Algorithmes distribués de consensus de moyenne et leurs applications dans la détection des trous de couverture dans un réseau de capteurs

Anas Hanaf

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

ÉCOLE DOCTORALE SCIENCES TECHNOLOGIE SANTE (547)

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : ***AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE***

Présentée et soutenue publiquement par

Anas HANAF

Le 21 novembre 2016

ALGORITHMES DISTRIBUÉS DE CONSENSUS DE MOYENNE ET LEURS APPLICATIONS DANS LA DÉTECTION DES TROUS DE COUVERTURE DANS UN RÉSEAU DE CAPTEURS

Thèse dirigée par **Guillaume GELLE**

JURY

Mme. Maryline HELARD**,**	, Professeur,	INSA de Rennes,	**, Président**
M. Abbas DANDACHE,	, Professeur,	à l'Université de Lorraine,	**, Rapporteur**
M. Laurent CLAVIER,	, Professeur,	TELECOM Lille 1,	**, Rapporteur**
M. Mohamed TABAA,	, Docteur,	LPRI EMSI Maroc,	**, Examinateur**
M. Alban GOUPIL,	, Maître de Conférences,	à l'Université de Reims Champagne-Ardenne,	**, Examinateur**
M. Maxime COLAS,	, Maître de Conférences,	à l'Université de Reims Champagne-Ardenne,	**, Examinateur**
M. Guillaume GELLE,	, Professeur,	à l'Université de Reims Champagne-Ardenne,	**, Examinateur**

À ma chère mère
À mon cher père
À mon frère
À toute ma famille
À tous ceux
qui m'ont aidé dans la réalisation de ce travail

Remerciements

Au terme de ce travail, je tiens à remercier tous ceux qui y ont collaboré, directement ou indirectement, lors des quatre années de cette thèse.

Je tiens tout d'abord à remercier mon directeur de thèse M. Guillaume Gellé professeur et président de l'université de Reims Champagne-Ardenne, qui a su diriger cette thèse avec compétence et intérêt. J'ai pu au cours de ces quatre années de thèse profiter de ses conseils avisés et apprécier ses qualités scientifiques et humaines. Je tiens également à remercier mes co-encadrants de thèse, M. Alban Goupil et M. Maxime Colas pour m'avoir encadré au cours de ces quatre années de thèse. J'ai notamment apprécié d'avoir eu, tout au long de cette thèse, leurs conseils de grande rigueur scientifique, ainsi que des échanges très intéressants m'ayant permis d'enrichir mon expérience personnelle et professionnelle. Je remercie particulièrement M. Alban Goupil pour ses conseils et sa disponibilité pour mener à bien cette thèse. Je le remercie également pour la qualité scientifique de son travail qu'il m'a transmise, j'espère qu'elle m'aidera dans mon parcours professionnel. Merci beaucoup à vous trois, j'ai beaucoup appris à vos côtés et je vous adresse toute ma gratitude.

Je tiens à remercier M. Abbas Dandache et M. Laurent Clavier de m'avoir fait l'honneur de rapporter mes travaux de recherches.

Mes remerciements sont aussi adressés à Mme. Maryline Hélard pour avoir accepté de présider le jury.

Je remercie également M. Mohamed Tabaa pour avoir accepté de participer au jury.

Je tiens à remercier également tous les membres du Laboratoire CReSTIC qui m'ont offert une ambiance sympathique et amicale.

Enfin, je réserve une reconnaissance particulière à mes parents, qui m'ont encouragé à suivre ce chemin, ainsi que mon frère pour leur compréhension et leur soutien. C'est une grande joie de leur offrir le résultat de mon travail.

Résumé

Les algorithmes distribués de consensus sont des algorithmes itératifs de faible complexité où les nœuds de capteurs voisins interagissent les uns avec les autres pour parvenir à un accord commun sans unité coordinatrice. Comme les nœuds dans un réseau de capteurs sans fil ont une puissance de calcul et une batterie limitées, ces algorithmes distribués doivent parvenir à un consensus en peu de temps et avec peu d'échange de messages. La première partie de cette thèse est basée sur l'étude et la comparaison des différents algorithmes de consensus en mode synchrone et asynchrone en termes de vitesse de convergence et taux de communications. La seconde partie de nos travaux concerne l'application de ces algorithmes de consensus au problème de la détection de trous de couverture dans les réseaux de capteurs sans fil.

Ce problème de couverture fournit aussi le contexte de la suite de nos travaux. Il se décrit comme étant la façon dont une région d'intérêt est surveillée par des capteurs. Différentes approches géométriques ont été proposées mais elles sont limitées par la nécessité de connaitre exactement la position des capteurs ; or cette information peut ne pas être disponible si les dispositifs de localisation comme par exemple le GPS ne sont pas sur les capteurs. À partir de l'outil mathématique appelé topologie algébrique, nous avons développé un algorithme distribué de détection de trous de couverture qui recherche une fonction harmonique d'un réseau, c'est-à-dire annulant l'opérateur du Laplacien de dimension 1. Cette fonction harmonique est reliée au groupe d'homologie H_1 qui recense les trous de couverture. Une fois une fonction harmonique obtenue, la détection des trous se réalise par une simple marche aléatoire dans le réseau.

Mots clefs réseaux de capteurs, consensus de moyenne, détection distribuée, trou de couverture, topologie algébrique, Laplacien combinatoire

Abstract

Distributed consensus algorithms are iterative algorithms of low complexity where neighboring sensors interact with each other to reach an agreement without coordinating unit. As the nodes in a wireless sensor network have limited computing power and limited battery, these distributed algorithms must reach a consensus in a short time and with little message exchange. The first part of this thesis is based on the study and comparison of different consensus algorithms synchronously and asynchronously in terms of convergence speed and communication rates. The second part of our work concerns the application of these consensus algorithms to the problem of detecting coverage holes in wireless sensor networks.

This coverage problem also provides the context for the continuation of our work. This problem is described as how a region of interest is monitored by sensors. Different geometrical approaches have been proposed but are limited by the need to know exactly the position of the sensors ; but this information may not be available if the locating devices such as GPS are not on the sensors. From the mathematical tool called algebraic topology, we have developed a distributed algorithm of coverage hole detection searching a harmonic function of a network, that is to say canceling the operator of the 1-dimensional Laplacian. This harmonic function is connected to the homology group H_1 which identifies the coverage holes. Once a harmonic function obtained, detection of the holes is realized by a simple random walk in the network.

Keywords sensors network, consensus algorithm, distributed detection, coverage hole, algebraic topology, combinatorial Laplacian

Table des matières

Table des figures

Liste des tableaux

List of Algorithms

Chapitre 1

Introduction

1.1 Motivation

Dans un réseau de capteurs à grande échelle, la transmission de toutes les données vers un centre de fusion pour le traitement final est une mauvaise solution car le délai de communication ou le nombre de sauts à partir des nœuds de capteurs distants peut être extrêmement élevé et par conséquent peut entrainer une congestion dans le réseau. À cela s'ajoute un épuisement rapide des ressources énergétiques dû à un routage important des capteurs lointains vers le centre de traitement : les nœuds intermédiaires utilisent leurs ressources uniquement pour faire suivre des informations.

L'approche privilégiée dans ce genre de situation est le déploiement décentralisé avec un traitement des informations purement local entre nœuds de capteurs. Les calculs sont alors effectués de manière distribuée, grâce à des algorithmes distribués. Parmi ces derniers, apparaissent les algorithmes de consensus, et plus particulièrement les algorithmes de consensus de moyenne. Ils peuvent être définis comme étant des algorithmes itératifs à faible complexité, dont la consommation d'énergie est proportionnelle au temps.

Ces algorithmes itératifs communiquent entre eux en vue de parvenir à un accord commun au sujet d'une fonction des mesures, sans qu'il y ait un besoin de transmettre des informations au centre de fusion. Par exemple, l'algorithme distribué de consensus de moyenne calcule la moyenne des valeurs initiales attribuées aux nœuds de capteurs [OSM04]. L'objectif principal de cet algorithme est de trouver une mesure fiable de la moyenne tout en optimisant la vitesse de convergence.

Le problème du consensus de moyenne reçoit et a reçu beaucoup d'attention parce que, non seulement il illustre les avantages liés aux applications distribués [NAB06], mais il peut également être utilisé comme un élément de base pour des applications plus complexes en traitement du signal distribué [JLM06] et [XBL05]. Notons aussi que ces algorithmes de consensus peuvent être appliqués dans différentes applications liées aux réseaux de capteurs à savoir la détection de trous de couverture. Cette dernière présente la deuxième partie de nos travaux dans cette thèse.

La couverture est décrite comme étant la façon dont une région d'intérêt est surveillée ou suivie par des capteurs. En premier lieu, le domaine couvert par le RCSF ne doit pas contenir de trou de couverture.

Différents algorithmes à cet effet ont été largement étudiés [Gag92]. L'une des approches les plus utilisées pour faire face au problème de couverture a été l'approche géométrique, dans laquelle les outils géométriques standards tels que la triangulation de Delaunay ou le diagramme de Voronoï sont utilisés pour déterminer la couverture [CMKB02].

Toutefois, ces différentes approches géométriques sont ici limitées par la nécessité de connaitre exactement la position des capteurs. Or cette information peut ne pas être disponible si les dispositifs de localisation comme par des capteurs GPS ne sont pas installés sur les nœuds de capteurs pour des raisons de coût ou de contexte comme des RCSF en intérieur.

Récemment, les espaces topologiques et la topologie algébrique ont été utilisés pour répondre aux problèmes de couverture en l'absence de données géométriques telle que la position des capteurs [GM05]. Ainsi, la branche mathématique de la topologie algébrique [Hat01], [Mas91] et [Gib10] permet de résoudre ces problèmes en utilisant les outils d'algèbre linéaire qui permettent un calcul effectif par une caractérisation généralement matricielle.

1.2 Contributions de la thèse

Dans le cadre de cette thèse, nos travaux ont porté dans un premier temps sur l'utilisation des algorithmes distribués de consensus de moyenne. Nous avons en effet utilisé le principe de bavardage synchrone nommé Push-Sum développé par Kempe et al. [KDG03] pour l'adapter au modèle temporel asynchrone, et par la suite, nous avons intégré le principe du bavardage par diffusion [AYSS09] dans notre algorithme.

Ensuite, nous avons choisi un facteur de convergence que nous appelons β et qui assure la convergence de notre algorithme vers la moyenne. Pour cela, une étude statistique a été menée pour chercher la valeur optimale du paramètre β afin de converger plus rapidement vers le consensus.

Notons toutefois qu'une recherche bibliographique, nous a permis malheureusement de trouver des résultats récents similaires à notre algorithme mais avec une méthode différente

[ICHJ12]. Nous expliquerons en détail les deux résultats dans le chapitre 4.

Ces résultats ont été présentés lors des journées doctorales (JDSI) en 2014 et 2015 au Maroc.

Dans la deuxième partie des contributions, nous avons développé un nouvel algorithme de détection des trous de couverture dans un RCSF. Une première solution consiste à utiliser l'algorithme de consensus de moyenne. Pour découvrir de tels trous de couverture, nous avons construit un espace combinatoire abstrait et des résultats issus de la topologie algébrique. Cet algorithme a fait l'objet d'une présentation lors de la conférence internationale [HGCG15] 27th IEEE International Conference on Microelectronics (ICM) en 2015.

Nous avons ensuite proposé un deuxième algorithme distribué de détection de trous de couverture qui recherche une fonction harmonique d'un réseau, c'est-à-dire annulant l'opérateur du Laplacien de dimension 1. Cette fonction harmonique est reliée au groupe d'homologie H_1 qui recense les trous de couverture. Une fois une fonction harmonique obtenue, la détection des trous se réalise par une simple marche aléatoire dans le réseau. Ces résultats sont en cours d'écriture pour une soumission à un journal.

1.3 Structure de la thèse

Ce manuscrit se divise en six chapitres. Le chapitre suivant « Les réseaux de capteurs sans fil » fournit une introduction aux réseaux de capteurs sans fil. Les RCSF sont décrits avec leurs caractéristiques, leurs architectures, ainsi que des exemples de leurs applications. Ce chapitre présente la notion de connectivité et de couverture, qui est l'origine de nos travaux.

Le troisième chapitre « Théorie des graphes » présente les concepts fondamentaux de la théorie des graphes qui seront employés par la suite et nécessaires à la compréhension des résultats.

La première partie du chapitre 4 « Consensus de moyenne » introduit les algorithmes distribués de consensus avec les différents modèles temporels utilisés notamment selon leur caractère synchrone ou asynchrone.

Ce chapitre se poursuit par une étude théorique détaillée du consensus de moyenne dans le cas synchrone, à savoir sa convergence, sa vitesse de convergence et son application sur des topologies déterministes et aléatoires.

La dernière partie du chapitre présente notre première contribution. Pour cela, une analyse détaillée des différents algorithmes de bavardage dans le cas asynchrone a été faite, ainsi que leur convergence. Une fois terminée, cette étude servira de support au développement de notre contribution, à savoir la proposition d'un nouvel algorithme de bavardage appelé Push-Sum par diffusion avec une nouvelle méthode permettant d'assurer sa convergence vers la moyenne.

La présentation de la topologie algébrique débute le chapitre 5 « Détection des trous de couverture dans un RCSF.» Cet outil servira de base au développement de notre algorithme de détection distribuée des trous de couverture dans un réseau de capteurs sans fil qui complète ce chapitre.

Nous concluons notre travail dans le dernier chapitre en présentant un certain nombre de perspectives offertes par ces travaux.

Chapitre 2

Les réseaux de capteurs sans fil

Dans nos travaux, nous nous sommes intéressés principalement aux algorithmes distribués de consensus et aux problèmes liés à la couverture dans les réseaux de capteurs sans fil. C'est la raison pour laquelle, ce chapitre présente rapidement les RCSF en insistant sur leurs spécificités par rapport aux autres types de réseaux. Nous détaillerons leurs domaines d'application, leurs caractéristiques, ainsi que leur structure (section 2.1). La section 2.2 présente l'impact de la connectivité sur les performances d'un réseau de capteurs sans fil. Puis, la section 2.3 s'intéresse et présente plus précisément les différents types de couverture utilisés dans des applications RCSF. Enfin la section 2.4 conclura ce chapitre.

2.1 Les réseaux de capteurs sans fil (RCSF)

Les progrès technologiques dans la fabrication des circuits intégrés ont permis le déploiement d'unités de petites tailles, peu coûteuses et de faibles puissances. Ces unités sont appelées nœuds de capteurs et sont capables de mesurer les conditions physiques d'un environnement et de traiter l'information.

Ainsi, un réseau de capteurs sans fil peut être décrit comme un ensemble de nœuds de capteurs qui s'organisent en vue d'effectuer une action bien précise. Cette collaboration

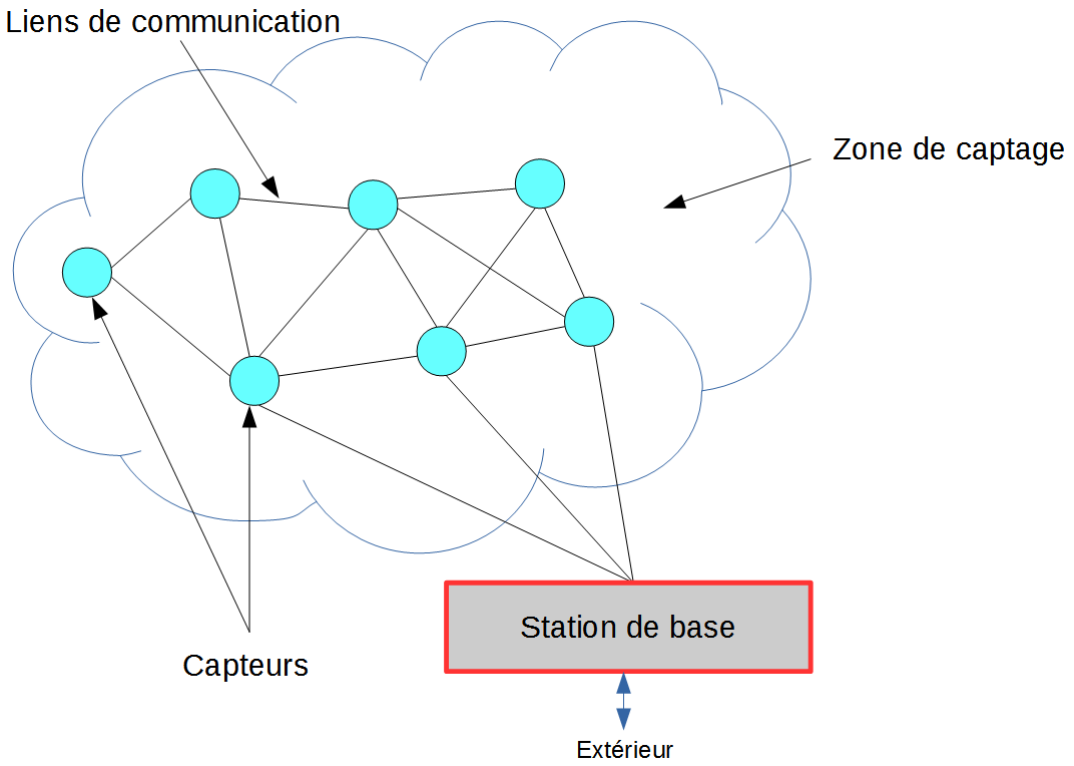


FIGURE 2.1 – Schéma d'un RCSF.

suit souvent le schéma de la figure 2.1 où les nœuds de capteurs communiquent via des liens de communication sans fil. Chaque nœud de capteur mesure l'environnement dans sa zone de captage, puis fait un traitement localisé ou distribué, et communique ses informations pertinentes à une station de base via des méthodes de communication appropriées. Ces informations peuvent être gérées localement ou à distance via un interfaçage avec des réseaux classiques comme Internet. Ainsi, l'architecture de communication d'un réseau de capteurs sans fil est légèrement différente par rapport aux réseaux informatiques (voir la figure 2.1).

2.1.1 Historique

On peut fixer la date de naissance des réseaux de capteurs aux années 50. En effet, la marine americaine avait des difficultés à localiser les sous-marins soviétiques en raison du manque de visibilité sous l'eau. À cet égard, ils ont développé un réseau d'hydrophones connectés appelé SOSUS (Sound Surveillance System) pour localiser ces sous-marins. SOSUS était un système composé d'hydrophones capables de détecter les sous-marins les plus proches par un microphone acoustique sous-marin. SOSUS est considéré comme l'un des premiers réseau de capteurs sans fil à grande échelle [CK93].

Vers les années 1980-1990, l'agence DARPA (Defense Advanced Research Projects Agency)

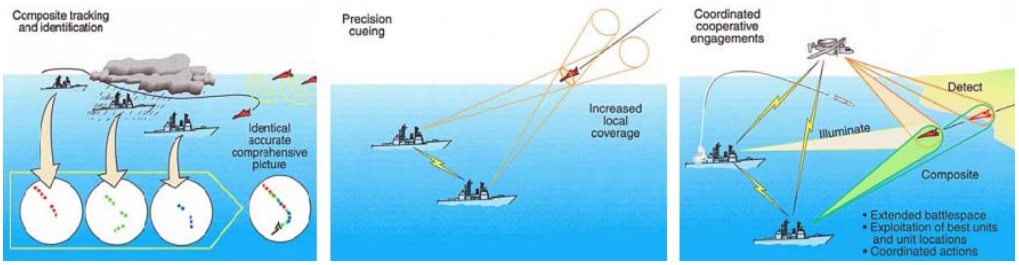


FIGURE 2.2 – Fonctions du CEC (Cooperative Engagement Capability).

a travaillé sur un réseau de capteurs distribués. Dans ce réseau, chaque nœud fonctionne de manière autonome et traite l'information de manière distribuée [ONe07]. Au début des années 1990, dans le cadre du « Cooperative Engagement Capability, » la marine américaine a installé un nouveau système utilisant les informations détectées par des navires voisins. Ce type de communication produit une image plus claire de la cible.

À partir des années 2000, l'agence DARPA a développé une nouvelle classe de logiciels pour des micro-capteurs distribués [KS01]. Ces travaux rentrent dans le cadre du programme SensIT (Sensor Information Technology) pour le développement de nouvelles méthodes de mise en réseau et de détection entre micro-capteurs distribués.

L'université de Californie, Los Angeles, a développé une architecture de réseau de capteurs sans fil appelée WINS [PK00] qui fournit une nouvelle capacité de surveillance et de contrôle (frontière du pays). De plus, le MIT a réalisé un projet appellé μAMPS [Cha] et l'université de Berkley un programme appelé Smart dust [PKB99] ou PicoRadio [RAd$^+$00]. La collaboration de ces différentes entités scientifiques spécialisées dans les communications numériques RCSF a permis l'émergence de la technologie ZigBee.

2.1.2 Caractéristiques des RCSF

Un RCSF se compose généralement d'un grand nombre de nœuds de capteurs de faible coût et de faible puissance qui sont déployés dans une région d'intérêt (voir figure 2.1). Ces nœuds de capteurs sont de petites tailles, équipés de microprocesseurs embarqués et d'émetteurs-récepteurs radio. De plus, la capacité de ces petites unités ne se limitent pas à la détection mais peuvent aussi faire un traitement localisé et distribué. Ces nœuds de capteurs communiquent entre eux sans fil dans l'objectif d'accomplir une tâche commune. Les RCSF possèdent des caractéristiques et des contraintes uniques par rapport à d'autres réseaux traditionnels de communication sans fil tels que les systèmes cellulaires et les réseaux mobiles ad-hoc : ils sont généralement alimentés par des batteries. Dans la plupart des situations, ils sont déployés dans des environnements hostiles, où il est très difficile, voire impossible de changer ou de recharger les batteries. De plus, en raison du grand nombre de nœuds de capteurs, il est généralement impossible de mettre en œuvre un système d'adressage global pour un RCSF, car il introduirait une surcharge élevée pour le maintien de l'identification.

Un RCSF est habituellement conçu et déployé pour des applications spécifiques. Autrement dit, les exigences en matières de conception dépendent du choix de l'application.

La topologie d'un RCSF change fréquemment en raison d'une défaillance d'un nœud de capteurs, de l'épuisement d'énergie ou des contraintes liées au canal de transmission. En effet, les nœuds de capteurs peuvent être déployés aléatoirement dans des environnements hostiles où le maintien de connectivité du réseau est difficile, ainsi, ils sont exposés à des dommages physiques.

L'auto-organisation est très importante dans le cas des réseaux de capteurs à grande dimension, où la gestion du flux par un seul nœud de capteur n'est pas possible. En effet, les nœuds de capteurs doivent établir une structure organisationnelle qui ne nécessite pas de coordination centrale. En d'autres termes, les entités interagissent directement les unes avec les autres et réagissent de façon continue à des changements dans leur environnement local. Les auteurs de [PB05] ont décrit un système auto-organisé en prenant comme exemple un banc de poissons. Tout d'abord, les auteurs montrent que ce système s'adapte en permanence aux changements d'une manière coordonnée, de telle sorte que le système se réorganise en réaction à différents changements. Une autre caractéristique importante des systèmes auto-organisés est la robustesse, le système doit être en mesure de fonctionner malgré l'endommagement d'un de ses éléments.

2.1.3 Architecture des RCSF

En raison de leurs caractéristiques, les RCSF proposent de nombreux défis de développement et de recherche. La structure du réseau est très importante et doit être prise en compte avant le déploiement du RCSF. Le réseau de capteurs doit être conçu de manière à ce que toutes les unités collaborent efficacement pour pouvoir remonter l'information à l'utilisateur final. Pour ce faire, certains objectifs de conception d'architecture doivent être considérés [Hil03].

Les nœuds de capteurs possèdent des ressources limitées. Par conséquent, il est important de concevoir le réseau de manière optimale de telle sorte que les nœuds de capteurs effectuent des tâches avec un minimum de ressources en basculant par exemple les nœuds de capteurs dans le mode sommeil pour augmenter leur durée de fonctionnement.

Les RCSF sont dynamiques et peuvent être constitués de différents types de nœuds de capteurs. Tout d'abord, le déploiement d'un nœud de capteur doit prendre en considération les coûts. Ensuite, comme les capteurs sont à bas coût, ils peuvent être facilement endommagés, ainsi il faut éviter que ces dommages paralysent le réseau. Sans oublier la consommation d'énergie qui doit être prise en compte pour augmenter la durée de vie de l'ensemble du réseau.

La structure d'un nœud de capteur est constituée d'une unité sensorielle (capteurs et convertisseurs analogique-numérique), d'une unité de communication, d'une unité de traitement et d'une alimentation électrique généralement une batterie [SK11]. Les principaux

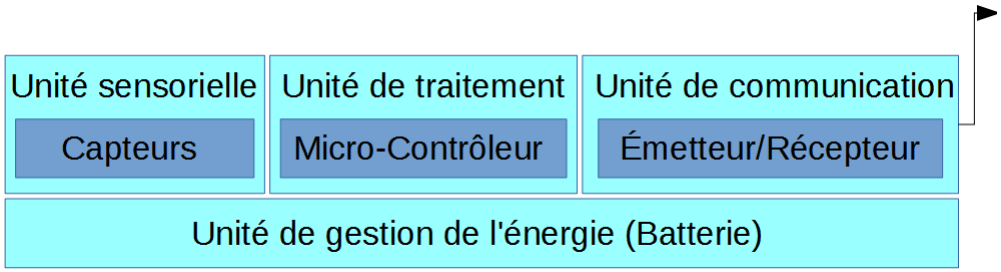


FIGURE 2.3 – Synopsis d'un capteur.

éléments constituant un nœud de capteur sont présentés dans la figure 2.3 et développés ci dessous :

- **Unité sensorielle (de captage)** Elle est composée de différents types de capteurs qui mesurent l'état de l'environnement immédiat. Les capteurs sont choisis en fonction de l'application par exemple des capteurs pour la surveillance d'un environnement (température, humidité,...) [LSL+04]. Ils collectent des signaux physiques analogiques qui doivent être conditionnés puis transformés en signaux numériques. L'objectif de cette conversion est de pouvoir communiquer avec le micro-contrôleur.
- **Unité de communication** Elle utilise un émetteur-récepteur. La communication est réalisée via un support de communication sans fil (radio, infrarouge, communication optique,...). Cette unité qui consomme le plus d'énergie en comparaison aux autres utilise des protocoles tels que ZigBee, LoRa et Sigfox [KW07].
- **Unité de traitement** Elle se compose d'un micro-contrôleur [LSL+05] avec une mémoire de stockage. Le rôle de l'unité de traitement comprend la collecte de données provenant de diverses sources (capteurs et réseau), puis le traitement et le stockage. De plus, cette unité utilise des algorithmes distribués pour répondre aux requêtes du système ou de l'opérateur.
- **Unité de gestion de l'énergie** Cette unité doit fournir l'énergie au noeud de capteur. En effet, la durée de vie d'un capteur dépend principalement de la batterie ou d'une puissance qui est reliée à l'unité d'alimentation. L'unité de gestion de l'énergie est nécessaire pour une utilisation efficace de la batterie. Il existe des possibilités de reconstitution de l'énergie consommée par un réapprovisionnement en cherchant d'autres sources externes comme l'énergie solaire [SK11], ou la mise en veille de certains capteurs dans le réseau [GD08] et [ZZSH12].

Les applications dans certains domaines requièrent de plus en plus de capteurs miniaturisés. La figure 2.4 présente quelques réalisations de capteurs qu'on peut trouver sur le marché et entièrement intégrés dans une seule puce (System On Chip).

2.1.4 Critères de conception

Pour concevoir un RCSF en vue d'une application, il faut prendre en considération certains aspects concernant l'unité de captage, de communication et de traitement. L'environne-

FIGURE 2.4 – Exemples de réalisation de capteurs.

ment s'ajoute aussi aux aspects à prendre en compte car naturellement, il influence le choix de l'application du RCSF. Nous expliquons ci-dessous les différents points essentiels à satisfaire pour concevoir un réseau de capteurs sans fil performant.

Environnement Les nœuds de capteurs sont conçus pour détecter un événement dans leur environnement. Cependant, ils sont déployés dans des zones éloignées sans intervention humaine, cela peut être le fond de l'océan, le volcan en éruption ou encore dans des zones dangereuses comme les champs de bataille. Les nœuds de capteurs doivent être en mesure de résister aux contraintes de l'environnement, tout en collectant le maximum de données environnementales.

Tolérance aux fautes Certains nœuds de capteurs peuvent être endommagés ou bloqués en raison d'un manque de puissance. Mais, l'échec de certains nœuds de capteurs ne doit pas paralyser le réseau. Le RCSF doit pouvoir maintenir ses fonctionnalités même si des nœuds de capteurs sont endommagés. Par exemple, les auteurs [LZWM15] utilisent une approche qui vise à améliorer la qualité de détection d'évènements tout en réduisant le scénario de dépendance. La solution consiste à détecter rapidement le défaut du nœud de capteur selon un seuil critique. Cependant, ces résultats concernent uniquement la détection d'un seul capteur défaillant et ne permettent pas de résoudre le problème quand on est face à plusieurs nœuds de capteurs endommagés.

Passage à l'échelle Le passage à l'échelle est un facteur très important pour mesurer la performance d'un réseau de capteurs sans fil : pour cela l'utilisation d'algorithmes distribués est primordiale. Le système doit être en mesure de supporter l'expansion du réseau de capteurs sans fil afin d'éviter la congestion des transmissions.

Un autre problème qui doit être pris en considération concerne le bottleneck ou goulot d'étranglement : comme le trafic de données dans un RCSF peut être concentré vers un nombre restreint de nœuds de capteurs proches de la station de base, ces derniers doivent transmettre les données à d'autres nœuds de capteurs lointains dont le nombre peut être grand. Ainsi, ces nœuds de capteurs déployés autour de la station de base épuisent leurs batteries beaucoup plus rapidement que les autres, et leur durée de vie s'en trouve amoindrie. Des solutions ont été proposées afin de résoudre ce problème, par exemple, les

auteurs de [LH05] proposent que la station de base soit mobile, afin de mieux répartir la charge entre les nœuds de capteurs.

Contraintes matérielles Comme on a vu précédemment, chaque RCSF se compose de quatre éléments de base. La taille des capteurs est l'une des contraintes matérielles du RCSF. De même, la qualité des capteurs permet de déterminer si un RCSF peut résister à l'endommagement de certains capteurs. Il existe d'autres contraintes matérielles pour la conception d'un RCSF : les nœuds de capteurs doivent par exemple avoir un coût de production réduit car le coût de production d'un seul capteur est très important pour évaluer le coût global du réseau, et s'ajoute à cela, l'utilisation de puces GPS qui s'avèrent très coûteuses en terme de matériel.

Transmission Selon le choix de l'application, le bloc de transmission d'un RCSF doit être choisi avec soin. En effet, le module radio est le principal responsable de l'épuisement des batteries. Pour résoudre ce problème, des chercheurs ont essayé d'optimiser les paramètres radio à savoir la modulation, les antennes et la transmission de puissance.

Par exemple, l'auteur [CO10] compare trois schémas de modulation (MPSK, MFSK et MQAM) et propose des paramètres optimaux pour réduire la consommation d'énergie. L'objectif est de trouver un compromis entre le débit d'information, le temps de transmission, la taille de la constellation, la distance entre les nœuds de capteurs et le bruit.

D'autres solutions consistent à réduire la quantité de données envoyée à la station de base en vue d'augmenter la capacité réelle du réseau telles que la compression des données [KL05], le codage réseau [WWJ+11] et l'agrégation des données [RV06]. En effet, le débit n'est pas primordial dans les RCSF comme il l'est pour les réseaux plus classiques.

2.1.5 Quelques applications des RCSF

Les RCSF peuvent être utilisés pour détecter ou surveiller un certain nombre de paramètres ou de conditions physiques [ASSC02] (humidité, lumière, son, température, qualité de l'eau, position, vitesse, direction,...). Nous faisons dans cette section un rapide survol d'applications des RCSF selon leur domaine.

Les réseaux de capteurs sans fil ont des avantages significatifs par rapport aux réseaux câblés classiques. Ils peuvent être appliqués à tout environnement, en particulier ceux dans lesquels, il est impossible de déployer des réseaux câblés classiques. En effet, les RCSF ont été initialement motivés par des applications militaires qui vont des systèmes de surveillance à grande échelle aux petits réseaux de capteurs autonomes (surveillance de l'environnement) [LWS04]. Des exemples d'applications des RCSF sont donnés dans la figure 2.5.

FIGURE 2.5 – Exemples d'applications des RCSF.

Domaine environnemental Les réseaux de capteurs servent à prévenir et aider les secours en cas de catastrophes naturelles comme des éruptions volcaniques [WALW+06], des feux de forêt [YWM05] et des inondations. Par exemple, la surveillance des tsunamis est devenue une préoccupation majeure des chercheurs dans ce domaine. Tout d'abord, des scientifiques ont commencé par déployer des bouées de capteurs en mer qui détectent la pression de l'eau. Cependant, cette solution a des inconvénients, car l'acquisition des données se fait manuellement et retarde entre autres, la prise de décisions des autorités en cas de tsunamis. Une autre solution pour la détection des tsunamis [CLD08] a été proposée. L'idée se base sur le déploiement de capteurs souterrains pour surveiller la pression de l'eau. La nouveauté de cette solution par rapport à la précédente est l'acquisition des données en temps réel et la possibilité pour les autorités locales de prendre des décisions rapides pour évacuer la population en danger.

D'autres travaux [MPS+04] s'intéressent à la surveillance de l'habitat des animaux en déployant des capteurs afin de faire une étude environnementale d'une petite île du Maine. Ces mêmes capteurs mesurent l'humidité, la température, la pression atmosphérique ainsi que l'occupation des terriers de pétrel.

Domaine lié à la sécurité Dans un cadre lié à la sécurité des personnes, les auteurs [SHW+10] ont déployé des capteurs dans trois infrastructures de bâtiment à savoir le Ferriby Road bridge, le Humber suspension bridge et le tunnel de la ligne Jubilee du métro de Londres. Ils ont placé 26 capteurs dans le tunnel souterrain afin de contrôler la température, l'humidité dans le béton et l'inclinaison. Les auteurs admettent que le déploiement du RCSF est encore en phase de prototype et exige beaucoup de recherche

avant un usage commercial. D'autres travaux traitent la même thématique sur la capacité des RCSF à suivre l'évolution de bâtiment [Che04] et [BSW+10].

Un autre problème lié à la sécurité est celui du terrorisme, car les risques d'activités terroristes sont susceptibles de se produire dans les lieux à forte population. Ainsi, les RCSF ont permis d'éviter les contraintes de déploiement en raison de leurs caractéristiques uniques à savoir leur petite taille, leur faible coût et l'auto-organisation de ses entités. Une meilleure détection d'intrusion d'individus ou de véhicules en stationnement minimise les risques du terrorisme et facilite le travail de la sûreté nationale.

Domaine médical Selon un rapport de l'inspection générale des affaires sociales en France [gdas12], la prospérité d'une nation se mesure par la qualité des installations de soins de santé dont jouissent ses citoyens. Les salles d'urgence sont toujours surpeuplées alors que la santé du patient court un risque en raison du nombre limité du personnel hospitalier. La solution la plus évidente pour réduire le surpeuplement des salles d'urgence et faciliter l'accès aux soins à tous les citoyens est tout simplement d'avoir l'avis du médecin sans qu'il soit présent physiquement. Ainsi, un RCSF capable de surveiller les signaux vitaux du patient en continu et de transmettre les informations enregistrées par le biais du réseau de capteurs est une solution prometteuse [MSMSMSM13].

Le déploiement d'un réseau de capteurs peut se faire de façon différente [Mor07]. Le premier exemple consiste à surveiller les asthmatiques à domicile, en détectant par exemple les odeurs qui se dégagent des tapis pour éviter les crises d'asthme. Le deuxième exemple concerne un patient atteint d'une blessure osseuse, le RCSF est capable de surveiller le mouvement du patient et d'envoyer les données au médecin spécialiste pour faire le diagnostic. Le dernier exemple est une combinaison des deux exemples précédents en surveillant les signaux vitaux du patient et son milieu environnant.

Les informations médicales du patient transmises au médecin doivent être conservées et sécurisées. L'article [SP04] présente des outils pour la sécurisation de ces données confidentielles au niveau des RCSF.

Les auteurs de [KGT09] ont présenté un protocole d'application médicale MEDiSN. Le réseau de capteurs sans fil est déployé dans une salle d'urgence d'un hôpital. Les capteurs mesurent la fréquence cardiaque ainsi que le niveau d'oxygène dans le sang des patients.

