Cadre méthodologique et applicatif pour le développement de réseaux de capteurs fiables

Farid Lalem

THÈSE / UNIVERSITÉ DE BRETAGNE OCCIDENTALE
sous le sceau de l'Université Bretagne Loire

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE
Mention : Informatique
École Doctorale MathSTIC

présentée par
Farid LALEM
Préparée au Lab-STICC UMR CNRS 6285
UFR Sciences et Sciences de l'Ingénieur

Cadre méthodologique et applicatif pour le développement de réseaux de capteurs fiables.

Thèse soutenue le 11 septembre 2017
devant le jury composé de :

M. Congduc Pham
Professeur, Université de Pau / *Rapporteur*
M. Mohand-Tahar Kechadi
Professeur, Université UCD Dublin / *Rapporteur*
M. Basel Solaiman
Professeur des universités, Télécom Bretagne / *Examinateur*
M. César Viho
Professeur, Université de Rennes / *Examinateur*
M. Loic Lagadec
Professeur des universités, ENSTA Bretagne / *Examinateur*
M. Ahcène Bounceur
Maître de Conférences-HDR, Université de Brest / *Directeur de thèse*
M. Reinhardt Euler
Professeur des universités, Université de Brest / *Co-Directeur de thèse*
M. Rahim Kacimi
Maitre de conférences, Université de Toulouse / *Co-Encadrant*
M. Laurent Nana
Professeur des universités, Université de Brest / *Invité*

Je dédie ce travail
A mes parents Ahmed et Mebarka,
A Nadjet et à mes enfants Akram, Assil et Miral,
A mes frères et soeurs,
A mes grands-parents et à mes oncles et tantes,
A mes beaux parents.

Remerciements

Je tiens tout d'abord à remercier le bon dieu tout puissant d'avoir terminé ce modeste travail.

Je veux exprimer ma reconnaissance et ma gratitude à mon directeur de thèse Mr. Ahcène Bounceur pour son aide efficace et précieux durant ces trois années de thèse, et pour son encouragement, sa patience, ainsi que pour sa disponibilité, ses idées et ses conseils qui m'ont permis de mener à bien cette thèse.

Je suis également très reconnaissant envers Mr. Reinhardt Euler, Professeur à l'UBO, pour ses conseils très constructifs et son soutien académique et scientifique. Je suis vraiment honoré de faire partie d'un travail important avec lui.

Je suis très reconnaissant envers Mr Rahim Kacimi, mon co-encadrant, pour sa disponibilité malgré la distance qui nous sépare, son ouverture d'esprit, sa générosité et les nombreuses discussions fructueuses qui ont animé ces trois années de thèse.

Je souhaite également remercier tous les membres du laboratoire Lab-STICC, chercheurs, thésards, stagiaires et visiteurs, pour leur disponibilité, leur sympathie et pour l'ambiance amicale qui règne au sein du laboratoire Lab-STICC.

Bien sûr, je ne trouverais sans doute pas les mots pour remercier assez les personnes qui me sont les plus chères : mes parents, pour leur sacrifices, leur patience, et tout ce qu'ils ont fait pour m'apporter le bonheur. Qu'ils sachent à travers ces quelques mots combien je leur suis reconnaissant et combien je sais tout ce que je leur dois. Enfin ma tendre épouse pour son soutiens, ses encouragements, son dévouement et surtout ses sacrifices dans les instants les plus difficiles.

J'adresse aussi mes sincères remerciements à tous mes amis, qui m'ont soutenu de près ou de loin durant ces trois années de thèse.

A tous ceux qui sont proches de mon cœur et dont je n'ai pas cité les noms.

Table des Matières

Liste des Figures

Liste des Tableaux

Liste des Algorithmes

Introduction générale

Il vaut mieux honorer sa profession que d'être honoré par elle.
(proverbe)

Les réseaux de capteurs sans fil émergent comme une technologie innovatrice qui peut révolutionner et améliorer notre vie quotidienne. Néanmoins, l'utilisation d'une telle technologie soulève de nouveaux défis concernant le développement de systèmes fiables et sécurisés.

Un réseau de capteurs sans fil (RCSF) est constitué d'un grand nombre de nœuds capteurs autonomes distribués spatialement et densément déployés dans un terrain d'intérêt, qui effectuent un processus de mesures collaboratif, pour surveiller ou contrôler des conditions physiques ou environnementales, telles que la température, l'humidité, la lumière, le son, les vibrations, la pression, le mouvement, les polluants, etc.

Ces nœuds capteurs sont caractérisés par des ressources limitées, tels que l'énergie, la capacité de calcul, la mémoire et une courte portée de communication, etc.

Grâce à leur capacité d'auto-organisation et leurs faible coût, ils permettent de créer des applications à grande échelle. Jusqu'à présent, leur déploiement est toujours complexe dans de nombreux domaines tels que l'environnement, la médecine, et le domaine militaire. Cependant, en particulier pour les applications militaires ou médicales, ils ont besoin de solutions sûres et fiables. Dans de telles applications, l'évènement détecté doit être envoyé à la station de base de manière urgente, fiable, et avec une faible consommation d'énergie.

En outre, l'objectif d'un RCSF est non seulement de collecter des données de la zone d'intérêts, mais aussi d'analyser ces données au moment opportun afin de prendre des décisions importantes.

Cependant, les mesures brutes collectées par les nœuds capteurs sont peuvent être peu fiables et souffrir d'imprécision et d'incomplétude pour diverses raisons, telles que le bruit, les erreurs, les évènements imprévus, les attaques malveillantes et le vieillissement. D'autres raisons liées à la nature de l'environnement peuvent éventuellement ajouter une difficulté de déploiement des nœuds. Ce qui peut conduire à une collecte de données incorrectes.

Par ailleurs, si les données collectées ne fournissent pas une représentation fiable du phénomène physique surveillé, la transmission et le traitement de ces données peuvent entraîner un gaspillage énorme des ressources.

De plus, les nœuds capteurs sont sujets à l'échec à cause des défaillances subites (circuit cassé, intrusion, batterie épuisée, etc.), ce qui peut également contribuer d'une part, à générer des mesures inexactes et affecter la fiabilité des données et, d'autre part, influencer strictement les performances du réseau. Par conséquent, les nœuds capteurs défaillants doivent être exclus ou remplacés dans le réseau.

Aussi, la sécurité dans ce type de réseaux est d'une importance stratégique, voire vitale, car leur bon fonctionnement peut conduire à des conséquences dangereuses dans des contextes applicatifs critiques. En effet, étant donné le fait que les nœuds capteurs ont des ressources limitées, les algorithmes de sécurité traditionnels ne conviennent pas bien aux RCSFs [20], puisque le processus de cryptage et de décryptage doit être effectué au niveau de chaque nœud, ce qui génère des coûts de calcul très élevés. De plus, les attaques malveillantes comme la modification des données, la

suppression et l'insertion des données, peuvent affecter la qualité des données collectées par les nœuds capteurs. Par conséquent, la protection de l'intégrité et de l'authenticité de ces données est un processus nécessaire pour garantir la qualité de celle-ci avant de les utiliser pour prendre des décisions.

Dans ce contexte et dans le cadre du travail de cette thèse, nous visons à proposer des solutions permettant de garantir un certain niveau de fiabilité dans un RCSF dédié aux applications sensibles. Nous avons ainsi proposé trois solutions, qui sont :

- Le développement de méthodes permettant de détecter des nœuds capteurs défaillants dans un RCSF,
- Le développement de méthodes permettant de détecter les anomalies dans les mesures collectées par les nœuds capteurs, et par la suite, les capteurs usés (fournissant de fausses mesures),
- Le développement de méthodes permettant d'assurer l'intégrité et l'authenticité des données transmises dans un RCSF.

Pour le premier objectif, nous avons proposé un algorithme de détection de sommets frontières dans un graphe Euclidien connecté nommé (LPCN: Least Polar-angle Connected Node Algorithm to Find a Polygon Hull in a Connected Euclidean Graph [21]). Par la suite, nous avons utilisé la version distribuée de cet algorithme (D-LPCN [22]) pour pouvoir l'appliquer dans le cas des RCSFs afin de détecter des nœuds défaillants situés dans la frontière du réseau [23].

Pour ce qui est du deuxième objectif, nous avons proposé une nouvelle approche, fondée sur la théorie des copules pour modéliser des distributions multidimensionnelles des mesures collectées par les capteurs [24][25][26].

Enfin, pour le troisième objectif, nous avons proposé une technique de tatouage entièrement distribué qui se distingue en terme de ressources, par des exigences légères, afin d'assurer l'intégrité et l'authenticité des données transmises dans un RCSF [27].

Dans la suite de ce manuscrit, nous allons détailler davantage nos propositions autour des trois objectifs cités ci-dessus. Ce rapport est structuré comme suit:

Dans le chapitre 1, nous présentons un état de l'art sur les méthodes de détection de sommets frontières dans les graphes Euclidiens connectés, ensuite, nous présentons l'algorithme LPCN où nous essayons de trouver un ensemble de sommets permettant de représenter la forme géométrique du graphe sous forme d'enveloppe polygonale. Par la suite, nous montrons l'application de l'algorithme LPCN dans quelques domaines utiles, en présentant la version centralisée appliquée à des réseaux de capteurs sans fil, enfin, nous présentons l'implémentation et les résultats obtenus de l'algorithme LPCN avec le simulateur CupCarbon.

Dans le chapitre 2, nous présentons en premier lieu, un état de l'art qui comprend, les méthodes de détection de nœuds frontières dans les RCSFs, et les méthodes de détection des défaillances dans les RCSFs, ensuite, la présentation de la version distribuée de l'algorithme LPCN qui est D-LPCN, suivi par les résultats de simulation et d'implémentation réel de l'algorithme D-LPCN avec des nœuds capteurs TelosB. En second lieu, nous présentons une approche entièrement distribuée,

appelée **Boundary Node Failure Detection** (BNFD), pour un contrôle efficace d'une frontière en se basant sur la détermination de la frontière d'un RCSF et la détection des nœuds défaillants dans celle-ci. La frontière est déterminée à l'aide d'un algorithme qui a la propriété de déterminer à chaque itération, le voisin à un seul saut du nœud courant de la frontière [22]. Par conséquent, chaque nœud de la frontière connaît son prochain voisin direct et peut communiquer avec lui afin de tester périodiquement sa présence. Lorsqu'une situation d'échec est détectée, une restructuration du réseau sera lancée pour trouver une nouvelle frontière et une alarme sera déclenchée. Finalement, nous présentons les résultats obtenus de l'implémentation de cette approche avec le simulateur Castalia, en terme de consommation énergétique.

Dans le chapitre 3, nous présentons en premier lieu, un état de l'art sur les méthodes de détection des anomalies dans les RCSFs ainsi que les méthodes de détection des capteurs usés fournissant de fausses mesures. En second lieu, nous présentons quelques notions fondamentales sur la théorie des copules, et son application pour détecter les anomalies et les capteurs usés dans les RCSFs multivariés. En effet, un polygone local résultant de l'application de cette théorie à des mesures historiques de données est téléversé dans chaque nœud. Les anomalies sont alors détectées en ligne de manière distribuée en comparant le flux de données d'entrée lié à la mesure et à la surveillance de phénomènes physiques avec ce polygone. Si les valeurs détectées n'appartiennent pas à ce polygone, elles seront considérées comme des anomalies. Et si le capteur détecte plusieurs anomalies de manière successive, alors dans ce cas, il sera considéré étant le générateur de celles-ci, car il est usé. Ainsi, il doit être remplacé ou calibré. En dernier lieu, nous présentons nos résultats expérimentaux sur des jeux de données, collectés par des réseaux de capteurs réels en utilisant le logiciel de statistique R [28].

Dans le chapitre 4, nous présentons, premièrement, quelques définitions et une petite introduction sur le tatouage numérique, ainsi qu'un état de l'art sur les méthodes de tatouage numérique appliqué aux RCSFs. Deuxièmement, nous présentons une nouvelle approche de tatouage numérique entièrement distribuée pour les RCSFs. Cette approche vise à assurer l'intégrité et l'authenticité des données transmises dans un RCSF. Dans l'étape de collecte des données, chaque nœud du réseau, intègre dynamiquement dans chaque paquet de données, la même signature (*watermark*) fixée localement avant de le transmettre à ses voisins. La vérification de la signature est effectuée par les nœuds récepteurs. Lorsque le paquet de données est reçu, il sera utilisé pour calculer une nouvelle signature qui sera ensuite comparée avec la signature fixée localement. Si leurs valeurs ne sont pas identiques, le paquet de données reçu sera rejeté. Finalement, nous présentons les résultats d'implémentation et de simulation de l'approche proposée avec le simulateur CupCarbon, tout en comparant les résultats obtenus avec d'autres approches citées dans l'état de l'art.

Dans le chapitre 5, nous présentons une synthèse sur les travaux réalisés et les résultats obtenus dans le cadre de cette thèse, ensuite, nous concluons par des perspectives.

On commence par vouloir tout le bien, et on finit par espérer le moindre mal.
(proverbe)

1

Algorithme LPCN

Ce chapitre est consacré à la présentation de l'algorithme proposé LPCN [29]. En fait, de nombreuses applications de la vie réelle peuvent être modélisées sous forme d'un graphe Euclidien connecté, comme les réseaux de capteurs sans fil (RCSF), les pixels d'une image, les villes d'un pays, les ordinateurs personnels, etc. La recherche de l'enveloppe polygonale de ce type de graph peut aider à résoudre des problématiques réelles comme par exemple la surveillance de sites sensibles, la détection d'anomalies, la localisation d'une région d'intérêt, la reconstruction de formes ou l'extraction de clusters dans un ensemble de données, dessiner des contours dans les images médicales et biologiques, la vision assistée par ordinateur, la théorie des jeux, etc. [30][31][32][24][23][33][34].

De ce fait, nous avons proposé un nouvel algorithme permettant de trouver les sommets frontières d'un graphe Euclidien connecté, où nous essayons de trouver un ensemble de sommets permettant de représenter la forme géométrique du graphe sous forme d'enveloppe polygonale, un polygone simple formé par des arêtes du graphe de sorte que tous les sommets du graphe soient sur le polygone, soient entourés par celui-ci. Plus précisément, nous cherchons un cycle fermé de longueur minimale dans le graphe tel que tous les sommets soient sur ou entourés par ce cycle. Ce problème peut être formulé comme suit. Nous considérons un graphe non orienté $G = (V, E)$, où $V = \{v_0, v_1, ..., v_{n-1}\}$ est l'ensemble des sommets du graphe et E son ensemble d'arêtes. Nous supposons que toutes les arêtes de G sont représentées par des droites. L'enveloppe polygonale peut être donnée par un ensemble de sommets $\mathbb{B}_V$ et un ensemble d'arêtes $\mathbb{B}_E$ comme suit:

$$\mathbb{B}_V = \{v_0, v_1, ..., v_{i-1}, v_i, v_{i+1}, ..., v_h\} \subseteq V,$$
$$\mathbb{B}_E = \{\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{h-1}, v_h\}\} \subseteq E$$

tel que,
a) $v_0 = v_h$ (C'est-à-dire, v_0 est le même que v_h),
b) les sommets dans $V \backslash \mathbb{B}_V$ représentent l'ensemble $\mathrm{INT}(\mathbb{B}_V)$,
C'est-à-dire, l'ensemble de sommets situés à l'intérieur du cycle.

La solution que nous avons proposée par l'algorithme LPCN permet de trouver l'enveloppe polygonale (ou une frontière) d'un graphe Euclidien connecté donné. Le principal avantage de l'algorithme LPCN est la façon dont il définit la frontière. Son idée principale est donnée comme suit. Dans chaque itération i le prochain sommet frontière v_{i+1} est déterminé par le sommet qui a l'angle polaire minimum $\varphi_{min}(v_{i-1}, v_i, v_{i+1})$ formé par les arêtes $\{v_i, v_{i-1}\}$ et $\{v_i, v_{i+1}\}$ (voir Figure 1.1), où v_i est le sommet frontière sélectionné dans l'itération actuelle i et v_{i-1} est le sommet frontière trouvé dans l'itération précédente $i - 1$.

Notre objectif est de déterminer un simple polygone fermé, mais il s'avère qu'il peut suffire d'obtenir une marche qui n'est pas fermée. l'algorithme fonctionne sous l'hypothèse que certains sous-graphes particuliers n'existent pas, par exemple les graphes en bateau (comme représenté dans

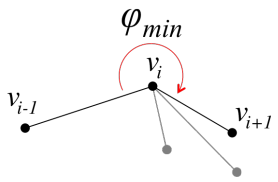


Figure 1.1: Angle formé par trois sommets d'un graphe.

la Figure 1.5), les graphes d'ancre (donnés dans la Figure 1.10), ou les sommets de degré 1. Ces graphes nécessitent un traitement spécial, et les marches non-fermées pourraient être suffisantes au cas où nous voulons juste visiter leurs sommets. Nous montrerons plus tard comment surmonter ces limitations en présence de telles structures. Notre premier algorithme détermine dans chaque itération le prochain sommet frontière voisin du sommet frontière actuellement trouvé. Sa complexité est $O(kh^2)$, où k est le degré du graph et h le nombre des sommets sur l'enveloppe polygonale.

Ce chapitre est organisé comme suit. Section 1 présente un état de l'art sur les méthodes de détection de sommets frontières dans des graphes Euclidiens connectés.
Section 2 introduit l'algorithme de Jarvis [2] et la version de base de l'algorithme LPCN proposé. La section 3 présentera les différentes limitations causées par les sous-graphes qui conduisent notre premier algorithme à échouer et les moyens de les surmonter. Cette section comprend également la version finale de l'algorithme LPCN proposé. La validation de l'algorithme est donnée dans la section 4. Les résultats de la simulation et la complexité sont présentés dans la section 6. Certaines applications seront présentées dans la section 6.

1 Les méthodes permettant de trouver l'enveloppe convexe ou polygonale

Dans ce qui suit, nous examinons quelques algorithmes utiles permettant de trouver une enveloppe convexe ou polygonale (concave) pour un ensemble donné de points dans le plan. Parfois, un algorithme d'enveloppe convexe peut être modifié pour déterminer une enveloppe polygonale comme c'est le cas pour l'algorithme LPCN, où nous proposons un algorithme d'enveloppe polygonale basé sur l'algorithme d'enveloppe convexe de Jarvis [2] qui utilise également une méthode basée sur l'angle.

1.1 L'enveloppe Convexe

1.1.1 Algorithme de Graham [1]

Cet algorithme part d'un point extrême appelé pivot, qui peut être le point avec la coordonnée x minimale. Les points restants seront triés dans l'ordre des angles croissants par rapport au pivot et l'axe des abscisses. L'algorithme s'arrête avec un polygone en étoile. Enfin, l'enveloppe convexe sera construite en marchant autour du polygone en forme d'étoile, en ajoutant des arêtes lorsque nous faisons un tournant à gauche, un retour en arrière lorsque nous faisons un tournant à droite. La complexité de cet algorithme est $O(n \log n)$, où n est le nombre de points donnés.

1.1.2 Algorithme de Jarvis [2]

Cet algorithme commence par le point ayant une coordonnée x minimale, par exemple. Puis récursivement, il ajoute le point ayant l'angle minimum par rapport au point précédent. Cet algorithme est détaillé dans la section 2.1 puisque notre algorithme proposé est basé sur dessus. La complexité de l'algorithme de Jarvis est $O(nh)$ où h est le nombre de points sur l'enveloppe convexe.

1.1.3 Quick-hull [3]

L'algorithme Quick-hull commence par le calcul des points avec les coordonnées minimales et maximales pour l'axe des abscisses et l'axe des ordonnées. De toute évidence, ces 4 points seront sur l'enveloppe convexe. L'interconnexion de ces quatre points conduira à un quadrilatère convexe. Tous les points à l'intérieur de ce quadrilatère sont éliminés. Pour chaque côté extérieur du quadrilatère, on trouve le point le plus éloigné pour former avec lui un tringle d'angle, tous les points qui se trouvent à l'intérieur de ces triangles sont encore éliminés. Et, les nouvelles arêtes des triangles formés sont ajoutées aux cotes à traiter. La même procédure sera répétée pour les triangles nouvellement obtenus jusqu'à ce qu'il n'y ait pas de point en dehors des triangles. La complexité de cet algorithme est $O(n^2)$.

1.1.4 Algorithme Incremental [4]

Cet algorithme peut être utilisé dans un espace multidimensionnel. Il opère en insérant les points un par un et avec une mise à jour incrémentale d'une enveloppe initiale. Si le nouveau point est à l'intérieur de l'enveloppe initiale, il n'y a rien à faire. Autrement, toutes les arêtes que le nouveau point peut voir doivent être supprimées. Ensuite, le nouveau point sera connecté à ses deux points voisins et l'enveloppe est mise à jour. En répétant ce processus pour les points restants à l'extérieur de l'enveloppe actuelle, une enveloppe convexe sera finalement construite. La complexité de cet algorithme est $O(n^{d+1/2})$ où d est la dimension de l'espace considéré.

1.1.5 Algorithme TORCH [5]

L'algorithme (TORCH) baptisé $TheTotalORderheuristic-basedConvexHull$ est un algorithme de tri qui commence par trier tous les points selon l'axe des abscisses x. Ensuite, il détermine les quatre points extrêmes qui sont les plus à gauche, à droite, en bas, et en haut afin de trouver les quatre sous enveloppes latérales entre ses points de retournement. Ce procédé de tri nous permet d'obtenir immédiatement une enveloppe convexe approximative qui contient tous les sommets extrêmes de l'enveloppe convexe réelle ainsi que quelques sommets concaves. Ensuite, en éliminant ces sommets concaves de l'enveloppe convexe approximative en utilisant la méthode monotone d'Andrew [35], qui est une opération géométrique qui se fait dans le sens contraire des aiguilles d'une montre (CCW), nous obtenons l'enveloppe convexe finale. L'algorithme TORCH a une complexité de $O(n \log n)$.

1.1.6 Algorithme de Xing [6]

Cet algorithme a pour objet de calculer l'enveloppe convexe d'un ensemble de points planaires. L'algorithme commence par construire un polygone convexe initial nommé (ICP) en calculant huit points extrêmes, et mesure la largeur et la longueur de l'ICP calculé. Ensuite, il transforme l'ensemble de points dans un nouvel espace en utilisant une transformation affine où la plupart des nouveaux points restent à l'intérieur du nouveau polygone convexe initial (NICP). Après cela, il supprime les points intérieurs qui sont près de la frontière du NICP et applique l'algorithme de Quick-hull aux points restants. Finalement, il transforme les sommets de l'enveloppe convexe trouvée à leurs

coordonnées d'origine afin d'obtenir l'enveloppe convexe finale. Cet algorithme a une complexité de $O(n + n \log n)$.

1.1.7 Algorithme de Mei [7]

Un algorithme accéléré par GPU pour le calcul d'une enveloppe convexe est présenté. L'algorithme commence par éliminer les points qui sont à l'intérieur d'un quadrilatère formé par les quatre points extrêmes. Ensuite, les points restants seront répartis en quatre sous-régions. Les points situés dans la même région seront triés en parallèle selon leurs coordonnées. Par la suite, une nouvelle approche de prétraitement basé sur le tri qui s'appelle (SPA) est effectuée pour éliminer les points internes. Après cela, pour chaque sous-région, une chaîne simple est formée avec les points restants. En reliant les quatre chaînes obtenues dans le sens contraire des aiguilles d'une montre (CCW), un polygone simple est formé. Enfin, l'algorithme de Melkman [36] est appliqué pour calculer l'enveloppe convexe du polygone formé. Cette approche a une complexité de $O(n \log n)$.

1.1.8 Algorithme de Ruano [8]

Il s'agit d'un algorithme d'approximation aléatoire pour des ensembles de données multidimensionnelles. L'algorithme commence par la mise à l'échelle de chaque dimension dans l'intervalle $[-1, 1]$, puis identifie les points minimums et maximums par rapport à chaque dimension. Ces points sont considérés comme des sommets de l'enveloppe convexe initiale. Ensuite, il génère une population de k facettes sur la base des sommets de l'enveloppe convexe initiale, et identifie les points les plus éloignés de chaque facette de la population comme des nouveaux sommets de l'enveloppe convexe. Cette opération est répétée itérativement jusqu'à ce qu'il n'y ait pas de nouveaux sommets trouvés. Enfin, nous obtenons l'enveloppe convexe. La complexité de cet algorithme n'est pas donnée, mais elle dépend de l'ensemble de données à traiter et ses caractéristiques, de la taille de la population (paramètre d'entrée k), du nombre d'itérations, du nombre de sommets de l'enveloppe convexe et de la distribution des points dans l'ensemble de données de départ.

1.1.9 Algorithme S-CH [9]

L'algorithme (S-CH) baptisé Smart Convex Hull, commence par appliquer une méthode de subdivision spatiale afin d'éliminer un maximum de points de départ. Ensuite, il détermine l'enveloppe convexe sur les points restants en appliquant, par exemple, un algorithme standard pour la détermination de l'enveloppe convexe. La complexité de cet algorithme est de $O(n \log n)$.

1.2 L'enveloppe concave ou polygonale

1.2.1 Diviser et fusionner [10]

Cet algorithme commence par une enveloppe convexe de points et obtient l'enveloppe concave en deux étapes: fractionnement suivi d'une fusion. Lors du fractionnement, un ou plusieurs côtés des points de l'enveloppe convexe seront supprimés et de nouveaux côtés seront ajoutés pour prendre soin de la concavité inhérente. Pour obtenir une enveloppe polygonale lisse, deux ou plusieurs côtés sont fusionnés en un seul. La complexité de cet algorithme est $O(nh)$.

1.2.2 La forme d'un ensemble de points [11]

Dans ce travail, les auteurs introduisent le concept de α-*forme* qui représente la *forme extérieure* d'un ensemble de point (convexe ou non). Ils décrivent un algorithme qui permet de trouver la valeur

de α. Pour un ensemble fini P de points dans le plan, la $\alpha - enveloppe$ pour $\alpha \neq 0$ est l'intersection de tous les disques fermés de rayon $1/\alpha$ contenant les points de P (Où pour des valeurs négatives de α un disque fermé de rayon A/α est interprété comme le complément d'un disque ouvert de rayon $-1/\alpha$). Lorsque α approche de 0, la $\alpha - enveloppe$ s'approche de l'enveloppe convexe ordinaire. Par conséquent, l'enveloppe $0 - enveloppe$ est stipulée être l'enveloppe convexe. La $\alpha - forme$ est un graphe linéaire (habituellement un polygone) dérivé directement de l' $alpha - enveloppe$. Lorsque $\alpha = 0$, c'est l'enveloppe convexe. Pour les grandes valeurs négatives de α, il représente l'ensemble initial de points P. La complexité de cet algorithme est $O(n \log n)$.

1.2.3 Extraction des limites perceptives [12]

Cette approche est basée sur une nouvelle définition appelée $s - forme$, qui est obtenue en divisant le plan en un réseau de cellules carrées de longueur latérale s. $S - forme$ est simplement l'union des cellules de réseau contenant tous les points de P. Pour obtenir une enveloppe polygonale, une autre $r - forme$ est définie, où r est obtenu à partir de s. Pour trouver la $r - forme$, l'union de tous les disques de rayon r centrée autour des points de P est prise. Pour des points $p, q \in P$, l'arête $\{p, q\}$ est sélectionné si et seulement si, les limites des disques centrés sur p et q intersectent en un point qui se trouve sur la Limite de l'union de tous les disques. La $r - forme$ de P est alors la réunion des arêtes sélectionnées. La complexité de cet algorithme est $O(n)$.

1.2.4 K-voisin le plus proche [13]

Cet algorithme est utilisé pour calculer une enveloppe concave d'un ensemble de points dans le plan. Il suppose que le point actuellement choisi est connecté à k points les plus proches. Ensuite, il sélectionne le point qui a l'angle polaire minimum par rapport au point courant, comme dans l'algorithme de Jarvis (Section 1.1.2). Une enveloppe concave n'est pas obtenue pour chaque ensemble de points, et la valeur de k doit être adaptée pour chaque cas. Sa complexité est $O(kh^2)$. Notre algorithme proposé utilise le même concept sous l'hypothèse que le graphe est connecté. Cependant, dans notre cas, le nombre k est connu et peut être différent pour chaque point. De plus, les k voisins les plus proches ne sont pas nécessairement connectés.

1.2.5 Mesure de concavité [14]

Cet algorithme vise à déterminer une enveloppe concave multidimensionnelle. Il se compose de quatre étapes. Tout d'abord, un ensemble d'arêtes d'une enveloppe convexe est sélectionné et une valeur de seuil N est choisie. Ensuite, les points intérieurs les plus proches des arêtes de l'enveloppe convexe sont identifiés. La distance la plus courte appelée *distance de décision*, entre ces points intérieurs les plus proches et les arêtes de l'enveloppe convexe est calculée. Troisièmement, une décision de chercher ou non est faite en comparant N à la distance de décision. Si $(longueur\ de\ l'arête)/(distance\ de\ decision) > N$, le processus de recherche est exécuté. Enfin, la deuxième et la troisième étapes sont répétées jusqu'à ce qu'il n'y ait plus de point intérieur à trouver. La complexité de cet algorithme est $O(n \log n + rn)$ où r dépend de la dimension d de l'espace considéré. Par exemple, pour un espace tridimensionnel, r est égal à $d/2$.

1.2.6 Algorithme de Braune [15]

Cet algorithme partitionne un ensemble de données sous forme de clusters (groupes), en se basant sur la notion de l'enveloppe concave. L'idée principale est de commencer à partir de l'enveloppe convexe de l'ensemble de données et de rétrécir itérativement l'enveloppe convexe en remplaçant les arêtes trop longues par de nouvelles arêtes qui ajustent au plus les données, puis, de diviser

récursivement l'enveloppe convexe en deux parties séparées fermées dès qu'ils convergent en un seul point. La complexité en temps de cet algorithme est $O(n \log k)$, ou k est le nombre de points sur l'enveloppe convexe et n le nombre total de points.

1.2.7 Algorithme de Gheibi [16]

Un algorithme de reconstruction de formes qui détermine l'enveloppe concave est présenté. L'algorithme commence à partir de l'enveloppe convexe de l'ensemble de points d'entrée, et le rendre concave progressivement pour atteindre une sortie polygonale réelle. Ainsi, à chaque étape, chaque arête sélectionnée sera remplacée par deux nouvelles arêtes construites de telle sorte que la forme reste un polygone. Le critère de sélection de ces nouvelles arêtes est basé sur une combinaison des facteurs de perception visuelle suivants: diagramme de Voronoi, proximité des points, longueur des arêtes, etc. Ce processus est répété jusqu'à ce que la condition d'arrêt soit satisfaite. La complexité temporelle de cet algorithme est $O(n \log n)$.

1.2.8 Ec-shape Algorithme [17]

Cet algorithme vise à trouver une enveloppe concave qui se rapproche le mieux de la forme géométrique d'un ensemble donné de points dans le plan. Il commence par construire le graphe de triangulation de Delaunay G [37] pour l'ensemble de points. Ensuite, il construit une file d'attente de priorité (PQ) des arêtes externes (EEs) de G par ordre décroissant de longueur. Puis il supprime, à plusieurs reprises, l'arête externe (EE) du triangle externe (ET) de la tête de (PQ) et du graphe courant G, mais seulement s'il satisfait une contrainte de cercle qui est défini, et que $G - EE$ (G sans l'arête EE) est régulier. Une fois que l'arête externe EE est retirée de G, les arêtes adjacentes du triangle externe (ET) sont ajoutées à (PQ) en maintenant toujours l'ordre décroissant par longueurs. Ce processus est répété jusqu'à ce qu'il n'y ait aucune possibilité de supprimer des arêtes externes du graphe G. La complexité temporelle de cet algorithme est $O(n \log n)$.

1.2.9 Gift Opening Algorithme [18]

L'idée de cet algorithme est de trouver l'enveloppe convexe en utilisant n'importe quel algorithme connu, puis, la transformer en une enveloppe concave. Pour ce faire, l'algorithme commence par ajouter toutes les arêtes externes à une liste triée par leurs longueurs. L'arête la plus longue sera sélectionnée et supprimée de la liste. Ensuite, un nouveau point sera sélectionné de sorte qu'il forme le plus petit angle avec les extrémités de l'arête supprimée. Après cela, deux nouvelles arêtes seront créées à partir de ce nouveau point et les extrémités de l'arête supprimées. Ces deux arêtes seront ajoutées à la liste. L'algorithme s'arrête lorsque aucune arête de la liste n'a une longueur supérieure à un seuil prédéfini. La complexité temporelle de cet algorithme est $O(n)$.

1.2.10 RGH Algorithme [19]

L'algorithme (RGH) baptisé Regularized Geometric Hull est utilisé principalement pour la segmentation d'images biomédicales. Il transforme les points d'un ensemble de données en un ensemble de triangles. Chaque triangle dont la longueur de bord maximal est supérieure à un paramètre donné ζ sera supprimé. La valeur de ζ régularise la convexité ou la concavité de l'enveloppe géométrique. La complexité en temps de cette procédure est $O(n^3)$ dans le cas habituel. Cependant, dans le cas où la triangulation serait basée sur l'algorithme de Delaunay, la complexité du temps sera $O(n \log n)$.

D'autres publications peuvent être trouvées dans [38, 39, 40, 41, 42].

La table 1.2.10 résume toutes, les approches proposées dans la section état de l'art concernant les algorithmes de détermination de l'enveloppe convexe ou concave, également en termes de complexité, d'optimalité et de dimensionnalité, où n est le nombre total de points, k un nombre fixe de voisins les plus proches d'un point, g le degré maximum du graphe, h le nombre de points sur l'enveloppe convexe, d la dimension de l'espace considéré, où r est un nombre qui dépend de la dimension d de l'espace considéré, qui est égal à $\frac{d}{2}$ pour un espace tridimensionnel.

2 Présentation de l'algorithme LPCN

Dans cette section, nous présentons la première version de notre algorithme LPCN (Least Polar-angle Connected Node). Nous allons montrer comment modifier l'algorithme de Jarvis, initialement décrit pour trouver une enveloppe convexe, afin de trouver l'enveloppe polygonale d'un ensemble de points d'un graphe connecté.

2.1 Algorithme de Jarvis

L'algorithme de Jarvis [2] détermine l'enveloppe convexe d'un ensemble de points finis. Il ne peut pas être utilisé directement dans notre travail puisque nous recherchons une enveloppe polygonale. Cependant, nous pouvons le modifier en considérant dans chaque itération seulement les points qui sont connectés au point courant au lieu de prendre n'importe quel point.

Nous rappelons que l'algorithme de Jarvis sélectionne à chaque itération le point qui a l'angle minimum avec le segment horizontal gauche passant par le point courant. Cela ne peut fonctionner que d'un seul côté d'un ensemble de points finis. Dans [43], une amélioration de l'algorithme de Jarvis est proposée en divisant l'ensemble des points en deux ensembles et en changeant l'orientation des angles dans chaque ensemble. Une autre amélioration de l'algorithme de Jarvis utilise l'angle polaire $\varphi(P_p, P_c, P_j)$ formé par le point suivant P_j, le point courant P_c et celui trouvé dans l'itération précédente P_p. Par conséquent, l'algorithme de Jarvis peut être décrit comme l'algorithme 1 suivant:

Algorithm 1 Algorithme de Jarvis

1: **procedure** JARVIS(V)
2: $P_c \leftarrow$ Point ayant la coordonnée x minimale
3: $\mathbb{B} \leftarrow \{P_c\}$
4: $P_{first} \leftarrow P_c$
5: $P_p \leftarrow$ Point fictif situé à gauche de P_{first}
6: **repeat**
7: $P_k = \underset{P_j \in V}{\text{argmin}}\{\varphi(P_p, P_c, P_j)\}$
8: $\mathbb{B} \leftarrow \mathbb{B} \cup \{P_k\}$
9: $P_p \leftarrow P_c$
10: $P_c \leftarrow P_k$
11: **until** $P_k = P_{first}$
12: **return** $\mathbb{B}$
13: **end procedure**

Soit V un ensemble de n points et h le nombre de points définissant l'enveloppe convexe de V. L'algorithme trouve les points P_0, P_1, ..., P_{h-1} de l'enveloppe convexe, ordonnée un par un, où P_0 est le point avec la coordonnée x minimale. Étant donné le point P_i, nous voulons trouver

Algorithme	Enveloppe convexe	Enveloppe Concave	Complexité	Optimalité	Dimension	Observations
Graham [1]	✓		$O(n \log n)$	-	$2D$	
Jarvis [2]	✓		$O(nh)$	-	$2D$	
Quick-hull [3]	✓		$O(n^2)$	-	$2D$	
Incremental [4]	✓		$O(n^{(d+1)/2})$	-	Multidimensionnel	
TORCH [5]	✓		$O(n \log n)$	-	$2D$	
NICP [6]	✓		$O(n + n \log n)$	-	$2D$	Nécessite l'algorithme Quick-hull
Mei [7]	✓		$O(n \log n)$	-	$2D$	GPU-accelerated convex hull algorithm
Ruano et al. [8]	✓		[not given]	-	Multidimensionnel	La complexité dépend de nombreux paramètres
S-CH [9]	✓		$O(n \log n)$	-	$3D$	Nécessite un algorithme standard de l'enveloppe convexe
Split and Merge [10]		✓	$O(nh)$	Non Optimal	$2D$	Démarre à partir de l'enveloppe convexe des points
Alpha-Shape [11]		✓	$O(n \log n)$	Non Optimal	$2D$	Dépend d'un paramètre α
PBE [12]		✓	$O(n)$	Non Optimal	$2D$	
KNN [13]		✓	$O(nh^2)$	Non Optimal	$2D$	
CM [14]		✓	$O(n \log n + rn)$	Non Optimal	Multidimensionnel	
Braune et al. [15]		✓	$O(n \log h)$	Non Optimal	$2D$	Démarre à partir de l'enveloppe convexe des points
Gheibi et al. [16]		✓	$O(n \log n)$	Non Optimal	$2D$	Démarre à partir de l'enveloppe convexe des points
EC-Shape [17]		✓	$O(n \log n)$	Non Optimal	$2D$	Nécessite la triangulation de Delaunay (DT)
Gift Opening [18]		✓	$O(n)$	Non Optimal	$2D$	Démarre à partir de l'enveloppe convexe des points
RGH [19]		✓	$O(n^3)$	Non Optimal	$2D$	
LPCN		✓	$O(kh^2)$	Optimal	$2D$	Nécessite seulement un point de départ

le point P_{i+1}, consécutif à P_i sur l'enveloppe, pour chaque $1 \leq i \leq (n-2)$. Ce point est tel que l'arête $\{P_i, P_{i+1}\}$ a l'angle polaire minimum avec l' arête $\{P_i, P_{i-1}\}$. L'algorithme s'arrête une fois que nous revenons au point P_0. La Figure 1.2 montre un exemple illustrant cet algorithme. Nous considérons un ensemble de 8 points (voir Figure 1.2(a)) et nous partons du point A. Le point d'angle polaire minimum par rapport à A est le point C (voir Figure 1.2(b)). Ensuite, nous définissons le point d'angle polaire minimum par rapport à C et A qui est le point E (voir Figure 1.2(c)). De la même façon, nous définissons le point d'angle polaire minimum par rapport à E et C qui est le point H. Le dernier point défini après H est A qui était le point de départ de la première itération. On trouve donc l'enveloppe convexe (voir Figure 1.2(d)).

