ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس - الرباط -
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre 3076

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche : Laboratoire de Recherche en Informatique et Télécommunications
Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Télécommunications

Présentée et soutenue le : 3 mars 2018 par :

Asmae EL GHAZI

Contribution au routage dans les réseaux de capteurs sans fil en utilisant les métaheuristiques

JURY

Mohamed OUADOU	*PES, Université Mohammed V Faculté des Sciences de Rabat*	*Président*
Salma MOULINE	*PES, Université Mohammed V Faculté des Sciences de Rabat*	*Examinateur*
Mohammed RZIZA	*PES, Université Mohammed V Faculté des Sciences de Rabat*	*Rapporteur*
Mohammed ABBAD	*PES, Université Mohammed V Faculté des Sciences de Rabat*	*Directeur de thèse*
Belaïd AHIOD	*PH, Université Mohammed V Faculté des Sciences de Rabat*	*Encadrant*
Ouadoudi ZYTOUNE	*PH, Université Ibn Tofail ENCG de Kenitra*	*Rapporteur*
Aziz OUAARAB	*PA, Université Cadi Ayyad EST d'Essaouira*	*Invité*

Années Universitaire : 2017/2018

Je dédie ce travail

A

Ma chère mère

...

A

La mémoire de mon père

...

A

Mes chères sœurs et mes chers frères

...

A toute la famille

...

A

A mes amis et collègues

Avec tous mes souhaits de bonheur, de santé et de réussite.

Et enfin à toute personne ayant contribué de près ou de loin à la réalisation de ce travail.

Avant-Propos

Les travaux présentés dans ce mémoire sont effectués au Laboratoire de Recherche en Informatique et Télécommunications (LRIT), à la Faculté des Sciences de Rabat, sous la direction du Professeur Mohammed ABBAD et le co-encadrement du Professeur Belaïd AHIOD

Je commence par présenter ma plus vive gratitude à mon directeur de thèse Pr. Mohammed ABBAD, Professeur de l'Enseignement Supérieur à la Faculté des Sciences de Rabat. Grâce à ses encouragements, sa pédagogie et ses précieux conseils, il a su me guider pour mener à bien mes travaux de recherche. J'exprime ici ma profonde gratitude à son égard et l'estime respectueuse que je lui porte.

Je tiens à exprimer mes remerciements à mon co-encadrant, Pr. Belaïd AHIOD, Professeur Habilité à la Faculté des Sciences de Rabat, pour ces années de soutien, pour ses précieux conseils scientifiques et humains, ainsi que pour ses encouragements.

Je veux exprimer toute ma reconnaissance et ma gratitude à Pr. Mohamed OUADOU, Professeur de l'Enseignement Supérieur à la Faculté des Sciences de Rabat et directeur du laboratoire LRIT d'avoir accepté de présider le jury de ma thèse.

Je tiens également à exprimer mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse. Merci à Pr. Salma MOULINE, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat d'avoir accepté d'examiner mon travail.

Je remercie aussi Pr. Mohammed RZIZA, Professeur de l'Enseignement Supérieur et chef de département d'informatique à la Faculté des Sciences de Rabat, d'avoir accepté de rapporter ce travail et pour le temps qu'il a consacré pour lire et évaluer mon travail.

Je souhaite remercier profondément Pr. Ouadoudi ZYTOUNE, Professeur Habilité à l'Université Ibn Tofail de Kénitra, d'être un rapporteur de ce manuscrit. Je le remercie pour le temps qu'il a consacré pour lire et évaluer mon travail.

Je voudrais également remercier Pr. Aziz OUAARAB, Professeur Assistant, Université Cadi Ayyad de Marrakech, pour avoir accepté d'examiner ce travail et de se déplacer pour participer au jury.

A l'issue de la rédaction de cette recherche, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur. Je souligne le soutien amical et chaleureux de tous les doctorants du Laboratoire LRIT, qui ont

croisé ma route durant ce parcours doctoral et, plus particulièrement, à tous ceux qui m'ont soutenu et encouragé. Un merci spécial à mes chères amies Salma EL FELLAH, Maryam LAFKIH, Salwa LAGDALI et Zineb AARAB pour les moments magnifiques et inoubliables que nous avons passé ensemble, pour leur sympathie, leur bonne humeur permanente, leur énergie positive.

Finalement, mais pas pour autant moins important à mes yeux, je voudrais témoigner tout mon amour et ma reconnaissance à ma chère famille, à qui je dédie cette thèse. Je tiens à remercier chaudement ma mère, mes chers frères Abdellatif, Yacine, Jalil, Anass et mes chères sœurs Maryam et Nadia, pour leurs encouragements et leur assistance aussi bien matérielle que morale qui m'ont permis de faire cette thèse dans de bonnes conditions, je ne saurais jamais être reconnaissante pour toute leur confiance en moi, leur affection, leurs conseils, mais aussi leur soutien indéfectible. Merci à vous, même si je ne pourrai jamais vous remercier assez.

Résumé

Dans les Réseaux de Capteurs Sans Fil (RCSF), plusieurs problèmes d'optimisation sont considérés NP-difficiles, tels que le problème de couverture, d'ordonnancement et de routage. De tels problèmes nécessitent des solutions de bonne qualité dans un délai acceptable, d'où l'utilisation des métaheuristiques. Dans cette thèse, nous nous intéressons particulièrement au routage, puisque ce problème présente un des grands défis des RCSF. Dans cet ordre d'idées, nous avons proposé des protocoles de routage basés sur différentes métaheuristiques inspirées de la nature. Les travaux effectués durant cette thèse ont mené à trois contributions principales :

La première contribution consiste à améliorer une approche basée sur l'algorithme de colonie de fourmis (Ant Colony Optimization (ACO)) afin de concevoir une nouvelle. Comparée à l'approche de base, la nouvelle approche ACO proposée a minimisé la consommation énergétique et a prolongé la durée de vie d'un RCSF, selon les résultats trouvés.

La deuxième contribution se focalise sur des études comparatives entre l'approche ACO et une autre basée sur l'optimisation par essaim de particules (Particle Swarm Optimization (PSO)). Ainsi, nous avons évalué l'impact de la mobilité aléatoire sur ces deux protocoles, dans des RCSF composés de nœuds dynamiques suivant le modèle de mobilité Random Waypoint (RWP).

Dans la troisième contribution, nous avons proposé un nouveau protocole de routage basé sur une nouvelle métaheuristique nommé Teaching Learning Based Optimization (TLBO) développée récemment pour les problèmes d'optimisation continus. Nous avons pu adapter cette métaheuristique au problème de routage qui est un problème discret. L'adaptation est basée sur la redéfinition des équations tout en gardant l'aspect général de la méthode qui se base sur la relation enseignant-étudiants et les interactions entre les étudiants.

Afin de valider nos travaux effectués et tester les performances de notre protocole de routage vis-à-vis des autres protocoles dans des simulations réelles, nous sommes en cours de réaliser une application de surveillance réelle utilisant un RCSF composé de capteurs Arduino.

Mots-clés : Réseaux de capteurs sans fil ; Optimisation combinatoire ; Métaheuristiques inspirées de la nature ; Algorithme de colonie de fourmis ; Algorithme d'essaim de particules ; Optimisation basée sur Teaching-Learning ; Routage ; Arduino.

Abstract

Many problems in Wireless Sensor Network (WSN) are qualified as NP-hard optimization problems, requiring robust and efficient techniques to solve this kind of problems such as metaheuristics. In this thesis work, we chose to study the routing problems. Hence, we proposed many routing protocols based on different metaheuristics inspired by nature.

The first contribution is to improve an approach based on the ant colony optimization (ACO) by modifying the probabilistic decision rule, that gives rise to a new ACO approach better than the original one in the energy consumption and the WSN's lifetime.

The second contribution focuses on comparative studies between two routing protocols based on ACO and PSO in different scenarios and studied cases. Thus, we evaluated the impact of the random waypoint (RWP) on these two protocols.

In the third contribution, we propose a new routing protocol based on Teaching-learning based optimization (TLBO), a metaheuristic recently developed for continuous optimization problems. We have adapted TLBO to the discrete routing problem by the redefinition of its equations using the Edge Recombination Operator (ERO). The TLBOR protocol is better than many other state-of-the-art routing protocols in terms of energy consumption and network lifetime.

Finally, in order to highlight our proposed contribution, we have performed a real application in order to validate our research and test the performance of our routing protocol against other protocols in real environment using a WSN composed by Arduino sensors.

Keywords : Wireless Sensor Networks ; Combinatorial Optimization ; Metaheuristics inspired by nature ; Ant Colony Optimization ; Particle Swarm Optimization ; Teaching-Learning Based Optimization ; Routing ; Arduino.

Table des matières

Liste des notations et abréviations

RCSF Réseaux de Capteurs Sans Fil
RWP Random Waypoint
ACO Ant Colony Optimization
PSO Particle Swarm Optimization
TLBO Teaching Learning Based Optimization
WSN Wireless Sensor Network
ERO Edge Recombination Operator
IACOR Improved Ant Colony Optimization for Routing
TLBOR Teaching-Learning Based Optimization based Routing
QoS Quality of Service
LEACH Low-energy Adaptive Clustering Hierarchy
CBRP Cluster Based Routing Protocol
SPIN Sensor Protocols for Information via Negotiation
GPS Global Positioning System
GPSR Greedy Perimeter Stateless Routing
CH Cluster Head
CDS Connected Dominating Set
DS Dominating Set
MCDS Minimum Connected Dominating Set
SMET Strong Minimum Energy Topology
TDMA Time Division Multiple Access
MAC Medium Access Control
RO Recherche Opérationnelle
IA Intelligence Artificielle
EEABR Energy Efficient Ant- Based Routing
HS Harmony Search
IHSBEER Improved Harmony Search Based Energy-Efficient Routing
SC Sensor-driven Cost-aware ant routing
FF Flooded Forward ant routing
FP Flooded Piggy-backed ant routing
ADR Adaptive ant-based Dynamic Routing

AR Adaptive Routing
IAR Improved Adaptive Routing
GA Genetic Algorithms
BCO Bee Colony Optimization
ABC Artificial Bee Colony
CS Cuckoo Search

Liste des figures

Liste des tableaux

Liste des algorithmes

INTRODUCTION GÉNÉRALE

Contexte

Nous vivons aujourd'hui dans un monde très connecté. En effet, les progrès de l'électronique et de nouvelles technologies ont permis l'émergence de nouveaux moyens de communication. D'abord totalement statiques, à cause des besoins d'infrastructures adaptées, puis portables et mobiles avec la généralisation des technologies sans fil. Les technologies continuent d'évoluer et proposent depuis quelque temps déjà des réseaux sans infrastructure fixe, ou réseaux ad-hoc. Ces nouveaux réseaux ont permis l'émergence de nouvelles applications difficilement envisageables autrement, comme les réseaux véhiculaires, les systèmes de surveillance sans fil, etc.

Le monde de demain sera encore plus connecté, parmi les réseaux ad-hoc qui émergent, on distingue les réseaux de capteurs sans fil (RCSF). Ce sont des réseaux dont les éléments constitutifs, les capteurs, sont particulièrement contraints. Chaque capteur consiste en un petit système embarqué aux ressources matérielles très limitées. Ces réseaux sont dès lors généralement employés pour la détection dans des situations critiques ou dangereuses. L'importance des RCSF a été initialement remarquable pour les applications militaires, mais leurs propriétés en font des solutions pratiques dans de nombreux domaines de notre vie courante notamment, la surveillance des environnements hostiles, la surveillance de santé, l'agriculture, etc.

