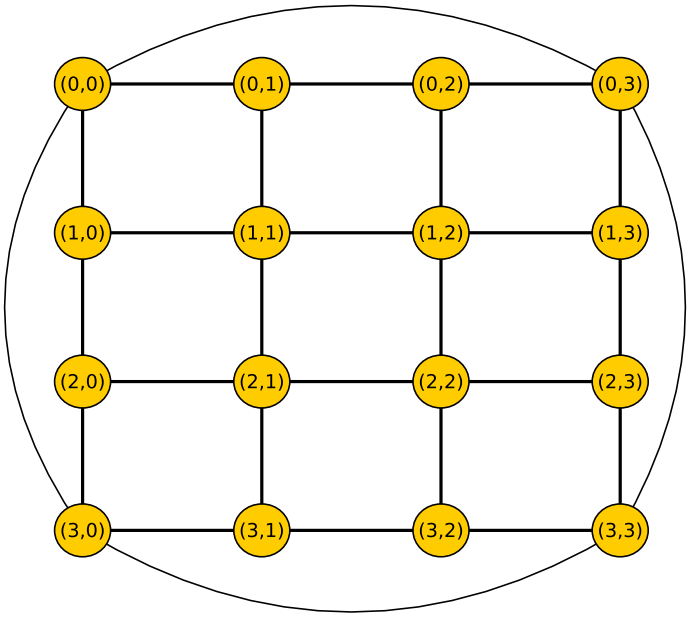

Figure 5.17: La grille bouclée $G_{4,4}$: elle est composée de 16 sommets, 28 arêtes et 10 cellules.

Si $k = k' = 2$, la grille bouclée $G_{2,2}$ est la cellule $C_{0,0}$.

Propriété 2. *Le graphe $G_{k,k'}$ possède $k \times k'$ sommets, $k \times (k'-1) + k' \times (k-1) + 4$ arêtes et $(k-1) \times (k'-1) + 1$ cellules.*

5.4.1.2 Correction des grilles bouclées

Nous allons considérer les modifications de l'ensemble des arêtes de $G_{k,k'}$ afin d'obtenir un line-graphe. La première modification se base uniquement par ajout d'arêtes et la seconde sur la suppression d'arêtes uniquement. Nous supposons que les deux opérations conduisent sur la même borne supérieure de $DL(G_{k,k'})$.

5.4.1.2.1 Modification par *ajout d'arêtes uniquement*

Soit le graphe $G_{k,k'}$ contenant $(k-1) \times (k'-1) + 1$ cellules. Pour transformer chaque cellule en cliques comme cela est illustré dans la figure 5.18, nous ajoutons 2 arêtes. Nous considérons le sommet $(0, 0)$ contenu dans les cellules $C_{0,0}$ et $C_{k-1,k'-1}$. Nous ajoutons 2 arêtes dans $C_{0,0}$ et $C_{k-1,k'-1}$. Ces cellules deviennent des cliques K_4.
L'arête $\{(i, j+1), (i+1, j)\}$ appartient aux cellules $C_{i,j}$ et $C_{i,j+1}$. Or cette arête est déjà couverte par une clique K_4 de la cellule $C_{i,j}$. Alors nous ne pouvons pas ajouter d'arêtes dans la cellule $C_{i,j+1}$. L'arête $\{(i, j+1), (i, j+2)\}$ forme une clique K_2. Le sommet $(i, j+1)$ est couvert par une clique K_4 et

une clique K_2. Le sommet $(i+1, j)$ est aussi couvert par une clique K_4 et une clique K_2 parce que les cellules $C_{i,j}$ et $C_{i+1,j}$ partagent l'arête $\{(i+1, j), (i+1, j+1)\}$ et cette arête forme une clique K_4 avec la cellule $C_{i,j}$. Les cellules $C_{i,j}$ et $C_{i+1,j+1}$ ne partagent que le sommet $(i+1, j+1)$. En plus les arêtes $\{(i+1, j+1), (i+2, j+1)\}$ de $C_{i+1,j}$ et $\{(i+1, j+1), (i+1, j+2)\}$ de $C_{i,j+1}$ ne sont pas couvertes par une clique K_4. Nous pouvons alors transformer $C_{i+1,j+1}$ en une clique K_4 en ajoutant 2 arêtes.
Ainsi, dans des cellules successives en lignes (avec k) ou en colonnes (avec k'), nous ajoutons des arêtes dans $\lceil \frac{k \times k'}{2} \rceil + 1$ cellules. L'arête d'une cellule qui n'est pas contenue par une clique K_4 forme une clique K_2. Les cellules ayant un seul sommet en commun sont transformées en des cliques K_4.
À la fin de la correction, la grille bouclée $G_{k,k'}$ est partitionnée en des cliques finies K_4 et K_2. Dans cette construction, nous remarquons que chaque sommet est couvert par 2 cliques. De cette construction découle le lemme suivant :

Lemme 7. *La distance line d'un graphe cellule $G_{k,k'}$ avec l'opération* ajout uniquement *est*