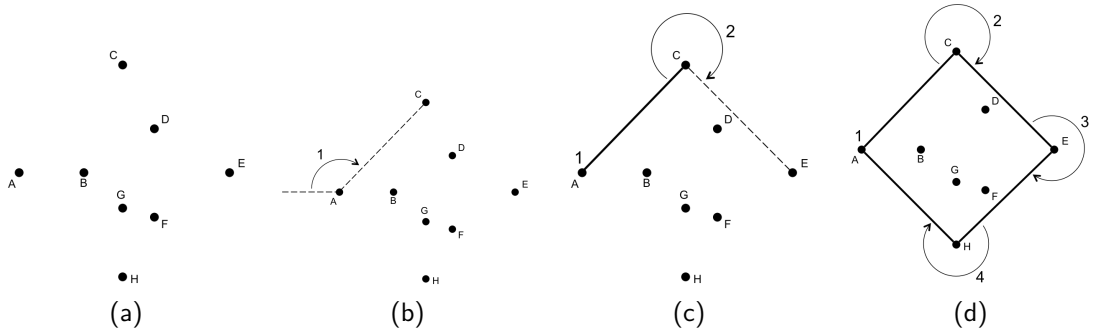


Figure 1.2: Algorithme de Jarvis pour (a) l'ensemble de points dans le plan, (b) itération 1, (c) itération 2, (d) dernière itération.

2.2 LPCN1: la première version

Étant donné un graphe Euclidien connecté $G = (V, E)$, la première version de notre algorithme LPCN (Least Polar-angle Connected Node) diffère de l'algorithme de Jarvis dans la sélection du point (ou sommet) qui suit le point courant: Le point suivant ne peut être qu'un voisin dans G avec le point courant. Cette version fonctionne pour les graphes planaires ou pour les graphes non planaires à l'exception de quelques sous-graphes problématiques qui seront présentés dans la Section 3.

L'algorithme LPCN1, commence à partir d'un point qui appartient à l'enveloppe polygonale et peut-être décrit comme suit. Soit V l'ensemble des n points de G et h le nombre de points de l'enveloppe polygonale finale de V. L'algorithme trouve d'abord les points P_0, P_1, ..., P_{h-1} de l'enveloppe polygonale, où P_0 est un point avec la coordonnée x minimale. Soit le point P_i, on veut trouver le prochain point consécutif P_{i+1} sur l'enveloppe, qui est connecté à P_i, pour chaque $1 \leq i \leq k-1$. Ce point P_{i+1} est celui qui a l'angle polaire minimum par rapport à P_i. Lorsque nous atteignons le dernier point P_{h-1}, nous avons construit une enveloppe polygonale de V. L'algorithme s'arrête dès que nous revenons au point P_0. Il est présenté comme l'algorithme 2, dans lequel $N(P_c)$ représente le voisinage de P_c dans G.

La Figure 1.3 montre comment fonctionne l'algorithme. Considérons le graphe de la Figure 1.3(a) où $V = \{A, B, C, D, E, F, G, H\}$. Tout d'abord, nous choisissons le point P_c ayant la cordonnée x minimale . Dans cet exemple, ce point est A ($P_c = A$) qui est également le premier point de l'enveloppe polygonale $\mathbb{B}_V = \{A\}$.

L'algorithme s'arrête lorsque le point suivant P_k de l'enveloppe polygonale est égal au premier point de l'enveloppe polygonale P_{first}, dans cet exemple, c'est le point A (c'est-à-dire, $P_{first} = A$).

Algorithm 2 LPCN1: La première version de l'algorithme LPCN

procedure $\mathrm{LPCN1}(V, E)$
  $P_c \leftarrow$ Point ayant la coordonnée x minimale
  $\mathbb{B}_V \leftarrow [P_c]$
  $\mathbb{B}_E \leftarrow \emptyset$
  $P_{first} \leftarrow P_c$
  $P_p \leftarrow$ Point fictif situé à gauche de P_{first}
  repeat
    $P_v \leftarrow \underset{P_j \in N(P_c)}{\text{argmin}} \{\varphi(P_p, P_c, P_j)\}$
    $\mathbb{B}_V \leftarrow \mathbb{B}_V \cup \{P_v\};\ \mathbb{B}_E \leftarrow \mathbb{B}_E \cup \{\{P_c, P_v\}\}$
    $P_p \leftarrow P_c$
    $P_c \leftarrow P_v$
  until $P_v = P_{first}$
  return $\mathbb{B}_V$, $\mathbb{B}_E$
end procedure

Dans la première itération, P_c est toujours égal à P_{first}. Pour démarrer l'algorithme, nous pouvons considérer un point fictif A' qui a une coordonnée x plus petite que celle de A ($P_p = A'$). Notez que d'autres points fictifs peuvent être considérés. La Figure 1.3(b) montre les différents points de départ et leurs voisins fictifs conçus par des points gris.

Ensuite, nous devons trouver l'angle polaire minimum formé par l'arête $\{P_c, P_p\}$ (c'est-à-dire, $\{A, A'\}$) et les arêtes formées par P_c avec chacun de ses voisins (c'est-à-dire, $\{A, B\}$, $\{A, C\}$, $\{A, D\}$ et $\{A, G\}$). Dans cet exemple, le voisin obtenu est $P_k = C$ (voir Figure 1.3(c)). Alors, la première arête de l'enveloppe polygonale recherchée est $\mathbb{B}_E = \{\{A, C\}\}$. Par conséquent, $\mathbb{B}_V = \{A, C\}$.

Dans la prochaine itération, nous appliquerons la même procédure en recherchant l'angle polaire minimum formé par l'arête $\{C, A\}$ et les arêtes formées par C avec ses voisins, c'est-à-dire, $\{C, B\}$ et $\{C, D\}$. Le point obtenu est D (voir Figure 1.3(d)) et l'enveloppe polygonale courante est maintenant donnée par $\mathbb{B}_V = \{A, C, D\}$ avec $\mathbb{B}_E = \{\{A, C\}, \{C, D\}\}$. De la même manière, nous déterminons les autres points. Nous avons trouvé E (voir Figure 1.3(e)), H (voir Figure 1.3(f)), G (voir Figure 1.3(g)) et enfin A (voir Figure 1.3(h)). Puisque $A = P_{first}$, nous arrêtons l'algorithme. L'enveloppe polygonale obtenue est donnée par $\mathbb{B}_V = \{A, C, D, E, H\}$ et $\mathbb{B}_E = \{\{A, C\}, \{C, D\}, \{D, E\}, \{E, H\}, \{H, A\}\}$ (voir Figure 1.3(i)).

Cet algorithme ne fonctionne pas en présence des cas suivants:

- nœuds de degré un,
- graphe d'ancrage et graphe en bateau (Figures 1.5 et 1.10).

Ces deux cas peuvent conduire à une situation de boucle infinie. Dans la section suivante, nous allons décrire toutes les situations conduisant notre premier algorithme à l'échec, et nous allons montrer comment les éviter.

3 Sous-graphes problématiques et LPCN2

LPCN1 présenté comme l'algorithme 2 dans la Section précédente 2.2 ne fonctionne correctement que si le graphe considéré ne contient pas de sous-graphes problématiques pouvant mener soit à

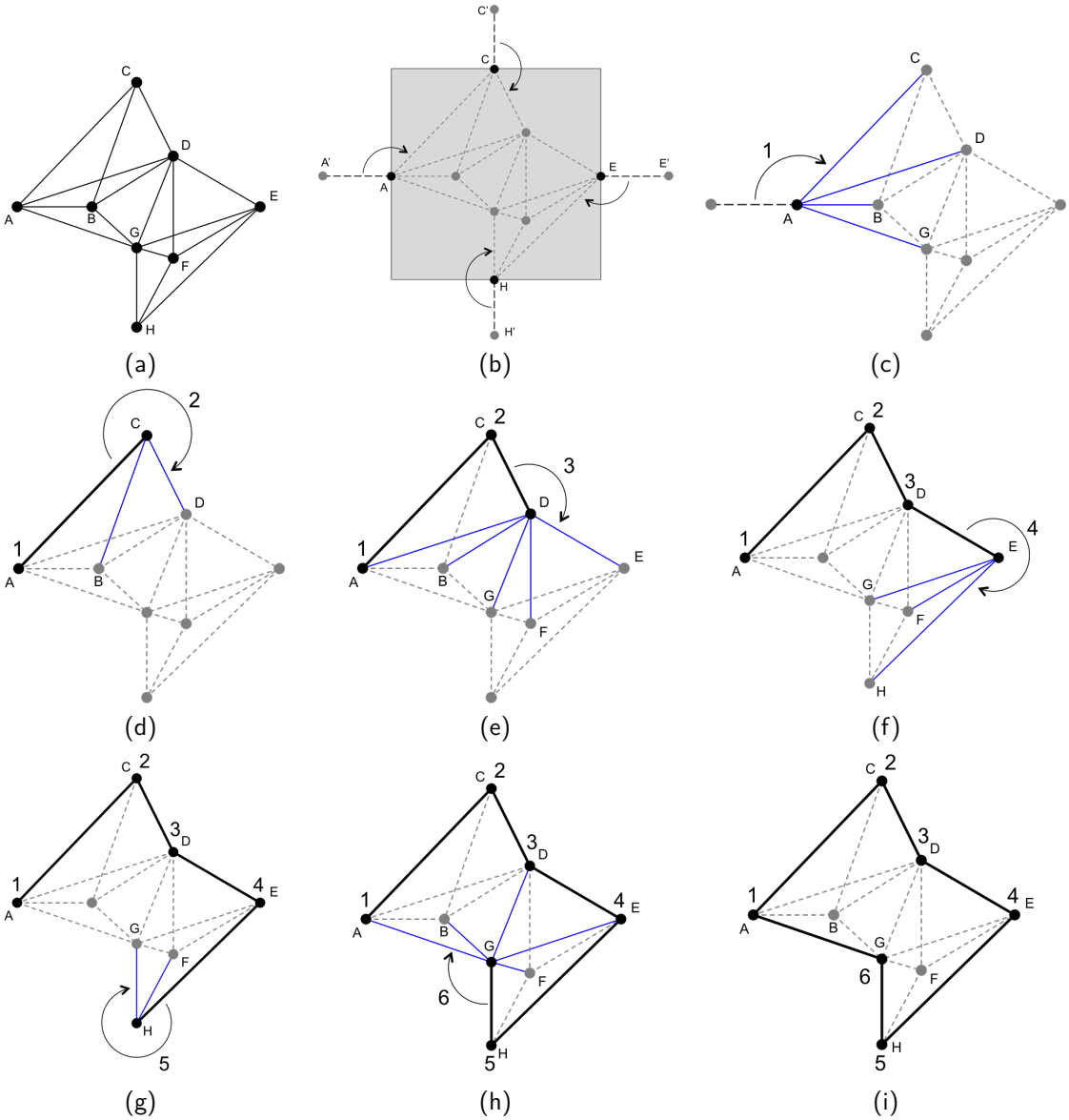


Figure 1.3: Illustration de l'algorithme LPCN.

une étape de blocage, soit à une solution non optimale.

3.1 Cas 1: Angle de zéro degré

Pour trouver le point suivant sur une enveloppe polygonale, les angles sont calculés entre les arêtes $\{P_i, P_{i-1}\}$ et $\{P_i, P_{i+1}\}$, avec $P_{i+1} \in N(P_i)$, et le point pour lequel l'angle calculé est le plus petit sera choisi comme suivant. Puisque le point précédent est parmi les voisins du point courant, ce sera celui qui sera choisi, car l'angle formé par les arêtes correspondantes est égal à zéro. Cela conduit à une situation de blocage. Pour surmonter cette limitation, nous considérons cet angle comme 360^o au lieu de 0^o. La même solution peut être utilisée pour le cas de nœuds ayant un seul voisin. La Figure 1.4 montre cette situation. En prenant en compte la modification proposée, la frontière obtenue est alors donnée par $\mathbb{B}_V = \{A, B, C, E, D\}$ et $\mathbb{B}_E = \{\{A, B\}, \{B, C\}, \{C, E\}, \{E, C\}, \{C, D\}, \{D, A\}\}$.

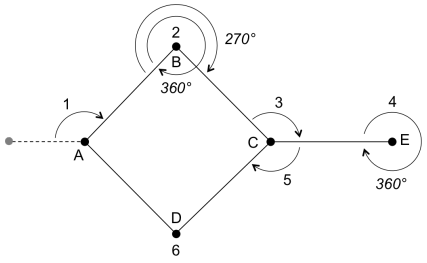


Figure 1.4: De 0^o à 360^o angles.

3.2 Cas 2: Graphe en bateau

Figure 1.5 montre un exemple de *Graphe en bateau* formé par les arêtes $\{A, C\}$, $\{A, D\}$ et le triangle BCD. On obtient un *Graphe généralisé en bateau* de la même manière si on considère une subdivision de $\{A, D\}$.

Lorsque nous exécutons l'algorithme LPCN1 présenté ci-dessus en commençant ou en arrivant au point B, alors quelques situations problématiques surgiront.

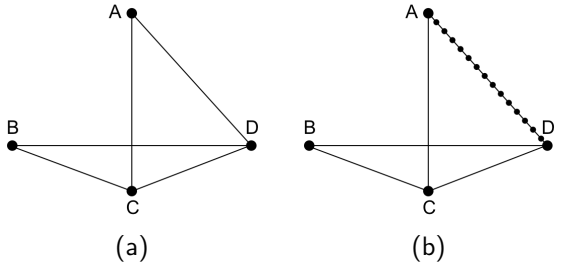


Figure 1.5: Graphe en bateau avec des points possibles entre A et D.

Pour expliquer comment ce type de graphiques peut être troublant, prenons deux exemples avec graphe en bateau. La Figure 1.6 montre le premier exemple d'un graphe en bateau qui est formé par les arêtes $\{E, C\}$, $\{E, D\}$, et le triangle BCD. L'algorithme commence à partir du point B

puis passe à D. De D il va à E puis à C. De C il retourne à D. Cependant, cette fois, il ne passera pas à E mais à B à nouveau. Parce que, nous partons du point B, l'algorithme s'arrêtera, et malheureusement, il ne visitera pas l'arête $\{E, F\}$ ni les points et les arêtes qui sont entre F et B. Par conséquent, dans ce cas, l'enveloppe polygonale ne peut pas être trouvée.

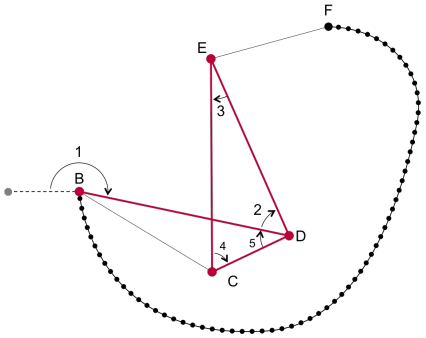


Figure 1.6: Graphe en bateau (exemple 1).

La Figure 1.7 montre le second exemple où l'algorithme commence par le point gauche A puis passe à B, qui est un point de départ d'un graphe en bateau formé par les arêtes $\{E, C\}$, $\{E, D\}$ et le triangle BCD. De B l'algorithme va à D, à E puis à C. De C, il retourne à D. De D, cette fois, il va à B, puis à C et à E à nouveau. Enfin, il continue à F et aux autres points. Dans ce cas, il n'y a pas de situation de blocage mais la solution n'est pas optimale. Elle est donnée par $\mathbb{B}_V = \{A, B, D, C, E, F\}$ et $\mathbb{B}_E = \{\{A, B\}, \{B, D\}, \{D, E\}, \{E, C\}, \{C, D\}, \{D, B\}, \{B, C\}, \{C, E\}, \{E, F\}\}$ avec 6 sommets et 9 arêtes, alors qu'une solution optimale est donnée par $\mathbb{B}_V = \{A, B, D, E, F\}$ et $\mathbb{B}_E = \{\{A, B\}, \{B, D\}, \{D, E\}, \{E, F\}\}$ a 5 sommets et 4 arêtes.

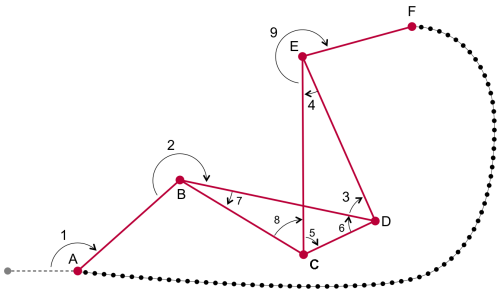


Figure 1.7: Graphe en bateau (exemple 2).

La solution précédente peut être améliorée en éliminant l'arête $\{E, C\}$, qui est justifié par son intersection avec l'arête $\{B, D\}$, qui conduit elle-même à un point C à l'intérieur de l'enveloppe polygonale $\{A, B, D, E, F\}$, comme le montre l'exemple de la Figure 1.8. Cette dernière enveloppe polygonale sera considérée comme une solution améliorée par rapport à la précédente. Ensuite, cette solution est optimale, et il n'y a pas de meilleure solution puisque B n'est pas connecté à E.

Nous pouvons noter que si les distances doivent être prises en compte, et que la distance totale doit être minimisée, on peut prendre la solution $\mathbb{B}_V = \{A, B, C, E, F\}$ si la longueur du chemin $[B, D, E]$ est plus courte que celle du chemin $[B, C, E]$. Notez que, dans ce cas, la règle de l'angle polaire minimum n'est pas respectée.

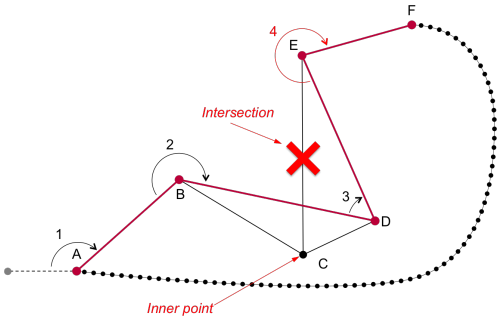


Figure 1.8: Graphe en bateau (solution).

Nous concluons de ce premier cas que toute arête de l'enveloppe polygonale ne doit pas se croiser avec aucune autre.

3.3 Cas 3: Intersection d'arêtes

Nous concluons par le Cas 2 que deux arêtes de l'enveloppe polygonale ne doivent pas se croiser en dehors du point d'extrémité correspondant au prochain sommet choisi par l'algorithme. À titre d'exemple, la Figure 1.9(a) montre que l'acceptation des arêtes intersectées $\{A, B\}$ et $\{E, F\}$ de l'enveloppe polygonale conduira à des sommets non visités sur l'enveloppe polygonale, qui sont I, J et G. Cependant, si cette intersection est considérée alors, comme montrée par la Figure 1.9(b), tous les sommets de l'enveloppe polygonale sont visités (c-à-d, A, B, C, D, E, G, H, I, J). De plus, comme le sommet suivant choisi est B, cette intersection ne doit pas considérer le sommet B de l'arête $\{D, B\}$. C'est-à-dire si nous avons deux arêtes $\{A, B\}$ et $\{C, D\}$ et si l'intersection avec B résulte en un des sommets A, B, C ou D alors, elle ne sera pas considérée comme une intersection. Cette situation est justifiée par les Figures 1.9(c) et (d). Dans la première Figure, si nous acceptons l'intersection normale $\{B, C\} \cap \{D, B\} = \{B\}$, l'algorithme choisira un sommet différent de B. Ensuite, il choisira le prochain sommet C, qui mènera à une boucle infinie B, C et D. Cependant, si l'on considère l'intersection sans les extrémités des arêtes, alors quand l'algorithme revient à B, aucune intersection n'est détectée et l'algorithme choisira le sommet A comme le montre la Figure 1.9(d).

3.4 Cas 4: Graphe d'ancre

Figure 1.10 montre un exemple de *Graphe d'ancre* formé par l'arête $\{A, C\}$ et le triangle BCD. Il diffère simplement du *Graphe en bateau* par l'arête manquante $\{A, D\}$. Exécuter l'algorithme LPCN1 présenté ci-dessus dans des graphes contenant un graphe d'ancre peut mener à une solution non optimale comme nous allons le montrer maintenant.

Comme premier exemple, considérons le cas où le graphe est lui-même un graphe d'ancre. Si l'algorithme part du point B, la solution que nous trouverons par notre algorithme est $\mathbb{B}_V = \{B, D, C\}$ et $\mathbb{B}_E = \{\{B, D\}, \{D, C\}, \{C, B\}\}$. On peut voir que dans cette solution, comme illustré par la Figure 1.11(a), le point A ne peut pas être atteint. Cependant, si nous partons du point A, comme le montre la Figure 1.11(b), on peut l'atteindre. Notez que l'arête sélectionnée à une itération n'a pas à se croiser avec les arêtes de l'enveloppe polygonale obtenue dans les itérations précédentes.

Un second exemple lié à un graphe d'ancre est représenté par la Figure 1.12. Nous pouvons

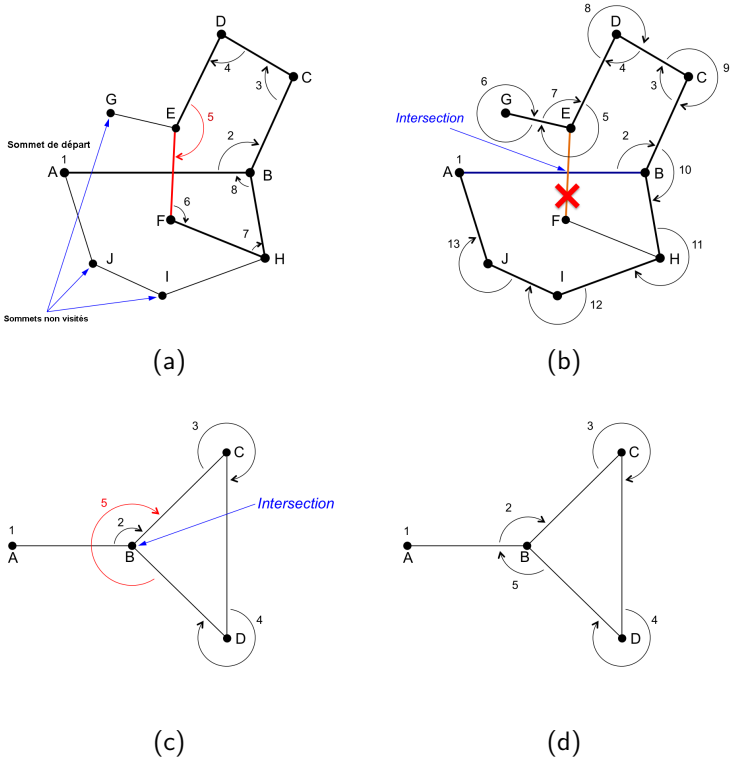


Figure 1.9: Intersection d'arêtes.

constater que si nous partons du point H, A ne peut pas être atteint. Mais si nous partons du point A, la Figure 1.13 montre qu'il est possible de trouver l'enveloppe polygonale correcte en revenant à A. Nous concluons de cette situation que l'algorithme doit partir de tous les points frontière ayant un seul voisin.

Notez qu'à l'itération 3, nous n'allons pas du sommet D au sommet B parce que l'arête $\{D, B\}$ intersecte avec l'arête $\{A, C\}$. Dans ce cas, notre solution est évidemment correcte. Cependant, si sur l'enveloppe polygonale il existe un autre graphe d'ancre, comme le montre la Figure 1.14, alors le sommet J ne peut pas être atteint par l'algorithme. Si une autre partie du graphe part du sommet J comme le montre la Figure 1.15, cette partie ne sera pas atteinte par l'algorithme. Nous concluons que tous les sommets des arêtes qui se croisent avec les arêtes de l'enveloppe polygonale trouvée, peuvent être atteints en exécutant l'algorithme une autrefois sur les sommets non visités et en connectant le sommet J à l'autre partie de l'ancre.

3.5 Cas 5: Premier et dernier nœud de frontière (condition d'arrêt)

Nous avons déjà supposé que l'algorithme s'arrête lorsque le dernier nœud choisi est égal au nœud de départ. Cela signifie que l'algorithme se termine lorsque le premier nœud choisi est sélectionné une seconde fois. Cependant, les deux cas présentés dans la Figure 1.16 (la présence d'un graphe d'ancre et la Figure 1.17 (la présence de plusieurs branches descendantes du nœud de départ) Montrent qu'il est possible de revenir au nœud de départ sans visiter tous les nœuds de bordure. Comme on peut le voir, dans la première figure, l'ensemble des nœuds situés entre les nœuds A et B n'est pas atteint

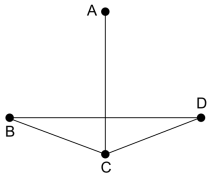


Figure 1.10: Graphe d'ancre.

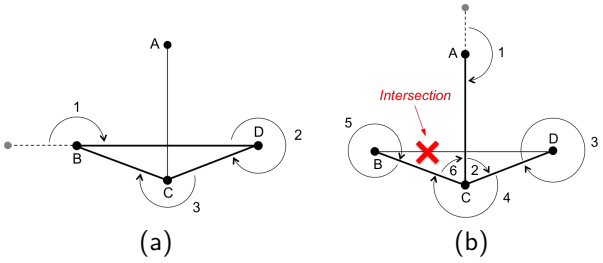


Figure 1.11: Graphe d'ancre (Situation problématique 1).

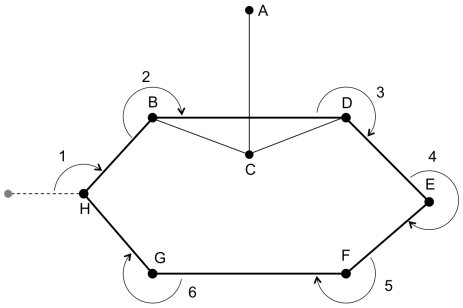


Figure 1.12: Graphe d'ancre (Situation problématique 2).

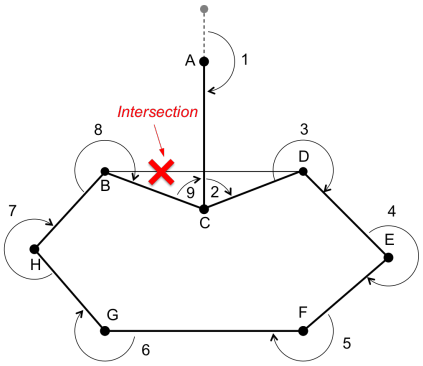


Figure 1.13: Graphe d'ancre (Situation problématique 3).

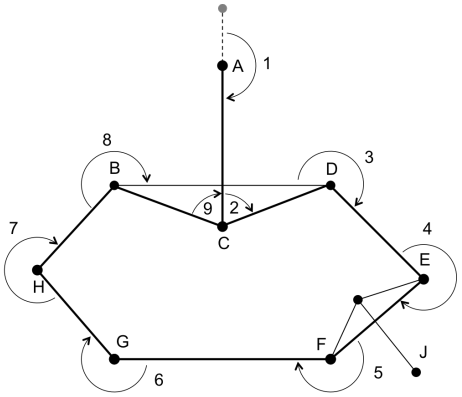


Figure 1.14: Graphe d'ancre (Situation problématique 4).

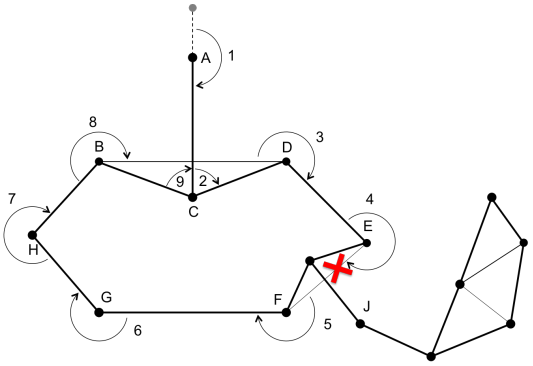


Figure 1.15: Graphe d'ancre (Situation problématique 5).

par l'algorithme. Dans la deuxième figure, seuls les nœuds de la première branche descendante sont atteints. Par conséquent, la condition d'arrêt n'est pas basée sur le nœud de départ, mais sur le deuxième nœud de frontière visité, car chaque nœud ne peut pas sélectionner le même angle minimum deux fois, comme le confirme le Corollaire 1 dans la Section 4.

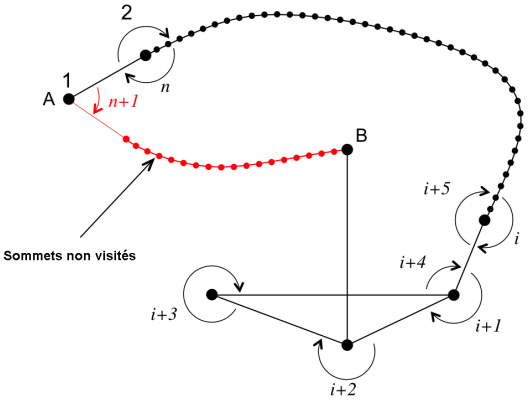


Figure 1.16: Premier et dernier nœud de frontière (condition d'arrêt): cas 1.

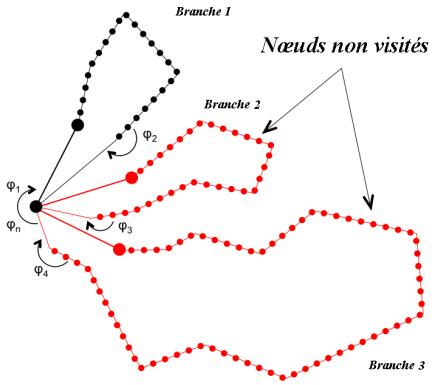


Figure 1.17: Premier et dernier nœud de frontière (condition d'arrêt): cas 2

3.6 Cas 6: Trois points sur la même ligne

Si, en plus, les trois points du triangle d'un graphe d'ancre sont situés sur la même ligne (voir Figure 1.18), alors du point B les deux points D et C peuvent être choisis. Dans ce cas, le point avec la distance minimale doit être choisi. Dans le cas de la Figure 1.18, par exemple, le point C doit être sélectionné. Cependant, si ces trois points n'appartiennent pas à un graphe d'ancre, le point avec la distance maximale doit être choisi pour garantir la cardinalité minimale de $\mathbb{B}_V$. Cette situation est illustrée par la Figure 1.19 ou dans (a), lorsque le sommet C est choisi, quatre sommets frontière sont obtenus. Néanmoins, dans (b), on choisit seulement trois sommets frontière si le sommet D est directement choisi de B, parce qu'il est plus éloigné de B que C de B.

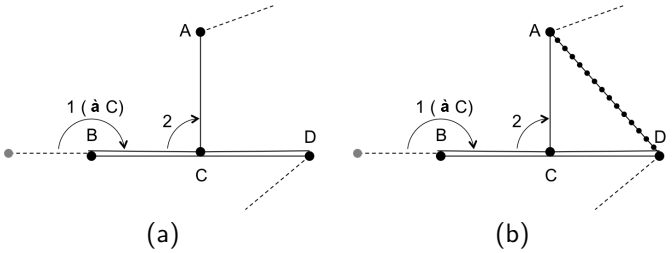


Figure 1.18: graphe d'ancre (a) et graphe en bateau (b) avec trois points du triangle sur la même ligne.

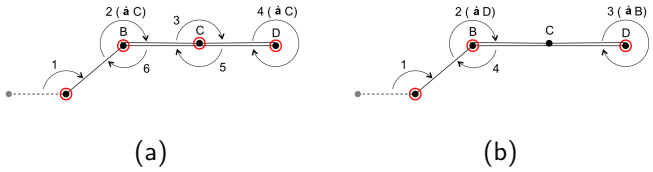


Figure 1.19: Trois points sur la même ligne.

Algorithm 3 LPCN2: La deuxième version de l'algorithme LPCN.

procedure $\mathrm{LPCN}(V, E)$
  $P_0 \leftarrow P_c \leftarrow$ Point ayant la coordonnée x minimale
  $P_p \leftarrow$ Point fictif situé à gauche de P_c
  $\mathbb{B}_V \leftarrow \{P_c\}$
  $\mathbb{B}_E \leftarrow \emptyset$
  $once \leftarrow true$
  repeat
    $\mathbb{A} \leftarrow \{P \in N(P_c)/\mathbb{B}_\mathbb{E} \cap \{\{P_c, P\}\} = \emptyset\}$
    $P_{min} \leftarrow \underset{P \in \mathbb{A}}{\text{argmin}}\{\varphi(P_p, P_c, P)\}$
    $\mathbb{B}_V \leftarrow \mathbb{B}_V \cup \{P_{min}\}$
    $\mathbb{B}_E \leftarrow \mathbb{B}_E \cup \{\{P_c, P_{min}\}\}$
    $P_p \leftarrow P_c$
    $P_c \leftarrow P_{min}$
    if $(once = true)$ **then**
      $once \leftarrow false$
      $P_{first} \leftarrow P_{min}$
    end if
  until $((P_c = P_0)$ and $(P_{min} = P_{first}))$
  return $\mathbb{B}_V, \mathbb{B}_E$
end procedure

Nous compléterons cette section en discutant un autre exemple traitant le graphe d'ancre. La Figure 1.20 montre cet exemple. Comme nous pouvons le voir, la marche obtenue n'est malheureusement pas fermée. Cette solution n'est pas correcte, mais si l'on suppose que l'objectif est simplement de visiter les sommets d'une enveloppe polygonale au lieu de trouver sa forme géométrique, cette solution peut être considérée comme acceptable et donc l'algorithme LPCN2 (Algorithme 3) peut être utilisé dans ce cas.

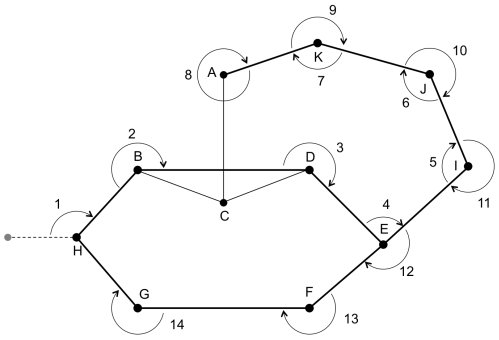


Figure 1.20: Un cas particulier impliquant un graphe d'ancre (solution 1).

Cependant, si nous voulons trouver le polygone géométrique, nous considérons la solution indiquée dans la Figure 1.21. Dans ce cas, LPCN2 doit être modifié comme suit. Nous exécutons les mêmes itérations de 1 à 8 comme dans la Figure 1.20. Le sommet suivant sera C qui mène à une arête $\{A, C\}$ qui intersecte avec l'arête de frontière $\{B, D\}$. Maintenant, au lieu d'éliminer cette arête et chercher une autre comme dans l'algorithme précédent, nous testons si ces deux arêtes forment un graphe d'ancre. Si tel est le cas, nous reviendrons au premier sommet de ce graphe d'ancre, dans ce cas le sommet B. Et puis, nous choisirons le sommet C du graphe d'ancre au lieu du sommet D. Enfin, nous continuerons avec l'itération 9 comme montré dans la Figure 1.21. En effet, cela ajoutera une complexité insignifiante au cas où nous considérons une application du domaine des réseaux de capteurs sans fil (RCSFs), où cette situation est très rare.

