REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Étude et optimisation du déploiement d'un réseau de capteurs sans fil dans un bâtiment intelligent

---

Imene SI HADJ MOHAND

Sous la direction de
Dr. Nour el Houda BENALIA, Maître de Conférence à l'Ecole Nationale Polytechnique

Présenté et soutenu publiquement le (07/09/2020)

Composition du jury :

M. Mourad ADNANE	**Professeur**	**ENP**	**Président**
Mme. Nour El-Houda BENALIA	**Docteur**	**ENP**	**Promotrice**
Mme. Nesrine BOUADJENEK	**Docteur**	**ENP**	**Examinatrice**

ENP 2020

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Étude et optimisation du déploiement d'un réseau de capteurs sans fil dans un bâtiment intelligent

---

Imene SI HADJ MOHAND

Sous la direction de
Dr. Nour el Houda BENALIA, Maître de Conférence à l'Ecole Nationale Polytechnique

Présenté et soutenue publiquement le (07/09/2020)

Composition du jury :

M. Mourad ADNANE	**Professeur**	**ENP**	**Président**
Mme. Nour El-Houda BENALIA	**Docteur**	**ENP**	**Promotrice**
Mme. Nesrine BOUADJENEK	**Docteur**	**ENP**	**Examinatrice**

ENP 2020

Dédicaces

Je dédie ce modeste travail à mon papa, ma maman, mes petits anges : Sabah, Yasmine, Walide et Djamel. Ainsi qu'à tous ceux qui m'ont apporté leur soutien.

Remerciements

Je tiens à exprimer toute ma gratitude à ma promotrice **Dr. Nour el houda BENALIA** pour tout le temps qu'elle a consacré à la direction de ce travail. Je lui suis très reconnaissante pour son soutien constant, sa patience et sa motivation. Je la remercie pour tous les précieux conseils qu'elle m'a apporté, et qui ont contribué à alimenter ma réflexion et m'ont aidé à surmonter bien des obstacles.

Je remercie également l'ensemble des membres du jury de m'avoir fait l'honneur d'examiner ce travail, tout en leur adressant mon plus profond respect.

Je souhaite par la même occasion remercier tous mes enseignants du département des classes préparatoires ainsi que ceux du département d'électronique de **l'École Nationale Polytechnique** pour m'avoir inspiré et guidé vers la réussite au cours des cinq dernières années. Enfin, je remercie tous les acteurs de l'école qui ont contribué à m'offrir tous les outils nécessaires à la réussite de ma formation d'ingénieur.

Toujours plus loin, toujours plus haut, toujours plus fort ! Telle est la devise que mon cher père n'a cessé de me répéter tout au long de mon parcours. J'en profite pour le remercier de m'avoir toujours poussé au-delà de mes limites. Pour finir, je remercie infiniment ma chère maman, mes frères et soeurs ainsi que tous mes proches de m'avoir soutenue sans relâche tout au long de cette belle aventure.

ملخص

خلال مشروع نهاية الدراسة هذا ، تناولنا مشكلة نشر شبكات الاستشعار اللاسلكية في المباني الذكية. كان هدفنا هو تطوير أداة من شأنها إنشاء مخطط نشر فعال ، وتقديم خدمة ذات جودة عالية

بدأنا بتقديم شبكات الاستشعار اللاسلكية. ثم قمنا بنمذجة المعلمات الرئيسية لنشر شبكات الاستشعار اللاسلكية ، وهي التكلفة والاتصال والتغطية والتغطية الزائدة. بعد ذلك ، اخترنا ثلاث خوارزميات تطورية متعددة الأهداف قمنا بتنفيذها من أجل حل مشكلة النشر. حصلنا على نتائج مرضية من المحاكاة

الكلمات الرئيسية : شبكة الاستشعار اللاسلكية ، النشر ، الاتصال ، التغطية ، التحسين متعدد الأهداف ، الخوارزمية التطورية

Abstract

During this graduation project, we dealt with the problem of deploying wireless sensor networks in smart buildings. Our goal was to develop a tool that would establish an efficient deployment scheme, offering good quality of service.

We started with the introduction of wireless sensor networks. Then, we modeled the key parameters of a wireless sensor network deployment, namely cost, connectivity, coverage and over-coverage. Thereafter, we selected three evolutionary multi-objective algorithms that we implemented in order to solve the deployment problem. We obtained satisfactory results from the simulations.

Keywords: wireless sensor network, deployment, connectivity, coverage, multi-objective optimization, evolutionary algorithm.

Résumé

Nous avons traité au cours de ce projet de fin d'étude le problème de déploiement des réseaux de capteurs sans fil dans les bâtiments intelligents. Notre objectif était de développer un outil qui permette d'établir un schéma de déploiement efficace, offrant une bonne qualité de service.

Nous avons commencé par l'introduction des réseaux de capteurs sans fil. Puis nous avons modélisé les paramètres clés d'un déploiement d'un réseau de capteurs sans fil, à savoir le coût, la connectivité, la couverture et la sur-couverture. Par la suite, nous avons sélectionné trois algorithmes évolutionnaires multi-objectifs que nous avons implémenté afin de résoudre le problème de déploiement. Nous avons obtenu à l'issu des simulations des résultats satisfaisants.

Mots clés: réseau de capteurs sans fil, déploiement, connectivité, couverture, optimisation multi-objectif, algorithme évolutionnaire.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

2D : 2 dimensions

3D : 3 dimensions

ABC : Artificial Bee Colony

ACO : Ant Colony Optimization

AEMO : Algorithme évolutionnaire multi-objectif

AG : Algorithme génétique

AGMO : Algorithme génétique multi-objectif

BAC : Binary Ant Colony

CMA-ES : Covariance Matrix Adaptation Evolution Strategy

dBm : Décibel par rapport au milliwatt

GTB : Gestion technique du bâtiment

IDDT-GA : Improved dynamic deployment technique based-on genetic algorithm

Iot : Internet of things

ITU : International Telecommuication Union

NLOS : Non line of sight

NP : Non-deterministic Polynomial time

NSGA : Non dominated sorting genetic algorithm

PSO : Particle Swarm Optimisation

QoS : Quality of Service

RSSI : Recieved Signal Streght Indication

SPEA : Strength Pareto evolutionary algorithm

TDMA : Time Division Multiple Access

TSP : Travel Salesman Problem

UHF : Ultra Heigh Frequency

UWB : Ultra wideband

RCSF : Réseau de capteurs sans fil

Rd : Rayon de détection

Introduction générale

Contexte et motivation L'être humain toujours soucieux de son confort ne cesse de chercher des solutions et d'innover dans le but de se faciliter la vie. Avec l'émergence des nouvelles technologies de communication et la miniaturisation des dispositifs électroniques, un nouveau concept est né : le bâtiment intelligent (Smart Building). Grâce à des réseaux de capteurs sans fils (RCSFs) déployés au sein des bâtiments, il est maintenant possible de déléguer la gestion des habitations et des espaces de travail à des systèmes de gestion appelés GTB (Gestion technique du bâtiment). Ces derniers, en collectant des informations liées à l'environnement (température, humidité, luminosité, présence ...) prennent des décisions afin de maximiser le confort des occupants et d'optimiser la consommation énergétique des bâtiments.

Problématique Notre problématique s'inscrit dans ce contexte. Elle vise à étudier la conception et le déploiement de ce type de réseaux. En effet, pour pouvoir déployer un RCSF, il faut respecter un certain nombre de contraintes dans le but d'aboutir à un réseau fonctionnel qui réponde aux besoins de l'application à laquelle il est destiné. Généralement, ces contraintes sont attachées directement à la connectivité de l'ensemble des capteurs, à la couverture de la zone d'intérêt, à leurs durée de vie, etc. Le déploiement d'un RCSF consiste à déterminer le nombre et les positions des noeuds. Ces derniers doivent former un réseau respectant les taux de connectivité, couverture et sur-couverture désirés.

Avec l'avancée technologique et le développement des ordinateurs qui sont devenus très performants et dotés d'une plus grande capacité de mémoire et de calcul, plusieurs outils de déploiement ont été proposé pour aider les concepteurs à trouver la topologie optimale d'un réseau en utilisant plusieurs notions et techniques. Ces dernières vont être abordées dans les chapitres de notre document.

Ces dernières décennies, les méthodes d'optimisation et les méta-heuristiques ont été utilisées pour concevoir et déployer les RCSFs. Dans ce sens et à part quelques travaux, les solutions proposées traitent que d'une seule ou deux contraintes au maximum. Alors que, le développement d'un outil capable de traiter trois objectifs ou plus est une nécessité primordiale pour la conception d'un RCSF efficace. Pour cela, une bonne modélisation du système et un bon choix de la méthode d'optimisation de déploiement doivent être fait.

Objectif Le but de notre travail est de développer une application qui permette d'établir un plan de déploiement d'un RCSF dans un espace fermé (Indoor). Pour ce faire, il est nécessaire en premier lieu de modéliser les différents critères de déploiement. Ces critères sont : le coût de déploiement qui est proportionnel au nombre de noeuds capteurs et noeuds routeurs déployés et qui doit être minimisé, la couverture du bâtiment qui doit être complète pour une surveillance sans faille, la connectivité du réseau, qui elle aussi doit être complète pour assurer une bonne qualité de service, et finalement, la sur-couverture pour éviter de déployer des noeuds sans intérêt. Ensuite, une méthode d'optimisation multi-objectif doit être appliquée afin de déterminer le nombre et l'emplacement des différents noeuds constituants le réseau.

Organisation du mémoire Le document est organisé en deux parties. La première partie est composée de trois chapitres. Dans le premier chapitre, nous présenteront un état de l'art sur les bâtiments intelligents et les RCSFs. Le deuxième chapitre portera sur la modélisation des différents critères de déploiement. Finalement, dans le dernier chapitre de cette partie, nous exposerons les différentes méthodes d'optimisation utilisées pour la résolution du problème de déploiement des RCSFs. La deuxième partie du document sera consacrée à la présentation des méthodes que nous avons sélectionnées. La démarche entreprise pour leur implémentation ainsi que les résultats des simulations y seront également exposés. Pour finir, nous conclurons notre travail et présenteront les perspectives que nous jugeons intéressantes.

Chapitre 1

Bâtiment intelligent et réseaux de capteurs sans fils

1.1 Introduction

La notion de confort dans l'habitat est apparue dans les années 80 grâce à l'introduction des appareils électroménagers et l'automatisation de certaines tâches ménagères. En effet, la domotique vient du mot latin "Domus" qui signifie Maison et "-tique" désignant l'automatique. Avec l'introduction des appareils mobiles communicants vers la fin du $20^{ème}$ siècle un nouveau concept est né : l'intelligence ambiante. Jean-Baptiste Waldner définit l'intelligence ambiante comme : Un environnement numérique qui, de manière proactive, mais sensible, soutient les gens dans leur vie quotidienne.

Grâce aux progrès fait dans le domaine de la miniaturisation des systèmes micro-électro-mécaniques et dans le marché des réseaux et des applications sans fil mais aussi grâce à l'augmentation de la pénétration d'internet et des réseaux mobiles dans le monde et du nombre d'utilisateurs qui ne cessent d'accroître (Voir figure 1.1 et 1.2), de nouveaux concepts ont émergé tels que les VANETs (Vehicular Ad hoc NETworks), Bâtiment Intelligent et Smart Grids.

Ce chapitre est consacré au bâtiment intelligent et à l'infrastructure réseau lié à celui-ci. Nous présentons les réseaux de capteurs sans fils, leurs caractéristiques, leurs applications et les défis qu'ils présentent. Enfin, nous discutons des différents facteurs qui ont une influence sur les performances des RCSFs.

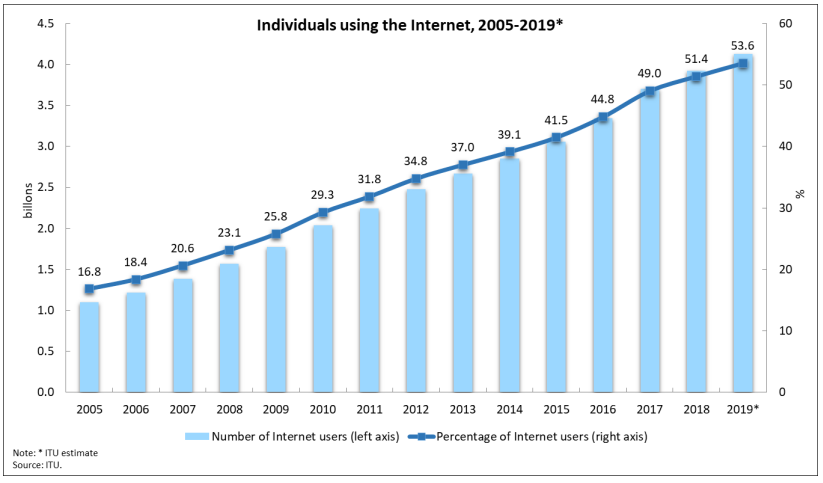

FIGURE 1.1: Nombre d'individus utilisant internet dans le monde.

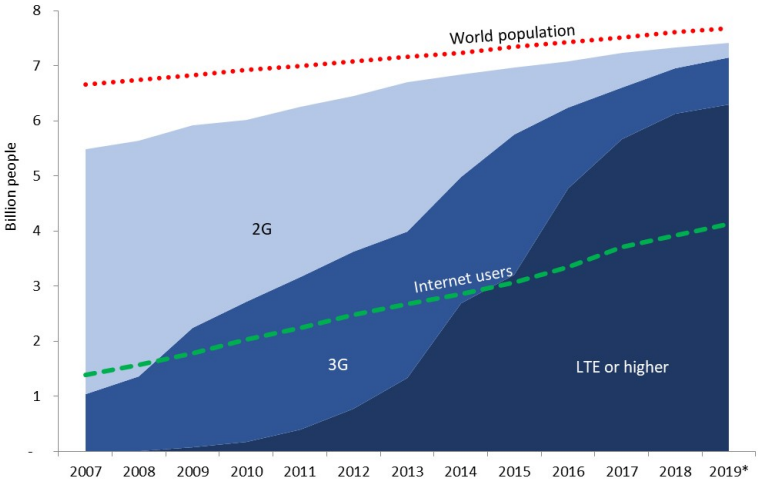

FIGURE 1.2: Couverture réseau dans le monde.

1.2 Bâtiment intelligent

1.2.1 Qu'est ce qu'un bâtiment intelligent

Un bâtiment intelligent se définit comme un bâtiment utilisant la technologie et les processus pour devenir plus efficace sur le plan opérationnel, assurer le confort et la sécurité de ses occupants, améliorer la productivité des employés et réduire son impact sur l'environnement [5]. Il repose sur l'utilisation des capteurs et des actionneurs. Les capteurs sont employés dans le but de récupérer un gros volume de données : température, pression, niveau sonore, détection de présence physique, etc[53]. Les actionneurs quant à eux permettent d'exécuter des commandes sur les équipements électriques auxquels ils sont connectés. Les installations d'un bâtiment intelligent fonctionnent grâce à un système de gestion technique de bâtiment ou GTB, qui leur permettent de communiquer entre elles [16].

1.2.2 Concept d'un bâtiment intelligent

Le bâtiment intelligent peut être vu comme une problématique à 3 dimensions (voir figure 1.3). En premier lieu nous avons les éléments constitutifs du bâtiment : électricité, éclairage, chauffage, ventilation, air conditionné, contrôle d'accès... Ensuite, il y a une première couche d'intégration de ces systèmes permettant le suivi, la gestion sur site et à distance des interventions. Enfin, la troisième dimension est

une intégration avec les utilisateurs du bâtiment, au travers par exemple de la gestion des salles de réunion et des espaces de travail grâce notamment à des applications mobiles mises au service des occupants du bâtiments [5].

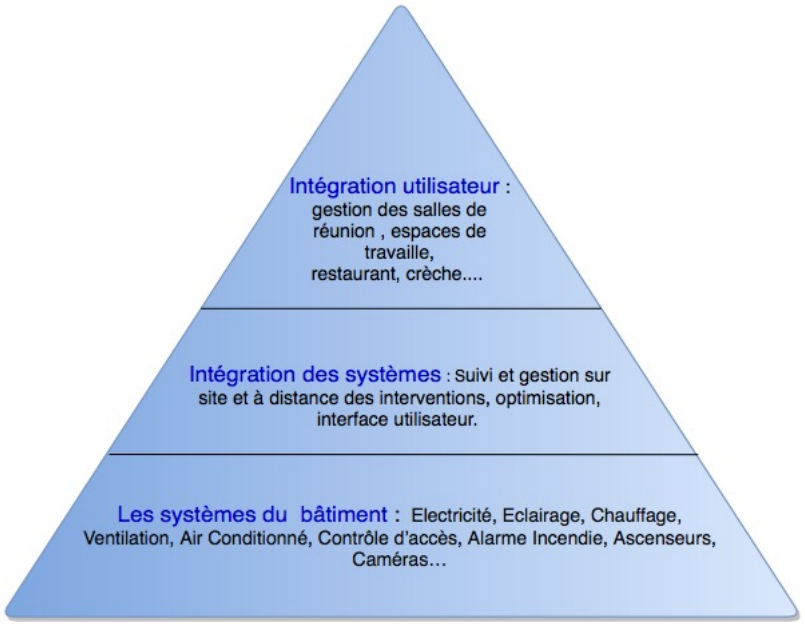

FIGURE 1.3: Concept du smart building [5].

1.2.3 Objectifs principaux d'un bâtiment intelligent

Le smart building consiste en la conception d'une maison ou d'un bâtiment où le fonctionnement du plus grand nombre d'équipements électriques possible est à la fois automatisé et coordonné. Les objectifs principaux d'un bâtiment intelligent sont les suivants [5] :

- Réaliser des économies d'énergie : selon un rapport du World Business Council for Sustainable Development (WBCSD), les bâtiments consomment 40% de l'énergie produite dans le monde. La commission européenne a publié une étude démontrant qu'il est possible d'économiser 23% de l'énergie nécessaire, en adoptant des concepts de Smart Building.
- Assurer la sécurité des occupants : grâce à l'introduction de capteurs et actionneurs connectés tel que les détecteurs de fumée, ou les dispositifs anti-intrusion.
- Gérer les pannes et les interventions sur les installations : la gestion des pannes et l'intervention sur les équipements génèrent de nombreux coûts. L'objectif du

smart building est de prévenir les éventuelles pannes en adoptant une maintenance prédictive.

- Optimiser le confort dans le bâtiment : de nombreuses études ont montré que la qualité de l'air dans un bâtiment a une influence direct sur le bien-être au bureau, le taux d'absentéisme ainsi que sur le rendement des employés. Le Syndrome du Bâtiment Malsain (SBM) recouvre un ensemble de symptômes et de maladies liées à la qualité de l'air intérieur. Il a également été démontré une relation directe entre la concentration et le taux de CO2 intérieur. Les bâtiments intelligents visent à améliorer les conditions en leur sein afin de palier à ces problèmes.