$$DL(G_{k,k'}) \leq k \times k' + 3 \tag{5.4}$$

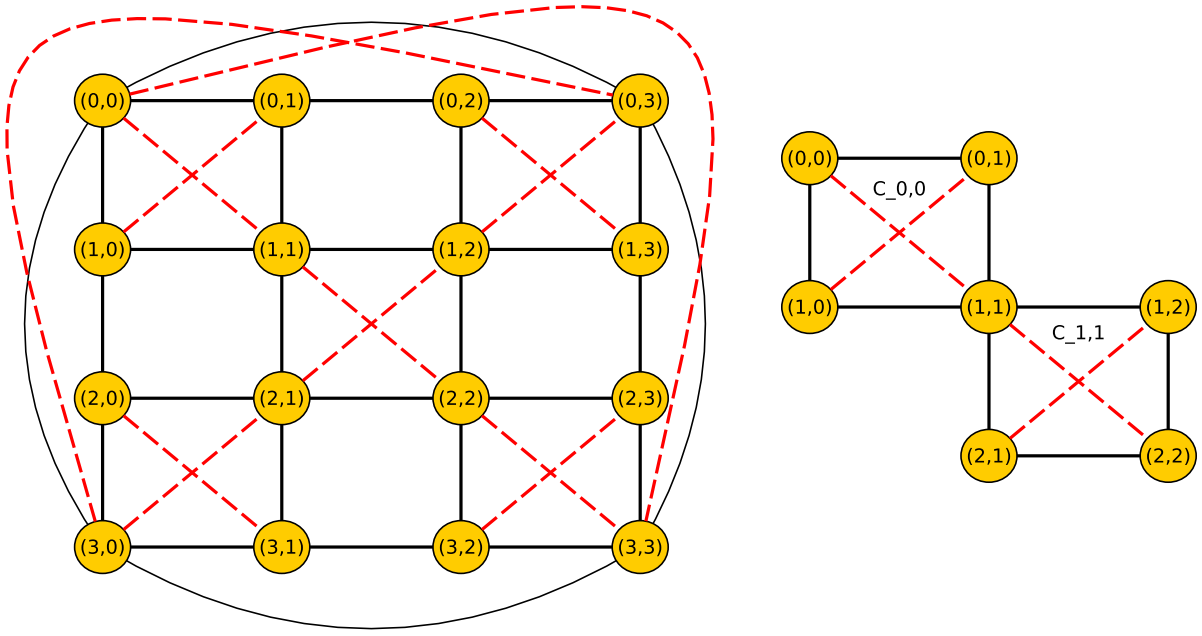

Figure 5.18: La grille bouclée corrigé $G_{4,4}$: elle est composée de 16 sommets, 33 arêtes. Il contient 4 cliques K_2 et 6 cliques K_4. Les arêtes ajoutées sont les traits de couleur rouge.

Dans la figure 5.18, nous réalisons la correction de distance line $DL(G_{4,4}) \leq 12$ en transformant les cellules partageant un sommet en cliques K_4. C'est le cas des cellules $C_{1,2}$ et $C_{0,2}$ qui ont le sommet

$(1,2)$ en commun. Les arêtes partagées entre deux cellules ont un des sommets couvert par une clique K_2 et l'autre sommet couvert par une clique K_4. Tel est le cas avec le sommet $(3,2)$ qui forme l'arête $\{(2,2),(3,2)\}$ et cette arête est contenue par une clique K_4.

5.4.1.2.2 Modification par *suppression d'arêtes uniquement*

Soit le graphe $G_{k,k'}$ contenant $(k-1)\times(k'-1)+1$ cellules. Nous supprimons les arêtes $[\{(0,0),(k-1,0)\},\{(0,0),(0,k'-1)\},\{(k-1,0),(k-1,k'+1)\}]$ de la cellule $C_{k-1,k'-1}$. Cette cellule contient uniquement l'arête $\{(0,k'-1),(k-1,k'-1)\}$ et cette arête forme la clique K_2. Nous supprimons également les arêtes $\{(0,k'-2),(0,k'-1)\}$ et $\{(k-2,k'-1),(k-1,k'-1)\}$ incidentes respectivement aux sommets $(0,k'-1)$ et $(k-1,k'-1)$ de sorte que ces sommets soient couverts par deux cliques K_2. Ils sont couverts par deux cliques K_2 c'est-à-dire $\{(0,0),(1,0)\}$ et $\{(k-2,0),(k-1,0)\}$.
Considérons les sommets de degré 4. Soit (i,j) un tel sommet. Pour former une bipartition autour de ce sommet, nous allons supprimer 2 arêtes. Chaque arête appartient à deux cellules voisines. Dans notre cas, nous supprimons l'arête $\{(i-1,j),(i,j)\}$ entre les cellules $C_{i-1,j-1}$ et $C_{i-1,j}$ et aussi l'arête $\{(i,j),(i+1,j)\}$ entre les cellules $C_{i,j-1}$ et $C_{i,j}$. Ces sommets sont couverts aussi par des cliques K_2.
Les arêtes incidentes à un sommet de degré 3 et n'étant pas des cliques K_2 sont aussi supprimées.
À la fin de l'algorithme de correction, la couverture de corrélation ne contient que des cliques K_2. Le graphe $G_{k,k'}$ est un graphe hamiltonien. La distance de correction entre $G_{k,k'}$ et $L(G_{k,k'})$ est $DC_{k,k'}=(k-1)\times(k'-1)+3$ et cette distance est minimale.

Lemme 8. *La distance line d'une grille bouclée $G_{k,k'}$ avec l'opération* suppression uniquement *d'arêtes est*

$$DL(G_{k,k'})=(k-1)\times(k'-1)+3 \tag{5.5}$$

Une illustration de la correction avec l'opération *suppression uniquement* est faite avec la grille $G_{4,4}$ dans la figure 5.19. La grille $G_{4,4}$ possède $k\times k'=16$ cliques K_2 et 12 arêtes sont supprimées.

Conclusion : nous avons déterminé deux types de modifications d'arêtes qui ont une borne supérieure de la distance line pour chaque opération. En revanche, les cliques formant la couverture de corrélation sont différentes selon la modification. Dans la modification *ajout d'arêtes uniquement*, le grille bouclé $G_{k,k'}$ contient des cliques K_4 et K_2 alors que $G_{k,k'}$ ne contient que des cliques K_2 dans la *suppression d'arêtes uniquement*.

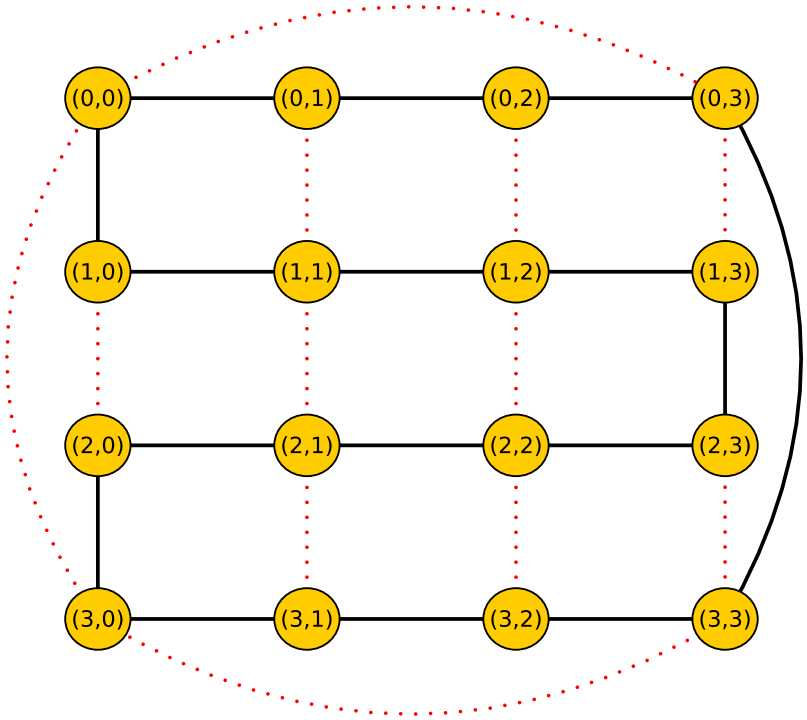

Figure 5.19: La grille bouclée $G_{4,4}$: elle est composée de 16 sommets, 16 arêtes et 16 cliques K_2. Les arêtes supprimées sont les traits en pointillées rouges