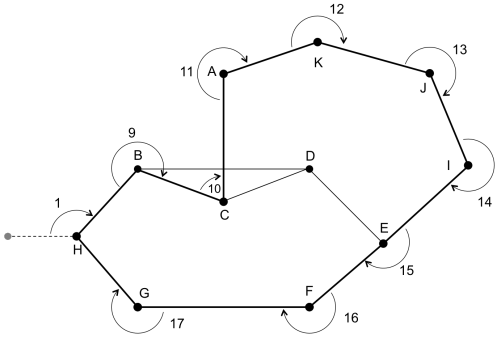


Figure 1.21: Un cas particulier impliquant un graphe d'ancre (solution 2).

Nous terminons cette section avec d'autres sous-graphes problématiques que nous appelons *Pseudo-Graphe en bateau* et *Pseudo-Graphe d'ancre* comme le montrent les Figures 1.22(a), 1.22(b), 1.22(c) et 1.22(d). Ces graphiques donnent les mêmes résultats que ceux présentés ci-dessus pour le graphe en bateau et le graphe d'ancre. Cependant, si nous remplaçons le graphe d'ancre de la

Figure 1.20 par le *Pseudo-Graphe d'ancre* de la Figure 1.22(b) alors ni le polygone géométrique ni la solution représentée par la Figure 1.21 peut être obtenue (voir Figure 1.23). Cependant, pour ce cas, il est possible d'envisager la solution de la Figure 1.20.

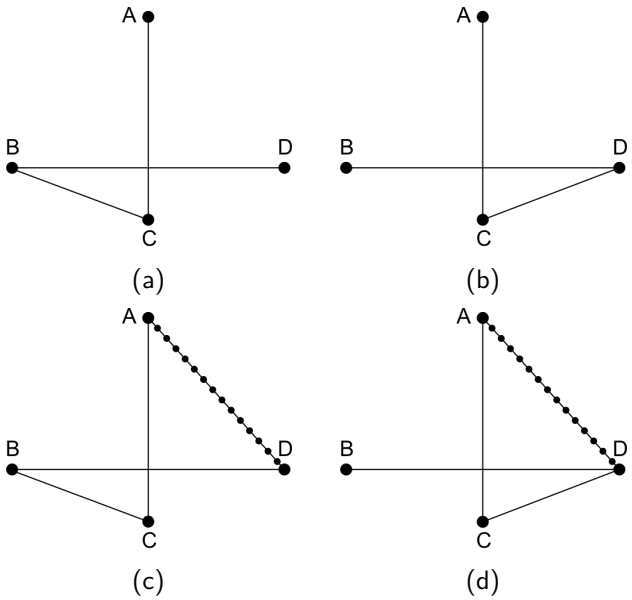


Figure 1.22: Pseudo-Ancre ((a), (b)) et Pseudo-Bateau ((c), (d)) graphes.

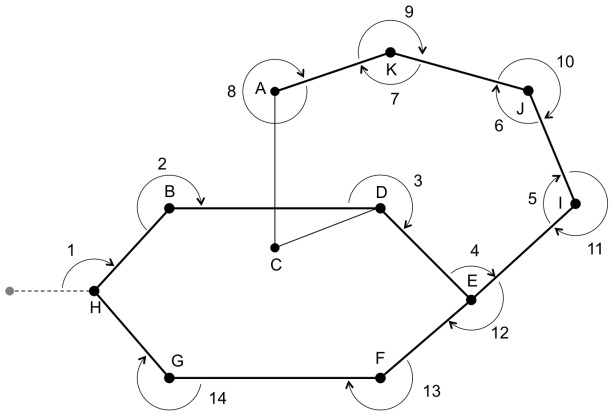


Figure 1.23: Un cas particulier impliquant un graphe pseudo-ancre.

4 Validation de l'algorithme

Dans cette section, nous validons l'exactitude de l'algorithme proposé concernant la convergence et l'optimalité.

Théorème 1.

1. *L'algorithme $LPCN1$ ne calcule jamais le même angle polaire plus d'une fois.*
2. *L'algorithme $LPCN2$ ne calcule jamais le même angle polaire plus d'une fois.*

Démonstration.

1. On suppose que l'algorithme part du sommet P_{first} et à l'itération i, sans perte de généralité, on peut représenter l'ensemble courant des sommets de l'enveloppe polygonale $\mathbb{B}_V^i = \{P_{first}, ..., P_i\}$ par un sommet $P_{\mathbb{B}_V}^i$. D'autre part, nous supposons que les angles ne sont calculés que dans le sens des aiguilles d'une montre. Par conséquent, tout angle, tel que donné par deux arêtes spécifiques, ne sera calculé qu'une seule fois. Sinon, si un angle est calculé une seconde fois, il doit passer une seconde fois à travers le sommet $P_{\mathbb{B}_V}^i$. Cela signifie que nous passons une seconde fois à travers le sommet de départ P_{first} qui représente la condition d'arrêt de LPCN1 (voir ligne 12 de l'algorithme 2). Cela conduit à une contradiction et donc il n'est pas possible de calculer un angle plus d'une fois.
2. Pour les graphes planaires, $LPCN2$ se comporte de la même manière que $LPCN1$. Pour les graphes non planaires, et dans le cas où le croisement d'arête est détecté, la condition ajoutée à la ligne 9 du LPCN2 (Algorithme 3) empêche l'algorithme de revenir à un angle déjà visité.

□

Corollaire 1.

1. *L'Algorithme $LPCN1$ ou l'Algorithme $LPCN2$ ne choisit jamais le nœud suivant de frontière du même nœud de frontière plus d'une fois.*

Démonstration.

1. Ceci est une conséquence directe du théorème 1, car tout nœud de frontière est sélectionné en fonction du calcul d'un angle. Supposons que le point v soit sélectionné deux fois à partir du point v_c en calculant l'angle minimum $\varphi(v_p, v_c, v)$. Cela signifie que cet angle est visité deux fois, ce qui contredit le théorème 1.

□

Corollaire 2.

1. *L'algorithme $LPCN2$ visite le point v_i au plus $d(v_i)$ fois.*
2. *L'algorithme $LPCN1$ visite le point v_i au maximum $d(v_i)$ fois.*

Démonstration.

1. Tout sommet $v_i \in V$ a $d(v_i)$ angles polaires. Puisque $LPCN2$ calcule l'angle d'un sommet donné au maximum une fois, par le Théorème 1 le nombre de fois où l'algorithme scanne le sommet v_i ne peut pas dépasser le nombre d'angles polaires. Par conséquent, l'algorithme visite n'importe quel sommet v_i au plus $d(v_i)$ fois.

2. En l'absence d'arêtes croisées (c-à-d, si G est planaire), $LPCN1$ se comporte de la même manière que $LPCN2$.

□

Corollaire 3. *L'algorithme $LPCN1$ ou $LPCN2$ trouve une solution dans un nombre fini d'étapes.*

Démonstration. La preuve résulte directement du Théorème 1. En effet, puisque le graphe est connecté et qu'il contient un nombre fini de points pour former un nombre fini d'angles qui sont calculés au plus une fois par $LPCN1$ et $LPCN2$, par le Théorème 1 le nombre d'itérations des deux algorithmes est fini. □

Théorème 2. *L'algorithme $LPCN1$ ou $LPCN2$, trouve l'enveloppe polygonale avec un nombre minimum de sommets,sans rencontrer un graphe d'ancre.*

Démonstration. Considérons la solution donnée par $\mathbb{B}_V$. Si cette solution n'est pas optimale, il existe une autre enveloppe polygonale $\mathbb{B}'_V$ contenant moins de sommets que $\mathbb{B}_V$. Donc:

$$\mathbb{B}_V = \mathbb{B}'_V \cup \chi$$

Pour un sommet $v \in \chi$ deux cas sont possibles:

- Cas 1: v n'est pas un sommet de frontière. Ce cas est impossible car v doit satisfaire la condition $v = \underset{P_j \in N(P_c)}{\arg\min} \{\varphi(P_p, P_c, P_j)\}$ car il s'agit d'un élément de $\mathbb{B}_V$, où P_c est un sommet voisin de v. Ainsi, v est aussi un sommet de l'enveloppe polygonale.
- Cas 2: v est un sommet de l'enveloppe polygonale. Puisque, v n'est pas un élément de $\mathbb{B}'_V$, cette dernière n'est pas une enveloppe polygonale de G.

Par conséquent, l'ensemble $\mathbb{B}_V$ représente une enveloppe polygonale de G avec un nombre minimum de sommets. □

Du Corollaire 3, on déduit que les deux algorithmes proposés sont convergents. D'après le Théorème 2, nous concluons que nos algorithmes proposés déterminent une enveloppe polygonale avec un nombre minimum de sommets, ce qui valide entièrement nos algorithmes.

5 Résultats de la simulation

Considérons un graphe avec n sommets et une enveloppe polygonale de h sommets, et soit k le degré maximum de G. Alors la complexité de LPCN1 est $O(kh)$. Pour le cas de LPCN2, puisque dans chaque itération nous recherchons une intersection avec les arêtes du polygone trouvées dans les itérations précédentes, nous devons prendre en compte le facteur h. Ainsi, la complexité de cet algorithme est $O(kh^2)$.

Tous les algorithmes ont été programmés en Java et implémentés dans *CupCarbon* [44], un simulateur de réseau de capteurs sans fil (RCSF). Ce logiciel offre une API qui facilite le développement d'algorithmes et la visualisation de leurs résultats dans un environnement RCSF réaliste. Il offre la possibilité de simuler les objets mobiles et les cibles. Le code source est disponible dans [45]. Quelques résultats plus illustratifs sont discutés dans la suite.

Notre algorithme peut être utilisé pour trouver les nœuds frontières d'un RCSF. L'objectif de trouver cette frontière pourrait être la réduction de la consommation d'énergie d'un RCSF en travaillant uniquement avec les nœuds de la frontière. Si un nœud de frontière échoue, l'algorithme peut être exécuté à nouveau pour déterminer une nouvelle frontière, ce qui le rend adaptatif et tolérant aux pannes.

Dans ce qui suit, nous allons discuter trois exemples de test pour l'algorithme proposé LPCN2. Le premier exemple représente un réseau aléatoire connecté avec 100 points situés dans une zone rectangulaire de $1200m \times 600m$ (voir Figure 1.24(a)). Dans le deuxième exemple, nous utilisons un réseau de 500 points (voir Figure 1.25) qui sont situés dans une zone rectangulaire de $2500m \times 1500m$. Dans le troisième exemple, nous avons 1500 points (voir Figure 1.26) dans les mêmes conditions que pour la seconde. Le nombre de points obtenus sur l'enveloppe polygonale est de 62 dans le premier réseau, de 202 dans le second et de 85 dans le troisième. Le degré maximal est de 17 dans le premier réseau, de 10 dans le second et de 50 dans le troisième. La complexité théorique dans chaque exemple est respectivement $O(17 \times 62^2) = O(65k)$, $O(10 \times 202^2) = O(408k)$ et $O(50 \times 85^2) = O(361k)$. Le nombre réel d'itérations pour chaque exemple est, respectivement, $9k$, $63k$ et $53k$.

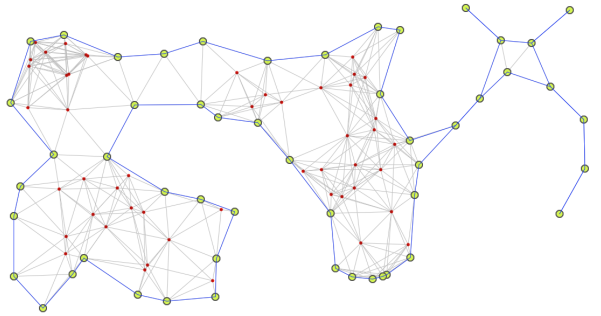


Figure 1.24: Exemple avec 100 points.

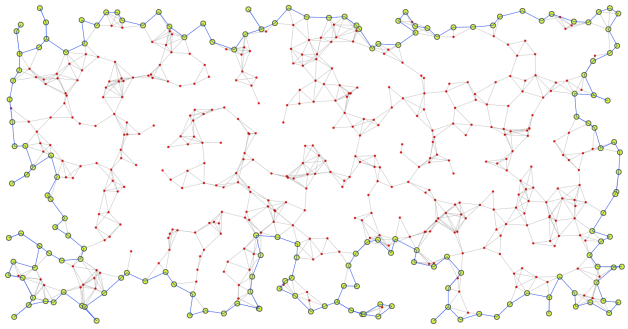


Figure 1.25: Exemple avec 500 points.

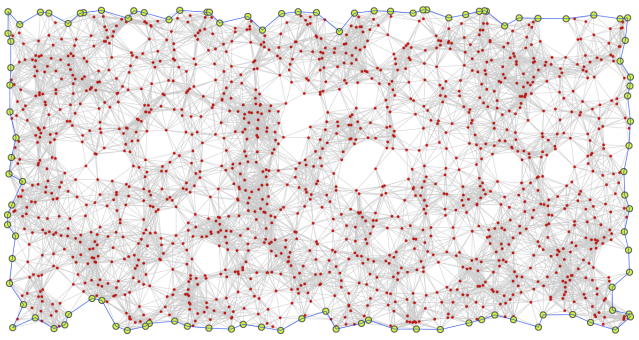


Figure 1.26: Exemple avec 1500 points.

6 Quelques applications

Dans cette section, nous présentons trois exemples pour lesquels l'algorithme présenté peut être utilisé. Le premier exemple montre comment trouver les nœuds frontières d'un réseau de capteurs sans fil. Le second exemple montre comment extraire des clusters complexes dans un ensemble de données bidimensionnelles. Le dernier montre comment dessiner un contour d'une zone d'intérêt sur une image médicale.

6.1 Détermination des nœuds frontières d'un réseau de capteurs sans fil

Trouver les nœuds frontières d'un **RCSF** peut se faire de deux façons différentes: centralisée et distribuée. Nous présentons ci-dessous la version centralisée puisque l'algorithme est appliqué tel que présenté dans ce chapitre. Cependant, dans la version distribuée, la procédure est complètement différente et elle est présentée dans [22]. Dans la version centralisée, comme le montre la Figure 1.27(a), chaque nœud du réseau envoie ses coordonnées à la station de base. La station exécutera ensuite l'algorithme LPCN pour trouver les nœuds qui forment la frontière. Une fois terminé, elle envoie à chaque nœud capteur l'information qu'il s'agisse ou non d'un nœud frontière (voir Figure 1.27(b)).

6.2 Recherche de clusters et reconstruction de formes

Pour trouver les clusters d'un ensemble de données, nous devons d'abord connecter les points entre eux. Pour ce faire, on peut utiliser de nombreuses méthodes existantes, comme les diagrammes de Voronoi [46], les k-voisins les plus proches, les voisins situés dans un certain rayon, etc. Deuxièmement, on part du point ayant la plus petite coordonnée x, et on exécute le LPCN jusqu'à retourner à ce point de départ. Cela signifie que le premier cluster est trouvé, qui est défini par tous les points de l'enveloppe polygonale et tous les points qui sont à l'intérieur de celle-ci. Afin de trouver le deuxième cluster, nous devons retirer de l'ensemble de données tous les points du cluster précédemment trouvé, puis redémarrer la même procédure avec les points restants. Cette procédure est renouvelée jusqu'à ce que tous les clusters soient trouvés. La Figure 1.28 montre un exemple d'ensemble de données bidimensionnel (voir Figure 1.28(a)), la connexion entre les points (Figure 1.28(b)) et enfin les clusters trouvés (voir Figure 1.28(c)). La même méthodologie peut être utilisée pour la reconstruction de la forme. Comme la complexité de l'algorithme LPCN dépend du

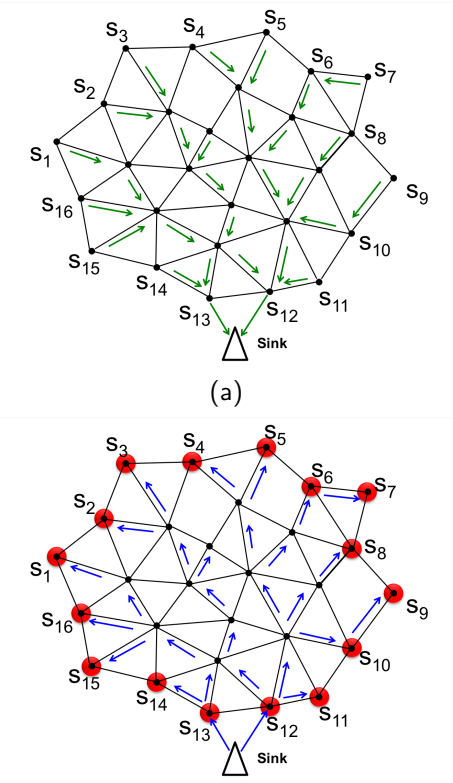
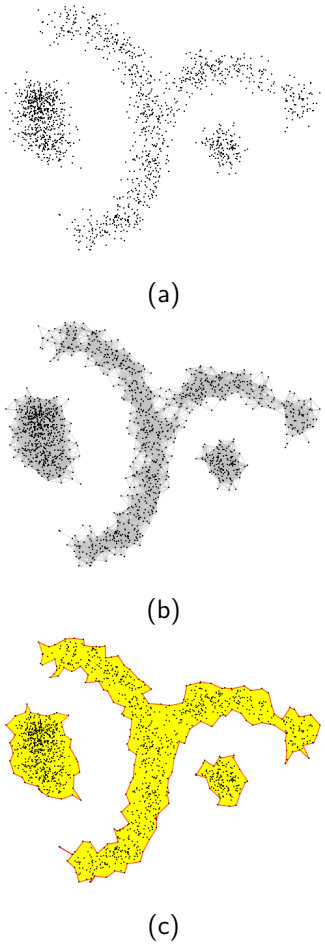


Figure 1.27: Nœuds frontière d'un RCSF.

nombre de points de l'enveloppe polygonale, ce dernier peut être utilisé pour de gros ensembles de données, ce qui le rend très utile dans le contexte de Big Data.



(a)

(b)

(c)

Figure 1.28: Recherche de clusters.

6.3 Dessin de contour

L'algorithme LPCN peut être utilisé pour dessiner des contours dans des images médicales. Dans le cas du domaine d'extraction de la zone d'intérêt, il doit être combiné avec un autre algorithme qui permet de caractériser une telle zone. Par exemple, considérons l'extraction de la zone grise de l'image de la Figure 1.29(a). Tout d'abord, nous transformons la région contenant cette zone en une matrice de pixels. Cette matrice peut être modélisée sous la forme d'un graphe connexe Euclidien comme le montrent les Figures 1.29(b) et (c). Si l'on considère une extraction simple basée sur l'intensité des pixels, on ne considérera que les points bleus du graphe, représenté par la Figure 1.29(d), qui correspondent aux pixels gris de l'image. Pour ce graphe, nous allons exécuter l'algorithme LPCN afin de trouver les points frontières (voir Figure 1.30(e)). Les Figures 1.30(f) et (g) montrent que ces points représentent le contour de la zone grise de l'image originale. Comme

exemple plus réaliste, Figure 1.31 montre le contour d'une tumeur dessinée en utilisant l'algorithme LPCN. Cette image représente une IRM de tumeur (imagerie par résonance magnétique) prise à l'hôpital de Brest.

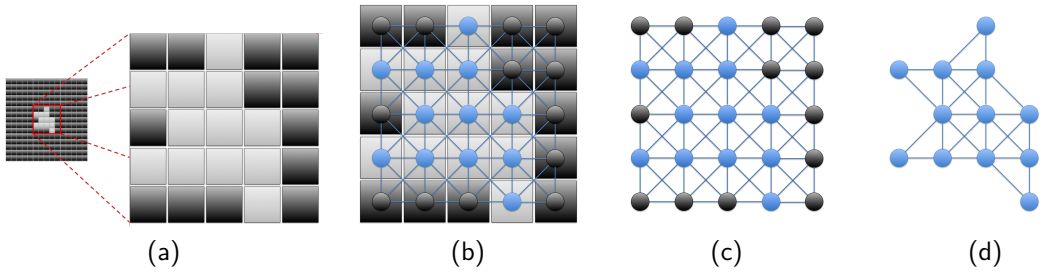


Figure 1.29: Contour d'une zone d'intérêt.

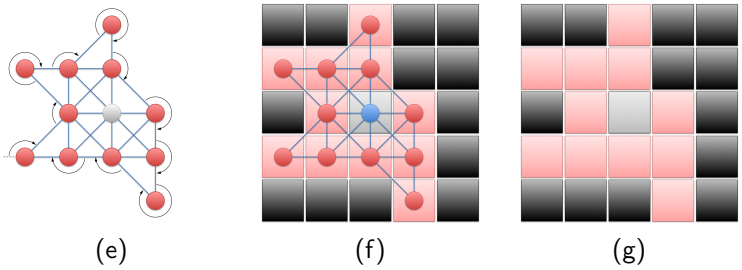


Figure 1.30: Contour d'une zone d'intérêt(suite).

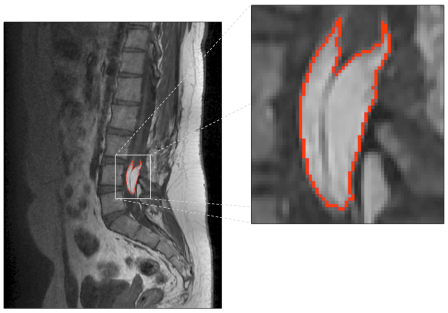


Figure 1.31: Contour d'une tumeur dans une image médicale réelle.

7 Conclusion

Nous avons présenté l'algorithme LPCN permettant de trouver l'enveloppe polygonale d'un graphe Euclidien connecté. Cet algorithme a été inspiré de l'algorithme de Jarvis qui doit être adapté, car

il est conçu pour trouver une enveloppe convexe au lieu d'une enveloppe polygonale. Nous avons proposé un nouvel algorithme qui choisit pour chaque sommet, le sommet d'angle polaire minimum par rapport au sommet trouvé dans l'itération précédente. Sa complexité est $O(kh^2)$, où k est le degré maximal du graphe et h le nombre de sommets sur l'enveloppe polygonale. Nous avons montré que l'exécution de l'algorithme en présence de structures de graphe spécifiques peut conduire à des solutions non valides et non optimales, et nous avons indiqué comment surmonter ces difficultés. De plus, nous avons prouvé la convergence de l'algorithme et l'optimalité de la solution. Enfin, nous avons présenté certaines applications qui peuvent être résolues en utilisant l'algorithme proposé. Nous travaillons maintenant sur la version où l'algorithme peut démarrer à partir de n'importe quel point du graphe au lieu du point ayant une coordonnée x minimale, et nous préparons également une version tridimensionnelle.

2

Détection de Défaillances dans une Frontière de RCSF

1 Introduction

Les réseaux de capteurs sans fil ont émergés au cours de la dernière décennie comme un outil puissant pour relier le monde physique et le monde numérique [47].

Toutefois, les réseaux de capteurs fonctionnent souvent dans des environnements incontrôlables potentiellement hostiles et durs, et en ajoute à cela, les contraintes de ressources limitées des nœuds capteurs en termes d'énergie, de mémoire, de communication, de capacité de calcul et de traitement [48].

Un scénario d'application intéressant pour les RCSFs est le domaine des systèmes de surveillance de zones sensibles. Les applications dédiées à la surveillance des frontières de sites stratégiques et dangereux (p.ex., sites pétroliers ou nucléaires, frontières d'un pays, etc.) sont sensibles aux défaillances. Dans de telles applications, l'évènement détecté doit être envoyé au puits de manière urgente, fiable, et avec une faible consommation d'énergie.

La présence de nœuds défaillants dans ce type d'application peut réduire la qualité de service et peut par la suite causer des dommages économiques et même impliquer des vies humaines. Ces défaillances sont généralement dues aux causes suivantes: épuisement de la batterie, la défaillance des modules (comme module de communication et de détection), les attaques ennemies, les facteurs environnementaux, un circuit cassé et étant hors de la portée de communication de l'ensemble du réseau, etc.

Par conséquent, les nœuds capteurs défaillants doivent être détectés en temps réel, pour les exclure ou les remplacer dans le réseau, afin de garantir une bonne qualité de service et permettre au réseau d'assurer les services pour lesquelles il a été conçu. De surcroît, l'une des principales techniques pour contourner les défaillances est l'exploitation de la redondance, qui est souvent une condition par défaut dans les RCSFs [49].

Dans certaines applications de RCSFs, la reconnaissance des nœuds frontière est nécessaire pour découvrir la topologie du réseau [50], pour effectuer le routage géographique [50, 51], le suivi et le guidage [50], pour surveiller les frontières des sites sensibles et stratégiques (p.ex., champs pétrolifères ou sites nucléaires, frontières d'un pays, etc.).

De ce fait, ce chapitre sera consacré à la présentation d'une approche permettant la surveillance d'un site sensible à l'aide d'une frontière de réseau de capteurs sans fil. La solution proposée est tolérante aux pannes du fait qu'elle permet la détection des nœuds défaillants au niveau de la frontière tout en assurant la fonction de surveillance. En fait, après le déploiement, les noeuds frontières seront déterminés à l'aide de l'algorithme D-LPCN [52], qui est la version distribuée de notre algorithme proposé LPCN [21] appliqué sur un RCSF, où, dans chaque itération, le nœud frontière suivant n_{i+1} est déterminé par le nœud ayant l'angle minimum $\varphi_{min}(n_{i-1}, n_i, n_{i+1})$ formé par les arêtes $\{n_i, n_{i-1}\}$ et $\{n_i, n_{i+1}\}$, où n_i est le nœud frontière trouvé dans l'itération i, et n_{i-1} est le nœud de frontière trouvé dans l'itération précédente $i-1$. La raison pour laquelle cet algorithme a été choisi pour ce travail, est que ce dernier repose sur la caractéristique, que chaque nœud frontière connaît ses deux voisins frontières.

En effet, une fois que les nœuds frontière sont déterminés, chacun de ces nœuds envoie périodiquement un message à son prochain voisin, qui devrait répondre. Si une réponse n'est pas reçue, une situation d'échec sera déclenchée et une restructuration du réseau sera effectuée pour trouver une nouvelle frontière. L'approche proposée nécessite un nombre limité de calculs locaux simples, et elle a besoin juste des informations sur les voisins à un seul saut. Les nœuds de frontière agissent comme une sentinelle avec un cycle de service élevé (high duty cycle) alors que les nœuds qui sont les voisins des nœuds frontière peuvent avoir un cycle de service faible (low duty cycle). Les autres nœuds sont utilisés uniquement pour remplacer les nœuds frontière défaillants.

Ce chapitre est organisé comme suit. Il est divisé en trois parties principales. La première partie présente une étude bibliographique sur les méthodes de détection de nœuds frontière dans un RCSF (Section 2), ainsi que sur les méthodes permettant la détection des nœuds défaillants dans un RCSF (Section 3).

Dans la deuxième partie, la Section 4 présente l'algorithme D-LPCN et les résultats de simulation avec le simulateur CupCarbon. Ensuite, la Section 5 introduit la méthode proposée pour détecter les nœuds défaillants dans une frontière de RCSF.

La troisième partie (Section 6) est consacrée à la présentation des résultats et l'analyse des performances de l'approche proposée.

2 Les méthodes permettant la détermination de noeuds frontière dans un RCSF

Afin de trouver les nœuds frontière d'un réseau de capteurs sans fil, plusieurs méthodes ont été développées. Ils peuvent être classés en trois catégories: géométrique, statistique et topologique. Les méthodes géométriques [53, 50, 54] sont les plus précises. Ils utilisent les informations de localisation de chaque nœud pour déterminer les nœuds de la frontière [54, 50]. Les méthodes statistiques font des hypothèses sur la distribution de probabilité du déploiement des nœuds et identifient les nœuds frontière en se basant sur des propriétés statistiques sous certaines conditions du réseau. [54, 50]. Les méthodes topologiques utilisent les informations de connectivité pour déterminer les nœuds de la frontière [54]. Ces méthodes permettant l'échange d'informations de connectivité entre les nœuds voisins et détectent les nœuds frontière sans aucune information sur leur emplacement. En général, ils surpassent les méthodes statistiques [53, 54, 50].

Dans [55], un algorithme distribué basé sur une méthode topologique a été proposé. Il détermine

d'abord la frontière d'un ensemble de nœuds qui sera considérée comme un ensemble de seeds. Ensuite, il détermine les distances maximales de saut autour de chaque seed et examine si le contour autour d'un seed forme un cycle fermé. Cependant, seuls les nœuds qui sont proches des frontières sont identifiés et ils ne sont pas explicitement connectés comme un polygone. En outre, cette approche nécessite des densités de réseau très élevées.

Un autre algorithme distribué basé sur des méthodes topologiques est présenté dans [51]. Pour déterminer les nœuds de la frontière, cet algorithme suppose que les nœuds peuvent communiquer avec leurs nœuds voisins à un seul ou deux sauts. Il teste si les nœuds dont les voisins à deux sauts peuvent être utilisés pour former un cycle avec le nœud concerné. Sinon, ce dernier est classé comme un nœud de frontière. Cet algorithme distribué ne distingue pas les nœuds près de la frontière de ceux qui sont proches des trous, et il ne fonctionne qu'avec des réseaux denses.

Dans [50], une heuristique basée sur la recherche de frontières des cycles à l'intérieur d'un réseau est développée. L'idée principale de cette approche est de partir d'un premier cycle qui sera progressivement agrandi en ajoutant d'autres sommets jusqu'à ce qu'il atteigne la frontière du réseau. Un sommet de plus haut degré sera choisi comme centre du cycle. Dans le cas où d'autres cycles seraient trouvés, ils seront fusionnés. Cette approche peut fonctionner avec des réseaux de capteurs sans fil à faible densité.

Les auteurs de [56] ont présenté une technique pour obtenir un nœud frontière à l'aide d'un algorithme de traitement d'images après transformation du graphe du réseau en image. Malheureusement, cette technique est centralisée et ne définit que l'enveloppe convexe.

Dans [57], un algorithme est proposé qui démarre à partir de la station de base et qui sélectionne le nœud avec la coordonnée maximale ou minimale selon x ou y. Cette procédure est répétée pour chaque nœud sélectionné jusqu'à ce que la station de base soit atteinte. Malheureusement, cet algorithme ne gère pas certains cas qui peuvent provoquer des boucles infinies.

Dans [58], un algorithme appelé *ABBD* (Angle Based Boundary Detection Algorithm) est développé pour détecter les frontières en utilisant la distance entre les nœuds voisins. L'idée principale de cet algorithme est de tester si chaque nœud capteur est recouvert par le polygone formé par ses nœuds voisins. Un nœud capteur sera déclaré comme nœud non-frontière, s'il est recouvert par au moins trois nœuds. Cette condition n'est pas toujours vérifiée car il est possible de trouver des nœuds qui sont couverts par moins de trois nœuds et qui ne sont pas des nœuds de frontière (p.ex, un nœud capteur interne avec un seul voisin). Cet algorithme nécessite beaucoup de communications.

Un autre algorithme proposé par [59] est basé sur les capteurs **Ircod**. Ces capteurs déterminent un ordre cyclique pour leurs voisins et choisissent le premier voisin dans cet ordre. Ensuite, la règle de la main droite sans croisement (RHWoC) est utilisée pour déterminer la frontière complète. L'inconvénient de cet algorithme est qu'il suppose qu'il n'y a pas plus d'un seul voisin du nœud capteur **Ircod** avec le même angle ou sur la même ligne. Cependant, il ne détecte que l'enveloppe convexe d'un ensemble donné de nœuds . Cet algorithme souffre de la condition d'arrêt, c'est-à-dire, il se termine dès que la station de base est revisitée, et il nécessite de déterminer le centre du réseau qui est une tâche difficile.

Dans [60], deux algorithmes appelés sélection séquentielle de nœud de frontière (SBNS) et sélec-

tion distribuée des nœuds de frontière(DBNS) sont développés pour déterminer les nœuds frontière d'un RCSF avec ou sans la présence d'obstacles au niveau de la région de déploiement. La résolution de l'algorithme commence à partir de la station de base au centre du réseau, et sélectionne le réseau et ses voisins. Après des processus itératifs, seuls les derniers nœuds sont sélectionnés comme nœuds de frontière formant le périmètre du réseau.

Dans [61], un algorithme appelé Distributed Boundary Detection (DBD) est développé pour identifier les frontières des obstacles et des réseaux. Chaque nœud ne requiert que l'information de ses voisins à trois sauts. Un nœud qui exécute cet algorithme peut déterminer s'il est lui-même un nœud frontière de manière distribuée et identifie la frontière extérieure d'un réseau. Par conséquent, cette méthode est basée sur une approche heuristique qui estime approximativement la frontière du réseau.

Dans [62], les auteurs introduisent un modèle de communication basé sur un graphique d'unité de disque virtuel (QUDGs). Ils ont présenté un nouveau cadre pour l'auto-organisation d'un grand réseau de capteurs. Le scénario de base peut être décrit comme suit: Étant donné un grand nombre de nœuds capteurs stationnaires qui ont été dispersés dans une région polygonale, les auteurs donnent une approche déterministe distribuée pour identifier les nœuds qui se trouvent dans la zone polygonale ou près de sa limite. Leur algorithme est basé sur des considérations topologiques , ainsi que des arguments géométriques. En utilisant la structure de la frontière, ils décrivent une méthode distribuée déterministe pour extraire la topologie de l'ensemble de nœuds de départ.

Dans [63], les auteurs proposent une technique décentralisée de détection des frontières sans information de localisation. Dans cette technique, chaque nœud capteur connaît les informations sur les voisins à trois sauts, deux sauts et un seul saut à l'aide des messages HELLO. En fait, cette technique assure la précision de la détection des frontières et des vides par rapport à une technique heuristique. Cependant, cette méthode ne prend pas en charge les architectures densément déployées en raison de la limitation de ressources et de son principe de demander des informations sur les voisins à un, deux ou trois sauts.

Les auteurs de [64] proposent un nouvel algorithme appelé Sensor Localization and Obstacle Discovery (SLOD) pour localiser les obstacles et les nœuds capteurs en se basant sur quatre nœuds spéciaux activés par GPS appelé nœuds d'ancre. Les auteurs estiment la position des nœuds capteurs avec une frontière rectangulaire approximative de chaque obstacle détecté. Ensuite, ils déterminent les formes précises des obstacles, basées sur la communication de données entre les nœuds capteurs voisins, déployés autour des obstacles et en utilisant une méthode heuristique.

Dans [65], les auteurs proposent deux algorithmes distribués basés sur le calcul géométrique, DSCS (Distributed Sector Cover Scanning) et Directional Walk (DW). Le premier algorithme est utilisé pour identifier les nœuds sur les frontières des trous et la frontière externe. Le deuxième est utilisé pour localiser les trous de couverture basés sur les nœuds frontière identifiés avec DSCS et ce qu'ils entourent. Ces algorithmes sont basés sur l'observation qu'un nœud peut avoir sur sa zone de détection. Cette dernière est divisée en différents secteurs par ses voisins. Chaque nœud doit avoir les informations de ses voisins à un et deux sauts. Le DSCS est exécuté après que l'information requise ait été recueillie. Ensuite, chaque nœud calcule l'angle absolu de chaque voisin et les ordonne en ordre croissant. Sur la base des deux informations (voisins à un et à deux sauts et leurs angles absolus), le nœud peut décider d'être un nœud de frontière ou non. Cependant, ces algorithmes nécessitent beaucoup d'informations (voisins à un et à deux sauts) qui nécessitent une communication extensive et une forte consommation d'énergie. De plus, les algorithmes ne sont pas précis, c-à-d.,

que 10% des nœuds frontières calculés sont faux. Les deux algorithmes sont évalués et implémentés dans MATLAB qui n'est pas une plate-forme réaliste pour les RCSFs.

Le travail de [66] présente une méthode d'analyse théorique pour la détection de nœuds frontière basée sur des polygones localisés de Voronoi. Cette méthode provient du calcul géométrique, dans le sens où chaque nœud capteur dans le réseau teste s'il est sur la frontière en calculant le polygone de Voronoi lié à lui-même et en se basant sur la portée du captage et celle de la communication.

L'étude de [67] est focalisée sur la détection d'une frontière externe d'un RCSF en utilisant un mécanisme d'identification des nœuds de déclenchement. L'idée de l'algorithme est d'initier le processus en identifiant un ou plusieurs nœuds de frontières. Ces nœuds du réseau sont situés le plus loin possible du sink (puits). Chaque nœud calcule sa distance à la fois du puits et de ses voisins. Le nœud qui est le plus éloigné du puits représente un nœud frontière qui sera utilisé comme déclencheur pour découvrir la frontière externe. Ensuite, ce nœud utilise une méthode géométrique pour détecter les nœuds frontière suivants à gauche et à droite de celui-ci. Cependant, cet algorithme n'est pas adapté aux différentes types de topologies, et il suppose que le puits est dans l'extérieur du réseau, ce qui n'est pas toujours le cas.