Domaine commercial Les réseaux de capteurs peuvent être intégrés dans un processus à la fois industriel et commercial. En effet, le contrôle de la qualité et le suivi des marchandises pendant la phase de transport devient une préoccupation croissante pour les fournisseurs et les producteurs. Le transport est souvent effectué par des véhicules frigorifiques et les conteneurs sont équipés de systèmes de refroidissement intégrés. Dans un tel environnement, la température augmente très rapidement si une unité du système échoue comme l'a montré [RGBRL10]. Les auteurs de [SLPB04] ont proposé un RCSF dans des véhicules frigorifiques en utilisant ZigBee comme protocole de communication en raison de sa faible consommation d'énergie [WC07].

Ces dernières années la domotique [TVP+11] devient de plus en plus présente chez les particuliers. Les gens veulent vivre dans des espaces de vie intelligents équipés de systèmes domotiques [NVR+11]. Ces installations non seulement leur fournissent le confort, mais aussi la sécurité. Ainsi, la demande des réseaux de capteurs croît de plus en plus chez les particuliers car de nombreuses PME rendent ce marché accessible au plus grand nombre.

Domaine militaire Les applications militaires sont certainement les plus représentatives des applications trouvées aujourd'hui dans le domaine des réseaux de capteurs sans fils. En effet, les caractéristiques des RCSF telles que la tolérance aux pannes, l'auto-organisation et le déploiement rapide en font des atouts majeurs dans le domaine militaire. Il existe plusieurs types de capteurs [DTDM12] tels que des capteurs acoustiques déployés à grande échelle [LLSL09] dans des champs de bataille ou des zones hostiles pour surveiller le mouvement des troupes.

La sécurité dans les RCSF est primordiale dans le domaine militaire, des solutions comme dans [SP04] peuvent être utilisées pour ce type d'application.

Domaine écologique La gestion des déchets dans les zones urbaines est essentielle en raison de la forte concentration de la population. Traditionnellement, les déchets solides sont stockés et évacués périodiquement par des camions sans aucune optimisation des itinéraires. Une nouvelle façon d'aborder ce problème est l'utilisation des RCSF pour surveiller l'état des poubelles et faire un calcul optimal de l'itinéraire, afin de minimiser le coût du carburant. Les auteurs de [CV14] ont déployé des capteurs pour mesurer non seulement la quantité des déchets jetés dans la poubelle, mais aussi pour surveiller la position des poubelles à travers un système GPS.

La surveillance du climat est une préoccupation d'une grande importance dans le monde entier. Des conférences sur le climat comme la COP21 à Paris en 2015 ont permis d'élaborer un plan de travail à long terme dans le financement des projets de recherche afin de minimiser l'émission de CO_2. Jusqu'à aujourd'hui, des travaux comme [MRG+08] ont permis de trouver des solutions partielles à ce problème en déployant des capteurs mobiles et statiques. Les capteurs sont basés sur la technologie GUSTO (Generic Ultra violet Sensor Technologies and Observations) qui utilise les rayons ultraviolets pour mesurer la quantité de CO_2 en temps réel. Les auteurs de [MRG+08] étudient actuellement la fusion de données et des techniques d'agrégation afin d'améliorer la performance de ce système lorsque de grandes quantités de données sont collectées et transmises comme par exemple les données relatives au trafic et les données météorologiques.

2.2 Connectivité

La connectivité est une notion particulièrement importante puisque les données collectées par les nœuds de capteurs doivent être acheminées vers des centres de données générale-

ment des stations de base. Ce processus est uniquement possible s'il existe un chemin à partir de chaque nœud de capteur.

Un réseau est dit connecté si le graphe associé est connecté, c'est-à-dire qu'entre deux sommets du graphe, il existe au moins un chemin [Die00]. Néanmoins, la connectivité ne mesure pas la capacité du RCSF à résister à la défaillance de certains liens de communications entre capteurs. Ainsi, il est utile d'envisager des critères de connectivité plus puissants tels que la k-connectivité pour répondre à ce problème en calculant le nombre minimal de chemins indépendants k entre deux nœuds distincts [Die00].

La performance d'un RCSF se mesure par la connectivité du graphe du réseau. Il en va de même pour le routage. La mise en place d'un routage nécessite la prise en compte des spécificités des RCSF. En effet, le routage doit être performant pour résister à la défaillance des nœuds de capteurs. Une solution basée sur le clustering permet de réaliser un routage efficace dans un RCSF [TM03]. Elle permet de simplifier les opérations de routage. Le RCSF est divisé en plusieurs groupes et chaque groupe est géré par une tête de cluster. Les communications inter-clusters se produisent seulement entre les têtes de cluster. Toutefois, les nœuds de capteurs doivent d'abord se connecter avec leurs tête de cluster avant de communiquer avec le reste du réseau.

La connectivité d'un RCSF est directement lié à la position géographique des nœuds de capteurs et de la portée des communications. Il existe deux types de déploiement de capteurs : le déploiement déterministe où les nœuds de capteurs sont placés à des positions bien précises et choisies selon des critères spécifiques et le déploiement aléatoire où les nœuds de capteurs sont déployés sans contrôle particulier. Le choix du schéma de déploiement dépend fortement du type de capteurs, de l'environnement où les nœuds de capteurs sont déployés et de l'application. Dans certaines situations, le déploiement déterministe est envisageable. La connaissance préalable de la région d'intérêt, ainsi que l'utilisation de capteurs coûteux sont des facteurs déterminants de ce choix du déploiement déterministe. Ce type de déploiement est généralement utilisé pour des applications de surveillance à l'intérieur de bâtiment comme par exemple, Accenture Technology avec son projet de capteur multiple de surveillance à l'intérieur de batiments [PSR+06] et l'université de Sidney avec son projet Actif Sensor Network (ASN) [BMK04]. Ces projets sont orientés vers des applications de surveillance telles que la gestion des actifs et la surveillance des installations sécurisées. Le déploiement déterministe est également utilisé dans des applications de surveillance corporelle (signaux vitaux d'un patient,...), des applications de contrôle (maintenance industrielle, domotique,...) et des applications multimédia (imagerie, caméra,...). Ce type d'applications exige des mesures précises et de qualité, ainsi, la position des nœuds de capteurs joue un rôle important pour renforcer la connectivité du réseau et assurer son bon fonctionnement. De plus, dans le déploiement déterministe, il est facile de déterminer si le réseau est connecté, et d'ajouter si nécessaire des nœuds de capteurs intermédiaires.

Par contre, dans le déploiement aléatoire, on retrouve des applications où le réseau est de grande taille et inaccessible par exemple dans une forêt ou autour d'un volcan, ainsi, les nœuds de capteurs sont déployés aléatoirement. De ce fait, l'analyse de la connectivité

d'un tel réseau nécessite la modélisation d'un réseau aléatoire.

Après une certaine période, les capteurs peuvent être endommagés ou indisponibles en raison d'un incident extérieur, ainsi le nombre et la position de ces capteurs endommagés peuvent fragiliser le réseau. Il est donc nécessaire d'intervenir par un redéploiement des capteurs afin d'assurer les objectifs du RCSF. Il est aussi possible d'exploiter la mobilité de certains capteurs dans la détection d'un incendie ou le suivi d'un intrus.

De nombreux résultats sur la connectivité dans les RCSF sont dérivés de la théorie de la percolation [Hal85] et [Men86]. La notion de percolation a été introduit par [Bro54] et [BH57] pour modéliser mathématiquement un fluide circulant à travers des matériaux. L'étude de la percolation sur des graphes aléatoires infinis avec un ensemble infini de sommets peut être transposée sur des réseaux sans fil qui contiennent un nombre infini de nœuds de capteurs. Dans un plan infini, la probabilité qu'au moins un nœud de capteur soit isolé tend généralement vers 1, ainsi la connectivité du réseau est dite partielle.

Néanmoins, des solutions [HM06] ont permis de quantifier le problème de connectivité dans les réseaux aléatoires. Tout d'abord, les auteurs montrent que les variations aléatoires de la puissance du signal radio ont des effets positifs sur la connectivité du réseau en réduisant principalement la corrélation entre les liens (moins de connexions inutiles entre les nœuds). Ensuite, ces variations dues par exemple aux obstacles lors de la transmission peuvent augmenter la probabilité des liaisons lointaines, ce qui augmente la probabilité de connectivité du réseau.

Certaines techniques ont été proposées pour améliorer la connectivité en élargissant la portée de transmission des nœuds de capteurs. Ces techniques concernent par exemple les transmissions coopératives [SH03]. En effet, ce type de transmissions permet d'accumuler la puissance provenant des différents nœuds de capteurs afin d'obtenir une plus grande puissance pour transmettre des données identiques, et par conséquent, la portée de transmission peut être élargie et naturellement, avoir une connectivité améliorée. Ces méthodes sont efficaces pour éviter l'isolement de certains nœuds de capteurs.

D'autres techniques permettent aussi d'améliorer la connectivité dans des RCSF en utilisant des antennes directionnelles [KKW05]. En effet, l'utilisation de ces antennes permet de réduire la puissance d'émission, car la transmission est effectuée dans une seule direction. En d'autres termes, cette approche diminue le gaspillage d'énergie durant la transmission. Ainsi, les nœuds de capteurs peuvent transmettre des informations à des récepteurs lointains, ce qui bien évidemment améliore la connectivité du réseau.

L'objectif étant de minimiser le nombre de nœuds de capteurs tout en garantissant la détection. Dans la section suivante, nous détaillerons un peu plus la notion de couverture et les contraintes liées à la connectivité. De ce fait, il existe un compromis entre une densité plus élevée du réseau qui a pour objectif de renforcer la connectivité du RCSF mais avec un risque d'interférences entre les nœuds de capteurs et une densité moins élevée dont l'objectif est de minimiser les interférences entre les nœuds de capteurs afin d'améliorer la probabilité d'erreur de transmission.

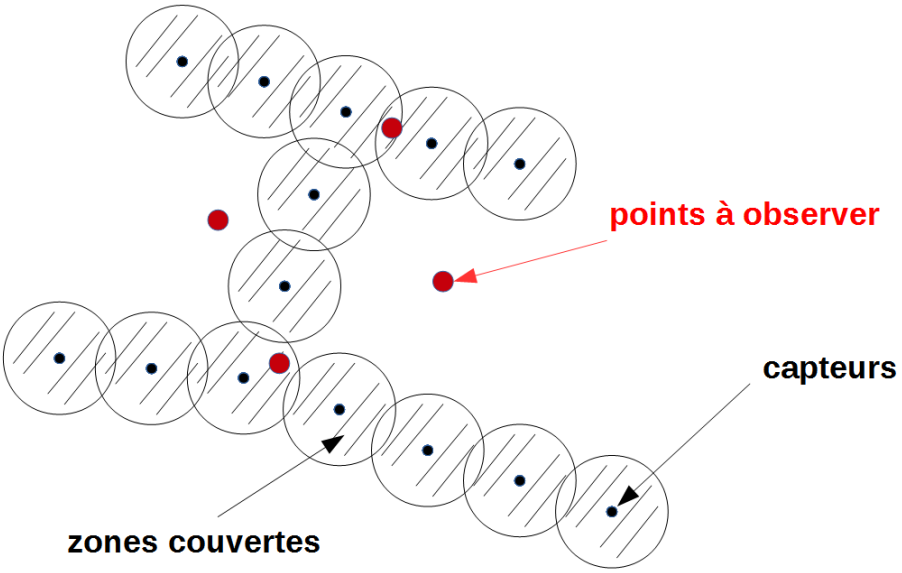


FIGURE 2.6 – Couverture ponctuelle.

2.3 Couverture

La fonction fondamentale d'un réseau de capteurs sans fil est de surveiller des zones ou des cibles. Comme les nœuds de capteurs sont souvent déployés dans des environnements éloignés ou inaccessibles, la question de couverture se pose dans ce contexte. Nous décrivons ci-dessous les différents types de couverture utilisés dans des applications RCSF.

— **Couverture ponctuelle** (point coverage) consiste à couvrir un ensemble de points particuliers dans la zone où le RCSF est déployé. La figure 2.6 montre un exemple d'un ensemble de capteurs déployés aléatoirement pour surveiller une zone en utilisant une couverture ponctuelle.
La couverture présentée dans [CD05] est destinée à une application militaire. Les auteurs considèrent un nombre limités de points avec des positions connues qui doivent être surveillés. Un grand nombre de capteurs est dispersé de manière aléatoire à proximité de la cible. Le principe de cette couverture consiste à ce que chaque cible soit surveillée par au moins un capteur et que chaque capteur soit en mesure de surveiller toutes les cibles dans sa zone de détection. D'après [CD05], l'objectif de cette approche consiste à ce que l'intervalle de temps entre deux activations pour chaque capteur soit le plus long possible. Ainsi, la distribution des tâches au niveau des capteurs permettra d'économiser l'énergie et naturellement la durée de vie du réseau. D'autres travaux [KB03] utilisent la couverture ponctuelle pour la surveillance d'une zone. Le déploiement des capteurs se fait d'une manière déterministe. L'objectif est de déterminer un nombre minimal de capteurs et leurs positions de telle sorte que les points cibles connus soient couverts.
— **Couverture de frontière** (barrier coverage) Le principe de cette couverture est

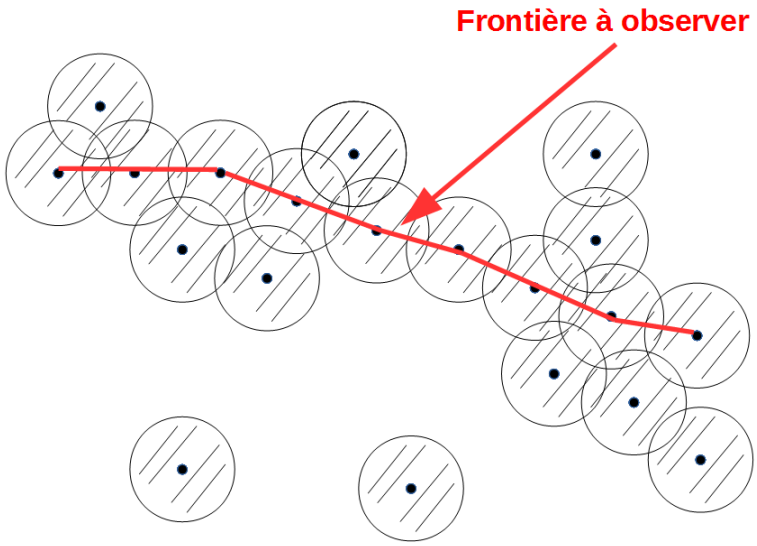


FIGURE 2.7 – Couverture de frontière.

de déployer des capteurs en vue de détecter les intrusions qui passent à travers la frontière d'une région. Elle a été étudiée pour la première fois dans le cadre de robots coopératifs pour des applications militaires [Gag92]. De nombreux travaux traitent le problème d'intrusion [LDWS08] et [MKQP01]. Un exemple de couverture de frontière est illustré dans la figure 2.7.

— **Couverture glissante** (sweep coverage) L'objectif principal de cette couverture est la réalisation d'une tâche de surveillance en utilisant un ou plusieurs capteurs mobiles. La mobilité des capteurs leur permet de couvrir une région plus grande comme l'illustre la figure 2.8. Ce système est efficace dans des environnements de surveillance (forêts, volcans,...). Des algorithmes distribués [LCL+11] ont été proposés pour assurer la mobilité des capteurs et l'échange d'information de localisation entre eux, ainsi les capteurs sont capables de détecter les changements sur les trajectoires. D'autres outils [Ada13] peuvent aussi aider à déterminer si la couverture est garantie sachant que les capteurs sont en mouvement.

— **Couverture de zone** (area coverage) La couverture la plus utilisée dans le domaine des RCSF est la couverture de zone où l'objectif principal du RCSF consiste à couvrir une zone comme le montre la figure 2.9. La majorité des applications militaires, environnementales ou autres utilisent la surveillance d'une zone. Le principe de k-connectivité est ici étendue par la notion de k-couverture [HT03] qui permet de connaitre le nombre de capteurs déployés pour chaque point de la zone à observer. Si le déploiement du réseau n'est pas aléatoire, alors il faut choisir des positions optimales pour les capteurs de façon à assurer la couverture [BKX+06], la k-couverture [BXLJ08] et la connectivité du réseau.

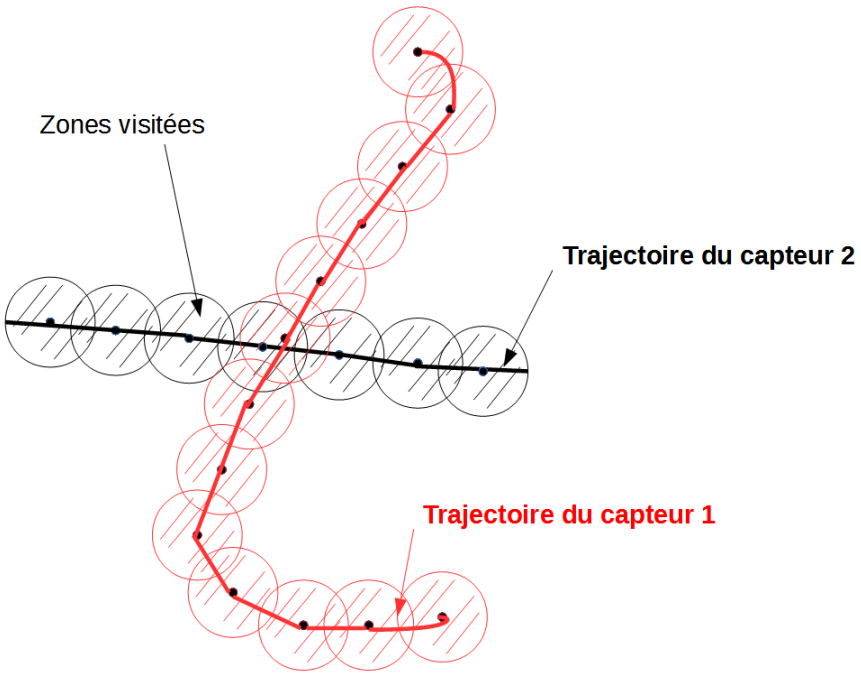


FIGURE 2.8 – Couverture glissante.

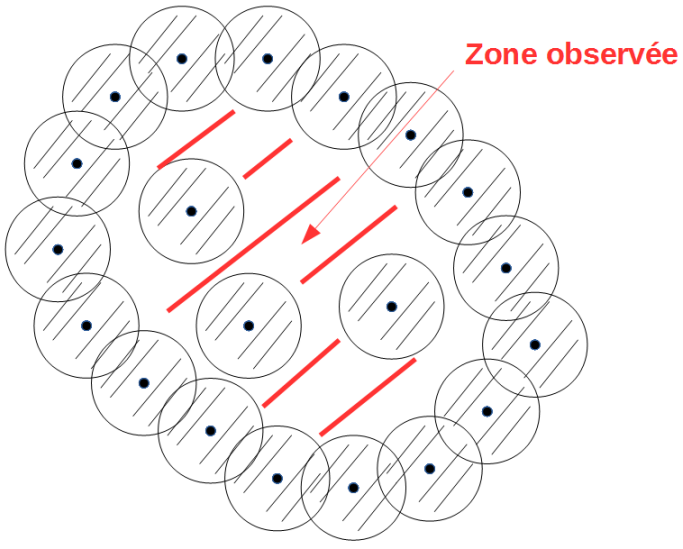


FIGURE 2.9 – Couverture de zone.

2.4 Conclusion

Dans ce chapitre, nous avons détaillé les différents aspects des réseaux de capteurs sans fil à savoir leur définition, leurs caractéristiques et leurs contraintes. Nous avons aussi détaillé la notion de connectivité et de couverture qui est à l'origine de nos travaux.

Dans le chapitre suivant, nous présenterons les différentes notions de la théorie des graphes. Ensuite, nous passerons en revue les différentes topologies de graphes largement utilisées en réseau et plus précisément, le graphe géométrique aléatoire qui est le modèle le plus représentatif des RCSF.

Chapitre 3

Théorie des graphes

Dans ce chapitre, nous présentons les concepts fondamentaux de la théorie des graphes [Die00] et [GR01] nécessaires à la compréhension de la suite du manuscrit. Pour cela, quelques notions de base sur la connectivité des graphes orientés et non-orientés y seront présentées notamment par des exemples simples. Ensuite, nous présentons les différentes représentations algébriques des graphes [GR01]. Nous allons donc dans un premier temps expliquer de manière générale les différentes matrices associées aux graphes [Mur09]. Ensuite, nous allons rappeler quelques concepts de base du spectre du Laplacien ainsi que ses propriétés [Chu97] et [Moh91]. La dernière section sera consacrée à la présentation de quelques topologies de graphes largement utilisées dans la littérature notamment en réseau et qui fourniront les réseaux de nos simulations.

3.1 Généralités

Considérons un réseau de communication statique où les capteurs communiquent en fonction d'une topologie réseau spécifique à l'application. Cette topologie peut être représentée à l'aide d'un graphe G noté $G = (V, E)$, où V est l'ensemble des sommets (capteurs) et E

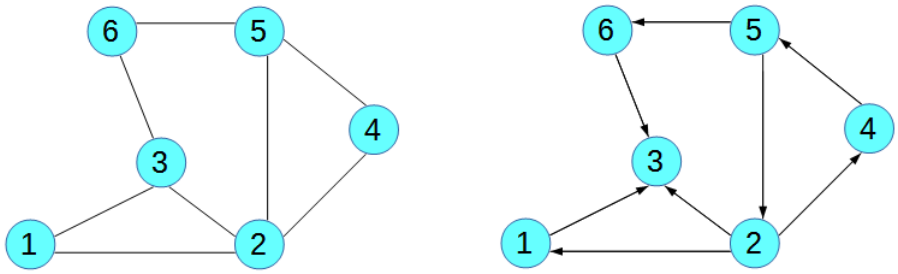


FIGURE 3.1 – Graphe non-orienté avec $N = 6$ sommets (à gauche) et graphe orienté avec $N = 6$ sommets (à droite).

est l'ensemble des arêtes e_{ij} entre le sommet i et le sommet j. Le lien de communication reliant deux sommets i et j est représenté par e_{ij}.

Les graphes sont représentés graphiquement sous forme de diagrammes, dans lesquels, les sommets sont représentés sous forme de points et les arêtes sous forme de lignes allant d'un sommet à un autre. Cette représentation est illustrée dans la figure 3.1.

Un graphe $G = (V, E)$ peut être orienté ou non-orienté selon le type de communication.

Dans un graphe non-orienté, les arêtes n'ont pas de directions spécifiques $e_{ij} \in E \Leftrightarrow e_{ji} \in E$. Si deux sommets i et j sont connectés, le lien les reliant appartient à l'ensemble E et i et j sont appelés voisins. L'ensemble des voisins du sommet i est noté par N_i et son degré est noté par $d_i = |N_i|$ où $|.|$ signifie la cardinalité.

Un graphe est dit orienté si une direction est choisie pour chaque arête, ainsi les relations deviennent asymétriques. Un sommet i est relié à j par une arête orientée, le sommet j est alors voisin de i. L'arête e_{ij} est donc une arête sortante pour i et entrante pour j (voir la figure 3.1). Dans le cas orienté, une arête est aussi appelée un arc.

Le degré sortant d_i^{out} d'un sommet i est le nombre de sommets auxquels il est connecté (son nombre d'arêtes sortantes). Le degré entrant d_i^{in} est le nombre de sommets qui lui sont connectés (son nombre d'arêtes entrantes).

Un chemin d'un sommet i vers un sommet j est une séquence de sommets distincts de telle sorte que les sommets successifs sont voisins. Dans un graphe non-orienté G, deux sommets i et j sont connectés s'il existe un chemin de i vers j. Dans un graphe orienté G, un chemin d'origine i et d'extrémité j est défini par une suite finie d'arcs consécutifs, reliant i à j.

Un graphe non-orienté G est dit connexe, si pour tout couple de sommets dans G, il existe un chemin les reliant. Un graphe orienté est fortement connexe si entre chaque paire de sommets distincts (i, j) dans G, il existe un chemin orienté partant de i et arrivant à j.

La relation indiquant que deux sommets peuvent être reliés par un chemin est une relation

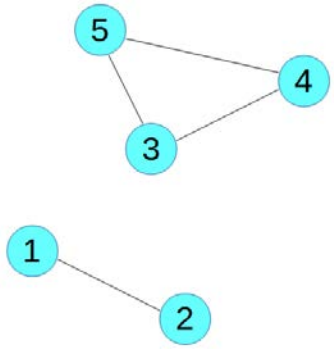


FIGURE 3.2 – Graphe non connexe.

d'équivalence dont les classes d'équivalence sont les composantes connexes. Autrement dit, un graphe peut être décomposé en une ou plusieurs composantes connexes. Les sommets d'une même composante sont joignables par un chemin au contraire des sommets dans des composantes différentes.

La vérification de la connexité est un des premiers problèmes rencontrés dans la théorie des graphes. L'algorithme de recherche des composantes connexes dans un graphe G est basé sur l'algorithme de parcours en profondeur de Trémaux (1882) et Tarjan (1972), noté DFS (Depth First Search) [Tar72]. Le principe de DFS est simple, les sommets sont marqués au fur et à mesure du numéro de la composante à laquelle ils appartiennent ; on part d'un sommet i, on avance dans le graphe le plus loin possible de i, sans former de cycle, on remonte jusqu'au dernier embranchement laissé de côté, ceci jusqu'à revenir au sommet de départ i après avoir visité toutes les arêtes issues de i. Ainsi, l'ensemble des sommets visités forme la première composante connexe du sommet de départ i. Si tous les sommets du graphe ont été visités, alors celui-ci est connexe, sinon on recommence l'exploration à partir d'un sommet non visité, qui nous mène à une deuxième composante connexe à construire et ainsi de suite.

La distance d_G sur les sommets d'un graphe associe à un couple de sommets la longueur du plus court chemin allant du premier au second sommet. Par exemple, dans un réseau de communication, la distance correspond au nombre de sauts minimum entre émetteur et récepteur. Le diamètre $D(G)$ d'un graphe G est défini comme la plus grande distance entre deux de ses sommets.

3.2 Représentations algébriques des graphes

Lorsque qu'un graphe possède un petit nombre de sommets, il est facile de le visualiser et de l'étudier directement. Cependant, pour des graphes représentant des réseaux à grand échelle, leur étude devient beaucoup plus difficile. C'est la raison pour laquelle, leur manipulation par des outils d'algèbre linéaire devient extrêmement utile pour automatiser les calculs.

Un autre point intéressant de l'utilisation de l'algèbre linéaire est la notion de spectre : pour chaque matrice carrée, un ensemble de valeurs propres peut être associées à des vecteurs propres. La signification physique d'un espace « propre » est mieux comprise en considérant la matrice sous la forme d'une transformation géométrique de « points » dans un espace. Ces points définissent un vecteur. La transformation (rotation, translation,...) du vecteur est à nouveau un vecteur dans le même espace, mais généralement différent du vecteur d'origine. Le vecteur qui, après la transformation se révèle être proportionnelle avec lui-même est appelé un vecteur propre et la force de proportionnalité est la valeur propre. Le terme « propre » signifie quelque chose qui est inhérent à lui-même. Ainsi, sachant que chaque graphe est représenté par une matrice, il est naturel d'étudier son spectre.

Dans un contexte plus large, les transformations se sont avérées très utiles dans la science comme par exemple la transformée de Fourrier ou la transformée de Laplace. De nombreuses branches de la science telles que les mathématiques, la physique et l'ingénierie montrent avec des exemples l'intérêt de la transformée de Fourier. Le principe général de ces transformations consiste à utiliser deux domaines différents, en exploitant leur correspondance. Par exemple, un signal est une fonction continue dans le temps qui peut représenter un message ou une information produite au fil du temps. Certaines propriétés du signal sont étudiées de façon plus appropriée dans le domaine du temps, tandis que d'autres sont étudiées dans le domaine fréquentiel. Cette analogie nous a motivé à étudier certaines propriétés d'un graphe dans le domaine topologique (représentation algébrique des graphes), tandis que d'autres propriétés peuvent être plus facilement traitées dans le domaine spectral, telles que les valeurs propres et les vecteurs propres.

Dans cette section, nous allons définir les différentes matrices qui permettent de décrire un graphe matriciellement. Ensuite, nous allons donner une interprétation physique de la matrice Laplacienne en expliquant brièvement l'impact de ses valeurs propres sur la connectivité du graphe.

3.2.1 Matrice d'adjacence et matrice des degrés

La structure d'un graphe avec N sommets est décrite à partir de la matrice d'adjacence $\mathbf{A}$ de taille $N \times N$ avec

$$a_{ij} = \begin{cases} 1, & \text{si } (i,j) \in E \\ 0, & \text{sinon} \end{cases} \tag{3.1}$$

L'élément a_{ij} est égal à un si le sommet j est un voisin du sommet i. Les éléments diagonaux de la matrice $\mathbf{A}$ sont tous égaux à zéro. Si G est un graphe non orienté, $a_{ij} = a_{ji}$, ce qui signifie que $\mathbf{A}$ est symétrique. Dans le cas d'un graphe orienté, $\mathbf{A}$ est asymétrique. Un exemple d'une matrice d'adjacence d'un graphe non orienté est représenté dans la figure 3.3 :

Le degré entrant et sortant d'un sommet i est respectivement la somme du poids des arêtes entrantes et sortantes : $d_i^{in} = \sum_{j=1}^{N} a_{ij}$ et $d_i^{out} = \sum_{j=1}^{N} a_{ji}$, c'est-à-dire, la somme des

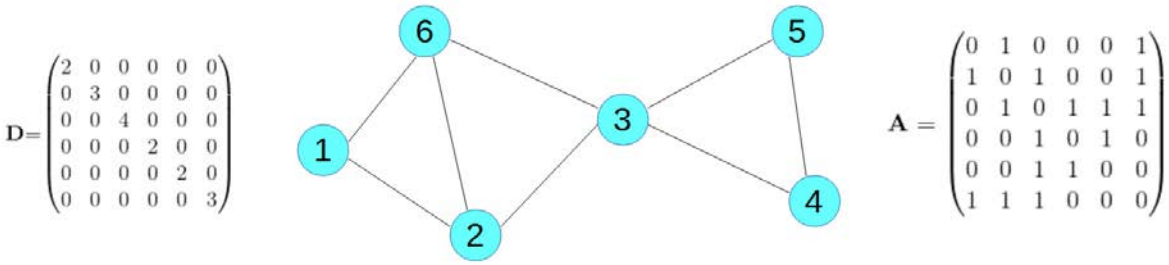


FIGURE 3.3 – Graphe non-orienté avec $N = 6$ sommets : sa matrice d'adjacence et sa matrice des degrés.

lignes et la somme des colonnes de la matrice d'adjacence $\mathbf{A}$.

On dit qu'un graphe est pondéré si on affecte à chaque arête un nombre positif. Ce nombre positif est alors appelé poids de l'arête. Le poids d'un chemin est la somme des poids des arêtes qui la compose. Le plus court chemin entre deux sommets est le chemin de poids minimal entre ces deux sommets. Cependant, il ne faut pas confondre les notions de plus court chemin et le chemin de longueur minimale.

La matrice des degrés $\mathbf{D}$ d'un graphe G est la matrice diagonale $N \times N$ dont les éléments diagonaux sont égaux au degré de chaque sommet.

3.2.2 Matrice d'incidence

La matrice d'incidence d'un graphe G est une matrice qui décrit le graphe en indiquant quels liens arrivent sur quels sommets. La matrice d'incidence est une matrice $N \times P$, où N est le nombre de sommets et P est le nombre d'arêtes. Si le graphe est non-orienté (voir la figure 3.4), le coefficient de la matrice d'incidence en ligne i et en colonne j vaut 1 si le sommet i est une extrémité de l'arête j.

La figure 3.4 représente un graphe avec dessous la matrice d'incidence associée. Pour en faciliter la lecture, les arêtes sont aussi étiquetées.

La lecture par ligne de la matrice $\mathbf{B}$ fournit les incidences entre les sommets et les arêtes : la première ligne indique que le sommet 1 est relié aux deux arêtes 1 et 2.

Les colonnes indiquent plutôt quelles sont les extrémités des arêtes. Par exemple, la dernière colonne indique que l'arête 8 relie les sommets 3 et 5. Cette interprétation rend évident qu'il y a toujours deux 1 par colonne. Le lien entre les matrices d'adjacence, des degrés et d'incidence est :

$$\mathbf{B}\mathbf{B}^T = \mathbf{A} + \mathbf{D} \tag{3.2}$$

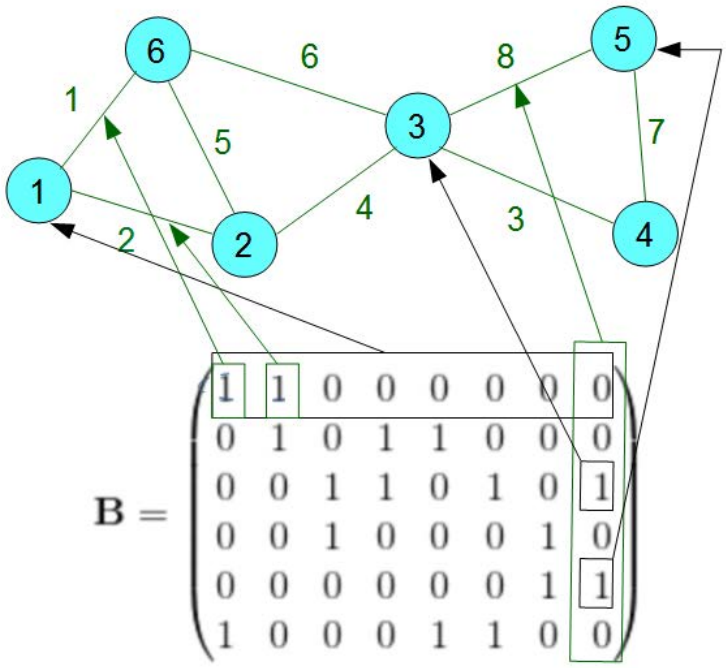


FIGURE 3.4 – Graphe non-orienté avec $N = 6$ sommets et sa matrice d'incidence.

Il existe aussi une seconde matrice d'incidence $\mathbf{M}$ qui repose sur une certaine orientation des arêtes qui peut être choisie de manière totalement arbitraire. Dans ce cas, la colonne correspondant à une arête sera composée d'un 1 et d'un -1 pour indiquer la source et la destination de l'arête orientée. Cette matrice sera réexaminée au chapitre 5 sous la forme de l'opérateur de bord et y sera étendue à des espaces plus généraux que les graphes.

La matrice $\mathbf{M}$ permet aussi de calculer un équivalent discret d'un gradient sur les graphe. En effet, en notant $\mathbf{x}$ le vecteur contenant les valeurs $\mathbf{x}_i$ associées aux sommets i, alors $\mathbf{M}^T\mathbf{x}$ est un vecteur dont le k-ème coefficient est de la forme $\mathbf{x}_i - \mathbf{x}_j$ avec i et j les sommets aux extrémités de l'arête k. Ainsi $\mathbf{M}^T\,\mathbf{x}$ contient toutes les différences des valeurs des sommets voisins.

3.2.3 Laplacien

Nous nous intéresserons au prochain chapitre aux algorithmes de consensus de la moyenne, qui modifient la valeur associée à chaque sommet d'un graphe afin de la rendre la plus proche possible des valeurs des sommets voisins. Ainsi l'écart entre la valeur d'un sommet et celles des voisins sera de plus en plus faible. Une mesure de ces écarts se révèle donc indispensable pour l'étude des algorithmes de consensus.

3.2.3.1 Définition

La matrice d'incidence $\mathbf{M}$ permet déjà de calculer les écarts de valeurs entre deux sommets voisins par l'équivalent d'un gradient. La norme de ce gradient, qui est une forme quadratique, permet donc de mesurer globalement les écarts entre les valeurs voisines,

$$q(\mathbf{x}) = \left\|\mathbf{M}^T\,\mathbf{x}\right\|^2 = \sum_{(i,j)\in E}\Big(\mathbf{x}_i - \mathbf{x}_j\Big)^2, \tag{3.3}$$

où $\mathbf{x}$ est le vecteur contenant les valeurs associées à tous les sommets. Notons que par définition, $q(\mathbf{x})$ est une forme quadratique positive.

La nullité de cette mesure indique que les voisins partagent la même valeur, donc dans notre cas, la convergence des algorithmes de consensus.