1.2.4 Exemples de bâtiments intelligents

Ces dernières décennies, de nombreux bâtiments intelligents ont vu le jour à travers le monde. Certains sont fonctionnels [34] [50], d'autres sont conçus et équipés pour servir comme banc d'essai pour la recherche scientifique. Ces structures sont appelées Living Labs [7] [12]. Ci-après quelques exemples de Smart Buildings qui ont révolutionné le monde :

Mansion ZCB, Hong Kong : c'est le premier bâtiment "carbone zéro" ; il est doté de systèmes de contrôle intelligents qui peuvent réduire les besoins en énergie de 25 %. Le bâtiment produit toute l'énergie nécessaire à son fonctionnement (voir figure 1.4). Il dispose d'un système qui affiche des données en temps réel et qui mesure la performance environnementale du bâtiment. Ce système fournit des informations sur la consommation générale d'énergie, l'utilisation de l'eau, l'occupation des locaux, la qualité de l'air à l'intérieur du bâtiment ... Ceux-ci sont contrôlés par le GTB qui collecte et traite les données de 2 800 points de détection répartis dans tout le bâtiment.

FIGURE 1.4: Extérieur du bâtiment Mansion ZBC.

The Edge - Amsterdam : Le bâtiment a été construit en 2014, il dispose des dernières innovations en matière de développement durable et des technologies de l'information (voir figure 1.5). La structure est dotée de capteurs et d'objets connectés. Une application mobile permet aux utilisateur d'interagir avec les différents systèmes du bâtiment. Les occupants peuvent notamment réserver des espaces de travail selon la disponibilité de ceux-ci, régler la luminosité et la température selon leurs préférences, etc. Le système d'exploitation étant entièrement connecté permet une gestion efficace de la maintenance et réduit considérablement son coût. The Edge n'est pas seulement le bâtiment le plus intelligent au monde, il est aussi le plus écologique. Il s'est vu attribué par l'agence de notation et de certification BREEAMl le score de durabilité le plus élevé jamais accordé : 98.4%.

FIGURE 1.5: Façade du bâtiment The Edge. Photo de Ronald Tilleman

L'appartement GERHOME : Cet appartement a été réalisé dans le but de mener des expérimentations dans la thématique du maintien à domicile des personnes

handicapés ou à mobilité réduite (voir figure 1.6). Les projets menés dans cet appartement touchent plusieurs domaines qui sont : la santé, l'informatique et l'électronique. Le bâtiment intègre une multitude de capteurs, tel que les capteurs de présence. En cas de chute de la personne, le système avertit, instantanément l'administrateur qui peut intervenir à n'importe quel moment [7].

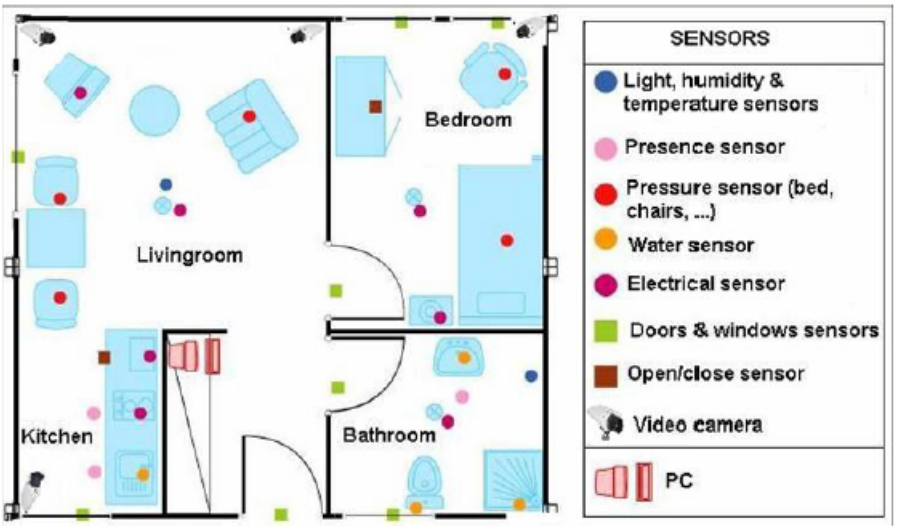

FIGURE 1.6: Architecture et infrastructure réseaux de l'appartement GERHOME

Le bâtiment Adream : Le bâtiment Adream a été construit en 2010 à l'université de Toulouse (voir figure 1.7). Le projet a été réalisé dans le but de mettre en oeuvre une méthodologie ainsi que les technologies nécessaires aux traitements des problèmes d'hétérogénéité, d'interopérabilité des systèmes, de sécurité, et surtout d'autonomie et de gestion de l'énergie dans le bâtiment. Le bâtiment s'étend sur une surface de $1700m^2$ et dispose de plusieurs panneaux photovoltaïque d'une superficie de $720m^2$ qui couvrent toute la toiture et la façade. Plusieurs nouvelles technologies ont été intégrées au bâtiment telles que les capteurs de : présence, luminosité, consommation d'énergie, etc [12].

FIGURE 1.7: Façade et toit photovoltaïque du bâtiment Adream

1.3 Architecture d'un réseau de capteurs sans fils

1.3.1 Noeuds d'un RCSF

Un RCSF est composé d'un grand nombre de noeuds déployés dans une région d'intérêt, ajoutés à ceux-la, des noeuds relais ou sink ainsi qu'une ou plusieurs stations de base (voir figure 1.8) [24].

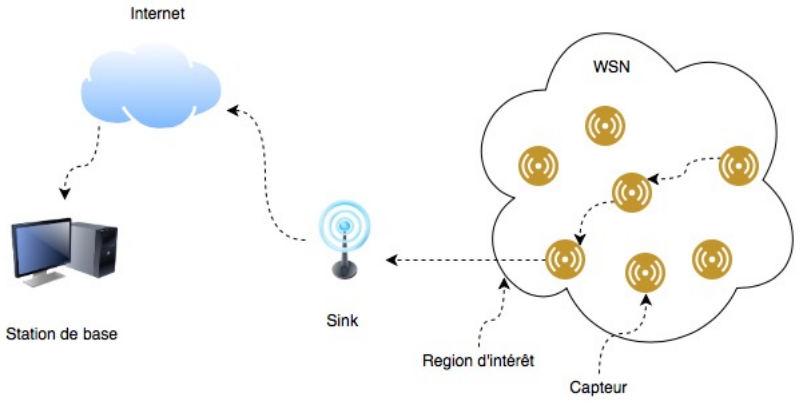

FIGURE 1.8: Architecture d'un réseau de capteurs sans fil

Station de base : c'est un point centralisé de contrôle dans le réseau. Elle est employée pour extraire des informations du réseau et envoyer des informations de contrôle aux différents noeud du réseau.

Sink : aussi appelé noeud puit ou routeur, le sink sert de passerelle entre le champ de captage et la station de base. Ce dispositif est également un capteur mais qui peut

recevoir, traiter et enregistrer des données provenant des noeuds capteurs. Il détient des capacités supérieures en termes de puissances de traitement, capacité de mémoire et autonomie d'énergie.

Noeud capteur : les noeuds capteurs recueillent les informations liées à leur environnement de déploiement. Il existe deux types de noeud capteur : les capteurs fixes et les capteurs mobiles. Un noeud capteur est généralement composé des cinq modules suivants :

- Module d'acquisition des données ;
- Module de traitement ;
- Module de stockage ;
- Module de communication ;
- Module d'énergie.

1.3.2 Topologie des réseaux

Un RCSF peut être organisé selon différents types de topologie dont voici les principales. La figure 1.9 illustre ces différentes topologies.

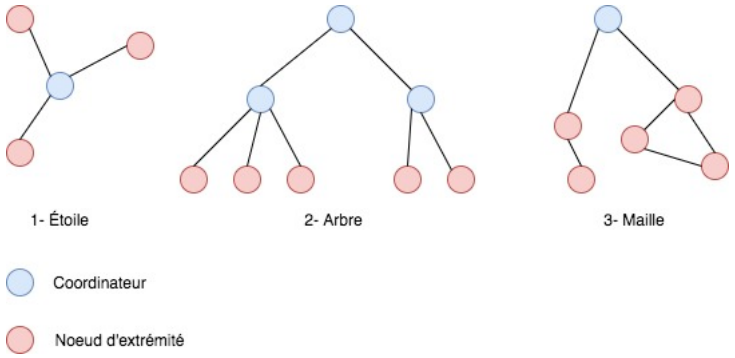

FIGURE 1.9: Topologies des réseaux.

1.3.2.1 Étoile (Star)

Le réseau est constitué d'un ensemble de noeuds directement reliés à un coordinateur central. Les noeuds ne sont pas autorisés à interagir entre eux et communiquent via le coordinateur central [35]. Cette topologie a pour avantages d'être facile à déployer, et de garantir des communications à faible latence entre les noeuds distants et la station

de base [36]. L'inconvénient de cette topologie est que si le coordinateur central est est défectueux, tout le réseau tombe en panne.

1.3.2.2 Arbre (Tree)

Le réseau contient un nœud racine, de nombreux routeurs et des périphériques d'extrémité. Tous les nœuds sont liés sous forme d'arborescence. Les nœuds d'extrémité sont directement liés au coordinateur et aux routeurs en tant que nœuds enfants. Un terminal ne peut interagir avec un autre terminal que via son nœud parent. L'inconvénient de la topologie arborescente et que si l'un des parents devient désactivé, les enfants de celui-ci ne peuvent pas interagir avec d'autres périphériques du réseau [35].

1.3.2.3 Maille (Mesh)

Une topologie maillée est autogérée, c'est-à-dire que pendant la transmission, si l'un des chemins échoue, le nœud découvrira un chemin alternatif vers le nœud de destination. Tout appareil source peut interagir avec n'importe quel appareil de destination du réseau [35]. L'inconvénient de cette topologie et que l'augmentation du nombre de sauts d'une communication à une autre engendre une augmentation de la consommation d'énergie dans le réseau. Son avantage principal est qu'elle permet de maintenir la connectivité globale du réseau [10].

1.3.2.4 Topologie hybride

La topologie hybride [43] conjugue la topologie en étoile et la topologie maillée. Elle fournit un réseau robuste et minimise la consommation d'énergie des capteurs. Les noeuds ayant une capacité énergétique plus élevée assurent une communication multi saut. En revanche, ceux à faible puissance sont désactivés.

1.4 Standards de communication

1.4.1 ZIGBEE

Zigbee est un standard utilisé dans les communications à très faible puissance et sur des distances réduites notamment dans les réseaux de capteurs sans fils. Il prévoit des vitesses allant jusqu'à 250 kbps sur une plage de 10 à 100 m et utilise la bande de fréquence 2,4 GHz. Il repose sur les couches basses du standard IEEE 802.15.4. Le protocole Zigbee peut effectuer les opérations suivantes : Découverte du voisinage, création de la topologie, adressage et routage [22].

1.4.2 Wi-Fi

Wi-Fi est un ensemble de protocoles de communication sans fil régis par les normes IEEE 802.11. Les normes Wi-Fi, permettent de créer des réseaux locaux sans fils à haut débit. Pour les réseaux de capteurs sans fils une nouvelle version de Wi-Fi appellée Wi-Fi à faible puissance "Wi-Fi low power" est utilisée. Elle est caractérisée par une faible consommation d'énergie [12].

1.4.3 UWB

Ce protocole peut procurer un débit de 55 Mbps pour une distance allant jusqu'à 100 mètres. Il utilise la bande de 2,4 GHz ce qui élimine le risque d'interférences avec les autres types de réseaux. La norme IEEE 802.15.3 est caractérisée par une bonne qualité de service étant donné qu'elle inclut le protocole TDMA (Time Division Multiple Access). Le standard UWB bénéficie de très bonnes performances en termes de sécurité [38].

1.4.4 Bluetooth

Bluetooth utilise des ondes radios UHF destinées à simplifier les connexions entre les appareils électroniques. Cette norme a été proposée pour transmettre la voix et les données. Elle est peu utilisée dans les réseaux de capteurs sans fils parce que très gourmande en énergie et dispose d'une topologie réseau complexe [30].

1.4.5 Wibree

Wibree, plus connu sous Bluetooth Low Energy (BLE), est une technique de transmission sans fil créée par Nokia. Ce standard est basé sur Bluetooth. Wibree consomme 10 fois moins d'énergie que Bleutooth pour un même débit. Cela rend possible l'utilisation de cette technique dans des équipements à faible puissance comme les capteurs sans fils. Sa limite principale est la faible portée de communication : 5-10m [48].

1.5 Domaines d'application des RCSFs

Les RCSFs ont connu un grand succès dans plusieurs domaines (voir figure 1.10) et ce grâce au rétrécissement de la taille des capteurs, leur facilité d'adaptation à n'importe quel milieu et enfin la communication sans fil. Parmi les domaines où les RCSFs sont très utilisés nous pouvons citer :

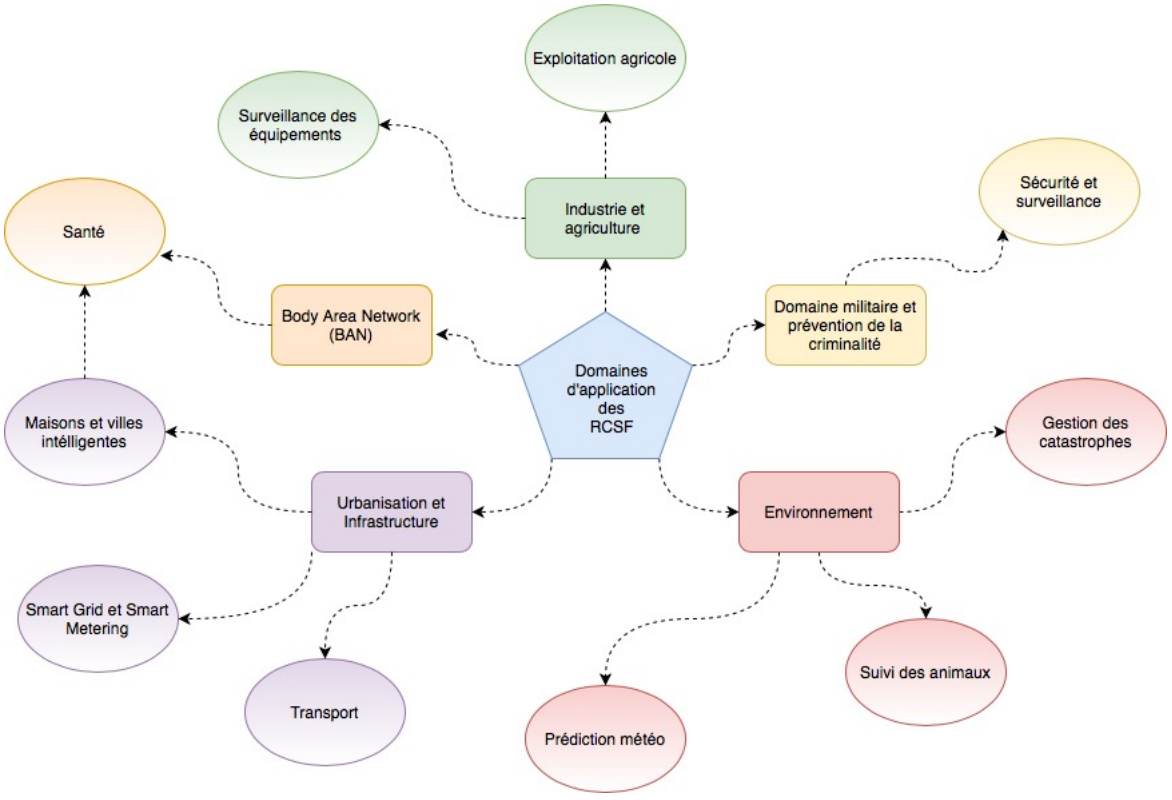

FIGURE 1.10: Taxonomie des applications des RCSFs.

1.5.1 Militaire

Les RCSFs ont été initialement développé pour des application militaires. Les RCSFs offrent plusieurs avantages tels que : le déploiement rapide et l'auto-organisation [6]. Ils sont utilisés pour les applications de surveillance (la surveillance des champs de batailles, la surveillance des munitions et des équipements) , reconnaissance des forces ennemies et le ciblage.

1.5.2 Environnement

On peut classer les applications des RCSFs dans le domaine de l'environnement selon trois axes principaux. L'agriculture de précision qui consiste en la surveillance des récoltes et du bétail et les systèmes d'irrigation intelligents. La gestion des catastrophes naturelles grâce à la détection des incidents naturels (incendies et inondations). Les prévisions météo, l'étude du climat et de l'empreinte carbone [8].

1.5.3 Santé

Avec la miniaturisation de la taille des capteurs , ces derniers se sont révélés très utiles dans le domaine de récolte d'informations de biométrie. Une des applications des RCSFs est le suivi des patients atteints d'une maladie dans le but de traitement ou de contrôle (mesure de glycémie, taux de cholestérol, détection des cancers ...) [11]. Les RCSFs sont également utilisés pour l'aide des personnes handicapés. On peut citer les travaux de C. Jacquet et al. [33] qui ont proposé un système d'aide au déplacement qui grâce à des capteurs intégrés à sa canne, le non-voyant est guidé sur tout le long de son trajet.

1.5.4 Bâtiment intelligent

Les RCSFs sont utilisés pour la collecte des données liées à l'environnement du bâtiment, ainsi que les informations sur les phénomènes physiques qui s'y produisent. Les RCSFs permettent une gestion intelligente des maisons ou bâtiments dans lesquelles ils sont déployés. Ils améliorent ainsi les performances énergétiques des bâtiments et économisent leurs ressources [11]. Les RCSFs sont également utilisés

dans des applications de surveillance, contrôle d'accès et le monitoring de l'occupation des espaces.

1.6 Défis posés par les RCSFs

Comme nous l'avons vu dans la section précédente, les RCSFs sont utilisés dans plusieurs domaines et pour diverses applications. Les capacités limitées des capteurs en terme de stockage d'énergie, traitement des données, et coût des communications font que les RCSFs sont sujets à de nombreux défis (voir figure 1.11). Beaucoup de recherches ont été conduite dans le but de pallier à ces limites. Notamment en ce qui concerne la localisation des capteurs [37], le stockage des données [59], le routage des données des capteurs vers le sink, la consommation d'énergie, la fiabilité des réseaux ainsi que leur évolutivité, les interférences et le déploiement des capteurs [23].

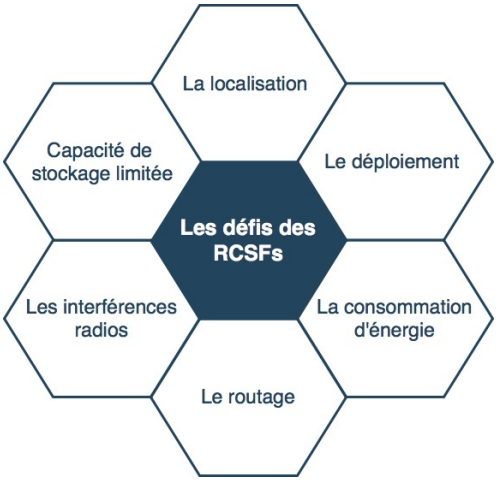

FIGURE 1.11: Défis des réseaux de capteurs sans fils.

Le déploiement des capteurs consiste à placer des noeuds capteurs dans la région d'intérêt de manière à avoir une couverture totale de celle-ci, tout en assurant une connectivité avec le ou les noeuds relais (sinks). Elle est considérée comme l'un des problèmes les plus critiques dans la mise en œuvre des RCSFs. La couverture [28] est parmi les problèmes les plus importants pour la réalisation d'un RCSF. Elle est affectée essentiellement par le rayon de détection (Rd) du noeud capteur. D'autre part, la connectivité [60] dépend de la portée de communication (rayon de communication : Rc) des nœuds capteurs et de leur disposition dans l'espace. La consommation d'énergie

est un facteur important qui doit être pris en compte. La planification des modes actif et veille des nœuds de capteurs après le déploiement optimal des nœuds minimise la consommation d'énergie et prolonge la durée de vie du réseau [23]. Chacun de ces défis sera amplement détaillé dans le chapitre suivant.