Les auteurs de [68] proposent une approche topologique pour la détection des frontières des trous et la frontière externe d'un réseau de capteurs. Cette approche est basée sur l'identification et la vérification de la connectivité avec les voisins à x sauts entourant chaque nœud, et elle comprend trois étapes: la collecte d'informations, la construction de chemin, et la vérification de chemin. Dans la première étape, chaque nœud identifie la liste de ses voisins à x sauts. Dans la deuxième étape, la connectivité avec les voisins à x sauts est utilisée pour construire un chemin de communication autour de chaque nœud. Ces chemins sont vérifiés dans une troisième étape pour détecter la frontière et les nœuds internes, c-à-d., lorsque le chemin est fermé, le nœud est interne, sinon le chemin est brisé, et d'autres tests sont nécessaires. Cet algorithme souffre du choix du facteur x, étant donné que le facteur x a une valeur élevée, ce qui signifie une surcharge de communications très élevées et donc une consommation d'énergie excessive. De plus, si le facteur x reçoit une valeur faible, la précision de l'algorithme diminue.

Le travail de [69] est focalisé sur la détection du périmètre à l'aide d'un algorithme décentralisé, dans des réseaux de capteurs sans fil basé sur l'information de localisation de voisinage combinée avec la technique barycentrique (utilisée pour déterminer si un point est à l'intérieur ou à l'extérieur d'un triangle ABC). L'algorithme utilise pour chaque nœud l'information de localisation de trois nœuds voisins pour former un triangle, et détermine s'il est entouré par eux. Si c'est le cas, il n'est pas un nœud de périmètre. Sinon, il s'agit d'un nœud de périmètre. Cet algorithme n'est pas précis et nécessite une densité de réseau élevée, ce qui signifie qu'il présente un taux de fausses alarmes très élevé pour détecter un nœud de périmètre dans un réseau à faible densité, et également dans un réseau avec des nœuds ayant un faible rayon de communication.

Dans [70], les auteurs proposent un algorithme distribué pour détecter un sous-ensemble de nœuds de frontière. Ainsi, ils ont développés un algorithme de routage glouton, qui est utilisé pour collecter des informations sur les nœuds frontière au niveau du sink. Ils supposent un nœud de départ qui soit affecté, si ses données collectées dépassent une valeur de seuil Th. Alors, il exécute l'algorithme distribué proposé pour détecter les nœuds frontière. L'algorithme est divisé en trois phases. Dans la première phase, chaque nœud affecté se déclare comme un nœud non-frontière si ses voisins sont tous affectés. Sinon, si certains de ses voisins ne sont pas affectés, il calcule

la fonction f en fonction du nombre de ses voisins et de la valeur Th. Dans la deuxième phase, il diffuse la valeur de f afin de donner un vote pour d'autres nœuds. À la phase finale, chaque nœud identifié comme un nœud frontière, vérifie s'il a la plus petite valeur de f, cas dans lequel il reste un nœud frontière, sinon il devient un nœud interne. Cependant, cet algorithme nécessite une densité de réseau élevée, il n'est pas très précis, et il consomme beaucoup d'énergie en raison de la communication avec les nœuds voisins à plusieurs sauts.

Nous avons observé que certains de ces algorithmes fonctionnent avec des réseaux planaires, et qu'ils ne prennent pas en compte les intersections entre les arêtes. En effet, celle-ci peuvent conduire à des situations de blocage découlant de sous-graphes spécifiques, ou à des cycles produisant une boucle infinie. Nous avons illustré ces situations dans [29], et montré, comment les éviter. Nous avons également observé qu'en général, ils ne donnent pas la frontière optimale en termes de nombre de nœuds. La table 2.1 résume 8 algorithmes de détermination de frontière en termes de complexité du message (nombre de messages échangés par nœud), et de précision qui représente le pourcentage des nœuds frontière réels, où n est le nombre de nœuds, n_b est le nombre de nœuds de frontière, k est le nombre de nœuds qui exécutent simultanément le calcul local, d est le degré maximum du réseau, h la taille de la table des voisins à 3-sauts et h_{max} est le nombre maximum de sauts d'un nœud définie par l'utilisateur. Nous avons ajouté quelques observations sur la surcharge de communication dans la dernière colonne. Nous avons comparé notre algorithme avec seulement ceux pour lesquels la consommation d'énergie est fournie dans leur papier. Une autre comparaison fondée sur la consommation d'énergie est présentée dans la section 4.2.3.

Tableau 2.1: Comparaison avec les algorithmes existants.

Algorithme	Complexité du message	Précision	Observations
ABBD [58]	$O(n/k + n_b/k)$ = nombre de tours $O(kd^3)$ = nombre de messages par tour	$\simeq 88\%$	Surcharge de communication
EBDG [70]	$O(d\ h_{max})$	$\simeq 92\%$	Nécessite un réseau dense
LVP [66]	$O(n\ d)$	$\simeq 100\%$	Surcharge de calcul
PDiscovry [69]	$O(n\ d\ h)$	$\simeq 95\%$	Surcharge de communication
DBD [61]	$O(n(d^3 + d^2 + d))$	$\simeq 90\%$	Nécessite l'information des voisins à 3-sauts
DBNS [60]	$O(n/k)$ = nombre de tours $O(k\ d)$ = nombre de messages par tour	$\simeq 100\%$	Nécessite la suppression des nœuds redondants pour former une frontière complète
Hop-based [68]	$O(n\ d\ h_{max})$	3% to 99%	Dépend de la liste de x-saut voisins. Surcharge de communication
D-LPCN	$O(d\ n_b + n)$	$= 100\%$	La communication et l'énergie sont réduites ; seulement les noeuds frontières et leurs voisins sont utilisés.

3 Les méthodes de détection de défaillances dans les RCSFs

Depuis plusieurs années, l'efficacité énergétique est le critère de conception majeure pour de nombreuses solutions RCSFs proposées. Cependant, il reste encore des efforts importants à accomplir notamment en ce qui concerne la détection de nœuds défaillants.

Dans [71], les auteurs ont proposé MANNA, un système de détection de défaillance dans un réseau de capteurs sans fil. Chaque nœud vérifie son niveau d'énergie et envoie un message aux gestionnaires ou agents externes. En cas de changement d'état, le gestionnaire situé du côté externe du RCSF effectue une détection centralisée des défaillances en se basant sur l'analyse des données collectées du RCSF. Cependant, cette approche consomme beaucoup d'énergie en raison de la communication entre les nœuds et les gestionnaires.

Dans [72] et [73], les auteurs ont proposé un algorithme pour estimer le temps de défaillance d'une batterie en analysant la courbe de décharge de la batterie et le débit de décharge courant.

Dans le projet FleGSens [74], un réseau de capteurs sans fil pour la surveillance des zones et des propriétés critiques est développé, qui intègre des mécanismes garantissant la sécurité de l'information. Le prototype prévu est constitué de 200 nœuds capteurs pour surveiller une longue bande terrestre de $500m$. Le système vise à garantir l'intégrité et l'authenticité des alarmes générées, ainsi que la disponibilité en présence d'un attaquant qui peut utiliser un nombre optimal de nœuds capteurs. Deux protocoles ont été développés et présentés: un protocole de suivi pour assurer la détection sécurisée d'entrées non autorisées dans la zone surveillée, et un autre protocole, pour une détection sécurisée des nœuds défaillants qui assure l'intégrité du réseau de capteurs et évite toute violation de la couverture. Cela se fait en sélectionnant un nombre aléatoire de nœuds appelés copains (buddies) pour écouter d'autres nœuds appelés heartbeats.

Dans [75], une méthode pour détecter la défaillance du nœud capteur, ou le dysfonctionnement à l'aide d'un facteur de confiance est présentée. Le facteur de confiance d'une trajectoire aller-retour dans le réseau est estimé, en utilisant le délai aller-retour (RTD) round trip delay, qui est le temps requis pour qu'un signal se déplace depuis un nœud source spécifique le long d'un chemin contenant d'autres nœuds, et le retour vers le même nœud source. Les facteurs de confiance de tous les trajets aller-retour sont stockés dans des tables de consultation. Ensuite, en analysant l'état du facteur de confiance de tous les chemins dans ces tables de consultation, les nœuds capteurs défaillants sont détectés facilement.

La technique présentée dans [76] est basée sur le calcul périodique du débit d'une zone rectangulaire contenant un certain nombre de nœuds capteurs. Le débit calculé sera comparé à un seuil prédéfini. S'il est inférieur à ce seuil, la zone sera ensuite divisée en quadrants. Le débit de chaque quadrant sera calculé et comparé à un autre seuil. Si le débit d'un quadrant est inférieur au seuil correspondant, il sera de nouveau divisé en quatre autres quadrants. Ce processus sera répété jusqu'à ce qu'un quadrant contenant un seul nœud capteur soit atteint. Le nœud entouré par ce quadrant peut être identifié comme un nœud suspect sur la base de son débit qui est faible. Après cela, les nœuds endormis autour et près du nœud suspect seront éveillés afin de le tester.

Dans [77], un outil appelé *Sympathy* est développé pour détecter et déboguer les échecs dans les réseaux de capteurs. Sympathy calcule les métriques qui permettent de détecter efficacement les défaillances, comme les paquets statistiques générés par les nœuds et les transmettent au puits.

Sympathy détecte une défaillance et déclenche la localisation lorsqu'un nœud génère moins de trafic surveillé que prévu.

4 Algorithme de détermination de la frontière (D-LPCN)

Dans cette section, nous présentons les étapes principales nécessaires à la découverte des nœuds frontière dans un RCSF à l'aide de l'algorithme (D-LPCN). Tout d'abord, nous introduisons quelques définitions et primitives. Ensuite, nous discutons les résultats de simulation obtenus avec cet algorithme.

4.1 Algorithme D-LPCN

4.1.1 Définitions et Primitives

Nous supposons que la communication entre deux nœuds capteurs est symétrique. Dans ce cas, un RCSF peut être modélisé comme un graphe non orienté $G = (S, E)$, où $S = \{s_1, s_2, ..., s_n\}$ est l'ensemble des nœuds capteurs, $n =\mid S \mid$ leur nombre total et E l'ensemble des liens de communication. Le lien entre deux nœuds peut être défini comme suit:

$$e_{ij} = \begin{cases} 1 & si\ le\ noeud\ s_i\ communique\ avec\ s_j \\ 0 & autrement \end{cases} \tag{2.1}$$

Les nœuds voisins $N(s_i)$ d'un nœud donné s_i sont les nœuds qui communiquent avec lui.

$$N(s_i) = \{s_j / e_{ij} = 1,\ \ j = 1, ..., n\ \ and\ \ j \neq i\} \tag{2.2}$$

Pour une meilleure compréhension de l'algorithme proposé, nous définissons dans le Tableau 2.2 quelques primitives de message et leurs définitions, et dans le Tableau 2.3 les fonctions utilisées dans l'algorithme. Notez que, l'algorithme proposé fonctionne dans le cas de la communication bidirectionnelle, afin d'obtenir les nœuds frontière de manière optimale, et chaque nœud connaît sa position dans l'espace en termes de coordonnées (x, y).

Tableau 2.2: Primitives de message et leurs définitions

Primitive	Définition
AC	Demander des coordonnées
CS	Envoyer des coordonnées
SN	Sélectionner un nœud

Tableau 2.3: Fonctions de l'algorithme D-LPCN

Fonction	Définition
`getId()`	Retourne l'identificateur du nœud
`getCoord()`	Renvoie les coordonnées du nœud, (x, y)
`getNumberOfNeighbors()`	Renvoie le nombre de voisins du nœud
`send(`a`,` b`)`	Envoie le message a au nœud capteur ayant l'identifiant b, ou en diffusion générale ($b = *$)
`read()`	En attente de réception de messages

4.1.2 L'algorithme D-LPCN

L'algorithme D-LPCN utilise le même principe que **LPCN** à l'exception du fait que le programme est écrit sous une forme distribuée. Cela signifie que chaque nœud capteur va exécuter son propre programme, et en communiquant avec ses voisins, il peut décider s'il s'agit d'un nœud frontière ou non. L'algorithme 4 montre les principales étapes de l'algorithme D-LPCN qui peuvent être décrites comme suit:

- *Étape 1 (lignes 1 à 4)*: Initialisation.
- *Étape 2 (ligne 5)*: L'algorithme D-LPCN part du nœud (first_node) ayant la coordonnée x minimale.
- *Étape 3 (lignes 6 à 10)*: Si le nœud courant est un nœud de départ, il enverra un message "AC" à ses voisins afin de demander leurs coordonnées.
- *Étape 4 (lignes 12 à 13)*: Chaque nœud attend pour recevoir un message.
- *Étape 5 (lignes 14 à 17)*: La variable i détermine le nombre de messages "CS" reçus. La réception de n "CS" messages signifie que les coordonnées de tous les voisins ont été reçues.
- *Étape 6 (lignes 18 à 20)*: Si le nœud reçoit un message "AC", il enverra un message "CS" contenant ses coordonnées à l'émetteur ayant l'identifiant id.
- *Étape 7 (lignes 21 à 27)*: Pour tous les messages "CS" reçus, le nœud calcule l'angle minimum formé avec eux en prenant en compte les intersections avec l'ensemble de la frontière reçu (dans la ligne 15). Ce calcul s'effectue à l'aide de la fonction angleWI (angle sans intersections).
- *Étape 8 (lignes 28 à 23)*: La réception d'un message "SN" par un nœud signifie que ce nœud a été sélectionné en tant que nœud frontière par son voisin frontière précédent. Ce nœud redémarrera alors le processus de recherche du nœud frontière suivant en diffusant un message "CS".

Enfin, l'algorithme 4 s'arrête lorsque le premier nœud frontière est sélectionné une seconde fois avec un message "SN". Cet algorithme, tel qu'il est présenté, continuera à déterminer continuellement les nœuds de la frontière. Si nous voulons ajouter la condition d'arrêt, il suffit d'ajouter le code suivant après la ligne 28:

if (first_node) **then**
STOP;
end if

Pour mieux expliquer comment cet algorithme fonctionne, nous allons utiliser l'exemple de la Figure 2.1 qui représente un RCSF avec huit nœuds capteurs. Considérons l'ensemble $S = \{S1, S2, ..., S8\}$ de ces huit nœuds et l'ensemble B des nœuds de la frontière, initialement vide.

Tout d'abord, après l'initialisation (*Etape 1*), le seul nœud qui sera considéré comme un nœud de départ est $S1$ (*Etape 2*), et donc l'ensemble des nœuds frontière est mis à jour pour $boundary\_set = \{S1\}$.

Ensuite, le nœud $S1$ diffuse un message "AC" à ses voisins $N(S1) = \{S2, S3, S4, S7\}$ (*Étape 3*) pour demander leurs coordonnées (voir Figure 2.1(a)) tandis que les autres nœuds attendent la réception des messages (*Étape 4*).

Ensuite, chaque nœud voisin $S2, S3, S4$ et $S7$, qui reçoit le message "AC", envoie un message "CS" (*Etapes 5 et 6*) au nœud frontière $S1$ (voir Figure 2.1(b)), ce qui permet de calculer l'angle formé par le nœud fictif $S1$ avec lui-même et avec chacun de ses voisins $S2, S3, S4$ et $S7$ (*Étape 7*). Cette situation est illustrée par la figure Figure 2.1(c).

Alors, comme le montre la Figure 2.1(d), le nœud frontière $S1$ enverra un message "SN" au nouveau nœud frontière $S3$ qui mettra à jour sa liste des nœuds frontière à $boundary\_set = \{S1\}$ (*étape 8*). La situation obtenue est représentée par la Figure 2.1(e).

Le nœud frontière suivant $S3$ exécutera alors la même procédure à partir de l' *Étape 2*. La Figure 2.1(f) montre l'itération finale de l'algorithme D-LPCN.

En tout, le pseudo-code de D-LPCN est donné par l'Algorithme 4.

4.2 Implementation et Résultats de simulation

4.2.1 Simulation

Pour valider l'algorithme proposé D-LPCN, nous avons utilisé le simulateur CupCarbon [**?**] qui est un simulateur de ville intelligente et d'Internet des objets. Son objectif est de concevoir, visualiser, déboguer et valider des algorithmes distribués de surveillance, de collecte de données environnementales, et de créer des scénarios environnementaux, généralement dans le cadre de projets éducatifs et scientifiques. CupCarbon propose deux environnements de simulation. Le premier environnement de simulation est un environnement multi-agents, qui permet la conception de scénarios de mobilité, et la génération d'événements tels que les incendies et le gaz ainsi que la simulation de mobiles tels que des véhicules et des objets volants. [78].

Les nœuds rouges de la Figure 2.2, mis en surbrillance par les flèches, montrent les nœuds avec la coordonnée x minimale dans chaque sous-réseau d'un réseau contenant plusieurs composantes connexes. Les nœuds frontière, en jaune, sont trouvés par l'algorithme 4 (Algorithme D-LPCN) présenté dans la section 4.1.2 qui part des nœuds rouges déterminés à l'étape précédente.

Algorithm 4 L'algorithme D-LPCN

```
1: boundary = false; phi_min = 10;
2: c_id = getId(); c_coord = getCoord();
3: boundary_set = ∅;
4: n = getNumberOfNeighbors(); i=0;
5: first_node = Point ayant la coordonnée x minimale;
6: if (first_node) then
7:     boundary = true;
8:     p_coord = (c_coord.x−1, c_coord.y);
9:     send(c_id+"|"+"AC", *);
10: end if
11: repeat
12:     id = read();
13:     type = read();
14:     if (i==n) then
15:         boundary_set = boundary_set ∪ {c_id};
16:         send(c_id+"|"+"SN"+"|"+c_coord+"|"+boundary_set, n_id);
17:     end if
18:     if (type=="AC") then
19:         send(c_id+"|"+"CS"+"|"+c_coord, id);
20:     end if
21:     if (type=="CS") then
22:         n_coord = read(); i=i+1;
23:         phi = angleWI(p_coord, c_coord, n_coord, boundary_set);
24:         if (phi<phi_min) then
25:             phi_min = phi; n_id = id;
26:         end if
27:     end if
28:     if (type=="SN") then
29:         boundary = true; phi_min = 10; i=0;
30:         p_coord = read();
31:         boundary_set = read();
32:         send(c_id+"|"+"AC", *);
33:     end if
34: until false
```

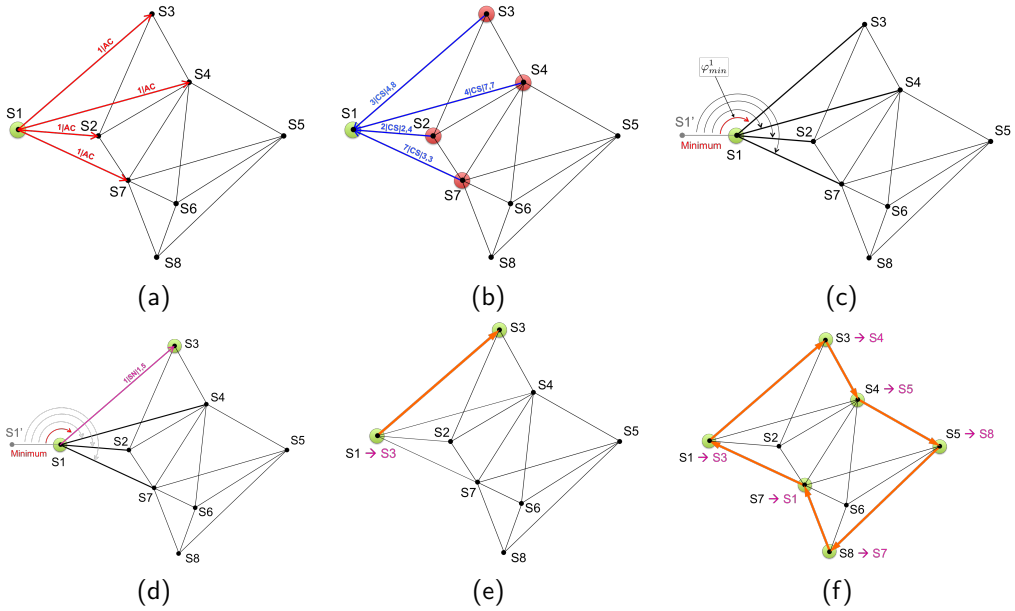


Figure 2.1: Un exemple pour l'algorithme D-LPCN.

4.2.2 La Consommation d'énergie

Pour estimer la consommation d'énergie de l'algorithme proposé, nous avons utilisé le modèle énergétique des nœuds capteurs TelosB décrit dans [79]. Sa consommation d'énergie est estimée à $59.2\mu J$ pour la transmission d'un octet et à $28.6\mu J$ pour la réception d'un octet. Pour profiler l'énergie consommée par les nœuds capteurs, nous avons utilisé l'outil *Powertrace* [80] qui estime la consommation d'énergie avec une précision très proche de la consommation réelle. Les expériences dans [81] montrent que la consommation d'énergie obtenue par l'outil *Powertrace* est avec 94% très proche de la consommation d'énergie d'un dispositif réel. La consommation d'énergie est calculée en mW.

La Figure 2.3 donne une illustration où l'on considère un réseau avec 35 nœuds incluant 17 nœuds de frontière (souligné en jaune) et 11 nœuds connectés aux nœuds frontière (en rouge). Les nœuds restants ne sont pas connectés aux nœuds frontière.

La Figure 2.4 montre la consommation d'énergie de ces nœuds. Les barres vertes représentent la consommation d'énergie des nœuds jaunes, les barres rouges représentent la consommation d'énergie des nœuds rouges et les barres noires, égales à zéro, représentent la consommation d'énergie des autres nœuds. Comme nous pouvons le voir, le nœud S_{15} est le nœud qui consomme le maximum d'énergie parmi les voisins des nœuds frontières parce qu'il a le nombre maximum de nœuds voisins qui appartiennent à la frontière. Nous pouvons également observer que même si ce nœud n'est pas un nœud frontière, il consomme plus d'énergie que le nœud frontière S_{33}, par exemple. De la même manière, le nœud S_{26} est le nœud qui consomme l'énergie maximale parmi les nœuds frontière parce qu'il a le nombre maximum de voisins. En outre, si nous comparons la consommation d'énergie de ces deux nœuds, nous constatons que S_{26} consomme plus d'énergie que S_{15} bien que S_{15} ait plus de voisins. Ceci est justifié par le fait que le nœud S_{15} est seulement sollicité 5 fois par ses 5 voisins de frontière pour envoyer ses coordonnées. Cependant, alors que le nœud S_{26} est invité à envoyer ses coordonnées par seulement 3 voisins de frontière, il doit envoyer un message pour demander aux

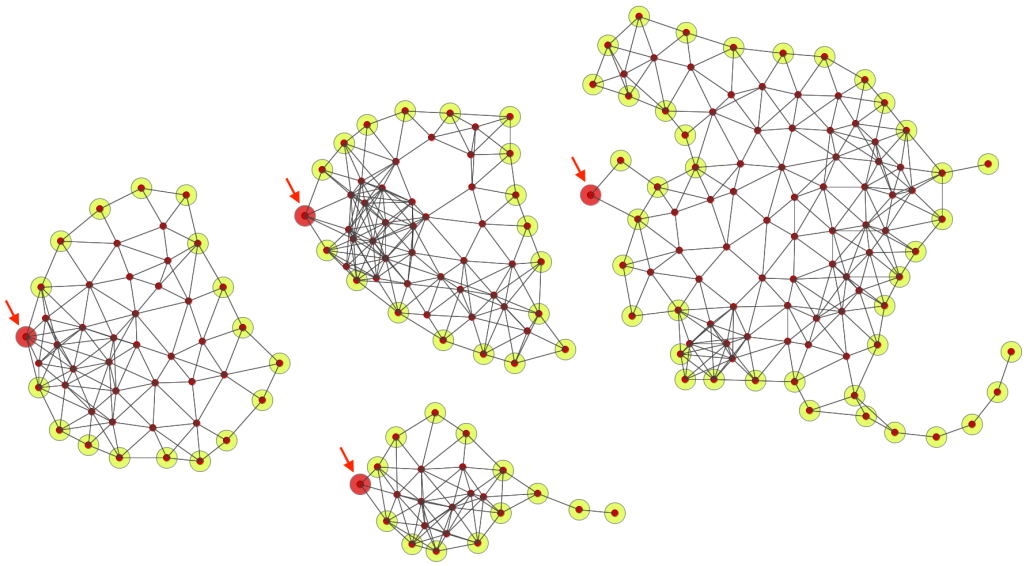


Figure 2.2: Simulation CupCarbon de l'algorithme D-LPCN pour un réseau non-connexe.

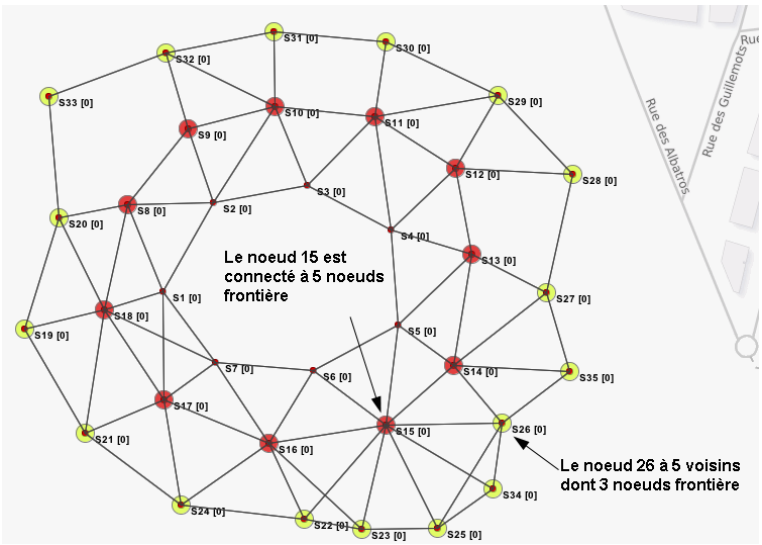


Figure 2.3: Un Réseau de Capteurs sans Fil avec 35 noeuds.

autres nœuds leurs coordonnées et envoyer un autre message pour choisir le nœud frontière suivant. De plus, ce nœud a reçu un message du nœud frontière précédent indiquant qu'il a été choisi comme nœud frontière. Enfin, les nœuds internes qui ne sont pas connectés aux nœuds frontière consommeront une énergie négligeable puisqu'ils ne communiquent pas avec eux. Ces nœuds peuvent être utiles pour le remplacement de nœuds éventuellement défaillants, ce qui garantit un véritable réseau de capteurs sans fil tolérant aux pannes.

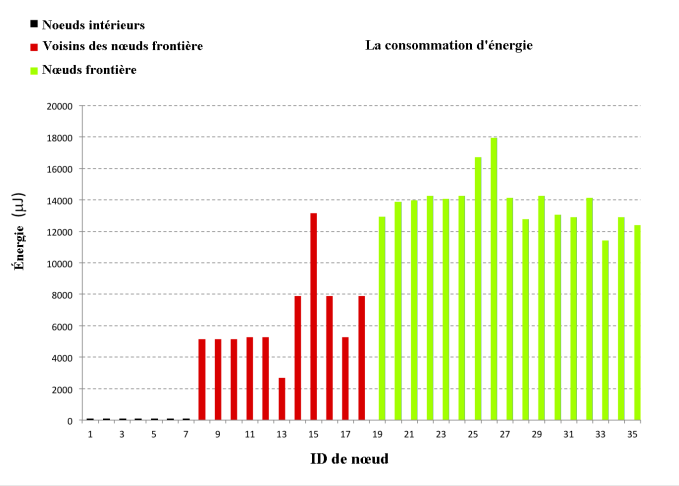


Figure 2.4: Consommation d'énergie des nœuds de la Figure 2.3.

Nous concluons que la consommation d'énergie ne dépend pas du nombre de nœuds du réseau ni du nombre de nœuds frontière, mais plutôt du nombre de voisins des nœuds frontière.

4.2.3 Comparaison avec les méthodes existantes

La Figure 2.5 présente la consommation d'énergie des nœuds frontière par rapport au nombre de leurs nœuds voisins, donné par certains algorithmes existants [60][65][67]. Elle montre que la consommation d'énergie augmente avec le nombre de nœuds voisins (degré).

Comme nous pouvons le voir, les consommations d'énergie obtenues par les algorithmes LVP, DBD, D-LPCN, sont proches les unes des autres (entre $10mJ$ et $150mJ$). Cependant, l'énergie obtenue par l'algorithme Hop-based est importante (entre $10,000mJ$ et $100,000mJ$). Cela est dû au nombre de sauts important requis par cet algorithme. L'algorithme D-LPCN consomme moins d'énergie que tous les autres. Cela peut être expliqué pour le cas de l'algorithme DBD par la nécessité d'utiliser des communications à 3 sauts, dans le cas de l'algorithme LVP par les calculs locaux importants (utilisation des diagrammes de Voronoï), et pour le cas de l'algorithme Hop-based par le nombre important de communications multi-sauts. En outre, l'algorithme D-LPCN utilise uniquement les nœuds de frontière et leurs voisins. Les autres nœuds restent stables en termes d'énergie.

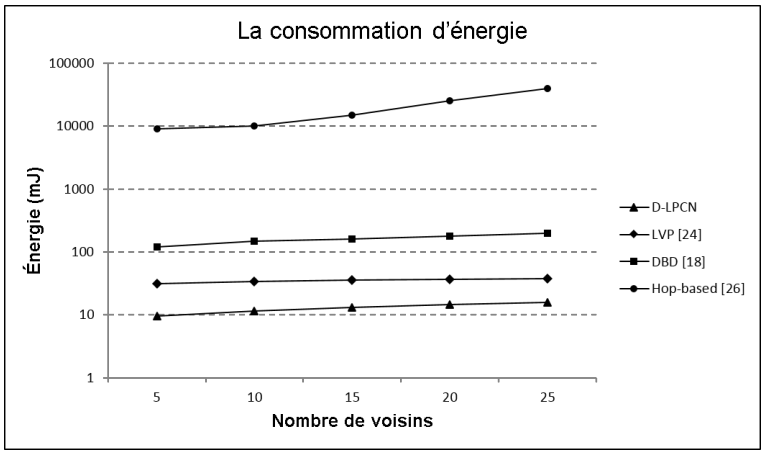


Figure 2.5: Comparaison de la consommation d'énergie avec celle des méthodes existantes.

5 Détection de nœuds défaillants

Dans cette section, nous présentons la méthode proposée pour la détection de défaillance de nœuds frontière nommée BNFD (Boundary Node Failure Detection) en utilisant l'algorithme D-LPCN présenté dans la section précédente et l'algorithme BM (Border Monitoring) pour tester la présence des nœuds, l'algorithme 5. Les principales étapes de cette approche sont présentées par l'organigramme de la Figure 2.7. Tout d'abord, l'algorithme D-LPCN sera exécuté sur un RCSF donné. À titre d'exemple, Figure 2.6 montre un RCSF avec 11 nœuds capteurs et ses nœuds frontière (c-à-d., S_1, S_2, S_3, S_4, S_5, S_6 et S_7). Pendant l'exécution de cet algorithme et à chaque itération, chaque nœud frontière stocke localement le id de son voisin frontière suivant. Une fois la frontière est entièrement déterminée, chaque nœud frontière S_i commence à envoyer périodiquement un message de test A "*Êtes-vous là ?*" afin de tester la présence de son voisin frontière comme montré par la Figure 2.6(a). Notez que dans cette figure, les messages-A sont envoyés par chaque nœud en même temps. Cependant, en réalité, ces messages seront envoyés séquentiellement afin d'éviter les collisions, tout comme décrit par l'organigramme de la Figure 2.7. Une fois qu'un message A est envoyé, l'émetteur attendra, pour un temps limité, un message B -"*Oui, je suis là*" de son prochain voisin frontière, qui est représenté par la flèche rouge dans les Figures 2.6(b) et (d). Si le voisin échoue, il ne peut pas répondre. Par conséquent, une fois que le temps limité est atteint, et qu'aucune réponse n'est reçue du voisin (voir Figure 2.6(c)), le voisin suivant sera déclaré comme un nœud défaillant (voir Figure 2.6(d)). Dans ce cas, une alarme sera déclenchée et l'algorithme D-LPCN sera réexécuté une autre fois pour recalculer une nouvelle frontière afin d'assurer toujours la fonction de surveillance.

L'algorithme 5 montre le pseudo-code de la procédure de test de présence, où chaque nœud envoie le message 'A' à son prochain voisin nid (voir lignes 2 et 3) une fois les nœuds frontière déterminés. Ensuite, chaque nœud attendra une période donnée t_1 pour un message de réponse 'B' de son prochain voisin (voir ligne 4). Si le nœud ne reçoit aucun message pendant cette période, son voisin sera déclaré comme un nœud défaillant (voir lignes 5 et 6). Dans ce cas, l'algorithme D-LPCN sera de nouveau exécuté pour trouver les nouveaux nœuds frontière et une alarme sera déclenchée. Sinon, si un nœud reçoit le message 'A' de son voisin précédent pid qu'il doit répondre

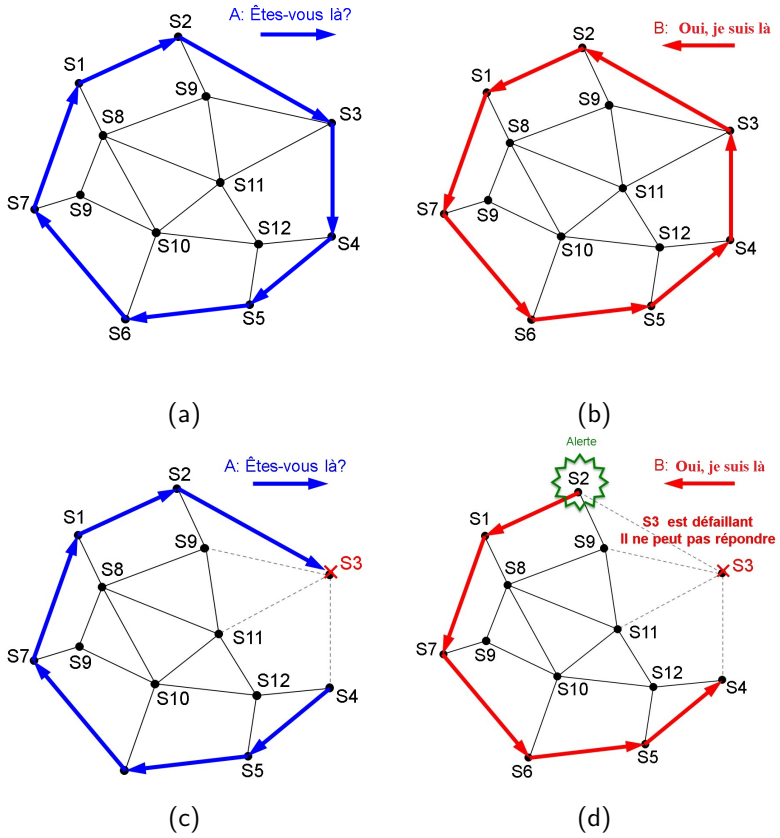


Figure 2.6: Algorithme de surveillance de frontières (BM).

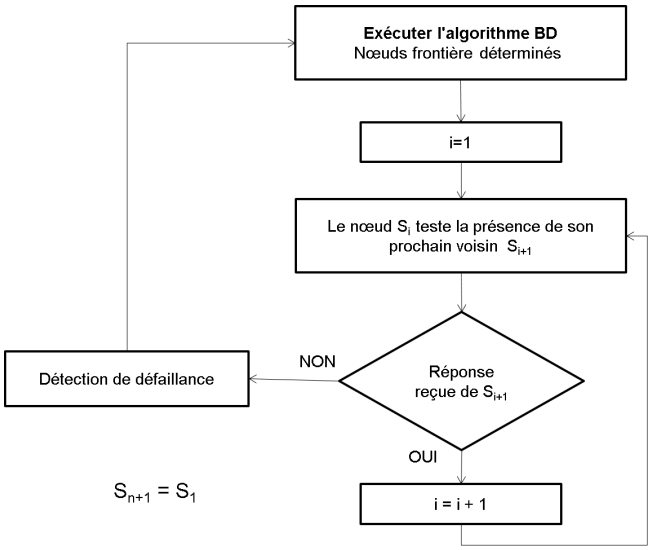


Figure 2.7: Diagramme de l'approche BNFD.

par le message 'B' pour confirmer qu'il ne s'agit pas d'un nœud défaillant (voir lignes 10, 11 et 12). La réception du message 'B' de son prochain voisin nid signifie que le nœud nid n'est pas défaillant (voir lignes 14 et 15). Cet algorithme est répété tous les t_2 secondes.