5.4.2 Protocole d'expérimentation sur les grilles bouclées

Nous allons comparer cette borne supérieure de l'équation 5.5 avec les distances de correction obtenues par l'algorithme de correction.
Considérons $G_{k,k'}$ une grille bouclée dans laquelle le nombre de sommets par lignes est identique le nombre de sommets par colonnes ($k = k'$). Nous le notons G_k.
Nous construisons 48 grilles bouclées contenant chacune $(k-1)\times(k-1)+1$ cellules, avec $k \in \{2, \cdots, 98\}$ un nombre pair. Dans chaque graphe G_k, nous exécutons 50 fois notre couple d'algorithmes avec chaque modification d'arêtes et la distance de correction obtenue est comparée avec la borne supérieure (équation 5.5).

Soient $\phi^+(u, v)$ le coût de l'opération *ajouter une arête* et $\phi^-(u, v)$ le poids de l'opération *supprimer une arête* (voir section 4.3.3).
La modification *ajout d'arêtes uniquement* est telle que

- L'ajout d'arêtes a un coût $\phi^+(u, v) = 1$,

- La suppression d'arêtes a un coût $\phi^-(u,v) = 10$.

Quant à la modification *suppression d'arêtes uniquement*, elle se définit comme suit :

- L'ajout d'arêtes a un coût $\phi^+(u,v) = 10$.
- La suppression d'arêtes a un coût $\phi^-(u,v) = 1$.

Nous allons comparer l'évolution des distances de correction des 48 graphes en fonction la borne supérieure pour chaque modification réalisée.

5.4.3 Analyse des résultats

Notre objectif est de présenter les variations des distances de correction par rapport à la borne supérieure de la distance line pour chaque modification réalisée sur les graphes pendant l'algorithme de correction. Pour ce faire, nous regroupons notre analyse en 5 expérimentations.

Les deux premières expérimentations comparent les distances de correction avec la borne supérieure pour la modification *ajout d'arêtes uniquement* (figure 5.20 (a) et pour la modification *suppression d'arêtes uniquement* figure 5.20 (c). Nous constatons que les courbes des distances de correction et celle de la borne supérieure sont croissantes. La courbe de la borne supérieure, désignée par *borneSup* dans les graphiques (a) et (c), évolue lentement par rapport aux courbes des distances et l'écart entre ces courbes croit linéairement. Pour comprendre cet écart croissant, nous vérifions le pourcentage d'arêtes supprimées pour chaque modification. Ce sont les deux autres expérimentations faites et représentées par la figure 5.20 (b) pour la modification *ajout d'arêtes uniquement* et la figure 5.20 (d) pour la modification *suppression d'arêtes uniquement*. En effet, nous avons choisi les arêtes supprimées parce que le nombre d'arêtes est déjà connu c'est-à-dire 0 pour l'ajout uniquement et la borne supérieure pour la suppression uniquement.
Nous remarquons que les arêtes de G_k supprimées avoisinent en moyenne de 40% quand le nombre k de cellules est faible ($k \leq 14$) dans la modification *ajout arêtes uniquement*. Au délà de $k > 14$, une arête sur deux du graphe G_k est supprimée. La courbe *aretes_supprimees_ajout_1* dans le graphique (b) représente le pourcentage d'arêtes supprimées. En effet, nous expliquons ces chiffres par l'ajout d'arêtes entre des sommets de cellules voisines et ces sommets ne sont pas partagés entre les cellules voisines. Ces arêtes ajoutées impliquent la suppression des arêtes de G_k parce que la partition π_s (voir section 4.3.3) des arêtes à supprimer pendant la compression n'est pas vide et contient des arêtes de G_k en général. Ainsi ces arêtes provoquent l'abandon des arêtes diagonales ajoutées à partir du sommet commun entre des cellules comme indiqué dans l'exemple du graphe $G_{4,4}$ dans la figure 5.18.