Le Laplacien $\mathbf{L}$ correspond à la matrice associée à la forme quadratique $q(\mathbf{x})$:

$$q(\mathbf{x}) = \mathbf{x}^T\,\mathbf{L}\,\mathbf{x}, \tag{3.4}$$

et par conséquent, le lien entre la matrice d'incidence orientée et le Laplacien est

$$\mathbf{L} = \mathbf{M}\,\mathbf{M}^T. \tag{3.5}$$

Il est possible de considérer le produit scalaire $\langle \mathbf{x}, \mathbf{y}\rangle$ associé à la forme quadratique $q(\mathbf{x})$ et donc au Laplacien

$$\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^T\,\mathbf{L}\,\mathbf{y}. \tag{3.6}$$

Le Laplacien ne dépend pas de l'orientation choisie pour construire la matrice d'incidence orientée $\mathbf{M}$ car les signes sont perdus dans le carré de (3.3). Un exemple d'une matrice Laplacienne d'un graphe G est donné dans la figure 3.5. Il est donc tout à fait naturelle que le Laplacien s'écrive aussi en fonction de la matrice d'adjacence $\mathbf{A}$. Un simple calcul mène au résultat

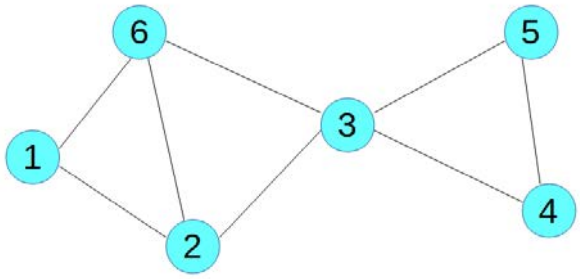
$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{3.7}$$

D'autres opérateurs de Laplace peuvent être définis en utilisant une norme pondérée à la place de (3.3). Toutes les constructions ci-dessus restent encore envisageables et cela permet de moduler les poids associés aux arêtes par exemple selon leur longueur ou encore de moduler les poids associés aux sommets par exemple selon leur degré.

Nous verrons au chapitre 5 une autre extension du Laplacien à d'autres dimensions dans des espaces combinatoires plus génériques que les graphes. Les idées sous-jacentes resteront cependant identiques.

3.2.3.2 Spectre du Laplacien

Comme la matrice $\mathbf{L}$ est positive, les valeurs propres sont toutes positives ou nulles. Le spectre de $\mathbf{L}$ est donc un ensemble de N valeurs positives ou nulles $0 \leq \lambda_1 \leq \cdots \leq \lambda_N$ comptées avec leur multiplicité.



$$\mathbf{L}=\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 3 & -1 & 0 & 0 & -1 \\ 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 2 & 0 \\ -1 & -1 & -1 & 0 & 0 & 3 \end{pmatrix}$$

FIGURE 3.5 – Graphe non-orienté avec $N = 6$ sommets et sa matrice Laplacienne.

Le Laplacien code toute l'information sur la connectivité du graphe dans sa matrice. Sa décomposition en éléments propres code ces mêmes informations mais sous une forme parfois plus directement accessible.

Par exemple, les vecteurs $\mathbf{x}$ dont les coefficients associés aux sommets de chaque composante connexe sont identiques vérifient $\mathbf{L}\,\mathbf{x} = 0$ comme la définition (3.3) le montre. Ainsi la dimension de l'espace propre associé à la valeur propre 0 donne directement le nombre de composantes connexes. Cet espace propre est aussi le noyau $\ker \mathbf{L}$. Bien sûr, comme un vecteur constant est dans $\ker \mathbf{L}$, il y a toujours au moins une composante connexe dans un graphe, et par conséquent la première valeur propre est 0 et sa multiplicité donne la dimension du noyau. Par exemple, la matrice du graphe de la figure 3.2 est

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

et les vecteurs propres associés à la valeur propre 0 sont $\mathbf{x}_1 \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ et $\mathbf{x}_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ de telle sorte que $\mathbf{L}\mathbf{x}_1 = 0$ et $\mathbf{L}\mathbf{x}_2 = 0$.

Les vecteurs propres associés aux autres valeurs propres permettent aussi de faire une segmentation du graphe en communautés [SM00] car les vecteurs propres associés aux faibles valeurs propres indiquent les parties du graphes faiblement connectées. Comme nous venons de le voir, à la limite, la déconnection complète de ces parties est donnée par la valeur la plus faible, 0.

La vitesse de convergence des algorithmes distribués de consensus traitée dans le chapitre 4 dépend fortement des valeurs propres de la matrice Laplacienne $\mathbf{L}$ [GR01]. En effet, notre étude sur le consensus de moyenne est basée sur la vitesse de convergence qui fait

intervenir le rayon spectrale d'une matrice de pondération **W** qui est dérivée de la matrice Laplacienne.

Enfin, nous verrons comment le spectre d'une extension du Laplacien nous permettra de repérer les trous de couverture au chapitre 5.

3.3 Quelques topologies des graphes

Dans cette section, nous allons définir quelques exemples de graphes déterministes et aléatoires nécessaires à la compréhension des résultats du chapitre 4.

3.3.1 Topologie déterministe

La figure 3.6 illustre les trois topologies déterministes décrits dans cette section.

Graphe étoile Un graphe étoile est un graphe connexe dont tous les sommets sauf un sont de degré 1. Son diamètre est égal à 2. Ce type de graphe est généralement utilisé dans les réseaux informatiques et plus précisément dans l'informatique distribuée [Gho07] car il permet d'étudier le fonctionnement des algorithmes au niveau local puisqu'il représente simplement un sommet et son voisinage immédiat.

Graphe grille Un graphe grille est un graphe à deux dimensions. Ce type de graphe est un produit cartésien de graphe de deux graphes chemin. Tous les sommets ont le même degré, 4, hormis ceux au bord de la grille. Cette régularité permet d'éviter d'avoir de trop nombreux effets dus à une hétérogénéité des situations dans l'étude des résultats de simulation d'algorithmes.

Graphe complet Le graphe complet est composé de sommets tous reliés directement les uns aux autres par des arêtes, par exemple le graphe complet à 5 sommets est illustré sur la figure 3.6 à droite. Ce graphe, très régulier, permet de modéliser des situations locales d'un réseau sans fils lorsque les nœuds sont suffisamment proches pour pouvoir communiquer ensemble en broadcast.

3.3.2 Topologie aléatoire

Les graphes aléatoires sont souvent utilisés pour représenter géométriquement un réseau de communication aléatoire. Mathématiquement, un graphe aléatoire est un graphe généré par un processus stochastique [Bol01]. On peut fixer l'acte de naissance de la théorie des graphes aléatoires en 1959 avec l'introduction des graphes d'Erdős-Rényi.

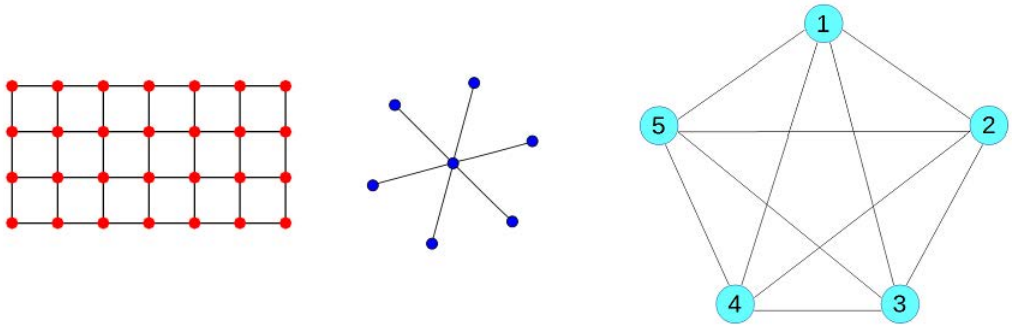


FIGURE 3.6 – Graphe en grille, graphe en étoile et graphe complet.

Cependant, l'auteur [CF99] montre que la dépendance probabiliste entre les liens aléatoires, autrement dit l'égalité de la probabilité de l'existence d'un lien entre chaque sommet, limite l'application du modèle Erdős-Rényi aux réseaux de communications sans fil. En effet, l'utilisation du graphe Erdős-Rényi ne permet pas de modéliser un réseau sans fil (RCSF, mobile, ad-hoc) car d'après [CF99], ce modèle n'introduit aucune corrélation arbitraire entre différents liens. C'est pourquoi nous n'utiliserons pas ce modèle et nous nous concentrerons sur les graphes géométriques aléatoires.

Bien que l'origine des graphes géométriques aléatoires remonte aux travaux de Gilbert en 1961 [Gil61]. Ces graphes ont traditionnellement été associés à des domaines tels que la physique statistique et aux tests d'hypothèses [Pen03], mais ont acquis une nouvelle attention avec les problèmes liés aux réseaux de capteurs sans fils.

Les graphes géométriques aléatoires sont construits en plaçant un ensemble de N sommets dans une région du plan fixée, généralement le carré de largeur unité. La place de chaque sommet est choisie aléatoirement et indépendamment les unes des autres selon un échantillonnage uniforme de la région.

Puis les arêtes sont ajoutées entre tout les sommets ayant une distance inférieure au seuil r de connexion. Intuitivement, ce rayon de connexion représente la portée maximale des communications : deux sommets sont voisins s'ils peuvent communiquer directement. Ainsi la distance dans le graphe entre deux sommets mesure le nombre de sauts à effectuer au minimum pour transmettre un message entre eux.

Dans le cas limite où la taille de la région grandit indéfiniment et où la densité de sommets par unité de surface reste identique, il existe un effet de transition de phase : si le rayon de connexion est inférieur à un seuil, le graphe comporte beaucoup de petites composantes connexes alors qu'au dessus du seuil, il n'y a principalement qu'une seule composante. Ce type de résultat s'obtient par la théorie de la percolation [Hal85] et [Men86]. Dans notre cas, nous nous arrangerons pour que les réseaux soient bien connectés dans nos simulations en choisissant un rayon de connexion suffisant. La figure 3.7 illustre un exemple de graphe géométrique aléatoire avec un rayon de connexion $r = 0.18$.

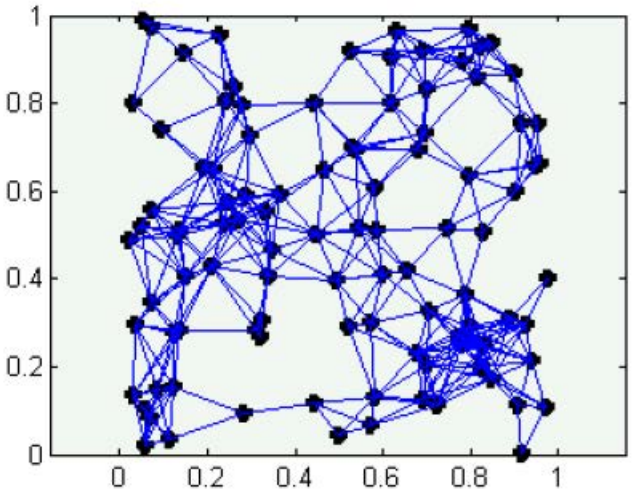


FIGURE 3.7 – Graphe géométrique aléatoire $N = 100$ et $r = 0.18$.

3.4 Conclusion

Dans ce chapitre, nous avons passé en revue les différentes notions de la théorie des graphes, et notamment, celles de la théorie spectrale qui jouent un rôle important pour l'étude de la convergence des algorithmes de consensus. Les informations les plus pertinentes que nous avons pu déduire du spectre Laplacien est par exemple le nombre de composantes connexes ou la connexité du graphe.

Nous avons aussi décrit les topologies des graphes, aussi bien déterministes qu'aléatoires, qui seront utilisées en exemple dans les futurs simulations.

Chapitre 4

Consensus de moyenne

Dans ce chapitre, nous traitons les différents algorithmes de consensus de moyenne, nous commençons tout d'abord par les cas temporels synchrones et ensuite nous expliquons en détail les algorithmes de bavardage dans les cas asynchrones.

Le noyau de notre contribution est basé sur un algorithme de bavardage appelé Push-Sum introduit par Kempe et al. [KDG03] dans lequel chaque capteur maintient deux variables, la première initialisée avec sa valeur de mesure et la deuxième à 1. Les deux variables sont alors mises à jour simultanément à chaque itération et l'estimée de la moyenne par le capteur est le quotient de ces deux variables.

Plus précisément, notre objectif dans ce travail est basé sur l'exploitation des résultats de [KDG03] pour les adapter au modèle temporel asynchrone en utilisant les caractéristiques du graphe et les principaux avantages du bavardage par diffusion [AYSS09]. Ensuite, nous avons optimisé un facteur β de cet algorithme qui permettra d'assurer une convergence rapide.

Malheureusement une recherche bibliographique, nous a permis de trouver des travaux récents similaires à nos résultats [ICHJ12]. Comme la preuve théorique de la convergence est déjà fournie, nous nous sommes concentrés sur une analyse statistique de notre facteur de convergence β en utilisant une autre méthode pour expliquer l'algorithme Push-Sum par diffusion.

4.1 Introduction

Comme on a vu précédemment dans le chapitre 2, un nœud de capteur est généralement composé d'un émetteur-récepteur radio pour les communications sans fil, une unité de traitement de faible complexité, une unité sensorielle en charge de la détection des paramètres physiques et une alimentation souvent une batterie. Les nœuds de capteurs collectent des mesures à partir de l'environnement et, éventuellement, font un traitement simple des données. Ces données peuvent être transférées vers une unité centrale d'un réseau centralisé, ou peuvent être traitées localement dans un réseau décentralisé.

Dans des déploiements centralisés, les nœuds de capteurs transmettent leurs mesures à une unité centrale en charge de la collecte des données du réseau et du calcul final. Les réseaux centralisés nécessitent une bonne organisation des nœuds de capteurs, ainsi que des protocoles de routage pour transmettre les données à l'unité centrale. Dans des situations d'urgence, le flux d'informations destiné à l'unité centrale peut devenir élevé. De plus, la configuration matérielle requise pour les communications sans fil peut également conduire à une augmentation du coût des dispositifs, et, par conséquent, à un coût global élevé du réseau, en particulier lorsque le nombre de nœuds de capteurs devient important. Pour ces raisons, un RCSF centralisé peut être très inefficace et coûteux en termes de consommation d'énergie.

Afin d'éviter les problèmes liés à l'architecture centralisée, la communauté scientifique

FIGURE 4.1 – Exemples d'algorithmes distribués dans la nature : comportement social des animaux.

utilise de plus en plus l'architecture décentralisée où tous les nœuds de capteurs ont les mêmes capacités et sont capables d'effectuer les mêmes tâches. En effet, le principe des systèmes décentralisés est que les nœuds de capteurs s'organisent localement et effectuent des calculs sans la nécessité de transmettre les informations à l'unité centrale. Chaque nœud de capteur communique avec ses voisins pour échanger leurs informations et prendre des décisions.

Dans les architectures décentralisées, les calculs sont effectués de manière distribuée, selon des algorithmes dits distribués. En outre, les éléments du réseau doivent interagir les uns avec les autres pour atteindre un objectif commun [Gho07] et [CDKB11]. Les exemples de telles organisations s'observent dans la nature ; figure 4.1.

De ce point de vue, un RCSF peut donc être considéré comme la mise en œuvre distribuée d'un calcul de fonctions paramétrées par les données collectées par les nœuds de capteurs.

4.2 Algorithmes distribués de consensus

4.2.1 Historique

Les algorithmes distribués de consensus sont des algorithmes itératifs de faible complexité où les nœuds de capteurs voisins interagissent les uns avec les autres pour parvenir à un accord commun au sujet d'une fonction des mesures, sans faire appel à une unité centrale.

Ainsi, l'objectif de ces algorithmes de consensus est de réduire la consommation d'énergie pour garantir une durée de vie plus longue pour l'ensemble du réseau.

Les algorithmes de consensus permettent aussi une mise en œuvre facile. Les capteurs ne nécessitent aucune connaissance sur la topologie globale du réseau. Ainsi, les échanges d'informations entre voisins se font de manière simple, sans l'utilisation de protocoles de routage. Un autre avantage des algorithmes de consensus concerne leur évolutivité. Les capteurs disposent du même taux de communication même avec le passage à l'échelle.

Connu sous le nom de problème de l'accord, le consensus a été initialement étudié par Tsitsiklis [Tsi84] et a reçu une attention particulière dans différents domaines en raison de sa

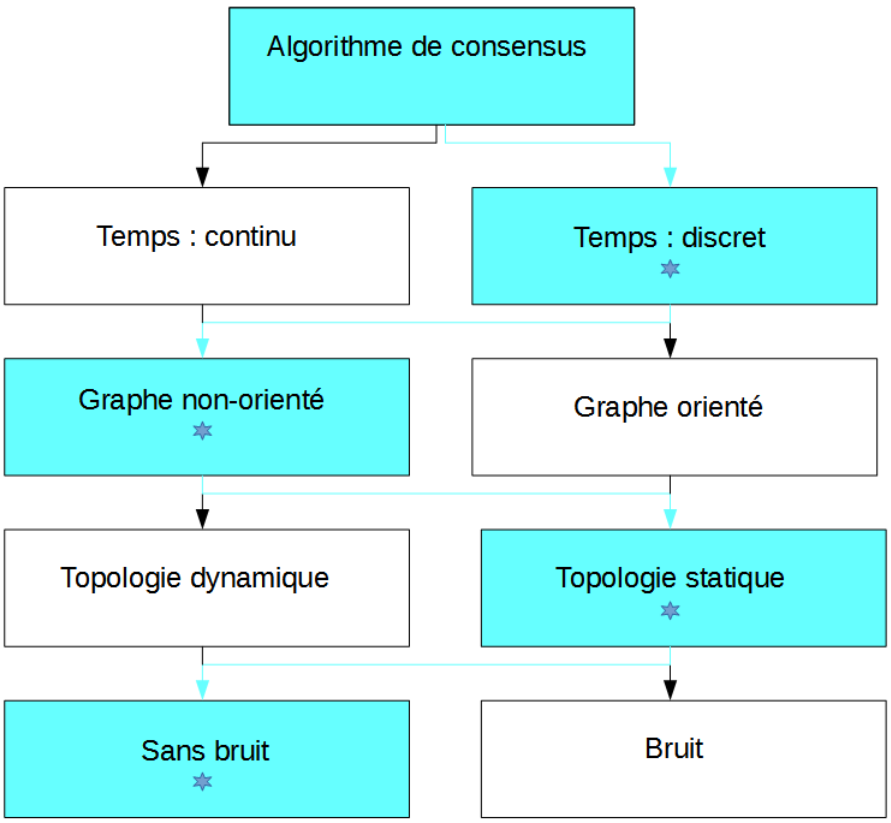


FIGURE 4.2 – Différents types d'algorithmes de consensus.

large gamme d'applications telles que le contrôle distribué [OSM04], [XB04] et [MdW09], le problème de fusion de données [OSS05], [SOSM05] et [XBL05], la coordination entre agents autonomes [LFM05] et [Mor05], le contrôle de couverture [GBCJ06] ou l'équilibrage de charge dans le calcul parallèle [DMN05]. Le consensus pourrait également être considéré comme une forme d'auto-synchronisation des oscillateurs couplés [BC05].

Le principe du consensus peut aussi être utilisé dans le domaine de la surveillance environnementale pour calculer par exemple la moyenne de température d'un environnement [KH10] à partir des informations collectées par les nœuds de capteurs et leurs voisins. L'algorithme de consensus de moyenne [OSM04] est un algorithme itératif où tous les capteurs communiquent localement dans l'objectif de calculer la moyenne des valeurs initiales et ceci en utilisant les propres mesures des capteurs, ainsi que les mesures de leurs voisins.

Dans le cadre de nos travaux, les échanges entre nœuds de capteurs seront matérialisés par un ensemble de mesures ou données discrètes. Nous en détaillerons plus dans la suite de ce chapitre. La figure 4.2 montre une classification des différents algorithmes de consensus. Le travail s'est basé sur des graphes statiques non orientés où les communications sont bidirectionnelles entre capteurs dans un canal sans bruit.

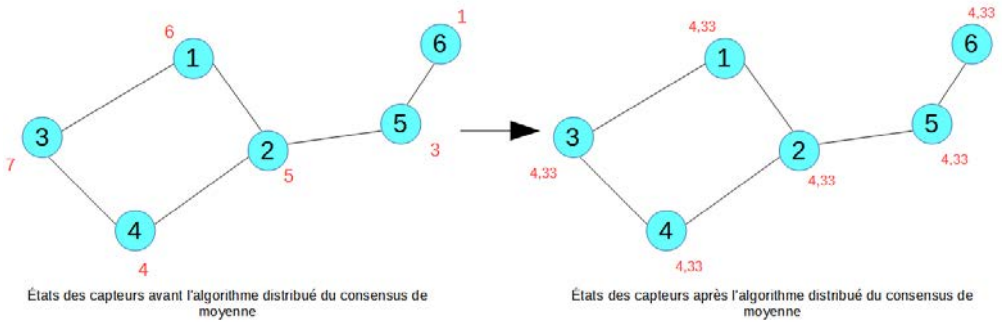


FIGURE 4.3 – L'algorithme distribué de consensus de moyenne.

4.2.2 Consensus de moyenne

Pour modéliser le réseau, nous considérons un graphe non-orienté $G = (V, E)$. Par exemple, considérons le réseau de 6 capteurs qui communiquent entre eux illustré dans la figure 4.3. Chaque capteur détient une valeur initiale. L'algorithme distribué de consensus est une règle d'interaction qui spécifie l'échange d'informations entre un capteur et ses voisins pour parvenir à un accord commun au sujet d'une fonction des mesures. Dans cet exemple, il s'agit de la moyenne des mesures. Dans la figure 4.3, les capteurs convergent vers une valeur moyenne commune qui est la moyenne des valeurs initiales.

4.2.3 Consensus des croyances

Les tests d'hypothèses forment une autre classe d'applications du consensus de moyenne. Le consensus des croyances introduit par Olfati-Saber et al. [OSFFS06] est un algorithme de test d'hypothèse dans un cadre distribué, où plusieurs capteurs tentent de se mettre d'accord sur la classification la plus probable d'un évènement. En particulier l'algorithme est dérivé de la règle de Bayes donnée par

$$\Pr[h|z_1, z_2, ..., z_N] = \frac{\Pr[h] \Pr[z_1, z_2, ..., z_N|h]}{\Pr[z_1, z_2, ..., z_N]} \tag{4.1}$$

$$= \frac{\Pr[h] \prod_{i=1}^{N} \Pr[z_i|h]}{\Pr[z_1, z_2, ..., z_N]} \tag{4.2}$$

où h est une réalisation de l'espace des hypothèses H, et $Z = \{z_1, z_2, ..., z_N\}$ est l'ensemble des mesures des N capteurs supposées indépendantes. Chaque élément de la règle de Bayes (4.2) joue un rôle différent :

Probabilité a priori $\Pr[h]$ Elle modélise l'expertise quand la prise de connaissance précède les données.

Fonction de vraisemblance $\Pr[z_i|h]$ Elle mesure l'adéquation entre les données et l'hypothèse.

Probabilité a posteriori $\Pr[h|Z]$ Elle correspond à la nouvelle hypothèse en prenant en compte l'expérience acquise sous forme des données z_i.

Le dénominateur $\Pr[Z]$ est indépendant, et peut donc être considéré comme une constante pour normaliser les probabilités résultantes. Il est supposé que chaque capteur possède une croyance indépendante sur la probabilité de h qui doit être partagée avec d'autres capteurs en fonction de l'équation (4.2). Les auteurs de [OSFFS06] ont défini la fonction de vraisemblance d'un capteur i dans le réseau par

$$\log\left(\prod_{i=1}^{N}\Pr[z_1, z_2, ..., z_N|h]\right) = \sum_{i=1}^{N}\log\left(\Pr[z_1, z_2, ..., z_N|h]\right) \tag{4.3}$$

$$= \sum_{i=1}^{N}\log\left(\Pr[z_i|h]\right) \tag{4.4}$$

Ainsi l'algorithme de consensus de moyenne doit être en mesure de calculer la moyenne des fonctions de vraisemblance de tous les capteurs i dans le réseau.

4.3 Modèles temporels : synchrone et asynchrone

On distingue deux catégories d'algorithmes distribués de consensus de moyenne : les algorithmes synchrones et les algorithmes asynchrones. Dans les deux cas, les deux algorithmes sont itératifs et chaque capteur détient une estimation de la moyenne des mesures. Ces algorithmes sont conçus de telle sorte que toutes les estimations du réseau convergent vers le consensus qui est la moyenne dans notre cas d'étude.

Dans un algorithme synchrone, tous les capteurs du RCSF se réveillent à chaque itération et diffusent leurs valeurs d'états. Ainsi, tous les capteurs mettent à jour leurs valeurs d'états avant que la prochaine itération ne commence. Cependant, le modèle temporel synchrone nécessite la synchronisation du réseau, qui peut être une contrainte difficile à satisfaire dans certains environnements distribués. Pour cette raison, le modèle synchrone semble inadapté pour l'application de l'algorithme distribué de consensus de moyenne dans la réalité, mais il est important de l'étudier et de le comprendre car il est souvent beaucoup plus simple à analyser que le modèle asynchrone, et qu'il donne souvent une bonne information sur les vitesse de convergence.

Dans un algorithme asynchrone, un seul capteur se réveille à chaque itération et communique avec ses voisins : un seul groupe de capteurs du réseau met à jour ses valeurs d'états à la fin de chaque itération. Dans ce modèle, chaque capteur possède une horloge indépendante où les coups d'horloge sont distribués selon un processus de Poisson de taux λ. Un capteur se réveille à chaque coup d'horloge. Ainsi, les capteurs se réveillent à des instants aléatoires dans un ordre aléatoire. Le temps est mesuré en termes du nombre de

coups d'une seule horloge virtuelle globale selon un processus de Poisson de taux $N\lambda$, où N est le nombre de capteurs dans le réseau. Si λ est trop grand, alors il y a de fortes chances qu'un capteur se réveille alors que certaines itérations sont en cours d'exécution. Si plusieurs itérations sont réalisées en même temps dans le réseau, alors il y a un risque d'interférences et les capteurs peuvent ne pas se mettre à jour. Dans les RCSF, la gestion de l'énergie est primordiale et il est préférable de ne pas gaspiller l'énergie dans des communications inutiles. Ainsi, si λ est assez petit, alors il n' y a qu'une seule communication à la fois avec une forte probabilité, et chaque communication a de plus grande chance de réussir.

Tous les algorithmes asynchrones étudiés dans ces travaux considèrent qu'il y a toujours une seule itération à la fois dans le RCSF. De plus, nous comptons le temps en termes de nombre d'itérations. En effet, le temps entre deux coups d'horloge consécutifs est considéré comme un intervalle de temps et à chaque intervalle de temps, une itération se produit.

4.4 Consensus de moyenne synchrone

4.4.1 Modélisation mathématique de l'algorithme

Chacun des N capteurs possède une valeur définie en tant qu'état du capteur i. Soit $\mathbf{x}(0) = [x_1(0), x_2(0), ..., x_N(0)]^T$ le vecteur des états initiaux du RCSF, l'objectif principal du système est de calculer la valeur de la moyenne x_{moy} de manière distribuée. Chaque itération implique une communication locale entre les capteurs, en mettant à jour sa valeur comme une combinaison linéaire de sa propre valeur et celles de ses voisins. Mathématiquement, l'algorithme distribué de consensus de moyenne peut être formulé par l'équation suivante :

$$x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), \qquad i = 1, 2, ..., N \tag{4.5}$$

ou sous forme matricielle :

$$\mathbf{x}(k+1) = \mathbf{W}(k)\mathbf{x}(k), \tag{4.6}$$

où $\mathbf{x}(k) = [x_1(k), x_2(k), ..., x_N(k)]^T$ est le vecteur de tous les états des capteurs à l'itération k et la matrice $\mathbf{W}(k)$ est la matrice décrivant l'algorithme en prenant en compte le réseau. Il faut en effet que $w_{ij}(k) = 0$ si i et j ne sont pas voisins.

Les propriétés de la matrice $\mathbf{W}(k)$ influencent la convergence de l'algorithme vers le consensus. L'objectif est de concevoir une politique de pondération telle que $\lim_{k\to\infty} \mathbf{x}(k) = \bar{\mathbf{x}}$ où $\bar{\mathbf{x}} = x_{moy}\mathbf{1}$ avec $\mathbf{1}$ le vecteur dont les entrées valent 1. Ce qui signifie que tous les capteurs se sont mis d'accord sur la valeur x_{moy}. L'algorithme distribué synchrone de consensus de moyenne peut être programmé suivant l'algorithme 4.1.

Algorithm 4.1: Algorithme distribué synchrone de consensus de moyenne

Input : Graphe représentant le réseau $G = (V, E)$
Input : Valeurs initiales $\mathbf{x}(0)$
`/* Les méthodes de construction de W(k) sont démontrées dans la sous-section 4.4.4 */`
Data: Construction de la matrice de pondération $\mathbf{W}(k)$

1 **while** *la convergence n'est pas atteinte* **do**
2 Chaque capteur i transmet sa valeur $x_i(k)$ à ses voisins $j \in N_i$ et reçoit les valeurs $x_j(k)$ de son voisinage j
3 Chaque capteur i met à jour sa valeur par une combinaison linéaire de sa propre valeur $x_i(k)$ et celles de ses voisins $x_j(k)$ à l'itération k : $x_i(k+1) \leftarrow w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k)$
4 **end**
Output: $\mathbf{x}(\infty)$

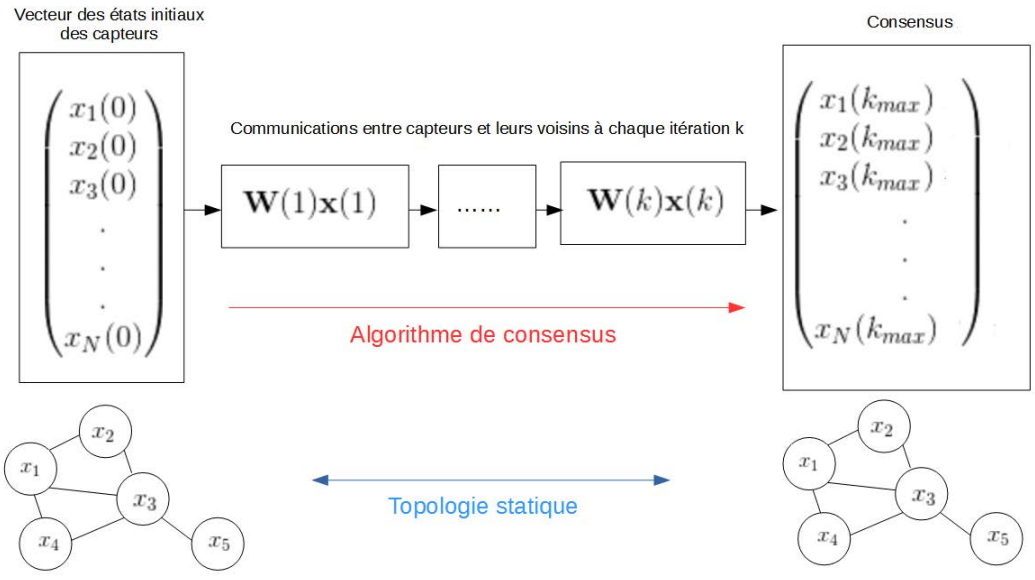


FIGURE 4.4 – Synopsis de l'algorithme distribué synchrone de consensus de moyenne.

4.4.2 Conditions de convergence de l'algorithme distribué de consensus de moyenne synchrone

Si $\mathbf{W}(k)$ ne dépend pas de l'itération k, l'équation linéaire (4.6) implique que $\mathbf{x}(k) = \mathbf{W}^k\mathbf{x}(0)$. Nous choisissons la matrice de pondération $\mathbf{W}$ de telle sorte que pour tout vecteur initial $\mathbf{x}(0)$, $\mathbf{x}(k)$ converge vers le vecteur de moyenne :

$$\bar{\mathbf{x}} = \left(\underbrace{\mathbf{1}^T\mathbf{x}(0)/N}_{x_{moy}\ :\ \text{Moyenne}} \right) \mathbf{1} = \left(\mathbf{1}\mathbf{1}^T/N \right) \mathbf{x}(0) \tag{4.7}$$

Ce qui signifie que :

$$\lim_{k\to\infty} \mathbf{x}(k) = \lim_{k\to\infty} \mathbf{W}^k\mathbf{x}(0) = \frac{\mathbf{1}\mathbf{1}^T}{N}\mathbf{x}(0), \tag{4.8}$$

L'équation (4.8) peut s'écrire comme suit :

$$\lim_{k\to\infty} \mathbf{W}^k = \frac{\mathbf{1}\mathbf{1}^T}{N}, \tag{4.9}$$

car $\mathbf{x}(0)$ peut prendre n'importe quelle valeur. Il a été démontré dans [XB04] que la convergence de (4.9) dans le modèle temporel synchrone est obtenue si et seulement si les trois conditions suivantes sont respectées :

$$\mathbf{1}^T\mathbf{W} = \mathbf{1}^T \tag{4.10}$$

$$\mathbf{W}\mathbf{1} = \mathbf{1} \tag{4.11}$$

$$sp\left(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N} \right) < 1 \tag{4.12}$$

avec $sp(.)$ le rayon spectrale de la matrice, c'est-à-dire le plus grand module de valeur propre.

L'équation (4.10) montre que $\mathbf{1}$ est le vecteur propre à gauche de $\mathbf{W}$ de valeur propre un. Cette condition implique que $\mathbf{1}^T\mathbf{x}(k+1) = \mathbf{1}^T\mathbf{x}(k)$ pour tout k. Ainsi la condition (4.10) garantit que la moyenne soit conservée à chaque itération.

L'équation (4.11) montre que $\mathbf{1}$ est le vecteur propre à droite de $\mathbf{W}$ de valeur propre un. Cette condition implique que $\mathbf{1}$ est un point fixe de l'itération linéaire (4.6) associée à $\mathbf{W}$, c'est-à-dire que la moyenne est un consensus.

Ainsi, avec les deux premières conditions (4.10) et (4.11), la condition (4.12) signifie que 1 est une valeur propre simple de $\mathbf{W}$, et que toutes les autres valeurs propres sont de module strictement inférieures à 1. En résumé, la condition nécessaire et suffisante pour assurer la convergence de l'algorithme du consensus de moyenne dans un graphe non-orienté est que la matrice de pondération $\mathbf{W}$ doit être une matrice doublement stochastique (les sommes des éléments de chaque ligne et chaque colonne sont égales à 1). On peut remarquer aussi que les graphes ayant plus d'une composante connexe (voir chapitre 3) ne respectent pas la condition (4.12) puisque le rang de la matrice de pondération est nécessairement plus grand que 1.

4.4.3 Vitesse de convergence de l'algorithme

Nous évaluons les performances de l'algorithme de consensus de moyenne en fonction du temps de convergence. Pour cela, nous utilisons deux métriques : le taux de convergence r_{asymp} et le temps de convergence τ_{asymp} définis dans [XB04]. Le taux de convergence est donné par

$$r_{asymp} = \sup_{\mathbf{x}(0)\neq\bar{\mathbf{x}}} \lim_{k\to\infty} \left(\frac{\|\mathbf{x}(k) - \bar{\mathbf{x}}\|_2}{\|\mathbf{x}(0) - \bar{\mathbf{x}}\|_2} \right)^{\frac{1}{k}}. \tag{4.13}$$

Le taux de convergence r_{asymp} peut être directement dérivé directement de la matrice de pondération $\mathbf{W}$

$$r_{asymp}(\mathbf{W}) = sp\left(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N} \right). \tag{4.14}$$

La convergence est d'autant plus rapide que $r_{asymp}(\mathbf{W})$ est petit. Le temps de convergence a été défini dans [XB04] par

$$\tau_{asym} = \frac{1}{\log(1/r_{asymp})} \tag{4.15}$$

qui correspond au nombre asymptotique d'itérations nécessaires pour que l'erreur décroisse d'un facteur $1/e$, soit environ 37%. Il nous permet de mesurer et de comparer la vitesse de convergence des différents algorithmes.

4.4.4 Méthodes de construction de la matrice de pondération W

Dans la littérature, il y a quelques travaux consacrés aux matrices de pondérations [XB04] et [XBK07] dont la construction satisfait aux conditions de convergence des algorithmes distribués de consensus de moyenne.

4.4.4.1 Degré maximum

Une approche pour la construction de la matrice de pondération $\mathbf{W}$ dans un graphe (avec une topologie fixe) consiste à attribuer un poids à chaque arête. Ce poids est égal à l'inverse du degré maximum du réseau.

$$w_{ij} = \begin{cases} \frac{1}{1+d_{\max}}, & \text{si } j \in N_i \\ 1 - \frac{d_i}{1+d_{\max}}, & \text{si } i = j \\ 0, & \text{sinon} \end{cases} \tag{4.16}$$

où $d_{\max} = \max_{i\in V} d_i \leq N$. Nous remarquons que le degré maximum du réseau peut être obtenu par les nœuds de capteurs en utilisant un algorithme distribué de consensus du maximum.

4.4.4.2 Metropolis hasting

La construction Metropolis hasting pour un graphe à topologie fixe (invariant dans le temps) est proposé dans [XB04],

$$w_{ij} = \begin{cases} \frac{1}{1+\max\{d_i,d_j\}}, & \text{si } j \in N_i \\ 1 - \sum_{j \in N_i} \frac{1}{1+\max\{d_i,d_j\}}, & \text{si } i = j \\ 0, & \text{sinon} \end{cases} \tag{4.17}$$

Dans cette construction, chaque capteur ne nécessite que la connaissance de l'information locale à savoir les degrés de ses voisins, alors que la méthode du degré maximum exige la connaissance du degré maximum dans le réseau. Autrement dit, la connaissance des informations globales du réseau par les capteurs n'est pas nécessaire.