1.7 Mesure de performance des RCSFs

De nombreuses mesures de performances doivent être prises en compte dans les RCSFs. Les facteurs qui ont une influence sur la qualité de service (QoS) d'un RCSF sont la consommation d'énergie, l'évolutivité, la fiabilité, la puissance du signal, la précision, la latence, la couverture, et la connectivité. Cette liste est bien évidemment non exhaustive. Ci-dessous une brève discussion de chaque métrique[23] :

1.7.1 Consommation d'énergie

La consommation d'énergie est un facteur important à prendre en compte lors de l'implémentation d'algorithmes dans les WSNs. Diminuer la consommation d'énergie par nœud peut être obtenue en réduisant le nombre d'échanger de messages entre les nœuds. En outre, planifier des intervalles de sommeil pour les nœuds redondants, tout en laissant les nœuds restants actifs pour maintenir la couverture du réseau et la connectivité, augmente la durée de vie du réseau. Outre la diminution de la taille des messages transmis entre les nœuds, la sélection de la meilleure méthode de routage et la réduction de la mobilité des nœuds réduisent la consommation d'énergie dans les RCSFs.

1.7.2 Évolutivité et Fiabilité

L'évolutivité est la capacité du réseau à être étendue par ajout de nœuds tout en maintenant les performances du réseau. La fiabilité représente la livraison des données. L'évolutivité et la fiabilité sont des problèmes critiques dans les RCSFs en raison du très haut nombre de nœuds.

1.7.3 Puissance du signal

La puissance du signal est une mesure de la qualité de la liaison. On utilise le distance entre deux nœuds pour déterminer l'accessibilité des nœuds pendant le processus de communication. L'indication de la puissance du signal reçu (RSSI) décrit la force d'un signal sans fil et est définie comme suit :

$$RSSI = -10 \cdot n \cdot log_{10}(d) + p \tag{1.1}$$

où d est la distance en mètre, n est la constante de propagation et p est la puissance en mode réception (dBm).

1.7.4 Précision et latence

La latence est une mesure du retard. Elle mesure le temps qu'il faut pour une donnée pour arriver à la destination sur le réseau. La précision représente l'efficacité des données livrées à la destination. La réduction du retard grâce à la transmission des données garantit la précision du réseau.

1.7.5 Couverture et connectivité

La couverture est l'une des mesures du QoS des RCSFs. La couverture sans connectivité complète diminue la qualité du RCSF car, une défaillance au niveau de la connectivité influe directement sur la réception des données par le noeud récepteur. De plus, la connectivité sans couverture provoque des points non couverts dans la zone cible et des trous de couverture apparaissent. Par conséquent, la couverture et la connectivité doivent être prises en compte simultanément lors du déploiement d'un RCSF.

1.8 Conclusion

Dans ce premier chapitre, nous avons défini le bâtiment intelligent et identifié les infrastructures réseaux nécessaires pour son bon fonctionnement. Nous avons également abordé la notion d'un RCSF ainsi que les différentes parties qui le composent. Nous avons aussi présenté les différentes applications qui se basent sur cette technologie.

Enfin, nous avons exposé les défis posés par cette technologie, qui reste malgré son avancée un terrain riche en terme de recherches scientifiques.

Nous allons aborder dans le prochain chapitre les différents types de modélisation du problème de déploiement des RCSFs. Nous exposerons par la suite les modèles retenus qui nous seront utiles lors de la phase de la résolution et d'optimisation du problème de déploiement des capteurs au sein d'un bâtiment intelligent.

Chapitre 2

Déploiment d'un RCSF : modélisation de la problématique

2.1 Introduction

Le but de notre travail est d'optimiser le déploiement d'un réseau de capteur sans fil dans un bâtiment intelligent. Pour ce faire il est nécessaire de modéliser les différents paramètres qui ont une influence sur le coût ainsi que sur la qualité de service du réseau de capteur.

Nous allons donc dans ce chapitre modéliser l'espace de déploiement, le coût de déploiement ainsi les différents objectifs à optimiser qui sont (La couverture, la sur-couverture, la connectivité et la durée de vie)

Étant donnée que le RCSF sera déployé dans un environnement indoor (bâtiment) nous avons opté pour un placement statique des noeuds. Ceci étant la stratégie la plus adoptée dans ce type d'application [56].

2.2 Critères de déploiement

2.2.1 Espace de déploiement

Les capteurs sans fils peuvent être placés dans différents endroits (murs, plafond, meubles ...) au sein d'un bâtiment. L'espace de déploiement peut donc être modélisé en 3D ou bien en 2D. Nous considérons dans notre travail un espace de déploiement en 2D (le plafond du bâtiment). L est la longueur du bâtiment intelligent et l sa largeur. Nous modélisons l'espace de déploiement avec une grille de L*l cellules de $1m^2$. Les capteurs seront placés au centre des cellules.

2.2.2 Coût de déploiement

L'objectif principal de notre travail est de réduire au maximum le coût du RCSF à déployer tout en maximisant sa qualité de service. Il est donc très important de modéliser le coût de déploiement afin de l'intégrer dans notre fonction objectif. Le coût de déploiement comprend le coût d'achat des capteurs, celui des routeurs et enfin leur coût d'installation.

Si on considère un RCSF composé de l'ensemble des noeuds $R = r_1, r_2, r_3, \ldots, r_n$ routeurs et l'ensemble des noeuds $S = s_1, s_2, s_3, \ldots, s_m$ capteurs. Le coût de déploiement peut être calculé comme suit :

$$Coût = C_s \cdot \|S\| + C_r \cdot \|R\|. \tag{2.1}$$

Avec

C_s : Coût d'achat et d'installation d'un noeud capteur ;

C_r : Coût d'achat et d'installation d'un noeud routeur ;

$\|S\|$: Le nombre de noeuds capteurs déployés ;

$\|R\|$: Le nombre de noeuds routeurs déployés.

En se basant sur la modélisation de l'espace de déploiement nous pouvons définir la matrice de déploiement comme suit :

$$D(i,j) = \begin{cases} 1 & \text{si un capteur ou un relais est déployé dans la cellule ;} \\ 0 & \text{sinon.} \end{cases} \tag{2.2}$$

La fonction coût peut être obtenue à partir de la matrice de déploiement comme suit :

$$Coût = \sum_{i=0}^{l-1} \sum_{j=0}^{L-1} D(i,j) \tag{2.3}$$

Il s'agit d'un coût unitaire qui sera ensuite multiplié par le coût des capteurs pour obtenir le coût de déploiement des capteurs. L'opération sera répétée en ce qui concerne les routeurs pour ainsi avoir le coût global du déploiement du réseau.

2.2.3 Couverture

On dit qu'une région ou qu'un point est couvert par un capteur si cette région ou ce point se trouve dans la zone de couverture d'un ou plusieurs capteurs actifs. Selon Harizan et al. [31], il existe trois types de couverture :

2.2.3.1 Couverture de champ

L'objectif de ce type de couverture est que chaque point de la région d'intérêt soit dans le champ de couverture d'au moins un capteur. Comme on peut le voir dans la figure 2.1, la région d'intérêt (le carré) est complètement couverte par les capteurs déployés. En fonction de l'application, il existe deux types de couverture de champ : couverture complète et couverture partielle.

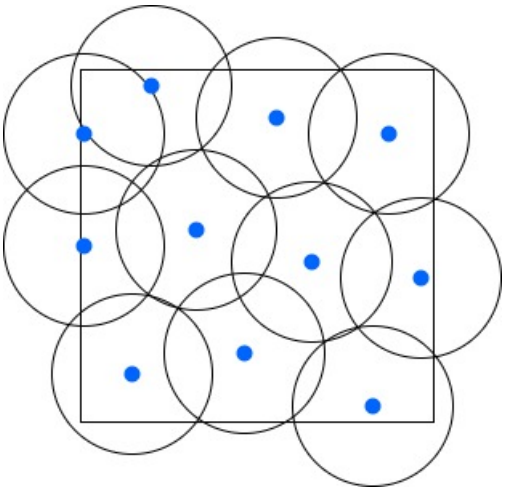

FIGURE 2.1: Couverture complète de la région d'intérêt.

Dans le cas d'une couverture partielle, les capteurs sont déployés de manière à couvrir un pourcentage p de la surface totale. Adopter une couverture partielle permet de réduire le coût de déploiement et d'augmenter la durée de vie du réseau de capteurs [31]. La couverture partielle est utilisée dans des applications liées à l'environnement, tel que le relevé de température dans une région donnée et la détection des feux de forêts.

La couverture complète est obtenue quand le taux de couverture est égale à 1. Le coût de déploiement est plus élevé que dans le cas d'une couverture partielle puisque le nombre de capteurs nécessaires est plus important. Le degré de couverture dépend du type d'application. Certaines applications nécessitent une couverture simple où chaque point est couvert par au moins un capteur. Pour des applications qui nécessitent un haut degré de précision, une k-couverture est mise en place. La k-couverture implique la couverture d'un point par au moins k capteurs avec k>1.

2.2.3.2 Couverture de cible (Point d'intérêt)

Lorsqu'il s'agit de surveiller des points spécifiques d'une région, on a recours à une couverture de cible. Comme on peut le voir dans la figure 2.2, les cibles représentées par des hexagones noirs se trouvent dans le rayon de couverture d'au moins un capteur. Le nombre de cibles étant fixe, le coût de déploiement du réseau se voit réduit puisqu'il s'agit d'une couverture partielle qui ne nécessite pas un nombre élevé de capteurs.

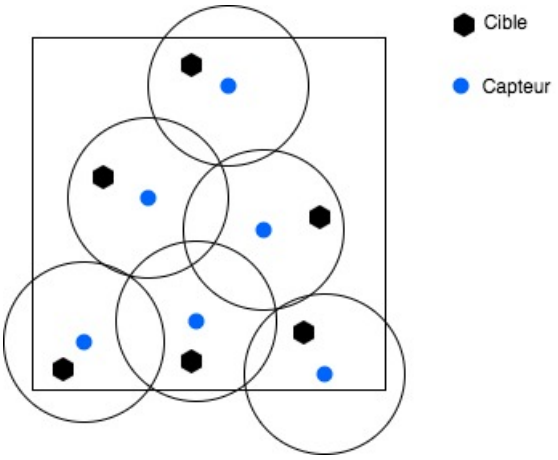

FIGURE 2.2: Couverture des point d'intérêts (cibles).

2.2.3.3 Couverture de barrière

Dans un schéma de couverture de type barrière les noeuds sont déployés sous forme de barrière sur un chemin spécifique. Ces noeuds signalent d'éventuels activités suspectes. La couverture de barrière convient à la détection d'intrusions, aux applications militaires et à la surveillance des frontières. En fonction des applications on distingue deux types de régions d'intérêts : ROI en forme de ceinture ouverte, ROI en forme de ceinture fermée [55]. L'objectif de ce type de couverture est de minimiser la probabilité d'intrusion à travers la barrière.

2.2.4 La sur-couverture

Afin de réduire le coût de déploiement du réseau de capteurs sans fil et de diminuer les interférences entre les capteurs, il est nécessaire de réduire la surface des zones dites sur-couvertes, c'est à dire les zones couvertes par deux ou plusieurs capteurs comme on peut le voir dans la figure 2.3 . Pour ce faire, nous allons utilisé une des

représentations existantes de la sur-couverture, définit comme suit :

$$Sur{-}couv(i,j) = \begin{cases} 1 & \text{si la cellule (i,j) est couverte par au moins deux capteurs ;} \\ 0 & \text{sinon.} \end{cases} \tag{2.4}$$

Nous implémentons par la suite une fonction qui calcule le taux de sur-couverture à l'aide de la formule suivante :

$$Sur - couv_{RI} = \frac{\sum_{i=0}^{l-1} \sum_{j=0}^{L-1} Sur - couv(i,j)}{L \cdot l} \cdot 100 \tag{2.5}$$

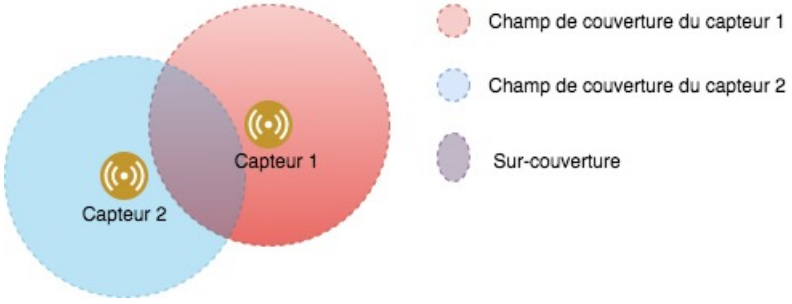

FIGURE 2.3: Modélisation de la zone de surcouverture.

2.2.5 Détection des événements

La détection des événements et la collecte des données sont parmi les principaux objectifs des réseaux de capteurs sans fils. Plus on s'éloigne du capteur plus sa capacité de détecter les phénomènes diminue. On retrouve dans la littérature deux modèles de détection [4] : les modèles de détection déterministes et les modèles de détection probabilistes. Ces modèles sont basés sur la probabilité de détection [31].

2.2.5.1 Modèle de détection déterministe (binaire)

Le modèle le plus utilisé dans la littérature est le modèle de détection binaire. Il a l'avantage d'être rapide, simple à implémenter et il a un faible coût de calcul. Il est aussi appelé modèle 0-1. Si un événement se produit dans le Rd du capteur alors il est détecté à 100% (probabilité de détection = 1). Par contre, si celui-ci se produit à une distance supérieure au rayon Rd, la probabilité de le détecter est nulle (Pd =

0). La figure 2.4 illustre ce modèle de détection d'événement. D'autre modèles plus réalistes intègrent le fait que la probabilité de détection d'un capteur diminue avec l'accroissement de la distance. Il s'agit des modèles de détection probabilistes que nous allons exposer dans la section suivante.

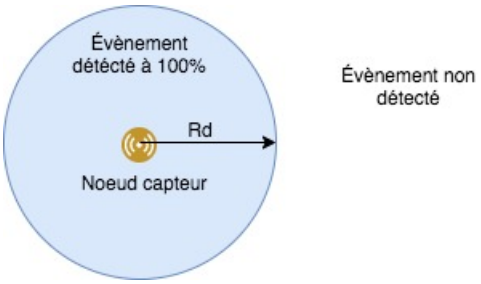

FIGURE 2.4: Modèle de détection binaire.

2.2.5.2 Modèle de détection probabiliste

Les modèles asymptotiques dépendent de la distance. En effet, lorsqu'un évènement se produit près du capteur, il a une plus grande probabilité d'être détecté. Cette probabilité diminue en s'éloignant du capteur. On distingue deux fonctions pour le calcul de la probabilité de détection : exponentielle, polynomiale.

Modèle exponentiel : Dans ce modèle, la probabilité de détection d'un événement diminue exponentiellement par rapport à la distance. La probabilité de détection est donnée par la fonction suivante :

$$P_s^p = \exp(-\alpha d(s,p)) \tag{2.6}$$

Où α représente la qualité de détection du capteur s et d(s,p) la distance euclidienne entre le capteur s et la position p [61].

Modèle polynomial : Dans ce modèle, la probabilité de détection d'un évènement se dégrade de manière polynomiale d'ordre (-k). Elle est donné par la fonction suivante :

$$P_s^p = \lambda d(s,p)^{-k} \tag{2.7}$$

où k représente la qualité de détection du capteur et λ est lié aux configurations matérielles de celui-ci.

Une combinaison du modèle de détection binaire et du modèle de détection asymptotique est apparue. Ce nouveau modèle considère deux zones de détection : une zone de confiance et un cercle maximal. Comme on peut le voir dans la figure 2.5, on distingue 3 zones :

Zone (1) : appelée zone de confiance. Si l'événement se produit dans cette zone, il est détecté avec une probabilité égale à 1.

Zone (2) : située entre la zone de confiance et la limite maximale de détection. Un événement est détecté avec une probabilité Pp (qui peut être une fonction exponentielle ou polynomiale). Cette probabilité diminue avec la distance.

Zone (3) : située en dehors du champ de détection du capteur , au delà de la limite maximale. Si un événement se produit dans cette région, il ne sera pas détecté.

Ce modèle est représenté par les équations suivantes :

$$P_p = \begin{cases} 1 & \text{si } d(s,p) \leqslant R_{min} \\ \exp(-\alpha d(s,p)) & \text{si } R_{min} \leqslant d(s,p) \leqslant R_{max} \\ 0 & \text{si } d(s,p) \geqslant R_{max} \end{cases} \tag{2.8}$$

$$P_p = \begin{cases} 1 & \text{si } d(s,p) \leqslant R_{min} \\ \frac{\lambda}{d(s,p)^k} & \text{si } R_{min} \leqslant d(s,p) \leqslant R_{max} \\ 0 & \text{si } d(s,p) \geqslant R_{max} \end{cases} \tag{2.9}$$

Ou R_{min} désigne le rayon de la zone de confiance, R_{max} la limite maximale[61].

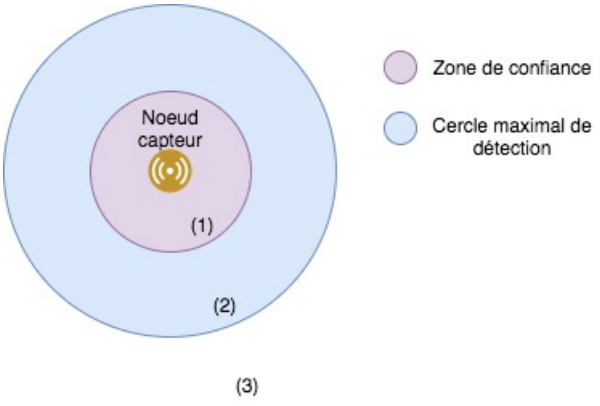

FIGURE 2.5: Modèle de détection probabiliste.

Autres modèles de détection probabilistes : Ces modèles sont affectés par plusieurs facteurs tels que le bruit, les obstacles et les interférences. Kumar et al. dans leur article "Sensing coverage prediction for wireless sensor networks in shadowed and multipath environment" publié en 2013 [41] ont exposé quelques modèles probabilistes, dont les suivants :

Modèle de détection d'ombre : Le modèle de détection d'ombre prend en considération les obstacles. La capacité de détection du capteur n'est pas uniforme dans toute les directions. Dans un modèle log normale la probabilité de détecter un évènement qui se produit à une distance x du capteur est donnée par l'équation suivante :

$$P_{det}(x) = Q\left(\frac{10n\log_{10}(x/r_s)}{\sigma}\right) \tag{2.10}$$

où n représente l'exposant d'affaiblissement, r_s représente le rayon de détection sans évanouissement et σ représente le paramètre d'évanouissement.

Modèle d'évanouissement de Nakagami : Ce modèle est utilisé dans les canaux de propagation sans fils. Il convient à la modélisation de l'évanouissement à petite échelle dans les canaux sans fils longue distance.