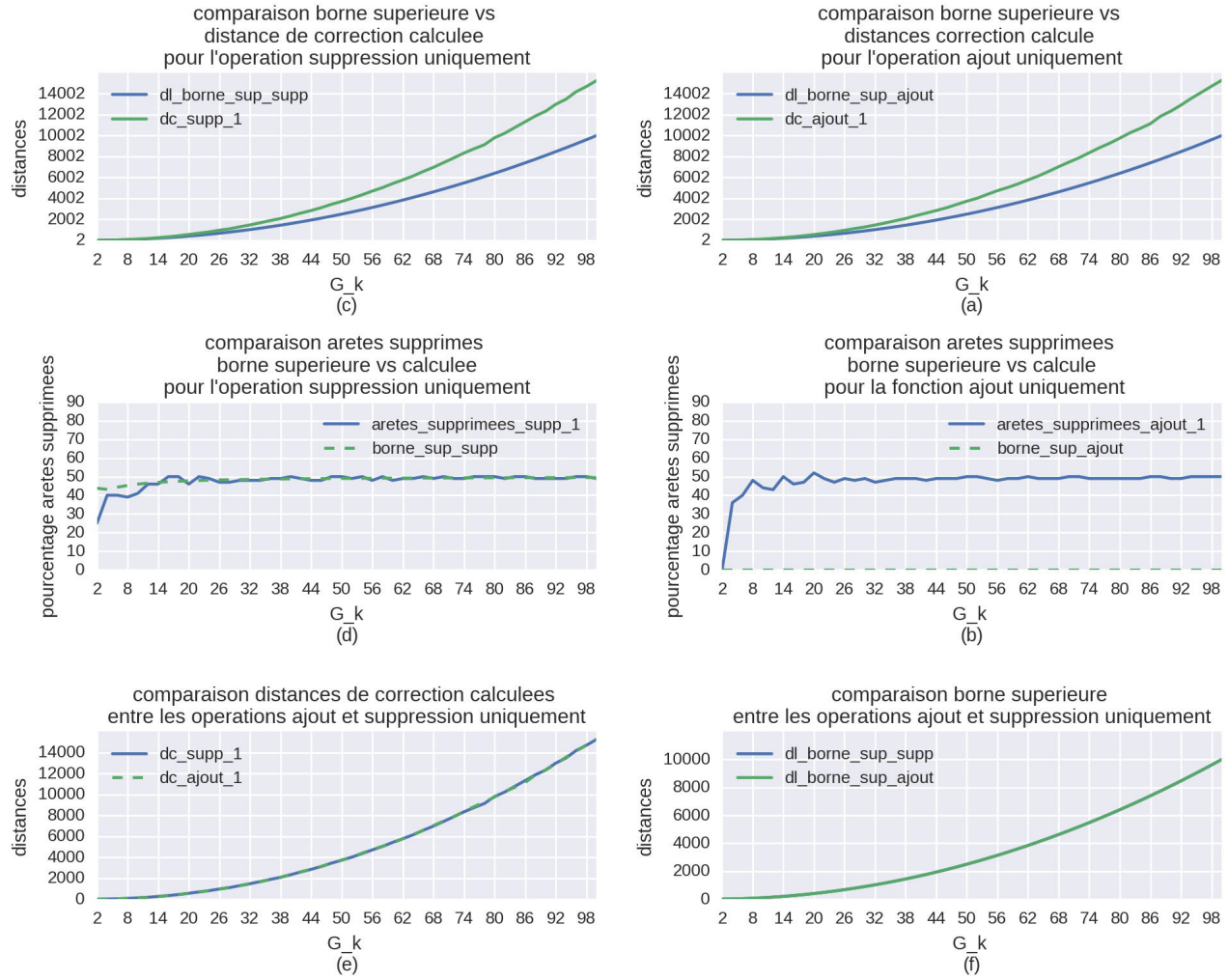

Figure 5.20: Comparaison entre la borne supérieure de la distance line et les distances de correction calculées selon des fonctions de coût *suppression* et *ajout* : la figure (a) désigne la comparaison entre les distances de correction et la borne supérieure de l'équation 5.5 avec la modification *ajout d'arêtes uniquement*, la figure (c) désigne la comparaison entre les distances de correction et la borne supérieure de l'équation 5.5 avec la modification *suppression d'arêtes uniquement*, la figure (b) compare le pourcentage d'arêtes supprimées dans les graphes bouclées avec celui de la borne supérieure de l'équation 5.5 dans la modification *ajout d'arêtes uniquement*, la figure (d) compare le pourcentage d'arêtes supprimées dans les graphes bouclées avec celui de la borne supérieure de l'équation 5.5 dans la modification *suppression d'arêtes uniquement*, la figure (e) compare les distances de correction entre les différentes modifications.

En ce qui concerne la modification *suppression d'arêtes uniquement*, les arêtes supprimées proviennent aussi des cellules voisines. Dans les cellules partageant un sommet, l'algorithme ajoute généralement une arête diagonale à partir de ce sommet commun dans une seule cellule. Les arêtes diagonales ajoutées

sont responsables de l'augmentation de la distance de correction (courbe *dc_supp_1* dans le graphique (*c*)). Pour des cellules partageant une arête, l'algorithme supprime certaines arêtes communes comme indiqué dans la section 5.4.1.2.2. Les autres arêtes communes non supprimées sont dues à la présence des arêtes diagonales. Cela explique pourquoi nous avons en moyenne 50% des arêtes de G_k qui sont supprimées dans le graphique (*d*). Ce pourcentage est identique à celui des arêtes supprimées de la borne supérieure lorsque le nombre de cellules devient élevé $k \geq 18$ et il ne signifie pas que l'algorithme de correction supprime les arêtes de la section 5.4.1.2.2.
Par ailleurs, les courbes *dc_ajout_1* de la modification d'*ajout d'arêtes uniquement* et celle *dc_supp_1* de la modification *suppression d'arêtes uniquement* ont les mêmes tendances et sont superposées (figure (*e*)) parce que dans la modification d'*ajout*, l'algorithme ajoute la même proportion d'arêtes qu'il supprime dans la modification *suppression*.

Conclusion : l'expérimentation montre que le line-graphe $L(G_k)$ proposé par l'algorithme de correction pour un k donnée est identique en terme de distance de correction quelle que soit la modification réalisée comme illustre la figure 5.20(e). Toutefois, les graphes corrigés ne sont pas optimaux en terme de distances de correction parce que l'algorithme priorise dans certains cas les modifications *ajout d'arêtes uniquement* et *suppression uniquement* et cela leur permet d'atteindre des minimums locaux. Cependant le minimum global n'est pas atteint généralement.

5.4.4 Conclusion de l'expérimentation 3

Les grilles bouclées ont la particularité d'avoir des arêtes qui ne peuvent être partitionnées en cliques. Nous avons décrit la construction de ces graphes et nous définissons deux méthodes pour les corriger. La première méthode est la modification d'*ajout uniquement* qui consiste à ajouter uniquement des arêtes et la seconde méthode est la modification *suppression uniquement* qui supprime uniquement des arêtes des grilles bouclées. Avec ces méthodes, nous avons trouvé une borne supérieure unique de la distance line des grilles bouclées et nous avons comparé cette borne supérieure avec les distances de correction obtenues après l'algorithme de correction.
Nous remarquons que les distances de correction varient très peu entre les modifications *ajout uniquement* et *suppression uniquement*. Les graphes corrigés n'ont pas de distances de corrections optimales parce que l'algorithme supprime des arêtes pendant la modification *ajout d'arêtes uniquement* et ajoute des arêtes pendant la modification *suppression uniquement*.

5.5 Conclusion du chapitre 5

Le chapitre 5 analyse les performances de nos algorithmes de couverture et de correction selon 3 expérimentations.

La première expérimentation consiste à modifier les k cases de la matrice d'adjacence du line-graphe d'un réseau électrique. Ces cases modifiées sont divisées en deux sous-ensembles disjoints (cases *fausses négatives* et cases *fausses positives*) selon une variable $p \in [0, 1]$. Si $p = 0$ alors l'ensemble des cases modifiées est composé que de cases *fausses positives* et si $p = 1$ alors nous avons que des cases *fausses négatives*. Le but est de borner le nombre de cases corrigées par nos algorithmes. Ainsi, nous avons défini les distances de correction et de Hamming. La distance de correction est le nombre minimum de cases à modifier dans un graphe de k cases erronées pour en faire un line-graphe. Quant à la distance de Hamming, elle est la différence de cases entre les matrices de line-graphe proposé par nos algorithmes et le line-graphe du réseau électrique. Nous avons comparé le nombre de cases corrigées avec 5 approches de correction qui sont : *aléatoire sans remise* ($2c$), *degré minimun sans remise* ($2a$), *coût minimum sans remise* ($2b$), *degré minimun avec remise* ($1a$), *coût minimum avec remise* ($1b$). À chaque approche, nous avons 3 coûts de modification d'une case : *unitaire*, *ajout* et *suppression*. Nous avons conclut que l'approche *aléatoire sans remise* ($2c$) proposait des distances de correction minimales quelle que soit la repartition effectuée p et la fonction de coût utilisée. Ces distances constituent la borne supérieure de la distance line quand le nombre k de cases modifiées est faible $k \leq 5$. D'autre part, nous avons montré que les distances de correction et de Hamming deviennent très corrélées quand le nombre de cases modifiées initiales est élevé $k > 10$. Dans ce cas où $k \leq 5$, le line-graphe proposé par l'algorithme de correction est celui de réseau électrique. La distance de correction peut être utilisée comme une métrique lorsque la topologie initiale du réseau est inconnue.