Dans un RCSF, les nœuds communiquent via des fréquences radio et peuvent s'auto-organiser et coopérer pour répondre à des besoins et fournir des services. Chaque nœud doit être en mesure de capturer les grandeurs physiques avoisinantes, de traiter des données reçues, de prendre une décision locale et de la communiquer de façon autonome avec ses voisins auxquels il est connecté. Cette coopération est destinée à assurer les meilleures prises de décision possible malgré les limites des nœuds en termes de consommation énergétique et de puissance de traitement.

En effet, les RCSF sont soumis à des contraintes fortes de natures multiples, énergétiques et calculatoires, ce qui limite les capacités de traitement et de communication des nœuds du réseau. De ce fait, de nouveaux problèmes complexes dans les domaines de la recherche opérationnelle et de l'optimisation sont apparus. Parmi les problèmes fondamentaux des réseaux de capteurs sans fil on trouve les problèmes liés au routage, la couverture, le contrôle de topologie, la mobilité et l'ordonnancement. Néanmoins, le grand défi reste les problèmes liés au routage. Jusqu'à présent, le besoin d'acheminer l'information d'une manière efficiente et de réduire au minimum la consommation d'énergie avec les contraintes des RCSF peut s'exprimer comme un problème d'optimisation. Il s'agit du problème de routage qui a fait objet de la plupart des recherches d'optimisation.

La solution optimale d'un problème d'optimisation ne peut que très rarement être déterminée en un temps polynomial. Il est donc souvent nécessaire de trouver des méthodes de résolution qui fournissent une solution de bonne qualité dans un temps raisonnable, c'est ce que font les métaheuristiques depuis une vingtaine d'années. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum d'une fonction objectif. Ces méthodes s'adaptent aux caractéristiques du problème étudié afin d'en trouver une approximation de la meilleure solution. Il existe un grand nombre de métaheuristiques allant de la simple recherche locale à des algorithmes complexes de recherche globale. Cependant, elles utilisent un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

Dans cette thèse, nous nous intéressons aux problèmes d'optimisation liés au routage dans les RCSF en essayant de proposer des algorithmes de résolutions considérés comme inspirés de la nature. Dans le but de minimiser le délai de transmission et la consommation d'énergie en plus d'améliorer grandement le fonctionnement des nœuds, nous avons décidé de faire face à ces problèmes d'optimisation qui sont généralement difficiles à résoudre. Notre objectif principal sera donc d'étudier le problème de routage dans les réseaux de capteurs en utilisant plusieurs métaheuristiques inspirées de la nature permettant de préserver l'énergie et d'acheminer les paquets de la manière la plus optimale qui puisse être.

Problématique

La mise en œuvre des traitements, de stockage, de détection et de communication dans des dispositifs de petite taille, à faible coût et leur intégration dans les RCSF ouvre la porte à une multitude de nouvelles applications. Dans certains types d'applications, un grand nombre de capteurs est susceptible d'être déployé, sur de vastes terrains. Dans ce cas, un grand nombre de flots sont transmis au centre de contrôle (station de base, ou SINK) pour l'analyse et la prise de décision. Il y a donc un besoin important en termes de bande passante, avec surtout une forte contrainte en termes de délai de transmission. Sans oublier de compter, la conservation de l'énergie des capteurs et le maintien du réseau fonctionnel le plus longtemps possible. Pour ces raisons, l'objectif principal de notre travail est d'utiliser ces dispositifs avec efficacité. Cette efficacité est d'une importance vitale étant donné que ces capteurs fonctionnent généralement avec des piles, et que ces piles ne peuvent pas être remplacées ou rechargées dans la plupart des cas.

Les techniques conçues pour les réseaux ad hoc traditionnels ne sont pas bien adaptées aux réseaux de capteurs sans fil, car de nombreuses contraintes doivent être résolues. Dans cette thèse, nous nous intéressons aux contraintes imposées par le routage et par la gestion de l'énergie dans ce type de réseaux afin de prolonger sa durée de vie. Pour cela, notre objectif est de proposer des algorithmes de routage basés sur la conservation de l'énergie en faisant participer des nœuds ayant des batteries pleines par rapport à leurs voisins tout en évitant les zones pauvres en énergie ou en capteurs.

Nos algorithmes sont basés sur les métaheuristiques inspirées de la nature afin de trouver une nouvelle manière de router l'information. L'idée principale de notre travail est de concevoir des protocoles de routage basés sur les métaheuristiques inspirées de la nature (colonies de fourmis, essaim de particules et l'enseignement/apprentissage) pour minimiser la consommation énergétique des capteurs.

Contributions

Dans ce mémoire de thèse, notre contribution se concentre sur l'implémentation des algorithmes de routage dont l'objectif est de traiter le problème de gestion de ressources énergétiques et garantir les qualités de service afin de maximiser la durée de vie du réseau et son efficacité. Pour cela, nous avons proposé des algorithmes basés sur la fusion de deux aspects :

— le routage avec qualité de service dans les RCSF ;
— les métaheuristiques inspirées de la nature ;

Les contributions de cette thèse s'articulent autour de deux axes qui sont :

— L'implémentation et l'amélioration des approches basées sur les métaheuristiques les plus utilisées dans le traitement des problèmes de routage dans les RCSF. Ces métaheuristiques sont les algorithmes de colonies de fourmis (ACO) et les algorithmes d'essaim de particules (PSO). Nous avons proposé une approche basée sur ACO intitulée Improved Ant Colony Optimization for Routing (IACOR) (El Ghazi *et al.*, 2014) dont le principe est de considérer le nœud collecteur (Sink) comme une source de nourriture des fourmis et le nœud source d'évènement comme le nid. Les fourmis portent les données à acheminer et choisissent les nœuds voisins qui vont participer au routage des paquets à l'aide d'une règle probabiliste et de la nouvelle métrique que nous avons définies. Les résultats de cette approche ont était satisfaisants en comparaison avec d'autres basées sur ACO. Dans le but de découvrir les limites de cette méthode, nous l'avons comparé a une approche de routage basée sur l'optimisation par essaim de particules (PSO) (EL Ghazi et Ahiod, 2016) dans des réseaux de capteurs sans fil statiques et mobiles (El Ghazi et Ahiod, 2016).
— L'adaptation de la nouvelle métaheuristique Teaching-Learning Based Optimization (TLBO) (qui a était mise en œuvre pour les problèmes continus) au problème de routage discret afin de réaliser un nouveau protocole de routage dans les RCSF "Teaching-Learning Based Optimization based Routing (TLBOR)" (El Ghazi et Ahiod, 2017). Malgré les bons résultats fournis par des métaheuristiques, il y a toujours des problèmes liés à la complexité d'adaptation et au paramétrage. TLBO est une méthode récente, robuste, ne nécessitant aucun paramétrage et simple d'implémentation. TLBO a été proposée pour des problèmes d'optimisation continue, dans ce travail, cette métaheuristique a été adaptée au problème de routage dans les RCSF qui est un problème discret. L'approche proposée nommée TLBOR est bien fondée théoriquement et détaillée algorithmiquement.

Organisation de la thèse

Cette thèse se décompose en quatre chapitres :

Dans le premier chapitre, nous définissons les réseaux de capteurs sans fil, leurs architectures, leurs contraintes de conception (tolérance aux pannes, consommation d'énergie, qualité de service, etc), leurs topologies, leurs types et leurs applications.

Le deuxième chapitre introduit les différents problèmes d'optimisation des RCSF : les problèmes de couverture, de contrôle de la topologie et d'ordonnancement qui nécessitent des algorithmes robustes et performants. Ensuite, on met l'accent sur les méthodes de résolution des problèmes d'optimisation combinatoires : méthodes exactes, méthodes

approchées et métaheuristiques inspirées de la nature. Finalement, on présente le problème de routage dans les RCSF, des protocoles de routage et des approches basés sur les métaheuristiques inspirées de la nature.

Dans le troisième chapitre, on présente la première contribution qui est une amélioration d'un protocole de routage basé sur les algorithmes de colonie de fourmis, commençant par l'adaptation d'ACO au problème de routage, description du fonctionnement de l'approche et finalement l'implémentation et les résultats. Ensuite, on propose une étude comparative entre l'approche ACO proposée et une approche PSO dans des RCSF statiques et mobiles.

Le quatrième chapitre présente un nouveau protocole de routage basé sur la méthode TLBO. Ainsi que l'ensemble des étapes d'adaptation, redéfinition des équations et les résultats trouvés qui montre la supériorité de cette approche par rapport aux plusieurs protocoles de l'état de l'art.

Chapitre

1

LES RÉSEAUX DE CAPTEURS SANS FIL

Sommaire

1.1 Introduction

De nombreuses techniques et technologies dans les domaines de la micro-électronique, de la micro-mécanique, et des technologies de communication sans fil ont permis de créer, avec un coût raisonnable, de petits objets communicants équipés de capteurs. Ces nouveaux objets, appelés nœuds capteurs ou couramment capteurs, sont équipés d'unités de capture, de calcul, de mémorisation et de communication. Pour leur alimentation en courant, ces nœuds sont munis de batteries ou d' un système de récupération d'énergie à partir de l'environnement. De ce fait, les nœuds capteurs sont de véritables entités embarquées. Le déploiement de ces entités, en vue de collecter et transmettre des données environnementales vers un ou plusieurs points de collecte, de manière autonome, forme un réseau de capteurs sans fil (RCSF).

Dans ce chapitre, nous faisons un état de l'art sur les réseaux de capteurs sans fil. Nous y présentons les définitions de base concernant la structure du réseau, ses classes de comportement, ses contraintes et ses différentes topologies. Nous nous appuyant ensuite sur quelques applications des RCSF pour exposer et montrer l'importance de cette technologie dans plusieurs domaines.

1.2 Histoire des RCSF

Dans les années 1990, une idée qui paraissait plutôt un rêve pour cette époque est apparue : celle d'imaginer un système nerveux central pour la Terre, capable de surveiller en temps réel les événements, ayant comme principaux bénéfices de pouvoir empêcher les accidents et d'économiser l'énergie. C'est ce que le professeur Kristofer PISTER de l'Université de Californie à Berkeley a appelé la poussière intelligente (Smart Dust) (Pister, 1999). La poussière intelligente a mis le temps mais a finalement arrivée. Aujourd'hui le réseau de capteurs sans fil est devenu une des merveilles technologiques qui a montré son impact sur notre vie quotidienne. Les derniers progrès en terme de miniaturisation, ainsi que le remplacement du câblage classique par des technologies de communication radio, ont généré de nouvelles catégories d'applications qui visent de nombreux domaines : l'aéronautique, l'automobile, le médical, l'environnement, etc. De plus, les progrès des communications sans fil permettent aujourd'hui de répondre des exigences peu envisageables auparavant.