4.4.4.3 Constante optimale

Un modèle largement utilisé pour la construction de la matrice de pondération dans les deux topologies de graphes (fixe et variable dans le temps) consiste à pondérer la différence des états de capteurs voisins avec une constante positive à chaque itération [XB04].

Ainsi, l'équation de mise-à-jour de l'algorithme distribué de consensus de moyenne est donnée par :

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in N_i} (x_j(k) - x_i(k)), \quad i = 1, 2, ..., N \tag{4.18}$$

Les éléments de la matrice de pondération $\mathbf{W}$ sont donnés par :

$$w_{ij} = \begin{cases} \epsilon, & \text{si } j \in N_i \\ 1 - |N_i|\epsilon, & \text{si } i = j \\ 0, & \text{sinon} \end{cases} \tag{4.19}$$

où $|.|$ désigne la cardinalité de l'ensemble des voisins du capteur i. La matrice de pondération $\mathbf{W}$ peut aussi s'écrire sous forme matricielle :

$$\mathbf{W} = \mathbf{I} - \epsilon\mathbf{L}, \tag{4.20}$$

où $\mathbf{L}$ est la matrice Laplacienne dont les propriétés ont été données dans le chapitre 3 et $\mathbf{I}$ est la matrice identité $N \times N$. La constante optimale ϵ doit être choisie de façon à garantir la convergence de l'algorithme.

Nous pouvons exprimer les valeurs propres de la matrice de pondération $\mathbf{W}$ en fonction des valeurs propres de la matrice Laplacienne $\mathbf{L}$ à partir de l'équation (4.20) :

$$\lambda_i(\mathbf{W}) = 1 - \epsilon\lambda_{N-i+1}(\mathbf{L}), \qquad i = 1, ..., N \tag{4.21}$$

où $\lambda_i(.)$ est la i-ème plus grande valeur propre de la matrice de pondération $\mathbf{W}$. Notons que la plus petite valeur propre de la matrice Laplacienne $\mathbf{L}$ est 0 et par conséquent la plus grande valeur propre de $\mathbf{W}$ est 1. Le rayon spectrale de la matrice $\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N}$ est sa plus grande valeur propre en valeur absolue donc on peut l'exprimer comme suit :

$$sp\left(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N}\right) = \max\left(\lambda_2(\mathbf{W}), -\lambda_N(\mathbf{W})\right) = \max\left(1 - \epsilon\lambda_{N-1}(\mathbf{L}), -1 + \epsilon\lambda_1(\mathbf{L})\right). \quad (4.22)$$

Nous pouvons ainsi déterminer l'intervalle de ϵ pour assurer la convergence de l'algorithme. Si $1-\epsilon\lambda_{N-1}(\mathbf{L})$ est le maximum, la condition est assurée pour tout $\epsilon > 0$ puisque dans un graphe connexe $\lambda_{N-1}(\mathbf{L})$ est toujours supérieure à 0 (voir chapitre 3). Ainsi, $sp(\mathbf{W}-\frac{\mathbf{1}\mathbf{1}^T}{N}) < 1$, si et seulement si

$$0 < \epsilon < \frac{2}{\lambda_1(\mathbf{L})}. \quad (4.23)$$

Le choix de ϵ minimisant le taux de convergence r_{asymp} donné par l'équation (4.22) pour optimiser la vitesse de convergence est donné par :

$$\epsilon^* = \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{N-1}(\mathbf{L})}. \quad (4.24)$$

Ainsi, nous obtenons la valeur optimale de la constante des poids des arêtes.

4.4.4.4 Comparaison des trois méthodes de construction de W

Le tableau 4.1 résume les trois méthodes (4.16), (4.17) et (4.19) et les informations nécessaires que les capteurs doivent connaître a priori pour la construction de la matrice de pondération $\mathbf{W}$.

Intuitivement, nous remarquons à travers le tableau 4.1 que la méthode du degré maximum (DM) nécessite un premier consensus pour déterminer le degré maximum dans le graphe et sa performance dépend du diamètre de ce dernier. Elle peut toutefois se satisfaire d'une borne du degré maximum qui est souvent simple à déterminer.

Pour la méthode du Metropolis Hasting (MH), l'algorithme de consensus de moyenne définit sa matrice de pondération $\mathbf{W}$ sur la base de son voisinage, autrement dit, les capteurs doivent connaitre leurs voisins à un saut. Ainsi, cette méthode n'effectue aucun traitement centralisé, car les informations sont localement traitées par les capteurs.

En ce qui conçerne la méthode de la constante optimale (CO), le calcul du poids constant est plus complexe puisqu'il est nécessaire d'avoir des informations sur le spectre du Laplacien du graphe, qu'il est difficile d'obtenir par une méthode distribuée.

Nous retiendrons ainsi, après cette comparaison, qu'une méthode de construction ne tenant pas compte du spectre du Laplacien est plus intéressante car on se retrouve dans un contexte où les capteurs doivent communiquer localement sans passer par des solutions centralisées.

Méthodes de construction	w_{ij}	Informations
CO	$\frac{2}{\lambda_1(\mathbf{L})+\lambda_{N-1}(\mathbf{L})}$	Topologie du graphe
MH	$\frac{1}{1+\max\{d_i,d_j\}}$	Informations sur les voisins
DM	$\frac{1}{1+d_{\max}}$	Degré maximum

TABLE 4.1 – Méthodes de construction de la matrice de pondération $\mathbf{W}$.

L'algorithme de consensus de moyenne présente l'avantage d'être simple car les deux méthodes DM et MH sont faciles à mettre en œuvre et nécessitent peu d'informations sur la topologie du réseau. De plus, l'avantage de ces constructions est que l'algorithme garde son efficacité même dans des situations de passage à l'échelle telles que l'augmentation du nombre de capteurs, ou la défaillance de certains capteurs. En effet, cet algorithme distribué permet aux capteurs de disposer du même taux de communication dont l'objectif est d'éviter par exemple, les problèmes de congestion des transmissions.

La prochaine section présentera en détail les performances de chaque méthode sur des exemples de graphes.

4.4.5 Résultats de simulations

Nous avons mis en place deux axes de simulations. Tout d'abord, nous testons l'algorithme de consensus de moyenne sur des modèles de graphes déterministes tels que le modèle étoile et grille. Ensuite, nous avons considéré le modèle géométrique aléatoire (voir le chapitre 3 pour plus de précisions) pour analyser les performances de l'algorithme dans des conditions plus réalistes.

Pour les deux topologies en étoile et en grille, nous avons choisi un nombre de capteurs égal à 9 et à 100. Les valeurs initiales attribuées aux capteurs sont aléatoires. La figure 4.5 illustre les résultats obtenus.

Intuitivement, on pourrait penser qu'une convergence rapide est possible dans le cas d'une topologie en étoile puisque son diamètre est égal à deux. Le degré du capteur central est élevé, celui-ci a accès à la totalité de l'information, et par conséquent, il agit comme un goulot d'étranglement dans le réseau en diminuant la vitesse de convergence de l'algorithme. Sur la figure 4.5 qui illustre la convergence de l'algorithme sur une topologie en étoile par les deux méthodes DM et MH, nous pouvons remarquer ce phénomène. En effet, le capteur central dont la valeur initiale est égale à 1, se rapproche en une seul itération vers la moyenne globale, cependant, la convergence des autres capteurs du réseau est moins rapide par rapport à celle du capteur central. Nous remarquons aussi sur la figure 4.5 à gauche en utilisant la méthode de la constante optimale CO, un phénomène d'oscillation lors de la convergence du capteur central. Ceci est notamment dû aux choix de la valeur de la constante optimale calculée à partir du spectre du Laplacien, et qui agit directement sur les mesures locales du capteur central en diminuant sa vitesse de convergence. La dynamique (4.6) est très proche d'un régime critique. Donc contrairement aux

	Étoile (N=9)	Grille (N=9)	Étoile (N=100)	Grille (N=100)
DM	7.4	3.4	98.5	40.3
MH	7.4	2.9	98.5	39.4
CO	4.4	2.8	50	39.8

TABLE 4.2 – Temps de convergence avec différentes méthodes DM, MH et CO dans une topologie en étoile et en grille pour $N = 9$ et $N = 100$.

deux méthodes DM et MH, la convergence du capteur central devient moins rapide par la méthode CO. Toutefois, cette dernière offre globalement une meilleure convergence de l'algorithme par rapport aux autres méthodes DM et MH.

La figure 4.5 à droite montre que la topologie en grille présente l'avantage de converger plus rapidement que la topologie en étoile. Ceci s'explique du fait que la répartition des degrés dans ce type de graphe permet d'éviter les problèmes liés aux effets dûs à une disparité des degrés comme c'est le cas de la topologie en étoile où la totalité de l'information se concentre sur le capteur central. En ce qui concerne les méthodes de construction, les deux méthodes CO et MH permettent une convergence rapide de l'algorithme par rapport à DM.

Cette comparaison entre deux types de topologies déterministes, nous a permis de constater qu'une bonne répartition des degrés (grille dans notre cas), permet à l'algorithme de converger beaucoup plus rapidement vers le consensus de moyenne.

À des fins de comparaison numérique, le tableau 4.2 représente les performances des différentes méthodes en fonction du temps de convergence τ_{asym} (4.15). Dans le cas d'une topologie en étoile, la méthode CO offre un meilleur temps de convergence égal à 4.48 pour $N = 9$, alors que les deux méthodes DM et MH présentent un temps de convergence identique égal à 7.49. Même remarque pour $N = 100$, la méthode CO est beaucoup plus performante avec un temps de convergence égal à 50 contre 98.5 pour DM et MH.

Comme nous nous y attendions, la vitesse de convergence dans une topologie en grille s'avère être meilleure par rapport à la topologie en étoile. De plus, le temps de convergence des deux méthodes CO et MH devient quasiment identique pour $N = 9$ et $N = 100$, et offre une meilleure performance par rapport à DM.

Dans la deuxième partie de simulation 4.6, nous évaluons la convergence de l'algorithme de consensus de moyenne dans un graphe géométrique aléatoire $N = 100$ (voir chapitre 3). Nous avons choisi deux rayons de connexion $r = 0.18$ et $r = 0.35$. Les valeurs initiales attribuées aux capteurs sont aléatoires. Le temps de convergence pour les différentes méthodes DM, MH et CO est résumé dans le tableau 4.3.

Avec un $r = 0.18$, nous remarquons que la méthode CO offre une meilleure vitesse de convergence par rapport à DM et MH. Cependant, dans un contexte distribué, la méthode MH serait un choix judicieux pour la construction de la matrice de pondération puisque l'échange d'informations s'effectue localement entre capteurs et la perte de performance reste limitée. La performance de MH est meilleure par rapport à celle de DM en terme de

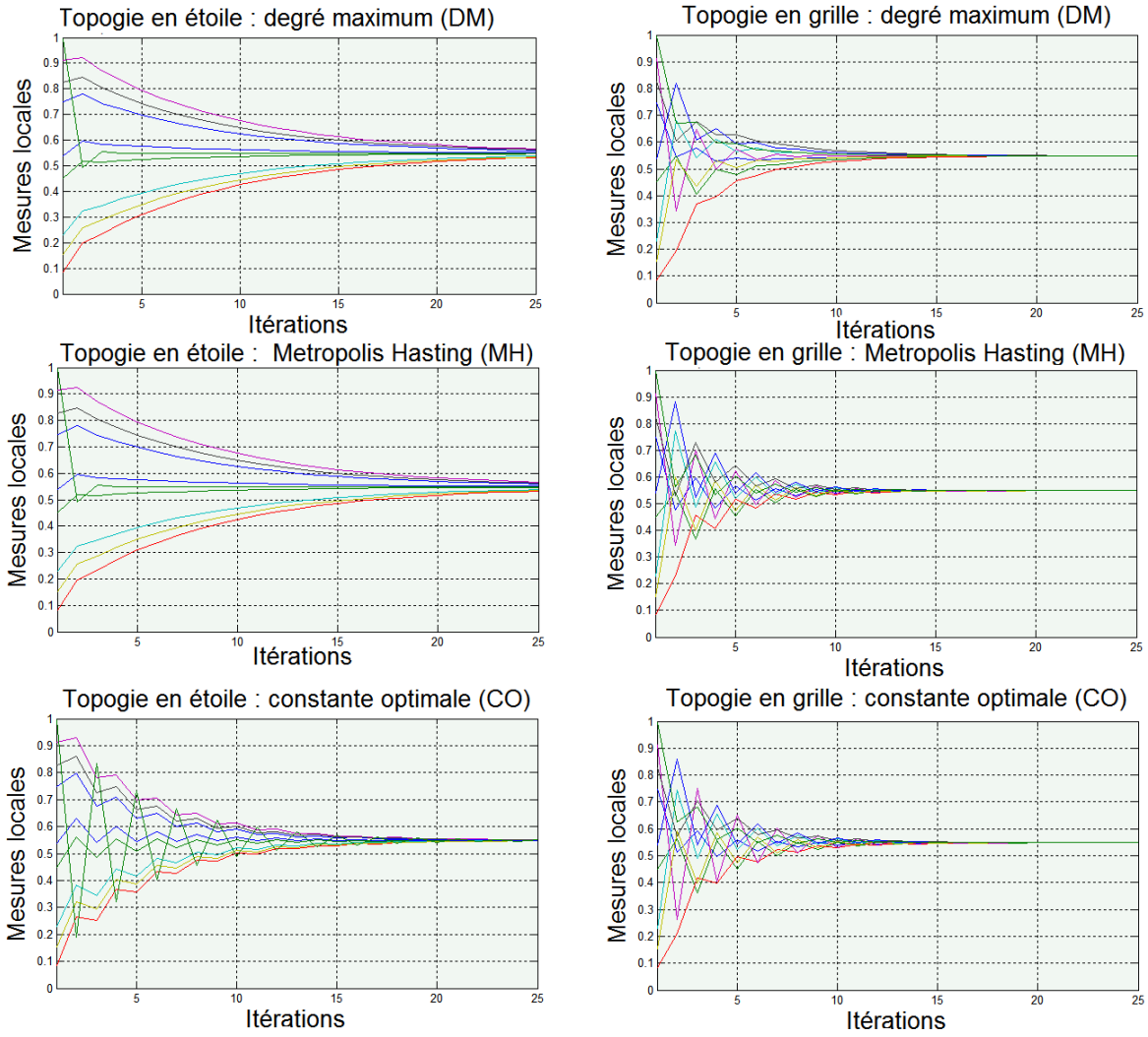


FIGURE 4.5 – Convergence de l'algorithme de consensus de moyenne en utilisant la méthode du degré maximum (DM), la méthode Metropolis hasting (MH) et la méthode de la constante optimale (CO) dans une topologie en étoile et en grille pour $N = 9$.

	GGA ($r = 0.18$)	GGA ($r = 0.35$)
DM	85.3	12
MH	48	9.5
CO	45.9	6.5

TABLE 4.3 – Temps de convergence des algorithmes de consensus de moyenne sur des graphes géométriques aléatoires $N = 100$ avec $r = 0.18$ et $r = 0.35$.

vitesse de convergence (voir tableau 4.3).

Lorsque $r = 0.35$, soit avec un réseau plus connecté, les trois algorithmes convergent nettement plus rapidement vers le consensus que pour $r = 0.18$, comme l'indique la figure 4.6 et le tableau 4.3). Toutefois, l'ordre des performances des algorithmes reste inchangé. Ici aussi la méthode MH reste proche de la méthode CO.

4.5 Algorithmes de bavardage asynchrones

4.5.1 Introduction

Classiquement, la tâche de calculer la valeur moyenne d'un ensemble de capteurs dans un RCSF a été destinée à un centre de fusion qui rassemble les informations de tous les capteurs du réseau, calcule la moyenne et communique le résultat aux capteurs. Néanmoins, cette approche présente des inconvénients. Par exemple, si un capteur est défaillant, les informations ne seront pas acheminées vers le capteur central, ainsi la moyenne que les capteurs veulent estimer peut être erronée. D'autre part, dans un scénario décentralisé, nous supposons que les capteurs partagent leur moyenne avec leur voisin. On peut montrer qu'avec une forte probabilité, après un certain nombre d'itérations, chaque capteur obtient une estimation précise de la moyenne du réseau. Dans cette section, nous utilisons les algorithmes de bavardage asynchrones pour décrire le processus décentralisé du calcul de la moyenne globale d'un réseau de capteurs sans fil.

Les algorithmes de bavardage apparaissent comme une approche pour maintenir l'évolutivité et la simplicité tout en conservant des performances acceptables. Lorsqu'un RCSF présente une défaillance de certains de ses capteurs, l'algorithme de bavardage dit gossip en anglais ne nécessite pas de modification ou d'action de récupération des données. En bref, les trois principaux avantages des algorithmes de bavardage sont leur simplicité, leur évolutivité et leur décentralisation. La simplicité implique que l'algorithme de bavardage est facile à mettre en œuvre et ne nécessite aucune infrastructure organisée. Les échanges entre voisins se font de manière simple sans la nécessité des protocoles de routage. L'évolutivité signifie le fait que chaque capteur dans le réseau dispose du même taux de bavardage, même avec le changement de taille du réseau, et enfin, la décentralisation implique qu'il n'y a pas de problèmes liés à l'effet de goulot d'étranglement ou bottleneck qui peut provoquer la congestion du réseau.

Un autre avantage des algorithmes de bavardage concerne l'asynchronisme de leur modèle temporel. Un seul capteur se réveille à chaque itération et échange sa valeur avec ses voisins, alors que dans le modèle synchrone traité dans la section 4.4, tous les capteurs sont réveillés et mettent à jour leur valeur simultanément. Donc, d'un point de vue énergétique, l'asynchronisme présente l'avantage d'augmenter la durée de vie des batteries, en mettant en veille certains capteurs qui ne participent pas au bavardage. De plus, certaines entraves telles que les contraintes physiques des composants électroniques, la synchronisation d'horloge, ainsi que les protocoles de synchronisation, nous incite à utiliser le modèle

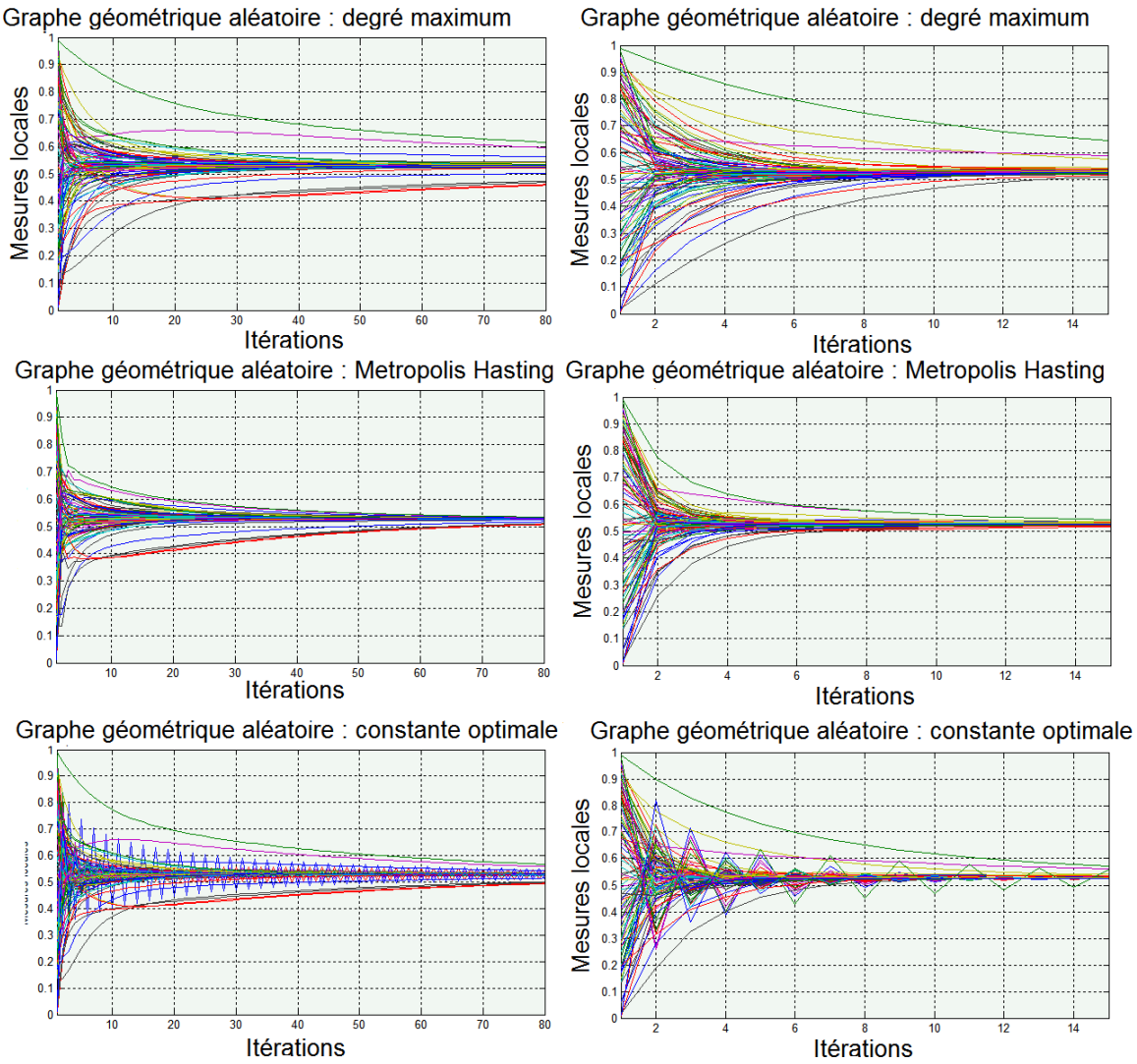


FIGURE 4.6 – Convergence de l'algorithme de consensus de moyenne (méthodes DM, MH et CO) dans un graphe géométrique aléatoire pour $N = 100$ avec $r = 0.18$ (à gauche) et $r = 0.35$ (à droite).

temporel asynchrone car il répond aux différentes exigences réelles des réseaux de capteurs sans fil.

4.5.2 Généralités et convergence

La valeur moyenne des valeurs initiales des capteurs est rassemblée dans un vecteur initial $\mathbf{x}(0)$ et les estimations à chaque itération k sont rassemblées dans $\mathbf{x}(k)$. Ces estimations sont mises à jour linéairement par l'équation

$$\mathbf{x}(k) = \mathbf{W}(k)\mathbf{x}(k-1). \tag{4.25}$$

L'asynchronisme des algorithmes se retrouve dans la forme particulière des $\mathbf{W}(k)$. D'une part, ces matrices sont aléatoires car elles dépendent des capteurs qui s'éveillent à l'itération k. Nous supposerons toutefois que le processus est stationnaire. D'autre part, les matrices $\mathbf{W}(k)$ sont proches de l'identité car seuls les coefficients associés aux lignes et aux colonnes du voisinage du capteur choisi par l'horloge sont concernés. En effet, les autres capteurs n'effectuent pas d'opérations à l'itération k.

Rappelons que la construction de $\mathbf{W}(k)$ est différente d'un algorithme à l'autre. Nous expliquerons ce point en détail dans la description de chaque algorithme de bavardage.

En prenant en compte tout l'historique, nous avons

$$\mathbf{x}(k) = \mathbf{W}(k)\mathbf{W}(k-1)....\mathbf{W}(1)\mathbf{x}(0) = \mathbf{P}(k)\mathbf{x}(0) \tag{4.26}$$

où $\mathbf{P}(k)$ est le produit $\mathbf{W}(k)....\mathbf{W}(1)$. L'objectif de la conception d'un algorithme de bavardage consiste à trouver un processus $\mathbf{W}(k)$ tel que le processus associé tende vers $\mathbf{1}\mathbf{1}^T/N$.

La convergence des algorithmes de consensus de moyenne nous mène à poser deux questions primordiales. La première est de vérifier si chaque capteur converge vers un consensus, en d'autres termes, est-ce qu'il existe un scalaire x_{moy} tel que $\lim_{k\to\infty} \mathbf{x}(k) = x_{moy}\mathbf{1}$. La seconde question consiste à savoir si le consensus est égal à la moyenne x_{moy}.

Pour un choix de simplification, nous posons $\mathbf{J} = \frac{\mathbf{1}\mathbf{1}^T}{N}$. Ensuite, nous mesurons la distance par rapport à la moyenne à chaque itération k qui peut également être interprétée comme une distance par rapport au consensus : $dist(k) = \mathbf{x}(k) - \mathbf{J}\mathbf{x}(k) = (\mathbf{I} - \mathbf{J})\mathbf{x}(k)$. L'erreur $\epsilon(k) = \mathbf{x}(k) - x_{moy}\mathbf{1} = \mathbf{x}(k) - \mathbf{J}\mathbf{x}(0)$ est utile pour mesurer la distance par rapport à la moyenne des valeurs initiales attribuées aux capteurs.

Pour répondre à la première vérification, nous considérons un algorithme distribué de consensus de moyenne, dans lequel, à chaque itération k, il existe une matrice $\mathbf{W}(k)$. Supposons que la séquence $\{\mathbf{W}(k)\}_{k\geq 1}$ est indépendante et identiquement distribuée.

Notons $\mathbf{E}[\mathbf{W}]$ la moyenne du processus stationnaire $\mathbf{W}(k)$. Ainsi, l'algorithme converge en moyenne vers la moyenne [Bén09] si et seulement si :

$$\mathbf{E}\left[\mathbf{W}\right]\mathbf{1} = \mathbf{1} \tag{4.27}$$

$$\mathbf{1}^T \mathbf{E}[\mathbf{W}] = \mathbf{1}^T \tag{4.28}$$

$$sp(\mathbf{E}[\mathbf{W}] - \mathbf{J}) < 1 \tag{4.29}$$

Nous retrouvons les mêmes critères que dans le cas synchrone (4.10), (4.11) et (4.12) mais avec la matrice $\mathbf{E}[\mathbf{W}]$ à la place de $\mathbf{W}$, ce qui s'explique par une preuve très similaire.

Un autre type de convergence a été proposé par les auteurs de [BGPS06] où l'algorithme de bavardage converge en moyenne quadratique.

4.5.3 Bavardage par paire

En 2006, Boyd et al. [BGPS05] et [BGPS06] ont proposé un algorithme asynchrone de bavardage par paire pour la modélisation des informations échangées entre capteurs dans un réseau connecté aléatoirement. À chaque coup d'horloge, un capteur i tiré aléatoirement choisit un de ses voisins j avec une probabilité P_{ij}. L'algorithme 4.2 montre la stratégie de bavardage par paire :

Algorithm 4.2: Algorithme de bavardage par paire : Boyd et al. [BGPS05] et [BGPS06]

1 **while** *la convergence n'est pas atteinte* **do**
2 Un capteur i est activé aléatoirement et choisit un voisin j parmi ses voisins N_i
3 Le capteur j envoie sa valeur au capteur i
4 Les deux capteurs i et j mettent à jour leur valeur par la moyenne de leur valeur à l'itération précédente : $x_i(k) \leftarrow \frac{x_i(k-1)+x_j(k-1)}{2}$, $x_j(k) \leftarrow \frac{x_i(k-1)+x_j(k-1)}{2}$
5 **end**

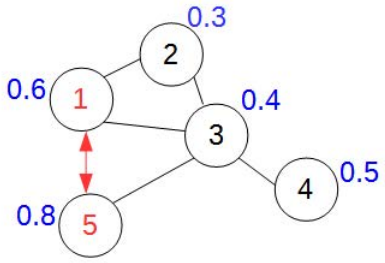
La figure 4.7 illustre une simple mise à jour de l'algorithme de bavardage par paire, ainsi que sa matrice de pondération $\mathbf{W}(k)$ dans un réseau composé de 5 capteurs déployés aléatoirement.

Ainsi, chaque paire de capteurs i et j met à jour sa valeur comme suit

$$x_i(k) = x_j(k) = \frac{x_i(k-1) + x_j(k-1)}{2}.$$

Les valeurs de tous les autres capteurs $v \neq \{i, j\}$ restent inchangées $x_v(k) = x_v(k-1)$. Cette mise à jour est exprimée par l'équation linéaire (4.25). Rappelons que $\mathbf{W}(k)$ dans le cas du bavardage par paire est une matrice de mise à jour avec comme éléments $W_{ii}(k) = W_{ij}(k) = W_{ji}(k) = W_{jj}(k) = \frac{1}{2}$ et $W_{vv}(k) = 1$ pour tous les capteurs qui ne participent pas au bavardage. La matrice $\mathbf{W}^{(ij)}$ peut être écrite sous forme matricielle

$$\mathbf{W}^{(ij)} = \mathbf{I} - \frac{(e_i - e_j)(e_i - e_j)^T}{2}, \tag{4.30}$$



$$\mathbf{W} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \qquad \mathbf{x}(0) = \begin{pmatrix} 0.6 \\ 0.3 \\ 0.4 \\ 0.5 \\ 0.8 \end{pmatrix} \qquad \mathbf{x}(1) = \begin{pmatrix} \mathbf{0.7} \\ 0.3 \\ 0.4 \\ 0.5 \\ \mathbf{0.7} \end{pmatrix}$$

$$\mathbf{x}(1) = \mathbf{W}\mathbf{x}(0)$$

FIGURE 4.7 – Exemple de bavardage par paire sur un réseau de 5 capteurs.

où $e_i = [0...0\,1\,0...0]^T$ est un vecteur de taille $N \times 1$ avec une i-ème composante égale à 1.

Le bavardage par paire préserve la somme et la moyenne, ainsi la matrice $\mathbf{W}(k)$ satisfait la condition $\mathbf{W}(k)\mathbf{1} = \mathbf{1}$. Ensuite, toute solution de consensus où tous les capteurs ont la même valeur est un point fixe $\mathbf{1}^T\mathbf{W}(k) = \mathbf{1}^T$.

Pour étudier la convergence en moyenne, la valeur moyennée de (4.30) est déterminée en respectant les règles de bavardage par paire [BGPS06]. Rappelons que le capteur i est choisi aléatoirement et uniformément parmi l'ensemble des capteurs V et j est choisi aléatoirement parmi l'ensemble des voisins de i. La valeur de la moyenne de la matrice $\mathbf{W}(k)$ ne dépend donc pas de l'indice de l'itération. Soit $\mathbf{E}[\mathbf{W}]$ la matrice de mise à jour moyennée. Boyd et al. [BGPS06] ont présenté une expression analytique de cette matrice

$$\mathbf{E}[\mathbf{W}] = \mathbf{I} - \frac{1}{2N}\mathbf{D} + \frac{\mathbf{P} + \mathbf{P}^T}{2N},$$

où $\mathbf{I}$ est une matrice identité de taille $N \times N$, la diagonale de la matrice $\mathbf{D} = diag(d_1, ..., d_N)$ a comme entrées $d_i = \sum_{j=1}^{N} [\mathbf{P}_{ij} + \mathbf{P}_{ji}]$. La matrice $\mathbf{P}$ est une matrice de transition avec des éléments $\mathbf{P}_{ij} \geq 0$ et $\sum_{j \in V} \mathbf{P}_{ij} = 1$. En outre, $\mathbf{P}_{ij} > 0$ si et seulement si $(i, j) \in E$. Par conséquent, la matrice $\mathbf{E}[\mathbf{W}]$ dépend du choix des probabilités attribuées à chaque arête dans le réseau, ainsi $\mathbf{E}[\mathbf{W}]$ dépend de la topologie du réseau mais assure la convergence de l'algorithme.

4.5.4 Bavardage par diffusion

Le bavardage par diffusion proposé par les auteurs de [AYSS09] profite de la nature de diffusion dans un canal sans fil, et nécessite une seule communication pour mettre à jour les valeurs des capteurs. Ce bavardage par diffusion est un algorithme qui repose à chaque itération k sur un capteur i choisi aléatoirement qui diffuse sa valeur à tous ses voisins j. Le principe de mise à jour est simple, une fois la valeur diffusée par le capteur i et reçue par les voisins j, chaque capteur j met à jour sa valeur par une moyenne pondérée de sa propre valeur et de la valeur reçue en fonction de l'équation suivante :

$$x_j(k) = \gamma x_j(k-1) + (1-\gamma)x_i(k-1) \tag{4.31}$$

avec $\gamma \in (0,1)$ qui symbolise la paramètre de mixage [AYSS09]. Par contre, tous les autres capteurs v qui ne sont pas voisins de i et y compris le capteur i, conservent la valeur de l'itération précédente : $x_v(k) = x_v(k-1)$ et $x_i(k) = x_i(k-1)$. L'algorithme 4.3 illustre la méthode de moyennage par diffusion.

Algorithm 4.3: Algorithme de bavardage par diffusion : Aysal et al. [AYSS09]

1 **while** *la convergence n'est pas atteinte* **do**
2 Un capteur i activé aléatoirement diffuse sa valeur $x_i(k-1)$ à tous ses voisins j
3 Les capteurs j mettent à jour leur valeur par : $x_j(k) \leftarrow \gamma x_j(k-1) + (1-\gamma)x_i(k-1)$ avec $\gamma \in (0,1)$
4 **end**

Ainsi, la mise à jour du vecteur des valeurs suit le schéma linéaire (4.25). La matrice de mise à jour $\mathbf{W}(k)$ est exprimée comme suit

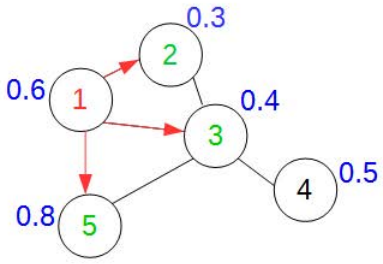
$$\mathbf{W}_{jk}^{(i)} \begin{cases} 1, & j \notin N_i,\ k = j \\ \gamma, & j \in N_i,\ k = j \\ 1-\gamma, & j \in N_i,\ k = i \\ 0, & \text{sinon} \end{cases}$$

en supposant que le capteur i se réveille à chaque itération et qui est la source de l'information à chaque itération.

La figure 4.8 présente un réseau de 5 capteurs avec $\gamma = 0.5$. Comme on peut le remarquer, les éléments de la diagonale de la matrice de pondération $\mathbf{W}(k)$ sont égaux à 1 pour les capteurs qui ne participent pas au bavardage par diffusion : c'est le cas du capteur 4, ainsi que du capteur 1 qui lui diffuse sa valeur à ses voisins 2, 3, et 5. La matrice $\mathbf{W}(k)$ est stochastique (la somme des lignes est égale à 1).

Deux propriétés importantes de la matrice $\mathbf{W}^{(i)}$ dans le cas d'un bavardage par diffusion :

1. $\mathbf{W}^{(i)}\mathbf{1} = \mathbf{1}$; $\mathbf{1}$ est un vecteur propre à droite.



$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \quad \mathbf{x}(0) = \begin{pmatrix} 0.6 \\ 0.3 \\ 0.4 \\ 0.5 \\ 0.8 \end{pmatrix} \quad \mathbf{x}(1) = \begin{pmatrix} 0.6 \\ \mathbf{0.45} \\ \mathbf{0.5} \\ 0.4 \\ \mathbf{0.7} \end{pmatrix}$$

$$\mathbf{x}(1) = \mathbf{W}\mathbf{x}(0)$$

FIGURE 4.8 – Exemple de bavardage par diffusion sur un réseau de 5 capteurs avec $\gamma = 0.5$.

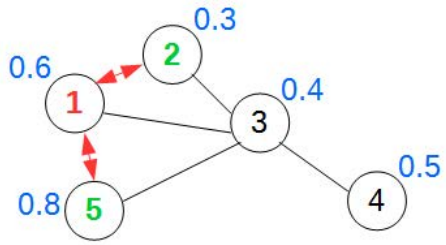
2. $\mathbf{1}^T\mathbf{W}^{(i)} \neq \mathbf{1}^T$; $\mathbf{1}$ n'est pas un vecteur propre à gauche pour toute matrice $\mathbf{W}^{(i)}$.

La matrice $\mathbf{W}^{(i)}$ n'est pas doublement stochastique et la convergence vers la moyenne n'est pas assurée. Si nous revenons à l'exemple de la figure 4.8, nous remarquons que la somme des lignes de la matrice $\mathbf{W}^{(i)}$ est égale à 1, par contre, la somme des colonnes est différente de 1, donc les propriétés d'une matrice doublement stochastique n'est pas applicable dans ce type de bavardage. En effet, la somme des valeurs des capteurs n'est pas conservée. Ainsi, l'algorithme ne converge pas vers la moyenne globale. Cependant, les auteurs de [AYSS09] admettent que la matrice moyennée de $\mathbf{W}(k)$ vérifie $\mathbf{1}^T\mathbf{E}[\mathbf{W}] = \mathbf{1}^T$. En d'autres termes, la somme des valeurs des capteurs est conservée en moyenne, cela garantit le rapprochement vers le consensus [AYSS09]. De plus, la condition sur le rayon spectral $sp(\mathbf{W} - \frac{\mathbf{1}\mathbf{1}^T}{N}) < 1$ est satisfaite (voir l'annexe B [AYSS09]).