Algorithm 5 Algorithme de test de présence (BM)

```
1: while (true) do
2:     p = cid+"|"+"A";
3:     send(p, nid);
4:     n = read(t1);
5:     if (n=="") then
6:         Une défaillance est détectée: exécutez l'algorithme BD;
7:         break();
8:     else
9:         type = read();
10:        if (type==A) then
11:            p = cid+"|"+"B";
12:            send(p, pid);
13:        end if
14:        if (type==B) then
15:            Aucune défaillance détectée: ne rien faire;
16:        end if
17:    end if
18:    wait(t2);
19: end while
```

6 Évaluation des performances

L'objectif de cette section est d'évaluer la robustesse de la méthode proposée et son efficacité énergétique. Pour ce faire, nous avons utilisé le simulateur (framework) Castalia-3.3 [82] qui est fondé sur la plateforme OMNeT ++ 4.6. Il fournit des modèles radio et des canaux sans fil et contient un profil de consommation d'énergie réaliste.

Dans le scénario considéré, N nœuds capteurs sont uniformément déployés afin d'obtenir la couverture maximale de la région considérée. Le niveau de puissance de transmission de tous les nœuds capteur est fixé à $-5\ dBm$, le modèle de la radio utilisé par tous les nœuds capteurs est le TelosB équipé des modules radio CC2420 basés sur le standard IEEE-$802.15.4$.

Figure 2.8 montre le scénario de déploiement de $N = 9$ nœuds capteurs dans une zone rectangulaire de 30×30 mètres. En appliquant l'algorithme D-LPCN, nous avons obtenu les nœuds frontière suivants: $B = \{N_0, N_3, N_6, N_7, N_8, N_5, N_2, N_1\}$.

Les Figures 2.9 et 2.10 montrent l'énergie consommée par chaque nœud frontière à l'aide respectivement de l'algorithme D-LPCN et BM. L'énergie consommée par chaque nœud est donnée par la somme des consommations en mode transmission, réception, écoute et sommeil. Comme on peut le voir, l'algorithme D-LPCN consomme une énergie moyenne de 0.368 Joules, ce qui représente 0.002% de la capacité de batterie initiale utilisée dans le simulateur Castalia et qui est égal à 18720 Joules (c-à-d, une capacité de deux piles AA). Par conséquent, l'algorithme D-LPCN peut être exécuté autour de 50869 fois jusqu'à l'épuisement de la batterie. Si nous supposons qu'une exécution de l'algorithme est faite chaque 30 minutes, alors sur un jour, cet algorithme peut être exécuté 48

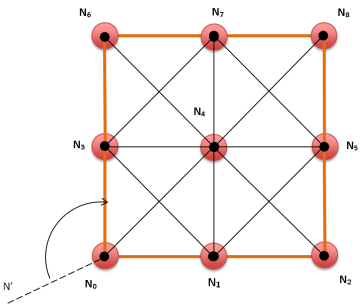


Figure 2.8: Le déploiement du réseau et la connectivité.

fois. Ainsi, une batterie peut être utilisée pour $50869/48 \approx 1060$ jours (c'est-à-dire, 3 années).

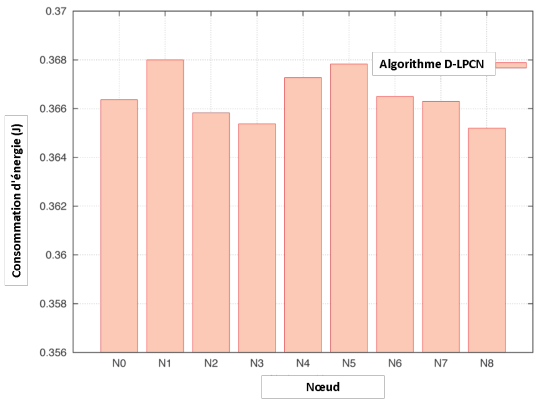


Figure 2.9: Consommation d'énergie par nœud (Algorithme D-LPCN).

De la même manière, pour le cas de l'algorithme BM, comme représenté par la Figure 2.10, chaque nœud consomme une énergie moyenne de 0.18 Joules, ce qui représente 0.001% de la capacité initiale de la batterie. Par conséquent, il peut être exécuté $104 \cdot 10^3$ fois jusqu'à l'épuisement de la batterie. Si l'on suppose qu'une exécution de l'algorithme est faite environ chaque 10 minutes, alors dans un jour cet algorithme peut être exécuté 144 fois. Ainsi, une batterie peut être utilisée pour $104000/144 \approx 722$ jours (c'est-à-dire, 2 années).

Par conséquent, comme représenté par la Figure 2.11, l'énergie moyenne totale consommée par l'approche BNFD (D-LPCN+BM) est égale à 0.548 Joules, ce qui représente 0.003% de l'énergie totale d'une batterie. Par conséquent, il peuvent être exécuté ensemble 18720 fois jusqu'à l'épuisement de la batterie. Basé sur les hypothèses données ci-dessus, c'est-à-dire, une exécution de l'ensemble d'algorithmes (D-LPCN + BM) est faite chaque $40 = (30 + 10)$ minutes, donc sur un jour ces deux algorithmes peuvent être exécutés 36 fois. Ainsi, une batterie peut être utilisée pour $18720/36 \approx 520$ jours (c'est-à-dire, 1.5 années).

Pour étudier l'impact de la taille du réseau sur la consommation d'énergie, nous varions le nombre de nœuds capteurs de 9 à 1000 et mesurons la moyenne de la consommation d'énergie des algorithmes D-LPCN et BNFD (D-LPCN+BM). La Figure 2.12 représente la consommation d'énergie sous

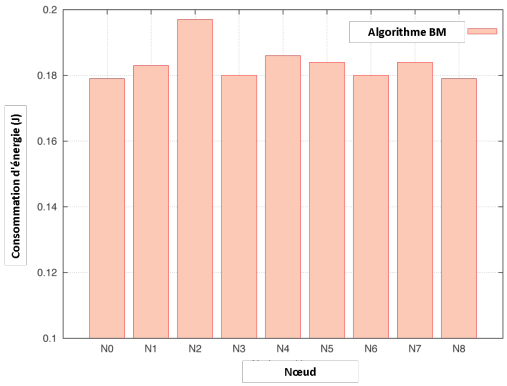


Figure 2.10: Consommation d'énergie par nœud (Algorithme BM).

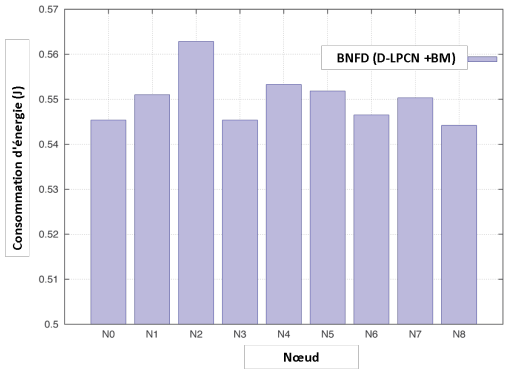


Figure 2.11: Consommation d'énergie par nœud BNFD (Algorithmes D-LPCN+BM).

différentes tailles de réseau. Elle montre que lorsque le nombre de nœuds déployés dans le champ d'intérêt augmente, la consommation moyenne d'énergie augmente également. En fait, c'est un résultat attendu parce que tous les nœuds sont impliqués dans le processus de communication. En effet, dans notre cas, nous sommes intéressés à surveiller un site sensible et à détecter les nœuds défaillants sur la frontière. Ainsi, si nous déployons des nœuds capteurs seulement autour de la frontière sous forme de couches, la consommation d'énergie diminue en raison de la réduction du nombre de nœuds qui participent au processus de la découverte des nœuds frontière car uniquement les nœuds frontière et leurs voisins sont impliqués dans ce processus, et le reste des nœuds à l'intérieur du réseau demeurent inactifs et serviraient uniquement pour remplacer les nœuds défaillants.

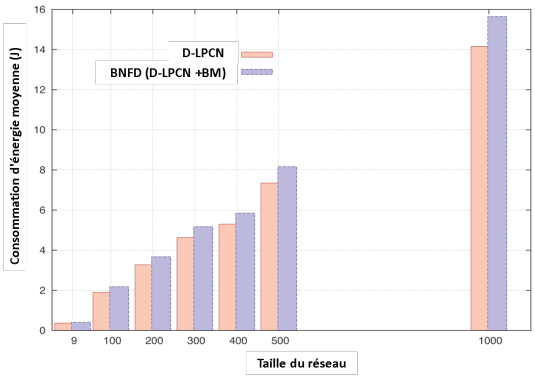


Figure 2.12: Consommation moyenne d'énergie (Algorithmes D-LPCN et BNFD).

Enfin, il est évident que l'algorithme D-LPCN est exécuté la première fois pour détecter la frontière, ensuite il est réexécuté uniquement lorsque un nœud échoue pour déterminer la nouvelle frontière. De plus, l'occurrence des défaillances de nœuds n'est pas fréquente, et par conséquent, la consommation d'énergie de l'approche proposée est largement inférieure aux valeurs cités ci-dessus. À noter, ces résultats ne tiennent pas compte de certains phénomènes tels que l'auto-décharge de la batterie.

7 Conclusion

Dans cet article, nous avons proposé une approche distribuée pour détecter les nœuds défaillants dans une frontière de réseaux de capteurs sans fil. Notre nouvelle approche détecte les nœuds frontière d'un RCSF et surveille la zone sensible souhaitée avec ces nœuds situés sur la frontière. Grâce à de nombreuses simulations avec le simulateur Castalia, nous avons évalué les performances de notre approche dans diverses conditions et nous avons montré son efficacité énergétique. Dans cette approche, le nombre de messages échangés pour identifier les capteurs défaillants est faible et une quantité substantielle d'énergie peut être économisée par les nœuds capteurs. De plus, l'approche proposée surpasse celle des précédentes en fournissant une haute précision de détection. Comme travaux futurs, nous envisageons d'analyser notre approche en termes de résilience aux changements des défaillances, en injectant des défauts dans le réseau, et d'étudier la performance du réseau dans ce cas et comment router immédiatement les notifications d'échec à travers le réseau.

3

Détection des anomalies et des capteurs usés

1 Introduction

Un réseau de capteurs sans fil (RCSF) se compose de nœuds sans fil à faible coût, travaillant sur des batteries, qui effectuent un processus de mesure collaboratif afin de percevoir des conditions physiques ou environnementales telles que la température, l'humidité, la lumière, le son, les vibrations, la pression, le mouvement, les polluants, etc. [48].
Cette fonctionnalité des réseaux de capteurs offre une large gamme d'applications à savoir, la surveillance militaire, la surveillance de l'habitat, la détection sismique, les applications de sécurité et de santé, etc.
En outre, l'objectif d'un RCSF est non seulement de collecter des données de la zone d'intérêt, mais aussi d'analyser ces données au moment opportun afin de prendre des décisions importantes [83].
En plus, les mesures brutes collectées par les nœuds capteurs sont souvent peu fiables et souffrent d'imprécision et d'incomplétude pour diverses raisons, telles que le bruit, les erreurs, les événements imprévus, les attaques malveillantes. D'autres raisons liées à l'environnement, y compris la dureté et les difficultés de la zone de déploiement peuvent également conduire à des données incorrectes [84].
De plus, les nœuds capteurs sont sujets à l'échec et sont limités en ressources (puissance, capacités de calcul, mémoire), ce qui peut également contribuer à générer des mesures inexactes et affecter la fiabilité des données [85].
De surcroît, les nœuds qui ont des mauvais capteurs (capteurs usés) peuvent aussi générer des données incorrectes, ce qui pourrait affecter l'analyse des données, éviter de prendre des décisions correctes et, au-delà, conduire à un gaspillage des ressources limitées dont disposent ces noeuds capteurs [24].
Afin d'assurer les performances du réseau et d'étendre la durée de vie de celui-ci, le RCSF devrait pouvoir détecter les mésures incorrectes, et par la suite, les capteurs défectueux si l'on existe, éviter l'échange de données erronées sur le réseau et maintenir les ressources des noeuds capteurs à un niveau élevé [86].
Par conséquent, assurer la fiabilité et la précision des données de manière efficace est une tâche nécessaire pour le processus décisionnel et la préservation des ressources limitées du réseau [86]. Pour atteindre cet objectif, un système robuste de détection des anomalies et des capteurs usés pour les RCSFs est essentiel.
Dans ce chapitre, nous présentons deux approches entièrement distribuées pour la détection en ligne des anomalies et les capteurs usés dans un RCSF, en se basant sur la théorie des Copules qui est un outil puissant pour modéliser et analyser les distributions multivariées. Pour la première approche, un polygone local résultant de l'application de cette théorie à des mesures historiques de données

est téléversé dans chaque nœud. Les anomalies sont alors détectées en comparant le flux de données d'entrée lié à la mesure et à la surveillance de phénomènes physiques avec ce polygone. Si les valeurs détectées n'appartiennent pas à ce polygone, elles sont considérées comme des anomalies.

Ensuite, pour la seconde approche, le même polygone, qui a été construit auparavant par l'application de la théorie des Copules sur des mesures historiques de données, sera assigné à chaque nœud capteur dans le réseau. Par la suite, les nœuds capteurs usés sont alors détectés en calculant la probabilité pendant une période de temps T qu'une valeur détectée se trouve à l'extérieur de ce polygone. Si la probabilité calculée est inférieure à un seuil prédéfini, le nœud capteur est considéré comme bon, sinon le nœud capteur est déclaré mauvais et une alerte sera envoyée à la station de base.
Ce chapitre est organisé en trois parties principales comme suit:
La première partie présente quelques définitions et concepts de base (Section 2), suivis par une étude bibliographique sur les méthodes de détection des anomalies dans un RCSF (Section 2.2), puis, les méthodes permettant la détection des capteurs usés dans un RCSF (Section 2.3). Par la suite, Section 3 présente une introduction à la théorie des Copules.
Dans la deuxième partie, la Section 4 présente la méthode proposée pour détecter les anomalies dans un RCSF. Ensuite, la Section 5 présente la méthode proposée pour détecter les capteurs usés dans un RCSF.
La troisième partie (Section 6) est consacrée à l'évaluation des performances et aux résultats de simulation des deux approches proposées.

2 Contexte et Travaux connexes

2.1 Définitions et Concepts de base

2.1.1 Caractéristiques des données des capteurs

Les données des capteurs sont collectées sous forme de flux de données qui représentent de grands volumes de données capturées par la surveillance et l'enregistrement des conditions à divers endroits [87]. En outre, les données des capteurs sont classées en données univariées et multivariées, selon les caractéristiques du phénomène surveillé. Pour les données univariées, les RCSFs sont conçus uniquement pour collecter un seul type de donnée telles que la température, l'humidité, la pression. Tandis que, pour les données multivariées, les nœuds d'un RCSF sont équipés de plus d'un capteur pour collecter simultanément différents types de données à partir du champ d'intérêt [88] [89].

2.1.2 Qu'est-ce qu'une anomalie ?

Le terme "anomalie", aussi connu sous le nom de "outlier" en anglais, vient du domaine de la statistique (Hodge et Austin, 2003). Les deux définitions classiques des anomalies sont:
(Hawkins 1980): "Une anomalie est une observation qui s'écarte tant d'autres observations qu'elle suscite des soupçons d'être générée par un mécanisme différent".
(Barnett et Lewis, 1994): "Une anomalie est une observation (ou un sous-ensemble d'observations) qui semble être incompatible avec le reste de l'ensemble des données".
Dans les RCSFs, les anomalies sont définies comme des écarts significatifs dans les mesures de données de détection par rapport au profil normal des données collectées [90].

2.1.3 Un évènement

Un "événement" dans un environnement sévère peut être considéré comme une "Condition désastreuse" qui peut changer l'état du monde réel et qui peut être caractérisé par un changement inattendu des conditions environnementales, par exemple, les feux de forêt, la pollution de l'air, etc. [91]. Par exemple, une anomalie se produit lorsqu'une lecture de capteur indique une température élevée indépendamment de ses capteurs voisins, alors qu'un flux continu de lectures de haute température (un flux d'anomalies) dans un ensemble de capteurs proches les uns des autres indiquent la présence d'un évènement.

2.1.4 Corrélation spatiale et temporelle

Dans les RCSFs, les lectures des capteurs sont corrélées spatialement et temporellement. Une corrélation spatiale signifie que les lectures de capteurs qui sont géographiquement proches les unes des autres devraient être semblables. Une corrélation temporelle signifie que les lectures collectées à une période de temps sont liées aux lectures collectées à la période précédente [92] [89]. En exploitant la corrélation spatiale et temporelle entre les attributs de lecture des capteurs, nous sommes en mesure de déterminer la source des anomalies et d'améliorer l'efficacité des modèles de détection [89].

2.2 Les méthodes de détection des anomalies dans un RCSF

Actuellement, les techniques de détection d'anomalie dans les réseaux de capteurs sans fil sont un domaine de recherche important. De plus, plusieurs schémas ont été proposés dans la littérature pour détecter les anomalies dans des données de capteurs de manière distribuée ou centralisée. Dans une architecture centralisée, toutes les données sont envoyées vers un emplacement central, par exemple, une station de base ou un Cluster-Head, pour une analyse ultérieure. Les techniques de détection des anomalies peuvent alors utiliser un modèle global construit à partir des données reçues pour détecter toute instance de donnée anormale. Dans une architecture distribuée, chaque nœud utilise ses propres lectures de données pour construire un modèle local et normal qui est utilisé ensuite pour le processus de détection d'anomalies, où le nœud doit exécuter tout ou une partie des opérations de traitement par lui-même, et collaborer avec ses voisins si nécessaire [93][94][88].

Sur la base des modèles de détection, ces schémas de détection sont classés en: basé-statistiques, basé-voisin le plus proche, basé-clustering, basé-classification et d'autres [95][89].

Dans les approches statistiques, les données des capteurs sont modélisées à l'aide d'une distribution stochastique, qui capture la distribution des données, et qui évalue les instances de données en fonction de leur adéquation au modèle trouvé. Une instance de données est déterminée comme étant anormale en fonction de sa relation avec ce modèle, c'est-à-dire si sa probabilité d'être généré par ce modèle est très faible.

Les auteurs de [96] ont proposé deux techniques locales pour l'identification des capteurs qui génèrent des anomalies ainsi que la détection des évènements dans les réseaux de capteurs. Pour identifier les capteurs, qui donnent de fausses lectures, chaque capteur calcule d'abord la différence entre sa propre lecture et la Médiane des lectures des capteurs voisins. Ensuite, chaque capteur collecte toutes les différences de son voisinage et les normalise. Un capteur est incorrect si la valeur absolue de sa différence normalisée est suffisamment supérieure à un seuil pré-sélectionné. La technique de détection d'événements est basée sur les résultats précédents de l'identification du capteur incorrect, et détermine un nœud comme un nœud d'évènement si la valeur absolue de l'écart de ce

nœud dans une zone géographique est beaucoup plus grande que dans une autre région. Cependant, ces méthodes ne fonctionnent que pour les données univariées.

Dans [97], une approche de détection des anomalies en ligne a été proposée pour les RCSFs. L'idée de base de cette approche est de comparer les mesures collectées avec une série chronologique de référence. L'algorithme (SSA-Segmented Sequence Analysis) a été utilisé pour construire un modèle linéaire par morceaux à partir d'une série temporelle de données de capteurs. Pour la détection des anomalies, nous comparons le modèle linéaire construit avec un modèle de référence à l'aide de mesures de similarité. En cas de différence significative, une alarme de défaut est signalée. Ce modèle compte deux couches de détection. La première couche est la détection au niveau du nœud local et la deuxième couche est dans le Cluster-Head. Cependant, il n'y a pas de retour d'information d'un emplacement central vers les nœuds locaux et le réglage dynamique des paramètres (par exemple, la période de linéarisation et la taille du modèle de référence initial) a été laissé pour des travaux futurs.

Dans [98], une technique locale (à base de Gauss) de détection des valeurs aberrantes pour identifier les erreurs et détecter les événements dans les applications écologiques de réseaux de capteurs. Chaque nœud apprend la distribution statistique des différences entre ses propres mesures et chacun de ses nœuds voisins, ainsi qu'entre ses mesures actuelles et précédentes. Une valeur de seuil approprié a été utilisé pour comparer la déviation avec les données des capteurs voisins. Si l'écart est supérieur au seuil prédéfini la mesure est considérée comme anormale. La mesure anormale détectée peut être considérée comme évènement si elle est susceptible d'être temporellement différente de ses mesures précédentes, mais spatialement corrélées.

Dans [99], une technique statistique qui utilise une distribution symétrique α-stable pour modéliser les valeurs aberrantes venant en forme de bruit impulsif. La technique utilise les corrélations spatio-temporelles des données de capteurs pour détecter localement les valeurs aberrantes. Chaque nœud dans un cluster détecte et corrige les valeurs aberrantes temporelles en comparant les données prédites et les données de détection. Le Cluster-Head reçoit les données rectifiées de tous les nœuds du cluster, et détecte sur la base de la corrélation spatiale, les valeurs aberrantes qui s'écartent notablement des autres données normales. Cette technique réduit le coût de communication dû à la transmission locale, et réduit également le coût de calcul parce que le Cluster-Head effectue la plupart des tâches de calcul. Cependant, la distribution α-stable peut ne pas convenir pour des données réelles et la structure fondée sur les clusters peut être sensible à des changements dynamiques de la topologie du réseau.

Dans [100], un histogramme de données de détection est envoyé à la station de base au lieu des données elles-même, la station de base ensuite est chargée d'extraire la distribution des données à partir des histogrammes reçus. L'identification des valeurs aberrantes est atteinte par une distance de seuil fixe. Cependant, cette technique n'est applicable que pour des données unidimensionnelles (univariées).

Dans [101], un système de détection des anomalies en ligne à base d'histogramme a été proposé pour un RCSF hiérarchique. Ce système est développé pour pallier les inconvénients des systèmes à base d'histogramme-existants, dont la procédure de vérification qui génère un coût de calculs et une surcharge de communication très élevés. Pour résoudre ce problème, les auteurs ont introduit une approche d'estimation simple qui fonctionne en ligne et d'une manière répartie. En revanche, certains inconvénients existent encore dans ce régime telles que la sélection de paramètre, et l'incapacité à faire face aux changements inattendus de comportement du modèle normal. En outre, comme

d'autres systèmes basés sur l'histogramme, ils sont limités à des données à une variable et ne peuvent pas prendre en charge les données multivariées ce qui limite la portée de ces systèmes.

Les approches basées sur le voisin le plus proche, calculent le degré de voisinage pour chaque instance de données et analysent le voisinage pour déterminer si l'instance de données est anormale ou non. En outre, ces approches sont classées en deux sous-méthodes, des méthodes basées sur la distance et d'autre basées sur la densité. Dans les méthodes basées sur la distance, les anomalies sont les instances de données les plus éloignées des autres instances de données, alors que dans les méthodes basées sur la densité, les anomalies sont les points de données dans des régions de faible densité.

Les auteurs de [102] ont proposé une technique basée sur la distance pour calculer les n top anomalies globales dans la structure d'un arbre d'agrégation, où chaque nœud d'arbre transmet quelques données utiles à son parent après avoir collecté toutes les données envoyées par ses fils, puis la station de base inonde les valeurs aberrantes (anomalies) à tous les nœuds du réseau pour vérification. Si un nœud trouve qu'il a deux types de données qui peuvent modifier le résultat global, il les envoie à son parent dans un intervalle de temps approprié. Cette procédure est répétée jusqu'à ce que tous les nœuds du réseau soient en accord avec le résultat global calculé par la station de base. Cependant, cette technique ne prend en compte que des données unidimensionnelles.

Dans [103], les auteurs ont proposé une technique de détection des valeurs aberrantes basée sur la densité, pour détecter les anomalies locales en utilisant la technique de k-distance voisinage, qui est basée sur un facteur baptisé LOF (Local Outlier Factor). Pour toute mesure donnée, LOF est la mesure du degré pour qu'une lecture collectée soit une valeur aberrante. Une valeur est marquée comme anomalie, si son LOF est significativement plus élevé que celui de ses voisins locaux. Par ailleurs, les auteurs ont présenté de nouvelles techniques de réduction de la complexité de calcul impliquée dans le calcul de LOF.

Dans les approches basées sur le clustering, les instances de données normales appartiennent à des clusters grands et denses, alors que les anomalies n'appartiennent à aucun des clusters ou forment des clusters très petits.

Les auteurs de [104, 105] ont proposé un algorithme de détection des anomalies distribuées basé sur le regroupement de données (clustering), et l'utilisation de clusters hypersphériques (appelés cluster à largeur fixe), pour détecter les anomalies globales. Cette technique suppose une structure de réseau hiérarchique. Les données de chaque nœud sont regroupées en utilisant les clusters hypersphériques ce qui donne un ensemble de clusters de largeur fixe. Le regroupement est alors effectué sur la base de la distance Euclidienne entre les vecteurs de données. Les nœuds capteurs envoient ensuite des résumés de leurs clusters qui peuvent être suffisamment représentés par un centroïde et le nombre de vecteurs de données qu'ils contiennent. Les nœuds capteurs intermédiaires fusionnent ensuite les résumés de cluster avant de communiquer avec d'autres nœuds. Ce processus se poursuit récursivement jusqu'au nœud puits, où un algorithme de détection d'anomalie est appliqué à ses clusters fusionnés pour identifier le cluster anormal. Le cluster est déclaré comme un cluster anormal, si la distance intercluster moyenne de n'importe quel cluster est supérieure à un écart type de la moyenne des distances intercluster de tous les clusters. Le résumé du cluster anormale global est communiqué à tous les nœuds du réseau. Ensuite, chaque nœud, en utilisant le cluster anormal global , identifie les vecteurs de données globalement anormaux correspondants au nœud. De plus, dans l'approche de [105], des anomalies locales et globales sont détectées, les

anomalies locales en considérant les mesures locales du nœud. Cependant, le principal inconvénient de ces deux techniques est le seuil de largeur fixe, ce qui n'est pas facile de choisir.

Dans les approches basées sur la classification, l'idée principale est de construire un modèle de classification pour les données normales et anormales en se basant sur des données d'apprentissage connues, ensuite, utiliser ce modèle pour classer chaque nouvelle instance de données inconnues.

Les auteurs de [106] proposent un algorithme intégré de compression de données et de détection d'anomalies dans les RCSFs, qui peut détecter des anomalies avec précision en combinant deux techniques (DWT) (Discrete Wavelet Transform) et OCSVM (A Single Class Support Vector Machine). Dans cette approche, DWT a été utilisé pour compresser les mesures de données à chaque nœud. Ensuite, OCSVM est utilisé pour classer les données en données correctes et anormales. En revanche, l'inconvénient majeur de ce schéma est la surcharge de communication élevée causée par la transmission de l'ensemble des données à la station de base et le coût de calcul élevé de la technique DWT.

Les auteurs de [107] ont proposé un algorithme de détection d'anomalie basé sur la technique S transform et SVM. S transform est appliquée afin de réduire la taille des données transmises, et d'extraire les composantes significatives de ces données de séries temporelles sans perdre les caractéristiques significatives, puis SVM est utilisé pour classer les données en normal et anormal. En revanche, cette approche génère des coûts de calculs très élevés car elle doit classer chaque composante extraite pour la décision finale.

Les auteurs de [108, 109, 110] ont proposé trois types de réseaux bayésiens pour détecter les anomalies dans un réseaux de capteur (RCSF), qui sont le Réseau NB (Naïve Bayesian) [108], le réseau BBN (The Belief Bayesian Network) [109], et le réseaux DBN (Dynamic Bayesian Network) [110]. Dans [108], les auteurs ont proposé une technique pour modéliser et apprendre la corrélation spatiale entre les nœuds spatialement adjacents ainsi que la corrélation temporelle entre les lectures historiques du même nœud. Ensuite, le nœud peut prédire localement ses lectures actuelles en se basant sur ses propres lectures passées et les lectures actuelles de ses voisins. En plus de cela, les auteurs de [109] considèrent les dépendances conditionnelles entre les attributs des données de capteurs pour améliorer la précision de la détection. Les deux techniques bayésiens présentées ci-dessus, ne s'adaptent pas à la nature dynamique du changement de réseau. Les auteurs de [110], ont proposé le réseaux DBN pour remédier à ce problème, en calculant la probabilité a posteriori des paternes de données récentes dans une période de temps spécifique appelée fenêtre de temps (sliding window). Cependant, cette technique génère une complexité de calcul très élevée.

Les auteurs de [111] ont proposé un algorithme de détection des anomalies, basé sur un algorithme de fusion de données. Cette méthode utilise l'algorithme PAA (Piecewise Aggregate Approximation) pour compresser les données chronologiques de chaque nœud. Ensuite, sur la base du résultat de PAA, les données compressées sont classées en données normales et anormales en utilisant les techniques KMeans et AIS (Artificial Immune System). Cependant, cette approche ne prend pas en compte les données multivariées.

À notre connaissance, la théorie des Copules n'a pas encore été appliquée à la détection d'anomalies dans les RCSFs. Par conséquent, le but ici, est de proposer une approche de détection en ligne, distribuée, efficace pour détecter les anomalies dans les lectures des capteurs, et qui maintient les ressources des RCSFs à un niveau élevé. Enfin, nous pouvons classer notre méthode

proposée dans les approches statistiques qui fournissent un taux de détection élevé et qui réduisent les surcharges de communication.

2.3 Les méthodes de détection des capteurs usés dans un RCSF

La détection de nœuds avec des capteurs défectueux (usés) peut influencer strictement les performances du réseau et prolonger la durée de vie du réseau. Les capteurs défectueux doivent être exclus ou remplacés dans le réseau. Ainsi, l'objectif principal de notre recherche est de proposer une approche distribuée efficace pour détecter les nœuds qui ont des capteurs défectueux dans les RCSFs. Nous examinons dans cette section certains modèles qui ont été proposés dans la littérature.

Dans [112], une méthode de détection des capteurs usés dans les réseaux de capteurs sans fil a été proposée baptisée (FIND). FIND détecte les nœuds capteurs usés en se basant uniquement sur leurs résultats de mesures relatives. L'idée principale de FIND est de considérer un nœud capteur défectueux si ses lectures violent considérablement la monotonie de la distance. FIND est basé sur le fait que les lectures sensorielles changent de façon monotonique lorsque la distance devient plus grande des nœuds à l'événement surveillé. FIND classe les nœuds en fonction de leurs lectures sensorielles ainsi que de leurs distances physiques de l'événement. Ensuite, s'il existe un décalage significatif entre le rang de données du capteur et le rang de distance, le nœud capteur est considéré comme défectueux. Cependant, cette approche est centralisée et génère ainsi une surcharge de communication très élevée.

Dans [113], les auteurs ont proposé une stratégie basée sur la modélisation d'un nœud capteur par un système d'inférence floue nommé FIS (Fuzzy Inference System), où une mesure d'un nœud capteur est approximée par une fonction de mesures réelles des nœuds voisins. Le nœud capteur est déclaré défectueux si la différence entre la valeur réelle détectée sur un nœud et la valeur estimée donnée par son modèle FIS correspondant est supérieure à un seuil donné. Cependant, cette approche génère une complexité de calcul élevée qui ne convient pas aux RCSFs.

Dans [114], une technique de détection statistique en ligne pour la détection des capteurs défectueux est présentée, cette dernière applique une méthode statistique non paramétrique afin d'identifier les capteurs ayant la plus grande probabilité d'être défectueux.

Les auteurs dans [96] ont proposé une technique locale pour l'identification des capteurs défectueux dans les RCSFs. Pour identifier les capteurs qui donnent des fausses lectures, chaque capteur calcule d'abord la différence entre sa propre lecture et la médiane des lectures voisines. Ensuite, chaque capteur collecte toutes les différences de ses voisins et les normalise. Un capteur est défectueux si la valeur absolue de sa différence standardisée est suffisamment supérieure à un seuil présélectionné.

Dans [115], les auteurs ont proposé un algorithme localisé de détection de défauts pour identifier les capteurs défectueux. En utilisant une comparaison locale avec un vote majoritaire, chaque nœud capteur prend une décision basée sur la comparaison entre ses propres lectures et celles de ses voisins, tout en considérant le niveau de confiance de ses voisins. Cependant, cette approche nécessite un grand nombre de messages à échanger entre les nœuds voisins pour identifier les capteurs défectueux, ce qui est très coûteux en termes de consommation d'énergie.

Dans [116], l'algorithme proposé est constitué de trois différentes phases a savoir, la phase de

collecte de données, la phase d'analyse de données et la phase de décision. Au début, chaque nœud capteur invite ses voisins en envoyant ses propres données détectées à un instant donné T. Durant la même durée T, chaque nœud accumule toutes les données détectées envoyées par ses voisins qui sont disponibles dans sa portée de transmission Tr. Puis il prédit ses données probables à l'instant T à partir des données de ses voisins dans la phase d'analyse en utilisant la technique Z-test. Après prédiction de données, il compare ses données détectées avec les données prédites en phase de décision pour calculer son statut de base et enfin, il décide s'il est mauvais ou non.

3 Introduction à la Théorie des Copules

Dans les RCSFs, la détection des anomalies avec des données univariées peut facilement être effectuée en notant qu'une instance de données de l'attribut unique est anormale par rapport aux autres instances de données. Cependant, dans les RCSFs multivariés, il est difficile de détecter les anomalies, parce que les attributs individuels peuvent ne pas montrer un comportement anormal, bien que, lorsqu'ils sont pris ensemble, ils peuvent afficher un comportement anormal. Par conséquent, exploiter les dépendances entre les différents attributs de lectures de capteurs, nous permet de débloquer ces difficultés, et de proposer une méthode de détection à haute précision. C'est pourquoi nous avons décidé d'utiliser la théorie des Copules, car elle permet de modéliser la dépendance entre les données multivariées.

3.1 Fondements mathématiques de la théorie des Copules

Dans ce chapitre, et pour des raisons de simplicité, nous présentons seulement la théorie des Copules bivariées, car la théorie multivariée n'est qu'une extension du cas bivarié. À cette fin, nous rappelons quelques définitions de base et des théorèmes qui seront utiles plus tard.

3.1.1 Définition d'une Copule

La notion de Copule a été introduite par Sklar en 1959 [117], motivée par le travail de Fréchet dans les années 1950 [118].

Formellement, **une Copule bivariée [23] est** une fonction de répartition conjointe dont les marginaux sont uniformes sur $[0\,,1]$.

Une Copule $C : [0\,,1]^2 \to [0\,,1]$ est une fonction qui satisfait les trois conditions suivantes:

- $C(u,0) = C(0,v) = 0$ pour chaque u , $v \in [0\,,1]$,
- $C(u,1) = u$ and $C(1,v) = v$ pour chaque u , $v \in [0\,,1]$,
- C est une fonction croissante d'ordre 2, c-à-d, pour chaque $0 \leq u_i \leq v_i \leq 1$, $C(v1,v2) - C(v1,u2) - C(u1,v2) + C(u1,u2) \geq 0$.

3.1.2 Théorème de Sklar (cas bivarié):

Soit $H(x,y)$ une fonction de répartition de dimension 2 avec deux marginales F et G. Il existe une Copule C telle que pour tout réel (x,y), on a $F(x) = U$ et $G(x) = V$, où U=$(u_1,...,u_n)$ et V=$(v_1,...,v_n)$ sont deux variables uniformément distribuées sur I $= [0\,,1]$. La fonction $H(x,y)$ peut être écrite en termes d'une seule fonction $C(u,v)$ comme suit:

$$H(x,y) = C(F(x),G(y)) \tag{3.1}$$

Si F et G sont continues, alors la Copule C est unique; sinon, C est déterminée de façon unique sur (rang de F) $\times$ (rang de G). Inversement, si C est une Copule, F et G sont des fonctions de répartitions, alors $H(x, y) = C(F(x), G(y))$ est une fonction de répartition jointe avec F et G comme marginales.

Schématiquement: étant donné, les marginales de chaque variable, nous les joignons simplement avec une fonction Copule ayant les propriétés de dépendance souhaitées, afin d'obtenir la distribution jointe de toutes les variables. La Figure 3.1 montre comment obtenir la fonction de distribution jointe de toutes les variables.

Figure 3.1: La structure de dépendance.

3.2 La Copule empirique

Pour modéliser la dépendance observée entre les variables aléatoires, nous pouvons utiliser la structure empirique de la Copule pour évaluer l'adéquation d'une Copule choisie pour le paramètre estimé.

Il est nécessaire d'introduire la notion de rang avant de donner la formule d'une Copule empirique. Étant donné un échantillon $x_1, ..., x_n$ à partir d'une variable aléatoire X, le rang R_i de x_i est défini comme étant le nombre d'observations qui sont inférieures ou égales à x_i. Alors, la plus petite observation de X a le rang 1 tandis que le plus grand a le rang n.

Soit U=$(u_1, ..., u_n)$ et V=$(v_1, ..., v_n)$ deux variables uniformément distribuées sur I $= [0, 1]$.

Nous notons (X,Y) deux variables aléatoires telles que $X = (x_1, ..., x_n)$ et $Y = (y_1, ..., y_n)$, et nous mettons R_i le rang de x_i, S_i le rang de y_i.