Génération	Période	Taille	Poids	Batterie
1^{er}	Les années 80 et 90	Grande boite à chaussures	Kilogrammes	Grosse
$2^{ème}$	$2000 - 2003$	Boite de cartes	Grammes	AA
$3^{ème}$	2010	Particule de poussière	Négligeable	Solaire

Tableau 1.1 – Histoire des réseaux de capteurs

Les auteurs dans (Chong et Kumar, 2003a) ont parlé de trois générations de nœuds de capteurs (voir tableau 1.1). Grâce au développement remarquable des capteurs, les réseaux de capteurs seront aussi communs dans nos vies quotidiennes que les ordinateurs et l'internet.

1.3 Définition des RCSF

Un réseau de capteurs sans fil (en anglais : Wireless Sensor Network (WSN)) est un ensemble de dispositifs très petits, nommés nœuds capteurs variants de quelques dizaines d'éléments à plusieurs milliers. Dans ces réseaux chaque nœud est capable de récolter et de transmettre des données environnementales (température, lumière, pression,etc) d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène surveillé. Les données captées sont acheminées à un nœud considéré comme un point de collecte, appelé nœud puits (ou Sink). Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite. Ainsi, l'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits.

1.4 Architecture d'un RCSF

Un réseau de capteurs est constitué essentiellement de plusieurs nœuds capteurs, un nœud Sink et un centre de traitement des données :

Nœuds capteurs :

Se sont des capteurs, leur type, leur architecture et leur disposition géographique dépendent de l'exigence de l'application en question. Leur énergie est souvent limitée puisqu'ils sont alimentés par des piles. Dans tous les domaines, le rôle d'un réseau de capteurs est quasiment toujours le même comme le montre la Figure 1.1. Le principe est que les nœuds doivent surveiller un environnement en récupérant des données grâce à leurs capteurs, puis ils envoient les informations à une station de base (puits ou sink en anglais).

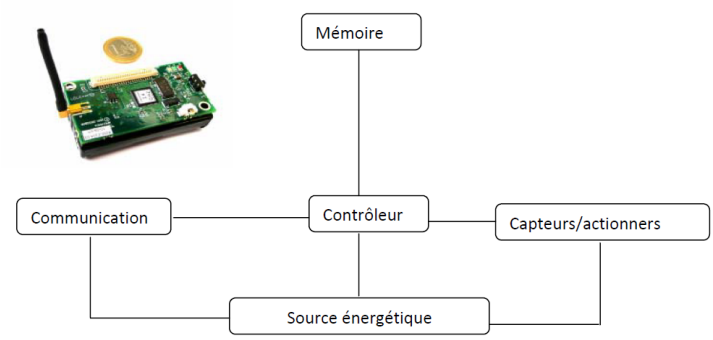


Figure 1.1 – Architecture d'un nœud capteur.

L'architecture de base d'un nœud capteur est illustrée en figure 1.1.

— *Unité de capture* : Pratiquement, chaque unité de capture est caractérisée par trois rayons : un rayon de communication qui lui permet d'échanger des messages et des données avec les autres nœuds capteurs, un rayon d'interférence permettant de détecter les collisions et un rayon plus large qui constitue le rayon de sensation. Ce dernier est la portée assurant la détection des évènements.

— *Contrôleur* : collecte les données produites par les capteurs associés à un nœud et celles envoyées par les nœuds capteurs voisins. Il assure également le traitement des données et la prise de décision concernant leur envoi (quand est ce qu'il faudra envoyer et quand ?).

— *Mémoire* : les valeurs lues ainsi que les paquets reçus sont stockés dans la RAM tandis que l'application du réseau de capteurs réside en ROM. Plus précisément, une EEPROM est requise afin de servir pour la sauvegarde des données en cas où la RAM n'est pas suffisante.

— *Communication* : la plupart des réseaux de capteurs s'appuient sur les communications radio fréquence (RF) puisqu'elles offrent un débit élevé et la distance de communication tolérée est bonne. A cela s'ajoute le fait que les erreurs sont acceptables et qu'aucune ligne de visée n'est requise. Pour cela, l'architecture d'un capteur intègre un émetteur/récepteur dont le rôle est de convertir les trames de bits émises par le microcontrôleur en une onde radio.

— *Alimentation* : la batterie constitue la source d'énergie la plus utilisée mais il existe

des capteurs capables de récupérer l'énergie à partir de l'environnement externe. Signalons que les batteries ont des caractéristiques matérielles (capacité, taux de décharge,...) qu'il faudra prendre en considération.

Station de base (Sink) :

C'est un nœud particulier du réseau. Il est chargé de la collecte des données issues des différents nœuds du réseau. Il doit être toujours actif puisque l'arrivée des informations est aléatoire. C'est pourquoi son énergie doit être illimitée. Dans un réseau de capteur sans fil plus ou moins large et à charge un peu élevée, on peut trouver deux Sink ou plus pour alléger la charge.

Centre de traitement des données :

C'est le centre vers lequel les données collectées par le Sink sont envoyées. Ce centre a le rôle de regrouper les données issues des nœuds et les traiter de façon à en extraire l'information utile et exploitable. Le centre de traitement peut être éloigné du Sink, par conséquent les données doivent être transférées à travers un autre réseau, c'est pourquoi on introduit une passerelle entre le Sink et le réseau de transfert pour adapter le type de données au type du canal (comme c'est illustré dans la figure 1.2).

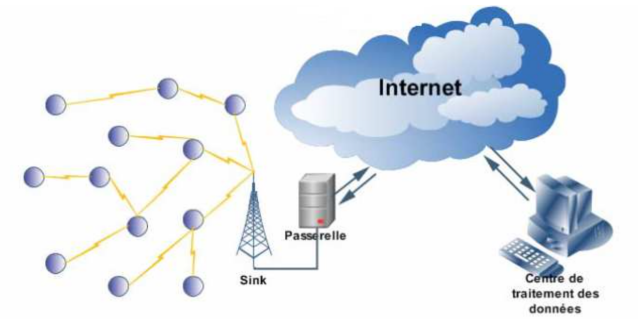


Figure 1.2 – Architecture générale d'un RCSF.

La figure 1.2 présente une architecture générale d'un RCSF où les nœuds capteurs transmettent leurs données vers un nœud Sink. Le centre de traitement des données étant éloigné, les données sont acheminées via Internet, et une passerelle est utilisée pour adapter le type des données au canal.

1.5 Contraintes d'un RCSF

La conception d'un RCSF est influencée par plusieurs paramètres qui peuvent être utilisées pour comparer entre deux RCSF. Parmi ces paramètres, nous citons :

La tolérance aux pannes :

Les nœuds capteurs sont vulnérables et souvent déployés dans un environnement dangereux. Les nœuds peuvent échouer en raison de problèmes matériels ou des dommages physiques ou par épuisement de leur énergie. Dans les RCSF, les défaillances de nœuds sont beaucoup plus élevées que dans les réseaux filaires ou sans fil basés sur les infrastructures. Les protocoles déployés dans un réseau de capteurs doivent être capables de détecter ces défaillances dès que possible et être suffisamment robustes pour gérer un nombre relativement élevé d'échecs tout en conservant la fonctionnalité globale du réseau. Ceci est particulièrement pertinent pour la conception de protocole de routage, qui doit veiller à ce que les chemins alternatifs soient disponibles pour le réacheminement des paquets. Les différents environnements de déploiement posent différentes exigences de tolérance aux pannes (Han *et al.*, 2010).

Le facteur d'échelle :

Le nombre de nœuds capteurs déployés peut atteindre des centaines, des milliers, voire même pour certaines applications, des millions. Les protocoles et algorithmes devront pouvoir fonctionner correctement dans tous les cas de figure. Classiquement, la densité des nœuds est de l'ordre de 300 nœuds pour $25m^2$ pour la surveillance de machines et de 10 nœuds par région pour la surveillance de véhicule. Une densité de 20 nœuds/m^2 est très courante. Dans une maison, on estime à 2 douzaines, le nombre d'appareils qui comporteront des capteurs. Pour la domotique est l'ensemble des techniques de l'électronique, de physique du bâtiment, d'automatisme, de l'informatique et des télécommunications utilisées dans les bâtiments et permettant de centraliser le contrôle des différents applicatifs de la maison., la densité devrait être comprise entre 20 et 100 nœuds par région. Les densités devraient être encore plus importantes si l'on insère des capteurs dans les lunettes, les vêtements, les chaussures, les montres, les bijoux ou encore le corps humain (Nekoogar *et al.*, 2004).

Les coûts de production :

Le coût de production d'un seul micro-capteur est très important pour l'évaluation du coût global du réseau, si ce dernier est supérieur à celui nécessaire pour le déploiement des capteurs classiques, l'utilisation de cette nouvelle technologie ne serait pas financièrement justifié. Par conséquent, réduire le coût de production jusqu'à moins de 1$ par nœud est un objectif important pour la faisabilité de la solution des réseaux de capteurs sans fil. La technologie Bluetooth a pu offrir un système radio connu d'être le moins chère du marché pour un coût de 10$, ce qui est dix fois plus chère que le coût désiré pour un nœud capteur. Ceci, sachant qu'un nœud contient d'autres systèmes que celui de transmission radio tel que les unités de captage et de traitement de données. De plus, le nœud peut être équipé d'éléments additionnels tel qu'un system de localisation GPS, ou un système de rechargement d'énergie. Dès lors, la minimisation du coût de production du nœud capteurs constitue un grand défis mené par les chercheurs, vu les fonctionnalités que doivent comporter ces nœuds et l'objectif désiré pour un coût inférieur à un dollar.

Les contraintes matérielles :

Un nœud capteur contient quatre composants de base : l'unité de captage, l'unité de traitement, l'unité de transmission, et l'unité de contrôle d'énergie. Il peut contenir également, suivant son domaine d'application, des éléments supplémentaires tels qu'un système de localisation, ou bien un système générateurs d'énergie. L'unité de captage englobe généralement deux sous-unités, le capteur lui-même en plus d'un convertisseur analogique-numérique qui transforme les signaux analogiques produits par les capteurs, et basés sur le phénomène observé en signal digitale, ce dernier est transmit par la suite à l'unité de traitement. L'unité de traitement, généralement associée à une petite unité de stockage, exécute les procédures permettant au nœud de collaborer avec les autres nœuds du réseau pour donner, en fin, le résultat de la tâche assignée au réseau. La connexion du nœud au réseau est gérée par l'unité de transmission. L'unité de contrôle d'énergie, cependant, constitue l'un des systèmes les plus importants dans un nœud capteur, celle-ci peut être représenté par un système de rechargement d'énergie tel que les cellules solaires.

Comme nous l'avons invoqué précédemment, un nœud capteur peut contenir d'autres unités dépendantes de l'application du réseau. en effet, la plupart des opération de captage et des algorithmes de routage dans les réseaux de capteurs sans fil requièrent la connaissance de la localisation des nœuds avec une grande précision, car ces nœuds sont déployés d'une manière aléatoire et fonctionne d'une façon autonome, ceci rend l'intégration d'une unité, consacré au système de localisation, très commune dans un nœud capteur. D'où, il est souvent supposé que ces nœuds possèdent un système de localisation GPS. le système GPS, et aider les autres nœuds à trouver leurs positions d'une manière terrestre. La conception des nœuds capteurs peut aller jusqu'à prévoir un système de mobilisation du capteur pour le déplacer en cas de nécessité.