L'idée des auteurs de [AYSS09] est d'exploiter l'intervalle de temps où la convergence est rapide par rapport aux autres algorithmes en moyennant plusieurs réalisations de l'algorithme pour avoir au final une estimation proche du consensus.

4.5.5 Bavardage par triplet

Un algorithme récemment introduit par Yang et al. [YWZ13], appelé bavardage par triplet améliore l'algorithme de bavardage par paire [BGPS05] en élargissant le groupe de bavardage, dans l'objectif d'atteindre une bonne estimation de la moyenne avec moins d'itérations [YWZ13]. Dans cet algorithme, à chaque itération, un capteur i se réveille aléatoirement, choisit deux de ses voisins j et j' et moyenne leur valeur par la nouvelle



$$\mathbf{W} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix} \quad \mathbf{x(0)} = \begin{pmatrix} 0.6 \\ 0.3 \\ 0.4 \\ 0.5 \\ 0.8 \end{pmatrix} \quad \mathbf{x(1)} = \begin{pmatrix} 0.56 \\ 0.56 \\ 0.4 \\ 0.5 \\ 0.56 \end{pmatrix}$$

$$\mathbf{x}(1) = \mathbf{W}\mathbf{x}(\mathbf{0})$$

FIGURE 4.9 – Exemple de bavardage par triplet de 5 capteurs.

estimation de la moyenne locale. La figure 4.9 montre un exemple de mise à jour de l'algorithme 4.4 dans un réseau de 5 capteurs.

Algorithm 4.4: Algorithme de bavardage par triplet : Yang et al. [YWZ13]

1 **while** *la convergence n'est pas atteinte* **do**
2 Un capteur i activé aléatoirement choisit deux voisins aléatoirement j et j' parmi ses voisins N_i
3 Les deux capteurs j et j' envoient leurs valeurs au capteur i
4 Les trois capteurs i, j et j' mettent à jour leur valeur par la moyenne de leur valeur à l'itération précédente : $x_i(k) \leftarrow \frac{x_j(k-1)+x_{j'}(k-1)+x_i(k-1)}{3}$, $x_j(k) \leftarrow \frac{x_j(k-1)+x_{j'}(k-1)+x_i(k-1)}{3}$, $x'_j(k) \leftarrow \frac{x_j(k-1)+x_{j'}(k-1)+x_i(k-1)}{3}$
5 **end**

Ainsi les capteurs i, j et j' mettent à jour leur valeur par

$$x_i(k) = x_j(k) = x_{j'}(k) = \frac{x_i(k-1) + x_j(k-1) + x_{j'}(k-1)}{3}.$$

Les valeurs de tous les autres capteurs restent inchangées $x_v(k) = x_v(k-1)$. La mise à jour est exprimée par l'équation (4.25). Ainsi, la matrice peut être écrite sous forme matricielle [YWZ13]

$$\mathbf{W}^{(ijj')} = \mathbf{I} - \frac{(e_i - e_j)(e_i - e_j)^T + (e_i - e_{j'})(e_i - e_{j'})^T + (e_j - e_{j'})(e_j - e_{j'})^T}{3},$$

où $e_i = [0...0\,1\,0...0]^T$ est un vecteur de taille $N \times 1$ avec une i-ème composante égale à 1.

L'expression analytique de $\mathbf{E}[\mathbf{W}]$ qui correspond au bavardage par triplet est notée

$$\mathbf{E}[\mathbf{W}] = \mathbf{I} - \frac{1}{3N}\mathbf{D} + \frac{\mathbf{P} + \mathbf{P}^T}{3N}$$

où $\mathbf{D} = diag(d_1, ..., d_N)$ est la diagonale de la matrice : $d_i = \sum_{j=1}^{N}[\mathbf{P}_{ij} + \mathbf{P}_{ji}]$.

Pour que $\mathbf{x}(k)$ converge vers $x_{moy}\mathbf{1}$, Yang et al. [YWZ13] ont prouvé que les trois conditions (4.27), (4.28) et (4.29) sont satisfaites dans le cas d'un bavardage par triplet.

En pratique, cet algorithme [YWZ13] nécessite quatre transmissions de messages par itération, une transmission est nécessaire à partir de i vers chaque capteur $\{j, j'\}$ voisins de i, et deux transmissions provenant de j et j' au capteur i. Enfin, le capteur i calcule la moyenne des trois capteurs i, j et j', et transmet la nouvelle estimation de la moyenne aux capteurs j et j'.

4.5.6 Bavardage géographique

L'idée de base du bavardage géographique [DSW08] est de permettre aux capteurs qui se situent loin du réseau de participer au bavardage à travers un routage multi-saut, alors que dans les autres types de bavardage étudiés précédemment, les échanges d'informations s'effectuent entre capteurs voisins à un saut. L'objectif principal de ce type de bavardage consiste à résoudre les problèmes liés au gaspillage inutile d'énergie entre capteurs, car certaines informations redondantes sont renvoyées dans le réseau comme par exemple dans le cas des algorithmes de bavardage par paire ou par diffusion où l'information circule entre voisins dont la valeur est quasi identique.

Algorithm 4.5: Algorithme de bavardage géographique : Dimakis et al. [DSW08]

1 **while** *la convergence n'est pas atteinte* **do**
2 Un capteur s est activé aléatoirement
3 Le capteur s commence le routage glouton vers le capteur le plus proche de la cible dont les coordonnées sont aléatoirement choisis
4 Le capteur t est supposé le plus proche de la cible et met à jour sa valeur après avoir reçu la valeur du capteur s : $x_t(k) \leftarrow \frac{x_s(k-1)+x_t(k-1)}{2}$
5 Retour au capteur s par un routage glouton, le capteur s met à jour sa valeur par la nouvelle valeur de t : $x_s(k) \leftarrow x_t(k)$
6 **end**

L'algorithme de bavardage géographique [DSW08] est basé sur l'hypothèse que les capteurs du réseau connaissent leur position exacte. L'idée principale consiste à ce que le routage géographique soit utilisé afin de permettre aux capteurs loin du réseau de participer au bavardage, plutôt que d'échanger des informations entre voisins à un seul saut. L'algorithme suppose que chaque capteur connaît sa propre position géographique, ainsi

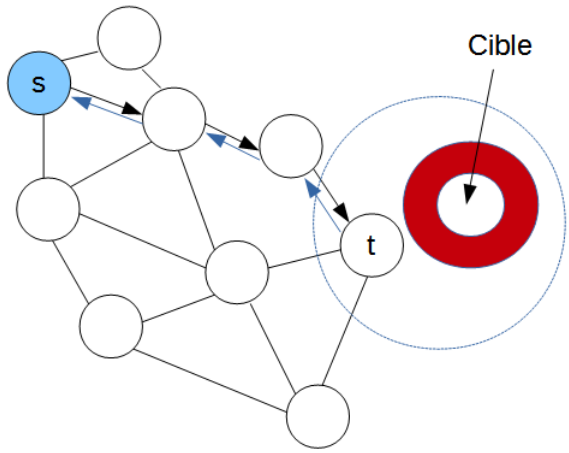


FIGURE 4.10 – Exemple de bavardage géographique.

que la position de ses voisins. Un capteur s dont la position est $l(s)$ est alors activé aléatoirement et choisit uniformément une cible dont les coordonnées sont (x, y) pour effectuer son bavardage. Le capteur s envoie ainsi sa valeur $x_s(k)$, sa position $l(s)$ et la position de la cible choisie aléatoirement (x, y) à son voisin (à un saut) le plus proche de la cible. Cette opération se répète jusqu'à ce qu'on arrive à un capteur t qui ne possède plus de voisins proches de la cible, et le seul parmi ses voisins à être proche de la cible (voir la figure 4.10). Il appartient donc au capteur t de prendre une décision aléatoire indépendante d'accepter ou non le message $m_s = \{x_s(k), l(s), (x, y)\}$ du capteur s. Si le message m_s est accepté par le capteur t, il met à jour sa valeur en fonction de $x_t(k) = \frac{x_s(k-1)+x_t(k-1)}{2}$ et génère ainsi un message $m_t\{x_t(k), l(t), l(s)\}$ pour atteindre le capteur s via le même routage glouton (greedy routing) [DSW08]. Ensuite, le capteur s met à jour sa valeur en fonction de $x_s(k) = \frac{x_t(k-1)+x_s(k-1)}{2}$. L'algorithme 4.5 résume les étapes du bavardage géographique.

Dimakis et al. [DSW08] ont prouvé que cette approche permet de converger plus rapidement vers le consensus de moyenne par rapport au bavardage par paire [BGPS05]. Cependant, l'inconvénient de cette méthode de bavardage est que cet algorithme a besoin d'un système pour connaitre les coordonnées globales du réseau et doit également envoyer des messages sur de longs trajets. Cela peut créer des problèmes de congestion [AYSS09].

4.5.7 Bavardage entre voisinage

L'algorithme de bavardage entre voisinage est très similaire au bavardage par paire ou par triplet. Il a été introduit par Nazer et al. [NDG09] et [NDG11]. Comme son nom l'indique, les estimations d'un ensemble de voisinage sont moyennées à chaque itération de l'algorithme 4.6. Un capteur i est activé aléatoirement, et met à jour sa valeur par la moyenne de la somme des valeurs de ses voisins j. Ainsi, tous les voisins j de i mettent à jour leur valeur par cette moyenne diffusée par le capteur i. Notons que ce type de

bavardage nécessite $m + 2$ messages à chaque itération, où m est la taille du voisinage.

Algorithm 4.6: Algorithme de bavardage entre voisinage : Nazer et al. [NDG09] et [NDG11]

while *la convergence n'est pas atteinte* **do**
- Un capteur i activé aléatoirement diffuse un message de réveil m_r
- Tout capteur j qui reçoit le message m_r répond au capteur i avec le message $m_j = \{x_j(k-1\})$
- Soit $N_i(k)$ l'ensemble des capteurs qui envoient les estimations au capteur i. Le capteur i met à jour sa valeur : $x_i(k) \leftarrow \frac{1}{|N_i(k)|} \sum_{j \in N_i(k)} x_j(k-1)$
- Le capteur i diffuse le message $m_i = \{x_i(k), N_i(k)\}$
- Tous les capteurs dans l'ensemble $N_i(k)$ (qui ont reçu le message m_i) mettent à jour leur valeur par $x_i(k)$

end

4.5.8 Résultats de simulations

La connectivité et la topologie du réseau joue un rôle clé en dictant la performance d'un algorithme de bavardage en termes de nombre de transmissions, ainsi que l'erreur d'estimation qui est est définie comme suit

$$e(k) = \frac{||\mathbf{x}(k) - \mathbf{x}_{moy}||}{||\mathbf{x}(0) - \mathbf{x}_{moy}||}$$

L'objectif de ces simulations consiste à examiner le temps de convergence de chaque algorithme de bavardage (voir les figures 4.11 et 4.12), et à évaluer les performances de chacun en calculant le nombre de messages transmis après k itérations. Ceci nous permet de déterminer le coût de communication nécessaire pour chaque algorithme.

Nous considérons un graphe géométrique aléatoire de 100 capteurs avec $r = 0.3$ et 200 capteurs avec $r = 0.2$. Les résultats sont moyennés sur 100 réalisations de chaque algorithme.

Chacun de ces algorithmes présente des avantages et des inconvénients et leurs performances sont représentées sur les figures 4.11 et 4.12. Comme l'axe des abscisses est en nombre de transmissions, la comparaison est possible. Notons que l'axe des ordonnées est en échelle logarithmique.

Nous constatons d'après les figures 4.11 et et 4.12 que le bavardage par paire converge très lentement. Les capteurs échangent leurs informations à plusieurs reprises avec leurs voisins à un saut. Ainsi, l'information est diffusée lentement dans le réseau.

Le bavardage géographique présente l'avantage d'être plus rapide par rapport au bavardage par paire, après un certain nombre d'itérations (figures 4.11 et 4.12). Ceci s'explique

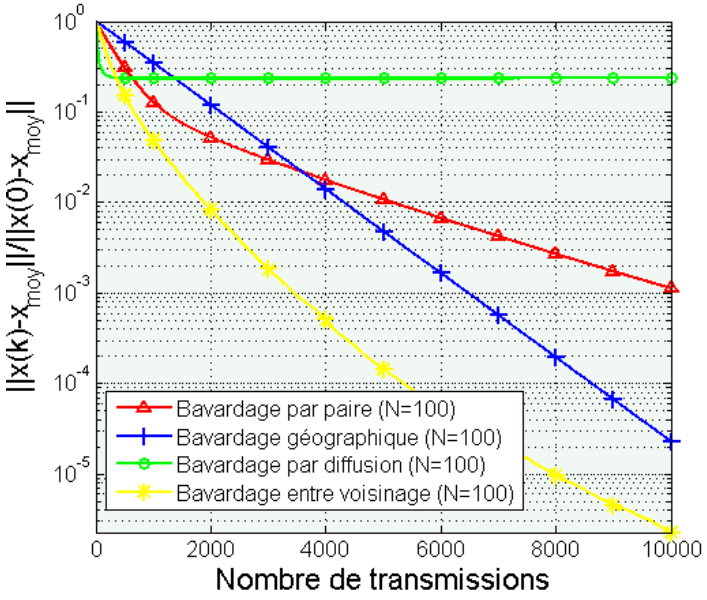


FIGURE 4.11 – L'erreur d'estimation en fonction du nombre d'itérations dans un réseau de 100 capteurs (graphe géométrique aléatoire) avec $r = 0.3$.

Coût de communication	Le nombre de messages transmis après k itérations
Temps de convergence	Le nombre d'itérations k pour atteindre le consensus

TABLE 4.4 – Définitions importantes : Coût de communication/Temps de convergence

du fait que les estimations des capteurs sont diffusées à l'ensemble du réseau [DSW08]. Cependant, l'inconvénient du bavardage géographique est qu'il nécessite une complexité de calcul pour le routage des informations entre les capteurs distincts.

L'algorithme de bavardage entre voisinage est un algorithme qui a été conçu afin d'exploiter la nature de diffusion des communications sans fil [NDG09] et [NDG11]. Nous avons vu que dans le cas du bavardage par paire, quand un capteur se réveille, un seul voisin reçoit l'information. Dans le bavardage entre voisins, au contraire, tous les voisins envoient leurs valeurs de retour à la source, et par la suite, ils mettent à jour leurs estimations par la moyenne de leurs valeurs. Ceci permet aux différents capteurs du réseau de se rapprocher de la valeur du consensus. Nous remarquons ainsi, que ce type de bavardage présente des performances supérieures à celles du bavardage par paire et géographique. Cependant, le coût de communication de cet algorithme est plus élevé. Autrement dit, le nombre de messages à chaque itération peut être très grand. Nous détaillerons ce point dans la suite des résultats.

Nous cherchons maintenant à évaluer les performances de chaque algorithme en calculant le nombre de messages transmis après k itérations. Avant de commenter les résultats du tableau 4.5, rappelons tout d'abord le nombre de messages nécessaire à transmettre pour chaque algorithme.

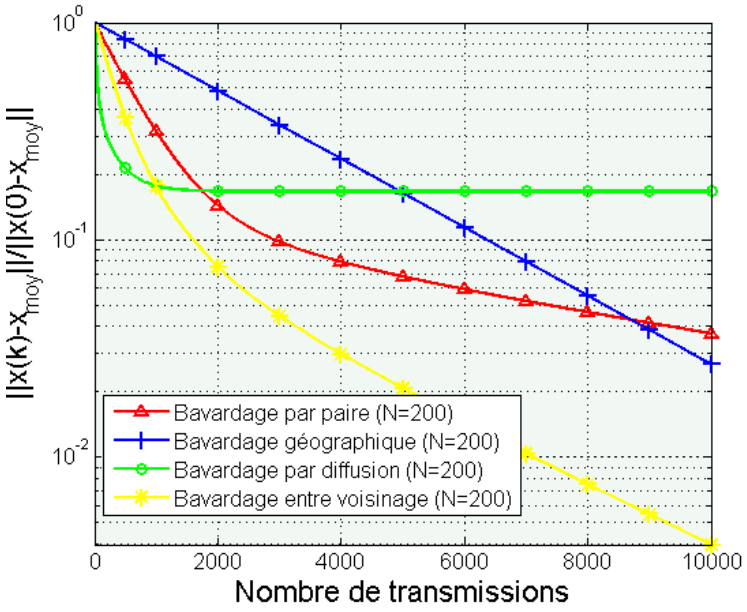


FIGURE 4.12 – L'erreur d'estimation en fonction du nombre d'itérations dans un réseau de 200 capteurs (graphe géométrique aléatoire) avec $r = 0.2$.

Paramètres de comparaison	**BPP**	**BG**	**BPD**	**BV**
Nombre de messages transmis après k itérations	2971	7285	1486	9505
Nombre d'itérations k	1486	1486	1486	443
Nombre de messages/nombre d'itérations	1.99	4.90	1	21.45

TABLE 4.5 – Évaluation des performances des algorithmes de bavardage : nombre de messages transmis après k itérations dans un réseau de 100 capteurs

Le bavardage par paire (BPP) nécessite deux transmissions (voir l'algorithme 4.2) pour faire la mise à jour entre chaque paire de capteurs. La communication est bidirectionnelle. Le coût de communication à chaque itération k est équivalent à

$$\text{Coût (en pratique BPP)} = 2 \text{ transmissions.}$$

Théoriquement, le coût du bavardage par paire est calculé jusqu'à convergence à ϵ près est donné par [BGPS05]

$$\text{Coût (théorique BPP)} = O\left(\frac{N}{r^2}\log(\epsilon^{-1})\right).$$

Comme le prouvent les auteurs de [BGPS06], un graphe est connexe avec une forte probabilité si $r \geq \sqrt{2 \times \frac{\log(N)}{N}}$, donc le coût de communication théorique du BPP peut être réécrit comme suit

$$\text{Coût (théorique BPP)} = O\left(\frac{N^2}{2 \times \log(N)}\log(\epsilon^{-1})\right),$$

où ϵ représente une précision de moyenne entre $(0, 1)$.

L'algorithme 4.5 montre le comportement du bavardage géographique (BG). Le routage est coûteux en termes de coûts de communications. Néanmoins, la convergence s'accélère par rapport au BPP après un certain nombre d'itérations k, car le BG permet une communication avec des capteurs qui sont loin du réseau [DSW08], ceci permet de propager l'information sur l'ensemble du réseau. Le bavardage géographique enregistre un facteur de $\sqrt{\frac{N}{\log(N)}}$ sur le bavaradage par paire en termes de coût de communication dans une topologie Euclidienne [DSW08], ainsi le coût de communication théorique du BG est donné par

$$\text{Coût (théorique BG)} = O\left(\frac{N^{1.5} \times \log(\epsilon^{-1})}{2 \times \sqrt{\log(N)}}\right).$$

Naturellement, dans un BG, le nombre de messages n'est pas constant à chaque itération k, donc le nombre de messages ne peut être calculé en pratique, mais nous pouvons l'exprimer par le formule suivante :

$$\text{Coût (en pratique BG)} = 2 \text{ transmissions} + 2 * \text{Nombre de sauts (distance)}$$

Dans le bavardage entre voisinage (BV), un capteur i (source) calcule la moyenne de tous ses voisins et la diffuse. La source et ses voisins mettent à jour leurs valeurs par cette moyenne. À la fin de chaque itération k, $m + 2$ messages sont envoyés, où m est la taille du voisinage (voir la figure 1 de [NDG11]). Selon Nazer et al. [NDG11] et [NDG09], le BV nécessite $O\left(\frac{N^2}{m^2}\log(\epsilon^{-1})\right)$ itérations pour converger vers le consensus et ceci dans une topologie en grille $\sqrt{N} \times \sqrt{N}$. Malgré le nombre de messages trop élevé du BV, les auteurs de [NDG11] montrent que les propriétés des interférences dans un canal sans fil

et l'utilisation des codes linéaires peuvent être exploitées pour faire un moyennage local rapide entre voisinage tout en réduisant la consommation d'énergie.

Le bavardage par diffusion (BPD) nécessite un seul message à chaque itération k. Aysal et al. [AYSS09] ont prouvé que l'algorithme de BPD converge après $O\left(\frac{N^{2.5}}{\sqrt{\log(N)}}\log(\epsilon^{-1})\right)$ itérations, cependant la valeur du consensus n'est pas la moyenne globale des valeurs initiales des capteurs. Les auteurs de [AYSS09] définissent un paramètre de mixage optimale pour l'équation (4.31) :

$$\gamma^* = \frac{N - \lambda_{N-1}(\mathbf{L})}{2N - \lambda_{N-1}(\mathbf{L})} = 0.5$$

Notons que le temps de convergence de chaque algorithme est différent de la notion du coût de communication comme expliqué dans le tableau 4.4. En effet le premier paramètre d'évaluation permet de savoir après combien d'itérations l'algorithme de bavardage converge vers le consensus $x_{moy}\mathbf{1}$, alors que le second paramètre permet le calcul du nombre de messages transmis après k itérations.

Si nous revenons aux résultats du tableau 4.5, nous constatons que le bavardage par diffusion nécessite moins de messages. D'ailleurs, c'est le seul bavardage où l'algorithme a besoin d'un seul message à chaque itération k, autrement dit, la communication est unidirectionnelle sans voie de retour.

Le bavardage entre voisinage est le bavardage le moins performant en termes de coût de communication avec 9505 messages après 443 itérations (voir le tableau 4.5), ceci est dû notamment au nombre important de messages transmis par les voisins vers la source.

Les résultats du bavardage géographique montrent que son coût de communication est aussi élevé. Nous avons expliqué précédemment que l'algorithme de BG a besoin d'un certain nombre de sauts pour arriver à destination, ainsi le nombre de messages croît si la distance entre la source et la destination est longue.

En ce qui conçerne le bavardage par paire, son coût de communication par itération est moins élevé par rapport au bavardage géographique, car les mises-à-jour sont effectuées entre paire de capteurs voisins.

4.6 Algorithme de bavardage Push-Sum

L'algorithme Push-Sum introduit par Kempe [KDG03] s'est montré particulièrement intéressent pour les algorithmes de bavardage. Ce dernier est entièrement distribué et ne nécessite aucun traitement centralisé. De plus, la défaillance de certains capteurs ne modifient pas le taux de bavardage dont dispose les capteurs du réseau. Un autre point intéressant de cet algorithme est son adaptabilité au modèle asynchrone.

Nous verrons par la suite que l'utilisation du formalisme du Push-Sum est différent du formalisme des autres algorithmes de bavardage, et que l'utilisation du principe de diffu-

sion, permet une convergence rapide vers la moyenne globale avec un nombre minimal de messages transmis.

4.6.1 Bavardage Push-Sum par paire sans voie de retour

Chaque capteur dispose de deux variables : la somme $s_i(k)$ et le poids $w_i(k)$. Au démarrage de l'algorithme, les deux variables sont initialisées par $s_i(0) = x_i(0) = x_i$ et $w_i(0) = 1$ avec $x_i(k)$ l'estimée de chaque capteur pour toute itération k. Un capteur i est activé aléatoirement, il met à jour la valeur de sa somme $s_i(k)$ par $\beta_i s_i(k-1)$ et la valeur de son poids $w_i(k)$ par $\beta_i w_i(k-1)$, ainsi, son estimée $x_i(k)$ est le quotient $\frac{s_i(k)}{w_i(k)}$. Après la mise à jour du capteur i, ce dernier choisit un voisin j aléatoirement parmi N_i, et il lui envoie l'information $(1-\beta_i)s_i(k-1)$ et $(1-\beta_i)w_i(k-1)$. Ensuite, le capteur j met à jour la valeur de sa somme $s_j(k)$ par $s_j(k) = s_j(k-1) + (1-\beta_i)s_i(k-1)$ et la valeur de son poids $w_j(k)$ par $w_j(k) = w_j(k-1) + (1-\beta_i)w_i(k-1)$, ainsi son estimée $x_j(k)$ est le quotient $\frac{s_j(k)}{w_j(k)}$. Nous utiliserons un paramètre $\beta_i \in [0,1]$ qui permettra de varier l'information transmise au voisin j. Nous expliquerons en détail dans la suite de cette section comment optimiser ce paramètre et ceci à travers une analyse statistique. L'algorithme 4.7 décrit les étapes de ce bavardage.

Algorithm 4.7: Algorithme de bavardage Push-Sum par paire sans voie de retour

1 **while** *la convergence n'est pas atteinte* **do**

`/* Mise à jour du capteur` i `activé aléatoirement */`

2 Un capteur i est activé aléatoirement, il met à jour sa valeur par $s_i(k) = \beta_i s_i(k-1)$ et $w_i(k) = \beta_i w_i(k-1)$, avec $x_i(k) = \frac{s_i(k)}{w_i(k)}$ l'estimée de i à l'itération k

3 Le capteur i envoie l'information $(1-\beta_i)s_i(k-1)$ et $(1-\beta_i)w_i(k-1)$ au capteur j (un voisin sélectionné parmi N_i voisins de i)

`/* Mise à jour du voisin de` i `activé aléatoirement */`

4 Le capteur j se met à jour après avoir reçu les informations de i : $s_j(k) = s_j(k-1) + (1-\beta_i)s_i(k-1)$ et $w_j(k) = w_j(k-1) + (1-\beta_i)w_i(k-1)$ avec $x_j(k) = \frac{s_j(k)}{w_j(k)}$ l'estimée de j à l'itération k

5 **end**

L'algorithme 4.7 est unidirectionnel, l'information circule uniquement dans un seul sens, alors que dans le bavardage par paire, les informations sont transmises par les deux capteurs voisins. Ainsi, cet algorithme nécessite une seule transmission. La convergence de l'algorithme 4.7 repose sur une propriété fondamentale appelée conservation de la masse [KDG03], selon laquelle les relations suivantes doivent être satisfaites pour toute itération

k :

$$\sum_{i=1}^{N} s_i(k) = \sum_{i=1}^{N} x_i(0) = x_{moy}N \tag{4.32}$$

$$\sum_{i=1}^{N} w_i(k) = N \tag{4.33}$$

Avec x_{moy} la moyenne des valeurs initiales attribuées aux capteurs et N le nombre de capteurs du réseau.

4.6.2 Bavardage Push-Sum par diffusion

Nous avons conclu précédemment que le bavardage par diffusion [AYSS09] présente des performances supérieures à celles des autres algorithmes de bavardage, c'est la raison pour laquelle, nous nous sommes basés sur l'idée d'introduire le mécanisme Push-Sum [KDG03] dans l'algorithme de bavardage par diffusion et par la suite analyser ses performances en termes de temps de convergence et de coût de communication. L'algorithme 4.8 présente la version Push-Sum par diffusion.

Algorithm 4.8: Algorithme de bavardage Push-Sum par diffusion

1 **while** *la convergence n'est pas atteinte* **do**

`/* Mise à jour du capteur` i `activé aléatoirement */`

2 Un capteur i est activé aléatoirement, il met à jour sa valeur par $s_i(k) = \beta_i s_i(k-1)$ et $w_i(k) = \beta_i w_i(k-1)$, avec $x_i(k) = \frac{s_i(k)}{w_i(k)}$ l'estimée de i à l'itération k

3 Le capteur i diffuse l'information $\beta_{ij}s_i(k-1)$ et $\beta_{ij}w_i(k-1)$ à ses voisins $j \in N_i$ avec $(\sum_j \beta_{ij} = 1 - \beta_i)$

`/* Mise à jour de tous les voisins` $j \in N_i$ `de` i `*/`

4 Le capteur j se met à jour après avoir reçu les informations de i : $s_j(k) = s_j(k-1) + \beta_{ij}s_i(k-1)$ et $w_j(k) = w_j(k-1) + \beta_{ij}w_i(k-1)$ avec $x_j(k) = \frac{s_j(k)}{w_j(k)}$ l'estimée des voisins j à l'itération k

5 **end**

Cet algorithme est unidirectionnel, le flux d'information part d'un capteur d'émission à un certain nombre de capteurs de réception. La conservation de la somme $s_i(k)$ et du poids $w_i(k)$ est aussi assurée dans l'algorithme Push-Sum par diffusion. A l'itération $k = 0$, nous avons $s_i(0) = x_i(0) = x_i$ et $w_i(0) = 1$. Quand l'horloge du i-ème capteur sonne, à l'itération k, le capteur i contrairement à l'algorithme Push-Sum par paire 4.7, envoie toutes les informations sur sa valeur à tous ses voisins $j \in N_i$. Cet algorithme nécessite une seule transmission à chaque itération k. La figure 4.13 illustre le schéma de fonctionnement du bavardage Push-Sum par diffusion.

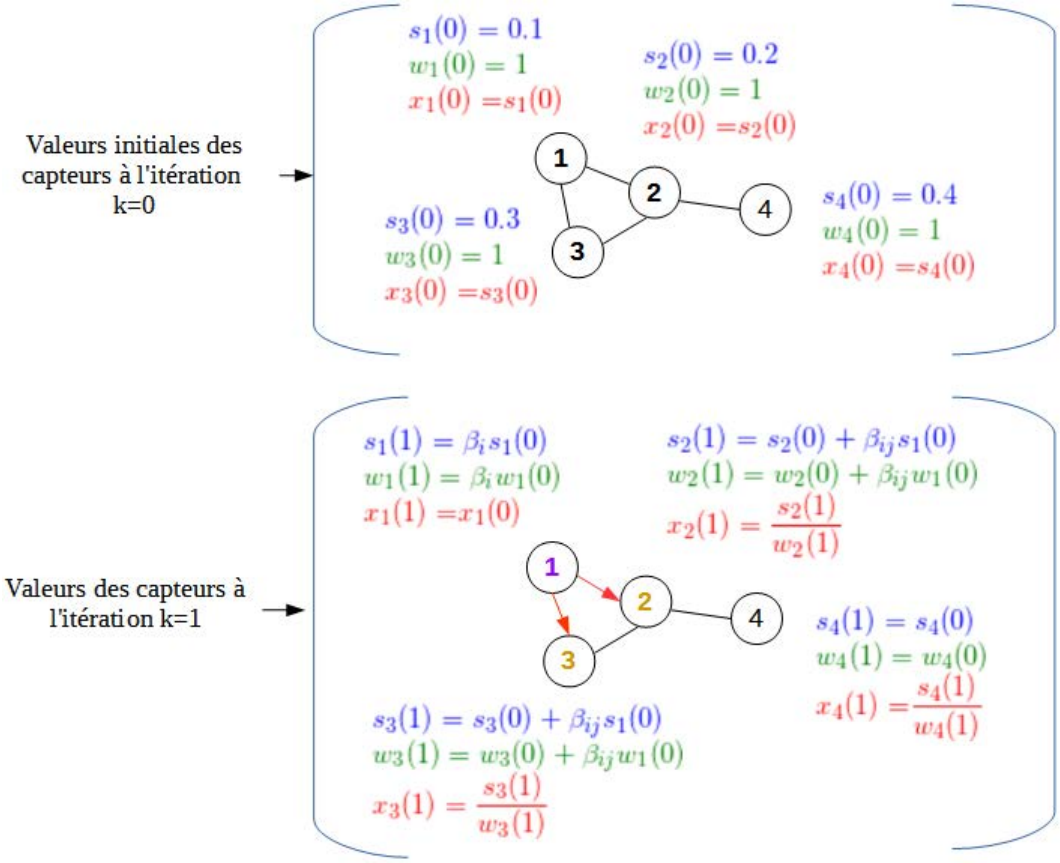


FIGURE 4.13 – Exemple de bavardage Push-Sum par diffusion dans un réseau de 4 capteurs.

Le capteur i et ses voisins j mettent à jour leurs valeurs par les équations suivantes :

$$\begin{cases} s_i(k) = \beta_i s_i(k-1) \\ w_i(k) = \beta_i w_i(k-1) \\ s_j(k) = s_j(k-1) + \beta_{ij} s_i(k-1) \\ w_j(k) = w_j(k-1) + \beta_{ij} w_i(k-1) \end{cases} \tag{4.34}$$

avec $x_i(k) = \frac{s_i(k)}{w_i(k)}$ et $x_j(k) = \frac{s_j(k)}{w_j(k)}$.

Nous avons défini deux méthodes pour le choix de β_{ij}. La première méthode consiste à rendre le paramètre β_{ij} dynamique avec comme condition $\sum_j \beta_{ij} = 1 - \beta_i$:

$$\beta_{ij} = \begin{cases} \beta_i & \text{si } i = j \\ N_i^{-1}(1-\beta_i) & \text{si } i \neq j \text{ et } j \in N_i \\ 0 & \text{sinon} \end{cases} \tag{4.35}$$

avec N_i le nombre de voisins de i.

La deuxième méthode consiste à utiliser l'information sur le nombre de voisins du capteur i [ICHJ12] :

$$\beta_{ij} = \begin{cases} \frac{1}{1+N_i} & \forall i, j \\ 0 & \text{sinon} \end{cases} \tag{4.36}$$

On définit les éléments de la matrice de diffusion $\mathbf{W}^{(ij)}$ à l'itération k en utilisant la méthode (4.35) comme suit

$$\mathbf{W}^{(ij)}(k) = \begin{cases} \beta_i & \text{si } i = j \\ N_i^{-1}(1-\beta_i) & \text{si } i \neq j \text{ et } j \in N_i \\ 1 & \text{si } i = j \text{ et } j \notin N_i \\ 0 & \text{sinon} \end{cases} \tag{4.37}$$

Les éléments de la matrice de diffusion $\mathbf{W}^{(ij)}$ sont représentés par la méthode (4.36) comme suit

$$\mathbf{W}^{(ij)}(k) = \begin{cases} \frac{1}{1+N_i} & \forall i, j \text{ et } j \in N_i \\ 1 & \text{si } i = j \text{ et } j \notin N_i \\ 0 & \text{sinon} \end{cases} \tag{4.38}$$

où N_i est l'ensemble des voisins du capteur i qui se réveille à l'itération k.

La matrice de diffusion stochastique $\mathbf{W}(k)$ satisfait les conditions suivantes :

— $\mathbf{1}$ est le vecteur propre à droite de toute la séquence $\{\mathbf{W}_{k\geq 1}\}$; $\mathbf{W}(k)\mathbf{1} = \mathbf{1}$

— $\mathbf{1}$ n'est pas le vecteur propre à gauche de $\{\mathbf{W}_{k\geq 1}\}$; $\mathbf{1}^T\mathbf{W}(k) \neq \mathbf{1}^T$

La convergence de la séquence $\{\mathbf{W}\}_{k\geq 1}$ vers le consensus a été prouvé théoriquement dans les travaux de [BBT+10] (théorème 4.1) et sa vitesse de convergence dans [ICHJ12].

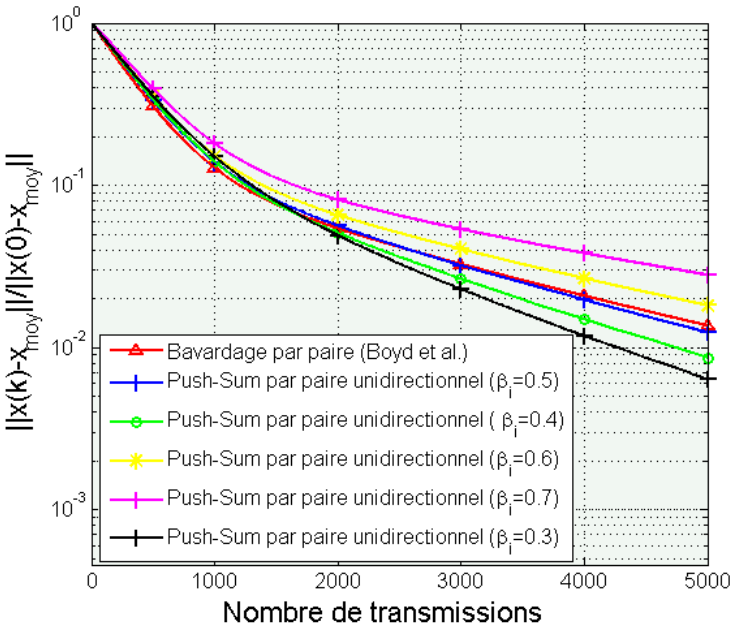


FIGURE 4.14 – L'erreur d'estimation en fonction du nombre d'itérations dans un réseau de 100 capteurs (graphe géométrique aléatoire) avec $r = 0.3$ (algorithme 4.7).