Pour un échantillon spécifique (x_i, y_i) choisi de (X,Y), on peut approximer la Couple correspondante (u_i, v_i) en utilisant les rangs R_i et S_i de x_i et y_i parmi $x_1, \ldots, x_n$ et $y_1, \ldots, y_n$ comme suit :

$$u_i = \frac{R_i}{n+1} \tag{3.2}$$

$$v_i = \frac{S_i}{n+1} \tag{3.3}$$

Ainsi, en utilisant ces rangs, nous pouvons construire une fonction de distribution empirique pour la variable aléatoire X et Y [119].

Formellement, le calcul de la Copule empirique est donné par l'équation suivante [120] :

$$C_n(u, v) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(\frac{R_i}{n+1} \leqslant u, \frac{S_i}{n+1} \leqslant v) \tag{3.4}$$

La fonction $\mathbb{1}_{arg}$ est la fonction indicatrice, qui est égale à 1 si arg est vrai et égale à 0 sinon.

3.3 Familles de Copules

Il existe différents types de Copules, chaque Copule exprime une structure de dépendance différente:

- Dépendance dans les valeurs petites.
- Dépendance dans les valeurs extrêmes.
- Dépendance de queue.
- Dépendance positive ou négative.

Et nous avons deux grandes familles de Copules, la famille des Copules Archimédienne et la famille des Copules Elliptique. La première famille de Copules comprend :

- Copule de Gumbel : Dépendance positive et plus accentuée sur la queue supérieure.
- Copule de Frank : dépendance aussi positive que négative.
- Copule de Clayton : Dépendance positive, surtout sur les évènements de faible intensité.
- Copule HRT : Dépendance à des évènements extrêmes de forte intensité (structure de dépendance inverse à la Copule de Clayton).

La seconde famille s'applique aux distributions symétriques et elle compte deux types de Copules : la Copule Gaussienne et la Copule de Student. Nous présentons ci-dessous une présentation rapide de la Copule gaussienne, car c'est prouvée dans la Section 6 comme la famille de Copules qui ajuste au mieux la Copule empirique.

La Gopule Gaussienne est dérivée de la distribution Gaussienne multivariée. Soit Φ la distribution Normale univariée standard, dont la formule est donnée par:

$$\Phi(x) = \int_{-\infty}^{x} \phi(t)\,\mathrm{d}t \tag{3.5}$$

Où $\phi(t)$ est la densité de probabilité de la variable aléatoire $t \sim N(\mu, \sigma)$ avec la moyenne de la distribution $\mu = 0$ et l'écart type $\sigma = 1$:

$$\phi(t) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{t^2}{2}\right\} \tag{3.6}$$

Et nous avons $\Phi_{\Sigma,m}$ la distribution Gaussienne m-dimensionnelle avec la matrice de corrélation Σ avec 1 sur la diagonale et le coefficient de corrélation $\rho \in (-1, 1)$ sinon. Alors, la m-Copule Gaussienne avec la matrice de corrélation Σ est donnée par :

$$C_\rho(u_1, ..., u_m) = \Phi_{\Sigma,m}(\Phi^{-1}(u_1), ..., \Phi^{-1}(u_m)) \tag{3.7}$$

dont la densité est :

$$c_\rho(\phi(x_1), ..., \phi(x_m)) = |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\mathbf{X}^{\mathbf{T}}(\Sigma^{-1} - I_m)\mathbf{X}\right\}, \tag{3.8}$$

où $X = (x_1, ..., x_m)$ et $U = (u_1, ..., u_m)$. Alors, en utilisant $u_i = \phi(x_i)$ on peut écrire de manière équivalente :

$$c_\rho(u_1, ..., u_m) = |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\zeta^{\mathbf{T}}(\Sigma^{-1} - I_m)\zeta\right\}, \tag{3.9}$$

où $\zeta = (\phi^{-1}(u_1), ..., \phi^{-1}(u_m))^T$

Par conséquent, à partir de la formule (3.7), la Copule Gaussienne bivariée est définie par :

$$C_\rho(u, v) = \Phi_\Sigma(\Phi^{-1}(u), \Phi^{-1}(v)) \tag{3.10}$$

où Σ est la matrice de corrélation, qui est une matrice 2×2 avec 1 sur la diagonale et le coefficient de corrélation ρ sinon.

Φ_Σ indique la fonction de répartition pour une distribution Normale bivariée avec moyenne nulle et matrice de covariance Σ. Puis, à partir de la formule (3.8) et après simplification, sa densité jointe bivariée est donnée par :

$$c_\rho(\phi(x_1), \phi(x_2)) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left\{-\frac{x_1^2 - 2\rho x_1 x_2 + x_2^2}{2(1-\rho^2)}\right\} \tag{3.11}$$

où $X = (x_1, x_2)$ et $U = (u_1, u_2)$ tels que $u_i = \phi(x_i)$.

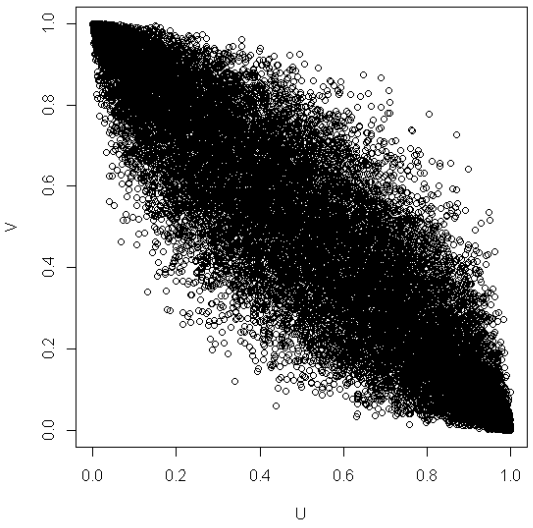


Figure 3.2: Exemple d'une Copule Gaussienne avec $\rho = -0.9$.

.

La Figure 3.2 montre un exemple de Copule Gaussienne de valeur $\rho = -0.9$:

3.4 Dépendogramme

Les Dépendogrammes nous permettent de comprendre graphiquement la structure de dépendance entre deux variables aléatoires. De plus, nous avons obtenu ces dépendogrammes par un graphique en nuage de points de marges uniformes (u, v) extraites d'un échantillon ou résultant de simulations d'une Copule théorique. Dans la Figure 3.3, nous présentons quelques dépendogrammes qui sont bien connus pour certaines familles de Copules.

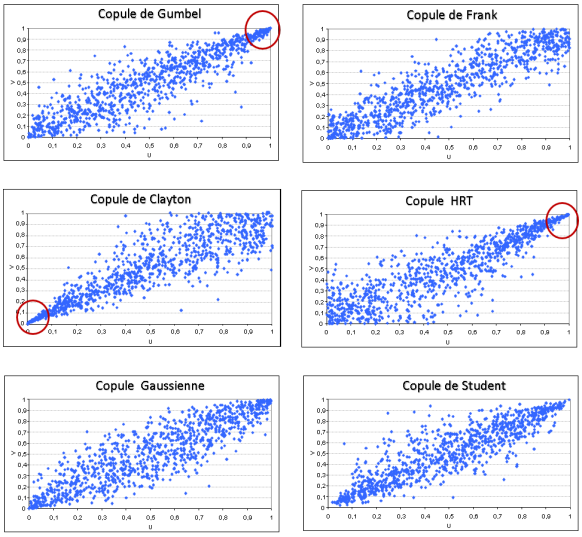


Figure 3.3: Dépendogrammes pour familles de Copules en 2D.

3.5 Choix de la meilleure Copule

Lors du traitement de données bivariées, nous supposons que nous avons un sous-ensemble fini de Copules, et nous sommes intéressés de savoir laquelle d'entre elles correspond le mieux aux données. Dans ce cas, nous devons choisir entre l'adéquation graphique et la méthode analytique.

3.5.1 Adéquation graphique

Dans cette méthode, nous comparons le dépendogramme empirique avec les dépendogrammes théoriques. Cette méthode permet de trouver un ensemble de Copules restreintes qui peuvent être utilisées pour ajuster la Copule empirique. Dans ce travail, nous utilisons l'adéquation graphique et la Copule choisie est validée en calculant la différence de surface entre la Copule empirique et la Copule théorique, générée à chaque fois avec un nouveau paramètre. Enfin, nous calibrons la Copule théorique avec le paramètre qui minimise cette différence de surface.

3.5.2 Adéquation analytique

Il existe de nombreuses méthodes proposées dans la littérature qui ont pour but de trouver la Copule qui corresponde le mieux aux données, et nous appelons ces méthodes Goodness of fit (GOF) test, parmi lesquels nous citons le test de Kolmogorov-Smirnov [121]. Dans ce test, nous comparons simplement la distribution empirique avec la distribution théorique de chaque famille et vérifions si les deux proviennent de la même distribution.

4 Détection des anomalies

Dans cette section, nous présentons la méthode proposée pour la détection des anomalies dans un RCSF en se basant sur la théorie des Copules. D'abord, nous présentons l'idée générale de l'approche. Ensuite, nous décrivons les étapes à suivre pour construire le modèle qui sera utilisé

pour la détection en ligne. L'approche proposée est divisée en deux étapes. Dans la première étape, nous construisons le modèle qui sera utilisé pour détecter les anomalies hors ligne, ensuite, dans la deuxième étape, le modèle obtenu sera téléversé dans l'ensemble des nœuds capteurs pour être utilisé par la suite, dans la détection des anomalies en ligne.

Dans ce qui suit, nous expliquons les étapes nécessaires pour la construction du modèle hors ligne pour le cas bivarié.

4.1 Construction du modèle (hors ligne)

L'organigramme présenté dans la Figure 3.4 décrit le processus suivi, et résume toutes les phases nécessaires pour construire le modèle proposé, pour assurer la fiabilité et la détection efficace avec une grande précision. Les différentes phases à suivre pour la construction du modèle au cours des différentes étapes peuvent être décrites comme suit :

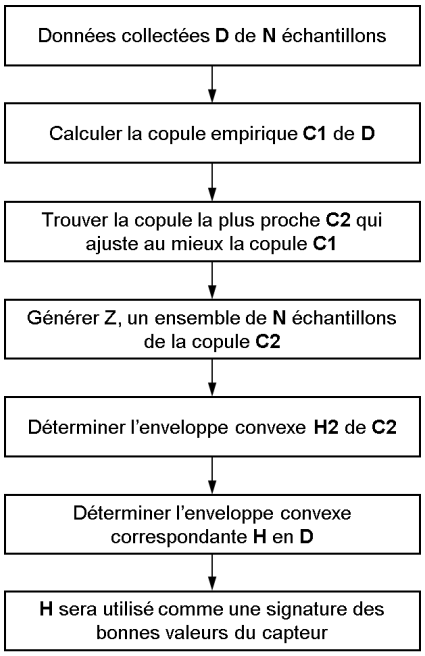


Figure 3.4: Traitement hors ligne des données des capteurs

- *Étape 1*: Sur la base de mesures historiques des données détectées, nous commençons par choisir N échantillons de deux variables aléatoires X et Y qui représentent deux attributs de données détectées telles que la température et l'humidité. Alors, soit $D = (X, Y)$ le jeu de données qui sera utilisé pour la construction du modèle basé sur une Copule bivariée avec $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_n)$.
- *Étape 2*: Dans cette étape, nous calculons la Copule empirique C_1 de D.

  Soit un échantillon aléatoire (x_1,y_1),...,(x_n,y_n) de D, on note R_i le rang de x_i et S_i le rang de y_i.

  Ensuite, nous calculons la Copule bivariée empirique C_1 de D en utilisant la formule (3.4).

- *Étape 3*: Dans cette étape, nous devons trouver la Copule théorique la plus proche C_2 qui ajuste le mieux la Copule empirique C_1. Dans notre cas, nous le faisons par la méthode d'adéquation graphique qui est expliquée dans la section 3.5.1 ci-dessus. Dans un premier temps, nous commençons par tracer le graphe de dispersion des paires (u_i,v_i) dérivé des équations (3.2) et (3.3), pour tout $i \in \{1, ..., n\}$ avec les rangs dérivés de l'ensemble de données d'apprentissage (x_i ,y_i), et en comparant le dépendogramme de la Copule empirique C_1 avec les dépendogrammes de la famille de Copules cités dans 3.3, pour trouver la famille la plus proche de la Copule théorique C_2 qui ajuste au mieux la Copule empirique C_1. Par la suite, pour estimer les paramètres de la Copule théorique choisi, nous calibrons la Copule théorique avec la Copule empirique, en calculant la différence de surface entre la Copule empirique et la Copule théorique, générée à chaque fois avec un nouveau paramètre. Enfin, nous calibrons la Copule théorique C_2 avec le paramètre qui minimise cette différence de surface.

- *Étape 4*: Une fois que la famille de la Copule théorique choisie et son paramètre sont connus, nous générons un échantillon uniformément distribué $Z = \{U, V\}$ avec $U = \{u_1, ..., u_n\}$ and $V = \{v_1, ..., v_n\}$, c-à-d., un ensemble de N échantillons de la Copule théorique C_2 ayant la même taille que la Copule empirique C_1, en utilisant la formule théorique qui correspond à C_2.

- *Étape 5*: Dans cette étape, nous calculons l'enveloppe convex H_2 de Z en utilisant l'algorithme de Eddy [122] pour le calcul de l'enveloppe convexe d'un ensemble planaire de points.

  On obtient ainsi, tous les points qui sont des sommets de l'enveloppe convexe, c-à-d., les points H_2=(u_j,v_j)$\in Z$ pour tout $j \in \{1, ..., k\}$, où k est le nombre de sommets de l'enveloppe convexe de Z.

- *Étape 6*: **Déterminer l'enveloppe convexe correspondante H dans D.**

  Après avoir calculé le sous-ensemble des points H_2 qui sont les sommets de l'enveloppe convexe de l'échantillon Z, généré par la Copule théorique C_2, pour chaque point (u_j,v_j)$\in H_2$, nous calculons les points correspondants (x_i,y_i)$\in D$ en utilisant la transformée d'échantillonnage inverse [123].

  Alors, pour chaque valeur échantillonnée (u_j,v_j) $\in H_2$, nous calculons la valeur correspondante (x_i,y_i) dans l'espace D, qui est donnée par $x_i = F^{-1}(u_j)$ et $y_i = F^{-1}(v_j)$.

  En conséquence, nous obtenons k points de D qui sont les sommets d'un polygone H.

- *Étape 7*: Enfin, le polygone H obtenu à partir de l'application de la théorie des Copules et le calcul de l'enveloppe convexe, sur les lectures de données historiques D, sera téléversé dans les nœuds capteurs pour être utilisé dans la détection en ligne comme une signature des bonnes valeurs du nœud capteur.

4.2 Processus de détection en ligne des anomalies

Nous illustrons le processus de détection en ligne, par l'organigramme représenté sur la Figure 3.5.

Dans cette étape, chaque nœud évalue ses données collectées par rapport au modèle construit dans la dernière étape. Initialement, le nœud vérifie si chaque nouvelle mesure collectée serait à l'intérieur ou à l'extérieur du polygone H en utilisant l'algorithme de [124]. Dans le cas où elle serait à l'intérieur du polygone H, il s'agit d'une mesure correcte, elle sera traitée, stockée ou transmise. Sinon, elle est considérée comme une donnée incorrecte (anomalie), dans ce cas le compteur i sera incrémenté et cette mesure ne sera pas traité ou envoyé. Si la valeur de i est inférieure à un seuil prédéfini, le nœud passe à une autre instance de données, autrement, il y'a une situation de défaut,

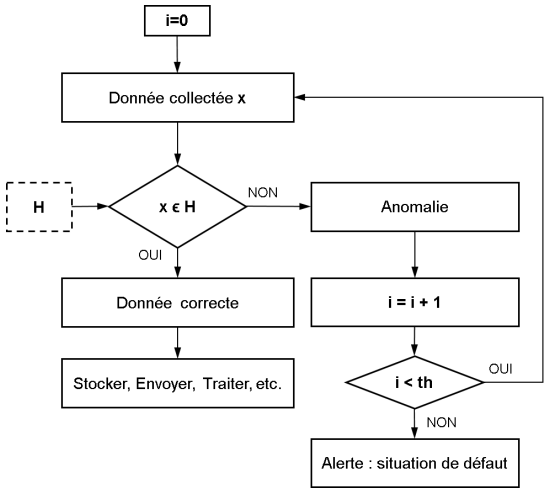


Figure 3.5: Détection en ligne des anomalies

et une alerte sera envoyée à la station de base. Dans notre cas, nous considérons que les mesures appartenant aux arêtes du polygone H sont considérées comme des données correctes.

Qu'est-ce qu'une situation de défaut?

Il existe de nombreuses possibilités pour des situations de défaut, par exemple lorsqu'une panne se produit en raison du mauvais fonctionnement d'un composant du nœud capteur ou de la présence de bruits. En outre, les capteurs nécessitent un calibrage à la suite d'une utilisation constante. De plus, il existe d'autres raisons, comme les évènements qui se produisent spontanément, par exemple, si un incendie se produit dans une zone donnée dans laquelle des nœuds capteurs sont déployés. En appliquant notre approche, chaque nœud qui appartient à la zone de l'incendie va détecter une situation de défaut, et envoyer une alerte au nœud puits, qui en ce moment observe que de nombreux nœuds qui appartiennent à la même zone géographique envoient la même alerte. De ce fait, le puits conclut qu'un évènement s'est produit, en se basant sur la situation de défaut signalée par tous ces nœuds, et la corrélation spatiale entre les nœuds capteurs qui appartiennent à la même zone.

5 Détection des capteurs usés

Dans cette section, nous présenterons la méthode proposée pour la détection des capteurs usés dans un RCSF en se basant sur la théorie des Copules .

Tout d'abord, nous construisons un modèle en hors ligne par l'application de la théorie des Copules sur des mesures historiques d'un nœud capteur, et c'est le même travail qui a été fait dans la Section 4.1. Ensuite, ce modèle sera téléversé dans les nœuds capteurs pour être utilisé différemment dans le processus de détection des capteurs usés en ligne avec un seuil prédéfini. Par conséquent, nous présentons uniquement la partie de détection des capteurs usés en ligne car on a déjà expliqué ci-dessous Section 4.1, comment construire le modèle hors ligne.

5.1 Détection des capteurs usés en ligne

Figure 3.6 illustre l'organigramme du processus de détection en ligne des capteurs usés.

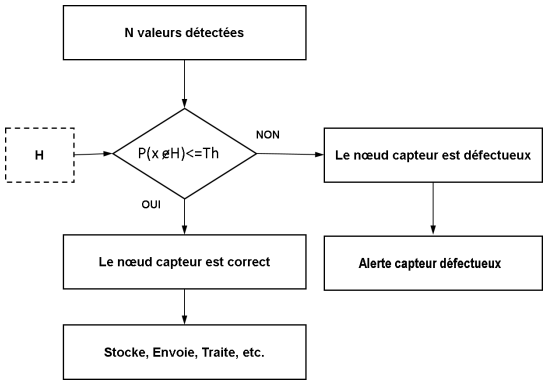


Figure 3.6: La détection en ligne d'un noeud capteur défectueux.

Dans cette étape, chaque nœud capteur évalue ses données détectées par rapport au modèle construit dans la dernière étape et un seuil prédéfini Th. Initialement, après avoir reçu n valeurs détectées pendant une période de temps T, qui varie selon le type d'application, le nœud calcule la probabilité qu'une valeur aléatoire détectée soit en dehors du polygone H à la fin de cette période. Si la probabilité calculée est inférieure à un seuil prédéfini Th, le nœud capteur est considéré comme bon sinon le nœud capteur est déclaré défectueux et une alerte sera envoyée à la station de base.

Comment calculer le seuil Th? Le seuil Th est le rapport entre le nombre de valeurs détectées qui se situent en dehors du polygone H, et le nombre total de valeurs détectées de l'ensemble de données de départ utilisé pour construire le modèle.

Comment calculer la probabilité qu'un point tombe en dehors du polygone H? Nous commençons par donner un exemple pour montrer comment cette probabilité peut être calculée. La Figure suivante 3.7 nous aidera à comprendre la méthode de calcul de la probabilité qu'un point aléatoire A se trouve à l'intérieur du polygone H. Dans notre cas, la distribution de données est obtenue à partir de l'application de la théorie des Copules.

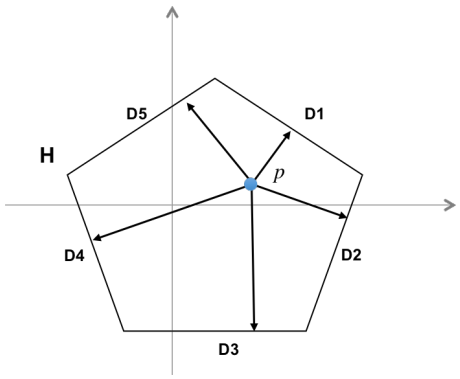


Figure 3.7: La probabilité qu'un point se trouve dans le polygone H

Ensuite, nous désignons par $F(x, y)$ la fonction de répartition (CDF) des variables aléatoires X et Y et par D_i une ligne droite qui appartient au polygone H, pour $i \in \{1, 2, 3, 4, 5\}$, où l'équation de D_i est donnée par : $y = \alpha_i x + \beta_i$.

Comme illustré dans la Figure 3.7, nous calculons la probabilité qu'un point aléatoire p se trouve à l'intérieur du polygone H. Cette probabilité est donnée par :

$$P\left(p \in H\right) = P(C_1 \Lambda C_2 \Lambda C_3 \Lambda C_4 \Lambda C_5)$$

où C_i est l'événement que le point p est dans le côté où le polygone est situé par rapport au segment D_i du polygone H.

Mathématiquement, calculer la probabilité $P\left(A \in H\right)$ est très efficace, mais dans notre cas de RCSF et selon le type d'application, le nombre de segments qui forment le polygone peut-être très large, ce qui entraîne une complexité de calcul trop élevée. Par conséquent, cette méthode n'est pas adaptée aux RCSFs.

Pour résoudre ce problème, nous proposons de calculer la probabilité $P\left(A \in H\right)$ en utilisant la simulation de Monte Carlo. Nous savons que la probabilité d'une variable aléatoire X est définie par: $P(X \leq x) = \int_H f(x)dx$ où $f(x)$ est la fonction de densité de probabilité de la variable aléatoire X.

Dans notre cas, cette probabilité est estimée selon la méthode de Monte Carlo et sa formule est donnée par :

$$\tilde{P} = \frac{1}{N} \sum_{1}^{N} \mathbb{1}_{x_i \in H} \tag{3.12}$$

La fonction $\mathbb{1}_{arg}$ est la fonction indicatrice, qui est égale à 1 si arg est vrai et égale à 0 sinon.

Enfin, nous avons :

$$\tilde{P}(x \notin H) = 1 - \tilde{P}$$

Donc, si la probabilité calculée est inférieure à un Th donné, le capteur est considéré comme bon, sinon le capteur est déclaré défectueux.

6 Évaluation de performances

Dans cette section, nous allons voir comment construire le polygone H hors ligne, en utilisant un jeu de données réel. Ce polygone sera utilisé après en ligne pour la détection des anomalies et les capteurs usés.

6.1 Résultats et discussion

Pour démontrer l'efficacité de notre approche, nous avons utilisé l'ensemble de données d'Intel lab [125], qui contient des mesures réelles collectées une fois tous les 31 secondes à partir de 54 nœuds capteurs, déployées dans le laboratoire d'Intel Berkeley Research entre 28 février et le 5 avril 2004. La Figure 3.8 montre le déploiement des nœuds capteurs dans ce laboratoire. Nous sélectionnons de manière aléatoire le nœud numéro 16 pour tester notre approche dans la même période citée plus haut. Nous prenons toutes les paires (température, humidité) du nœud 16, on obtient 21249 instances. Soit $D = (X \times Y)$ l'ensemble de données avec $X = temperature$ and $Y = humidity$.

Dans ce qui suit, nous montrons les résultats obtenus en utilisant le logiciel de statistique R (free-ware R) [28].

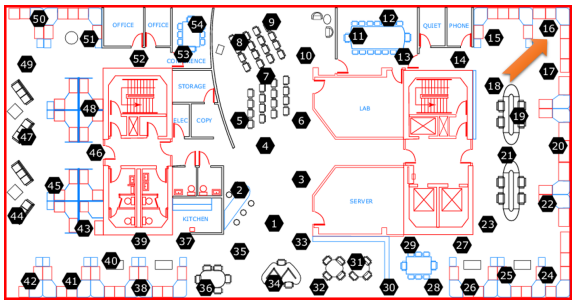


Figure 3.8: Déploiement des nœuds dans le laboratoire d'Intel Berkeley Research

Pour modéliser la dépendance entre la température X et l'humidité Y, nous commençons par dessiner le graphe de dispersion des paires (x, y), comme illustré dans la Figure 3.9. En analysant ce graphe de dispersion, nous ne pouvons rien dire sur la dépendance entre la température et l'humidité, et il n'y a pas non plus d'indication de la présence d'anomalies dans cet ensemble de données D.

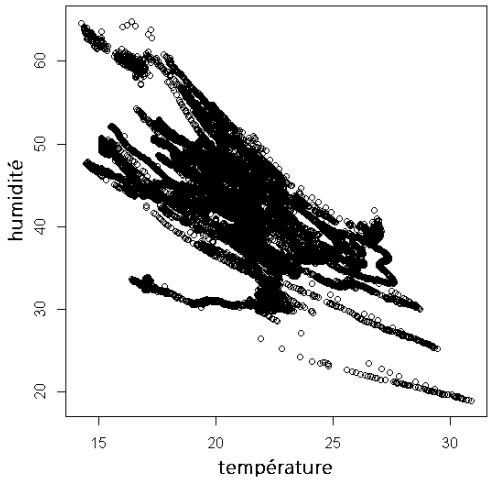


Figure 3.9: Le graphe de dispersion de la température et de l'humidité du noeud 16

Nous commençons par construire le modèle qui sera utilisé par les nœuds capteurs dans le processus de détection en ligne, en suivant les étapes mentionnées dans la section 4.1.

- *Étape 1*: L'ensemble de données sélectionné $D = (X \times Y)$ contient $N = 21249$ instances de paires (température, humidité) à partir du nœud 16.
- *Étape 2*: Dans cette étape, nous calculons la Copule bivariée empirique C_1 de D en utilisant la formule (3.4). Et en faisant un graphe de dispersion des paires (u_i, v_i) correspondant à l'échantillon (x_i, y_i) dans l'ensemble de données D, nous obtenons la Figure 3.10. Dans cette Figure, il est clair que les données, encerclées en rouge (zone A), devront être supprimées

de l'ensemble de données car elles ne suivent pas le comportement global de la majorité des données collectées. Et nous considérons que toutes les données qui appartiennent à cette zone sont incorrectes. En supprimant la zone A de la Figure 3.10, nous obtenons la Figure 3.11 qui représente le graphe de dispersion de 20610 paires de la Copule empirique C_1.

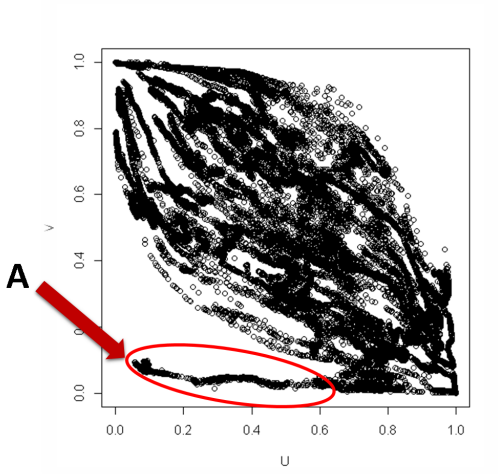


Figure 3.10: Le graphe de dispersion des paires (u_i,v_i)

- *Étape 3*: Dans cette étape, nous devons trouver la Copule C_2 qui ajuste au mieux la Copule empirique C_1. Par la méthode d'adéquation graphique, nous concluons que la Copule qui ajuste au mieux la Copule empirique C_1, est la Copule Gaussienne, obtenue en comparant les dépendogrammes de la Copule empirique de la Figure 3.11, avec les dépendogrammes de la Copule Gaussienne théorique illustrés dans les Figures 3.2 et 3.3. Maintenant, nous savons que la Copule théorique C_2 qui ajuste au mieux la Copule empirique C_1 est la Copule Gaussienne, après cela, nous devons estimer le paramètre qui calibre la Copule théorique C_2 avec la Copule empirique C_1, qui est le coefficient de corrélation ρ dans notre cas.

  Afin d'estimer le coefficient de corrélation ρ de la Copule gaussienne C_2, pour chaque valeur de ρ comprise entre -0.80 et -0.99, nous calculons la différence de surface entre la Copule empirique C_1 et la moyenne de surface de 100 Copules théoriques générées à partir de la Copule C_2, avec un échantillon de la même taille que C_1. Enfin, nous choisissons la valeur de ρ qui minimise cette différence de surface.

  Comme le montre la Figure 3.12, la valeur de ρ qui minimise la différence de surface est $\rho = -0.91$, donc la Copule qui ajuste au mieux la Copule empirique C_1 est la Copule Gaussienne C_2 avec la valeur du coefficient de corrélation $\rho = -0.91$.

- *Étape 4*: Nous générons un échantillon Z de 20610 paires (u_i, v_i) de la Copule Gaussienne C_2, avec un coefficient de corrélation $\rho = -0.91$, comme illustré dans la Figure 3.13.

- *Étape 5*: Dans cette étape, nous calculons l'enveloppe convexe H_2 de l'échantillon généré Z, on obtient $K = 28$ points des paires (u_i, v_i), qui représentent les sommets de l'enveloppe convexe calculée. En regardant la Figure 3.14, les points rouges sélectionnés représentent les sommets de l'enveloppe convexe de l'échantillon Z, et en joignant ces points, on obtient l'enveloppe convexe de l'échantillon Z qui est tracé en bleu juste pour l'illustration.

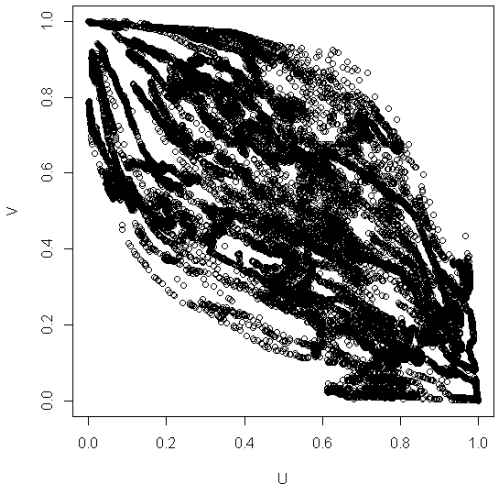


Figure 3.11: Le graphe de dispersion de la Copule empirique finale

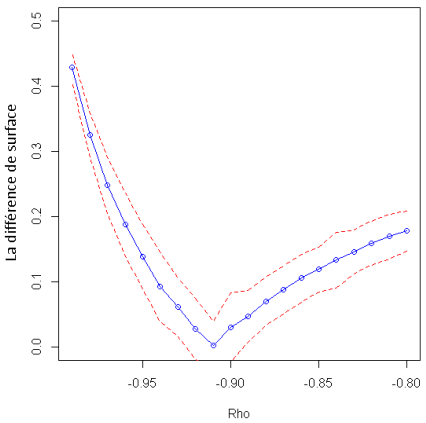


Figure 3.12: Le graphe de dispersion des valeurs de différence de surface selon différents valeurs de ρ.

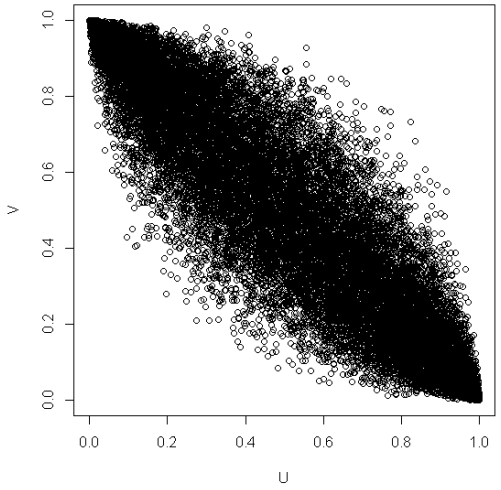


Figure 3.13: Le graphe de dispersion de l'échantillon généré Z.

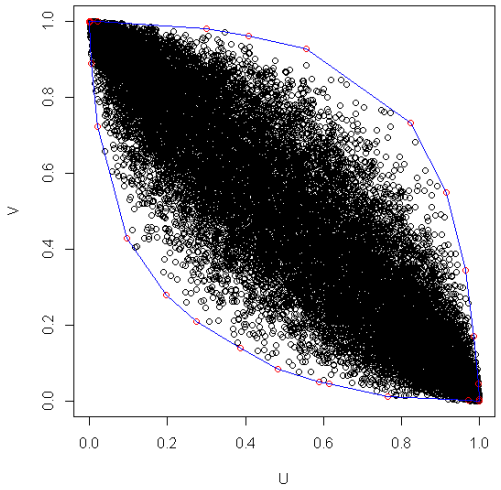


Figure 3.14: L'enveloppe convexe de l'échantillon généré Z.

- *Étape 6*: Maintenant, nous avons tous les points qui sont des sommets de l'enveloppe convexe H_2 dans l'espace Copule.

  Ensuite, pour chaque point de H_2 de l'espace Copule, nous calculons le point correspondant dans le l'ensemble de données de départ D.

  Par conséquent, on obtient le polygone H dans l'espace de départ D, comme illustré dans la Figure 3.15.

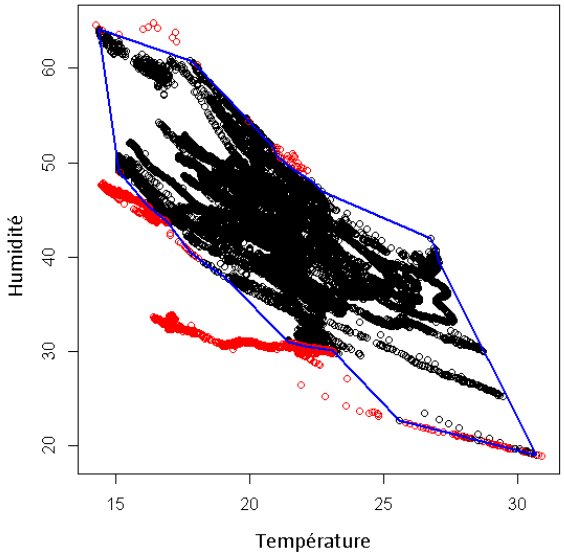


Figure 3.15: Le polygone H.

- *Étape 7*: Enfin, le polygone H sera téléversé dans les nœuds capteurs pour être utilisé dans la détection des anomalies en ligne , comme une signature des bonnes valeurs des nœuds capteurs.

Une fois le polygone H téléversé dans les nœuds capteurs, il sera utilisé d'une part, pour évaluer chaque instance de données entrantes par ces nœuds capteurs comme montré dans la Section 4.2, et d'autre part, pour la détection des capteurs usés en ligne comme montrés dans la Section 5.1.

Dans ce qui suit, nous allons juste vous montrer quelques résultats obtenus pour la détection des anomalies en ligne en utilisant le modèle qui a été construit. Cependant, nous avons laissé la détection des capteurs usés pour des travaux futurs.

6.2 Métriques de performance

Pour évaluer le modèle proposé, nous utiliserons le taux de détection (DR) et le taux de fausses alarmes (FAR) également connu sous le nom de taux de faux positifs (FPR) en tant que paramètres d'évaluation. Le taux de détection est le rapport entre les données anormales correctement détectées et le nombre total de données anormales. Et le taux de fausses alarmes est le rapport entre le nombre

Tableau 3.1: Matrice de confusion

Matrice de confusion		Valeurs prédites	
		Valeur normale	Valeur anormale
valeurs actuelles	Valeur normale	Vrai Négatif (TN)	Faux Positif (FP)
	Valeur anormale	Faux Négatif (FN)	vrai Positifs (TP)

Tableau 3.2:

	Valeur normale	Valeur anormale (anomalies)	DR (%)	ACC (%)	FPR (%)
Dataset 01	3000	300	100	100	0
Dataset 02	7000	700	100	100	0
Dataset 03	11000	1000	100	100	0

de données normales qui sont incorrectement détectées comme des données anormales et le nombre total de données normales. Dans notre cas, nous allons sélectionner les couples donnés par la température et l'humidité de l'ensemble de données D qui correspondent à la zone A de la Figure 3.11 comme des lectures anormales, et nous calculerons DR et FAR en fonction de ces données.

À partir de la table 3.1, les équations suivantes peuvent être définies pour calculer le taux de détection et le taux de fausses alarmes, ainsi que la précision (ACC) de la détection.

$$DR = \frac{TP}{TP + FN} \tag{3.13}$$

$$FAR = \frac{FP}{TN + FP} \tag{3.14}$$

$$ACC = \frac{TP + TN}{NombreTotaldeDonnees} \tag{3.15}$$

où:

TP : Lectures anormales détectées comme anomalies.

FP : Lectures normales détectées comme anomalies.