Toutes ces unités peuvent exiger leur intégration dans un boîtier de taille minimale inférieure à un centimètre cube, et avec un poids très léger qui permet aux nœuds de rester suspendu dans l'air, si l'application l'exige. A part la taille, il existe d'autres contraintes exigeantes pour la construction des nœuds capteurs. Les autres contraintes sont la consommation d'énergie qui doit être moindre pour que le réseau survive le plus longtemps possible, l'adaptation aux différents environnements (fortes chaleurs, eau,..), l'autonomie et la résistance vu qu'il est souvent déployé dans des environnements hostiles.

La topologie du réseau :

La forte probabilité de panne d'un nœud capteur ou la possibilité de rajouter des nœuds font que la topologie du réseau doit pouvoir changer dynamiquement. La topologie d'un RCSF doit s'adapter à toutes les situations, à savoir les pannes, la mobilité et le rajout des nœuds. Il faut donc gérer avec précision la maintenance de cette topologie. On distingue trois phases : le déploiement, le post-déploiement des capteurs (les capteurs peuvent bouger, ne plus fonctionner,...) et le redéploiement des nœuds capteurs ajoutés.

— **Phase de pré-déploiement et de déploiement :** Les nœuds capteurs peuvent être éparpillés sur le champ de captage en masse ou placés d'une manière individuelle et ceci par le biais de plusieurs moyen tel que : les jeter d'un avion, les placer nœud par nœud d'une façon manuelle ou en utilisant des robots.
Le nombre important de nœud utilisés dans un réseau de capteurs empêche leur déploiement suivant un plan soigneusement établi, cependant un schéma général pour le déploiement initial doit être conçu pour permettre de réduire les coûts d'ins-

tallation, d'augmenter l'arrangement des nœuds et de faciliter l'auto-organisation des nœuds et leur tolérance aux pannes

— **Phase de post-déploiement :** Après la phase de déploiement, la topologie du réseau peut subir des changement dues aux : changement de position des nœuds, accessibilité (à cause du brouillage ou des obstacles en mouvements), épuisement d'énergie ou malfonctionnnement des nœuds. En effet, Bien que les nœuds d'un réseau de capteurs peuvent être déployés d'une manière statique, la panne matérielle constitue un évènement très commun à cause de l'épuisement d'énergie ou la destruction. Il est possible également d'avoir un réseau de capteur avec des nœuds mobiles qui ont une mobilité très élevée. Par conséquent, la topologie du réseau de capteur est exposée fréquemment aux changements après la phase de déploiement.
— **Phase de redéploiement des nouveaux nœuds :** Des nœuds capteurs additionnels peuvent être installés pour remplacer ceux qui sont en panne ou bien pour répondre aux besoins des tâches assignées au réseau. Cette addition entraîne la réorganisation du réseau et le changement de sa topologie.

Support de transmission

Dans un réseau de capteurs, les nœuds sont liés via un moyen de communication sans fil, et ceci, en utilisant un support optique, ou des fréquences radio. Cependant, il faut s'assurer de la disponibilité du moyen de transmission choisi dans l'environnement de captage, afin de permettre au réseau d'accomplir la totalité de ses tâches.

Pour les lien de communication via les fréquences radio, les bandes ISM (Industrial scientific Medical bands) peuvent être utilisées, ces bandes de fréquence sont employées pour assurer des communications libres de charge dans le domaine industriel, scientifique ou médical, et ceci dans la plupart des pays du monde.

Pour les réseaux de capteurs, les unités de transmission intégrées au niveau des nœuds doivent être de petite taille et à faible consommation d'énergie. En effet, suivant les contraintes matérielles associés aux nœuds, ainsi que le compromis existant entre l'efficacité des antennes et la consommation d'énergie, limite le choix de la bande de fréquence utilisée sur les bandes à hautes fréquences.

L'avantage principal des bandes ISM est qu'elles sont libres de toute licence d'utilisation, elles présentent un choix immense d'allocation de fréquence et sont disponibles dans la plupart des pays du monde. Ces bandes ne sont décrites par aucun standard, mais elles offrent plus de liberté pour l'implémentation des protocoles de communication spécifiques aux réseaux de capteurs. Toutefois, cette implémentation reste toujours limitée par d'autres contraintes telles que la consommation d'énergie minimale et les interférences nuisibles avec les autres applications utilisant les mêmes bandes de fréquence.

La majorité des prototypes des nœuds capteurs construits utilisent les radiofréquences comme moyen de communication. Les ondes infrarouges représentent un autre support possible pour la communication inter-nœuds dans un réseau de capteurs. Ce type de communication est également libre de toute charge ou licence, il est robuste contre les interférence avec les appareils électriques, et les unités de transmission correspondantes sont moins chères sur le marché, et plus faciles à construire. Ceci peut expliquer l'existence des ports de communication infrarouge dans la plupart des ordinateurs, téléphones portables et PDAs. L'inconvénient majeur pour ce type de communication est qu'il exige la dispo-

nibilité permanente d'une ligne de vue entre l'émetteur et le récepteur, cette contrainte rend l'utilisation de ce support dans les réseaux de capteur un choix réticent.

Les contraintes liés aux domaines d'applications spécifiques pour les réseaux de capteurs rendent le choix du support de communication une étape critique pour la conception de ces réseaux. Par exemple, les applications liées au domaine maritime peuvent favoriser l'utilisation d'un support de transmission aqueux tel que les radiations à longueur d'onde élevée pouvant pénétrer la surface d'eau. Tandis que les applications pour les terrains hostiles, tel que les champs de batailles peuvent confronter un taux élevé d'erreurs et plus de brouillage et d'interférence avec l'environnement capté, pour cela, le choix du support de transmission doit, dans ce cas, prévoir des schémas de modulation robustes pour prendre en charge des canaux de communication ayant des caractéristiques étroitement différents.

L'environnement de déploiement :

Les nœuds peuvent être déployés tout près de l'objet à surveiller ou en son sein. L'environnement de déploiement peut être à l'intérieur d'une grosse machine, au fond d'un océan, dans un lieu contaminé biologiquement ou chimiquement, dans un champ de bataille, dans une maison ou un immeuble, sur un animal, sur un véhicule, etc. Ces situations très variées engendrent des contraintes très fortes de l'environnement sur les nœuds capteurs.

L'environnement détermine en partie le comportement du réseau de capteurs sans fil. En effet, le trafic de données est principalement généré par l'environnement. En particulier, dans le cas de la détection d'événements, le nombre d'alertes transmises à la station de base dépend du nombre d'événements dans l'environnement. Par ailleurs, un réseau de capteurs sans fil est un système dit réactif qui réagit en permanence à son environnement.

La consommation d'énergie :

Comme les nœuds capteurs sont des composant micro-électroniques, ils ne peuvent être équipés que par des sources limitées d'énergie (<0.5 Ampère-heure, 1.2 V). De plus, dans certaines applications, ces nœuds ne peuvent pas être dotés de mécanismes de rechargement d'énergie, par conséquent, la durée de vie d'un nœud capteur dépend fortement de la durée de vie de la batterie associée. Sachant que les réseaux de capteurs sont basés sur la communication multi-sauts, chaque nœud joue à la fois un rôle d'initiateur de données et de routeur également, le malfonctionnement d'un certain nombre de nœud entraîne un changement significatif sur la topologie globale du réseau, et peut nécessiter un routage de paquets différent et une réorganisation totale du réseau. C'est pour cela que le facteur de consommation d'énergie est d'une importance primordiale dans les réseaux de capteurs. La majorité des travaux de recherche menés actuellement se concentrent sur ce problème afin de concevoir des algorithmes et protocoles spécifiques à ce genre de réseau qui consomment le minimum d'énergie. En effet, dans les réseaux de capteurs, l'efficacité en consommation d'énergie représente une métrique de performance significative, qui influence directement sur la durée de vie du réseau en entier. Pour cela, les concepteurs peuvent au moment du développement de protocoles négliger les autres métriques de performance telle que la durée de transmission et le débit, au détriment du facteur de consommation d'énergie.

Détecter les évènements dans l'environnement capté, élaborer un traitement de don-

nées local et rapide, et transmettre les résultats à l'utilisateur sont les principales tâches d'un nœud dans un réseau de capteurs. Les étapes de consommation d'énergie par ce nœud peuvent être, dès lors, divisées en trois phases : le captage, la communication et le traitement de donnée.

— **Phase de captage :** L'énergie consommée au moment du captage varie suivant la nature de l'application. Un captage sporadique consomme moins d'énergie qu'un contrôle d'événement constant. La complexité de l'événement à détecter joue également un rôle crucial pour déterminer la quantité d'énergie consommée. Les environnements contenant un niveau de bruit élevé entraîne l'augmentation de l'énergie nécessaire pour cette phase.

— **Phase de communication :** Parmi les trois phases citées auparavant, la phase de communication de donnée est celle qui consomme la plus grande quantité d'énergie, ceci, à cause de la multitude de composants électroniques intégrés au circuit responsable de cette opération. Cette phase implique les deux étapes d'émission et de réception de données. Il est démontré que pour les communications à courte portée, avec une faible puissance de radiation (0 dbm), les coûts énergétiques pour l'émission et la réception de données sont pratiquement égaux. Durant cette phase, il est important de considérer l'énergie nécessaire pour la mise en marche du circuit de communication, le temps de démarrage étant égal à plusieurs centaines de micro-secondes rend l'énergie consommée durant cette période non négligeable. L'influence de cette étape sur la quantité globale d'énergie consommée par la communication augmente quand la taille des paquets transmis diminue. Par conséquent, une bonne politique de consommation d'énergie passe obligatoirement par éviter au maximum le recourt à la mise en marche et l'arrêt fréquents des circuits de communication.

— **Phase de traitement de données :** Comparé à la phase de communication, l'étape de traitement local des données consomme beaucoup moins d'énergie. En effet, le coût énergétique nécessaire pour transmettre 1 KB sur une portée de 100 m est approximativement égal à celui nécessaire pour exécuter 3 millions d'instructions à une vitesse de 100 millions instructions par seconde, ce fait, favorise largement le traitement local des données pour l'amélioration de la consommation d'énergie dans les réseaux de capteurs. Les nœuds capteurs doivent donc posséder des moyens de traitement local de données, tout en restant capable d'interagir avec les nœuds avoisinants. Enfin il est à noter qu'un nœud peut contenir des circuits additionnel pour le codage/décodage des données, en plus de certain circuits spécifiques aux applications du réseau, dans tous ces cas, la conception des algorithmes et protocoles du réseau est influencé largement par l'énergie consommée par ces circuits en plus de ceux invoqués précédemment.

La qualité de service (QoS) :

Différentes communautés peuvent interpréter la qualité de service Quality of Service (QoS) des RCSF de différentes manières (Chen et Varshney, 2004). Par exemple, dans les applications impliquant la détection et la poursuite de cibles, beaucoup de facteurs peuvent contribuer à l'échec dans la détection ou l'erreur dans la poursuite. Ceci peut être dû au déploiement et à la gestion du réseau, c'est-à-dire l'endroit où l'évènement se produit ne peut être couvert par aucun nœud capteur actif. Intuitivement, nous pouvons définir la couverture ou le nombre de capteurs actifs comme paramètres de mesure de la QoS. En outre, l'échec peut être provoqué par la fonctionnalité limitée des nœuds capteurs,

par exemple, l'exactitude d'observation inadéquate ou un taux de reportage faible des nœuds. De ce fait, on peut définir les erreurs d'exactitude ou de mesure d'observation comme paramètres de mesure de la QoS. De plus, cette dernière peut être induite par la perte de l'information lors de son routage. Il est aussi possible de définir quelques paramètres de circulation de l'information pour mesurer la QoS. Cependant, la séparation dans la définition des paramètres de mesure de la QoS n'est pas absolue puisqu'une seule application, telle que la mesure de performance liée à la détection d'évènements, peut impliquer tous ces paramètres.