2.2.5.3 Modèle retenu

En utilisant le modèle de détection binaire et dans le but d'assurer une couverture complète de la région d'intérêt (le bâtiment dans notre cas) la matrice de couverture peut être définie comme suit :

$$Couv(i,j) = \begin{cases} 1 & \text{si la cellule (i,j) est couverte par au moins un capteur ;} \\ 0 & \text{sinon.} \end{cases} \tag{2.11}$$

Le taux de couverture peut être calculé selon l'équation suivante :

$$Couv_{RI} = \frac{\sum_{i=0}^{l-1}\sum_{j=0}^{L-1} Couv(i,j)}{L \cdot l} \cdot 100 \tag{2.12}$$

2.2.6 Connectivité

La connectivité est un paramètre essentiel qu'il faut prendre en considération lors du déploiement des RCSFs. Elle permet au capteur de transmettre les données vers la station de base. La transmission des données peut se faire directement entre le capteur et la station de base ou bien à travers d'autres noeuds relais (communication multi-saut). Un réseau est dit connecté si chaque noeud déployé est connecté à au moins un noeud voisin. Si une liaison entre deux noeuds se rompt, cela peut entraîner une défaillance dans la communication. Pour palier à ce genre d'incident, on a recours au concept de la k-connectivité. Chaque capteur est connecté à au moins k autres capteurs (figure 2.6). Ainsi la communication est assurée même si k-1 liens sont défaillants [24].

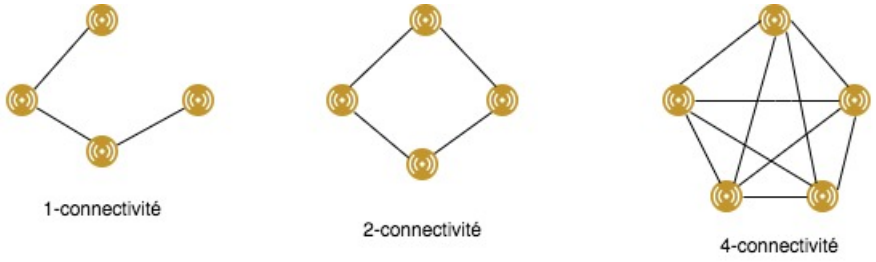

FIGURE 2.6: k-connectivité.

La connectivité d'un réseau est liée à la propagation d'ondes radios. Un noeud A est dit connecté à un autre noeud B si le RSSI du signal émis par le noeud B est supérieur à la sensibilité de l'antenne de réception du noeud A. Il existe plusieurs modèles de propagation radio qui peuvent exprimer le comportement de l'onde transmise par un noeud dans un environnement précis. Ces modèles sont présentés dans les parties suivantes.

2.2.6.1 Modèle de FRIIS

Le modèle de FRIIS [9] est l'un des premiers modèles établies dans la littérature. Ce modèle, aussi appelé équation de transmission de Friis, exprime le rapport entre la puissance du signal reçu et celui émis par deux antennes séparées d'une distance R. Ces antennes sont considérées isotropes. Le rapport de puissance est donné par l'équation suivante :

$$\frac{P_r}{P_t} = n * \left[\left(\frac{\lambda}{4\pi R} \right)^2 * G_t * G_r \right] \tag{2.13}$$

Où P_r et P_t représentent la puissance reçue et la puissance de transmission. G_t et G_r représentent respectivement le gain de transmission et le gain de réception de l'antenne. Le terme $\left(\frac{\lambda}{4\pi R}\right)^2$ est appelé facteur de perte en espace libre avec λ la longueur d'onde et R la distance entre les deux antennes. Enfin n représente les différentes pertes liées à la désadaptation (réflexion et polarisation).

2.2.6.2 Modèle Multi Wall

Le modèle de propagation le plus approprié aux environnements intérieurs est le MWM (Modèle Multi Wall). Il prend en compte les atténuations dues à la pénétration de l'onde entre l'émmetteur et le récepteur dans les obstacles (murs, portes, étages). Il ne considère pas les pertes par réfraction et par diffraction[42]. Le modèle est représenté par l'équation suivante :

$$PL_{MWF}[dB] = FSL + 10 * n * \log_{10}(d) + \sum_{i=1}^{I} \sum_{k=1}^{K_{wi}} (L_{wik}) + \sum_{j=1}^{J} \sum_{k=1}^{K_{fj}} (L_{fjk}) \tag{2.14}$$

$$RSSI[dBm] = P_{TX}[dBm] - PL_{MWF}[dBm] \tag{2.15}$$

Où :

-FSL : désigne Free-Space Losses ;

-L_{wik} : est l'atténuation du $k^{ème}$ mur de la catégorie i ;

-L_{fjk} : est l'atténuation du $k^{ème}$ étage de la catégorie j ;

-K_{wi} : est le nombre des murs de la catégorie i ;

-K_{fj} : est le nombre d'étages de la catégorie j.

Ce modèle est facile à implémenter et nécessite un temps de calcul réduit par rapport à d'autres modèles de propagation.

2.2.6.3 Modèle retenu

Afin de modéliser la connectivité nous allons adopté le modèle le plus simple et le plus utilisé dans la littérature [31]. A savoir le modèle de communication binaire sous forme de disque (voir figure 2.7). Comme nous pouvons le voir dans la figure, le rayon de communication Rc est supérieur au rayon de couverture Rd.

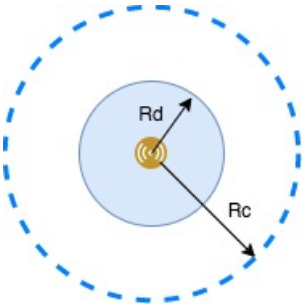

FIGURE 2.7: Rayon de communication d'un capteur.

La matrice de connectivité est définie comme suit :

$$Con(i,j) = \begin{cases} 1 & \text{si la cellule (i,j) est dans le rayon} \\ & \text{de communication d'au moins 1 capteur ;} \\ 0 & \text{sinon.} \end{cases} \tag{2.16}$$

Le taux de connectivité peut être calculé à partir de la matrice pécedente à l'aide de l'équation 2.17 :

$$Con_{RI} = \frac{\sum_i \sum_j Con(i,j)}{L * l} * 100 \tag{2.17}$$

2.2.7 Durée de vie

La durée de vie du réseau dépend principalement de la consommation d'énergie de ce dernier. En effet, les capteurs et les relais étant principalement alimentés par des batteries ont de ce fait une durée de vie limitée. Les composants radio-fréquences sont plus gourmands en énergie que l'unité de traitement des données. La durée de vie sera donc déterminée par la consommation en énergie de l'unité de communication qui dépend de la distance séparant les noeuds et du nombre de paquets à transmettre. Pour éviter les phénomènes de sur écoute, il sera préférable d'opter pour une communication directe entre les capteurs et les routeurs, dans une topologie en étoile [12].

2.3 Conclusion

Dans ce chapitre, nous avons décrit et modélisé tous les critères qui ont une influence sur les performances d'un réseau de capteurs. Nous avons fait une revue de littérature des différentes modélisations existantes et nous avons sélectionné celles qui sont les plus adaptées à notre problématique.

Dans le chapitre suivant, nous allons exposer les différentes méthodes utilisées pour la résolution et l'optimisation du problème de déploiement des RCSFs en environnement intérieur. Le but du chapitre 3 est de déterminer la méthode la plus adéquate qui puisse prédire l'emplacement des noeuds tout en respectant les contraintes de coût, connectivité, couverture et sur-couverture.

Chapitre 3

Stratégies de déploiement des réseaux de capteurs sans fils

3.1 Introduction

Le problème de déploiement d'un réseau de capteurs sans fil dans un bâtiment peut être assimilé au problème de galerie d'art [52]. Ce problème consiste à déterminer une solution pour le placement d'un nombre minimum d'agents de sécurité à l'intérieur d'un musée ou d'une galerie en faisant en sorte qu'ils puissent voir le maximum d'espace possible. Ce problème est classé comme étant NP-difficile. Les problèmes de la classe NP admettent un algorithme de complexité polynomiale pour vérifier si n'importe quelle proposition est solution du problème traité. Ces problèmes ne peuvent être résolus de manière exacte en un temps polynomial par rapport à leur taille.

Afin de déterminer le meilleur moyen de résoudre et d'optimiser le déploiement d'un RCSF nous allons faire un tour d'horizon des méthodes de résolution qui existent dans la littérature. Nous allons également établir un état de l'art concernant le déploiement des RCSFs. Enfin, nous sélectionneront la méthode la plus adaptée que nous mettront en oeuvre dans le dernier chapitre.

3.2 Méthodes de résolution du problème de déploiement des RCSFs

Un problème d'optimisation est défini comme étant une recherche de variables de décision qui minimisent ou maximisent une fonction objectif. Cette dernière comporte des paramètres et est généralement soumise à des contraintes. Plusieurs méthodes ont été développées pour résoudre ce genre de problèmes. Ces méthodes sont classées en deux grandes familles : méthodes déterministes et méthodes stochastiques.

3.2.1 Méthodes d'optimisation déterministes

3.2.1.1 Méthode du gradient

C'est une méthode de recherche linéaire utilisée pour l'optimisation des fonctions réelles, continues et dérivables. L'algorithme part d'une solution initiale x_0. Cette solution est ensuite améliorée en plongeant vers le minimum local le plus proche. Pour

ce faire, il calcule le gradient de la fonction objectif $\nabla f(x_0)$ au point x_0. Il obtient ainsi la direction de la plus forte pente $-\nabla f(x_0)$, sur laquelle il va avancer d'un pas α_k. Le solution est ensuite remplacé par $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$. On réitère les opérations jusqu'à ce que la solution ne puisse plus être améliorée [27].

3.2.1.2 Méthode du simplexe

La méthode du simplexe est aussi appelée méthode de Nelder-Mead [46]. Il s'agit d'une méthode d'optimisation locale. À chaque itération, l'algorithme améliore un ensemble de solutions candidates. Cet ensemble de solutions est appelé "polytope". Ce dernier est une représentation géométrique de $n + 1$ points (solutions), n étant la dimension du problème.

Pour former le polytope de départ, l'algorithme choisit au hasard une solution x_1 dans l'espace de recherche. Ensuite les autres points x_i sont générés à l'aide de la formule suivante :

$$x_{i+1} = x_i + \lambda e_i, \forall i \in [1, n] \tag{3.1}$$

ou les e_i sont des vecteurs linéairement indépendants et λ une constante adaptée à la caractéristique du problème.

À chaque itération les points x_i sont ordonnés de sorte à avoir (en cas de minimisation) $f(x_1) < f(x_2) < ... < f(x_{n+1})$. De nouvelles solutions sont calculées en utilisant des transformations géométriques élémentaires (réflexion, contraction, expansion, rétrécissement). Après chaque itération, le point (solution) donnant la plus mauvaise valeur de la fonction objectif est remplacé par le meilleur des nouveaux points calculés. Le polytope se transforme donc au fil des itérations et converge vers l'optimum. L'algorithme s'arrête lorsque la différence entre la meilleure et la plus mauvaise solution devient inférieur à un certain seuil. L'intérêt de cette méthode réside dans la non nécessité du calcul du gradient. Comme elle peut être utilisé pour des fonctions bruitées puisqu'elle se base uniquement sur l'évaluation de la fonction objectif.

3.2.2 Méthodes d'optimisation stochastiques

3.2.2.1 Monte-Carlo

La méthode de Monte-Carlo a été développée par N. Metropolis et S. Ulam [45]. Elle s'appuie sur la recherche stochastique. L'algorithme génère aléatoirement une solution initiale appartenant à l'espace de recherche et évalue la valeur de la fonction objectif. Il génère ensuite une autre solution et compare la valeur de la fonction objectif à celle donnée par la solution précédente. L'algorithme garde la solution donnant la meilleure valeur à la fonction objectif et continue jusqu'à ce qu'un critère d'arrêt soit vérifié. Cette méthode n'est pas très performante étant donné qu'elle ne fait qu'explorer l'espace de recherche sans améliorer la qualité des solutions trouvées.

3.2.2.2 Recherche Tabou

Cette méthode a été développée par Fred Glover [29]. Elle s'appuie sur la mémorisation des solutions, ou des caractéristiques de solutions, trouvées lors de la recherche, pour ainsi éviter d'être piégé dans des optimums locaux. Elle consiste en la création d'une liste qui contiendra toutes les solutions récemment explorées qui deviendront tabous. Ainsi, l'algorithme ne pourra plus les choisir comme solution. Ceci permet à l'algorithme de ne pas retomber dans l'optimum local duquel il vient de sortir. Ainsi à chaque itération l'algorithme choisit le meilleur voisin non tabou même si celui-ci détériore la valeur de la fonction objectif. Ceci permet une meilleure exploration de l'espace de recherche.

3.2.2.3 Recuit simulé

L'algorithme du recuit simulé à été proposé par Kirkpatrick et al. [39]. Il s'inspire du processus du recuit des matériaux utilisé en métallurgie. La méthode du recuit consiste à laisser refroidir un matériau de manière lente et contrôlée afin de l'amener vers l'équilibre thermodynamique. L'objectif étant d'obtenir une structure régulière du matériau à l'état solide et de minimiser son énergie.

Dans un problème d'optimisation et par analogie au processus de recuit, la fonction objectif est assimilée à l'énergie du système et le refroidissement et simulé à l'aide

d'un paramètre T (température) qui décroît avec le temps. L'algorithme commence par chercher une solution x_0, affecte une une valeur initiale à la température T et évalue la fonction objectif $f(x_o)$. Ensuite il génère une solution $x_0^{'}$ voisine de x_0 et calcule la valeur de la fonction objectif $f(x_0^{'})$. Puis il calcule l'écart de qualité entre la première et la seconde solution comme suit :

$$\Delta(f) = f(x_0^{'}) - f(x_0) \tag{3.2}$$

Dans le cas d'un problème de maximisation, si $\Delta(f) > 0$ alors la première solution x_0 et remplacée par la deuxième $x_0^{'}$. Sinon, l'algorithme génère un nombre aléatoire $r \in [0, 1]$. Ainsi, si $r < \exp(-\frac{\Delta(f)}{T})$ alors la solution initiale est remplacée par la deuxième solution. Donc même si la nouvelle solution est moins bonne que la première elle peut quand même être acceptée avec une probabilité égale à $\exp(-\frac{\Delta(f)}{T})$. Ceci permet d'explorer une plus grande partie de l'espace de recherche et évite de rester piégé dans un optimum local.

Le recuit simulé est une méthode de recherche local qui vise à trouver l'optimum global d'une fonction objectif.

3.2.2.4 Algorithmes d'intelligence en essaim

Les méthodes d'optimisation d'intelligence en essaim s'inspirent de phénomènes naturels. Elles s'appuient sur une population d'agents. Les agents ont des capacités individuelles très limitées et obéissent à un nombre de règles très simples. Les comportements simples des agents permettent l'apparition d'un comportement collectif complexe, c'est ce qu'on appelle l'intelligence collective. Nous allons nous intéresser aux méthodes les plus connues, à savoir les méthodes de colonies de fourmis et l'optimisation par essaim particulaire.

1. **Algorithme de colonie de fourmis**

   L'algorithme de colonie de fourmis a été proposé par Colorni et al. [17]. Il s'inspire du comportement des fourmis en quête de nourriture. En effet les fourmis communiquent entre elles via leur environnement et parviennent à trouver le

plus court chemin entre la fourmilière et une source de nourriture, bien qu'elles aient individuellement des capacités cognitives limitées.

La figure 3.1 illustre le comportement des fourmis :

(a) Une fourmi trouve la source de nourriture N en empruntant un chemin quelconque et revient vers la fourmilière F en déposant des phéromones sur son chemin.

(b) Les fourmis choisisent aléatoirement les quatre chemins possibles, mais le renforcement de la piste la plus courte la rend plus attractive ;

(c) À cause de l'évaporation des phéromones les chemins les plus longs perdent leur attractivité. Les fourmis empruntent donc le chemin le plus court ayant une densité de phéromone plus élevée.

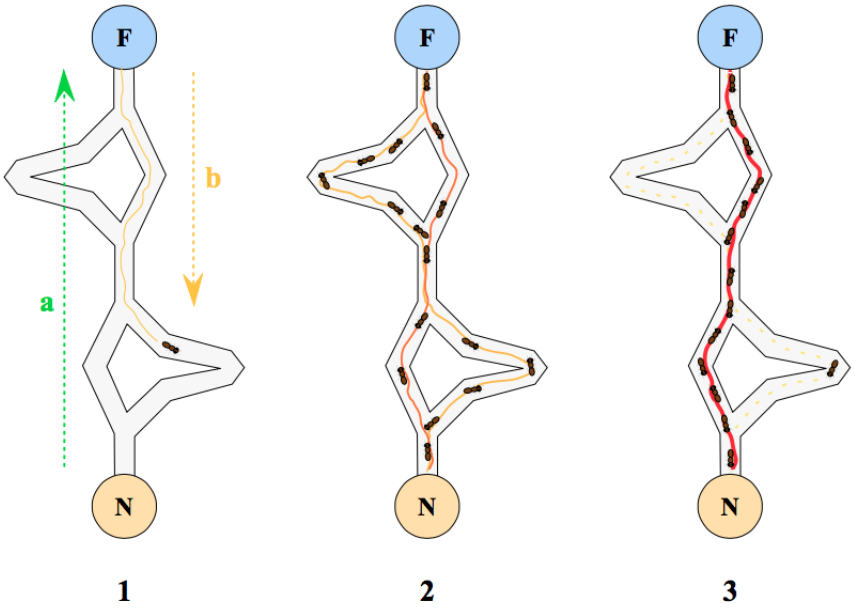

FIGURE 3.1: Optimisation par colonie de fourmis.

Colorni et al ont appliqué l'algorithme de colonie de fourmis pour résoudre le problème du voyageur de commerce qui consiste à chercher le plus court chemin qui visite chaque ville une seule fois et qui termine dans la ville de départ. L'algorithme 1 représente le pseudo code de l'ACO utilisée. Le critère d'arrêt peut être un temps de calcul, un nombre d'itération alloué dépassé,un seuil

d'amélioration des solutions qui n'est plus satisfaisant, ou une combinaison de critères.

Algorithm 1 Algorithme de colonies de fourmis pour le TSP

```
Initialiser une population de fourmis de taille m
Évaluer les m fourmis
while la condition d'arrêt n'est pas vérifiée do
  for i allant de 1 à m do
    Construire le trajet de la fourmis i ;
    Déposer les phéromones sur le trajet de la fourmis i ;
  end for
  Évaluer les m fourmis ;
  Évaporer les pistes de phéromones ;
end while
Retourner la ou les meilleures solutions trouvées.
```

2. **Optimisation par essaim particulaire**

Cette méthode a été développée par Eberhart et Kennedy en 1995[21]. Elle s'inspire du comportement social des animaux évoluant en essaim tels que les bancs de poissons ou vols d'oiseaux. L'essaim particulaire correspond à une population d'agents. Chaque agent (particule) représente une solution potentielle au problème traité et agis indépendemment des autres. Chaque particule possède une position, une vitesse et une mémoire pour enregistrer la meilleure position atteinte durant sa vie (notée p_{best}) et la meilleure position atteinte par les particules voisines (g_{best}). La nouvelle vitesse de la particule est calculée à chaque itération en fonction de sa vitesse actuelle ainsi que sa p_{best} et sa g_{best} à l'aide de la formule suivante :

$$v_{i+1} = \omega * v_i + c_1 * r_1 * (p_{best} - x_i) + c_2 * r_2 * (g_{best} - x_i) \tag{3.3}$$

ou ω désigne le paramètre d'inertie, c_1 détermine l'influence de la composante cognitive et c_2 détermine l'influence de la composante sociale.