La seconde expérimentation considère que chaque case de la matrice du line-graphe est associée à une valeur de corrélation. Les valeurs de corrélation sont générées en tenant compte des distributions des valeurs de corrélation du réseau électrique d'un data center *Champlan*. Ces corrélations sont calculées avec la *distance de Pearson*. Les valeurs de corrélation dans la matrice forment la *matrice de corrélation*. Nous avons défini un ensemble de seuil dans lequel chaque seuil est appliqué à la matrice de corrélation pour en construire la matrice d'adjacence du graphe G_s. Le graphe G_s contient des cases *fausses négatives* et des cases *fausses positives*. Notre objectif est de minimiser le nombre de cases erronées dans le graphe G_s après l'exécution de nos algorithmes et cela nécessite la sélection d'une valeur adéquate du seuil. Nous avons considéré l'approche de correction *aléatoire sans remise* ($2c$) et nous avons sélectionné quatre fonctions de coût : *unitaire*, *ajout*, *suppression* et *normale*. Les fonctions

de coût sont fonction des cases modifiées par l'algorithme de correction. Nous avons déduit que le bon seuil appartient à l'intervalle $s \in]0.6, 0.7]$ et la fonction *normale* donne de bons résultats pour le calcul dans les distances de correction et de Hamming.

La dernière expérimentation se focalise sur les graphes dans lesquelles un sommet et son voisinage ne peuvent être couverts par une ou deux cliques. Un exemple de ces graphes est la famille des graphes *grilles bouclées*. Une grille bouclée de k lignes et k' colonnes est composée de $(k-1) \times (k'-1) + 1$ cellules avec une cellule un graphe biparti $K_{2,2}$ non orienté. Nous avons montré que la correction de ces graphes peut se faire selon deux méthodes (modification *ajout d'arêtes uniquement* et *suppression d'arêtes uniquement*). Les deux modifications admettent la même borne supérieure de ses distances line. Notre objectif est de vérifier que la convergence des distances de correction vers la borne supérieure quelque soit la modification réalisée. Les résultats obtenus montrent que les distances de correction sont invariantes peu importe les modifications et que les graphes corrigés n'ont pas de distances de corrections optimales.

Au terme de ces 3 expérimentations, nous pouvons conclure que nos algorithmes n'ont pas un comportement optimal lorsque le mode de correction est *aléatoire sans remise* et le seuil de corrélation est contenu dans l'intervalle $]0.6, 0.7]$ peu importe la repartition des cases erronées et la fonction de coût. Ces conditions garantissent que la distance de correction est minimale pour des graphes ayant peu d'erreurs et le line-graphe proposé par l'algorithme de correction diffère de peu d'arêtes du line-graphe cible. Cependant pour la famille des graphes dont aucun sommet ne peut être couvert par une clique (cas des grilles *bouclées*), les distances line calculées ne convergent pas vers la borne supérieure définie. Toutefois, le type de correction (modifications *ajout uniquement* et *suppression uniquement*) sur ces graphes n'influencent pas les valeurs de distances de correction.

Chapitre 6

Conclusion générale

6.1 Conclusion

Les algorithmes présentés dans ce document s'inscrivent dans le cas général de l'étude de découverte de topologie. La problématique à laquelle nous avons tenté de repondre est : étant donnés un ensemble de mesures physiques souvent erronées et de lois physiques sur les liens d'un réseau de flots, est il possible de découvrir la topologie d'un réseau énergétique dans lequel les mesures sont extraites? Nous avons restreint le réseau énergétique à un réseau électrique parce que l'entreprise *DCbrain SAS*, à l'origine de la thèse, avait de soucis de schémas électriques pendant l'étude énergétique d'un data center. Elle a constaté que le schéma électrique n'était pas mis à jour entre deux maintenances successives. Pour repondre alors à ce problème, nous avons procédé en 4 étapes.

La première étape a été de considérer les mesures comme des séries temporelles ensuite de sélectionner les grandeurs physiques présentes dans le réseau et de définir des lois de conservation adaptées aux grandeurs de ce réseau. Ensuite nous avons modélisé le réseau par un graphe de flots dans lequel les mesures sont les flots, les sommets sont les équipements, les arcs sont les câbles électriques et le graphe est un DAG sans circuit.

Dans la seconde étape, nous avons remarqué, sur certains équipements adjacents dont nous connaissons leurs mesures, que les courbes des mesures ont les mêmes comportements aux mêmes instants de temps. Nous avons décidé de comparer les comportements des séries temporelles en supposant que les arcs, d'où proviennent les mesures, ont une extrémité commune si leurs séries temporelles ont des comportements similaires. Nous avons listé les différentes méthodes de comparaison et nous avons choisi la distance de Pearson parce qu'elle est une métrique, elle est de complexité linéaire et elle ne

traite pas le décalage temporel. Les valeurs de distances aussi appelées coefficients de similarité appartiennent à $[0, 1]$. Un coefficient proche de 1 signifie qu'il existe une extrémité commune entre une paire d'arcs. En testant la distance de Pearson sur un sous-réseau réel du data center *Champlan*, nous avons constaté des erreurs dans les coefficients de similarité c'est-à-dire un coefficient proche de 0 alors qu'il n'existe aucune extrémité commune entre une paire d'arcs. Ces erreurs sont dues aux mécanismes de fonctionnement, aux données et aussi à la distance de Pearson. Les coefficients de similarité forment la matrice de corrélation et le graphe associé à cette matrice est le *graphe de corrélation*. Si cette matrice ne contient aucune erreur alors le graphe de corrélation est le line-graphe du graphe non orienté sous-jacent au DAG du réseau électrique.