La mobilité :

La vertu principale de la communication sans fil est sa capacité de soutenir les participants mobiles. Dans les réseaux de capteurs sans fil, la mobilité peut apparaître sous trois formes principales :

— **Mobilité de nœud** : La signification d'une telle mobilité est fortement liée à l'application en question. Par exemple, dans le contrôle de l'environnement, la mobilité du nœud n'existe pas, par contre on l'observe dans la surveillance des animaux (nœud attaché à un animal). Face à la mobilité du nœud, le réseau doit se réorganiser assez fréquemment pour pouvoir fonctionner correctement (Karl et Willig, 2007).

— **Mobilité du Sink** :C'est un cas spécial de mobilité de nœud. L'aspect important c'est la mobilité d'un récepteur d'informations qui ne fait pas partie du réseau de capteurs, par exemple, l'information peut être demandée par un utilisateur via un PDA tout en se déplaçant dans le réseau (Karl et Willig, 2007).

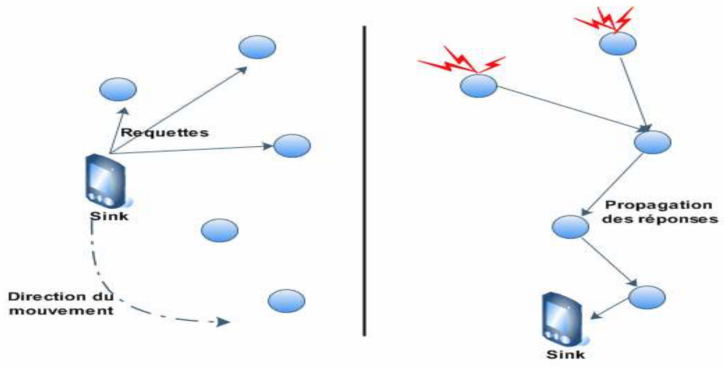


Figure 1.3 – Notion de la mobilité de Sink.

— **Mobilité de l'évènement** : Ce type de mobilité existe essentiellement dans les applications de détection des évènements et de suivi des cibles. Dans un tel type d'application, il est (habituellement) important que l'évènement observé soit couvert par un nombre suffisant de nœuds. Par conséquent, les nœuds vont se réveiller autour de l'objet, pour le surveiller avec un taux d'activité élevée, et puis entrent en sommeil (mode Sleep). Pendant que la source d'évènement se déplace à travers le réseau, elle est accompagnée d'un secteur d'activité dans le réseau qui le suit (Karl et Willig, 2007).

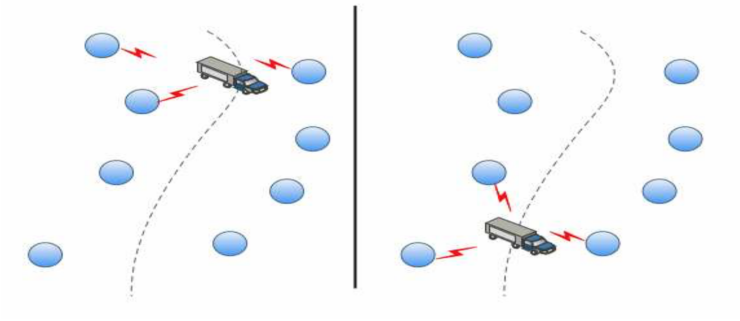


Figure 1.4 – Notion de la mobilité de l'évènement.

La figure 1.4 décrit la notion de la mobilité de l'évènement, ceci consiste à détecter un véhicule et de l'observer pendant son déplacement. Les nœuds qui ne détectent rien entrent en sommeil à moins qu'ils soient invités à transmettre de l'information de la zone d'activité à un certain Sink distant.

1.6 Topologie des RCSF

Un réseau de capteurs sans fil est composé d'un ensemble de nœuds capteurs et des Gateway qui s'occupent de collecter les données des capteurs et de les transmettre à l'utilisateur via l'internet ou le satellite, il existe plusieurs topologies pour les réseaux de capteurs :

1.6.1 Topologie physique

Topologie en étoile

La topologie en étoile est un système uni-saut. Tous les nœuds envoient et reçoivent seulement des données avec la station de base. Cette topologie est simple et elle demande une faible consommation d'énergie, mais la station de base est vulnérable et la distance entre les nœuds et la station est limitée.

Topologie en maille ou grille (Mesh Network)

Dans ce cas (dit « communication multi-sauts »), tout nœud peut échanger avec n'importe quel autre nœud du réseau (s'il est à portée de transmission). Un nœud voulant transmettre un message à un autre nœud hors de sa portée de transmission, peut utiliser un nœud intermédiaire pour envoyer son message au nœud destinataire. Cette topologie a plus de possibilités de passer à l'échelle du réseau, avec redondance et tolérance aux fautes, mais elle demande une consommation d'énergie plus importante. Une latence est créée par le passage des messages des nœuds par plusieurs autres avant d'arriver à la station de base.

Topologie hybride

La topologie hybride est un mélange des deux topologies ci-dessus. Les stations de base forment une topologie en toile et les nœuds autour d'elles sont en topologie étoile. Elle assure la minimisation d'énergie dans les réseaux de capteurs.

1.6.2 Topologie logique

La topologie détermine l'organisation logique des capteurs dans le réseau. Il existe deux principales topologies dans les protocoles de routage pour les WSNs.

Topologie plate

Un réseau de capteurs sans fil plat est un réseau homogène, où tous les nœuds sont identiques en termes de batterie et de complexité du matériel et ont le même rôle, excepté le nœud puits qui joue le rôle d'une passerelle et qui est responsable de la transmission de l'information collectée à l'utilisateur final. Selon le service et le type de capteurs, une densité de capteurs élevée (plusieurs nœuds capteurs/m^2) ainsi qu'une communication multi-sauts peut être nécessaire pour l'architecture plate.

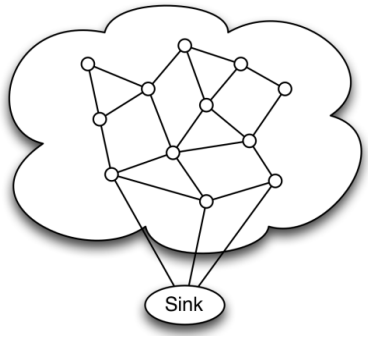


Figure 1.5 – Topologie plate.

Ce type de solution permet une grande tolérance aux pannes, cependant, elle souffre d'une faible scalabilité. En effet, si tous les nœuds opèrent de la même façon et d'une manière distribuée, on aura un grand nombre de messages de contrôle nécessaires pour le bon fonctionnement du réseau.

Topologie hiérarchique

Afin d'augmenter la scalabilité du système, les topologies hiérarchiques ont été introduites en divisant les nœuds en plusieurs niveaux de responsabilité. L'une des méthodes la plus employée est le Clustering, avec laquelle le réseau est partitionné en groupes appelés clusters. Un cluster est constitué d'un chef (ClusterHead) et de ses membres. Suivant l'application, les membres peuvent être des voisins directs ou indirects du ClusterHead.

Avec une approche hiérarchique, il est plus facile d'intégrer un mécanisme d'agrégation au système : les nœuds membres transmettent leurs données vers le ClusterHead, qui va par la suite agréger ces lectures afin de transmettre le résumé à la station de base. Pour augmenter la tolérance aux pannes, la sélection des ClusterHead doit être dynamique afin d'éviter la présence des membres isolés.

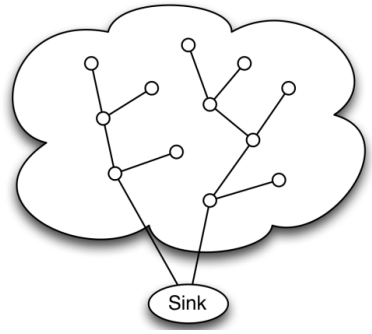


Figure 1.6 – Topologie hiérarchique.

L'inconvénient de la hiérarchisation est la surcharge des ClusterHeads, induisant à un déséquilibre de la consommation d'énergie sur le réseau.

1.7 Types des réseaux de capteurs sans fil

Les réseaux de capteurs actuels sont déployés sur terre, sous terre et sous l'eau. Selon l'environnement, un réseau de capteurs est confronté à différents défis et contraintes. Il existe cinq types de réseaux de capteurs : terrestre, souterrain, sous-marin, multimédia et mobile. Les réseaux de capteurs terrestres (Akyildiz *et al.*, 2002a) sont généralement constitués de centaines voire de milliers de nœuds capteurs sans fil peu coûteux déployés dans une zone donnée, que ce soit d'une manière ad hoc ou d'une manière pré-planifiée. Dans un déploiement ad hoc, les nœuds capteurs peuvent être largués d'un avion et placés aléatoirement dans la zone cible. Dans un déploiement pré-planifié, les nœuds se répartissent en grille, en placement optimal (Toumpis et Tassiulas, 2006), ou en modèles de placement 2D et 3D (Pompili *et al.*, 2006).

1.7.1 Terrestres

Dans un réseau de capteurs terrestre, la communication fiable dans un environnement dense est très importante. Les nœuds capteurs terrestres doivent être capables de communiquer efficacement les données vers la station de base. Alors que l'énergie de la batterie est limitée et ne peut être rechargeable, les nœuds capteurs terrestres peuvent être cependant équipés d'une source d'alimentation secondaire telles que les cellules solaires. Dans tous les cas, il est important de conserver l'énergie pour les nœuds capteurs. Pour un réseau de capteurs terrestre, l'énergie peut être conservée avec le routage optimal multi-saut, gamme de transmission courte, agrégation des données du réseau, en éliminant la redondance des données, en réduisant les délais, et en utilisant les opérations à faible rapport cyclique.

1.7.2 Souterrains

Les réseaux de capteurs souterrains (Akyildiz et Stuntebeck, 2006; Li et Liu, 2007) se composent d'un nombre de nœuds capteurs enfouis sous terre ou dans une grotte ou une mine utilisés pour surveiller les conditions souterraines. Les nœuds sink additionnels sont situés au dessus du sol pour relayer l'information à partir des nœuds capteurs vers la station de base. Un réseau de capteurs souterrain est plus cher qu'un réseau de capteurs terrestre en termes d'équipement, de déploiement et de maintenance. Les nœuds capteurs souterrains sont chers parce que les pièces d'équipements appropriés doivent être choisies pour assurer une communication fiable à travers le sol, les roches, l'eau et autres contenus minéraux. Le milieu souterrain rend la communication sans fil un challenge en raison des pertes du signal et des niveaux élevés d'atténuation. Contrairement aux réseaux de capteurs terrestres, le déploiement d'un réseau de capteurs souterrain nécessite une planification minutieuse et une considération d'énergie et de coût. L'énergie est une préoccupation importante dans les réseaux de capteurs souterrains. Comme dans un réseau da capteurs terrestre, les nœuds capteurs souterrains sont équipés d'une alimentation limitée de la batterie qui est une fois déployée dans le sol, il est difficile de la recharger ou de la remplacer. Comme précédemment, l'objectif essentiel est de conserver l'énergie afin d'augmenter la durée de vie d'un réseau qui peut être atteint par l'implémentation d'un protocole de communication efficace.