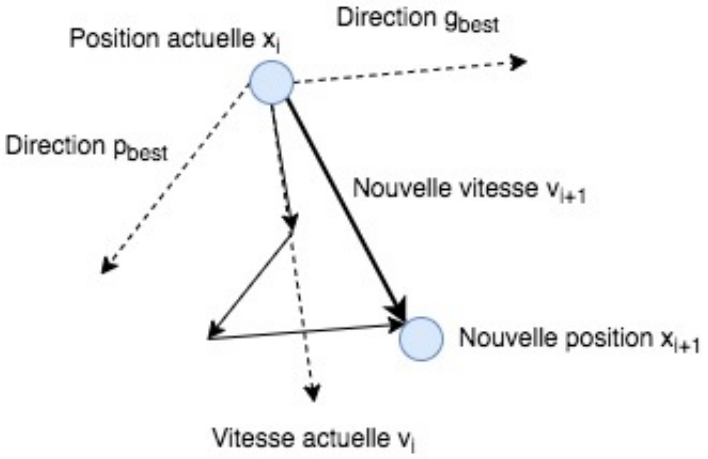

FIGURE 3.2: Calcul de la nouvelle position et de la nouvelle vitesse d'une particule.

Les particules communiquent entre elles à l'aide d'un réseau social. Deux particules "voisines" sont des particules liées dans le graphe du réseau social. Chaque agent obéit à des règles très simples qui permettent l'émergence d'un comportement collectif et adaptatif pour l'essaim entier. Ce qui mène l'essaim à converger rapidement vers un optimum.

L'algorithme 2 représente le pseudo code de l'algorithme PSO. Le critère d'arrêt peut être un temps de calcul, un nombre d'itération alloué dépassé,un seuil d'amélioration des solutions qui n'est plus satisfaisant, ou une combinaison de ces critères.

3.2.2.5 Algorithmes évolutionnaires

Les algorithmes évolutionnaires (AEs) sont inspirés de la théorie de l'évolution proposée par Charles Darwin[18]. Cette théorie repose sur le principe de la sélection naturelle et stipule que les individus les plus adaptés à l'environnement ont plus de chance de survivre et de se reproduire. En effet, les algorithmes font évoluer une population d'individus qui représentent les solutions du problème traité. Les performances des individus sont évaluées à chaque itération à l'aide d'une fonction fitness. Les meilleurs individus seront sélectionnés pour subir des transformations (croisement, mutation) afin de générer des descendants. Les individus (ou une partie des individus) de la nouvelle population vont remplacer ceux de la population courante pour

Algorithm 2 L'optimisation par essaim particulaire

Initialiser les paramètres et la taille S de l'essaim ;
Initialiser les vitesses et les positions des particules ;
Pour chaque particule $p_{best} = x_i$;
Calculer $f(x_i)$;
Calculer g_{best} ;
while la condition d'arret n'est pas vérifiée **do**
  for i allant de 1 à S **do**
    Pour chaque particule :
    Calculer v_{i+1} ;
    Calculer x_{i+1} ;
    Calculer $f(x_i)$;
    if $f(x_i)$ est meilleur que $f(p_{best})$ **then**
      $p_{best} = x_i$;
    end if
    if $f(p_{best})$ est meilleur que $f(g_{best})$ **then**
      $g_{best} = p_{best}$;
    end if
  end for
end while
Retourner la meilleur solution trouvée g_{best}.

former une nouvelle générations d'individus. Ces opérations sont répétées jusqu'à atteindre un critère d'arrêt. Une architecture typique d'un algorithme évolutionnaires est représenté par la figure 3.3.

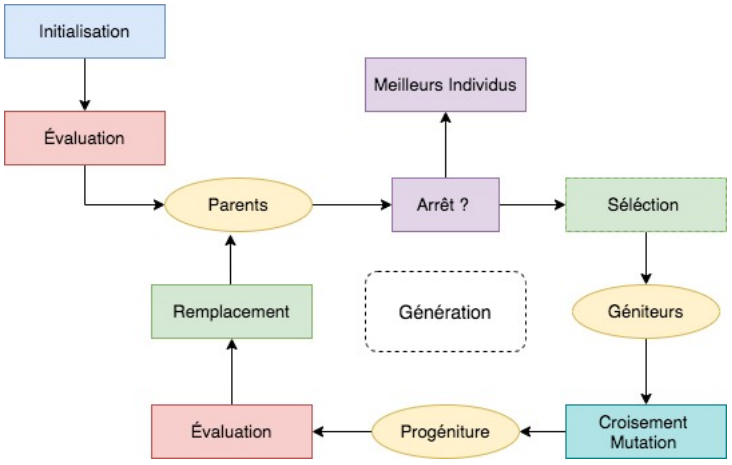

FIGURE 3.3: Organigramme d'un algorithme évolutionnaire.

Il existe plusieurs variantes d'algorithmes évolutionnaires qui se sont développées indépendemment dans les années 60 et 70. Ces variant proposés au fil du temps se basent sur le même principe mais s'adaptent au problème traité. En effet, ces familles

d'algorithmes diffèrent uniquement par la structure du génotype des individus ou par les opérateurs utilisés :

— Les stratégies d'évolution [51] : ces algorithmes permettent la résolution des problèmes d'optimisation continus. Ils sont dotés d'un opérateur de mutation qui utilise une distribution normal. Il existe différentes variantes selon le nombre de descendant que l'on souhaite générer ou garder à chaque itération.

— La programmation évolutionnaire [25] : cette famille d'algorithmes est utilisée pour faire évoluer des automates à états finis. Chaque individus est considéré comme une espèce unique qui génère un descendant unique lors de l'étape de reproduction.

— Les algorithmes génétiques [32] : ce sont les algorithmes les plus connus et les plus utilisés. Ils sont caractérisés pas la représentation des individus sous forme de vecteur binaire ou chaîne de caractère. Nous nous intéresserons plus en détails à cette famille dans la section suivante.

— La programmation génétique [40] : cette famille d'algorithmes diffère des algorithmes génétiques uniquement dans la représentation des données. En effet, les individus sont représentés sous la forme d'un arbre plutôt qu'une chaîne.

1. **Algorithmes génétiques :**

Les AGs s'inspirent de la génétique de population ainsi que la compréhension de la structure mendélienne et ses mécanismes. Ils utilisent le même vocabulaire que celui de la biologie et de la génétique classique. Ainsi, un gène est un ensemble de symboles représentant une variable, un ensemble de gènes forment un chromosome. Un ou plusieurs chromosomes forment un individu qui représente une solution possible au problème traité. Enfin un ensemble d'individus forme une population.

Un AG fait évoluer une population d'individus grâce aux opérations suivantes : Initialisation de la population et codage des individus, évaluation de la population initiale, sélections des individus les mieux adaptés, croisement et mutation pour engendrer de nouveaux individus, sélection des individus présentant la meilleure valeur de fitness pour former la génération suivante. Ces étapes sont répétées jusqu'à atteindre un critère d'arrêt. La taille de la population, la probabilité de croisement, la probabilité de mutation et le nombre maximum de génération sont des paramètres de base qui influencent les performances de l'algorithme.

Les individus [44] formant une population peuvent être codés en utilisant un codage binaire, octal, hexadécimale, réel, à caractère, arborescent ou à permutation. Il existe plusieurs type de croisement : croisement à n-points, croisement uniforme, croisement de 3 parents. Le croisement sert à l'exploitation des solutions déjà existantes. L'opérateur de mutation sert à l'exploration de l'espace de recherche et permet la diversification des solutions. On trouve dans la littérature plusieurs types de mutation : mutation de chaîne de bit, mutation uniforme, non uniforme, gaussienne ...

Les AGs peuvent traiter deux types de problèmes : les problèmes mono-objectif qui visent à optimiser un seul critère et les problèmes multi-objectifs

qui visent à optimiser simultanément plusieurs critères [44].

Algorithmes génétiques				
Codage	Fonction fitness	Croisement	Mutation	Séléction
Binaire	Mono-objectif	Un-point	Uniforme	Roulette
Octal	Multi-objectif	Deux-point	Non-uniforme	Tournoi
Hexadécimal	Dominance Pareto	Uniforme	Gaussienne	Élitiste
Arborescent	Basée critère	Cut and splice	Bit flip	Troncature
Permutation	Basée agrégation			Aléatoire

TABLE 3.1: Taxonomie des algorithmes génétiques.

3.3 Optimisation dans le problème de déploiement des RCSFs : état de l'art

Le déploiement des réseaux de capteurs sans fils est un problème largement traité dans la littérature. Nous allons exposer quelques travaux effectués dans le cadre de l'optimisation du déploiement des RCSFs.

Hanaa ZainEldine et al. [58] ont traité le problème de déploiement aléatoire de capteurs en introduisant une technique de déploiement dynamique améliorée basée sur les AGs. Le but était de maximiser la couverture du réseau et de réduire son cout en minimisant le nombre de capteurs. Les résultats des simulations ont montré l'efficacité de la solution technique proposée.

Yinggao Yue et al. [57] se sont intéressés à la connectivité et la couverture d'un réseau de capteur sans fils. L'objectif étant d'améliorer ces deux critères en utilisant l'algorithme de colonies d'abeilles artificielles. Les simulations ont prouvé que comparée à la distribution aléatoire des noeuds, la méthode proposée aboutit à un RCSF avec un taux de couverture et de connectivité plus élevé. La redondance des données est également réduite ainsi que le trafic du réseau ce qui améliore son efficacité.

Ali afhantoloee et al. [1] ont proposé une méthode pour le déploiement 3D des capteurs dans un environnement indoor. Le but est de faciliter la mobilité des personnes handicapés. Ils ont utilisé un algorithme basé sur la diagramme de Voronoï pour améliorer la couverture du réseau de capteurs.

Hanh et al. [47] dans leur article parrue en 2017 ont traité le problème de déploiement d'un réseau ayant différents types de capteurs. Ils ont proposé une méthode basée sur l'optimisation par essaim particulaire. Les résultats des simulations ont montré que cette méthode donne des solutions de meilleure qualité comparé à d'autres méthodes. La méthode a également fait ses preuves en terme de temps de calcul et de vitesse de convergence.

Mohamed Benatia et al. [13] traitent du problème de déploiement des noeuds capteurs ainsi que des noeuds relais. Ils ont étudié deux variantes d'algorithmes évolutionnaires : algorithme génétique classique et NSGA-II. Plusieurs simulations ont été effectuée afin de déterminer les meilleures valeurs des paramètres de base (taille de la population, probabilité de croisement, probabilité de mutation ...). Les chercheurs avaient pour but d'optimiser la couverture, la connectivité ainsi que le coût des réseaux de capteurs déployés dans un smart building.

Tian et al. [54] se sont intéressés aux réseaux de capteurs sans fils ayant des noeuds à énergie limitée. Leur objectif était d'optimiser la couverture du réseau et de réduire le nombre de noeuds actifs. Ils ont utilisé des méthodes basées sur les algorithmes génétiques et l'algorithme de colonies de fourmis binaire. Ils sont arrivés à la conclusion que la combinaison des deux approches donne lieu à un algorithme ayant une précision plus élevée et une convergence plus rapide.

Akbarzadeh et al. [3] ont proposé une adaptation de la méthode de la descente du gradient pour optimiser le posistionnement et l'orientation des capteurs. Ils ont utilisé des modèles réalistes qui prennent en considération la topographie de l'environnement. Les résultats ont montré que leur méthode avait un coût de calcul inférieur comparé à deux autres méthodes et donnait des résultats de meilleur qualité.

N. Aitsaadi et al. [2] ont utilisé une méthode pseudo-aléatoire basée sur la recherche Tabou afin de déterminer le nombre optimal de noeuds ainsi que leurs positions dans le but de couvrir en totalité la région surveillée. L'étude a également pris en considération la contrainte de durée de vie et la connectivité. Les résultats ont montré que la méthode

surpasse plusieurs autres approches utilisées dans la littérature.

Année	Premier auteur	Objectifs	Méthode utilisée
2020	Hanaa ZainEldine	- Couverture - Cout	IDDT-GA
2019	Yinggao Yue	- Couverture - Connectivité	ABC
2018	Ali Afghantoloee	- Couverture	- 3D Voronoi
2018	Nguyen Thi Hanh	- Couverture	PSO
2017	Mohamed Benatia	- Couverture - Coût - Connectivité	GA NSGA-II
2016	Jingwen Tian	-Couverture	GA BAC
2014	Vahab Akbarzadeh	- Couverture	Descente de Gradient CMA-ES
2011	N. Aitsaadi	- Coût - Connectivité - Durée de vie	Recherche Tabou

TABLE 3.2: Quelques travaux relatifs au déploiement des RCSFs.

La plupart des travaux cités ci-dessus ne traitent pas le problème de déploiement d'un réseau de capteurs de manière exhaustive. En effet, certains ne traitent qu'un objectif à la fois (couverture, connectivité, coût, durée de vie), d'autres deux mais rarement trois ou plus. Plus le nombre d'objectifs pris en considération est élevé meilleures sont les performances des réseaux.

3.4 Conclusion

Le déploiement des RCSFs est un problème NP-difficile, il ne peut être résolu par des méthodes déterministes. Nous allons donc opté pour une méthode stochastique afin

de le résoudre. Nous allons utiliser une des variantes des algorithmes évolutionnaires qui se sont révélés efficaces pour la résolution de ce type de problème.

Dans le chapitre suivant nous allons implémenter trois variantes d'algorithmes évolutionnaires en utilisant le langage Python. Nous allons adapter la structure des individus à notre modélisation du problème. Nous allons également procéder à plusieurs simulations afin de déterminer les valeurs optimales des paramètres. Et enfin déterminer la disposition la plus optimale des capteurs.

Chapitre 4

Résolution du problème de déploiement d'un RCSF dans un bâtiment intelligent

4.1 Introduction

Le déploiement d'un RCSF est un problème très complexe, surtout quand le nombre de critères pris en considération est supérieur à 2. Nous allons dans cette partie développer une application qui rendra la tache facile aux ingénieurs qui sont confrontés à ce genre de problèmes dans leur travail quotidien.

Nous prendrons en considération les différents modèles retenus dans le deuxième chapitre concernant le coût, la connectivité, la couverture et la sur-couverture. Nous allons ensuite les intégrer dans les trois méthodes d'optimisation multi-objectifs que nous avons choisies : SPEA-II, NSGA-II et NSGA-III.

Pour finir, nous discuterons les résultats obtenus à l'aide des trois algorithmes et nous établiront un plan de déploiement pour un bâtiment exemple.

4.2 Choix du langage de programmation

4.2.1 Python

Python est le langage de référence pour plusieurs domaines y compris l'intelligence artificielle. Il existe sous une licence libre et fonctionne sur la plupart des plates-formes informatiques, des smartphones aux ordinateurs, de Windows à Unix avec notamment GNU/Linux en passant par MacOS, ou encore Android, iOS, et peut aussi être traduit en Java ou .NET. Il est conçu pour augmenter la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe très clair et simple à utiliser ce qui induit à un temps de développement trés court comparé à d'autres langages tel que JAVA, C++ ou Ruby. Il peut s'utiliser dans de nombreux contextes et s'adapter à plusieurs type d'utilisation grâce à des bibliothèques spécialisées.

Il est également doté d'une communauté très active. c'est donc en connaissance de cause que notre choix s'est porté sur Python.

4.2.2 PyCharm

Pour développer notre application, nous avons choisi un environnement de développement intégré (IDE : integrated development environment) python nommé Pycharm. Développé par l'entreprise tchèque JetBrains. Il est utilisé par des programmeurs professionnels dans le monde entier. Il s'agit d'un environnement multi-plateforme qui fonctionne sur Windows, MacOS X et Linux. Il est disponible en trois versions, la version Community, la version Educational et la version Professional. Les deux premières versions sont open source, alors que la version Professional est payante. La version Community, celle que nous avons utilisé dans notre travail, possède différentes fonctionnalités tels que la coloration syntaxique, l'auto-complétion ou encore la vérification de code en direct. Cette version gratuite est très complète et répond à tous les besoins d'un développeur python [14].

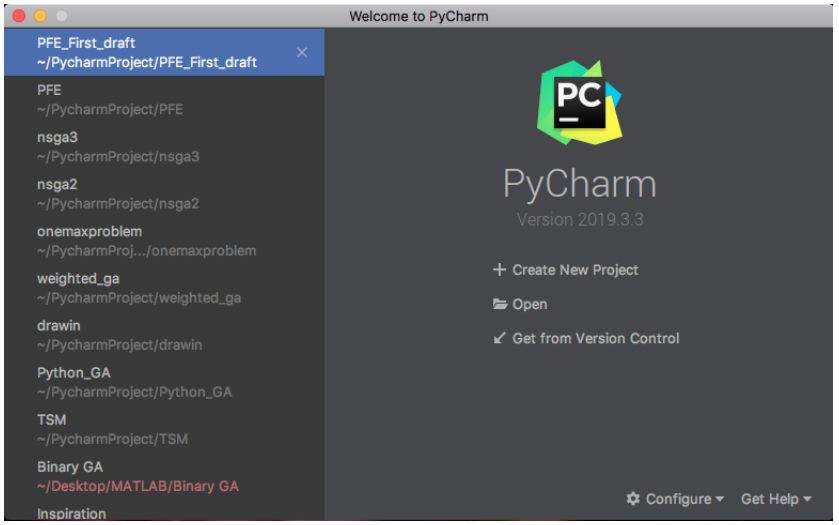

FIGURE 4.1: IDE PyCharm.

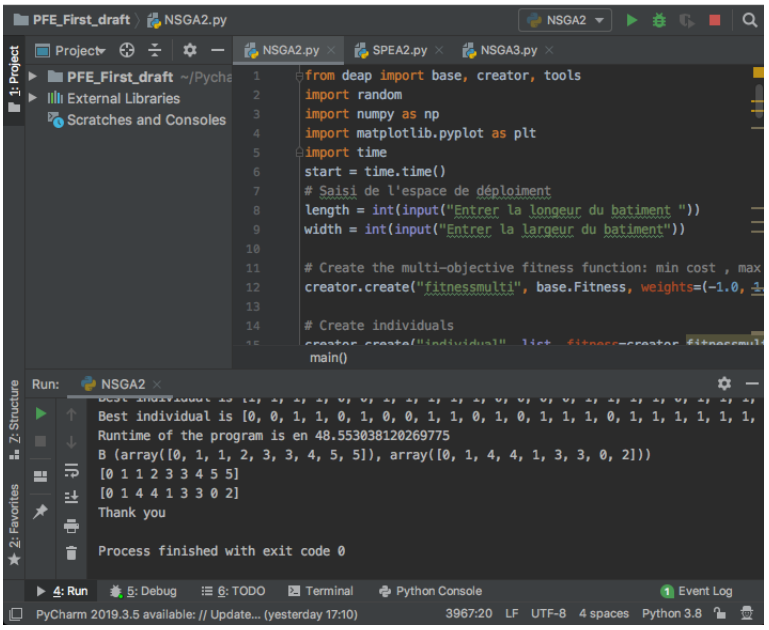

FIGURE 4.2: Composantes de l'IDE PyCharm.

4.2.3 DEAP

DEAP (Distributed Evolutionary Algorithm in Python) [26] est un logiciel open source, sous licence LGPL, développé principalement au Laboratoire de Vision et Systèmes Informatiques de l'Université Laval, Québec, Canada. Il a été conçu pour aider les chercheurs à développer des algorithmes évolutifs personnalisés. Il privilégie les algorithmes explicites et les structures de données transparentes, contrairement à la plupart des autres logiciels informatiques évolutifs qui ont tendance à encapsuler des algorithmes standardisés en utilisant l'approche de la boîte noire. Sa boîte à outils regroupe tous les opérateurs nécessaires et leurs arguments dans une structure pratique. Il comprend les fonctionalités suivantes :

- Les algorithmes génétiques avec possibilité d'utiliser n'importe quelle structure de données (liste, tableau, ensemble, dictionnaire, arbre ...) ;
- La programmation génétique ;
- La stratégie d'évolution ;
- L'optimisation multi-objectifs (NSGA-II, NSGA-III, SPEA2) ;
- La généalogie d'une évolution.