Le troisième étape a montré que le line-graphe a une partition unique de son ensemble d'arêtes en cliques appelé la *couverture de corrélation* sauf en cas d'ambiguïtés où nous utilisons les mesures électriques et la fonction $Verif - correl$ (voir section 2.2.2.5) pour déterminer la bonne partition au voisinage d'un sommet ambigu. Nous avons proposé l'algorithme de couverture qui retourne la couverture de corrélation du graphe de corrélation si celui-ci ne contient aucune erreur. Dans le cas contraire, les sommets n'étant pas couverts dans la couverture de corrélation sont corrigés avec l'algorithme de correction. La correction consiste alors à supprimer et à ajouter des arêtes incidentes à un sommet à corriger de telle sorte que le partitionnement de son voisinage forme deux cliques et le coût de modification des arêtes incidentes de ce sommet soit de coût minimum. Nous avons proposé, à partir de la couverture de corrélation, une méthode pour construire le graphe non orienté sous-jacent au DAG et aussi une heuristique pour orienter les arêtes de ce graphe. Cependant cette heuristique ne donne pas une solution optimale.

Dans la quatrième étape, nous avons évalué nos algorithmes sur des graphes générés. La première expérimentation consiste à modifier des cases de la matrice d'adjacence du line-graphe des graphes générés puis à vérifier le nombre d'arêtes corrigées par nos algorithmes. Nous avons constaté que l'approche *aléatoire sans remise* donne les meilleurs résultats en termes d'arêtes corrigées mais la fonction de coût n'a aucune influence sur ce résultat. Par ailleurs, toutes les cases modifiées sont corrigées par l'algorithme lorsque le nombre de cases à corriger est faible c'est-à-dire inférieure à 6. La seconde expérimentation consiste à attribuer des valeurs de corrélation entre les paires d'arcs à partir de la distribution des valeurs de corrélation du réseau de *Champlan*. Puis à determiner la valeur de seuil qui donne le nombre minimum d'arêtes corrigées par nos algorithmes. Nous avons déduit que cette valeur de seuil appartient à l'intervalle $]0.6, 0.7]$ parce que il y a peu d'erreurs dans cet intervalle avant la correction et l'algorithme de correction fournit de meilleurs résultats dans ces situations. La dernière expérimentation se déroule sur des graphes *grilles bouclées* dans lesquelles chaque sommet est couvert

par plus de deux cliques. Nous avons montré que les corrections de sommets en ajoutant et en supprimant uniquement des arêtes ont la même borne supérieure de la distance line. Cependant les distances de correction sont invariants quelque soit la méthode (ajouter ou supprimer uniquement) et ne tendent pas vers la borne supérieure définie.

Nous parvenons à déterminer un graphe isomorphe au graphe non orienté sous-jacent au DAG du réseau électrique lorsqu'il y a peu d'erreurs de corrélation (c'est-à-dire nombre inférieur à 6) entre les paires d'arcs. En revanche, il est difficile de trouver le graphe le plus proche isomorphe au graphe non orienté dont le nombre d'erreurs est supérieur à 6. Une solution est d'utiliser une expertise humaine qui identifie certaines sommets de DAG et les arêtes incidentes à ces sommets afin de réduire le nombre de sommets à corriger.

6.2 Perspectives

Certains problèmes rencontrés au cours de l'implémentation de ces algorithmes et de l'analyse des séries temporelles ouvrent des perspectives intéressantes pour les travaux futures.

Nous commencons par la détermination de cliques maximales dans un graphe. En effet, nous avons montré que la présence de cliques est indispensable dans un line-graphe car un line-graphe admet une couverture en cliques dans laquelle chaque clique est un sommet du graphe racine du line-graphe. Les travaux de [63, 64] affirment que la détermination de cliques maximales depend du nombre de sommets du graphe et le meilleur algorithme s'exécute dans le pire des cas en $\mathcal{O}^{3^{n/3}}$ avec n le nombre de sommets du graphe. Ainsi, pour des graphes racines de degrés maximals très élevés, la couverture en cliques est irréaliste à trouver. Par ailleurs, pour les graphes de degrés maximals très élevés dans lesquels nous devons appliquer l'algorithme de correction, la compression autour chaque sommet à corriger est difficile à obtenir à cause du nombre de partitions π_1, π_2, π_s à comparer pour avoir celles de coût minimum. Nous pensons que la définition d'une structure de données adaptée à ces types de graphes permettrait de résoudre les soucis évoqués plus haut.

Ensuite, le calcul des coefficients de similarité introduit aussi des erreurs dans la matrice de corrélation. Nous pensons qu'une méthode combinant des classifieurs dans le temps, des auto corrélations, des transformations de séries telles que les shapelets et la densité spectrale de puissance donnerait de coefficients proches de la réalité. Nous pensons à la méthode *Collection of Transformation Ensemble (COTE)* [65] qui fournit de meilleurs résultats [66] sur les données de la base de données UCR [33].

Nous démontrons la NP-complétude et l'approximabilité de notre problème. Aussi, une étude approfondie sur la manière de fixer les fonctions de coût pourrait améliorer les performances de l'algorithme de correction. Nous donnerons la latitude à l'algorithme de correction de choisir la fonction de coût adéquate selon les caractéristiques du graphe et le résultat de l'algorithme de couverture.

Enfin, nous pouvons étendre l'application de nos algorithmes à la découverte de topologie d'autres réseaux énergétiques tels que les réseaux de gaz, d'eau et de chaleur. La particularité de ces réseaux est le fluide transporté. Ce fluide implique la définition de nouvelles règles locales et de méthodes de similarité qui tiennent compte du délai de propagation du fluide.

Appendix A

Annexes

Figure A.1: Approche de correction aléatoire sans remise à coût unitaire pour $k = \{1, 2, 3, 4, 5\}$ cases modifiées : La première colonne représente la distribution des distances de correction $moy\_DC_{k,0.5}$. La seconde colonne est la distribution des distances de Hamming $moy\_DH_{k,0.5}$. La troisième colonne est la fonction de repartition de la corrélation entre les distances de correction et de Hamming avec en abscisse la corrélation entre ces distances (correlation_DC_DH). La quatrième colonne est la fonction cumulative des distances de Hamming. La première ligne est associée à $k = 1$ case modifiée, la seconde ligne à $k = 2$ cases modifiées, la troisième ligne à 5 cases modifiées et enfin la dernière à 9 cases modifiées.