1.7.3 Sous-marins

Les réseaux de capteurs sous-marins (Akyildiz *et al.*, 2004; Heidemann *et al.*, 2005) se composent d'un nombre de nœuds capteurs et des véhicules déployés sous l'eau. A la différence des réseaux de capteurs terrestres, les nœuds capteurs sous-marins sont plus chers et sont moins déployés. Les véhicules sous-marins autonomes sont utilisés pour l'exploration ou la collecte des données de nœuds capteurs. Comparé à un déploiement dense des nœuds capteurs dans un réseau de capteurs terrestre, un déploiement clairsemé de nœuds capteurs est placé sous l'eau. Les communications sans fil sous-marines sont établies par transmission d'ondes acoustiques. Un défi en communication acoustique sous-marine est la bande passante limitée, le long temps de propagation, et le fading du signal issu. Un autre défi est l'échec de nœud capteur en raison des conditions environnementales. Les nœuds capteurs sous-marins doivent être capables de s'auto-configurer et de s'adapter à l'environnement dur de l'océan. Les nœuds capteurs sous-marins sont équipés d'une batterie limitée qui ne peut être remplacée ou rechargée. La question de la conservation de l'énergie pour les réseaux de capteurs sous-marins consiste à développer des techniques de routage et de communication sous-marine efficaces.

1.7.4 Multimédias

Les réseaux de capteurs multimédias (Akyildiz *et al.*, 2007) ont été proposés pour permettre la surveillance et le suivi des événements dans la forme de multimédia comme la vidéo, l'audio et l'image. Les réseaux de capteurs multimédia se composent d'un certain nombre de nœuds capteurs à faible coût équipés de caméras et des microphones. Ces nœuds capteurs interconnectent les uns avec les autres via une connexion sans fil pour la restitution, le traitement, la corrélation et la compression de données. Les nœuds capteurs multimédias sont déployés de manière pré-planifiée dans l'environnement pour garantir une

couverture. Les défis dans les réseaux de capteurs multi-médias comprennent une forte demande de bande passante, une forte consommation d'énergie, une qualité de service (QoS), des techniques de traitement et de compression de données et une conception inter-couche (cross-layer design). Le contenu multimédia comme un flux vidéo nécessite une bande passante élevée afin que le contenu puisse être délivré. En conséquence, un débit élevé de données entraîne une consommation d'énergie élevée. Les techniques de transmission qui prennent en charge une bande passante élevée et une faible consommation d'énergie doivent être développées. L'approvisionnement de la qualité de service est une tâche difficile dans un réseau de capteurs multimédias en raison de la variable retard et la variable capacité du canal. Il est important qu'un certain niveau de qualité de service doive être atteint pour une livraison fiable de contenu. En réseau, le traitement, le filtrage et la compression peuvent. améliorer considérablement les performances du réseau en termes de filtrage et d'extraction d'informations redondantes et les contenus qui fusionnent. De même, l'interaction cross-layer entre les couches peut améliorer le processus de traitement et de livraison.

1.7.5 Mobiles

Les réseaux de capteurs mobiles sont constitués d'un ensemble de nœuds capteurs qui peuvent se déplacer par leurs propres moyens et d'interagir avec l'environnement physique. Comme les nœuds statiques, les nœuds mobiles ont la capacité de collecter, calculer et communiquer. Une différence clé est que les nœuds mobiles ont la possibilité de repositionner et de s'organiser en réseau. Un réseau de capteurs mobiles peut commencer avec un déploiement initial et les nœuds peuvent alors s'étaler pour recueillir des informations. L'information recueillie par un nœud mobile peut être communiquée à un autre nœud mobile quand l'un se trouve à la même portée que l'autre. Une autre différence essentielle est la distribution des données. Dans un réseau de capteurs statique, les données peuvent être distribuées à l'aide de routage ou par inondation alors que le routage dynamique est utilisé dans un réseau de capteurs mobiles. Les challenges dans un réseau de capteurs mobiles comprennent le déploiement, la localisation, l'auto-organisation, la navigation et le contrôle, la couverture, l'énergie, l'entretien et le traitement des données. Les applications des réseaux de capteurs mobiles incluent la surveillance de l'environnement, le suivi de cible, la recherche, le sauvetage et la surveillance en temps réel des matières dangereuses. Pour la surveillance de l'environnement dans les zones sinistrées, le déploiement manuel pourrait ne pas être possible. Avec les nœuds capteurs mobiles, ils peuvent se déplacer dans les zones d'événements après le déploiement pour fournir la couverture nécessaire. Dans la surveillance et le suivi militaire, les nœuds capteurs mobiles peuvent collaborer et prendre des décisions fondées sur la cible. Les nœuds capteurs mobiles peuvent atteindre un degré de couverture et une connectivité plus élevés par rapport aux nœuds capteurs statiques. En présence d'obstacles sur le terrain, les nœuds capteurs mobiles peuvent être planifiés à l'avance et déplacés de manière appropriée aux régions obstruées pour augmenter l'exposition de la cible.

1.8 Applications des RCSF

La miniaturisation, l'adaptabilité, le faible cout et la communication sans fil permettent aux réseaux de capteurs de s'installer dans plusieurs domaines d'application et permettent aussi d'étendre le domaine des applications existantes. Parmi ces domaines ou

ces réseaux se révèlent très utiles et peuvent offrir de meilleures contributions, on peut noter les domaines militaires, de la santé, de l'environnement,...(Akyildiz *et al.*, 2002b)

Applications militaires :

Le faible cout, le déploiement rapide, l'auto-organisation et la tolérance aux pannes sont des caractéristiques qui ont rendu les réseaux de capteurs efficaces pour les applications militaires. Plusieurs projets ont été lancées pour aider les unités militaires dans un champ de bataille et protéger les villes contre des attaques, telles que les menaces terroristes. Le projet DSN (Distributed Sensor Network) (Chong et Kumar, 2003b) au DARPA (Defense Advanced Research Projects Agency) était l'un des premiers projets dans les années 80 ayant utilisée les réseaux de capteurs pour rassembler des données distribuées. Les chercheurs du laboratoire national Lawrence Livermore ont mis en place le réseau WATS (Wide Area Tracking System) (Gosnell *et al.*, 1997). Ce réseau est composé de détecteurs des rayons gamma et des neutrons pour détecter et dépister les dispositifs nucléaires. Il est capable d'effectuer la surveillance constante d'une zone d'intérêt. Il utilise des techniques d'agrégation de données pour les rapporter à un centre intelligent.

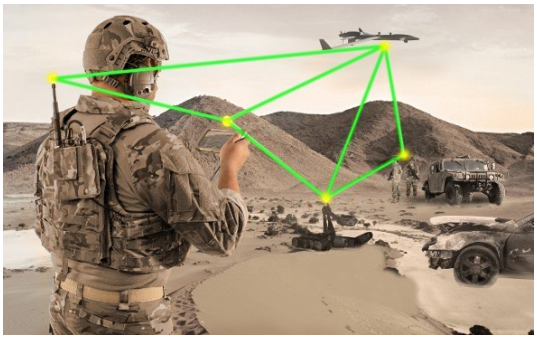


Figure 1.7 – Application militaire : surveillance des mouvements des forces ennemies.

Un réseau de capteurs peut être déployé dans un endroit stratégique ou hostile, afin de surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes (détection des armes chimiques, biologiques ou radiations). L'armée américaine a d'ailleurs réalisé des tests dans le désert de Californie.

Domotique :

Un autre type d'applications dans lequel les réseaux de capteurs émergent est la domotique (Campo *et al.*, 2012). Dans ces applications, le réseau de capteurs est déployé dans l'habitation (sur le plafond, dans le mur, ...) pour former un environnement pervasif. Son but est de fournir toutes les informations nécessaires aux applications de confort, de sécurité et de maintenance dans l'habitat. Les capteurs placés sont des détecteurs de présence, de son, et d'image ou vidéo (si les capteurs sont équipés de caméras). Un tel réseau déployé doit permettre de créer une maison intelligente capable de comprendre des situations suivant le comportement des occupants et d'en déduire des actions. Dans ce type d'applications, les réseaux sont très hétérogènes, des éléments d'électroménager

peuvent faire partie du réseau aussi bien que les ordinateurs personnels. Ils sont hautement configurables car d'une part la topologie du réseau peut changer d'un jour à l'autre avec l'aménagement, d'autre part on peut avoir besoin de changer le type d'applications pendant la vie du réseau.

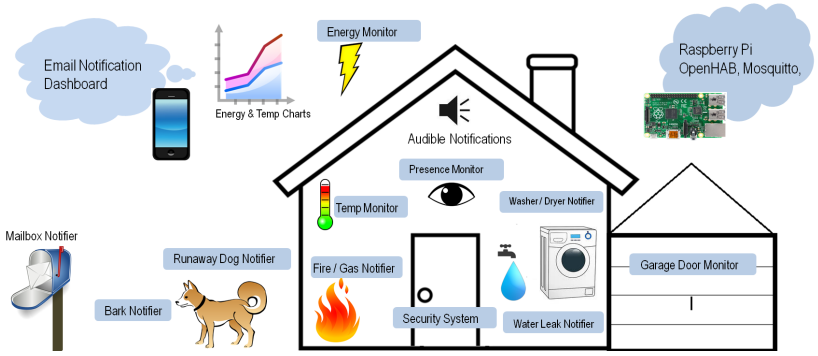


Figure 1.8 – Domotique : Maison Intelligente

Applications environnementales :

Les réseaux de capteurs sont beaucoup appliqués dans ce domaine pour remonter des alarmes ou surveiller des zone difficile d'accès. Les réseaux de capteurs sont notamment pressentis pour aider à la surveillance et à la prévention des feux de foret (Bouabdellah *et al.*, 2013). Dans ce cas, le déploiement d'un réseau de capteurs permet d'alerter les secours en cas d'incendie, et d'évaluer le risque de départ de feu grâce à des relevées périodiques (température, humidité ...).

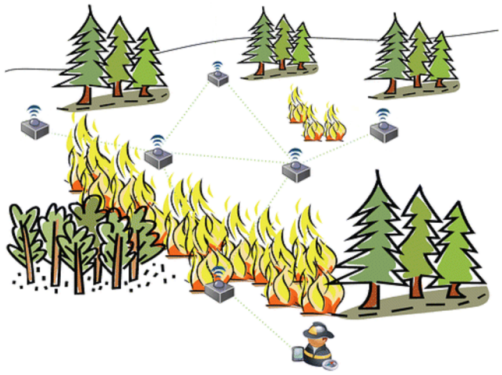


Figure 1.9 – Détection des feux de foret

Le déploiement des capteurs chimiques dans les milieux urbains peut aider à détecter la pollution et analyser la qualité de l'air. Dans ce cas, leur déploiement dans les sites industriels permettrait une détection des risques industriels plus rapidement et réduirait très considérablement la fuite de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole, etc). On retrouve aussi l'observation de la vie d'espèces rares, l'utilisation d'un

réseau de capteurs permet le comptage d'un espèce et en assurer la protection. Ce réseau peut aussi servir à l'étude de l'espèce précisément la relation entre plusieurs individus, la façon de fonctionner, etc.