TN : Lectures normales qui ne sont pas détectées comme des anomalies.

FN : Lectures anormales qui ne sont pas détectées comme des anomalies.

Maintenant, nous présentons les résultats obtenus à partir de la détection en ligne.

Nous avons pris trois ensembles de données (Dataset 01, Dataset 02, Dataset 03) à partir du jeu de données original D, où chaque ensemble de données contient des valeurs normales et anormales. Les anomalies sont insérées aléatoirement dans tous les jeux de données sélectionnées, à partir d'un ensemble de paires de données correspondantes de D dans la zone A .

Pour les tests, nous avons fixé le seuil à **1**, ce seuil peut varier en fonction de l'application à implémenter sur les nœuds capteurs. Par exemple, dans le cas d'une détection de nœuds défaillants, le seuil doit avoir une grande valeur pour être sûr que le nœud concerné est vraiment un nœud défaillant.

En appliquant le modèle proposé de détection des anomalies en ligne aux différents ensembles de données, et en calculant les différentes métriques citées précédemment en utilisant les équations 3.13, 3.14, 3.15, nous obtenons les résultats illustrés dans le Tableau 3.2.

Cela montre que notre modèle atteint un taux de détection élevé de 100 % DR et 0% du taux de fausse alarmes FAR, et la précision de détection est de 100 % pour tous les jeux de données. Par

conséquent, l'approche proposée surpasse clairement d'autres travaux connexes dans tous les cas par un taux DR élevé et un FAR nul. En outre, le modèle proposé est très efficace pour maintenir les ressources limitées du réseau, et éliminer toutes les lectures vérifiées comme anormales par les nœuds capteurs, en empêchant celle-ci d'être échangées entre les nœuds, ce qui réduit considérablement la consommation d'énergie et la surcharge de communication du réseau(Overhead). Par conséquent, la durée de vie du réseaux augmente.

7 Conclusion

Dans ce chapitre, nous avons proposé une approche efficace pour la détection, en ligne des anomalies et les capteurs usés dans les RCSFs. Nous commençons par construire le modèle qui sera utilisé pour la détection en ligne en se basant sur la théorie des Copules pour les deux approches, et après cela, nous téléversons le modèle construit dans les nœuds capteurs. Cela, afin qu'il soit utilisé par la suite, dans la détection des anomalies en ligne, ou la détection des capteurs usés. Les résultats expérimentaux avec de vrais ensembles de données montrent que l'efficacité de notre approche proposée pour la détection des anomalies en ligne est très élevée en atteignant 100 % de DR et ACC et 0% de FAR. En outre, notre approche est très efficace en termes d'utilisation de ressources limitées du réseau, telles que la mémoire et la consommation d'énergie. En fait, les nœuds capteurs stockent uniquement un polygone pour la détection en ligne, ce qui maintient toujours la capacité de stockage mémoire à un niveau haut, et seules les bonnes valeurs seront transmises au sein du réseau. Cela réduit la surcharge de communication et maintient la consommation d'énergie basse. Par conséquent, la durée de vie du réseau augmente. Comme travaux futurs, nous envisageons d'implémenter la seconde approche proposée pour un scénario du monde réel, et de montrer l'efficacité de la théorie des Copules pour traiter les données multivariées dans un RCSF.

4

Intégrité et authenticité des données dans un RCSF

1 Introduction

Les réseaux de capteurs sans fil (RCSFs) sont des systèmes embarqués où chaque unité est équipée d'une certaine quantité de ressources de calcul, de communication, de stockage et de captage [126].

Grâce à leur capacité d'auto-organisation, ils permettent de déployer des applications à grande échelle. Jusqu'à présent, leur déploiement est toujours complexe dans de nombreux domaines tels que l'environnement, la domotique, la médecine et l'armée. Cependant, en particulier pour les applications militaires ou médicales, ils ont besoin de solutions sûres et fiables, ce qui semble important pour combler cette lacune cruciale. Il est prouvé que la sécurité dans ce type de réseaux est d'une importance stratégique, voire vitale, car leur bon fonctionnement implique des vies humaines. En outre, étant donné le fait que les nœuds capteurs ont des ressources limitées, les algorithmes de sécurité traditionnels ne conviennent pas bien aux RCSFs [20] (le processus de cryptage et décryptage doit être effectué au niveau de chaque nœud, ce qui génère des coûts de calcul très élevés). De plus, les attaques malveillantes comme la modification des données, la suppression et l'insertion des données, peuvent affecter la qualité des données collectées par les nœuds capteurs. Par conséquent, la protection de l'intégrité et de l'authenticité de ces données est un processus nécessaire pour en garantir la qualité avant de les utiliser pour prendre des décisions. Certains travaux basés sur des techniques de tatouage ont été proposés pour résoudre certains de ces problèmes, tels que l'altération, l'authentification et l'intégrité des données, la détection et le droit d'auteur [127], etc.

Le tatouage numérique est un mécanisme intéressant pour assurer l'intégrité et l'authenticité des données transmises dans les RCSFs. C'est l'art de cacher une donnée (signature, marque, donnée) dans une donnée d'hôte (donnée originale) de manière sécurisée, où seul l'utilisateur autorisé peut extraire et utiliser cette donnée [128]. Il est utilisé pour la protection du droit d'auteur, la preuve de propriété, la surveillance de l'utilisation illégale des données, l'authenticité et d'autres problèmes de sécurité [129]. La technique de tatouage peut être classée selon la méthode d'insértion de la signature dans une donnée, en spatiale ou fréquentielle. La technique spatiale intègre directement la marque aux données. C'est une technique facile et rapide, mais faible en ce qui concerne certaines attaques, en particulier les attaques géométriques [130]. La technique fréquentielle intègre la marque dans les coefficients des données. Elle est robuste mais plus complexe que la technique spatiale [131].

En se basant sur la robustesse de l'approche de tatouage, nous pouvons classer la technique

en robuste, semi-fragile et fragile. Le tatouage robuste doit résister aux attaques géométriques et non géométriques. Ce type d'approche est utile pour les problèmes de droit d'auteur et de propriété [132]. Le tatouage semi-fragile peut accepter de légères modifications de la donnée de la part de l'utilisateur autorisé, tandis que le tatouage fragile détruira la signature en cas de la moindre modification de la donnée. Ces techniques de tatouage visent à prouver l'intégrité et l'authenticité des données [133].

Le tatouage numérique comprend généralement deux étapes qui sont le marquage et le démarquage, le processus de marquage consiste à insérer la marque dans la donnée d'origine, tandis que, le processus de démarquage extrait la marque de la donnée tatouée.

En plus, selon le processus de démarquage en peut classer la technique de tatouage en aveugle, semi-aveugle et non aveugle. La technique aveugle est le type de système de tatouage le plus difficile car il ne nécessite ni la donnée d'origine, ni la marque pendant le processus de démarquage. Pour la technique de tatouage semi-aveugle elle ne nécessite pas la donnée originale pendant le processus de démarquage de la marque. Ensuite, pour la technique non aveugle elle nécessite au moins la donnée d'origine pour la détection de la marque.

Dans ce chapitre, nous proposons une nouvelle approche distribuée basée sur une technique de tatouage **semi-aveugle** qui fait partie du domaine spatial. Dans l'étape de collecte des données, chaque nœud du réseau, intègre dynamiquement dans chaque paquet de données, la même signature fixée localement avant de le transmettre à ses voisins. La vérification de la signature est effectuée par les nœuds récepteurs. Lorsque le paquet de données est reçu, il est utilisé pour calculer une nouvelle marque (signature) puis comparé à la marque fixée localement. Si leur valeurs ne sont pas identiques, le paquet de données reçu est rejeté.

Ce chapitre est organisé comme suit. Dans la Section suivante, nous présentons quelque travaux de tatouage dans les RCSFs. La Section 3 présente la méthode proposée. Ensuite, dans la Section 4, la technique de tatouage proposée est illustrée. Dans la Section 5, un aperçu du simulateur CupCarbon est donné. Les résultats de simulation et d'analyse de performance par rapport à d'autres méthodes sont présentés dans la Section 6. Enfin, la Section 7 conclut le chapitre.

2 Étude bibliographique sur le tatouage numérique dans les RCSFs

Dans cette section, nous allons discuter de certains travaux existants traitant l'authenticité et l'intégrité des données dans les RCSFs en utilisant des techniques de tatouage numérique.

La technique présentée dans [134] est basée sur une méthode de tatouage fragile pour assurer l'intégrité des données dans un RCSF. Les données collectées à partir de chaque nœud capteur sont encapsulées dans de nouveaux paquets de données. chaque paquet de données contient divers champs de données et un espace redondant des octets de données, sachant que, aucune intrusion dans les données d'origine n'est effectuée. Avant d'envoyer un paquet sur le réseau, chaque nœud capteur utilise une fonction de hachage unidirectionnelle pour créer la signature à partir des données collectées, ensuite associe cette signature aux paquets de données en l'intégrant dans l'espace redondant des octets ciblés. Du côté de la station de base, un algorithme de tatouage est conçu pour extraire la signature, qui est comparée à une signature recalculée afin de vérifier l'intégrité des données transmise sur le réseau.

Dans [135], les auteurs ont proposé une technique de tatouage numérique fragile baptisé (FWC-D), pour détecter les altérations non autorisées dans les flux de données échangées sur les RCSFs. FWC-D organise les données collectées par les capteurs en groupes de données de tailles constantes, et génère ensuite un numéro de série SN qui sera attaché et mémorisé avec chaque groupe, ce qui aidera le récepteur à déterminer le nombre d'insertions ou de suppressions de groupes en cas d'attaques d'insertion ou de suppression de groupes. Après il concatène chaque groupe avec l'autre par le stockage de la signature d'un groupe dans le groupe suivant, afin de rendre plus difficile pour l'attaquant d'insérer ou de supprimer un groupe complet sans que cela soit détecté. La signature est calculée par une fonction de hachage, qui est appliquée à la concaténation de tous les éléments de données individuels d'un groupe avec une clé secrète connue seulement de l'émetteur et des récepteurs, et le groupe de numéro de série SN. La fonction de hachage peut être MD5 ou SHA. En utilisant la clé secrète, le nœud récepteur peut extraire la signature (calculée côté transmetteur) des données reçues. Pour vérifier l'intégrité du groupe reçu, le récepteur recalcule la signature et l'a compare avec la signature extraite. Si les deux signatures correspondent, le groupe est considéré comme authentique; Sinon, le groupe est déclaré non authentique.

Dans [136], les auteurs ont proposé une méthode de tatouage multiple, appelée Multi-Mark. Elle utilise deux types de signatures, une signature d'annotation et une signature fragile. Les informations personnelles de l'utilisateur sont chiffrées et intégrées dans les données comme une signature d'annotation, ensuite, la signature fragile est générée et intégrée dans le résultat obtenu du premier tatouage. Lorsque les données tatouées finales sont transmises via le RCSF, des erreurs peuvent se produire en raison de la mauvaise condition du réseau ou des attaques malveillantes. Ils peuvent être détectés au niveau de la station de base, où la détection de falsification des données est basée sur l'authentification de la signature de tatouage fragile. Si nécessaire, la signature d'annotation peut être extraite.

Dans [137], les auteurs ont proposé un schéma d'authentification appelé (RDE) basé sur un algorithme de tatouage fragile pour les RCSFs. Les nœuds capteurs utilisent une fonction de hachage unidirectionnelle pour générer les signatures en fonction des données adjacentes puis les intégrer dans ces données. Après réception des données, la station de base restaure les données d'origine et en vérifie la fiabilité. L'algorithme RDE peut vérifier les données des capteurs à travers les bits de la signature insérés, et restaurer complètement les données d'origine.

Dans [138], les auteurs ont proposé des algorithmes pour la génération d'identités, l'insertion et la détection. L'identité d'un nœud émetteur est générée en transformant une clé et les données collectées dans un laps de temps. Le résultat transformé forme la signature qui est inséré dans les données à envoyer. Le nœud récepteur juge l'authenticité des données en vérifiant cette signature. Une fois la signature détectée, les données seraient stockées et transmises. Sinon, les données seraient rejetées.

La plupart des solutions proposées sont basées sur des algorithmes centralisés dans lesquels les données tatouées sont envoyées entièrement pour être vérifiées par le Cluster-Head ou par la station de base. Malheureusement, les modèles centralisés génèrent une surcharge de communications très élevées en transmettant la totalité des données pour vérification. Comme mentionné précédemment, la majorité de l'énergie dont dispose un capteur est consommée pendant la transmission et la réception des données plutôt que par la tâche de traitement de données. Par conséquent, il est préférable de développer une solution distribuée pour assurer l'intégrité et l'authenticité des données

transmises dans le réseau, afin de minimiser la consommation d'énergie. Et cela permet aussi à chaque nœud capteur de vérifier localement l'intégrité et l'authenticité des données reçues par l'extraction de la signature. Ce qui permet d'obtenir rapidement l'authenticité ces données. Le nœud rejettera les données reçues dans le cas d'une signature non authentique, sinon, il accepte de recevoir les données. En outre, si l'intégrité et l'authentification des données est vérifiée par une solution centralisée, et que ces données sont falsifiées, tous les nœuds qui participent dans le routage de ces données vers la destination centralisée consommeront beaucoup d'énergie tout en transmettant des données falsifiées. Ceci est inapproprié en ce qui concerne les contraintes d'énergie dans les RCSFs.

3 La technique de tatouage

Afin d'assurer l'intégrité et l'authenticité des données dans les réseaux de capteurs sans fil, nous allons maintenant présenter la méthode proposée qui utilise une technique de tatouage semi-aveugle. Cette technique est pratique pour le domaine spatial, puisque la signature peut être insérée directement dans les données d'origine, afin de réduire la complexité en évitant plusieurs opérations supplémentaires et pour préserver l'énergie du nœud. La méthode est composée de deux phases: une phase de marquage et une phase de démarquage. Dans le réseau, chaque nœud peut agir comme émetteur ou récepteur.

3.1 Processus de marquage de la signature

Le schéma de marquage est illustré par la Figure 4.1. Le marquage de la signature se fait en utilisant l'interpolation linéaire (4.1) comme suit :

$$v' = (1 - \alpha) \cdot w + \alpha \cdot v \tag{4.1}$$

où v' représente la donnée tatouée, $\alpha \in]0, 1[$ détermine le degré de visibilité et la dégradation de la signature intégrée dans une donnée, w représente la signature d'origine, et v la donnée d'origine.

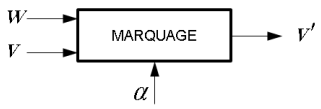


Figure 4.1: Schéma de marquage de la signature.

3.2 Processus de démarquage de la signature

L'approche de tatouage proposée est semi-aveugle, car la signature d'origine w et les données tatouées reçues v' sont nécessaires pendant la phase d'extraction pour calculer la signature extraite w' (après l'attaque). L'intégrité et l'authenticité des données sont basées sur cette signature. Le schéma d'extraction est illustré par la Figure 4.2. De plus, le processus d'extraction se fait en utilisant l'interpolation linéaire (4.2).

$$w' = (1/\alpha) \cdot w - ((1 - \alpha)/\alpha) \cdot v' \tag{4.2}$$

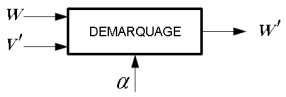


Figure 4.2: Schéma de démarquage de la signature.

Dans un RCSF, chaque nœud a la capacité de marquer ou de démarquer la signature en fonction de son rôle dans le réseau. Si un nœud est un nœud émetteur, alors ce nœud doit marquer la signature. Toutefois, si un nœud est un nœud récepteur, il doit démarquer la signature des données tatouées. La signature extraite sera comparée à la signature d'origine afin de prouver la fiabilité des données. Si la fiabilité est prouvée alors les données brutes peuvent être recalculées en utilisant l'inverse de l'equation (4.1). Pour atteindre une haute imperceptibilité et une faible dégradation, nous assignons à α la valeur 0.98 qui est proche de 1.

4 La méthode d'authentification

Dans cette section, nous proposons une méthode d'authentification pour l'intégrité et l'authenticité des données basée sur le tatouage numérique dans un RCSF. Tout d'abord, nous présenterons l'idée générale de l'approche. Ensuite, nous présenterons les algorithmes des processus de marquage et de démarquage.

4.1 Le modèle proposé

L'organigramme de la Figure 4.3 décrit le processus de la technique proposée et résume les phases exécutées par chaque nœud.

Tout nœud dans le réseau peut émettre ou recevoir des données. Lorsqu'un nœud reçoit des données, il extrait la signature w' et la compare à la signature originale w. Si ces valeurs sont identiques, le nœud conclut que les données sont authentiques et accepte de les recevoir pour le stockage, le traitement ou la transmission. Sinon, les données seront rejetées par le nœud. Si le nœud est un émetteur, il insère la signature dans les données avant de les envoyer.

4.2 L'algorithme de Marquage

Les étapes principales de l'algorithme de marquage de la signature, exécutées dans chaque nœud capteur S_i, sont décrites comme suit. Notez que l'algorithme proposé fonctionne avec des données numériques. Dans ce qui suit, nous supposons pour simplifier, que chaque nœud capteur S_i n'envoie qu'une seule donnée numérique v_i au nœud capteur S_j. Le nœud capteur S_i va alors:

- *Étape 1*: Préparer la donnée v_i pour l'envoi.
- *Étape 2*: Insèrer la signature w à la donnée v_i en utilisant l'équation (4.1). La donnée obtenue sera v_i'.
- *Étape 3*: Envoyer la donnée tatouée v_i' au nœud capteur de destination S_j.

Le pseudo-code du noeud émetteur est alors donné par l'Algorithme 6, où v est la valeur détectée obtenue par l'unité de captage du nœud capteur courant. Cette valeur est obtenue en utilisant la

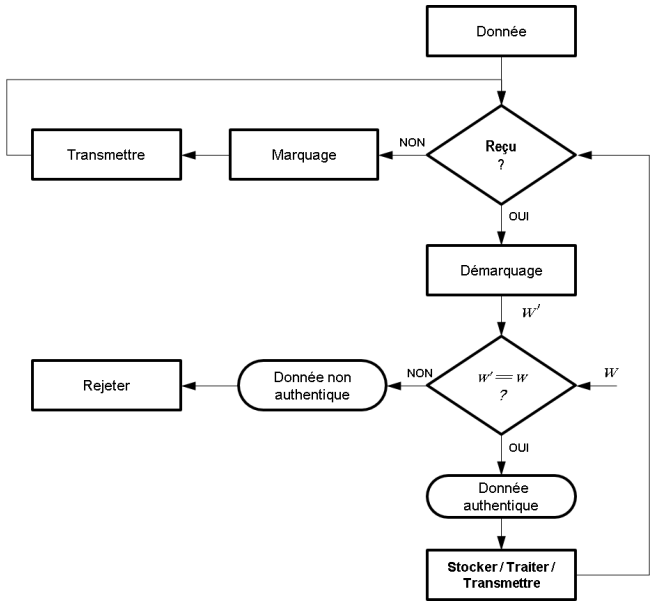


Figure 4.3: Diagramme de la méthode d'authentification proposée.

fonction `getSensedValue()`. La fonction `send(`v'`,` d`)` permet d'envoyer une donnée tatouée v' au nœud capteur ayant l'identifiant d.

Algorithm 6 Algorithme de marquage

Input: w, α

1: **repeat**
2: $v =$ getSensedValue()
3: $v' = (1 - \alpha) \cdot w + \alpha \cdot v$
4: send(v', d)
5: **until** false

4.3 L'algorithme de démarquage

Les principales étapes de l'algorithme d'extraction (démarquage), exécutées dans chaque nœud capteur S_i, sont décrites comme suit :

- *Étape 1*: Lire la donnée reçue v'_i du nœud émetteur S_j.
- *Étape 2*: Extraire la signature w'_i de la donnée reçue v'_i.
- *Étape 3*: Comparer la signature extraite w'_i avec la signature original w. S'ils sont égaux, alors la donnée est authentique. Sinon, le nœud capteur rejette cette donnée et déclare le nœud émetteur comme un *Nœud malveillant*.

Le pseudo-code d'un récepteur est donné par l'Algorithme 7. La fonction `read()` permet de lire les données reçues par le module radio. La fonction `store()` permet de stocker, de traiter ou

de transmettre les données reçues par le module radio. La fonction `reject()` permet de rejeter les données reçues par le module radio. Cela signifie qu'un nœud malveillant est détecté.

Algorithm 7 Algorithme de démarquage

Input: w, α

repeat
  v'=read()
  $w' = (1/\alpha) \cdot w - ((1-\alpha)/\alpha) \cdot v'$
  if (w'==w) **then**
    store(v')
  else
    reject(v')
  end if
until false

5 Le Simulateur CupCarbon

CupCarbon [44][139] est un simulateur libre source utilisé dans ce travail afin de valider la méthode proposée. Il s'agit d'un simulateur de réseaux de capteurs sans fil dédié aux villes intelligentes et d'Internet des objets (SCI-WSN). Son objectif est de concevoir, visualiser, déboguer et valider des algorithmes distribués de surveillance, de collecte de données environnementales, etc., et de créer des scénarios environnementaux tels que les incendies, le gaz, les mobiles, et généralement dans le cadre de projets éducatifs et scientifiques.

Les réseaux peuvent être conçus et prototypés par une interface ergonomique et facile à utiliser à l'aide de la structure OpenStreetMap (OSM) pour déployer des capteurs directement sur la carte. Il comprend, un script appelé SenScript [139] qui permet de programmer et de configurer individuellement chaque nœud capteur. La consommation d'énergie peut être calculée et affichée en fonction du temps simulé. Cela permet de clarifier la structure, la faisabilité et l'implémentation réaliste d'un réseau avant son déploiement réel.

6 Évaluation de performances

L'objectif de cette section est d'évaluer par simulation la validation de notre approche et son efficacité énergétique. Cela nous permet de vérifier si l'approche proposée est vraiment utile pour assurer l'authenticité et l'intégrité des données. Dans le cadre de ce travail, nous avons utilisé la plateforme CupCarbon qui permet de visualiser le processus de simulation, et offrir une interface facile à utiliser et déboguer. Dans ce qui suit, nous présentons les deux premiers modèles d'attaques qui sont utilisés dans ce travail, ainsi que la configuration de la simulation, suivie des résultats de la simulation, et nous présentons ensuite une comparaison de la méthode proposée avec deux approches existantes.

6.1 Scénarios d'attaque

Dans ce travail, nous nous focalisons sur deux types d'attaques qui sont:

1. Modification de données (Tampering Packet): Un noeud compromis modifie tout ou une partie des données qu'il est censé de transmettre [140].

2. Insertion de données erronées (Forgery Packet): Un adversaire peut compromettre les nœuds existants et injecter un faux message avec de fausses informations. Il est également possible que l'adversaire ajoute de nouveaux noeuds au réseau qui produisent de fausses données. Une telle attaque consomme également les ressources énergétiques des autres noeuds capteurs [135].

6.2 Configuration de la simulation

Les réseaux de capteurs sont générés manuellement dans un espace bidimensionnel. Les nœuds capteurs sont déployés dans une zone rectangulaire choisie, où la portée de communication de chaque nœud capteur est fixée à $100m$ (mètres). Les nœuds capteurs sont supposés être statiques pendant la simulation. Nous sélectionnons M nœuds émetteurs et K nœuds malveillants pour exécuter les deux attaques citées ci-dessus. Nous avons utilisé le modèle énergétique du nœud capteur TelosB. Sa consommation d'énergie est estimée à $59.2\mu J$ pour transmettre un octet, et à $28.6\mu J$ pour recevoir un octet [79]. Pour la batterie, nous avons utilisé le modèle Super Alkaline AALR6 qui est une source d'énergie portable avec une capacité de 9580 Joules.

6.3 Résultats et discussion

Pour mieux comprendre comment notre approche se comporte dans les conditions données dans la sous-section précédente, nous avons utilisé deux cas d'études. Dans le premier cas, comme illustré par la Figure 4.4, nous avons généré $N = 15$ nœuds capteurs dont $M = 1$ nœud transmetteur et $K = 1$ nœud malveillant. Dans le deuxième cas, comme illustré par la Figure 4.5, nous avons généré $N = 100$ nœuds capteurs incluant $M = 8$ nœuds transmetteurs et $K = 5$ nœuds malveillants.

Dans le premier cas d'étude, le nœud émetteur S_5 montré avec la couleur jaune est programmé pour envoyer des données en mode diffusion générale après le processus de marquage. Tous ses nœuds voisins S_1, S_2, S_4, S_6, S_7 et S_{10} recevront ces données. Ces nœuds vérifieront l'authenticité et l'intégrité des données. Un nœud malveillant S_{21} montré avec la couleur rouge est programmé pour créer soit des fausses données avec de fausses informations, soit pour modifier tout ou une partie des données reçues qu'il est censé de router. Dans ces deux cas, tous les nœuds voisins du nœud malveillant détecteront sa présence dans le réseau et rejetteront les données routées par celui-ci. Figure 4.4 illustre les situations où un nœud reçoit des données et après l'extraction de la signature. Un nœud sera montré d'une couleur verte lorsque les données reçues sont authentiques et avec une couleur orange lorsqu'un nœud malveillant est détecté.

Dans la Figure 4.5, les nœuds émetteurs sont montrés en jaune et les nœuds malveillants sont montrés en rouge. Comme nous pouvons le constater, tous les nœuds malveillants sont détectés par leur nœuds voisins, c'est-à-dire, ceux qui sont montrés de couleur orange. Les nœuds qui ont reçu des données authentiques des nœuds émetteurs sont indiqués en vert.

Selon les résultats de simulation, la Figure 4.6 montre que l'approche de tatouage proposée peut effectivement vérifier l'intégrité des données et assurer l'authenticité et la fiabilité lorsque le système réalise une détection de 100%. De plus, nous avons calculé le BER (Bit Error Rate) pour tous les cas et sa valeur est toujours zéro, ce qui signifie que la signature extraite est parfaitement égale à la signature d'origine. Par conséquent, notre approche garantit absolument l'intégrité et l'authenticité des données.

Pour étudier l'efficacité de la méthode proposée en termes de consommation d'énergie, nous avons calculé l'énergie consommée par les nœuds capteurs dans chacune des deux cas d'études considérées. Pour le premier cas d'étude, où nous avons un nœud transmetteur et un nœud malveillant, l'énergie consommée par chaque capteur est représentée par la Figure 4.7. La consommation

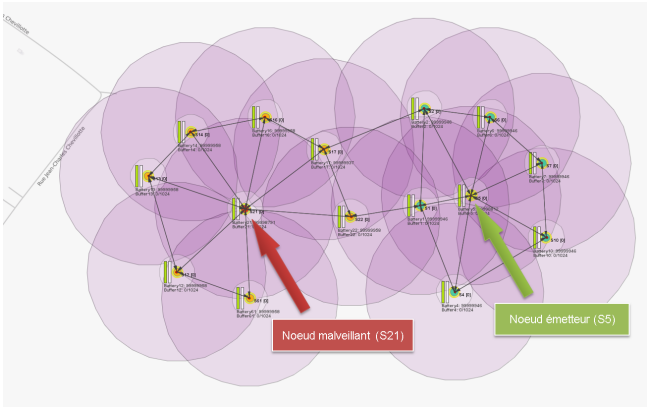


Figure 4.4: Cas d'étude 1: RCSF avec 15 noeuds.

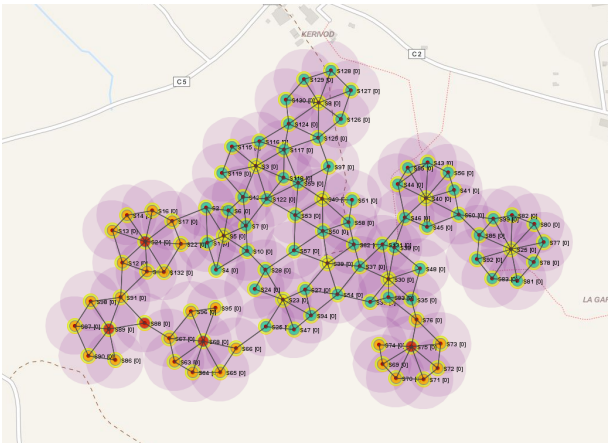


Figure 4.5: Cas d'étude 2: RCSF avec 100 noeuds.

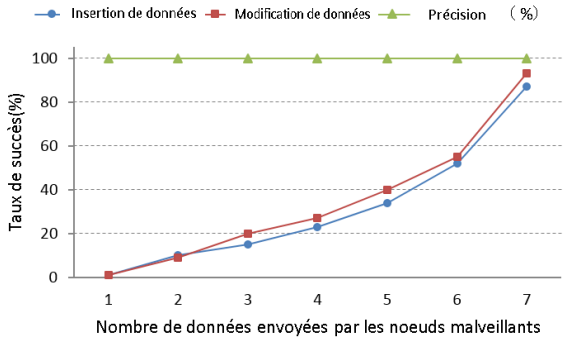


Figure 4.6: Précision de la signature extraite sous les deux attaques.

d'énergie de chaque nœud est la somme de ses consommations en transmission, de réception et de traitement. Nous remarquons que dans le premier cas d'étude, la consommation d'énergie moyenne est égale à $784.3\mu J$ qui représente $8.19 \cdot 10^{-06}\%$ de la capacité de la batterie initiale. En outre, nous pouvons également voir dans la Figure 4.7 que les nœuds capteurs $S5$ et $S21$ sont les plus consommateurs d'énergie. Ce qui est évident puisque $S5$ est un nœud émetteur et $S21$ est un nœud malveillant. Notez que ces résultats sont directement liés au modèle de batterie considéré.

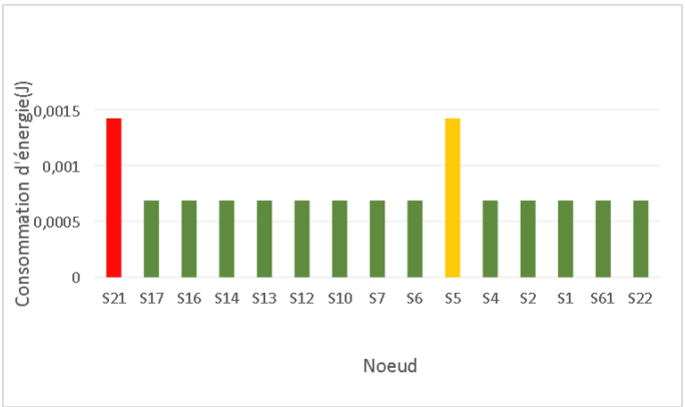


Figure 4.7: Consommation d'énergie par nœud.

Pour le deuxième cas d'étude, présentée par la Figure 4.5, où nous avons 9 nœuds émetteurs et 5 nœuds malveillants, le réseau consomme une énergie moyenne égale à $2673.2\mu J$, ce qui représente $2.79 \cdot 10^{-05}\%$ de la capacité de la batterie considérée.

Classiquement, lorsque le nombre de nœuds d'un RCSF augmente, le nombre de données échangées entre ces nœuds augmentera aussi. Par conséquent, la consommation d'énergie du réseau augmentera également, ce qui est évident. Cependant, dans notre cas, les nœuds récepteurs vérifieront l'intégrité et l'authenticité des données reçues. Si les données sont rejetées, elles ne seront

pas acheminées, ce qui entraînera une réduction du nombre de données échangées dans le réseau, ce qui réduira la consommation d'énergie du réseau. Par conséquent, l'approche proposée de tatouage est capable de minimiser la consommation d'énergie et le trafic du réseau considérablement en raison de l'auto-vérification effectuée par chaque nœud.

Il est à noter que, dans la méthode proposée, la consommation d'énergie liée au traitement des données, qui comprend les processus de marquage et de démarquage, est négligeable par rapport à la consommation d'énergie de transmission et de réception des données. Cela est dû à la simplicité de l'opération de traitement de données à effectuer.

6.4 Étude comparative

Pour comparer la méthode proposée avec celles de [129] et [134], nous avons calculé le nombre de bits de la signature utilisée dans le processus de marquage en fonction du nombre de données envoyées ,et nous les avons comparées avec celles des méthodes existantes. La Figure 4.8 montre les résultats obtenus. Dans notre cas, la taille de la signature est fixée à 4 bits, ce qui est très faible par rapport aux deux méthodes considérées. De plus, ces méthodes génèrent une nouvelle signature pour chaque donnée collectée et les fausses données sont détectées par la station de base. Cela signifie que s'il existe de nombreux nœuds malveillants dans le réseau essayant d'envoyer de fausses données, toutes ces données seront routées vers le puits ce qui augmente la consommation d'énergie du réseau de manière considérable en raison du très grand nombre de fausses données échangées dans le réseau. Dans le cas de la méthode proposée, chaque nœud est capable de vérifier par lui-même l'intégrité et l'authenticité des données reçues, et peut décider s'il rejette ou transmet les données reçues. Par conséquent, la consommation d'énergie dans ce cas sera considérablement réduite et sera très faible par rapport aux autres méthodes qui sont des approches centralisées.

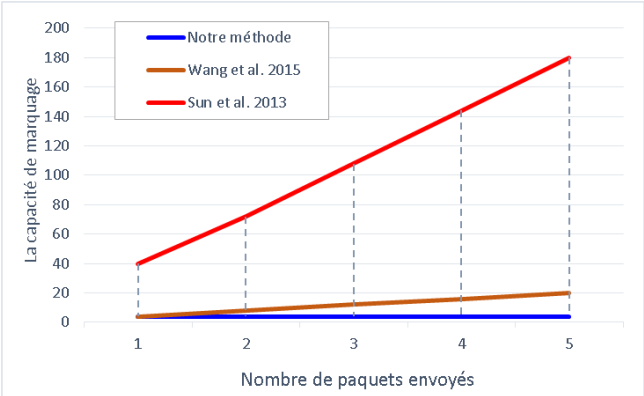


Figure 4.8: La taille de la signature (Watermark Payload).

En outre, la complexité de calcul de la méthode proposée en termes de temps d'exécution des processus de marquage et démarquage est très faible par rapport aux autres méthodes existantes. En fait, la taille de la signature sélectionnée est très faible et ne dépend pas de la taille des données collectées. En outre, la signature dans notre méthode n'est pas générée à chaque fois, mais fixée à l'avance et prête à être utilisée par tout nœud capteur pour le processus de marquage ou démarquage. En plus, la prise de décision concernant l'authenticité et l'intégrité des données reçues est effectuée instantanément par chaque nœud capteur tandis que les autres méthodes attendent la décision concernant l'intégrité des données à partir d'un emplacement centralisé. Par conséquent, la

technique de tatouage proposée est beaucoup plus rapide que les autres, ce qui la rend plus pratique pour les applications types réel des RCSFs.

7 Conclusion

Dans ce chapitre, nous avons proposé une méthode distribuée basée sur une technique de tatouage semi-aveugle. Chaque nœud du réseau vérifie l'authenticité et l'intégrité des données reçues. Pour étudier la performance de la méthode proposée, nous avons utilisé le simulateur CupCarbon. Nous avons envisagé deux types d'attaques: modification de données et falsification de données. Les résultats obtenus montrent que notre méthode est efficace en termes de consommation d'énergie. En outre, le nombre de données échangées est réduit de façon drastique en détectant localement dans chaque nœud capteur toutes les données non authentiques. Cela permet de réduire le trafic réseau et la consommation d'énergie de l'ensemble du réseau. L'approche distribuée par rapport à l'approche centralisée d'une part et, l'algorithme d'interpolation linéaire du tatouage d'autre part conduisent à une faible consommation d'énergie. Comme travaux futurs, nous prévoyons d'intégrer la méthode proposée sur un RCSF réel qui est déjà conçu pour exécuter une tâche pareille. Cela nous permet d'étudier la difficulté de cette intégration.

5
Conclusion générale

Les réseaux de capteurs sans fil (RCSFs) sont un instrument puissant qui a révolutionné et amélioré notre vie par diverses applications qui ne cessent de croitre chaque jour. Cependant, la réalisation d'une application à base de RCSF pose de nombreux défis liés aux enjeux de fiabilité et de sécurité surtout pour les applications critiques. Dans ce contexte, d'une manière générale, cette thèse a pour but de contribuer au développement de solutions permettant de garantir un certain niveau de fiabilité dans un RCSF dédié aux applications sensibles en tenant compte toujours des ressources limitées des nœuds capteurs. Nous avons proposé des solutions permettant de répondre aux objectifs cités ci-dessous.

1 Rappels des objectifs

Nous rappelons ici brièvement les objectifs de cette thèse qui s'articulent autour des trois points suivants :

- Le développement de méthodes permettant de détecter des nœuds capteurs défaillants dans un RCSF,
- Le développement de méthodes permettant de détecter les anomalies dans les mesures collectées par les nœuds capteurs, et par la suite, les capteurs usés (fournissant de fausses mesures),
- Le développement de méthodes permettant d'assurer l'intégrité et l'authenticité des données transmise dans un RCSF.

En considérant ces objectifs, le résumé des contributions de cette thèse est présenté dans la section suivante.