4.3 Algorithmes utilisés

Notre objectif est d'optimiser plusieurs critères (coût, couverture, connectivité, sur-couverture) simultanément. Afin de répondre à nos besoin, nous avons décidé d'utiliser des algorithmes évolutionnaires multi-objectifs (AEMOs). Dans le but de faire une étude un peu large, nous avons implémenté trois variantes d'AEMOs.

Dans ce qui suit, nous allons donner une brève description de chaque variante ainsi que son principe de fonctionnement. Mais avant cela, nous devons définir une notion très importante pour les AEMOs : la notion de dominance. Elle repose sur le postulat de Pareto [49] : "Il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères". Le critère de dominance selon Pareto est défini comme suit : Soit $f_i, i \in [1, m]$ un ensemble de critères à minimiser, et x et $x^{'}$ deux solution de l'espace réalisable. On dira que x domine $x^{'}$ au sens de Pareto si,$\forall i \in [1, m], f_i(x) \leq f_i(x^{'})$, avec au moins une inégalité stricte. Une solution dite Pareto optimale est une solution non dominée, c'est à dire qu'aucune autre solution de l'espace réalisable ne domine cette solution, selon le critère de dominance.

4.3.1 SPEA-II

SPEA-II (Strength Pareto Evolutionary Algorithm II) [15] est une version améliorée de l'algorithme évolutionnaire SPEA. Il s'agit d'un algorithme génétique multi-objectif (AGMO). En plus de toutes les caractéristiques d'un AG classique, SPEA-II intègre plusieurs fonctionnalités tel que l'utilisation de la notion de dominance selon Pareto, la création d'une archive dans laquelle l'algorithme stocke toutes les solutions non dominées à chaque itération et enfin la réduction du nombre de solution dans l'archive à l'aide d'une méthode de clustering. La figure 4.3 montre l'organigramme de l'algorithme.

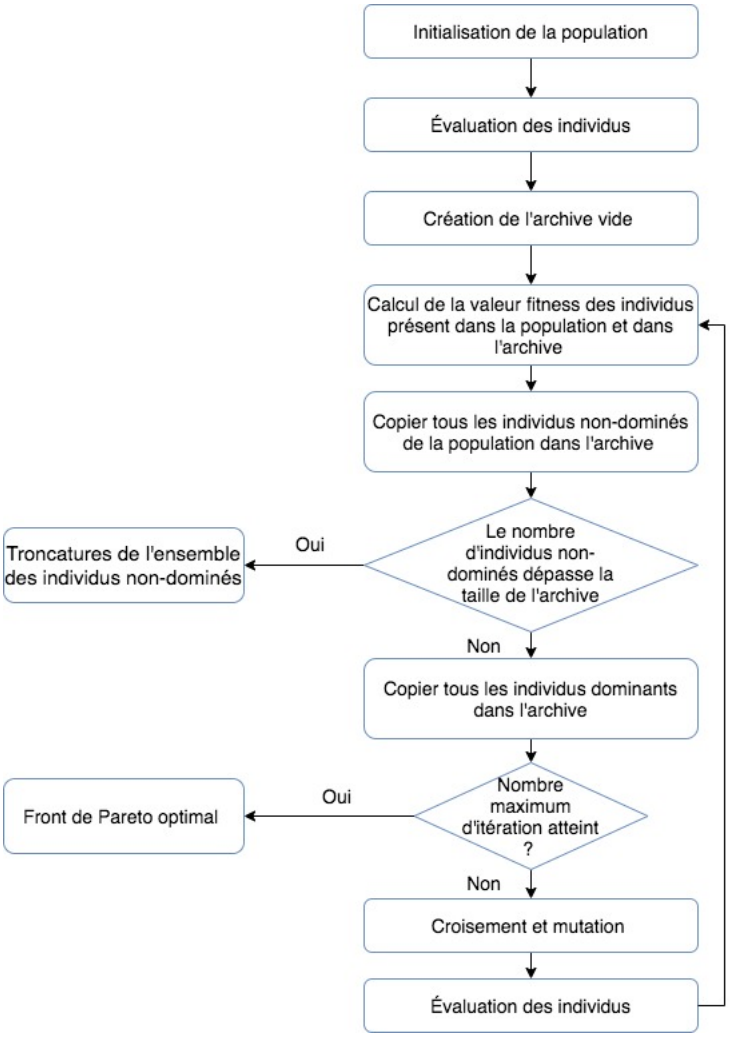

FIGURE 4.3: Principe de fonctionnement de l'algorithme SPEA-II.

4.3.2 NSGA-II

NSGA-II (Non-dominated sorting genetic algorithm) est une méthode proposée par Deb et al. [20]. Il s'agit d'un AGMO basé sur le principe de dominance selon Paréto. L'algorithme commence par générer la population initiale composée de N individus. Ensuite, les individus sont classés selon la notion de dominance de Paréto. Les individus non-dominés sont regroupés dans le premier front F1. Le reste des individus sont ensuite classés et d'autres fronts apparaissent. Le principe de la méthode est présenté dans la figure 4.4.

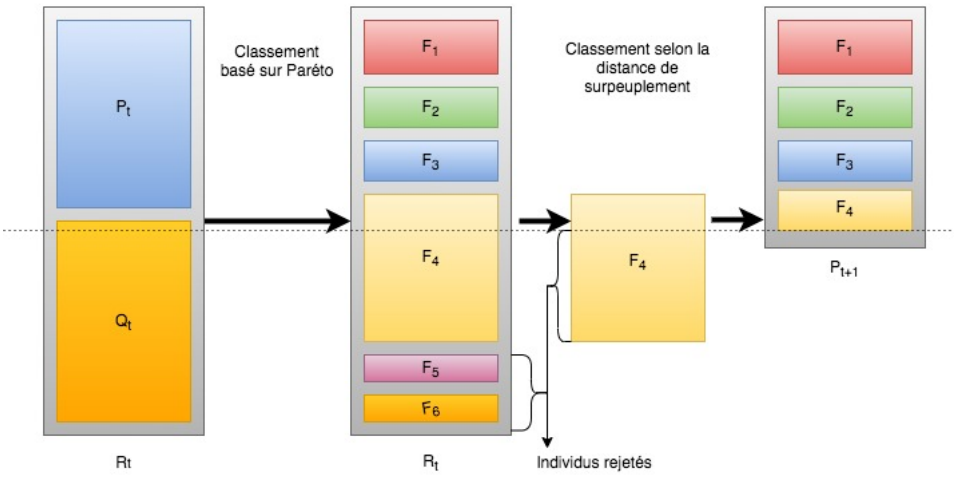

FIGURE 4.4: Principe de fonctionnement de l'algorithme NSGA-II.

4.3.3 NSGA-III

NSGA-III [19] est une version améliorée de NSGA-II. Cet algorithme est utilisé lorsque le nombre d'objectifs à optimiser est supérieur à 3. Il utilise une approche basé sur le point de référence. Le nombre de points de référence est égale au nombre d'individus présent dans la population. C'est ce qui le distingue de NSGA-II qui lui utilise la distance de surpeuplement. La différence réside aussi dans la sélection des individus. En effet, pour comparer entre deux individus, NSGA-II utilise la notion de dominance Paréto et une valeur de distance de surpeuplement alors que NSGA-III effectue une sélection si et seulement si au moins un des deux individus est une solution irréalisable.

4.4 Adaptation des algorithmes au problème de déploiement des RCSFs

Pour résoudre le problème du déploiement d'un RCSF dans un bâtiment intelligent, il faut adapter les méthodes décrites précédemment aux modèles que nous avons retenus dans le deuxième chapitre.

4.4.1 Codage du chromosome

Dans le deuxième chapitre, l'espace de déploiement a été modélisé avec une grille qui est implémentée sous forme de matrice. Les algorithmes que nous avons sélectionné ne peuvent manipuler ce type de structures de données. Il faut donc convertir cette matrice de déploiement en un vecteur comme on peut le voir dans la figure 4.5. Les individus qui sont les solutions du problème sont donc représentés sous forme de vecteur binaire. Chaque case porte deux informations : la présence ou l'absence d'un capteur, son emplacement.

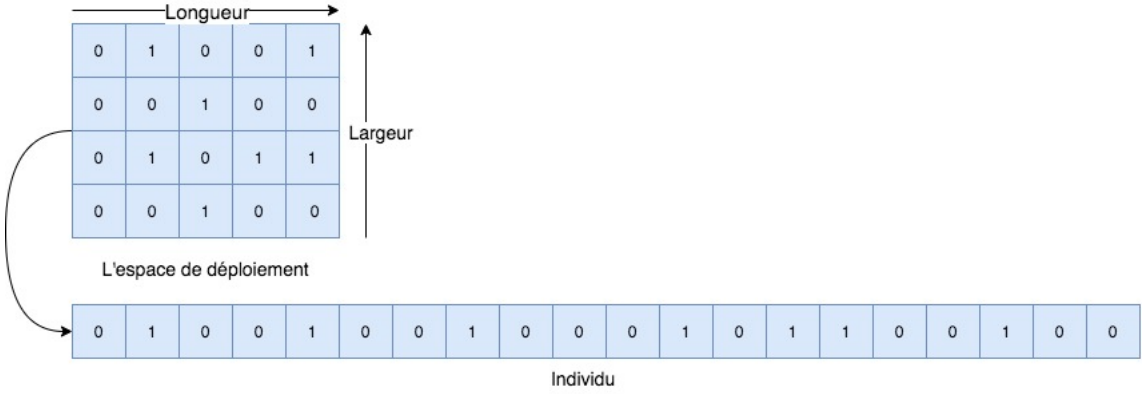

FIGURE 4.5: Codage d'un individu.

4.4.2 Initialisation de la population

L'initialisation de la population se fait au début de l'exécution de l'algorithme. Après avoir défini la taille de la population, les algorithmes génèrent aléatoirement les individus. Le fait de générer les individus de manière aléatoire permet d'avoir une population de solutions diversifiées et évite que l'algorithme tombe dans un optimum local. La figure 4.6 montre une exemple d'une population de N individus.

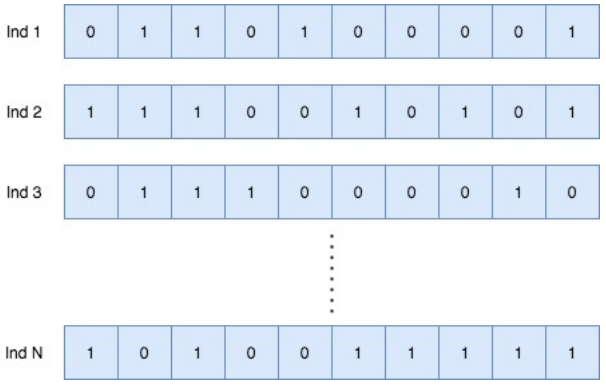

FIGURE 4.6: Population de N individus.

4.4.3 Évaluation et sélection des individus

L'évaluation des individus se fait à l'aide de la fonction fitness (fonction d'évaluation). Dans notre cas la fonction d'évaluation évalue le coût de chaque individus, son taux de couverture, son taux de sur-couverture et enfin son taux de connectivité. La sélection des individus se fait selon la valeur fitness. Les individus ayant une meilleure valeur fitness sont sélectionnés plus souvent pour participer à la reproduction. Le processus de sélection diffère d'un algorithme à un autre.

4.4.4 Choix de l'opérateur de croisement

Comme nous l'avons vu dans le chapitre précédent, il existe plusieurs types de croisement. Nous avons opté pour le croisement à 1-point dont le principe est représenté dans la figure 4.7. La probabilité de croisement désigne le nombre d'individus de la population (parents) qui vont subir l'opération de croisement pour former de nouveaux individus (enfants).

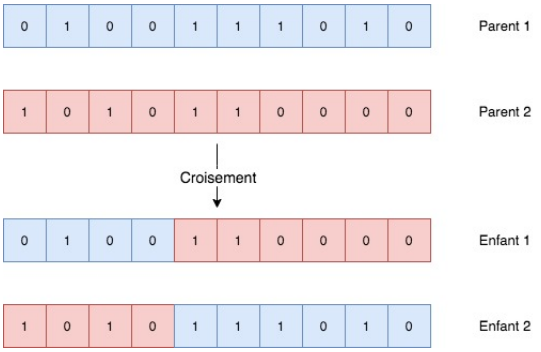

FIGURE 4.7: Croisement à 1 point.

4.4.5 Choix de l'opérateur de mutation

L'opération de mutation est effectuée après l'opération de croisement. Cette opération est réalisée avec un taux (pobabiblité de mutation) faible. Vu que nous avons opté pour une représentation binaire, nous ne pouvons utiliser un type de mutation autre que l'inversion binaire. Le principe de l'inversion binaire est représenté dans la figure 4.8. La mutation permet de générer des perturbations dans les solutions (individus) sélectionnés, il en résulte le maintien de la diversité de la population ainsi qu'une meilleure exploration de l'espace de recherche.

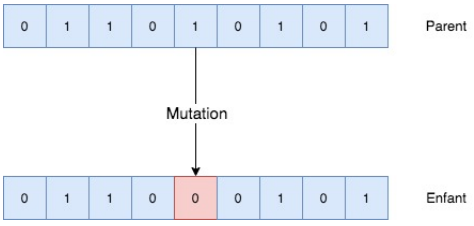

FIGURE 4.8: Mutation à inversement de bit.

4.5 Modélisation mathématique du problème

L'espace de déploiement est subdivisé en une grille, il en résulte un ensemble A d'emplacements possibles $(i, j) \in A$ pour les noeuds du réseau. Si un noeud est installé dans une position (i, j) de A, la variable $D_{i,j}$ est affectée de la valeur 1. La matrice de déploiement est transformée en un vecteur binaire pour les besoins des algorithmes, ce vecteur est noté X. S et R représentent respectivement l'ensemble des noeuds capteurs et des noeuds routeurs installés. En ce qui concerne les capteurs, les objectifs à optimiser sont les suivants :

$$\begin{aligned} \text{Min } \mathrm{F}_1 &= \text{Coût}\,(X) \\ \text{Max } \mathrm{F}_2 &= \text{Couv}_{RI}\,(X) \\ \text{Max } \mathrm{F}_3 &= \text{Con}_{RI}\,(X) \\ \text{Min } \mathrm{F}_4 &= \text{Sur-couv}_{RI}\,(X) \end{aligned} \tag{4.1}$$

Sous contrainte :

$$S \in \mathrm{A} \tag{4.2}$$

Quant au déploiement des noeuds routeurs, uniquement l'optimisation des deux objectifs coût et connectivité aura lieu. En effet les routeurs ont pour rôle d'assurer la transmission des données et n'effectuent aucune tache de surveillance. L'équation (4.3) représente ces objectifs :

$$\begin{aligned} \text{Min } \mathrm{F}_1 &= \text{Coût}\,(X) \\ \text{Max } \mathrm{F}_2 &= \text{Con}_{RI}\,(X) \end{aligned} \tag{4.3}$$

Sous contrainte :

$$R \in \mathrm{A} \tag{4.4}$$

4.6 Construction des algorithmes

Nous allons dans cette partie expliquer chacune des étapes des trois algorithmes que nous avons implémentés sous Python. Les étapes présentées dans les sections 4.6.1 jusqu'à 4.6.3 sont les mêmes pour les trois algorithmes. Les étapes exposée dans les deux dernières sections sont quant à elles propres à chaque algorithme.

4.6.1 Importation des modules

Pour commencer, nous avons importé les modules et les bibliothèques contenant tous les outils nécessaires à l'exécution des algorithmes. Les modules "base, creator, tools" importé depuis la bibliothèque "deap" sont nécessaires à la définition des caractéristiques des algorithmes. Le module "random" permet de générer aléatoirement les individus d'une population. Enfin, le module "time" nous permettra d'avoir le temps d'exécution des simulations.

```
1 from deap import base, creator, tools
2 import random
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import time
```

4.6.2 Création des individus

La première étape est bien évidement la définition de l'espace de déploiement. Pour ce faire, l'application demande à l'utilisateur d'insérer la longueur ainsi que la largeur du bâtiment. Ensuite, la fonction fitness est créée. Comme nous avons 4 objectifs à optimiser nous avons introduit 4 poids dans la fonction : 1 pour maximiser une fonction, -1 pour la minimiser. À l'aide des outils "creator" et "toolbox" nous avons crée des individus sous forme de liste (vecteur) ayant comme valeur des 0 et des 1, d'une taille égale à $longueur * largeur$ du bâtiment. La population est bien crée maintenant.

```
# Deployment space
length = int(input("Entrer la longeur du batiment "))
width = int(input("Entrer la largeur du batiment"))

# Create the multi-objective fitness function: min cost , max coverage,
    max connectivity, min over-coverage.
creator.create("fitnessmulti", base.Fitness, weights=(-1.0, 1.0, 1.0,
    -1.0))

# Create individuals
creator.create("individual", list, fitness=creator.fitnessmulti)

IND_SIZE = width * length  # Size of the individual = Width * Lenght

toolbox = base.Toolbox()

toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.individual,
    toolbox.attr_bool, n=IND_SIZE)

# Create the population
toolbox.register("population", tools.initRepeat, list, toolbox.
    individual)
```

4.6.3 Création de la fonction d'évaluation

Afin d'évaluer chaque individu, nous avons implémenté quatre fonctions objectifs à l'intérieur d'une seule et même fonction "evaluate" qui a pour argument la variable "individu". Pour ce faire, nous avons d'abord converti l'individu en forme de vecteur vers une forme matricielle. Puis nous avons extrait les coordonnées (i, j) des positions des noeuds. Ensuite nous avons définit le rayon de couverture des capteurs ainsi que leur rayon de connectivité (dans le cas des routeurs seul le rayon de connectivité est défini). Ceci étant fait, nous procédons au calcul des matrices de couverture, connectivité et sur-couverture. Nous avons utilisé le principe de la distance euclidienne. Une cellule est dite couverte (reçois la valeur 1) si la distance la séparant du capteur est inférieur à son rayon de couverture. De même pour dire qu'une cellule se situe dans la zone de connectivité d'un capteur ou routeur. Pour calculer la matrice de sur-couverture, nous parcourons la matrice de couverture et à chaque fois qu'une case porte une valeur supérieur ou égale à 2 (c'est à dire qu'elle est couverte par 2 capteurs ou plus). La même case est affectée de la valeur 1 dans la matrice de sur-couverture. Les fonctions coût, couverture, connectivité et sur-couverture sont calculées suivant les formules qu'on peut voir dans les ligne 37, 38, 39 et 40 de l'algorithme. Elles sont ensuite retournées par la fonction "evaluate" pour permetre l'évaluation des individus par la fonction fitness. Toutes ces étapes sont explicitées dans le code Python ci-dessous.

```
1  # Create the evaluation function
2  def evaluate(individual):
3      col1 = length
4      indiv_m = np.array([individual[i:i + col1] for i in range(0, len(
       individual), col1)])  # Convert list into a matrix
5      position = np.where(indiv_m == 1)  # Posistion of sensors
6      x = position[0]
7      y = position[1]
8      rayon_cov = 1.5 # coverage radius
9      rayon_con = 3  # connectivity_radius
10     coverage = np.zeros((width, length))  # calculate the coverage
       matrix including overcoverage
```

```
    for z in range(len(x)):
        for i in range(width):
            for j in range(length):
                d = np.sqrt(((i - x[z]) ** 2) + ((j - y[z]) ** 2))
                if d <= rayon_cov:
                    coverage[i, j] += 1
    cov = np.zeros((width, length))  # calculate the coverage matrix
    for h in range(width):
        for k in range(length):
            if coverage[h, k] >= 1:
                cov[h, k] = 1
    vect_a = np.ravel(cov)
    connectivity = np.zeros((width, length))  # calculate the
connectivity matrix
    for m in range(len(x)):
        for n in range(width):
            for p in range(length):
                dis = np.sqrt(((n - x[m]) ** 2) + ((p - y[m]) ** 2))
                if dis <= rayon_con:
                    connectivity[n, p] = 1
    vect_b = np.ravel(connectivity)
    over_coverage = np.zeros((width, length))  # calculate the over
coverage matrix
    for q in range(width):
        for s in range(length):
            if coverage[q, s] >= 2:
                over_coverage[q, s] = 1
    vect_c = np.ravel(over_coverage)
    a = (sum(individual))  # Cost
    b = ((sum(vect_a)) / IND_SIZE) * 100  # The sensing coverage rate
function
    c = ((sum(vect_b)) / IND_SIZE) * 100  # The connectivity rate
function
    d = ((sum(vect_c)) / IND_SIZE) * 100  # The overcoverage rate
function
    print("cost = ", a, "cov = ", b, "%", "con = ", c, "%", "over_cov =
", d, "%")
    return a, b, c, d
```