Applications agricoles :

L'agriculture par l'exploitation des réseaux sans fil a pour objectif une gestion optimisée des apports aux cultures (semences, eau d'irrigation, fertilisants et produits phytosanitaires) dans le but d'améliorer le rendement et de réduire les effets néfastes sur l'environnement. Les caractéristiques des sols, la topographie, les attaques parasitaires, la présence de mauvaises herbes peuvent varier beaucoup sur un espace restreint. Les réseaux de capteurs sans fil constituent un outil de premier ordre pour une meilleure compréhension des processus culturaux en offrant des mesures en temps réel indispensables à une gestion optimale des parcelles agricoles (Arun et Sudha, 2012). Les RCSF permettent de télé-surveiller de vastes zones géographiques dépourvues d'infrastructure sans recourir à un câblage onéreux, et permettent de relever de nombreux paramètres localisés (température, humidité, détection du gel, pannes etc.) dans un laps de temps très court. Ces données sont acheminées à un centre de décision qui peut agir de manière automatisée en envoyant des commandes à distance.

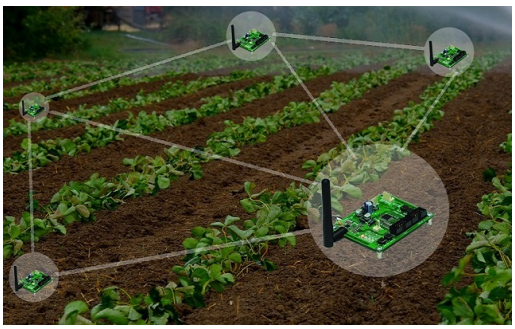


Figure 1.10 – Applications agricoles

Applications médicales :

Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sur le patient (surveillance de la glycémie, détection de cancers, ...) (Douglas *et al.*, 2012). Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des mesures physiologiques telles que : la tension artérielle, battements du cœur, température, ...à l'aide des capteurs ayant chacun une tache bien particulière. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri, ...) chez les personnes dépendantes (handicapées ou âgées).

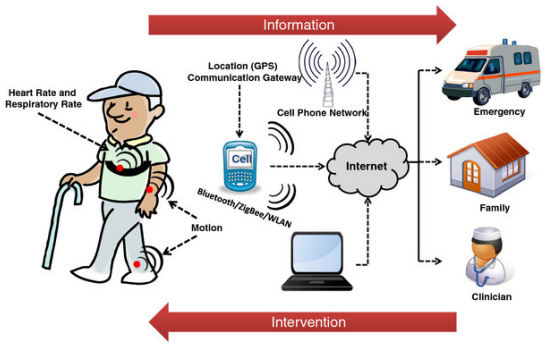


Figure 1.11 – Application médicale

le bâtiment :

lors de tremblements de terre, pour aider les secouristes à retrouver les victimes (capteurs emprisonnés dans le béton à la construction qui détectent le niveau de bruit) Quand nous sommes en face d'une tremblement de terre, un ouragan ou bien n'importe quel désastre, les réseaux sans fil peuvent s'avérer très utiles dans les opérations de la recherche et sauvetage. En général, les désastres laissent une grande population sans électricité et moyens de communication. Les RCSF peuvent être établis sans de telles infrastructures et peuvent fournir des transmissions entre les diverses équipes de recherche pour coordonner leurs opérations de sauvetage.

Applications commerciales :

Il est possible d'intégrer des nœuds capteurs au processus de stockage et de livraison. Le réseau pourra être utilisé pour connaître la position, l'état et la direction d'un paquet ou d'une cargaison. Il devient alors possible pour un client qui attend la réception d'un paquet, d'avoir un avis de livraison en temps réel et de connaître la position actuelle du paquet. Pour les entreprises manufacturières, les réseaux de capteurs permettront de suivre le procédé de production à partir des matières premières jusqu'au produit final livré. Grâce aux réseaux de capteurs, les entreprises pourraient offrir une meilleure qualité de service tout en réduisant leurs coûts.

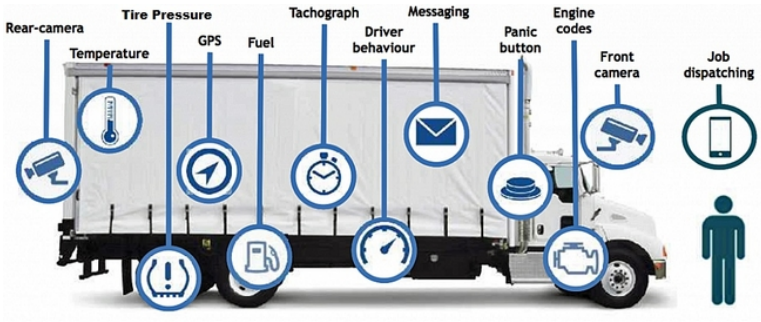


Figure 1.12 – Application commerciale

Applications dans le domaine sportif :

L'évolution des réseaux de capteurs est utilisée de plus en plus dans le domaine sportif, à savoir les systèmes de calcul de trajectoires (comme dans le tennis ou le golf), les systèmes de détection d'erreurs d'arbitrage (comme dans le football indiquent si la balle a franchi la ligne de but) et les systèmes de surveillance à distance des joueurs d'un équipe (par exemple un équipe de football) (Garcia *et al.*, 2011).

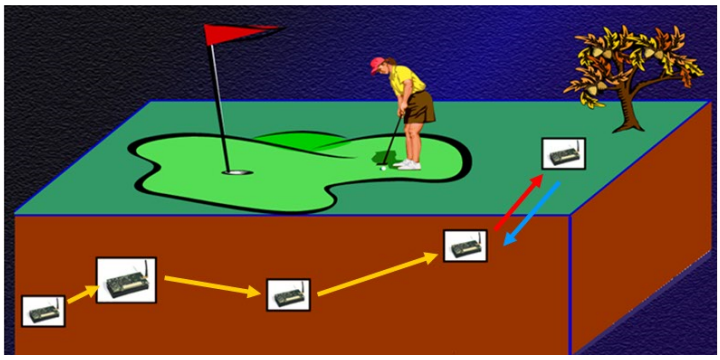


Figure 1.13 – Application dans le domaine sportif

1.9 Conclusion

Dans ce chapitre nous avons procédé à l'étude des réseaux de capteurs sans fil. Nous avons introduit quelques concepts nécessaires à la compréhension de nos problématiques dans la suite de ce mémoire. Nous avons décrit brièvement ce qu'est un réseau de capteurs, son architecture, ses principales contraintes de conception, puis nous avons mis l'accent sur ses principaux applications afin de montrer l'utilité des RCSF dans la vie quotidienne. D'après nos études nous avons remarqué que la minimisation de la consommation d'énergie d'un nœud-capteur est la plus grande préoccupation de tous les protocoles et méthodes proposés. En effet, lorsque ce n'est pas l'objectif principal, alors c'est sûrement un critère de performance capital. Dans le prochain chapitre nous conjuguerons cela avec la notion de la durée de vie du réseau et nous dresserons un panorama des problèmes d'optimisation fondamentaux proposés dans la littérature afin de trouver des solutions optimales pour prolonger la durée de vie des réseaux de capteurs.

Chapitre

2

L'OPTIMISATION DANS LES RCSF

Sommaire

2.1 Introduction

Les réseaux de capteurs sans fil (RCSF) représentent un domaine de recherche de plus en plus intéressant, grâce à leurs applications qui se multiplient d'un jour à l'autre et la capacité de représenter plusieurs systèmes de réseaux complexes. Cette complexité qui se figure dans des problèmes fondamentaux liés au routage, la couverture, le contrôle de la topologie, la mobilité et l'ordonnancement, nécessite des algorithmes qui se nomment robustes et performants. Ces dernières années, la majorité des chercheurs dans le domaine de l'optimisation ont convergé leur attention vers les métaheuristiques inspirées de la nature pour résoudre les problèmes d'optimisation NP-difficiles. Plusieurs problèmes dans les RCSF sont NP-difficiles, ce qui explique pourquoi les métaheuristiques sont les mieux adaptées pour les résoudre. Dans ce chapitre, nous allons présenté des problèmes d'optimisation dans les RCSF, ainsi que les méthodes de résolution proposées. Ensuite, nous allons mettre le point sur les métaheuristiques puisqu'ils sont les méthodes les plus adéquates pour ces problèmes. Finalement, nous allons présenté les travaux de recherche traitant le problème de routage dans les RCSF en utilisant les métaheuristiques inspirées de la nature.

2.2 Problèmes d'optimisation des RCSF

L'optimisation intervient dans de nombreuses problématiques des réseaux de capteurs sans fil : la gestion de la couverture de la zone à surveiller, problèmes de contrôle de topologie, Problèmes d'ordonnancement et problème de routage ...Nous commencerons

par présenter les différents modèles de couverture utilisés dans la littérature, puis nous aborderons successivement les différents axes énoncés ci-dessus.

2.2.1 Problèmes de couverture

La couverture d'un réseau de capteurs sans fil est la zone couverte par l'ensemble des capteurs déployés. Chaque capteur possède un rayon de communication (Rc) et un rayon de sensation (Rs). La Figure 2.1 montre les zones définies par ces deux rayons pour le capteur A. La zone de communication est la zone où le capteur A peut communiquer avec les autres capteurs (le capteur B dans la figure). D'autre part, la zone de sensation est la zone où le capteur A peut capter l'évènement. Ces zones Rc et Rs représentent la zone de couverture d'un capteur.

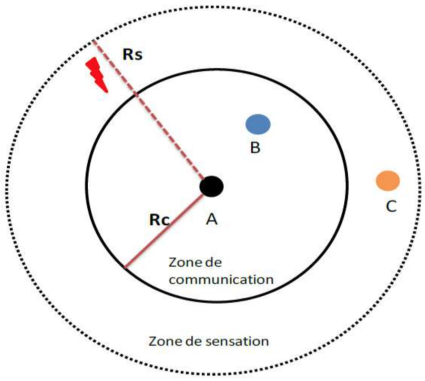


Figure 2.1 – Représentation de la zone de couverture du capteur A.

La couverture assurée par un réseau de capteurs peut être assimilée à la qualité de service : c'est la fonction et le but principal du réseau. Il existe plusieurs manières de définir cette couverture, en fonction de l'application attribuée au réseau de capteurs. En effet, on peut souhaiter une couverture globale d'une zone à observer ou encore n'exiger que l'observation d'un nombre défini de cibles. Dans tous les cas, la couverture est un objectif primordial dans les réseaux de capteurs et est souvent soit à maximiser, soit exprimée sous la forme de contraintes afin d'assurer une couverture totale.