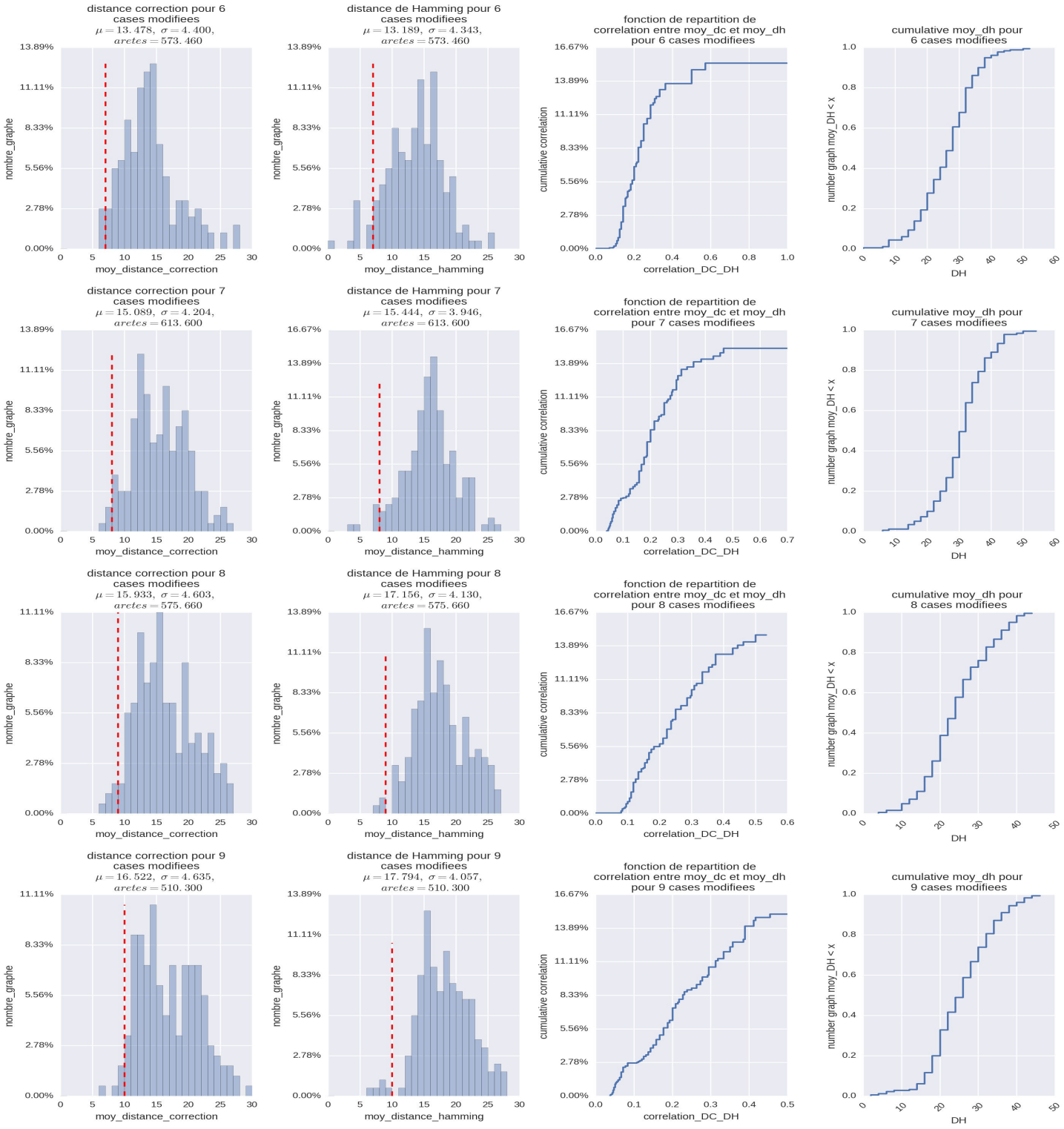

Figure A.2: Approche de correction aléatoire sans remise à coût unitaire pour $k = \{6, 7, 8, 9\}$ cases modifiées : La première colonne représente la distribution des distances de correction $moy\_DC_{k,0.5}$. La seconde colonne est la distribution des distances de Hamming $moy\_DH_{k,0.5}$. La troisième colonne est la fonction de repartition de la corrélation entre les distances de correction et de Hamming avec en abscisse la corrélation entre ces distances (correlation_DL_DH). La quatrième colonne est la fonction cumulative des distances de Hamming. La première ligne est associée à $k = 1$ case modifiée, la seconde ligne à $k = 2$ cases modifiées, la troisième ligne à 5 cases modifiées et enfin la dernière à 9 cases modifiées.

Bibliographie

[1] François Petitjean. Description des alignements formés par DTW. 2011.

[2] James B. Orlin Ravindra K. Ahuja, Thomas L. Magnanti. Networks flows. *A Simon & Schuster Company, Englewood Cliffs, NJ*, pages 166–196, 1993.

[3] Kenneth R Sheers. Hp openview event correlation services. *Hewlett Packard Journal*, 47:31–33, 1996.

[4] J. Postel. *Internet Control Message Protocol*. IETF, sep 1981. RFC 792.

[5] J. Klensin. *Simple Mail Transfer Protocol*. IETF, apr 2001. RFC 2821.

[6] Yuri Breitbart, Minos Garofalakis, Ben Jai, Cliff Martin, Rajeev Rastogi, and Avi Silberschatz. Topology discovery in heterogeneous ip networks: The netinventory system. *IEEE/ACM Trans. Netw.*, 12(3):401–414, jun 2004.

[7] Yuri Breitbart, Minos Garofalakis, Ben Jai, Cliff Martin, Rajeev Rastogi, and Avi Silberschatz. Topology discovery in heterogeneous ip networks: the netinventory system. *IEEE/ACM Transactions on networking*, 12(3):401–414, 2004.

[8] Kuangyu Qin and Chunquan li. Network topologic discovery based on snmp. pages 1–3, 12 2010.

[9] Bilal Saeed, Tarek Sheltami, and Elhadi Shakshuki. A network topology discovery tool for android smart phones. *Procedia Computer Science*, 63:104 – 111, 2015. The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015)/ The 5th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2015)/ Affiliated Workshops.

[10] Rafat Ahmed Alhanani and Jaafar Abouchabaka. An overview of different techniques and algorithms for network topology discovery. pages 530–535, 11 2014.

[11] Denisa Ghita, Katerina Argyraki, and Patrick Thiran. Network tomography on correlated links. *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2010.