4.6.3 Résultats de simulations

Dans cette première partie de simulations, nous évaluons les performances de l'algorithme Push-Sum par paire (4.7) pour un réseau de 100 capteurs avec $r = 0.3$ et 100 réalisations de l'algorithme. La comparaison des résultats de la figure 4.14 s'effectue par rapport au bavardage par paire. Les performances des algorithmes de bavardage sont en fonction du nombre d'itérations et l'erreur moyenne logarithmique. Nous avons exécuté l'algorithme Push-Sum par paire 4.7 en faisant varier le paramètre β_i dans l'intervalle $0.3 < \beta_i < 0.7$.

Nous remarquons à partir de la figure 4.14 que la convergence de l'algorithme 4.7 est rapide pour un $\beta_i = 0.3$. Ainsi, du point de vue résultats les performances de l'algorithme Push-Sum par paire est supérieure à celles du bavardage par paire. Ceci est essentiellement dû à l'utilisation du principe de diffusion dans un canal sans fil et ses avantages cités auparavant. De plus, le coût de communication du Push-Sum par paire est moins élevé puisque l'algorithme nécessite une seule transmission à chaque itération k.

Nous proposons dans cette deuxième partie de simulations une comparaison des résultats du bardage Push-Sum par diffusion (première méthode (4.35)) par rapport au bavardage Push-Sum par diffusion (deuxième méthode (4.36) [ICHJ12]). Les performances sont quasiment identiques. Au regard des simulations sur la figure 4.15, nous remarquons qu'avec des valeurs $\beta_i \simeq 0$, l'algorithme converge plus rapidement. Par exemple, avec un $\beta_i = \{10^{-3}, 10^{-5}, 10^{-10}\}$, la vitesse de convergence est plus rapide et permet à l'algorithme 4.8 d'atteindre une performance supérieure en convergeant plus rapidement par rapport à la méthode (4.36). De plus, notre approche (4.35) présente l'avantage de varier la vitesse

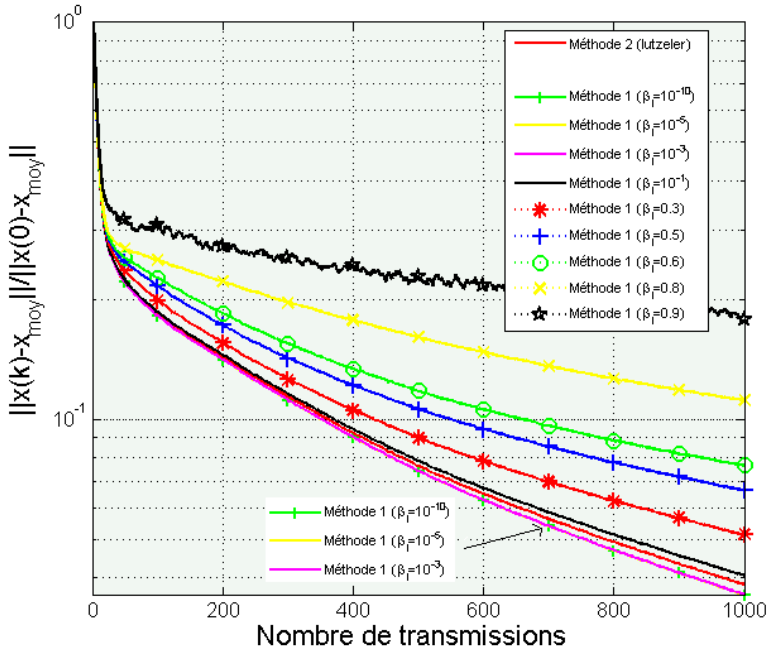


FIGURE 4.15 – L'erreur d'estimation en fonction du nombre d'itérations dans un réseau de 100 capteurs (graphe géométrique aléatoire) avec $r = 0.3$ (algorithme 4.8).

de convergence avec le paramètre β.

Les résultats de la figure 4.16, nous montrent l'avantage des algorithmes de diffusion en terme de temps de convergence. Précédemment, nous avons conclu que le bavardage par diffusion permettait une convergence rapide au début du bavardage, cependant le moyenne globale n'est pas conservée après k itérations. L'avantage de cette approche Push-Sum par diffusion est d'exploiter le principe de diffusion tout en conservant la moyenne. On voit clairement que le Push-Sum par diffusion dépasse largement les performances des autres algorithmes de bavardage avec un $\beta_i = 10^{-3}$.

4.7 Conclusion

Dans ce chapitre, nous avons examiné les différents algorithmes de consensus de moyenne et leurs avantages pour des applications décentralisées. Le chapitre 5 exploitera ces résultats afin de comprendre au mieux l'intérêt de ces algorithmes distribués dans la détection des trous de couverture mais surtout la gestion d'énergie dans les réseaux de capteurs.

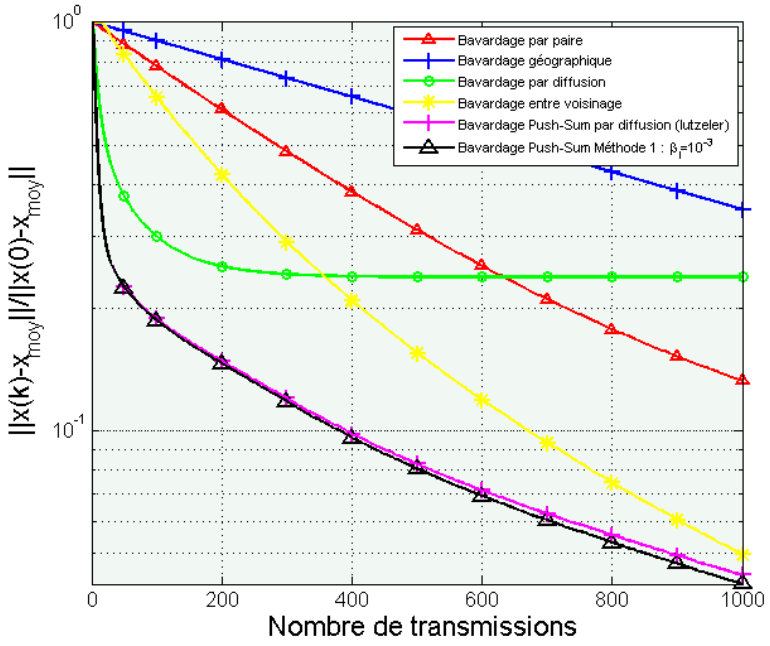


FIGURE 4.16 – L'erreur d'estimation en fonction du nombre d'itérations dans un réseau de 100 capteurs (graphe géométrique aléatoire) avec $r = 0.3$. Comparaison des performances de l'algorithme (4.8) avec d'autres algorithmes de bavardage.

Chapitre 5

Détection des trous de couverture dans un RCSF

5.1 Introduction

Un grand nombre de problématiques élémentaires dans le domaine des réseaux de capteurs peuvent se reformuler en utilisant le vocabulaire des espaces topologiques [HY61], [Arm83] et [Mas91]. L'intérêt de ce changement de point de vue est l'introduction de nombreux outils mathématiques qui caractérisent, même partiellement, ces espaces. En effet avant même d'y adjoindre une géométrie par l'introduction d'une distance, un espace topologique a une forme et cette forme est informative. Saisir ces informations est

l'enjeu du calcul des invariants topologiques et leur robustesse constitue la force de cette approche. Comme le soulignent les deux livres récents [AF07] et [Ghr14], la topologie algébrique s'est révélée particulièrement payante dans de nombreux domaines allant de la cinématique au clustering en passant par le traitement du signal et des données, le biomédical et la génétique.

Nous comptons parmi les invariants topologiques les groupes d'homologie [Hat01], [Sat99] qui permettent, selon l'interprétation d'Alexandrov, de compter les composantes connexes, les trous, les tunnels, etc. C'est-à-dire que les groupes d'homologie permettent de quantifier la connectivité d'un espace. C'est pourquoi ils sont des invariants de choix dans le domaine des réseaux de capteurs pour lesquels les questions de connectivité sont primordiaux.

D'autres invariants topologiques sont disponibles comme le groupe fondamental mais la plupart s'avère généralement difficile à calculer alors que les groupes d'homologie et de cohomologie sont foncièrement obtenus par des manipulations d'algèbre linéaire et donc de calcul matriciel [EH10], [DEG99].

La robustesse de cette approche est telle que de Silva et Ghrist dans [dSG06] et [dSG07] montrent qu'il est possible de déterminer la présence de trous de couverture d'un réseau de capteurs sans connaitre la position géographique des capteurs. La relation entre le champ mesuré et la géographie se fait au travers d'une relation entre les rayons de communication et les rayons de couverture de chaque capteurs qui permet de construire une approximation de l'espace couvert par le réseau de capteurs. Leur solution repose sur le calcul d'un groupe d'homologie.

De plus, s'il n'y a pas de trou de couverture et si le bord du champ couvert par le réseau est connu, alors, en calculant un autre groupe d'homologie, il est possible de limiter le nombre de capteurs actifs tout en garantissant la couverture. Naturellement, des mises en œuvre ont été depuis proposées, comme dans [TSJ10] où le problème est réinterprété comme un problème distribué d'optimisation ou dans [MJ07] qui utilise l'isomorphisme entre le groupe d'homologie et les fonctions harmoniques sur un graphe qui sera vu dans ce manuscrit ou dans [DGJM12] qui simplifie l'espace approchant avant de faire le calcul du groupe d'homologie.

5.2 Homologie

5.2.1 Espaces combinatoire

Les espaces sont un ingrédient de base de la méthode topologique. Toutefois, la généralité de ces espaces n'en permet pas une représentation simple qui s'avèrerait utilisable dans des calculs automatisés. C'est pourquoi, nous nous concentrerons sur les espaces combinatoires qui sont formés par un assemblage fini de briques de base, voir par exemple [Koz08]. Il suffit alors de gérer la notice d'assemblage pour manipuler l'espace. Cette notice d'assemblage est fournie par l'application de bord. Cette application contient aussi toutes informations

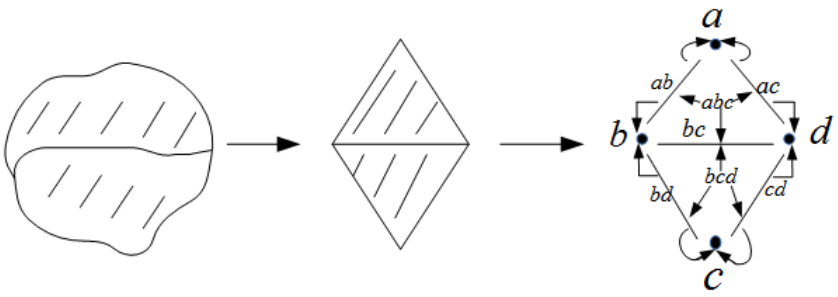


FIGURE 5.1 – De l'espace continu à l'espace combinatoire.

concernant les voisinages. Par exemple, l'espace représenté à droite de la figure 5.1 peut être triangulé pour obtenir l'espace topologiquement équivalent mais plus simple au centre. Ce dernier espace est une construction purement combinatoire qui assemble des sommets, des arêtes et des triangles et dont la notice d'assemblage est fournie par la notion de bord identifiée à droite de la figure 5.1 par des flèches. Des structures de données informatiques [BKT15] permettent de gérer efficacement ces espaces combinatoires qui seront alors à la base des algorithmes de calcul d'invariants topologiques.

5.2.2 Complexe simplicial

La discrétisation des espaces bidimensionnels est souvent réalisée par une triangulation en utilisant par exemple la triangulation de Delaunay [DO11]. Le résultat final est donné par un ensemble de sommets, d'arêtes et de triangles. Ces morceaux sont recollés entre eux en identifiant les bords de chaque arête avec ses sommets et le bord de chaque triangle avec ses arêtes.

Les complexes simpliciaux forment une généralisation de ces triangulations. Les éléments de bases sont les simplexes, c'est-à-dire les sommets, les arêtes, les triangles, les tétraèdres, les 5-simplexes, etc. Les simplexes sont regroupés selon leur dimension et les bords d'un n-simplexe sont des $n-1$ simplexes appelés ses faces.

Cette construction implique que les faces de chaque simplexe font aussi partie du complexe. Une description purement combinatoire est alors suffisante pour représenter un espace, sa réalisation en tant que complexe simplicial n'est plus nécessaire.

Ainsi, nous nous contenterons de l'approche combinatoire des complexes simpliciaux abstraits finis. Par définition de [EH10], un tel complexe est constitué d'un ensemble fini V de sommets et d'une famille S de sous-ensembles de V, nommés simplexes, qui satisfait la contrainte de clôture par sous-ensemble. C'est-à-dire que si σ est un élément de S et que $\tau \subset \sigma$ alors τ est aussi un élément de S. Une sous-partie τ d'un simplexe σ est appelée une face ; la contrainte exprime alors qu'une face d'un simplexe est aussi un simplexe.

Un n-simplexe σ, de dimension n, est donc un ensemble de $n+1$ sommets $\{v_0, v_1, ..., v_n\}$.

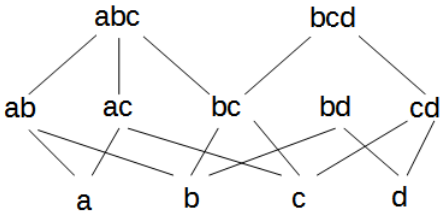


FIGURE 5.2 – Diagramme de Hasse.

Ses faces propres sont les $n-1$-simplexes $\{v_0, ..., \tilde{v_i}, ..., v_n\}$, où $\tilde{v_i}$ indique l'omission du sommet v_i.

Les règles d'assemblage des différents blocs de base que sont les simplexes sont encodés dans la relation d'un simplexe avec ses faces propres. Ainsi le bord d'un simplexe est la donnée de ses faces propres. Par exemple, le bord du 2-simplexes $\{a, b, c\}$, *i.e.* un triangle, est formé par les trois 1-simplexes *i.e.* arêtes, $\{a, b\}$, $\{a, c\}$ et $\{b, c\}$.

En se restreignant à la dimension 1, les complexes simpliciaux abstraits se particularisent en graphes qui se révèlent être des structures très souples aux multiples applications. Les nouvelles possibilités apportées par les dimensions supérieures et les outils y attenant commencent à porter leurs fruits [GGB16].

Un complexe simplicial abstrait est aussi encodable sous la forme d'un diagramme de Hasse [Cam94] pour la relation d'inclusion des simplexes entre eux. Cette forme met plus encore en exergue l'aspect combinatoire de cette structure.

Reprenons le complexe simplicial issu de la triangulation de la figure 5.1. Le diagramme de Hasse s'obtient en regroupant par ligne les simplexes selon leur dimension et en reliant les simplexes avec leurs faces propres. Ce diagramme suffit pour connaitre la connexité de l'espace sous-jacent. Par exemple, il est clair que le sommet a et d sont dans la même composante car il y a un chemin dans le diagramme de Hasse allant de a à d via c en passant par les arêtes ac puis cd. De même, l'arête ab peut être déformée en cd car, en balayant le triangle abc, elle peut être déformée en bc puis en cd via le triangle bcd.

Les complexes cellulaires réguliers sont une petite généralisation des complexes simpliciaux. Les briques de bases ne sont pas nécessairement des simplexes mais plutôt des sommets, des arêtes, des disques et des sphères de toutes dimensions. Le recollement d'une sphère de dimension d se fait le long de son bord qui est une sphère de dimension $d-1$.

5.2.3 Linéarisation

L'algèbre linéaire apporte un lot important d'outils mathématiques. Ainsi la linéarisation est une étape permettant de remplacer un problème se révélant difficile par un problème approché qui se révèle souvent plus simple.

Par exemple, le groupe d'homotopie $\pi(X)$ qui encapsule le groupe des lacets à déformation prêt est un invariant puissant mais malheureusement difficile à calculer en général. L'approximation par linéarisation produit le groupe d'homologie $H_1(X)$ qui se révèle certes moins puissant mais dont le calcul est automatisable sur les espaces combinatoires.

La première étape de linéarisation consiste à construire des espaces vectoriels qui encode l'espace topologique. La seconde étape ajoute la notion de bord par l'introduction d'applications linéaires idoines.

Nous supposerons dorénavant que les espaces vectoriels sont définis sur le corps binaire $\mathbb{Z}_2$. Ainsi, il n'est plus nécessaire d'introduire l'orientation des simplexes et l'interprétation intuitive des résultats en sera facilitée. De plus, cette restriction n'a pas d'incidence sur les applications qui seront présentés par la suite. Toutefois, si cette orientation est nécessaire à une application, l'extension des travaux présentés ici à des corps différents de $\mathbb{Z}_2$ comme $\mathbb{R}$ reste immédiate.

Pour chaque dimension d, un espace vectoriel C_d est formé par l'ensemble des combinaisons linéaires formelles de simplexes de dimension d. Les vecteurs de C_d sont appelés les d-chaînes. Nous ne ferons par la suite pas de différence de notation entre un simplexe σ et la chaîne de C_d correspondant. La dimension de C_d est le nombre de d-simplexes de l'espace topologique.

Sur l'exemple de la figure 5.1, C_2 est un espace de dimension 2 dont les vecteurs sont abc, bcd, $abc+bcd$ et le vecteur nul. L'espace C_1 est plus grand car sa dimension atteint 5 et sa base naturelle est formée des vecteurs ab, ac, bc, bd et cd. Quant à C_0, sa base est donnée par les 4 sommets. Les autres espaces C_d sont tous nuls car il n'y a pas de simplexe de dimension supérieure à 2.

Les d-chaînes sont des vecteurs qui sont associés à des sous-espaces topologiques de dimension d. Plus précisément, l'utilisation de $\mathbb{Z}_2$ permet de rendre équivalents les vecteurs de C_d et les ensembles de d-simplexes. Par exemple, la 1-chaîne $ab + bd$ est le chemin passant par les deux arêtes ab et bd.

Il reste maintenant à relier ces espaces vectoriels entre eux par la notion de bord. Il est naturel de considérer que le bord de l'arête ab est donné par les sommets a et b. Formellement, $\partial_1 ab = a+b$. L'opérateur de bord ∂_1 est donc défini pour toutes les arêtes de cette façon et par linéarité, il est défini pour tous les vecteurs. L'intuition est sauvegardée, en effet $\partial_1(ab+bc) = \partial_1 ab + \partial_1 bc = a+b+b+c = a+c$, autrement dit, le bord du chemin passant par ab et bc est a et c.

Pour les autres dimensions, l'opérateur de bord est défini de la même façon : par exemple $\partial_2 abc = ab + bc + ac$ et $\partial_2 bcd = bc + cd + bd$. Le bord de la surface est alors fourni par la

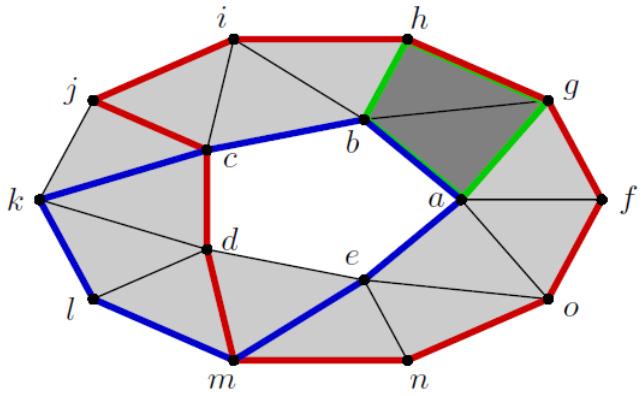


FIGURE 5.3 – Différents types de cycles : le cycle bleu est pertinent et génère le groupe d'homologie H_1 ; le cycle vert est sans importance parce qu'il n'entoure aucun trou ; le cycle rouge est équivalent au cycle bleu puisqu'il est possible de le modifier en le cycle bleu tout en restant dans l'anneau.

somme des deux qui est $\partial_2(abc+bcd) = ab+bd+cd+ac$. Bien sûr, les sommets n'ont pas de bord et par conséquent $\partial_0 = 0$. Souvent la dimension est implicitement donnée par le contexte et les opérateurs de bord sont tous notés ∂ sans indice.

L'ensemble des espaces vectoriels et des opérateurs de bord sont résumés dans l'équation suivante qui est le pendant algébrique de l'espace topologique combinatoire. Il s'appelle le complexe de chaînes associé et se note $C_\bullet$.

$$0 \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0 \tag{5.1}$$

Remarquons que cette linéarisation n'est pas liée aux complexes simpliciaux abstraits. Elle reste parfaitement correcte pour d'autres constructions comme les complexes cellulaires réguliers ou les complexes cubiques [KMM03].

5.2.4 Groupes d'homologie

Le groupe d'homologie $H_k(X)$ capture la notion de « trous » de dimension k de l'espace topologique X. Algébriquement, il s'agit de trouver les k-cycles qui entourent ces trous. Une équivalence entre ces cycles est à ajouter pour ne pas compter deux fois le même trou si deux cycles équivalents l'entourent. Dans un premier temps, les cycles σ sont les chaînes qui n'ont pas de bords c'est-à-dire $\partial\sigma = 0$. En suivant l'exemple de la figure 5.3, la chaîne $ab+bc$ n'est pas un cycle car son bord qui vaut $a+c$ n'est pas nul alors que la chaîne $ab+bh+hg+ag$ mise en exergue est un cycle car de bord nul.

Cependant, deux cycles peuvent entourés le même trou comme, par exemple, les cycles bleu et rouge sur la figure 5.3, en effet, il est possible de modifier continument le premier cycle en le second. De même, le cycle vert n'entoure pas de trou et est équivalent à un cycle

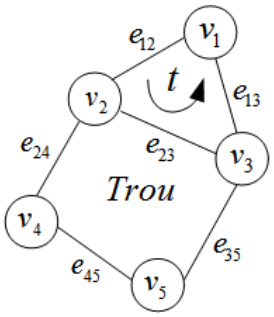


FIGURE 5.4 – Complexe simplicial.

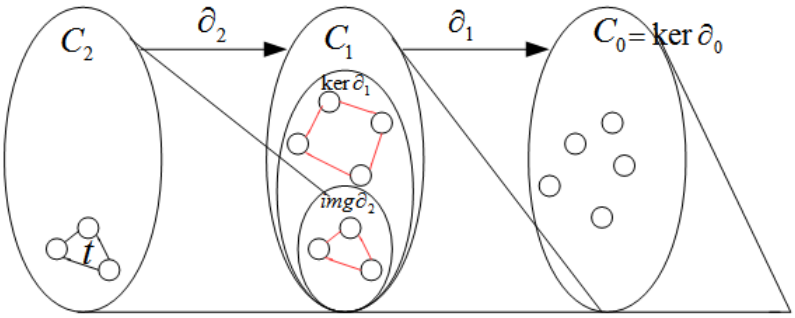


FIGURE 5.5 – Complexe de chaîne.

nul car il peut être réduit continument à un point. Intuitivement l'équivalence entre deux k-cycles σ et τ revient à montrer qu'il existe une chaîne de dimension $k+1$ dans laquelle les mouvements de déplacement de σ vers τ puissent s'effectuer. Comme, algébriquement, une telle transformation se traduit par une somme, la relation d'équivalence s'écrit : σ et τ sont deux k-cycles équivalents s'il existe une $k+1$-chaîne v telle que $\sigma = \tau + \partial v$.

Le groupe d'homologie $H_k(C_\bullet)$ est l'espace vectoriel de ces classes d'équivalence, formellement :

$$H_k(C_\bullet) = \frac{\ker \partial_k}{\operatorname{im} \partial_{k+1}} \tag{5.2}$$

Dans cette définition, $\ker \partial$ est l'espace des cycles, $\operatorname{im} \partial$ est l'espace des bords et le quotient est possible car $\partial_{k+1}\partial_k = 0$ et par conséquent $\operatorname{im} \partial_{k+1} \subset \ker \partial_k$.

La dimension de H_0 retourne le nombre de composantes connexes de l'espace, celle de H_1 retourne le nombre de « trous », celle de H_2, le nombre de « vides », etc. Ces dimensions sont appelées les nombres de Betti $\beta_0, ..., \beta_2$, et ils sont calculés comme suit :

$$\beta_k = \dim H_k = \dim \ker \partial_k - \dim \operatorname{im} \partial_{k+1} \tag{5.3}$$

Exemple pour le calcul des matrices de bord ∂_1 et ∂_2 :

La figure 5.4 présente un exemple d'un complexe simplicial composé de 5 sommets, 6 arêtes et 1 triangle. La figure 5.5 donne un aperçu de la relation entre les différents espaces vectoriels concernés par l'homologie. Les matrices de bord ∂_1 et ∂_2 sont calculées comme suit :

$$\begin{aligned}
\partial_1 e_{12} &= v_2 - v_1, \\
\partial_1 e_{13} &= v_3 - v_1, \\
\partial_1 e_{23} &= v_3 - v_2, \\
\partial_1 e_{24} &= v_4 - v_2, \\
\partial_1 e_{35} &= v_5 - v_3, \\
\partial_1 e_{45} &= v_5 - v_4,
\end{aligned}$$

Sous forme matricielle, ces relations s'écrivent

$$\partial_1 = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{12} & e_{13} & e_{23} & e_{24} & e_{35} & e_{45} \end{array} \\ \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{5.4}$$

De même pour le triangle t, nous avons

$$\partial_2 t = e_{12} - e_{13} + e_{23},$$

Soit matriciellement

$$\partial_2 = \begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{23} \\ e_{24} \\ e_{35} \\ e_{45} \end{array} \begin{array}{c} \begin{array}{c} t \end{array} \\ \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \tag{5.5}$$

D'après l'équation (5.2), nous pouvons calculer les deux groupes d'homologie H_0 et H_1,

$$H_0 = \frac{\ker \partial_0}{\operatorname{im} \partial_1}, \qquad H_1 = \frac{\ker \partial_1}{\operatorname{im} \partial_2}. \tag{5.6}$$

Comme tous les sommets de la figure 5.4 sont atteignables à partir de v_1 en partant d'une chaîne, le groupe H_0 est un espace vectoriel de dimension 1, c'est-à-dire qu'il n'y a qu'une seule composante connexe. Nous pouvons aussi calculer la dimension de H_0 ou le nombre de Betti β_0 par l'équation (5.3) : $\beta_0 = \dim H_0 = \dim \ker \partial_0 - \dim \operatorname{im} \partial_1 = 5 - 4 = 1$.

La dimension de H_1 comme on a vu précédemment permet de déterminer le nombre de trous dans le réseau, le nombre de Betti $\beta_1 = \dim H_1 = \dim \ker \partial_1 - \dim \operatorname{im} \partial_2 = 2 - 1 = 1$. Ainsi, nous remarquons qu'il existe un seul trou dans le réseau puisque la dimension de H_1 vaut 1. La classe d'équivalence de la chaîne $e_{23} + e_{35} + e_{45} + e_{24}$ en forme une base.

5.3 Laplacien, harmonique et détection de trous

5.3.1 Décomposition de Hodge

Les groupes d'homologie H_k ont précédemment été définis en utilisant un corps finis notamment binaire, c'est-à-dire en considérant l'espace des chaînes comme un espace vectoriel sur ce corps. Cette approche permet de rendre les algorithmes efficaces et exacts car alors aucun problème de précision ne vient s'immiscer dans les calculs numériques. Le corps binaire ayant de plus l'intérêt de rendre trivial l'orientation des chaînes.

Bien qu'attrayants pour les calculs exacts, ces espaces vectoriels sur des corps finis ne permettent pas d'y définir une notion de norme associée à un produit scalaire. Or cette notion permet de comparer les vecteurs entre eux aussi bien sur leur direction que sur leur taille.

Par la suite, les espaces C_k seront dorénavant des espaces vectoriels réels munis d'un produit scalaire défini positif noté $\langle u, v\rangle$ dont la norme est notée $\|u\| = \langle u, u\rangle$. À un opérateur linéaire A, est associé son adjoint A^* qui vérifie $\langle Ax, y\rangle = \langle x, A^*y\rangle$.

Par soucis de simplification mais sans perte de généralité excessive, nous prendrons par la suite le produit scalaire qui rend orthonormale la base de C_k composée des vecteurs associés aux k-simplexes. Ainsi la matrice associée à l'adjoint d'un opérateur s'obtient par la transposée de la matrice de l'opérateur.

Un élément du groupe d'homologie H_k est un vecteur qui est une classe d'équivalence de l'espace $\ker \partial_k$. En effet, comme

$$H_k = \frac{\ker \partial_k}{\operatorname{im} \partial_{k+1}}, \tag{5.7}$$

les éléments de H_k sont sous la forme

$$[c] = c + \operatorname{im} \partial_{k+1}, \tag{5.8}$$

où $[c]$ dénote la classe d'équivalence de $\ker \partial_k$ à laquelle appartient la chaîne $[c]$. Le choix du représentant c de la classe $[c]$ n'est pas canonique car n'importe quel vecteur de $[c]$ joue le même rôle, autrement dit, $[c + \partial d] = [c]$ pour $d \in C_{k+1}$.

Grâce à l'introduction du produit scalaire, il devient possible de privilégier un représentant de $[c]$. En choisissant le vecteur c_H de $c + \operatorname{im} \partial_{k+1}$ de norme minimale

$$c_H = \underset{c_H \in c + \operatorname{im} \partial_{k+1}}{\arg\min} \|c_H\| \tag{5.9}$$

De manière équivalente, c_H est le vecteur de $[c]$ orthogonal à tous les vecteurs de $\operatorname{im} \partial_{k+1}$, c'est-à-dire $c_H \in (\operatorname{im} \partial_{k+1})^\perp$ qui se réécrit $c_H \in \ker \partial^*_{k+1}$ en utilisant l'adjoint de ∂_{k+1}. La situation est schématisée par la figure 5.6.

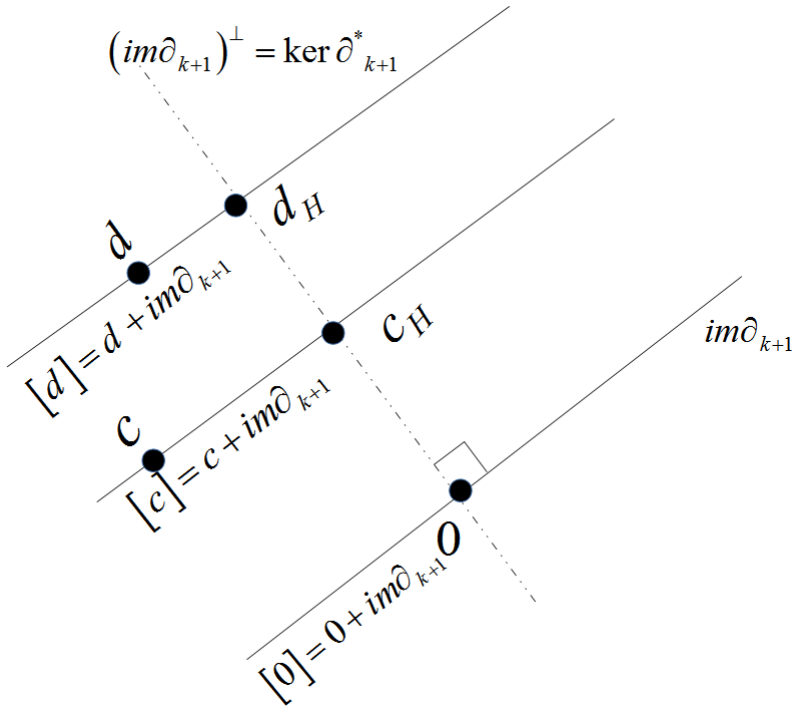


FIGURE 5.6 – Représentants harmoniques des classes de H_k.

En résumé, le groupe d'homologie H_k est un ensemble de classes d'équivalence $[c]$ qui peuvent chacune être représentée par une chaîne privilégiée c_H dite harmonique [Lim15] orthogonale à $\operatorname{im} \partial_{k+1}$, ou de façon équivalente dans $\ker \partial^*_{k+1}$. Nous avons donc

$$\begin{aligned} H_k &\cong \ker \partial_k \cap (\operatorname{im} \partial_{k+1})^{\perp} \\ &= ker\partial_k \cap ker\partial^*_{k+1} \end{aligned} \tag{5.10}$$

Notons maintenant

$$\mathbf{L}_k = \partial^*_k \, \partial_k + \partial_{k+1} \, \partial^*_{k+1} \tag{5.11}$$

l'endomorphisme de C_k appelé Laplacien. Remarquons que $\mathbf{L}_0$ est le Laplacien défini au chapitre 3 car $\partial_0 = 0$ et $\partial_1 = \mathbf{M}$. Les chaînes harmoniques forment l'espace vectoriel $\ker \partial_k \cap \ker \partial^*_{k+1}$. Or tous les éléments de cet espace sont aussi dans $\ker \mathbf{L}_k$ et inversement, d'où le nom de chaînes harmoniques car solutions de l'équation de Laplace $\mathbf{L}_k \, x = 0$.

Finalement, nous pouvons compléter les égalités ou les isomorphismes entre les espaces de (5.10) par

$$H_k = \frac{\ker \partial_k}{(\operatorname{im} \partial_{k+1})^{\perp}} \cong \ker \partial_k \cap (\operatorname{im} \partial_{k+1})^{\perp} = \ker \partial_k \cap \ker \partial^*_{k+1} \cong \ker \mathbf{L}_k. \tag{5.12}$$

Le supplémentaire orthogonal $(\ker \partial_k)^{\perp}$ de $\ker \partial_k$ permet de décomposer en somme directe l'espace C_k des chaînes : $C_k = \ker \partial_k \oplus (\ker \partial_k)^{\perp}$. Comme H_k est le quotient de $\ker \partial_k$ par

$\operatorname{im}\partial_{k+1}$, il est possible d'écrire la décomposition de $\ker\partial_k \cong H_k + \operatorname{im}\partial_{k+1}$. La relation (5.12) précédente permet d'exprimer cette décomposition en la somme directe $\ker\partial_k = \ker\mathbf{L}_k \oplus \operatorname{im}\partial_{k+1}$.

Au final, toutes les chaînes peuvent se décomposer en trois vecteurs orthogonaux deux à deux selon le schéma

$$C_k = \overbrace{\operatorname{im}\partial^*_{k+1} \oplus \underbrace{\ker\mathbf{L}_k}_{\ker\partial_k}}^{\ker\partial^*_{k+1}} \oplus \operatorname{im}\partial_{k+1}. \tag{5.13}$$

qui s'appelle la décomposition de Hodge dans laquelle apparaît le groupe d'homologie H_k sous la forme des chaînes harmoniques solution de $\mathbf{L}_k\, x = 0$.

Matriciellement, le Laplacien d'un complexe de cliques d'un graphe s'exprime sous la forme [ME06]

$$\mathbf{L}_k = \mathbf{D}_u - \mathbf{A}_u + (k+1)\mathbf{I} + \mathbf{A}_l k > 0 \tag{5.14}$$

où $\mathbf{I}$ est la matrice identité, $\mathbf{D}_u$ représente la matrice des degrés supérieure, $\mathbf{A}_l$ est la matrice d'adjacence inférieure entre les k-simplexes et $\mathbf{A}_u$ est la matrice d'adjacence supérieure. Pour $k = 0$, la matrice Laplacienne est équivalente à la matrice Laplacienne étudiée dans le chapitre 3 : $\mathbf{L}_0 = \mathbf{D} - \mathbf{A}_u$.

L'équation (5.14) implique que, comme dans le cas du Laplacien d'un graphe, la i-ème ligne de $\mathbf{L}_k$ dépend que des interactions locales entre le i-ème k-simplexe et ses simplexes adjacents supérieurs et inférieurs.

5.3.2 Exemples de chaînes harmoniques

Nous donnons maintenant un exemple simple pour visualiser une chaîne harmonique. Considérons le complexe simplicial X de l'exemple de la figure 5.4. À partir des opérateurs de bord ∂_1 et ∂_2 de l'exemple 5.4, nous calculons ses matrices Laplaciennes $\mathbf{L}_0$ et $\mathbf{L}_1$:

$$\mathbf{L}_1 = \begin{pmatrix} 3 & 0 & 0 & -1 & 0 & 0 \\ 0 & 3 & 0 & 0 & -1 & 0 \\ 0 & 0 & 3 & 1 & -1 & 0 \\ -1 & 0 & 1 & 2 & 0 & -1 \\ 0 & -1 & -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & -1 & 1 & 2 \end{pmatrix}, \qquad \mathbf{L}_0 = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}.$$

Dans notre exemple de la figure 5.7, une chaîne harmonique, *i.e.*, un vecteur du noyau de $\mathbf{L}_1$, est donnée par : $[0.1741, -0.1741, -0.3482, 0.5222, -0.5222, 0.5222]'$. Nous remarquons à travers la figure 5.7 que la somme des poids associés au chemin fermé $e_{12} - e_{13} + e_{24} - e_{35} + e_{45}$ et $e_{23} + e_{24} - e_{35} + e_{45}$ vaut 1.9118 chacun, indiquant la présence d'un trou à l'intérieur, alors que cette somme le long de $e_{12} - e_{13} + e_{24}$ vaut 0, indiquant l'absence d'un trou. Un vecteur du noyau de la matrice Laplacienne $\mathbf{L}_0$ est donné

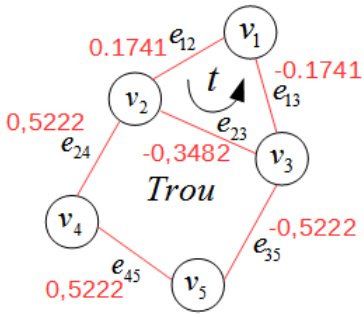


FIGURE 5.7 – Complexe de chaîne avec son vecteur dans l'espace nul de $\mathbf{L}_1$.

par $[0.4472, 0.4472, 0.4472, 0.4472, 0.4472]'$. La dimension du noyau de $\mathbf{L}_0$ qui est l'espace propre associé à la valeur propre 0, donne directement le nombre de composantes connexes, dans cet exemple, elle vaut 1 puisqu'il existe une seule composante connexe dans le graphe (voir chapitre 3 pour plus de précisions).