4.6.4 Définition des opérateurs génétiques

À l'aide de l'outil "toolbox.register", nous avons choisi les opérateurs à utiliser lors du processus d'évolution. Pour l'opération de croisement, nous avons choisi l'opérateur "cxOnePoint" (croisement à 1-point). Pour l'opération de mutation, nous avons choisi l'opérateur "mutFlipBit" (inversement de bit). Pour l'opération d'évaluation, nous avons choisi la fonction "evaluate" que nous avons créée au préalable. Enfin, en ce qui concerne l'opération de sélection, le type diffère selon l'algorithme. En effet, pour SPEA-II nous avons utilisé "selSPEA2", pour NSGA-II c'est "selNSGA2" et finalement pour NSGA-III l'outil de sélection est appelé "selNSGA3".

```
# D finition of the operators
toolbox.register("mate", tools.cxOnePoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.1)
toolbox.register("select", tools.selNSGA2)
toolbox.register("evaluate", evaluate)
```

4.6.5 Évolution de la population

4.6.5.1 Initialisation de la population :

Afin d'initialiser la population, il suffit de définir le nombre d'individus qu'elle contient (NPOP) et d'utiliser l'outil "toolbox.population". Ceci génèrent NPOP vecteurs auxquels il affecte aléatoirement des 0 et des 1 pour former les individus.

```
# create an initial population of individuals (where
    # each individual is a list  )
    NPOP = 80
    pop = toolbox.population(n=NPOP)
```

4.6.5.2 Processus d'évolution :

Avant d'entamer le processus d'évolution, il faut d'abord fixer les probabilités de mutation et de croisement. On commence par l'évaluation de la population initiale, puis on entre dans la boucle d'évolution. La première étape consiste en la sélection des

individus qui formeront la prochaine génération (les descendants). Ces descendants sont ensuite soumis aux opérations de croisement et de mutation puis à l'opération d'évaluation, le but de ces opérations étant l'exploration de l'espace de recherche dans le but de trouver de meilleurs individus. Enfin, la population N-1 est remplacée entièrement ou en partie par les descendants. Quand la boucle atteint le nombre prédéfini de générations (itérations), le processus d'évolution s'achève et on obtient la population finale. L'outil de sélection permet d'ordonner la population finale selon le principe de sélection de chaque algorithme. On obtient ainsi une liste d'individus avec leur coût, taux de couverture, taux de connectivité et taux de sur-couverture. L'utilisateur peut choisir la solution qui lui convient le plus selon l'ordre de priorité qu'il donne aux différents critères de déploiement.

```
# CXPB  is the probability with which two individuals are crossed
# MUTPB is the probability for mutating an individual
CXPB, MUTPB = 0.9, 0.1

print("Start of evolution")

# Evaluate the entire population
fitnesses = list(map(toolbox.evaluate, pop))
for ind, fit in zip(pop, fitnesses):
    ind.fitness.values = fit

print("  Evaluated %i individuals" % len(pop))

# Variable keeping track of the number of generations
g = 0

# Begin the evolution
while g < 100:

    # A new generation
    g = g + 1
    print("-- Generation %i --" % g)

    # Select the next generation individuals
```

```
    offspring = toolbox.select(pop, len(pop))
    # Clone the selected individuals
    offspring = list(map(toolbox.clone, offspring))

    # Apply crossover and mutation on the offspring
    for child1, child2 in zip(offspring[::2], offspring[1::2]):

        # cross two individuals with probability CXPB
        if random.random() < CXPB:
            toolbox.mate(child1, child2)

            # fitness values of the children
            # must be recalculated later
            del child1.fitness.values
            del child2.fitness.values

    for mutant in offspring:

        # mutate an individual with probability MUTPB
        if random.random() < MUTPB:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # Evaluate the individuals with an invalid fitness
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    fitnesses = map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    print("  Evaluated %i individuals" % len(invalid_ind))

    # The population is entirely replaced by the offspring
    pop[:] = offspring
    print("pop", pop)

print("-- End of (successful) evolution --")

best_ind = tools.selNSGA2(pop, NPOP, nd='standard')
```

```
63
64 for i in range(len(best_ind)):
65     print("Best individual is %s, %s" % (best_ind[i], best_ind[i].
   fitness.values))
```

4.6.6 Exemple de solutions

On considère un espace de déploiement de 64 m^2. On lance l'algorithme NSGA-II et on note les solutions données à la fin de l'exécution. Une partie des solutions générées est représenté dans le tableau 4.1. On remarque qu'aucune solution ne domine l'autre selon le principe de Paréto. Ce sont toutes des solutions dites Paréto optimales. C'est donc à l'utilisateur de choisir la solution qui lui convient le plus selon l'ordre d'importance qu'il donne aux critères de déploiement.

Solution	Coût (unité)	Couverture (%)	Connectivité (%)	sur-couverture (%)
1	10	100.00	100.00	17.18
2	9	96.87	100.00	6.25
3	8	90.62	100.00	3.12
4	7	79.68	100.00	0.00
5	6	70.31	98.43	0.00
6	5	56.25	90.62	0.00

TABLE 4.1: Exemple de solutions données par l'algorithme NSGA-II.

4.7 Simulation

4.7.1 Cas d'étude

Afin de tester et d'évaluer les performances des différents algorithmes, nous avons pris un cas d'étude. Il s'agit d'un espace restreint de $80m^2$ (un bureau) : 10 m de longueur et 8 m de largeur. Nous avons modélisé cet espace à l'aide d'un logiciel en ligne appelé *my3dplanner* (voir les figures 4.9 et 4.10). Nous allons d'abord étudier l'influence des paramètres suivants : probabilité de mutation , probabilité de croisement, taille de la population ainsi que le nombre de génération sur le temps d'exécution

des algorithmes et la qualité des solutions qu'ils donnent. Nous avons supposé que les capteurs avaient un rayon de couverture égale à 1.5m et un rayon de connectivité égale à 3m. En ce qui concerne les routeurs, nous avons supposé qu'ils avaient un rayon de 5m. Nous allons ensuite résoudre le problème de déploiement.

FIGURE 4.9: Plan 2D de l'espace de déploiement.

FIGURE 4.10: Vue 3D de l'espace de déploiement.

4.7.2 Influence des paramètres sur les performances des algorithmes.

Nous avons étudié l'influence des différents paramètres sur les performances des algorithmes dans le but de sélectionner la meilleur combinaison de paramètres. Nous nous somme d'abord intéressé à l'influence de la probabilité de mutation et la probabilité de croisement sur le temps d'exécution. Nous avons fait varier la probabilité de croisement de 0 à 0.9 avec un pas égale à 0.1, et celle de mutation de 0.01 à 0.1 avec un pas égale à 0.01. Nous avons fixé le nombre de générations à 100 et le nombre d'individu à 80. Le tableau 4.2 ainsi que la figure 4.11 représentent les résultats que nous avons obtenus.

Les résultats montrent que le temps d'exécution est proportionnel aux probabilités de croisement et de mutation. En effet plus les probabilités de croisement et de mutation sont grandes, plus le processus d'évolution prend du temps. Quant aux solutions, nous avons remarqué que plus la probabilité de croisement est grande plus les solutions donnés par les algorithme sont de meilleur qualité et satisfont nos quatre objectifs. Nous avons donc fixé la probabilité de croisement à 0.9 et la probabilité de mutation à 0.1, pour la suite des manipulations.

$\frac{Pmu}{Pcx}$	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	5.024	8.10	10.84	13.33	17.15	18.01	21.29	24.65	26.30	28.54	33.98
0.1	34.00	33.30	37.26	29.69	41.83	36.95	40.40	46.59	44.66	46.43	56.43
0.2	56.75	59.87	55.05	57.29	53.93	60.97	59.61	62.57	63.70	66.91	57.60
0.3	77.21	71.22	70.70	69.59	74.20	70.23	77.39	84.35	68.95	85.72	80.43
0.4	101.59	133.70	90.76	81.37	83.94	90.83	85.73	100.69	88.64	88.59	101.58
0.5	108.40	114.73	91.61	112.86	106.30	89.79	90.33	94.67	109.41	107.17	101.72
0.6	119.19	111.82	121.42	110.84	113.84	118.37	103.44	100.65	124.66	102.76	115.04
0.7	142.76	116.89	136.67	118.72	138.76	139.11	120.92	120.71	118.00	118.34	133.37
0.8	133.78	146.75	115.14	137.83	141.52	137.16	136.74	169.60	143.94	258.33	271.45
0.9	142.93	154.63	160.46	125.44	215.49	251.08	258.80	140.64	148.12	138.10	157.14

TABLE 4.2: Temps d'exécution (en seconde) en fonction de la probabilité de mutation et de la probabilité de croisement.

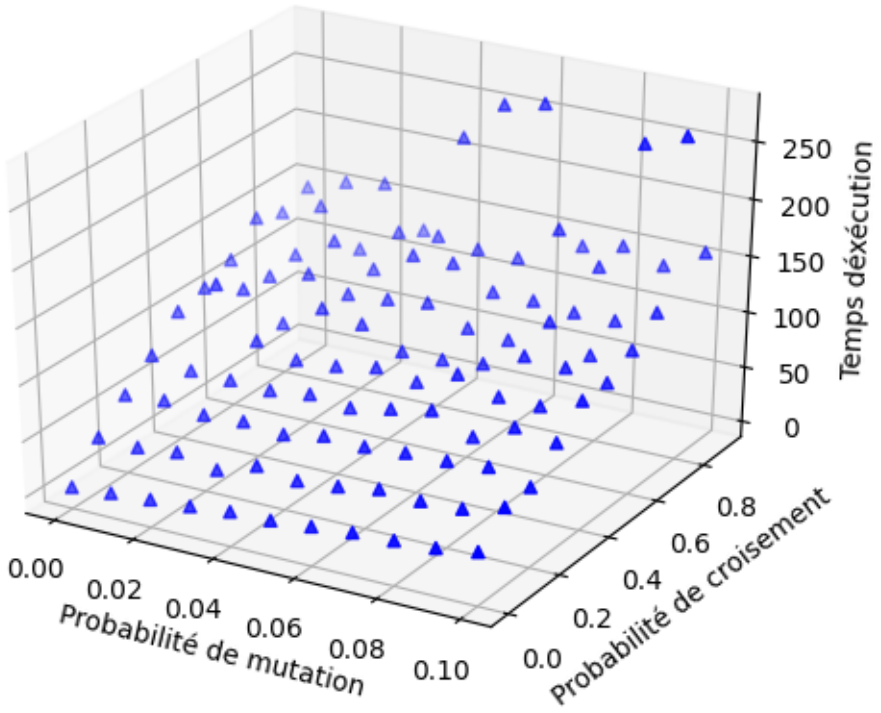

FIGURE 4.11: Variation du temps d'exécution en fonction des probabilité de mutation et de croisement.

Nous nous sommes ensuite penché sur l'influence de la taille de la population initiale sur le temps d'exécution. Nous avons fait varier la taille de la population initiale de 10 à 90 individus avec un pas de 10. Nous avons exécuté l'algorithme 3 fois pour chaque valeur. Les résultats obtenus sont représentés dans le tableau 4.3. Nous pouvons voir dans la figure 4.12 que le temps d'exécution est proportionnel à la taille de la population. Au fil des exécutions, nous avons remarqué que quand la taille de la population est petite, les solutions obtenus ne correspondent pas à nos attentes. Ceci car avec une population réduite les algorithmes ne peuvent faire une exploration complète de l'espace de recherche, ils tombent donc dans des optimum locaux. Pour palier à ce problème nous avons décidé de fixer la taille de la population à 80 individus. Nous avons jugé que les solutions obtenus avec cette taille sont satisfaisantes.

NPOP	Simulation 1	Simulation 2	Simulation 3
10	36.63	35.82	30.80
20	54.19	57.52	56.24
30	88.94	91.78	70.23
40	92.79	91.25	96.32
50	119.26	161.48	123.45
60	140.67	144.08	143.48
70	195.72	251.17	172.69
80	223.56	228.36	176.46
90	238.75	285.22	228.46

TABLE 4.3: Influence de la taille de la population sur le temps d'exécution (en seconde).

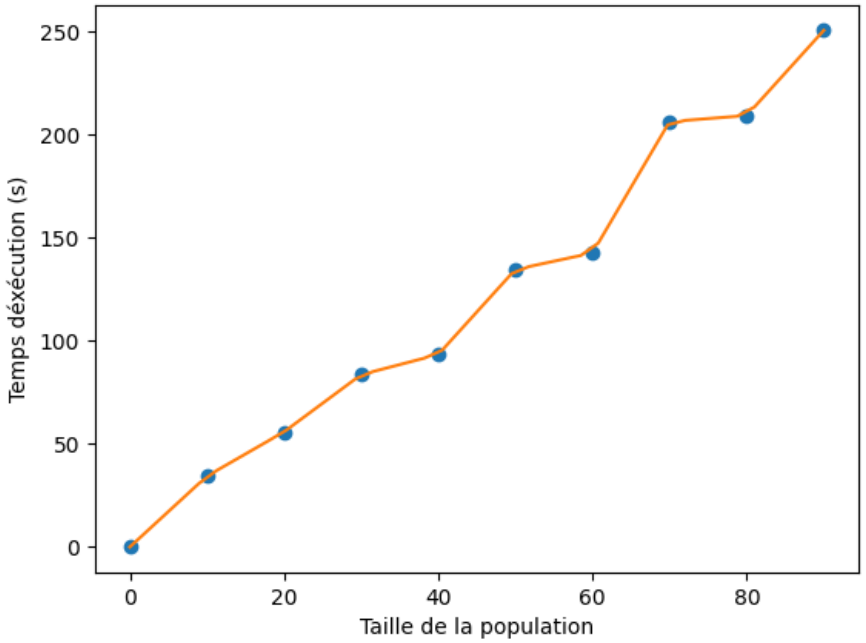

FIGURE 4.12: Temps d'exécution en fonction de la taille de la population.

4.7.3 Déploiement des noeuds routeurs

Afin de trouver un schéma de déploiement des noeuds routeurs, nous avons considéré uniquement 2 objectifs (le coût et la connectivité). Nous avons donc modifié la fonction fitness où nous avons mis 2 containtes au lieu de quatre. Nous avons également modifié la logique de la fonction d'évaluation : nous avons enlevé le calcul des matrices de couverture et de sur-couverture ainsi que le calcul de leurs taux. Le tableau 4.4 regroupe les solutions que nous avons retenues :

Algorithme	Coût (unité)	Connectivité (%)
SPEA-II	3	95.00
NSGA-II	2	100.00
NSGA-III	2	91.25

TABLE 4.4: Solutions retenues pour le déploiement des routeurs.

D'après le tableau, l'lgorithmes NSGA-II donne la meilleur disposition avec seulement 2 routeurs et une connectivité complète.

Algorithme	axe	R1	R2	R3
SPEA-II	x	1	1	7
	y	1	7	7
NSGA-II	x	3	4	/
	y	9	3	/
NSGA-III	x	1	4	/
	y	9	1	/

TABLE 4.5: Coordonnées des routeurs déployés selon l'algorithme utilisé.

4.7.4 Déploiement des noeuds capteurs

Pour établir la disposition des noeuds capteurs, nous lançons l'exécution des trois algorithmes en utilisant les mêmes paramètres (taille de la population = 80 individus, nombre de générations = 100, probabilité de mutation = 0.1 et probabilité de croisement = 0.9). Le tableau 4.6 regroupe les solutions que nous avons retenues :

Algorithme	Coût (unité)	Couverture (%)	Connectivité (%)	sur-couverture (%)
SPEA-II	12	96.25	100.00	12.5
NSGA-II	12	96.25	100.00	11.25
NSGA-III	12	91.25	100.00	13.75

TABLE 4.6: Solutions retenues pour le déploiement des capteurs.

À partir des solutions (chromosomes / individus) données par les algorithmes, nous pouvons extraire les coordonnées des positions des noeuds capteurs à l'aide d'une fonction. Ci après, les emplacements des solutions données dans le tableau 4.6.

Algorithme	axe	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
SPEA-II	x	0	0	0	1	3	3	3	5	6	6	6	6
	y	2	6	9	5	0	3	8	5	1	4	6	9
NSGA-II	x	0	1	1	1	3	4	4	4	6	6	7	7
	y	5	0	2	8	0	2	6	9	0	5	3	8
NSGA-III	x	0	0	1	1	2	3	4	4	6	6	6	7
	y	5	8	0	2	9	6	0	3	3	6	9	1

TABLE 4.7: Coordonnées des capteurs déployés selon l'algorithme utilisé.

4.8 Discussion

Les trois algorithmes utilisés précédemment, à savoir SPEA-II, NSGA-II et NSGA-III, convergent tous les trois au bout de 100 générations vers des solutions optimales. La différence entre les trois méthodes réside dans le temps d'exécution. En effet, l'algorithme NSGA prend plus de temps que les deux autres algorithmes. Lorsqu'il s'agit d'un bâtiment de taille moyenne, on peut choisir l'une des trois méthodes, mais lorsque le bâtiment a une surface importante, il est préférable de choisir NSGA-III ou SPEA-II qui convergent plus rapidement.

En analysant les résultats obtenus lors de la phase de déploiement des routeurs, ainsi que la phase de déploiement des capteurs, nous avons remarqué que la méthode NSGA-II donne le meilleur schéma de déploiement. En effet, avec seulement 2 noeuds

routeurs, il réalise une connectivité complète et avec 12 noeuds capteurs, il obtient un taux de connectivté égale à 100%, un taux de couverture égale à 96.25% et un taux de sur-couverture de seulement 11.25%. La figure 4.13 illustre l'emplacement des différents noeuds donné par l'algorithme NSGA-II.