[12] Andrew Adams, Tian Bu, Ramn Caceres, Nick Duffield, Joseph Horowitz, Francesco Lo Presti, Sue B. Moon, Vern Paxson, and Don Towsley. The use of end-to-end multicast measurements for characterizing internal network behavior. 38, 10 1999.

[13] Vijay Arya, Nick G Duffield, and Darryl Veitch. Temporal delay tomography. pages 276–280, 2008.

[14] Tian Bu, Nick Duffield, Francesco Lo Presti, and Don Towsley. Network tomography on general topologies. 30(1):21–30, 2002.

[15] Ramón Cáceres, Nick G Duffield, Joseph Horowitz, and Donald F Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information theory*, 45(7):2462–2480, 1999.

[16] Aiyou Chen, Jin Cao, and Tian Bu. Network tomography: Identifiability and fourier domain estimation. *IEEE Transactions on Signal Processing*, 58(12):6029–6039, 2010.

[17] Nick Duffield. Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, 2006.

[18] VN Padmanbhan, L Qiu, and H Wang. Server-based inference of internet performance. 2003.

[19] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Accurate and efficient sla compliance monitoring. 37(4):109–120, 2007.

[20] Yao Zhao, Yan Chen, and David Bindel. Towards unbiased end-to-end network diagnosis. 36(4):219–230, 2006.

[21] Mark Coates, Rui Castro, Robert Nowak, Manik Gadhiok, Ryan King, and Yolanda Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *ACM SIGMETRICS Performance Evaluation Review*, volume 30, pages 11–20. ACM, 2002.

[22] C Andrieu, A Doucet, W Fitzgerald, and JM Pérez. Bayesian computational approaches to model selection. *Nonlinear and Nonstationary Signal Processing*, page 1, 2000.

[23] Azer Bestavros, John W Byers, and Khaled A Harfoush. Inference and labeling of metric-induced network topologies. *IEEE Transactions on Parallel and Distributed Systems*, 16(11):1053–1065, 2005.

[24] Yehuda Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American statistical association*, 91(433):365–377, 1996.

[25] Jin Cao, Drew Davis, Scott Vander Wiel, and Bin Yu. Time-varying network tomography: router link data. *Journal of the American statistical association*, 95(452):1063–1075, 2000.

[26] Harsh Singhal and George Michailidis. Identifiability of flow distributions from link measurements with applications to computer networks. *Inverse Problems*, 23(5):1821, 2007.

[27] Harsh Chaudhary, Younghun Kim, Tarun Kumar, Abhishek Raman, and Rui Zhang. Network graph representation of physically connected network, oct 2016. US Patent 9,473,368.

[28] Pablo M Cincotta, Mariano Mendez, and Josue A Nunez. Astronomical time series analysis. i. a search for periodicity using information entropy. *The Astrophysical Journal*, 449:231, 1995.

[29] Paul T Hurley, Andreas Kind, and Marc Ph Stoecklin. Methods, systems and computer program products for detecting flow-level network traffic anomalies via abstraction levels, jun 2011. US Patent 7,962,611.

[30] Soumya D Mohanty. Robust test for detecting nonstationarity in data from gravitational wave detectors. *Physical Review D*, 61(12):122002, 2000.

[31] Flore Harlé. Détection de ruptures multiples dans des séries temporelles multivariées : application à l'inférence de réseaux de dependance. Master's thesis, Grenoble Alpes, 2016.

[32] Toni M Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[33] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, 2015.

[34] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[35] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 792–803. VLDB Endowment, 2004.

[36] Ronald I Greenberg. Fast and simple computation of all longest common subsequences. *arXiv preprint cs/0211001*, 2002.

[37] Pierre-François Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):306–318, 2009.

[38] Michael R Berthold and Frank Höppner. On clustering time series using euclidean distance and pearson correlation. *arXiv preprint arXiv:1601.02213*, 2016.

[39] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[40] Lexiang Ye and Eamonn Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery*, 22(1-2):149–182, 2011.

[41] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162. ACM, 2011.

[42] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[43] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297. ACM, 2012.

[44] Thanawin Rakthanmanon and Eamonn Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2013.

[45] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012.

[46] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[47] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2796–2802, 2013.

[48] Donghua Pan Chonghui Guo, Hailin Li. An improved piecewise aggregate approximation based on statistical features for time series mining. *4th International Conference, KSEM 2010*, pages 234–244, September 1-3, 2010.

[49] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[50] Eammon Keogh. The ucr time series data mining archive. *http://www. cs. ucr. edu/˜ eamonn/TSDMA/index. html*, 2002.

[51] Frank Harary and Robert Z Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960.

[52] Hassler Whitney. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54(1):150–168, 1932. Concept of root graph was introduced by Whitney and he proved that if two linegraphs are isomorphic and connected, then their root graphs are isomorphic except for the triangle graph.

[53] Frank Harary. Graph theory. addison wesley publishing company. *Reading, MA, USA*, pages 71–83, 1969.

[54] J Krausz. Démonstration nouvelle d'une théorème de Whitney sur les réseaux. *Mat. Fiz. Lapok*, 50(1):75–85, 1943.

[55] ACMM van Rooij and H Wilf. The interchange graph of a finite graph. *Acta Mathematica Hungarica*, 16(3-4):263–269, 1965.

[56] Lowell W Beineke. Derived graphs and digraphs. *Beiträge zur Graphentheorie*, pages 17–33, 1968.

[57] Mihalis Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264. ACM, 1978.

[58] Bjarni V. Halldórsson, Dima Blokh, and Roded Sharan. Estimating population size via line graph reconstruction. *Algorithms for Molecular Biology*, 8(1):17, Jul 2013.

[59] Nicholas D. Roussopoulos. A max m,n algorithm for determining the graph h from its line graph g. *Information Processing Letters*, 2(4):108 – 112, 1973.

[60] Simon Klaus Degiorgi Daniele Giorgio. A dynamic algorithm for line graph recognition. *Graph theoretic concepts in computer science (Aachen, 1995)*, lectures Notes in Computer Science, 1017, berlin:37–48, 1995.

[61] Philippe G.H Lehot. An optimal algorithm to detect a line graph and output its root graph. *ACM 21(4)*, pages 569–575, october 1974.

[62] Bjarni V Halldórsson, Derek Aguiar, Ryan Tarpine, and Sorin Istrail. The clark phaseable sample size problem: long-range phasing and loss of heterozygosity in gwas. *Journal of Computational Biology*, 18(3):323–333, 2011.

[63] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.

[64] David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011.

[65] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[66] Anthony Bagnall, Jason Lines, Aaron Bostrom, and James Large. A review and experimental evaluation of recent advances in time series classification.