2 Contributions

Afin de détecter les nœuds défaillants dans les RCSFs et pour faciliter une prise de décision correcte et efficace à l'aide de données recueillies par les nœuds capteurs, ainsi, assurer l'intégrité et l'authenticité des données transmises dans le réseau, nous avons proposé les solutions suivantes :

Dans le premier chapitre, nous avons proposé un algorithme de détection de sommets frontières dans un graphe Euclidien connecté nommé LPCN [21]. L'algorithme choisit pour chaque sommet, le sommet d'angle polaire minimum par rapport au sommet trouvé dans l'itération précédente. Sa complexité est $O(kh^2)$, où k est le degré maximal du graphe et h le nombre de sommets de

l'enveloppe polygonale. Nous avons montré que l'exécution de l'algorithme en présence de structures de graphe spécifiques peut conduire à des solutions non valides et non optimales, et nous avons indiqué comment surmonter ces difficultés. De plus, nous avons prouvé la convergence de l'algorithme et l'optimalité de la solution. Enfin, nous avons présenté certaines applications qui peuvent être résolues en utilisant l'algorithme proposé.

Par la suite, dans le deuxième chapitre, nous avons proposé une approche distribuée pour la détection des nœuds défaillants situés dans une frontière de réseaux de capteurs sans fil [23]. Dans cette approche, nous avons utilisé la version distribuée de l'algorithme LPCN [22] pour déterminer les nœuds frontière du réseau. Ensuite, une fois que tous les nœuds frontière sont déterminés, chacun de ces nœuds envoie périodiquement un message à son prochain voisin, qui devrait lui répondre. Si une réponse n'est pas reçue, une situation d'échec sera déclenchée et une restructuration du réseau sera effectuée pour trouver une nouvelle frontière. Notre approche nécessite un nombre limité de calculs locaux simples, et elle a juste besoin des informations sur les voisins à un seul saut. Nous avons évalué la performance de notre approche dans diverses conditions et nous avons montré son efficacité énergétique. Dans cette approche, le nombre de messages échangés pour identifier les capteurs défaillants est faible et une quantité substantielle d'énergie peut être sauvegardée par les nœuds capteurs. De plus, l'approche proposée surpasse celle des précédentes en fournissant une haute précision de détection.
Dans le troisième chapitre, en se basant sur la théorie des copules, nous avons proposé deux nouvelles approches qui permettent la détection en ligne des anomalies et des capteurs usés dans les RCSFs [24, 25, 26]. Les deux approches proposées utilisent le même modèle construit sur la base des mesures historiques collectées par un nœud capteur. Ce modèle est représenté par un polygone qui sera intégré dans l'ensemble des nœuds du réseau pour être utilisé dans le processus de détection en ligne des deux approches.

En ce qui concerne la première approche, les anomalies sont détectées en comparant le flux de données d'entrée lié à la mesure et à la surveillance de phénomènes physiques avec ce polygone. Si les valeurs détectées n'appartiennent pas à ce polygone, elles sont considérées comme des anomalies. Les résultats expérimentaux obtenus sur des ensembles de données réels en utilisant le logiciel de statistique R [28], ont montré que l'efficacité de notre approche proposée pour la détection des anomalies en ligne est très élevée et atteignent 100 % de DR(taux de détection) et ACC (précision de la détection) et 0% de FAR (taux de fausses alarmes). En outre, notre approche est très efficace en termes d'utilisation de ressources du réseau, telles que la mémoire et la consommation d'énergie. En fait, les nœuds capteurs stockent uniquement un polygone pour la détection en ligne, ce qui maintient toujours la capacité de stockage mémoire à un haut niveau, et seules les bonnes valeurs seront transmises au sein du réseau. Cela réduit la surcharge de communication et maintient la consommation d'énergie basse. Par conséquent, la durée de vie du réseau augmente.

Pour la deuxième approche, l'objectif c'est de détecter les capteurs usés parce que ceux-la doivent être exclus et remplacés dans le réseau afin de garantir une bonne qualité de service, que les décisions prises soient corrects, ainsi, pour éviter que des données erronées (anomalies) ne soient échangées sur le réseau. Par conséquent, seuls les bonnes mesures seront échangées dans le réseau, ce qui réduit la charge de communication dans le réseau d'une part, et, évite le gaspillage de l'énergie des nœuds capteurs d'autre part. Nous avons montré toutes les étapes nécessaires pour construire le modèle qui sera utilisé dans la phase de détection. Ce modèle qui est représenté par un polygone sera utilisé en ligne comme signature avec un seuil prédéfini pour détecter efficacement les capteurs usés, en calculant la probabilité qu'une valeur détectée (collectée) pendant une période de temps T se trouve à l'extérieur de ce polygone. Si la probabilité calculée est inférieure à un seuil prédéfini, le nœud capteur est considéré comme bon, sinon le nœud capteur est déclaré mauvais et une alerte

sera envoyée à la station de base.

Dans le chapitre 4, nous avons présenté une nouvelle approche de tatouage numérique entièrement distribuée pour les RCSFs afin de répondre au troisième objectif. Cette approche vise à assurer l'intégrité et l'authenticité des données transmises dans un RCSF. Dans l'étape de collecte des données, chaque nœud du réseau intègre dynamiquement dans chaque paquet de données la même signature fixée localement avant de le transmettre à ses voisins. La vérification de la signature est effectuée par les nœuds récepteurs. Lorsque le paquet de données est reçu, il est utilisé pour calculer une nouvelle signature qui sera comparée avec la signature fixée localement. Si leurs valeurs ne sont pas identiques, le paquet de données reçu est rejeté. L'avantage de notre approche proposé est que chaque nœud du réseau peut vérifier l'authenticité et l'intégrité des données reçues au niveau local et non au niveau central (station de base). Cela permet d'avoir d'une part, une réponse rapide concernant l'authenticité et l'intégrité des données reçues, et, d'autre part, de sauvegarder l'énergie des nœuds en empêchant les données falsifiées ou modifiées d'être échangées sur le réseau. Pour étudier les performances de cette méthode, nous avons utilisé le simulateur CupCarbon. Nous avons envisagé deux types d'attaques: modification de données et falsification de données. Les résultats obtenus montrent que notre méthode est efficace en termes de consommation d'énergie. En outre, par rapport à des méthodes centralisées, le nombre de données échangées est réduit de façon drastique en détectant localement dans chaque nœud capteur toutes les données non authentiques. Cela permet de réduire le trafic réseau et la consommation d'énergie de l'ensemble du réseau. Nous avons comparé notre approche par rapport aux méthodes existantes. La complexité de calcul de la méthode proposée en termes de temps d'exécution des processus de marquage et de démarquage est faible parce que, la taille de la signature sélectionnée est très faible et ne dépend pas de la taille des données collectées. En outre, la signature dans notre méthode n'est pas générée à chaque fois, mais fixée à l'avance et prête à être utilisée par tout nœud capteur pour le processus de marquage ou de démarquage. Enfin, l'avantage de l'approche distribuée par rapport à l'approche centralisée d'une part, et, la technique de tatouage légère proposée d'autre part conduisent à une consommation d'énergie minimale dans le réseau et rendent cette technique très pratique pour les applications des RCSFs.

3 Perspectives

Toutes les approches proposées dans le cadre de cette thèse ont autant de perspectives à étudier dans de futurs travaux, et méritent d'être étudiées et implémentées réellement dans le cadre d'applications plus spécifiques, liées au domaine des réseaux de capteurs, parce que chaque type d'application a des contraintes et des spécificités qui imposerait l'adaptation des méthodes proposées aux nouvelles exigences, afin d'assurer toujours une bonne qualité de service et d'obtenir toujours des modèles performants plus fidèles et plus efficaces contre les problématiques que l'on a traitées dans cette thèse.

Ci-dessous quelques perspectives pour les méthodes proposées:

- Pour l'algorithme LPCN, nous travaillons actuellement sur la version où l'algorithme peut démarrer à partir de n'importe quel point du graphe au lieu du point ayant une coordonnée x minimale, et nous préparons également une version tridimensionnelle.
- Concernant l'approche de détection de défaillance, nous envisageons de l'analyser en termes de résilience aux changements des défaillances, en injectant des défauts dans le réseau, et

d'étudier la performance du réseau dans ce cas et comment router immédiatement les notifications d'échec à travers le réseau.

- Pour la détection des capteurs usés, nous envisageons d'implémenter notre approche pour un scénario du monde réel dans un véritable réseau de capteurs, et de montrer l'efficacité de la théorie des Copules pour traiter les données multivariées.
- Concernant la technique de tatouage, nous prévoyons de l'implémenter sur un vrai RCSF, et d'étudier le niveau de sa robustesse contre les attaques de modification et de falsification de données en faisant varier la taille de la signature à chaque fois. Nous prévoyons ensuite d'étudier son impact sur la consommation d'énergie des nœuds capteurs.

Références Bibliographiques

[1] Ronald L Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972. i, 8, 14

[2] Ray A Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973. i, 8, 9, 13, 14

[3] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. i, 9, 14

[4] Michael Kallay. The complexity of incremental convex hull algorithms in r^d. *Information Processing Letters*, 19(4):197, 1984. i, 9, 14

[5] Abel JP Gomes. A total order heuristic-based convex hull algorithm for points in the plane. *Computer-Aided Design*, 70:153–160, 2016. i, 9, 14

[6] Changyuan Xing, Zhongyang Xiong, Yufang Zhang, Xuegang Wu, Jingpei Dan, and Tingping Zhang. An efficient convex hull algorithm using affine transformation in planar point set. *Arabian Journal for Science and Engineering*, 39(11):7785–7793, 2014. i, 9, 14

[7] Gang Mei. Cudachain: an alternative algorithm for finding 2d convex hulls on the gpu. *SpringerPlus*, pages 1–26, 2016. i, 10, 14

[8] Antonio Ruano, Hamid Reza Khosravani, and Pedro M Ferreira. A randomized approximation convex hull algorithm for high dimensions. *IFAC-PapersOnLine*, 48(10):123–128, 2015. i, 10, 14

[9] Vaclav Skala, Zuzana Majdisova, and Michal Smolik. Space subdivision to speed-up convex hull construction in e3. *Advances in Engineering Software*, 91:12–22, 2016. i, 10, 14

[10] Gautam Garai and B. B. Chaudhuri. A split and merge procedure for polygonal border detection of dot pattern. *Image and Vision Computing*, 17(1):75–82, 1999. i, 10, 14

[11] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559, 1983. i, 10, 14

[12] A Ray Chaudhuri, Bidyut Baran Chaudhuri, and Swapan K Parui. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68(3):257–275, 1997. i, 11, 14

[13] Adriano Moreira and Maribel Yasmina Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. 2007. i, 11, 14

[14] Jin-Seo Park and Se-Jong Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of information science and engineering*, 29(2):379–392, 2013. i, 11, 14

[15] Christian Braune, Marco Dankel, and Rudolf Kruse. Obtaining shape descriptors from a concave hull-based clustering algorithm. In *International Symposium on Intelligent Data Analysis*, pages 61–72. Springer, 2016. i, 11, 14

[16] A. Gheibi, M. Davoodi, A. Javad, F. Panahi, M. M. Aghdam, M. Asgaripour, and A. Mohades. Polygonal shape reconstruction in the plane. *IET Computer Vision*, 5(2):97–106, March 2011. i, 12, 14

[17] Subhasree Methirumangalath, Amal Dev Parakkat, and Ramanathan Muthuganapathy. A unified approach towards reconstruction of a planar point set. *Computers & Graphics*, 51:90–97, 2015. i, 12, 14

[18] E Rosén, E Jansson, and M Brundin. Implementation of a fast and efficient concave hull algorithm. Technical report, University of Uppsala, Sweden, 2014. i, 12, 14

[19] Marco Körner, Mahesh V Krishna, Herbert Süße, Wolfang Ortmann, and Joachim Denzler. Regularized geometric hulls for bio-medical image segmentation. *The Annals of the BMVA,(4)*, pages 1–12, 2015. i, 12, 14

[20] Bambang Harjito, Vidyasagar Potdar, and Jaipal Singh. Watermarking technique for wireless multimedia sensor networks: a state of the art, september 03-05, pune, india. In *Proceedings of the CUBE International Information Technology Conference*, pages 832–840. ACM, 2012. 3, 81

[21] Farid Lalem, Ahcène Bounceur, Madani Bezoui, Massinissa Saoudi, Reinhardt Euler, Tahar Kechadi, and Marc Sevaux. Lpcn: Least polar-angle connected node algorithm to find a polygon hull in a connected euclidean graph. volume 93, pages 38–50. Elsevier, 2017. 4, 38, 93

[22] Massinissa Saoudi, Farid Lalem, Ahcène Bounceur, Reinhardt Euler, M-Tahar Kechadi, Abdelkader Laouid, Madani Bezoui, and Marc Sevaux. D-lpcn: A distributed least polar-angle connected node algorithm for finding the boundary of a wireless sensor network. *Ad Hoc Networks*, 2016. 4, 5, 31, 94

[23] Farid Lalem, Rahim Kacimi, Ahcène Bounceur, and Reinhardt Euler. Boundary node failure detection in wireless sensor networks. In *IEEE International Symposium on Networks, Computers and Communications (ISNCC 2016), 11-13 May, Hammamet, Tunisia*, 2016. 4, 7, 94

[24] Farid Lalem, Ahcène Bounceur, Rahim Kacimi, Reinhardt Euler, and Massinissa Saoudi. Faulty data detection in wireless sensor networks based on copula theory. In *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies, November 10-11, Blagoevgrad, Bulgaria*, BDAW '16, pages 29:1–29:7, New York, NY, USA, 2016. ACM. 4, 7, 57, 94

[25] Farid Lalem, Ahcène Bounceur, Reinhardt Euler, Hammoudeh Mohammad, Kacimi Rahim, and Sanaa Kawther Ghalem. Distributed faulty sensor node detection in wireless sensor networks based on copula theory. In *Second International Conference on Internet of Things, Data and Cloud Computing (ICC 2017), Cambridge, United Kingdom*, 2017. 4, 94

[26] Farid Lalem, Ahcène Bounceur, Massinissa Saoudi, Reinhardt Euler, and Rahim kacimi. Anomaly and event detection in wireless sensor networks based on copula theory and spatial correlation. (submitted). 4, 94

[27] Farid Lalem, Alshaikh Muath, Ahcene Bounceur, Reinhardt Euler, Lamri Laouamer, Laurent Nana, and Anca Christine Pascu. Data authenticity and integrity in wireless sensor networks

based on a watermarking approach. In *The 29th International Florida Artificial Intelligence Research Society, Florid, USA*, 2016. 4

[28] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0. 5, 73, 94

[29] Ahcène Bounceur, Reinhardt Euler, Ali Benzerbadj, Farid Lalem, Massinissa Saoudi, Tahar Kechadi, and Marc Sevaux. Finding a polygon hull in wireless sensor networks. In *European Conference on Operational Research, University of Strathclyde, Glasgow, UK*. Invited talk, EURO 2015, July 2015. 7, 42

[30] José Oswaldo Cadenas, Graham M Megson, and Cris L Luengo Hendriks. Preconditioning 2d integer data for fast convex hull computations. *PloS one*, 11(3):e0149860, 2016. 7

[31] Jingfan Fan, Jian Yang, Mahima Goyal, and Yongtian Wang. Rigid registration of 3-d medical image using convex hull matching. In *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*, pages 338–341. IEEE, 2013. 7

[32] MA Jayaram and Hasan Fleyeh. Convex hulls in image processing: A scoping review. *American Journal of Intelligent Systems*, 6(2):48–58, 2016. 7

[33] Yali Li, Shengjin Wang, Qi Tian, and Xiaoqing Ding. A survey of recent advances in visual feature detection. *Neurocomputing*, 149:736–751, 2015. 7

[34] Feminna Sheeba, Robinson Thamburaj, Joy John Mammen, Mohan Kumar, and Vansant Rangslang. Convex hull based detection of overlapping red blood cells in peripheral blood smear images. In *7th WACBE World Congress on Bioengineering 2015*, pages 51–53. Springer, 2015. 7

[35] Alex M Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. 9

[36] Avraham A. Melkman. On-line construction of the convex hull of a simple polyline. *Inf. Process. Lett.*, 25(1):11–12, April 1987. 10

[37] Godfried T Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980. 12

[38] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000. 12

[39] Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. On the design of CGAL, the computational geometry algorithms library. 1998. 12

[40] Ivor D. Faux and Michael J. Pratt. *Computational geometry for design and manufacture*. Ellis Horwood Ltd, 1979. 12

[41] Ketan Mulmuley. *Computational geometry: An introduction through randomized algorithms*. Prentice Hall, 1994. 12

[42] Franco P. Preparata and Michael Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012. 12

[43] Selim G Akl. Two remarks on a convex hull algorithm. *Information Processing Letters*, 8(2):108–109, 1979. 13

[44] Kamal Mehdi, Massinissa Lounis, Ahcène Bounceur, and Tahar Kechadi. Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool. *In IEEE 7th International Conference on Simulation Tools and Techniques (SIMUTools'14), Lisbon, Portugal*, March 17-19 2014. 29, 87

[45] http://www.cupcarbon.com. 29

[46] Harith Alani, Christopher B. Jones, and Douglas Tudhope. Voronoi-based region approximation for geographical information retrieval with gazetteers. *International Journal of Geographical Information Science*, 15(4):287–306, 2001. 31

[47] Emad Felemban. Advanced border intrusion detection and surveillance using wireless sensor network technology. 2013. 37

[48] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002. 37, 57

[49] Luciana Moreira Sá De Souza, Harald Vogt, and Michael Beigl. A survey on fault tolerance in wireless sensor networks. *Interner Bericht. Fakultät für Informatik, Universität Karlsruhe*, 2007. 37

[50] Lanny Sitanayah, Amitava Datta, and Rachel Cardell-Oliver. Heuristic algorithm for finding boundary cycles in location-free low density wireless sensor networks. *Computer Networks*, 54(10):1630–1645, 2010. 37, 38, 39

[51] Thanh Le Dinh. Topological boundary detection in wireless sensor networks. *JIPS*, 5(3):145–150, 2009. 37, 39

[52] Massinissa Saoudi, Ahcene Bounceur, Reinhardt Euler, and Tahar Kechadi. Data mining techniques applied to wireless sensor networks for early forest fire detection. In *International Conference on Internet of things and Cloud Computing (ICC), 2015*, pages 71–77. ACM, 2015. 38

[53] Baoqi Huang, Wei Wu, Guanglai Gao, and Tao Zhang. Recognizing boundaries in wireless sensor networks based on local connectivity information. *International Journal of Distributed Sensor Networks*, 2014, 2014. 38

[54] Baoqi Huang, Wei Wu, and Tao Zhang. An improved connectivity-based boundary detection algorithm in wireless sensor networks. In *In 38th IEEE Conference on Local Computer Networks (LCN)*, pages 332–335. IEEE, 2013. 38

[55] Stefan Funke and Christian Klein. Hole detection or: how much geometry hides in connectivity? In *Proceedings of the twenty-second annual symposium on Computational geometry, June 5-7, Sedona, Arizona, USA*, pages 377–385. ACM, 2006. 38

[56] Kyle Luthy, Edward Grant, Nikhil Deshpande, and Thomas C Henderson. Perimeter detection in wireless sensor networks. *Robotics and Autonomous Systems*, 60(2):266–277, 2012. 39

[57] Prasan Kumar Sahoo, Kun-Ying Hsieh, and Jang-Ping Sheu. Boundary node selection and target detection in wireless sensor network. In *IFIP International Conference on Wireless and Optical Communications Networks (WOCN'07)*, pages 1–5. IEEE, 2007. 39

[58] Shailendra Shukla and Rajiv Misra. Angle based double boundary detection in wireless sensor networks. *Journal of Networks*, 9(3):612–619, 2014. 39, 42

[59] Carlos Lara-Alvarez, Juan J Flores, and Chieh-Chih Wang. Detecting the boundary of sensor networks from limited cyclic information. *International Journal of Distributed Sensor Networks, Hindawi Publishing Corporation, Volume 2015, Article ID 401838*, 2015. 39

[60] Prasan Kumar Sahoo, Jang-Ping Sheu, and Kun-Ying Hsieh. Target tracking and boundary node selection algorithms of wireless sensor networks for internet services. *Information Sciences*, 230:21–38, 2013. 39, 42, 50

[61] Wei-Cheng Chu and Kuo-Feng Ssu. Location-free boundary detection in mobile wireless sensor networks with a distributed approach. *Computer Networks*, 70:96–112, 2014. 40, 42

[62] Alexander Kröller, Sándor P Fekete, Dennis Pfisterer, and Stefan Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1000–1009. Society for Industrial and Applied Mathematics, 2006. 40

[63] Wei-Cheng Chu and Kuo-Feng Ssu. Decentralized boundary detection without location information in wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 1720–1724. IEEE, 2012. 40

[64] Supantha Das, Indrajit Banerjee, and Tuhina Samanta. Sensor localization and obstacle boundary detection algorithm in wsn, 29-31 aug, cochin, india. In *Advances in Computing and Communications (ICACC), 2013 Third International Conference on*, pages 412–415. IEEE, 2013. 40

[65] Li-Hui Zhao, Wenyi Liu, Haiwei Lei, Ruixia Zhang, and Qiulin Tan. The detection of boundary nodes and coverage holes in wireless sensor networks. *Journal of Mobile Information Systems, Volume 2016 (2016)*, 2016, 2016. 40, 50

[66] Chi Zhang, Yanchao Zhang, and Yuguang Fang. Detecting coverage boundary nodes in wireless sensor networks. In *2006 IEEE International Conference on Networking, Sensing and Control, 23-25 April, Ft. Lauderdale, FL, USA*, pages 868–873. IEEE, 2006. 41, 42

[67] Nihel Senouci, Moustafa Kouider El Ouahed, and Hafid Haffaf. Detecting boundary nodes in wsn. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014. 41, 50

[68] Ijaz Muhammad Khan, Nafaâ Jabeur, and Sherali Zeadally. Hop-based approach for holes and boundary detection in wireless sensor networks. *IET Wireless Sensor Systems*, 2(4):328–337, 2012. 41, 42

[69] Ahmed M Khedr, Walid Osamy, and Dharma P Agrawal. Perimeter discovery in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 69(11):922–929, 2009. 41, 42

[70] Srabani Kundu and Nabanita Das. Event boundary detection and gathering in wireless sensor networks. In *Applications and Innovations in Mobile Computing (AIMoC), 2015*, pages 62–67. IEEE, 2015. 41, 42

[71] Linnyer Beatrys Ruiz, José Marcos Nogueira, and Antonio AF Loureiro. Manna: A management architecture for wireless sensor networks. *Communications Magazine, IEEE*, 41(2):116–125, 2003. 43

[72] Daler Rakhamtov and Sarma Vrudhula. Time-to-failure estimation for batteries in portable electronic systems. In *International symposium on Low power electronics and design*, pages 88–91. ACM, 2001. 43

[73] Luca Benini, Giuliano Castelli, Alberto Macii, Enrico Macii, Massimo Poncino, and Riccardo Scarsi. A discrete-time battery model for high-level power estimation. In *Proceedings of the conference on Design, automation and test in Europe*, pages 35–41. ACM, 2000. 43

[74] Peter Rothenpieler, Daniela Krüger, Dennis Pfisterer, Stefan Fischer, Denise Dudek, Christian Haas, Andreas Kuntz, and Martina Zitterbart. Flegsens-secure area monitoring using wireless sensor networks. *Proceedings of the 4th Safety and Security Systems in Europe*, 2009. 43

[75] Ravindra N Duche and NP Sarwade. Sensor node failure or malfunctioning detection in wireless sensor network. *ACEEE Int. J. Commun*, 3(1):57–61, 2012. 43

[76] Anas Abu Taleb, Dhiraj K Pradhan, and Taskin Kocak. A technique to identify and substitute faulty nodes in wireless sensor networks. In *Sensor Technologies and Applications, 2009. Third International Conference on*, pages 346–351. IEEE, 2009. 43

[77] Nithya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 255–267. ACM, 2005. 43

[78] M. Lounis, K. Mehdi, and A. Bounceur. A cupcarbon tool for simulating destructive insect movements. *1st IEEE International Conference on Information and Communication Technologies for Disaster Management (ICT-DM'14), Algiers, Algeria*, March 24-25 2014. 46

[79] Arvinderpal S Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 324–328. IEEE, 2005. 48, 88

[80] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. 2011. 48

[81] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004. 48

[82] Giancarlo Fortino, Raffaele Greco, and Antonio Guerrieri. Modeling and evaluation of the building management framework based on the castalia wsn simulator. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 668–674. IEEE, 2013. 53

[83] Yang Zhang, Nirvana Meratnia, and Paul Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE*, 12(2):159–170, 2010. 57

[84] Eiman Elnahrawy. Research directions in sensor data streams: solutions and challenges. *Rutgers University, Tech. Rep. DCIS-TR-527*, 2:D3, 2003. 57

[85] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006. 57

[86] Shah Ahsanul Haque, Mustafizur Rahman, and Syed Mahfuzul Aziz. Sensor anomaly detection in wireless sensor networks for healthcare. *Sensors*, 15(4):8764–8786, 2015. 57

[87] Niki Trigoni et al. *Learning from Data Streams: Processing Techniques in Sensor Networks. Chapter 6: Querying of Sensor Data*. Springer-Verlag Berlin Heidelberg, 2007. 58

[88] Miao Xie, Song Han, Biming Tian, and Sazia Parvin. Anomaly detection in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 34(4):1302–1325, 2011. 58, 59

[89] Murad A Rassam, Anazida Zainal, and Mohd Aizaini Maarof. Advancements of data anomaly detection research in wireless sensor networks: A survey and open issues. *Sensors*, 13(8):10087–10122, 2013. 58, 59

[90] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 2007. 58

[91] Nauman Shahid, Ijaz Haider Naqvi, and Saad Bin Qaisar. Characteristics and classification of outlier detection techniques for wireless sensor networks in harsh environments: a survey. *Artificial Intelligence Review*, 43(2):193–228, 2012. 59

[92] Shawn R Jeffery, Gustavo Alonso, Michael J Franklin, Wei Hong, and Jennifer Widom. Declarative support for sensor data cleaning. In *Pervasive Computing*, pages 83–100. Springer, 2006. 59

[93] Dylan McDonald, Stewart Sanchez, Sanjay Madria, and Fikret Ercal. A survey of methods for finding outliers in wireless sensor networks. *Journal of Network and Systems Management*, 23(1):163–182, 2015. 59

[94] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. Detecting data anomalies in wireless sensor networks. *Secur. Ad Hoc. Sens. Netw*, 3:231–259, 2009. 59

[95] Satish S Bhojannawar, Chetan M Bulla, and Vishal M Danawade. Anomaly detection techniques for wireless sensor networks-a survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(10), 2013. 59

[96] Weili Wu, Xiuzhen Cheng, Min Ding, Kai Xing, Fang Liu, and Ping Deng. Localized outlying and boundary data detection in sensor networks. *Knowledge and Data Engineering, IEEE Transactions on*, 19(8):1145–1157, 2007. 59, 63

[97] Yuan Yao, Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation*, 67(11):1059–1075, 2010. 60

[98] Luís MA Bettencourt, Aric A Hagberg, and Levi B Larkey. Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks. In *International Conference on Distributed Computing in Sensor Systems*, pages 223–239. Springer, 2007. 60

[99] Minwook C Jun, H Jeong, and C-C Jay Kuo. Distributed spatio-temporal outlier detection in sensor networks. In *Proceedings of SPIE*, volume 5819, pages 273–284, 2005. 60

[100] Bo Sheng, Qun Li, Weizhen Mao, and Wen Jin. Outlier detection in sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 219–228. ACM, 2007. 60

[101] Miao Xie, Jiankun Hu, and Biming Tian. Histogram-based online anomaly detection in hierarchical wireless sensor networks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 751–759. IEEE, 2012. 60

[102] Kejia Zhang, Shengfei Shi, Hong Gao, and Jianzhong Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *Advanced Data Mining and Applications*, pages 158–169. Springer, 2007. 61

[103] N Chitradevi, V Palanisamy, K Baskaran, and K Swathithya. Efficient density based techniques for anomalous data detection in wireless sensor networks. *Journal of Applied Science and Engineering*, 16(2):211223, 2013. 61

[104] S. Rajasegarar, C. Leckie, M. Palaniswami, and J.C. Bezdek. Distributed anomaly detection in wireless sensor networks. In *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*, pages 1–5, 2006. 61

[105] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. Hyperspherical cluster based distributed anomaly detection in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1833–1847, 2014. 61

[106] Saowaluk Takianngam and Wipawee Usaha. Discrete wavelet transform and one-class support vector machines for anomaly detection in wireless sensor networks. In *Intelligent Signal Processing and Communications Systems (ISPACS'2011), International Symposium on*, pages 1–6. IEEE, 2011. 62

[107] Anshuman Bhargava and AS Raghuvanshi. Anomaly detection in wireless sensor networks using s-transform in combination with svm. In *Computational Intelligence and Communication Networks (CICN), 2013 5th International Conference on*, pages 111–116. IEEE, 2013. 62

[108] Eiman Elnahrawy and Badri Nath. Context-aware sensors. In *Wireless Sensor Networks*, pages 77–93. Springer, 2004. 62

[109] D Janakiram, V Adi Mallikarjuna Reddy, et al. Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and Middleware, Comsware 2006.*, pages 1–6. IEEE, 2006. 62

[110] David J Hill, Barbara S Minsker, and Eyal Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research*, volume 32, page 503. Citeseer, 2007. 62

[111] Xingfeng Guo, Dianhong Wang, and Fenxiong Chen. An anomaly detection based on data fusion algorithm in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2015. 62

[112] Shuo Guo, Heng Zhang, Ziguo Zhong, Jiming Chen, Qing Cao, and Tian He. Detecting faulty nodes with data errors for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(3):40, 2014. 63

[113] Safdar Abbas Khan, Boubaker Daachi, and Karim Djouani. Application of fuzzy inference systems to detection of faults in wireless sensor networks. *Neurocomputing*, 94:111–120, 2012. 63

[114] Farinaz Koushanfar, Miodrag Potkonjak, and Alberto Sangiovanni-Vincentelli. On-line fault detection of sensor measurements. In *Sensors, 2003. Proceedings of IEEE*, volume 2, pages 974–979. IEEE, 2003. 63

[115] Jinran Chen, Shubha Kher, and Arun Somani. Distributed fault detection of wireless sensor networks. In *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72. ACM, 2006. 63

[116] Meenakshi Panda and Pabitra Mohan Khilar. Distributed soft fault detection algorithm in wireless sensor networks using statistical test. In *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*, pages 195–198. IEEE, 2012. 63

[117] M Sklar. *Fonctions de répartition à n dimensions et leurs marges*. Université Paris 8, 1959. 64

[118] Roger B Nelsen. *An introduction to copulas*. Springer Science & Business Media, 2007. 64

[119] Christian Genest and Anne-Catherine Favre. Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of hydrologic engineering*, 12(4):347–368, 2007. 65

[120] Paul Deheuvels. La fonction de dépendance empirique et ses propriétés. un test non paramétrique d'indépendance. *Acad. Roy. Belg. Bull. Cl. Sci.(5)*, 65(6):274–292, 1979. 65

[121] Marek Omelka, Irène Gijbels, Noël Veraverbeke, et al. Improved kernel estimation of copulas: weak convergence and goodness-of-fit testing. *The Annals of Statistics*, 37(5B):3023–3058, 2009. 68

[122] William F Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):398–403, 1977. 70

[123] Ludger Rüschendorf. On the distributional transform, sklar's theorem, and the empirical copula process. *Journal of Statistical Planning and Inference*, 139(11):3921–3927, 2009. 70

[124] Joseph o'Rourke. *Computational geometry in C*. Cambridge University Press, 1998. 70

[125] Intel Lab Data, 2004. 73

[126] Jennifer L Wong, Jessica Feng, Darko Kirovski, and Miodrag Potkonjak. Security in sensor networks: watermarking techniques. In *Wireless sensor networks*, pages 305–323. Springer, 2004. 81

[127] Bambang Harjito and Vidyasagar Potdar. Secure transmission in wireless sensor networks data using linear kolmogorov watermarking technique. *arXiv preprint arXiv:1501.01376*, 2015. 81

[128] Ioan-Catalin Dragoi and Dinu Coltuc. On local prediction based reversible watermarking. *Image Processing, IEEE Transactions on*, 24(4):1244–1246, 2015. 81

[129] Baowei Wang, Jianwei Su, Youdong Zhang, Biqiang Wang, Jian Shen, Qun Ding, and Xingming Sun. A copyright protection method for wireless sensor networks based on digital watermarking. *International Journal of Hybrid Information Technology*, 8(6):257–268, 2015. 81, 91

[130] B Sai Sindhush, RV Keshava Rao, and R Bulli Babu. Digital data theft detection using watermarking. *Global Journal of Computer Science and Technology*, 14(9), 2015. 81

[131] Abdelhamid Benhocine, Lamri Laouamer, Laurent Nana, and Anca Christine Pascu. New images watermarking scheme based on singular value decomposition. *Journal of Information Hiding and Multimedia Signal Processing*, 4(1):9–18, 2013. 81

[132] Lamri Laouamer and Omar Tayan. A semi-blind robust dct watermarking approach for sensitive text images. *Arabian Journal for Science and Engineering*, 40(4):1097–1109, 2015. 82

[133] Xiaojun Qi and Xing Xin. A singular-value-based semi-fragile watermarking scheme for image content authentication with tamper localization. *Journal of Visual Communication and Image Representation*, 2015. 82

[134] Xingming Sun, Jianwei Su, Baowei Wang, and Qi Liu. Digital watermarking method for data integrity protection in wireless sensor networks. *International Journal of Security and Its Applications*, 7(4):407–416, 2013. 82, 91

[135] Ibrahim Kamel and Hussam Juma. A lightweight data integrity scheme for sensor networks. *Sensors*, 11(4):4118–4136, 2011. 83, 88

[136] Baowei Wang, Xingming Sun, Zhiqiang Ruan, and Heng Ren. Multi-mark: multiple watermarking method for privacy data protection in wireless sensor networks. *Information Technology Journal*, 10(4):833–840, 2011. 83

[137] Qun Ding, Baowei Wang, Xingming Sun, Jinwei Wang, and Jian Shen. A reversible watermarking scheme based on difference expansion for wireless sensor networks. *International Journal of Grid Distribution Computing*, 08(2):143–154, 2015. 83

[138] Xiaomei Dong and Xiaohua Li. An authentication method for self nodes based on watermarking in wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, pages 1–4. IEEE, 2009. 83

[139] CupCarbon. Anr project persepteur, cupcarbon simulator, 2015. 87

[140] B Kishore Kumar and GK Venkata Narasimha Reddy. Identification of packet dropping and modification in wireless sensor networks. *International Journal of Computational Engineering Research (IJCER)*, 2013. 87

Cadre méthodologique et applicatif pour le développement de réseaux de capteurs fiables

Les réseaux de capteurs sans fil émergent comme une technologie innovatrice qui peut révolutionner et améliorer notre façon de vivre, de travailler et d'interagir avec l'environnement physique qui nous entoure. Néanmoins, l'utilisation d'une telle technologie soulève de nouveaux défis concernant le développement de systèmes fiables et sécurisés. Ces réseaux de capteurs sans fil sont souvent caractérisés par un déploiement dense et à grande échelle dans des environnements limités en terme de ressources. Les contraintes imposées sont la limitation des capacités de traitement, de stockage et surtout d'énergie car ils sont généralement alimentés par des piles.
Nous visons comme objectif principal à travers cette thèse à proposer des solutions permettant de garantir un certain niveau de fiabilité dans un RCSF dédié aux applications sensibles. Nous avons ainsi abordé trois axes, qui sont :

- Le développement de méthodes permettant de détecter les nœuds capteurs défaillants dans un RCSF,
- Le développement de méthodes permettant de détecter les anomalies dans les mesures collectées par les nœuds capteurs, et par la suite, les capteurs usés (fournissant de fausses mesures).
- Le développement de méthodes permettant d'assurer l'intégrité et l'authenticité des données transmise dans un RCSF.

Mots clés : Réseaux de capteurs sans fil, Nœuds capteurs défaillants, Fiabilité de données, Nœuds frontières, Copule, Détection d'anomalies.

The Design of Reliable Sensor Networks: Methods and Applications.

Wireless sensor networks emerge as an innovative technology that can revolutionize and improve our way to live, work and interact with the physical environment around us. Nevertheless, the use of such technology raises new challenges in the development of reliable and secure systems. These wireless sensor networks are often characterized by dense deployment on a large scale in resource-constrained environments. The constraints imposed are the limitation of the processing, storage and especially energy capacities since they are generally powered by batteries.
Our main objective is to propose solutions that guarantee a certain level of reliability in a WSN dedicated to sensitive applications. We have thus proposed three axes, which are:

- The development of methods for detecting failed sensor nodes in a WSN.
- The development of methods for detecting anomalies in measurements collected by sensor nodes, and subsequently fault sensors (providing false measurements).
- The development of methods ensuring the integrity and authenticity of transmitted data over a WSN.

Keywords : Wireless sensor networks, Faulty sensor nodes, Data reliability, Border nodes, Copula, Anomaly detection.