Malheureusement cette approche pour la détection de trou est purement centralisée : elle nécessite une connaissance préalable du spectre du Laplacien combinatoire $\mathbf{L}_k$. Dans la suite des travaux, nous présenterons des résultats de détection distribuée en utilisant par exemple l'algorithme de consensus de moyenne.

Afin de visualiser les chaînes harmoniques pour des situations plus réalistes : nous avons construit un réseau aléatoire dans lequel des trous de couverture ont été percés. Ce réseau est représenté par la vignette en haut à gauche de la figure 5.8.

Les capteurs, ou les sommets, sont placés aléatoirement dans un rectangle deux fois plus large que haut et les capteurs trop près les uns des autres sont éliminés pour rendre la lecture plus aisée. Les sommets sont alors reliés les uns aux autres si leur distance est inférieure à un seuil de connexion. Trois sommets connectés deux à deux forment un triangle du complexe simplicial approchant la couverture. Sur la vignette de la figure, la couverture est en gris. Nous remarquons la présence nette de deux trous conséquents et d'un plus petit en bas à droite.

À partir de ce réseau et du complexe simpliciale de la couverture, le Laplacien $\mathbf{L}_1$ est calculé selon (5.11). Une base orthogonale de l'espace des chaînes harmoniques est ensuite extraite par la décomposition en valeur singulière du Laplacien. Comme il y a trois trous, l'espace $\ker \mathbf{L}_1$ est de dimension trois. Les trois chaînes de la base sont représentées sur les trois autres vignettes de la figure 5.8 avec un code couleur et une épaisseur de trait qui indiquent l'amplitude du coefficient de la chaîne associée à chaque arête.

Nous remarquons dans un premier temps que la dimension de $\ker \mathbf{L}_1$ fournit bien le nombre de trous de couverture. Ensuite, nous remarquons que les chaînes harmoniques ont tendance à avoir de forts coefficients localisés sur le bord des trous. Visuellement sur la figure 5.8, la somme des valeurs d'une chaîne harmonique le long d'un chemin fermé détermine bien si un trou est entouré ou non en la comparant à 0.

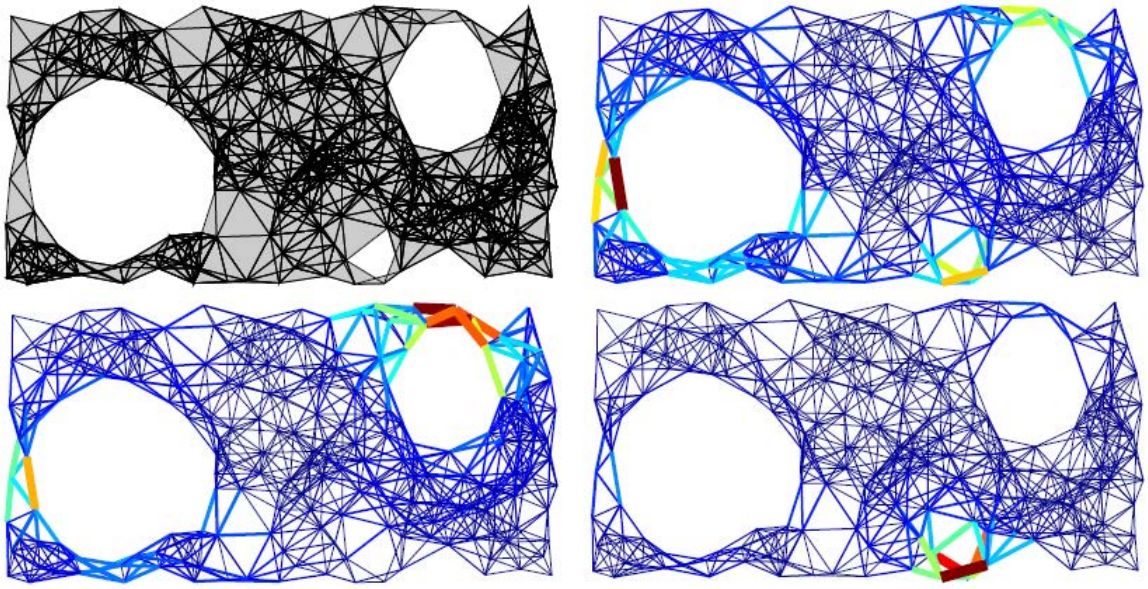


FIGURE 5.8 – Exemple de chaînes harmoniques.

Les résultats obtenus montrent toutefois que les chaînes harmoniques ne localisent pas les trous précisément et directement. En effet, hormis le dernier vecteur de la base harmonique en bas à droite de la figure 5.8, les chaînes harmoniques ont des coefficients non nuls autour de chaque trou.

5.3.3 Première application à la détection des trous

Le lien entre les chaînes harmoniques et le groupe H_1 permet aux auteurs de [TSJ10], en se reposant sur les travaux [ME06], de proposer une détection distribuée des trous de couverture, c'est à dire des générateurs de H_1.

Plus précisément, une chaîne $x(0)$ de C_1 est aléatoirement choisie, puis elle est sujette à la dynamique imposée par l'équation différentielle

$$\frac{dx}{dt} = -\mathbf{L}_1\, x(t), \tag{5.15}$$

dont les points fixes sont les chaînes x solutions de $\mathbf{L}_1\, x = 0$. Ainsi la chaîne $x(t)$ converge normalement [ME06] vers une chaîne harmonique, c'est-à-dire un représentant d'un générateur de H_1.

La forme distribuée de ce système est simple à mettre en œuvre car le Laplacien $\mathbf{L}_1$ d'une arête donnée ne fait intervenir que les arêtes incidentes.

En simulant ce système dynamique avec plusieurs chaînes aléatoires initiales, il est possible de récupérer une base de $\ker \mathbf{L}_1$ et donc de H_1 et par conséquent de repérer les trous de couverture selon les résultats de de Silva et Ghrist [dSG07].

Toutefois, comme nous l'avons vu dans l'exemple précédent de la figure 5.8, les chaînes harmoniques obtenues ne forment pas nécessairement une base praticable pour localiser précisément les trous. Pour y remédier, il est proposé dans [TSJ10] de modifier les chaînes harmoniques obtenues par un algorithme de sous-gradient distribué pour trouver une chaîne équivalente à la chaîne harmonique dans H_k de norme minimale.

5.4 Nouvel algorithme distribué de détection de trous

Simuler le système dynamique (5.15) puis améliorer le résultat pour le rendre lisible rend la mise en œuvre délicate car il faut synchroniser les deux étapes correctement et gérer les modifications de la topologie au cours du temps en recommençant l'ensemble du processus.

Pour y remédier, dans cette section, nous proposons un nouvel algorithme de détection de trou dans un RCSF. Nous recherchons une fonction harmonique du réseau par la recherche d'une chaîne dans l'intersection de $\ker \partial_2^*$ et de $\ker \partial_1$. Cet algorithme est itératif et tente de modifier la chaîne pour rendre le rotationnel de chaque triangle et la divergence à chaque sommet nuls. Nous comparerons cette approche avec la même méthode mais en recherchant une chaîne dans $\ker \partial_1$.

5.4.1 Détection distribuée des trous de couverture

Supposons qu'une chaîne harmonique, solution de $\mathbf{L}_1\, x = 0$, soit trouvée et répartie sur le réseau. Pour savoir si le chemin clos p dans le réseau passant par les arêtes $e_1, e_2, \ldots, e_n$ entoure un trou ou non il suffit de calculer la circulation de x le long de p : $x(p) = x(e_1) + x(e_2) + \cdots + x(e_n)$ où $x(e_i)$ est le coefficient pour l'arête e_i de la chaîne x. Si cette circulation $x(p) \neq 0$ alors un trou est détecté !

Si la chaîne harmonique x est choisie aléatoirement dans $\ker \mathbf{L}_1$, la probabilité de non détection d'un trou est faible. Elle peut encore être rendue plus faible en testant la circulation le long du chemin fermé non pas avec une mais avec plusieurs chaînes harmoniques en parallèle.

Une marche aléatoire dans le réseau afin de tester la présence de trou par cumul des pondérations associées à une chaîne harmonique est donnée dans l'algorithme 5.1. Cette algorithme teste si le chemin courant peut être fermé et si c'est le cas détecte la présence potentielle d'un trou grâce à la circulation le long de ce chemin.

Cet algorithme nécessiterait des améliorations certaines, mais il se présente sous une forme très simple, peu gourmande en calcul et complètement distribuée. Naturellement, plusieurs jetons peuvent être initialisés en même temps à des endroits différents du réseau.

Une première amélioration serait de sauvegarder dans le jeton la liste des sommets déjà visités avec les cumuls pour chaque. À chaque fois qu'un sommet voisin est déjà visité, le

Algorithm 5.1: Détection de trou de couverture par marche aléatoire.

Input : Un réseau avec des arêtes pondérées selon une chaîne harmonique ω
Output: Un chemin localisant un trou

```
1  Initialiser un jeton à 0
2  while vrai do
3  |  u ← sommet courant du jeton
4  |  x ← valeur du jeton
5  |  while le sommet initial v est voisin do
6  |  |  if x + ω(uv) ≠ 0 then
7  |  |  |  return trou détecté
8  |  |  end
9  |  end
10 |  if il existe des sommets voisins non visités then
11 |  |  Choisir aléatoirement un voisin v
12 |  |  Déplacer le jeton en v avec la valeur x + ω(uv)
13 |  else
14 |  |  Choisir aléatoirement un voisin v
15 |  |  Déplacer le jeton en v avec la valeur nulle
16 |  |  Considérer v comme le nouveau sommet initial du jeton
17 |  end
18 end
```

même test que cela de la ligne 5 pourrait être effectué pour la détection. Il y aurait alors plusieurs possibilités de fermer le chemin parcouru par le jeton.

Ensuite, les choix aléatoires des lignes 11 et 14 mériteraient une étude très approfondie pour tester le plus de chemins fermés entourant un trou. Utiliser les poids des arêtes incidentes serait sûrement très profitable de ce point de vue.

Enfin, il pourrait être envisageable de biaiser la marche du jeton selon un champ de vecteurs discrets qui serait construit lors de la construction de la chaîne harmonique.

De nombreuses autres pistes d'amélioration sont envisageables ici et pourraient profiter de la très large littérature sur les marches aléatoires et sur les processus de Markov sur les graphes.

5.4.2 Construction d'une chaîne harmonique

L'idée de la marche aléatoire nécessite toutefois l'obtention d'une chaîne harmonique. Nous proposons un algorithme distribué pour ce faire.

La première application de la décomposition de Hodge à la section 5.3.3 repose sur l'identification de H_1 et du noyau de $\mathbf{L}_1$ au travers le système dynamique (5.15). Nous proposons une approche différente utilisant le lien entre H_1 et $\ker \partial_1 \cap \ker \partial_2^*$.

Une chaîne est représentée par des pondérations le long des arêtes du complexe simplicial qui sont initialement choisies aléatoirement. Cette chaîne est ensuite modifiée localement pour la rendre plus proche soit de $\ker \partial_1$ soit de $\ker \partial_2^*$.

Notons tout d'abord dans un premier temps $\mathrm{div}(v)$, la divergence en v des poids, c'est-à-dire la somme des poids des arêtes incidentes au sommet v en prenant en compte l'orientation des arêtes. Par exemple, $\mathrm{div}(v) = a + b - c + d - e$ pour le sommet v de la figure 5.9. De même notons $\mathrm{rot}(t)$ le rotationnel du triangle t qui est la circulation des poids le long du bord de t en prenant en compte l'orientation. Sur la droite de la même figure, nous avons $\mathrm{rot}(t) = a - b - c$ pour le triangle t.

Il est rapide de vérifier que les chaînes de $\ker \partial_1$ ont une divergence en tout sommet nulle et que les éléments de $\ker \partial_2^*$ ont un rotationnel nul pour chaque triangle. Chercher une chaîne harmonique de $\ker \mathbf{L}_1$ revient donc à trouver une chaîne de divergence et de rotationnel nuls.

Remarquons que la divergence $\mathrm{div}(v)$ peut s'écrire sous la forme $\langle a, x \rangle$ où a est un vecteur très creux car ses coefficients non nuls concernent uniquement les arêtes incidentes à v. Le rotationnel $\mathrm{rot}(t)$ s'écrit aussi sous cette forme avec un vecteur a de coefficients non nuls pour les arêtes de t. Autrement dit, les calculs de divergences et de rotationnels peuvent être efficacement programmés pour être distribués.

Prenons un sommet v aléatoirement. Si $\mathrm{div}(v)$ est nulle, il n'y a rien à faire, mais sinon, il est possible de rendre cette divergence nulle en modifiant le poids des arêtes incidentes

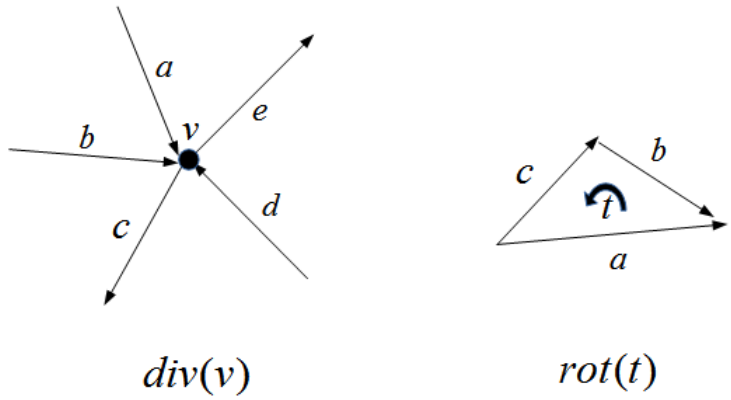


FIGURE 5.9 – Divergence et rotationnel.

à v. En effet, en écrivant $\mathrm{div}(v) = \langle a, x \rangle$, il suffit de modifier x pour le rendre orthogonal à a par la projection sur $a^{\perp}$, encore appelée la réjection,

$$x \leftarrow x - \frac{\langle a, x \rangle}{\|a\|} a. \tag{5.16}$$

La même opération peut se réaliser en prenant un triangle t aléatoirement et en rendant le rotationnel $\mathrm{rot}(t)$ nul.

L'algorithme 5.2 reprend ces idées pour les formaliser. Le choix entre annuler un rotationnel ou une divergence se fait aussi aléatoirement : la probabilité de l'annulation d'un rotationnel vaut p_{rot}, constante du système.

L'algorithme 5.2 ajoute toutefois un nouveau contrôle sur la norme de la chaîne x pour éviter sa dégénérescence. En effet, la norme de la partie du vecteur impliquée dans la divergence ou le rotationnel est conservée autant que possible. Ainsi, la norme de la chaîne est conservée au fil des projections.

Éviter cette dégénérescence n'est pas sans conséquence. En effet, s'il n'y a pas de trou dans le complexe simplicial, la seule chaîne harmonique est la chaîne nulle. Or celle-ci devient inatteignable par notre algorithme à cause de la renormalisation. Pour résoudre ce cas, il est parfois judicieux d'ajouter des trous dans le complexe qui sont purement artificiels et qui serviront à concentrer « l'énergie » de la chaîne harmonique sur des arêtes qui ne serviront pas à la détection des trous par la marche aléatoire. Ces trous et ces arêtes artificiels libèrent la contrainte de dégénérescence.

Nous n'avons pas étudié la qualité de la convergence de notre algorithme par manque de temps. En effet, elle n'est pas si simple à montrer car en annulant la divergence en un sommet v, respectivement le rotationnel en un triangle t, il est fort probable de rendre non nulle celle d'un sommet voisin, respectivement celui d'un triangle adjacent. Toutefois, les simulations n'ont jamais montré de cas de divergence profonde de l'algorithme.

Algorithm 5.2: Obtention d'une chaîne harmonique par projection.

Input : Un complexe simplicial
Output: Une chaîne harmonique

Initialiser une chaîne de C_1 aléatoirement
while *vrai* **do**
  Choisir nombre aléatoire u uniformément entre 0 et 1
  $x \leftarrow$ le vecteur des poids courants des arêtes du réseau
  if $u < p_{\text{rot}}$ **then**
    Choisir un triangle t aléatoirement
    Écrire $\text{rot}(t)$ sous la forme $\langle a, x \rangle$
  else
    Choisir un sommet v aléatoirement
    Écrire $\text{div}(v)$ sous la forme $\langle a, x \rangle$
  end
  $x_a \leftarrow$ sous-vecteur de x pour lequel les coefficients de a sont non nuls
  $n_x \leftarrow \|x_a\|$
  $x_a \leftarrow x_a - \dfrac{\langle a, x \rangle}{\|a\|}\, a$
  if $\|x_a\| > 0$ **then**
    $x_a \leftarrow \dfrac{n_x}{\|x_a\|} x_a$
  end
end

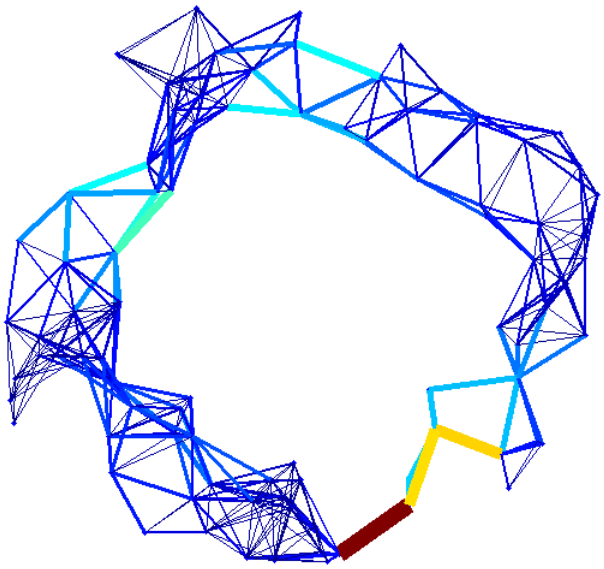


FIGURE 5.10 – Réseau : densité=40 r=0.3 r_a=0.5.

5.4.3 Simulation de construction de chaînes harmoniques

Nous avons généré des réseaux de capteurs uniformément répartis dans un anneau de rayon externe unité et de rayon interne r_a pour construire artificiellement un trou de couverture. À ce paramètre s'ajoutent la densité de capteurs, i.e. leur nombre par unité de surface et le rayon de connexion des capteurs r.

Nous avons exécuté notre algorithme 5.2 sur ces réseaux en faisant varier la probabilité p_{rot}. Nous avons aussi simulé l'algorithme en utilisant directement le Laplacien $\mathbf{L}_1$.

Les performances des algorithmes sont mesurées par la norme de $\mathbf{L}_1 x$. La convergence est atteinte lorsque la norme s'annule car alors x est une chaîne harmonique. Chaque simulation donne la représentation de la meilleure chaîne obtenue pour tout les p_{rot}.

La couleur et l'épaisseur de trait indiquent l'amplitude du coefficient de la chaîne associée à chaque arête, voir figure 5.10, et par la suite, nous pouvons détecter la meilleure chaîne.

La figure 5.10 représente un réseau avec une densité égale à 40, un rayon de connexion $r = 0.3$, et un rayon de l'anneau $r_a = 0.4$. Nous remarquons à travers la figure 5.11 que la convergence est rapide pour un $p_{rot} = 0.6$ et $p_{rot} = 0.7$. Cependant, la convergence avec le Laplacien $\mathbf{L}_1$ est faible mais plus rapide au début de la simulation. Ainsi, du point de vue résultats, la figure 5.10, nous montre que toute la chaîne se concentre sur quelques arêtes qui forment une coupe radiale de l'anneau.

Notons que la convergence sur la figure 5.11 se présente en 3 temps. Tout d'abord, une décroissance rapide au début de la simulation, ensuite la convergence atteint un palier, autrement dit, la convergence reste constante après un certain nombre d'itérations. Enfin, comme dernière étape, la décroissance reprend à nouveau pour devenir plus rapide à la

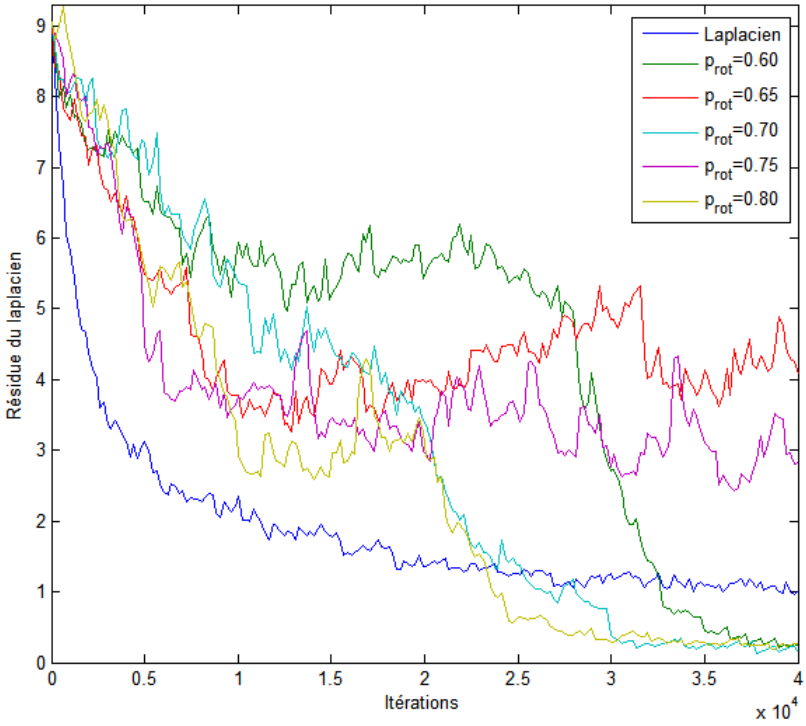


FIGURE 5.11 – Convergence : densité=40 r=0.3 r_a=0.5.

fin de la simulation.

Au regard des simulations de la figure 5.11, nous remarquons que l'effet de p_{rot} est peu compréhensible à première vue. Mais l'origine de ce phénomène semble être la méthode du choix aléatoire des triangles et des arêtes. L'étude de ce phénomène rentre dans le cadre de nos travaux futurs.

Dans la deuxième simulation de la figure 5.12, nous avons choisi un réseau plus connecté par rapport au précédent 5.10, en augmentant le rayon de connexion à $r = 0.5$ et $r_a = 0.6$. Nous remarquons ainsi, à partir de la figure 5.13 que la convergence est plus rapide et moins bruitée pour les p_{rot}. Par contre, la convergence avec le Laplacien $\mathbf{L}_1$ est lente et peut être incomplète.

Le résultat de la figure 5.12, nous montre que la concentration est sur deux arêtes en bas à gauche de la figure.

La connectivité du réseau sur la figure 5.14 est non uniforme ; il existe des zones très connectées et des zones faiblement connectées. Nous remarquons à travers la figure 5.15 que la convergence dépend fortement de p_{rot} mais tous convergent. Par contre, la convergence du Laplacien $\mathbf{L}_1$ est faible voire inexistante. La convergence se présente également en 3 temps avec une décroissance rapide au début de la simulation, ensuite un phénomène de stabilité se produit après un certain nombre d'itérations, et enfin une convergence rapide comme dernière étape.

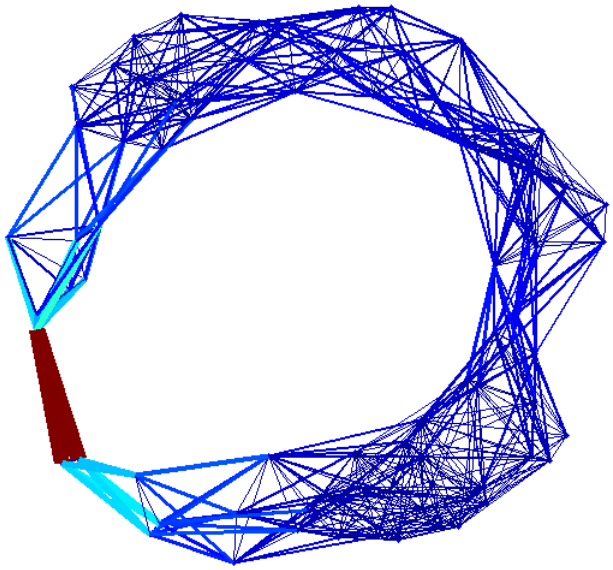


FIGURE 5.12 – Réseau : densité=40 r=0.5 r_a=0.6.

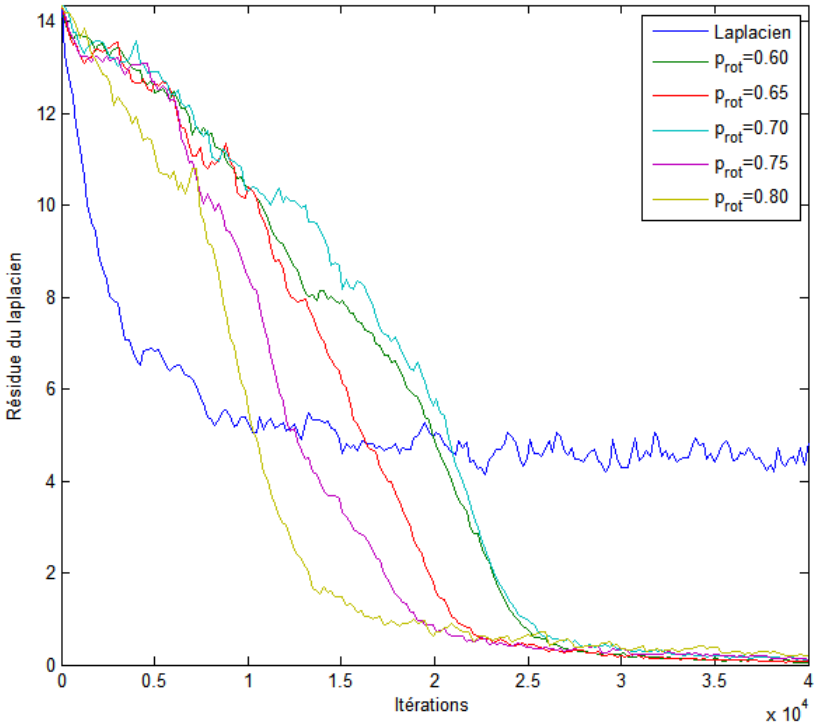


FIGURE 5.13 – Convergence : densité=40 r=0.5 r_a=0.6.

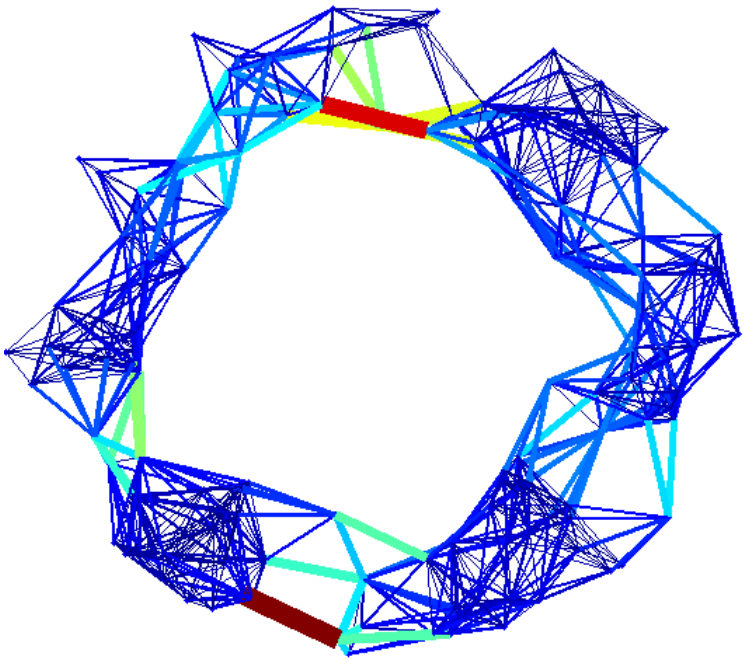


FIGURE 5.14 – Réseau : densité=60 r=0.3 r_a=0.5.

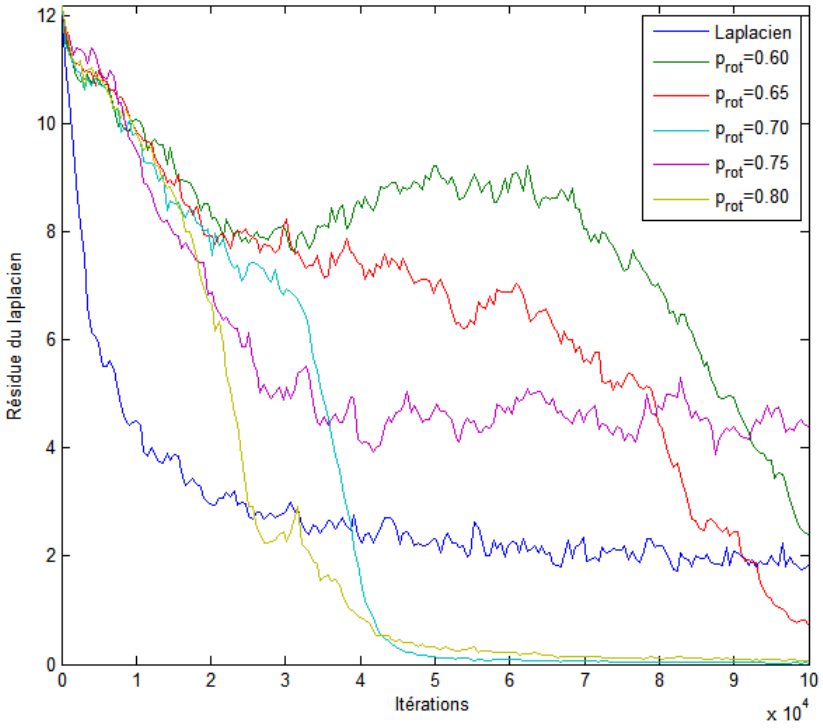


FIGURE 5.15 – Convergence : densité=60 r=0.3 r_a=0.5.

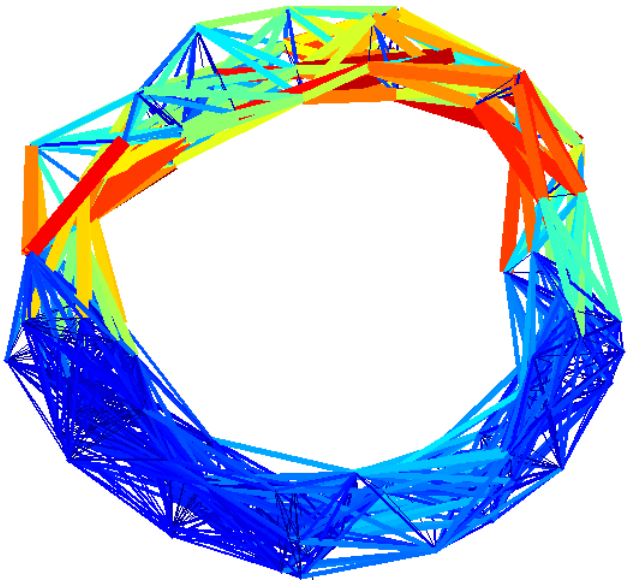


FIGURE 5.16 – Réseau : densité=60 r=0.5 r_a=0.6.

La figure 5.14 illustre le résultat de la troisième simulation, nous remarquons que la chaîne se concentre sur les arêtes proches des « petits trous » (en haut et en bas de la figure) ou pour le trou central (à gauche de la figure).

Le réseau de la figure 5.16 est fortement connecté, ce qui entraine une convergence rapide et moins bruitée que dans le cas précédent (voir figure 5.17) pour tout les p_{rot}. Même remarque que les autres simulations, la convergence se présente en 3 temps avec une décroissance rapide au début de la simulation, ensuite la convergence atteint un palier après un certain nombre d'itérations, et enfin une convergence beaucoup plus rapide à la fin de la simulation si nous la comparons avec les résultats des simulations précédentes.

La figure 5.18 montre une certaine dégénérescence de la chaîne obtenue par l'utilisation directe du Laplacien $\mathbf{L}_1$. Cette dernière reste limitée mais montre la fragilité potentielle de cette approche par rapport à une séparation d'utilisation de la divergence et du rotationnel.

5.5 Conclusion

En perspective, un algorithme utilisant le Laplacien, la divergence et le rotationnel permettrait sûrement d'améliorer la convergence tout en éliminant la faiblesse de chacun.

Dans ce chapitre, nous avons développé deux méthodes pour détecter les trous de couverture de manière distribuée. La première consiste à utiliser un système dynamique ou un consensus qui fait intervenir le Laplacien combinatoire $\mathbf{L}_1$ et la deuxième méthode consiste à rechercher une fonction harmonique du réseau par la recherche d'une chaîne

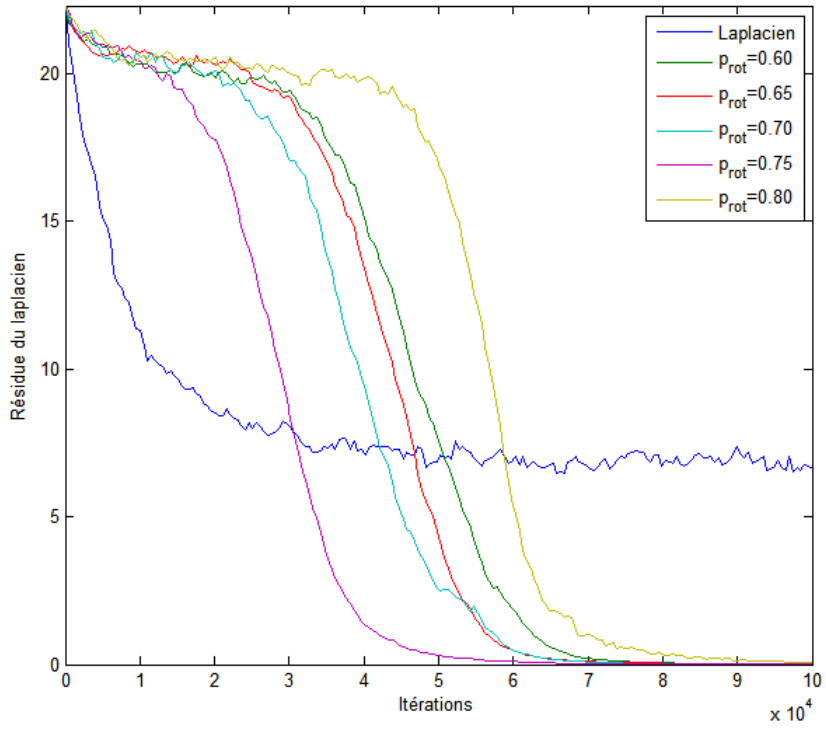


FIGURE 5.17 – Convergence : densité=60 r=0.5 r_a=0.6.