0	0	0	0	0	S1	0	0	0	0
S2	0	S3	0	0	0	0	0	S4	0
0	0	0	0	0	0	0	0	0	0
S5	0	0	0	0	0	0	0	0	R1
0	0	S6	R2	0	S7	0	0	0	S8
0	0	0	0	0	0	0	0	0	0
S9	0	0	0	0	S10	0	0	0	0
0	0	0	S11	0	0	0	0	S12	0

FIGURE 4.13: Schéma de déploiement obtenu avec NSGA-II.

4.9 Conclusion

Dans ce dernier chapitre, nous avons d'abord présenté les outils et logiciels utilisés pour la résolution de notre problème. Ensuite, nous avons présenté les trois méthodes d'optimisation multi-objectifs sélectionnées dans le chapitre précédent. Finalement, nous avons expliqué notre démarche pour la construction des trois algorithmes.

Afin d'évaluer les performances des trois méthodes, nous avons considéré un cas d'étude. En premier lieu, nous avons défini la meilleure combinaison de paramètre à utiliser. Par la suite, nous avons lancé les simulations pour trouver la solution à notre problème. Les résultats obtenus ont montré que la méthode NSGA-II était plus adaptée à notre cas d'étude.

À la fin, nous retenons de notre travail que les algorithmes génétiques permettent de résoudre des problèmes complexes en un temps réduit et offrent à l'utilisateur une multitude de solutions exploitables. Il peut donc choisir la solution qui lui convient selon la configuration de son espace de déploiement et l'ordre d'importance qu'il donne aux objectifs. Leur inconvénient est qu'ils ont plusieurs paramètres qu'il faut gérer. Il est nécessaire de faire plusieurs essaies avant de pouvoir trouver la bonne combinaison de paramètres qui donne de meilleures performances aux algorithmes.

Conclusion générale et perspectives

Ce projet de fin d'étude avait pour ambition de trouver une méthode de résolution et d'optimisation du problème de déploiement d'un RCSF dans un bâtiment intelligent. Le but était d'optimiser les différentes métriques qui ont une influence sur les performances du réseau, à savoir le coût, la connectivité, la couverture et la sur-couverture. Une application a été développée afin de faciliter cette tâche. Cette application permet de trouver une solution qui répond aux besoins des utilisateurs selon les objectifs fixés.

Afin de résoudre le problème, nous avons commencé par la modélisation des différents critères de déploiement. Nous nous sommes ensuite intéressés aux différentes méthodes de résolution trouvées dans la littérature. Le problème que nous avons traité est considéré comme étant un problème d'optimisation multi-objectifs de complexité NP-difficile. Nous avons donc choisi la méthode la plus adaptée à ce type de problèmes : les algorithmes évolutionnaires. Nous avons sélectionné ces trois variantes : SPEA-II, NSGA-II et NSGA-III. Pour pouvoir appliquer ces trois algorithmes, il a été nécessaire de les adapter au problème de déploiement des RCSFs.

Pour finir, nous avons effectué une série de simulations afin de définir les paramètres des algorithmes utilisés. Ces paramètres sont la probabilité de croisement, la probabilité de mutation et la taille de la population. Les résultats obtenus nous ont permis d'identifier la meilleure méthode à utiliser. Grâce à cette méthode, nous avons pu établir un schéma de déploiement dans un espace type.

Une perspective intéressante serait l'utilisation de modèles réalistes en ce qui concerne la couverture et la connectivité. Ceci engendrera une hausse considérable du temps d'exécution des algorithmes. Ce problème peut être atténué en hybridant les

algorithmes évolutionnaires avec d'autres méthodes telles que les réseaux de neurones ou les algorithmes d'intelligence en essaim. Il serait également utile de développer une interface graphique pour l'outil de déploiement.

Annexes

Annexe A

Code Python NSGA-II

```python
from deap import base, creator, tools
import random
import numpy as np
import matplotlib.pyplot as plt
import time

start = time.time()

# Saisi de l'espace de d ploiment
length = int(input("Entrer la longeur du batiment "))
width = int(input("Entrer la largeur du batiment"))

# Create the multi-objective fitness function: min cost , max coverage, max connectivity, min over-coverage.
creator.create("fitnessmulti", base.Fitness, weights=(-1.0, 1.0, 1.0, -1.0))

# Create individuals
creator.create("individual", list, fitness=creator.fitnessmulti)

IND_SIZE = width * length  # Size of the individual = Width * Lenght

toolbox = base.Toolbox()

toolbox.register("attr_bool", random.randint, 0, 1)
```

```
toolbox.register("individual", tools.initRepeat, creator.individual,
    toolbox.attr_bool, n=IND_SIZE)

# Create the population
toolbox.register("population", tools.initRepeat, list, toolbox.
    individual)


# Create the evaluation function
def evaluate(individual):
    col1 = length
    indiv_m = np.array([individual[i:i + col1] for i in range(0, len(
    individual), col1)])  # Convert list into a matrix
    position = np.where(indiv_m == 1)  # Posistion of sensors
    x = position[0]
    y = position[1]
    rayon_cov = 1.5 # coverage radius
    rayon_con = 3  # connectivity_radius
    coverage = np.zeros((width, length))  # calculate the coverage
    matrix including overcoverage
    for z in range(len(x)):
        for i in range(width):
            for j in range(length):
                d = np.sqrt(((i - x[z]) ** 2) + ((j - y[z]) ** 2))
                if d <= rayon_cov:
                    coverage[i, j] += 1
    cov = np.zeros((width, length))  # calculate the coverage matrix
    for h in range(width):
        for k in range(length):
            if coverage[h, k] >= 1:
                cov[h, k] = 1
    vect_a = np.ravel(cov)
    connectivity = np.zeros((width, length))  # calculate the
    connectivity matrix
    for m in range(len(x)):
        for n in range(width):
            for p in range(length):
                dis = np.sqrt(((n - x[m]) ** 2) + ((p - y[m]) ** 2))
```

```
57                 if dis <= rayon_con:
58                     connectivity[n, p] = 1
59     vect_b = np.ravel(connectivity)
60     over_coverage = np.zeros((width, length))  # calculate the over
       coverage matrix
61     for q in range(width):
62         for s in range(length):
63             if coverage[q, s] >= 2:
64                 over_coverage[q, s] = 1
65     vect_c = np.ravel(over_coverage)
66     a = (sum(individual))  # Cost
67     b = ((sum(vect_a)) / IND_SIZE) * 100  # The sensing coverage rate
       function
68     c = ((sum(vect_b)) / IND_SIZE) * 100  # The connectivity rate
       function
69     d = ((sum(vect_c)) / IND_SIZE) * 100  # The overcoverage rate
       function
70     print("cost = ", a, "cov = ", b, "%", "con = ", c, "%", "over_cov =
       ", d, "%")
71     return a, b, c, d
72
73
74 # D finition of the operators
75 toolbox.register("mate", tools.cxOnePoint)
76 toolbox.register("mutate", tools.mutFlipBit, indpb=0.1)
77 toolbox.register("select", tools.selNSGA2)
78 toolbox.register("evaluate", evaluate)
79
80
81 # Algorithms
82 # ----------
83
84 def main(seed=None):
85     random.seed(seed)
86     # create an initial population of individuals (where
87     # each individual is a list of integers)
88     NPOP = 80
89     pop = toolbox.population(n=NPOP)
```

```
90
91      # CXPB  is the probability with which two individuals are crossed
92      # MUTPB is the probability for mutating an individual
93      CXPB, MUTPB = 0.9, 0.1
94
95      print("Start of evolution")
96
97      # Evaluate the entire population
98      fitnesses = list(map(toolbox.evaluate, pop))
99      for ind, fit in zip(pop, fitnesses):
100         ind.fitness.values = fit
101
102     print("  Evaluated %i individuals" % len(pop))
103
104     # Variable keeping track of the number of generations
105     g = 0
106
107     # Begin the evolution
108     while g < 100:
109
110         # A new generation
111         g = g + 1
112         print("-- Generation %i --" % g)
113
114         # Select the next generation individuals
115         offspring = toolbox.select(pop, len(pop))
116         # Clone the selected individuals
117         offspring = list(map(toolbox.clone, offspring))
118
119         # Apply crossover and mutation on the offspring
120         for child1, child2 in zip(offspring[::2], offspring[1::2]):
121
122             # cross two individuals with probability CXPB
123             if random.random() < CXPB:
124                 toolbox.mate(child1, child2)
125
126                 # fitness values of the children
127                 # must be recalculated later
```

```
128                 del child1.fitness.values
129                 del child2.fitness.values
130
131         for mutant in offspring:
132
133             # mutate an individual with probability MUTPB
134             if random.random() < MUTPB:
135                 toolbox.mutate(mutant)
136                 del mutant.fitness.values
137
138         # Evaluate the individuals with an invalid fitness
139         invalid_ind = [ind for ind in offspring if not ind.fitness.valid
    ]
140         fitnesses = map(toolbox.evaluate, invalid_ind)
141         for ind, fit in zip(invalid_ind, fitnesses):
142             ind.fitness.values = fit
143
144         print("  Evaluated %i individuals" % len(invalid_ind))
145
146         # The population is entirely replaced by the offspring
147         pop[:] = toolbox.select(pop + offspring, NPOP)
148         print("pop", pop)
149
150     print("-- End of (successful) evolution --")
151
152     best_ind = tools.selNSGA2(pop, NPOP, nd='standard')
153
154     for i in range(len(best_ind)):
155         print("Best individual is %s, %s" % (best_ind[i], best_ind[i].
    fitness.values))
156
157     end = time.time()
158
159     print(f"Runtime of the program is {end - start} s ")
160
161     best_ind1 = best_ind[0]
162     col = length  # number of columns of the deployment matrix
163
```

```
164     # Convert the list (individual) into a matrix
165     best_ind_matrix = np.array([best_ind1[i:i + col] for i in range(0,
        len(best_ind1), col)])
166
167     B = np.where(best_ind_matrix == 1)
168     print("B", B)
169     X = B[0]
170     Y = B[1]
171     print(X)
172     print(Y)
173     plt.grid(color='k', linestyle='-', linewidth=0.5)
174     plt.scatter(X, Y, s=80, cmap="Blues", alpha=0.4, edgecolors="grey",
        linewidth=2)
175
176     plt.xlabel("Width")
177     plt.ylabel("Lenght")
178     plt.xlim(-1, width)
179     plt.ylim(-1, length)
180     plt.title("Wireless Sensor Network deployment plan")
181     plt.show()
182
183
184     print("Thank you")
185
186 if __name__ == "__main__":
187     main()
```

Bibliographie

[1] Ali Afghantoloee and Mir Abolfazl Mostafavi. Towards optimal deployment of a sensor network in a 3d indoor environment for the mobility of people with disabilities (short paper). In *10th International Conference on Geographic Information Science (GIScience 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[2] Nadjib Aitsaadi, Nadjib Achir, Khaled Boussetta, and Guy Pujolle. Artificial potential field approach in wsn deployment : Cost, qom, connectivity, and lifetime constraints. *Computer Networks*, 55(1) :84–105, 2011.

[3] Vahab Akbarzadeh, Julien-Charles Lévesque, Christian Gagné, and Marc Parizeau. Efficient sensor placement optimization using gradient descent and probabilistic coverage. *Sensors (Basel, Switzerland)*, 14 :15525–52, 08 2014.

[4] J. Amutha, Sandeep Sharma, and Jaiprakash Nagar. Wsn strategies based on sensors, deployment, sensing models, coverage and energy efficiency : Review, approaches and open issues. *Wireless Personal Communications*, 111(2) :1089–1115, 2020.

[5] Lionel ANCIAUX. Smart building : c'est quoi ?, 2018.

[6] Đurišić Milica Pejanović and al. "a survey of military applications of wireless sensor networks". In *2012 Mediterranean Conference on Embedded Computing (MECO)*, pages 196–199, 2012.

[7] Alain Anfosso and Stéphane Rebaudo. Gérontechnologies et contrôle de l'environnement au service du maintien à domicile : le projet gerhome. *Gérontologie et société*, 34 / 136(1) :119–131, 2011.

[8] T. Arampatzis, John Lygeros, and Stamatis Manesis. A survey of applications of wireless sensors and wireless sensor networks. *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, pages 719–724, 2005.

[9] Constantine A Balanis. *Antenna theory : analysis and design*. John wiley & sons, 2016.

[10] Abderrahmen Belfkih, Bruno Sadeg, Claude Duvallet, and Laurent Amanton. Les bases de données dans les réseaux de capteurs sans fil. *Techniques et sciences informatiques*, 33 :739–776, 12 2014.

[11] Hervé Belmonte. Étude, implémentation et évaluation d'un réseau de capteurs sans fil exploitant le concept de colportage de l'information. Master's thesis, June 2011.

[12] Mohamed Amin Benatia. *Multi-objective optimization of a network infrastructure dedicated to smart buildings*. Theses, INSA de Rouen, December 2016.

[13] Mohamed Amin Benatia, M'hammed Sahnoun, David Baudry, Anne Louis, Abdelkhalak El-Hami, and Belahcene Mazari. Multi-objective wsn deployment using genetic algorithms under cost, coverage, and connectivity constraints. *Wireless Personal Communications*, 94(4) :2739–2768, 2017.

[14] Jet Brains. Pycharm : L'edi python pour développeurs professionnels, 2020.

[15] Jason Brownlee. *Clever algorithms : nature-inspired programming recipes*. Jason Brownlee, 2011.

[16] Renaud Chaudoir. La construction intelligente ou « smart building », 2019.

[17] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. "an investigation of some properties of an ant algorithm.". In *Ppsn*, volume 92, 1992.

[18] Charles Darwin. On the origin of species by means of natural selection. 1859. *London : Murray Google Scholar*, 1968.

[19] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i : solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4) :577–601, 2013.

[20] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2) :182–197, 2002.

[21] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[22] S. C. Ergen. Zigbee/ieee 802.15.4 summary. 2004.

[23] M. Farsi, M. A. Elhosseini, M. Badawy, H. Arafat Ali, and H. Zain Eldin. Deployment techniques in wireless sensor networks, coverage and connectivity : A survey. *IEEE Access*, 7 :28940–28954, 2019.

[24] Soumaya FELLAH. *Optimisation Multi-objectif appliquée au déploiement et à la performance des réseaux de capteurs sans fil*. Theses, Université d'Oran1 - Ahmed Ben Bella, 2018.

[25] Lawrence J Fogel, Alvin J Owens, and Michael J Walsh. Artificial intelligence through simulated evolution. 1966.

[26] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP : Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13 :2171–2175, jul 2012.

[27] Vincent Gardeux. *Heuristics implementation for high-dimensional problem optimization : application in microarray data analysis*. Theses, Université Paris-Est, November 2011.

[28] Amitabha Ghosh and Sajal K Das. Coverage and connectivity issues in wireless sensor networks : A survey. *Pervasive and Mobile Computing*, 4(3) :303–334, 2008.

[29] Fred Glover. Future paths for integer programming and links to ar tifi cial intelli g en ce. *Computers operations research*, 13(5) :533–549, 1986.

[30] Jaap C. Haartsen. The bluetooth radio system. *IEEE Personal Communications*, 7 :28–36, 2000.

[31] Subash Harizan and Pratyay Kuila. *Evolutionary Algorithms for Coverage and Connectivity Problems in Wireless Sensor Networks : A Study*, pages 257–280. Springer Singapore, Singapore, 2020.

[32] John Holland. Adaptation in natural and artificial systems : an introductory analysis with application to biology. *Control and artificial intelligence*, 1975.

[33] Christophe Jacquet, Yacine Bellik, René Farcy, and Yolaine Bourda. Aides électroniques pour le déplacement des personnes non-voyantes : Vue d'ensemble et perspectives. volume 386, pages 93–100, 08 2004.

[34] RAPHAËLLE JEREZ-GRISEL. 5 exemples de smart building, 2018.

[35] Pragati Kapil and Shashi Lata. Review on selecting topologies in zigbee networks. 2016.

[36] G. Kaur and R. M. Garg. Energy efficient topologies for wireless sensor networks. *International Journal of Distributed and Parallel systems (IJDPS)*, 3(5) :179–192, 9 2012.

[37] Harsimran Kaur and Rohit Bajaj. Review on localization techniques in wireless sensor networks. *International Journal of Computer Applications*, 116 :4–7, 04 2015.

[38] Siwiak Kazimierz. *Ultra-Wideband Radio*. John Wiley Sons, Ltd, 2005.

[39] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598) :671–680, 1983.

[40] John R Koza. *Genetic programming : A paradigm for genetically breeding populations of computer programs to solve problems*, volume 34. Stanford University, Department of Computer Science Stanford, CA, 1990.

[41] Sushil Kumar and DK Lobiyal. Sensing coverage prediction for wireless sensor networks in shadowed and multipath environment. *The Scientific World Journal*, 2013, 2013.

[42] M. Lott and I. Forkel. A multi-wall-and-floor model for indoor radio propagation. In *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202)*, volume 1, pages 464–468 vol.1, 2001.

[43] Mohammad Matin. *Wireless Sensor Network : Technology and Protocols*. INTECH, 09 2012.

[44] Usama Mehboob, Junaid Qadir, Salman Ali, and Athanasios Vasilakos. Genetic algorithms in wireless networking : techniques, applications, and issues. *Soft Computing*, 20(6) :2467–2501, 2016.

[45] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247) :335–341, 1949.

[46] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4) :308–313, 1965.

[47] Hanh Nguyen, Nam Nguyen Hai, and Huynh Binh. Particle swarm optimization algorithms for maximizing area coverage in wireless sensor networks. pages 893–904, 09 2018.

[48] Hunn Nick. An introduction to wibree. *White paper*, 2006.

[49] Vilfredo Pareto. *Cours d'économie politique : professé à l'Université de Lausanne*, volume 1. F. Rouge, 1896.

[50] Tom Randall. The smartest building in the world :inside the connected futur», 2015.

[51] Ingo Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Library Translation 1122*, 1965.

[52] Hua Su, Gaoyong Wang, Xuemei Sun, and Dong Yu. Optimal node deployment strategy for wireless sensor networks based on dynamic ant colony algorithm. *International Journal of Embedded Systems*, 8(2-3) :258–265, 2016.

[53] Intent Technologies. Les enjeux et bénéfices du smart building des bâtiments connectés, 2020.

[54] Jingwen Tian, Meijuan Gao, and Guangshuang Ge. Wireless sensor network node optimal coverage based on improved genetic algorithm and binary ant colony algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2016(1) :1–11, 2016.

[55] Zhibo Wang. *Barrier Coverage in Wireless Sensor Networks*. Theses, University of Tennessee, 2014.

[56] Mohamed Younis and Kemal Akkaya. Strategies and techniques for node placement in wireless sensor networks : A survey. *Ad Hoc Networks*, 6(4) :621 – 655, 2008.

[57] Yinggao Yue, Li Cao, and Zhongqiang Luo. Hybrid artificial bee colony algorithm for improving the coverage and connectivity of wireless sensor networks. *Wireless Personal Communications*, 108(3) :1719–1732, 2019.

[58] Hanaa ZainEldin, Mahmoud Badawy, Mostafa Elhosseini, Hesham Arafat, and Ajith Abraham. An improved dynamic deployment technique based-on genetic algorithm (iddt-ga) for maximizing coverage in wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2020.

[59] Hanaa Zaineldin, Mostafa El-Hosseini, and Hesham Ali. A modified listless strip based spiht for wireless multimedia sensor networks. *Computers Electrical Engineering*, 12 2015.

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

[61] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization based on virtual forces. volume 2, pages 1293 – 1303 vol.2, 03 2003.