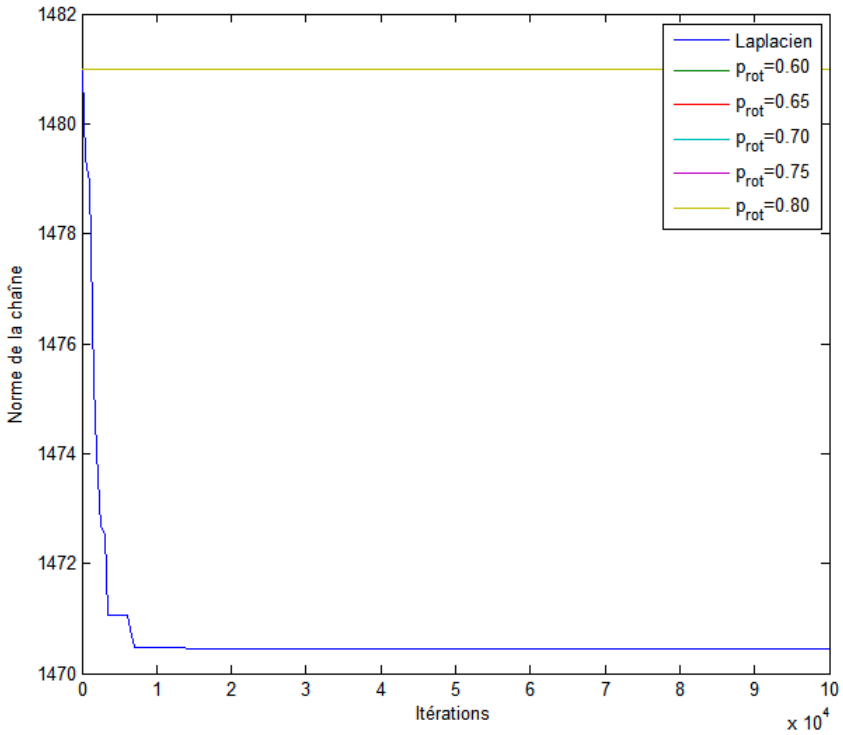


FIGURE 5.18 – Dégénérescence : densité=60 r=0.5 r_a=0.6.

dans l'intersection de $\ker \partial_2^*$ et de $\ker \partial_1$ pour ensuite détecter les trous de couverture par l'algorithme de marche aléatoire sur le graphe. Cependant, ce dernier profiterait des améliorations que nous développerons dans nos travaux futurs, tout en conservant ces forces : peu gourmand en calcul et une distribution de l'algorithme simple et efficace.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion générale

Dans la première partie de cette thèse, nous avons étudié le problème de consensus de moyenne distribué pour les réseaux de capteurs sans fil. Nous avons commencé par l'algorithme de consensus de moyenne dans le cas synchrone où tous les capteurs sont activés et échangent leurs information simultanément. Cette étude dans le modèle temporel synchrone, nous a permis d'avoir une bonne base théorique pour analyser les performances de l'algorithme sur des topologies déterministes et aléatoires.

Dans la deuxième partie du chapitre 4, nous avons particulièrement mis l'accent sur des algorithmes de bavardage asynchrone qui sont efficaces sur les graphes géométriques aléatoires, car ils modélisent les réseaux de capteurs sans fil.

La réussite de ces algorithmes de bavardage tient tout d'abord de leur simplicité. En effet l'algorithme de bavardage asynchrone est facile à mettre en œuvre, car il ne nécessite aucune infrastructure organisée. Les échanges entre capteurs se font de manière simple sans utilisation de protocoles de routage. Ensuite, un deuxième avantage important des algorithmes de bavardage asynchrone est leur évolutivité : chaque capteur dispose du même taux de bavardage, même avec le changement de taille du réseau.

En effet, grâce à cette étude sur ces algorithmes, nous avons proposé un nouveau type de bavardage dans le cas asynchrone appelé « Push-Sum » par diffusion. Il permet une convergence rapide grâce à la nature de diffusion de l'algorithme mais surtout à la construction de sa matrice de pondération qui représente les interactions entre capteurs.

Dans la deuxième partie de cette thèse, nous avons étudié un autre problème dans les réseaux de capteurs, il s'agit de la couverture. Nous avons proposé deux algorithmes qui permettent de détecter de manière distribuée les trous de couverture sans la nécessité de connaitre la position des capteurs. Le premier utilise le principe du consensus de moyenne pour chercher le vecteur qui appartient à l'espace nul de la matrice Laplacienne combinatoire [HGCG15]. Le deuxième algorithme permet la recherche d'une fonction harmonique d'un réseau par la recherche d'une chaîne dans l'intersection des différents opérateurs de bord. L'algorithme est itératif et tente de modifier la chaîne pour rendre le rotationnel de chaque triangle nul et la divergence à chaque sommet nulle. Ces méthodes distribuées se sont montrées efficaces par rapport aux méthodes centralisés où il est nécessaire d'avoir des informations sur le spectre du Laplacien combinatoire et par rapport à un algorithme distribué ne séparant pas la divergence et le rotationnel.

La réussite de cette détection distribuée est due à l'utilisation de la topologie algébrique, nous avons pu modéliser le réseau grâce à des entités d'ordres supérieurs à savoir les arêtes et les triangles. Les relations entre ces entités sont réalisables grâce à l'opérateur de bord décrit dans le chapitre 5. Avec des données fournies, il est aussi possible d'en calculer plusieurs invariants comme les groupes d'homologie.

6.2 Perspectives

6.2.1 Algorithmes distribués de consensus de moyenne

Les perspectives que nos travaux ouvrent sont nombreuses. Tout d'abord, nous proposons une étude approfondie des algorithmes de bavardage en les appliquant dans des environnements plus réalistes grâce au simulateur Castalia [Bou] en intégrant par exemple le bruit dans le canal de transmission.

Deuxième proposition qui s'avèrerait constructif et bénéfique est l'étude des algorithmes de consensus de moyenne dans un contexte dynamique. Autrement dit, nous pouvons tester notre algorithme de bavardage sur des capteurs mobiles, car jusqu'à maintenant notre travail s'est basé uniquement sur des capteurs statiques. Les outils développés par Chazelle [Cha12] et [Cha15] fourniraient par exemple un début d'analyse dynamique des algorithmes de consensus.

6.2.2 Détection et localisation des trous de couverture dans un RCSF

Les travaux réalisés durant cette thèse constituent un point de départ à toute une série de perspectives possibles. Nous proposons par exemple dans la suite de ce travail d'analyser la qualité de la convergence de notre algorithme de recherche d'une chaîne harmonique.

Nous pouvons aussi faire une comparaison entre les différents algorithmes de calcul d'une fonction harmonique et tirer profil de leurs performances.

Il serait également intéressant de faire une étude et une analyse détaillée de notre algorithme de détection de trous afin d'effectuer des améliorations sur ce dernier en profitant de la très large littérature sur les marches aléatoires.

À la lumière des résultats obtenus par notre algorithme, il serait intéressant de l'évaluer et de le développer pour des dimensions supérieurs, à savoir la détection de plusieurs trous dans un réseau. Sur ce dernier point, nous pouvons exploiter les résultats des auteurs de [GGB16] qui utilisent les structures d'ordre supérieurs.

La décomposition de Hodge mais sur corps finis serait un choix intéressant pour développer notre algorithme de détection de trous car il permettra une implémentation simple et une convergence sans problème. Mais une théorie entière est alors à construire.

Bibliographie

[Ada13] Henry Adams. *Evasion Paths in Mobile Sensor Networks*. PhD thesis, Stanford University, 2013. (Cité en page 18.)

[AF07] C. Adams and R. Franzosa. *Introduction to topology : Pure and applied. Prentice Hall*. 2007. (Cité en page 72.)

[Arm83] M.A. Armstrong. *Basic Topology*. Springer, 1983. (Cité en page 71.)

[ASSC02] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks : a survey. *Computer Networks*, 38(4) :393–422, 2002. (Cité en page 11.)

[AYSS09] Tuncer Can Aysal, Mehmet Ercan Yildiz, Anand D. Sarwate, and Anna Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57 :2748—-2761, 2009. (Cité en pages xix, 2, 34, 53, 54, 57, 62 et 64.)

[BBT+10] Florence Bénézit, Vincent Blondel, Patrick Thiran, John Tsitsiklis, and Martin Vetterli. Weighted gossip : Distributed averaging using non-doubly stochastic matrices. *IEEE International Symposium on Information Theory*, pages 1753–1757, 2010. (Cité en page 66.)

[BC05] Sergio Barbarossa and Francesco Celano. Self-organizing sensor networks designed as a population of mutually coupled oscillators. In *Proceedings IEEE Workshop on Signal Processing Advances in Wireless Communications*, 2005. (Cité en page 36.)

[BGPS05] Stephen Boyd, Arpita Ghosh, Salaji Prabhaka, and Devavrat Shah. Gossip algorithms : Design, analysis and applications. In *Proceedings of the IEEE Infocomm*, 2005. (Cité en pages xix, 51, 54, 57 et 61.)

[BGPS06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions On Information Theory*, 52(6) :2508–2530, 2006. (Cité en pages xix, 51, 52 et 61.)

[BH57] S. R. Broadbent and J. M. Hammersley. Percolation processes. i. crystals and mazes. In *Proceedings of the Cambridge Philosophical Society 53*, pages 629–641, 1957. (Cité en page 16.)

[BKT15] J.-D. Boissonat, C. S. Karthik, and S. Tavenas. Building efficient and compact data structures for simplicial complexes. 2015. (Cité en page 73.)

[BKX+06] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. *MobiHoc'06*, 2006. (Cité en page 18.)

[BMK04] Alex Brooks, Alexei Makarenko, and Tobias Kaupp. Implementation of an indoor active sensor network. In *Proceedings of the 9th International Symposium on Experimental Robotics Singapore*, 2004. (Cité en page 15.)

[Bén09] Florence Bénézit. *Distributed Average Consensus for Wireless Sensor Networks*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2009. (Cité en page 50.)

[Bol01] Béla Bollobás. *Random Graphs*. (Cambridge Studies in Advanced Mathematics) 2nd Edition, 2001. (Cité en page 29.)

[Bou] Athanassios Boulis. Castalia : A simulator for wireless sensor networks. `http://castalia.npc.nicta.com.au/.` (Cité en page 98.)

[Bro54] S. R. Broadbent. Discussion on symposium on monte carlo methods. *Journal of the Royal Statistical Society B 16*, 16, 1954. (Cité en page 16.)

[BSW+10] Peter J. Bennett, Kenichi Soga1, Ian Wassell, Paul Fidler, Keita Abe, Yusuke Kobayashi, and Martin Vanicek. Wireless sensor networks for underground railway applications : case studies in prague and london. *Smart Structures and Systems*, 6(5–6) :619–639, 2010. (Cité en page 13.)

[BXLJ08] Xiaole Bai, Dong Xuan, Ten H. Lai, and Weijia Jia. Complete optimal deployment patterns for full-coverage and k-connectivity wireless sensor networks. *MobiHoc'08*, 2008. (Cité en page 18.)

[Cam94] P. J. Cameron. Combinatorics topics, techniques, algorithms. *Cambridge University Press*, 1994. (Cité en page 74.)

[CD05] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11 :333—-340, 2005. (Cité en page 17.)

[CDKB11] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems : Concepts and Design (5th Edition)*. Addison Wesley, 2011. (Cité en page 35.)

[CF99] Imrich Chlamtac and András Faragó. A new approach to the design and analysis of peer-to-peer mobile networks. *Wireless Networks*, 5 :149–156, 1999. (Cité en page 30.)

[Cha] Anantha Chandrakasan. Mit μamps project website. `http://www-mtl.mit.edu/researchgroups/icsystems/uamps/.` (Cité en page 7.)

[Cha12] Bernard Chazelle. Natural algorithms and influence systems. *Communications of the ACM*, 55(12) :101–110, 2012. (Cité en page 98.)

[Cha15] Bernard Chazelle. An algorithmic approach to collective behavior. *J Stat Phys, Springer*, pages 514–548, 2015. (Cité en page 98.)

[Che04] Sivaram M.S.L. Cheekiralla. *Development of a wireless sensor unit for tunnel monitoring*. PhD thesis, Massachusetts Institute of Technology, 2004. (Cité en page 13.)

[Chu97] Fan R. K. Chung. Lectures on spectral graph theory. *University of Pennsylvania, Philadelphia, Pennsylvania 19104*, 1997. (Cité en page 21.)

[CK93] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks : evolution, opportunities, and challenges. *Proceedings of IEEE*, 91(8) :1247–1256, 1993. (Cité en page 6.)

[CLD08] Kenan Casey, Alvin Lim, and Gerry Dozier. A sensor network architecture for tsunami detection and response. *International Journal of Distributed Sensor Networks*, 4 :27—-42, 2008. (Cité en page 12.)

[CMKB02] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *in Proc. IEEE Int. Conf. Robot. Autom.*, 2002. (Cité en page 2.)

[CO10] Felipe M. Costa and Hideki Ochiai. A comparison of modulations for energy optimization in wireless sensor network links. In *Proceedings of the Global Telecommunications Conference (GLOBECOM 2010)*, 2010. (Cité en page 11.)

[CV14] Vincenzo Catania and Daniela Ventura. An approch for monitoring and smart planning of urban solid waste management using smart-m3 platform. In *Proceeding of the 15th Conference Of Fruct Association*, 2014. (Cité en page 14.)

[DEG99] T. K. Dey, H. Edelsbrunner, and S. Guta. Computational topology. *Contemporary Mathematics Series*, 1999. (Cité en page 72.)

[DGJM12] P. Dlotko, R. Ghrist, M. Juda, and M. Mrozek. Distributed computation of coverage in sensor networks by homological methods. *Applicable Algebra in Engineering, Communication and Computing*, 2012. (Cité en page 72.)

[Die00] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2000. (Cité en pages 15 et 21.)

[DMN05] Ralf Diekmann, S. Muthukrishnan, and Madhu V. Nayakkankuppam. Engineering diffusive load balancing algorithms using experiments. *Lecture Notes in Computer Science*, 1253 :111–122, 2005. (Cité en page 36.)

[DO11] S. L. Devadoss and J. O'Rourke. Discrete and computational geometry. *Princeton University Press*, 2011. (Cité en page 73.)

[dSG06] V. de Silva and R. Ghrist. Coordianate-free coverage in sensor networks with controlled boudaries via homology. *International Journal of Robotics Research*, 2006. (Cité en page 72.)

[dSG07] V. de Silva and R. Ghrist. Homological sensor networks. *Notices of the American Mathematical Society*, 2007. (Cité en pages 72 et 83.)

[DSW08] Alexandros D. G. Dimakis, Anand D. Sarwate, and Martin J. Wainwright. Geographic gossip : Efficient averaging for sensor networks. *IEEE Transactions on Signal Processing*, 56(3) :1205–1216, 2008. (Cité en pages xix, 56, 57, 59 et 61.)

[DTDM12] Milica Pejanovic Durisic, Zhilbert Tafa, Goran Dimic, and Veljko Milutinovic. A survey of military applications of wireless sensor networks. In *Mediterranean Conference on Embedded Computing MECO 2012*, pages 196—-199, Bar, Montenegro, June 2012. (Cité en page 14.)

[EH10] H. Edelsbrunner and J. L. Harer. Computational topology. *American Mathematical Society*, 2010. (Cité en pages 72 et 73.)

[Gag92] Douglas W. Gage. Command control for many-robot systems. In *Proceedings of AUVS-92, Huntsville AL*, 1992. (Cité en pages 2 et 18.)

[GBCJ06] Chunkai Gao, Francesco Bullo, Jorge Cortes, and Ali Jadbabaie. Notes on averaging over acyclic digraphs and discrete coverage control. In *Proceedings of the 45th IEEE Conference on Decision & Control*, 2006. (Cité en page 36.)

[GD08] Amitabha Ghosh and Sajal K. Das. Coverage and connectivity issues in wireless sensor networks : A survey. *Pervasive and Moblie Computing*, 4(3) :303–334, 2008. (Cité en page 9.)

[gdas12] Inspection générale des affaires sociales. L'hôpital. *Rapport remis au Président de la République, au Parlement et au Gouvernement*, 2012. (Cité en page 13.)

[GGB16] C. Giusti, R. Ghrist, and D. S. Bassett. Two's company, three (or more) is a simplex : Algebraic-topological tools for understanding higher-order structure in neural data. 2016. (Cité en pages 74 et 99.)

[Gho07] Sukumar Ghosh. *Distributed Systems : An Algorithmic Approach (Second Edition)*. Chapman & Hall/CRC Computer and Information Science Series, 2007. (Cité en pages 29 et 35.)

[Ghr14] R. Ghrist. *Elementary Applied Topology*. CreateSpace Independent Pusblishing Platform, 2014. (Cité en page 72.)

[Gib10] Peter Giblin. Graphs, surfaces and homology. *Cambridge University Press, 3rd edition*, 2010. (Cité en page 2.)

[Gil61] Edward N. Gilbert. Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*, 9 :533—-543, 1961. (Cité en page 30.)

[GM05] R. Ghrist and A. Muhammad. Coverage and hole-detection in sensor networks via homology. *in Proc. 4th Int. Symp. Inform. Processing Sensor Networks*, 2005. (Cité en page 2.)

[GR01] Chris Godsil and Gordon F. Royle. Algebraic graph theory. *Springer*, 2001. (Cité en pages 21 et 28.)

[Hal85] Peter Hall. On continuum percolation. *Ann. Probab.*, 13(4) :1250–1266, 1985. (Cité en pages 16 et 30.)

[Hat01] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001. (Cité en pages 2 et 72.)

[HGCG15] Anas Hanaf, Alban Goupil, Maxime Colas, and Guillaume Gellé. Distributed consensus algorithm and its application to detect coverage hole in sensor networks. *27th International Conference on Microelectronics (ICM)*, 2015. (Cité en pages 3 et 98.)

[Hil03] Jason Lester Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University Of California, Berkeley, 2003. (Cité en page 8.)

[HM06] Ramin Hekmat and Piet Van Mieghem. Connectivity in wireless ad-hoc networks with a log-normal radio model. *Mobile Networks and Applications*, 11 :351—-360, 2006. (Cité en page 16.)

[HT03] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA '03)*, 2003. (Cité en page 18.)

[HY61] John G. Hocking and Gail S. Young. *Topology*. Dover, 1961. (Cité en page 71.)

[ICHJ12] Franck Iutzeler, Philippe Ciblat, Walid Hachem, and Jérémie Jakubowicz. New broadcast based distributed averaging algorithm over wireless sensor networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3117–3120, 2012. (Cité en pages 3, 34, 66 et 67.)

[JLM06] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on,*, 2006. (Cité en page 2.)

[KB03] Koushik Kar and Suman Banerjee. Node placement for connected coverage in sensor networks. 2003. (Cité en page 17.)

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. *Proceedings of the 44th Annual IEEE Symposium*, pages 482–491, 2003. (Cité en pages 2, 34, 62, 63 et 64.)

[KGT09] JeongGil Ko, Tia Gao, and Andreas Terzis. Empirical study of a medical sensor application in an urban emergency department. In *Proceedings of the international conference on body area networks*, Los Angeles, California, USA, 2009. (Cité en page 13.)

[KH10] Alireza Khadivi and Martin Hasler. Fire detection and localization using wireless sensor networks. 2010. (Cité en page 36.)

[KKW05] Evangelos Kranakis, Danny Krizanc, and Eric Williams. Directional versus omnidirectional antennas for energy consumption and k-connectivity of networks of sensors. *Lecture Notes in Computer Science*, 3544 :357–368, 2005. (Cité en page 16.)

[KL05] Naoto Kimura and Shahram Latifi. A survey on data compression in wireless sensor networks. *Information Technology : Coding and Computing*, 2 :8–13, 2005. (Cité en page 11.)

[KMM03] T. Kaczynski, K. Mischaikow, and M. Mrozek. Computing homology. *Homology, Homotopy and Applications*, 2003. (Cité en page 76.)

[Koz08] D. Kozlov. Combinatorial algebraic topology. *Springer*, 2008. (Cité en page 72.)

[KS01] Srikanta Kumar and David Shepherd. Sensit : Sensor information technology for the warfighter. In *Proceedings of the 4th Conference on Information Fusion*, pages 3–9, Montreal, Canada, 2001. (Cité en page 7.)

[KW07] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley & Sons, 2007. (Cité en page 9.)

[LCL+11] Mo Li, Weifang Cheng, Kebin Liu, Yuan He, Xiang-Yang Li, and Xiangke Liao. Sweep coverage with mobile sensors. *IEEE Transactions on Mobile Computing*, 10(11) :1534–1545, 2011. (Cité en page 18.)

[LDWS08] Benyuan Liu, Olivier Dousse, Jie Wang, and Anwar Saipulla. Strong barrier coverage of wireless sensor networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008. (Cité en page 18.)

[LFM05] Zhiyun Lin, Bruce Francis, and Manfredi Maggiore. Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE Transactions On Automatic Control*, 50(1) :121–127, 2005. (Cité en page 36.)

[LH05] Jun Luo and Jean-Pierre Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. *IEEE INFOCOM*, 2005. (Cité en page 11.)

[Lim15] Lek-Heng Lim. Hodge laplacians on graphs. AMS Short Course on Geometry and Topology in Statistical Inference, 2015. (Cité en page 80.)

[LLSL09] Sang Hyuk Lee, Soobin Lee, Heecheol Song, and Hwang Soo Lee. Wireless sensor network design for tactical military applications : Remote large-scale environments. *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7, 2009. (Cité en page 14.)

[LSL+04] Jerome Peter Lynch, Arvind Sundararajan, Kincho H Law, Anne S Kiremidjian, and Ed Carryer. Embedding damage detection algorithms in a wireless sensing unit for operational power efficiency. *Smart Materials and Structures*, 13(4) :800–810, 2004. (Cité en page 9.)

[LSL+05] Jerome Peter Lynch, Arvind Sundararajan, Kincho H Law, Anne S Kiremidjian, and Ed Carryer. Nonintrusive precision instrumentation of microcontroller software. *ACM, New York, NY, ETATS-UNIS*, 40(7) :59–68, 2005. (Cité en page 9.)

[LWS04] Ruizhong Lin, Zhi Wang, and Youxian Sun. Wireless sensor networks solutions for real time monitoring of nuclear power plant. *IEEE*, 2004. (Cité en page 11.)

[LZWM15] Kezhong Liu, Yang Zhuang, Zhibo Wang, and Jie Ma. Spatiotemporal correlation based fault-tolerant event detection in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2015, 2015. (Cité en page 10.)

[Mas91] William S. Massey. *A Basic Course in Algebraic Topology*. Springer, 1991. (Cité en pages 2 et 71.)

[MdW09] Brandon J. Moore and Carlos Canudas de Wit. Formation control via distributed optimization of alignment error. In *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference (CDC/CCC)*, 2009. (Cité en page 36.)

[ME06] A. Muhammad and M. Egerstedt. Control using higher order laplacians in network topologies. *in Proc. 17th Int. Symp. Math. Theory Netw. Syst*, 2006. (Cité en pages 81 et 83.)

[Men86] M.V. Menshikov. Coincidence of critical points in percolation problems. *Soviet Math. Dokl.*, 33, 1986. (Cité en pages 16 et 30.)

[MJ07] A. Muhammad and A. Jadbabaie. Dynamic coverage verification in mobile sensor networks via switched higher order laplacians. *in Robotics : Science and Systems*, 2007. (Cité en page 72.)

[MKQP01] Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, and Miodrag Potkonjak. Exposure in wireless ad-hoc sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001. (Cité en page 18.)

[Moh91] Bojan Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, 1991. (Cité en page 21.)

[Mor05] Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions On Automatic Control*, 50(2) :169–182, 2005. (Cité en page 36.)

[Mor07] Patricia A. Morreale. Smart bridges, smart tunnels : Transforming wireless sensor networks from research prototypes into robust engineering infrastructure. *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pages 810–814, 2007. (Cité en page 13.)

[MPS+04] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. *Kluwer Academic Publishers*, pages 399–423, June 2004. (Cité en page 12.)

[MRG+08] Yajie Ma, Mark Richards, Moustafa Ghanem, Yike Guo, and John Hassard. Air pollution monitoring and mining based on sensor grid in london. In *Sensors*, volume 8, pages 3601—-3623, 2008. (Cité en page 14.)

[MSMSMSM13] Afsaneh Minaie, Ali Sanati-Mehrizy, Paymon Sanati-Mehrizy, and Reza Sanati-Mehrizy. Application of wireless sensor networks in health care system. In *ASSEE Conference & Exposition*, Atlanta, GE, USA, June 2013. (Cité en page 13.)

[Mur09] Richard M. Murray. Introduction to graph theory and consensus. *Caltech Control and Dynamical Systems*, 2009. (Cité en page 21.)

[NAB06] K. Nadiminti, M. De Assuncao, and R. Buyya. Distributed systems and recent innovations : Challenges and benefits. *InfoNet Magazine*, 2006. (Cité en page 2.)

[NDG09] Bobak Nazer, Alexandros G. Dimakis, and Michael Gastpar. Neighborhood gossip : Concurrent averaging through local interference. In *ICASSP*, 2009. (Cité en pages xix, 57, 58, 59 et 61.)

[NDG11] Bobak Nazer, Alexandros G. Dimakis, and Michael Gastpar. Local interference can accelerate gossip algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 5(4) :876–887, 2011. (Cité en pages xix, 57, 58, 59 et 61.)

[NVR+11] Vukasin Nuhijevic, Sasa Vukosavljev, Boris Radin, Nikola Teslic, and Mirko Vucelja. An intelligent home networking system. *IEEE International Conference on Consumer Electronics (ICCE-Berlin)*, 2011. (Cité en page 14.)

[ONe07] William D. ONeil. The cooperative engagement capability. transforming naval anti-air warfare. *Case Studies in National Security Transformation*, 2007. (Cité en page 7.)

[OSFFS06] Reza Olfati-Saber, Elisa Franco, Emilio Frazzoli, and Jeff S. Shamma. Belief consensus and distributed hypothesis testing in sensor networks. *Network Embedded Sensing and Control*, pages 169–182, 2006. (Cité en pages 37 et 38.)

[OSM04] Reza Olfati-Saber and Richard M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions On Automatic Control*, 49(9) :1520–1533, 2004. (Cité en pages 1 et 36.)

[OSS05] Reza Olfati-Saber and Jeff S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005. (Cité en page 36.)

[PB05] Christian Prehofer and Christian Bettstetter. Self-organization in communication networks : Principles and design paradigms. *IEEE Commun. Mag.*, pages 78–85, 2005. (Cité en page 8.)

[Pen03] Mathew Penrose. *Random Geometric Graphs*. (Oxford Studies in Probability) 1st Edition, 2003. (Cité en page 30.)

[PK00] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5), 2000. (Cité en page 7.)

[PKB99] Kristofer S. J. Pister, J. M. Kahn, and Bernhard E. Boser. Smart dust : wireless networks of millimeter-scale sensor nodes. *Electronics Research Laboratory Research Summary*, 1999. (Cité en page 7.)

[PSR+06] Valery A. Petrushin, Omer Shakil, Damian Roqueiro, Gang Wei, and Anatole V. Gershman. Multiple-sensor indoor surveillance system. In *Proceedings of The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, 2006. (Cité en page 15.)

[RAd+00] Jan M. Rabaey, M. Josie Ammer, Julio L. da, Silva Jr., Danny Patel, and Shad Roundy. Picoradio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7) :42–48, 2000. (Cité en page 7.)

[RGBRL10] Luis Ruiz-Garcia, Pilar Barreiro, Jose Ignacio Robla, and Loredana Lunadei. Testing zigbee motes for monitoring refrigerated vegetable transportation under real conditions. *Sensors*, 10 :4968–4982, 2010. (Cité en page 13.)

[RV06] Ramesh Rajagopalan and Pramod K. Varshney. Data-aggregation techniques in sensor networks : A survey. *IEEE Communications Surveys & Tutorials*, 8(4) :48–63, 2006. (Cité en page 11.)

[Sat99] H. Sato. Algebraic topology : An intuitive approach, translations of mathematical monographs. *American Mathematical Society*, 1999. (Cité en page 72.)

[SH03] Anna Scaglione and Yao-Win Hong. Opportunistic large arrays : cooperative transmission in wireless multihop ad hoc networks to reach far distances. *IEEE Transactions on Signal Processing*, 51(8) :2082—-2092, 2003. (Cité en page 16.)

[SHW+10] Frank Stajano, Neil Hoult, Ian Wassell, Peter Bennett, Campbell Middleton, and Kenichi Soga. Smart bridges, smart tunnels : Transforming wireless sensor networks from research prototypes into robust engineering infrastructure. *Ad Hoc Networks*, (8) :872—-888, 2010. (Cité en page 12.)

[SK11] S. Sudevalayam and P. Kulkarn. Energy harvesting sensor nodes : Survey and implications. *IEEE Communications Surveys & Tutorials*, 13(3) :443–461, 2011. (Cité en pages 8 et 9.)

[SLPB04] Qingshan Shan, Ying Liu, Gareth Prosser, and David Brown. Wireless intelligent sensor networks for refrigerated vehicle. *IEEE 6th CAS Symp. on Emerging Technologies : Mobile and Wireless Comm*, 2004. (Cité en page 13.)

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8) :888–905, 2000. (Cité en page 28.)

[SOSM05] Demetri P. Spanos, Reza Olfati-Saber, and Richard M. Murray. Dynamic consensus for mobile networks. In *Proceedings of the 16th IFAC World Congress, Elsevier*, 2005. (Cité en page 36.)

[SP04] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6) :38–43, December 2004. (Cité en pages 13 et 14.)

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1972. (Cité en page 23.)

[TM03] Malik Tubaishat and Sanjay Madria. Sensor networks : an overview. *IEEE Potentials*, 22(2) :20–23, 2003. (Cité en page 15.)

[Tsi84] John Nikolaos Tsitsiklis. *Problems in decentralized decision making and computation.* PhD thesis, Massachusetts Institute of Technology, Cambrigde, MA, 1984. (Cité en page 35.)

[TSJ10] A. Tahbaz-Salehi and A. Jadbabaie. Distributed coverage verification in sensor networks without location information. *IEEE Trans. Autom. Control*, 2010. (Cité en pages 72, 83 et 84.)

[TVP+11] Dan Stefan Tudose, Andrei Voinescu, Madi-Tatiana Petrareanu, Andrei Bucur, Dumitrel Loghin, and Adrian Bostan. Home automation design using 6lowpan wireless sensor networks. In *Distributed Computing in Sensor Systems and Workshops (DCOS)*, pages 1—-6, 2011. (Cité en page 14.)

[WALW+06] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Computer Society*, 2006. (Cité en page 12.)

[WC07] Andrew Wheeler and Ember Corporation. Commercial applications of wireless sensor networks using zigbee. *Communications Magazine, IEEE*, 45(4) :70–77, 2007. (Cité en page 13.)

[WWJ+11] Shuai Wang, Shuai Wang, Hongbo Jiang, Xiaoqiang Ma, Wenyu Liu, Kai Peng, Bo Liu, and Yan Dong. Energy efficient broadcasting using network coding aware protocol in wireless ad hoc network. In *IEEE ICC 2011*, 2011. (Cité en page 11.)

[XB04] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53 :65–78, 2004. (Cité en pages 36, 41, 42 et 43.)

[XBK07] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal on Parallel Distributed Computation*, page 33–46, 2007. (Cité en page 42.)

[XBL05] Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. *The 4th International Symposium on Information Processing in Sensor Networks*, pages 63–70, 2005. (Cité en pages 2 et 36.)

[YWM05] Liyang Yu, Neng Wang, and Xiaoqiao Meng. Real-time forest fire detection with wireless sensor network. *IEEE*, 2005. (Cité en page 12.)

[YWZ13] Bo Yang, Weimin Wu, and Guangxi Zhu. Distributed averaging in wireless sensor networks with triplewise gossip algorithms. *IEEE 2013 Tencon - Spring*, pages 178–182, 2013. (Cité en pages xix, 54, 55 et 56.)

[ZZSH12] Chuan Zhu, Chunlin Zheng, Lei Shu, and Guangjie Han. A survey on coverage and connectivity issues in wireless sensor, networks. *Journal of Network and Computer Applications*, 35(2) :619–632, 2012. (Cité en page 9.)

ALGORITHMES DISTRIBUES DE CONSENSUS DE MOYENNE ET LEURS APPLICATIONS DANS LA DETECTION DES TROUS DE COUVERTURE DANS UN RESEAU DE CAPTEURS

Les algorithmes distribués de consensus sont des algorithmes itératifs de faible complexité où les nœuds de capteurs voisins interagissent les uns avec les autres pour parvenir à un accord commun sans unité coordinatrice.
Comme les noeuds dans un réseau de capteurs sans fil ont une puissance de calcul et une batterie limitées, ces algorithmes distribués doivent parvenir à un consensus en peu de temps et avec peu d'échange de messages.
La première partie de cette thèse est basée sur l'étude et la comparaison des différents algorithmes de consensus en mode synchrone et asynchrone en termes de vitesse de convergence et taux de communications.
La seconde partie de nos travaux concerne l'application de ces algorithmes de consensus au problème de la détection de trous de couverture dans les réseaux de capteurs sans fil.
Ce problème de couverture fournit aussi le contexte de la suite de nos travaux.
Il se décrit comme étant la façon dont une région d'intérêt est surveillée par des capteurs.
Différentes approches géométriques ont été proposées mais elles sont limitées par la nécessité de connaitre exactement la position des capteurs ; or cette information peut ne pas être disponible si les dispositifs de localisation comme par exemple le GPS ne sont pas sur les capteurs.
À partir de l'outil mathématique appelé topologie algébrique, nous avons développé un algorithme distribué de détection de trous de couverture qui recherche une fonction harmonique d'un réseau, c'est-à-dire annulant l'opérateur du Laplacien de dimension 1.
Cette fonction harmonique est reliée au groupe d'homologie $H1$ qui recense les trous de couverture.
Une fois une fonction harmonique obtenue, la détection des trous se réalise par une simple marche aléatoire dans le réseau.

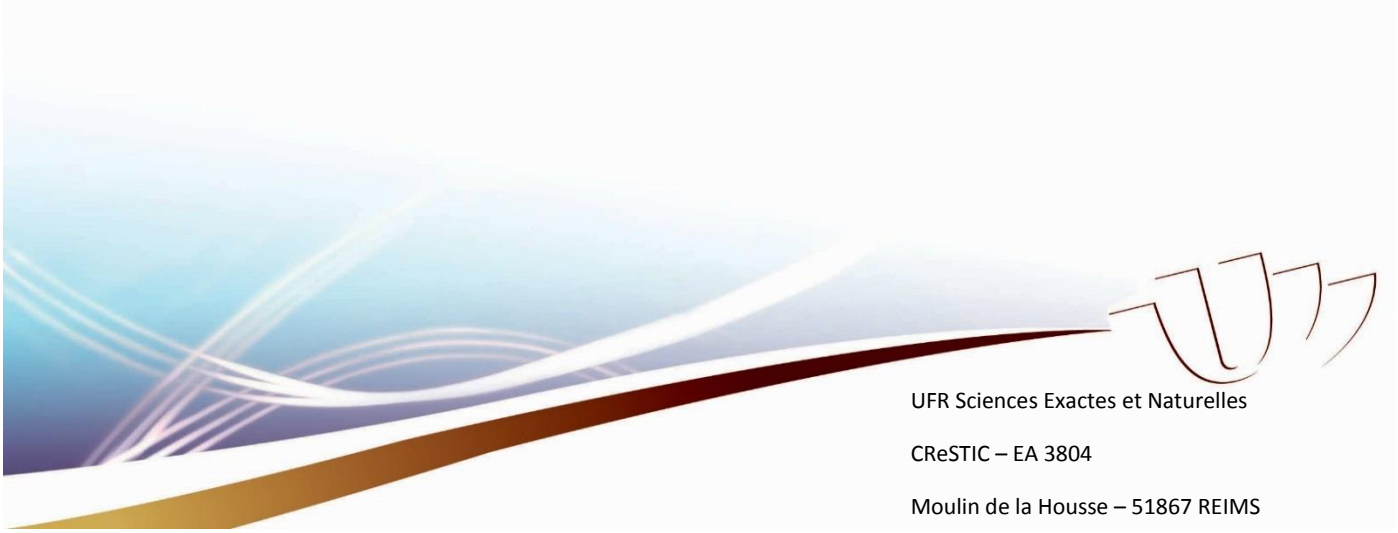
Réseau de capteurs, consensus de moyenne, détection distribuée, trou de couverture, topologie algébrique, Laplacien combinatoire

DISTRIBUTED AVERAGE CONSENSUS ALGORITHMS AND THEIR APPLICATIONS TO DETECT COVERAGE HOLE IN SENSORS NETWORK

Distributed consensus algorithms are iterative algorithms of low complexity where neighboring sensors interact with each other to reach an agreement without coordinating unit.
As the nodes in a wireless sensor network have limited computing power and limited battery, these distributed algorithms must reach a consensus in a short time and with little message exchange.
The first part of this thesis is based on the study and comparison of different consensus algorithms synchronously and asynchronously in terms of convergence speed and communication rates.
The second part of our work concerns the application of these consensus algorithms to the problem of detecting coverage holes in wireless sensor networks.
This coverage problem also provides the context for the continuation of our work.
This problem is described as how a region of interest is monitored by sensors.
Different geometrical approaches have been proposed but are limited by the need to know exactly the position of the sensors; but this information may not be available if the locating devices such as GPS are not on the sensors.
From the mathematical tool called algebraic topology, we have developed a distributed algorithm of coverage hole detection searching a harmonic function of a network, that is to say canceling the operator of the 1-dimensional Laplacian.
This harmonic function is connected to the homology group $H1$ which identifies the coverage holes.
Once a harmonic function obtained, detection of the holes is realized by a simple random walk in the network.

Sensors network, consensus algorithm, distributed detection, coverage hole, algebraic topology, combinatorial Laplacian.

Discipline : AUTOMATIQUE, SIGNAL, PRODUCTIQUE, ROBOTIQUE



UFR Sciences Exactes et Naturelles

CReSTIC – EA 3804

Moulin de la Housse – 51867 REIMS