Le problème de couverture consiste à minimiser le nombre de capteurs déployés tout en garantissant une couverture totale de la zone, plusieurs approches sont exploitées dans la littérature. La complexité du problème change selon la représentation de la zone. En effet, dans le cas d'une représentation par un ensemble fini de positions à couvrir, le problème a été démontré comme étant NP-complet (Ke *et al.*, 2011) (The critical-square-grid coverage problem). De nombreuses méthodes d'optimisation ont été adoptées pour résoudre des variations de ce problème (Ghosh et Das, 2008), (Zhu *et al.*, 2012). On peut notamment citer (Andersen et Tirthapura, 2009), où les auteurs ont défini le problème de déploiement de capteurs dans une zone comme un Set Covering Problem.

Dans (Wang et Wang, 2011) l'auteur traite un problème de compromis entre la couverture et la consommation d'énergie, c'est-à-dire on procède à un déploiement aléatoire

de capteurs mobiles et le modèle énergétique utilisé prend uniquement en compte l'énergie utilisée lors des déplacements. L'optimisation de l'objectif se fera par une variante de l'algorithme PSO.

De nombreuses applications nécessitent l'ajout de contraintes de connectivité aux modèles de couverture. En effet, couvrir une zone est inutile s'il n'est pas possible de transmettre les informations recueillies à l'utilisateur. Pour remédier à cela, plusieurs travaux de recherche ont été proposés tels que :

(Shakkottai *et al.*, 2005) ont présenté les conditions nécessaires et suffisantes pour la couverture et la connectivité dans une grille, tout en développant un panel de méthodes afin de maintenir à la fois la connectivité et la couverture dans le réseau déployé.

(Bai *et al.*, 2006) ont proposé une stratégie de déploiement optimal pour assurer à la fois la couverture et la 2-connectivité, en proposant également des schémas de déploiement améliorant la prise de décision.

(Liu *et al.*, 2006) présentent un problème de planification d'activation de capteurs pré-déployés, pour assurer connectivité et couverture.

(Tian et Georganas, 2005) ont démontré que si le réseau original est connecté, et que les nœuds choisis pour l'activation sont capables de couvrir la même région que les nœuds originaux, alors le réseau formé par ces nouveaux capteurs est connecté, à condition que le rayon de communication soit au moins deux fois plus grand que celui de couverture.

(Zhao et Gurusamy, 2008) ont développé un algorithme glouton permettant de proposer une solution assurant à la fois couverture et connectivité. Ils ont également proposé une approche pour la planification du temps d'activation des capteurs.

Dans (Gupta *et al.*, 2006), les auteurs présentent la notion de couverture de capteurs connectés, qui est définie par l'ensemble des capteurs apportant une couverture totale de la zone, cet ensemble étant un graphe connecté. Les auteurs ont également prouvé que la minimisation du nombre de capteurs à déployer sous la contrainte de connectivité était un problème NP-complet.

(Zhou *et al.*, 2004) ont également intégré la notion de connectivité dans un problème de K-coverage. Le K-coverage garantit que chaque position soit couverte par au moins K capteurs (Rebai *et al.*, 2013).

2.2.2 Problèmes de contrôle de topologie

Nous faisons la distinction entre le contrôle de topologie et le contrôle de puissance ou de portée. La portée désigne la distance depuis laquelle un message peut être reçu. Le contrôle de topologie et le contrôle de portée sont deux problèmes proches. Le contrôle de topologie vise à contrôler le voisinage logique d'un nœud. Le contrôle de puissance de transmission tel que nous le définissons vise à mettre en concordance le voisinage physique et le voisinage logique d'un nœud.

Un algorithme de contrôle de topologie prend en entrée un graphe $G = (V, E)$, $E \subset V$ et donne en sortie un sous-graphe de G. Soit $Gc = (Vc, Ec)$ ce sous-graphe avec $Vc \subset V$ et $Ec \subset E$. Il peut être obtenu par l'utilisation d'un algorithme de réduction de graphe, l'ordonnancement d'activité, l'établissement de hiérarchies, ou simplement en ignorant

certains nœuds.

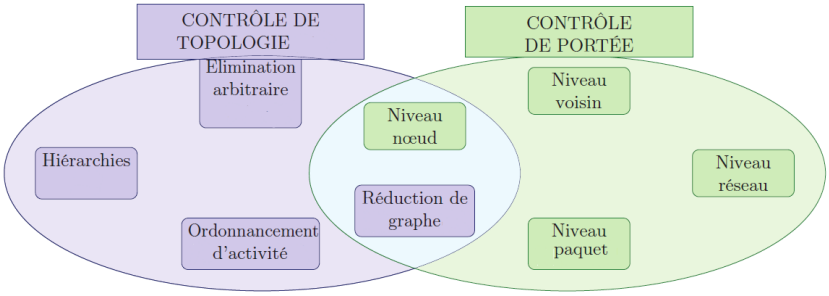


Figure 2.2 – Le contrôle de topologie vs le contrôle de portée.

Une aperçu des différentes algorithmes disponibles est fourni dans la figure 2.2. Le contrôle de topologie s'effectue par :

— L'élimination arbitraire consiste à l'ignorance de certains voisins. Chaque nœud conserve les k premiers voisins dont il a connaissance et ignore les suivants.

— L'établissement de hiérarchies (clustering) est l'approche adoptée par les algorithmes basés sur la construction de clusters ou d'ensemble dominants. Un ensemble dominant D est un ensemble de nœuds tels que tout nœud du réseau n'appartenant pas à cet ensemble possède un voisin à un saut qui est dans l'ensemble (dans ce cas, $Vc = V$). Le clustering consiste à répartir les nœuds dans des ensembles ayant des rôles différents.

— L'ordonnancement d'activité permet de réduire le nombre de nœuds actifs dans le réseau tout en préservant certaines propriétés intéressantes comme la couverture et la connexité. (dans ce cas, $Vc \subset V$).

— Un algorithme de réduction de graphe conserve tous les nœuds du réseau mais élimine certains liens considérés comme redondants de telles sortes que le graphe résultant satisfait certaines propriétés intéressantes telles que la planarité ou un degré borné.

Le premier souci dans les problèmes de contrôle de topologie est d'assurer la connectivité du réseau, mais d'après les applications, d'autres critères tels que la réduction de la consommation d'énergie et k-connectivité, peuvent être aussi envisagés dans ces problèmes. Parmi les problèmes de contrôle de topologie les plus étudiés, on cite :

1. Backbone-based topology ;
2. Strong minimum energy topology.

Backbone-based topology

Le problème Backbone-based topology est un problème traité dans les réseaux hiérarchiques. Ayant les mêmes contraintes ce problème peut être considéré comme le problème mathématique : Ensemble dominant connexe (Connected Dominating Set (CDS)) (Yuanyuan *et al.*, 2006). Un ensemble dominant (Dominating Set (DS)) d'un graphe G(V_{nodes},

E_{edges}) est un sous-ensemble de nœuds $S \subset V$, tel que chaque nœud de S a au moins une liaison avec un nœud de V qui n'appartient pas a S, comme illustré dans la figure 2.3.

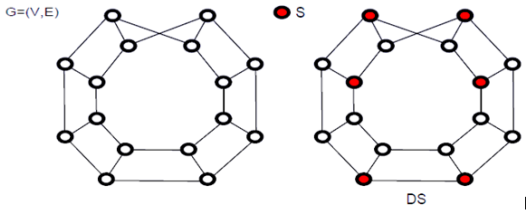


Figure 2.3 – L'ensemble dominant S du graphe G.

Soit le graphe $G = (V, E)$. On considère l'ensemble $S \subset V$. S est un ensemble dominant de G si$\forall v \in V \setminus S$, $\exists u \in S$ tel que $(u, v) \in E$.

Si on ajoute la contrainte de connectivité, S devient l'ensemble dominant connexe (CDS) présenté dans la figure 2.4.

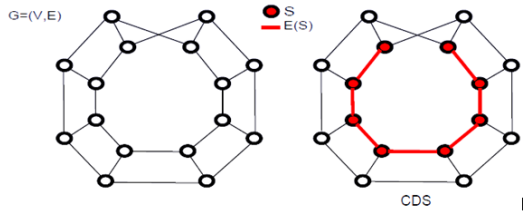


Figure 2.4 – L'ensemble dominant connexe.

Dans plusieurs applications, on cherche le plus petit ensemble dominant. Ce qui conduit au problème de l'ensemble dominant connexe minimal (Minimum Connected Dominating Set (MCDS)) qui est un problème d'optimisation NP-difficile. Un CDS ne conserve que 1-connectivité, alors que, si on prend les pannes en considération, il faut garantir la k-connectivité entre n'importe quelle paire de dominateurs ; le problème kmCDS (k-Connected m Dominating Set).

Strong minimum energy topology

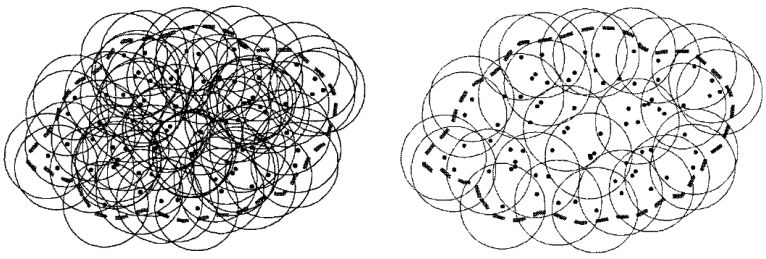
Étant donné un ensemble de nœuds capteurs dispersés dans une zone, le problème de Strong Minimum Energy Topology (SMET) est un problème d'optimisation NP-Difficile (Cheng *et al.*, 2003) qui consiste à trouver la puissance de transmission d'un nœud qui minimise la consommation énergétique de tous les nœuds tout en préservant au moins une connectivité bidirectionnelle entre deux paires de nœuds dans le réseau. Notons que la puissance de chaque nœud détermine la connectivité du réseau et par conséquent la topologie.

2.2.3 Problèmes d'ordonnancement

Les réseaux de capteurs sont généralement denses et redondants. En effet, suivant 1' application, on déploiera plus ou moins de capteurs dans un souci d'allongement de la durée de vie de l'application. À tout moment, il existe donc des capteurs qui observent une même portion de la zone de déploiement. Cette redondance est exploitée par l'ordonnancement d'activité : Ordonnancer l'activité dans un réseau de capteurs consiste à alterner les charges de façon à épuiser les nœuds équitablement. Pendant qu'une partie participe à l'application, les autres sont dans un mode passif, économisant ainsi leur énergie.

Certes, il existe des sources énergétiques rechargeables, telles que les batteries solaires, mais l'énergie n'est de toute façon pas la seule contrainte justifiant un recours à l'ordonnancement d'activité. Lorsqu'un événement se produit sur la zone de déploiement, il est d'une part inutile d'avoir un nombre élevé de capteurs qui le reporte. D'autre part, ce serait source de nombreux désagréments au niveau de l'accès au médium, toujours en raison de nœuds voulant émettre simultanément pour signaler l'événement.

Il est par conséquent crucial d'ordonnancer l'activité des capteurs dans le double but d'économiser de l'énergie et de minimiser les problèmes liés à la gestion d'une large population d'objets partageant un médium. Ainsi, à l'instar des RCSF, des changements de topologie se produisent au cours du temps, conséquences directes des mises en veille alternatives des capteurs. La diminution de la redondance est cruciale mais il ne doit pas mettre en péril l'application en cours. Parmi les critères existants, on citera la nécessaire communication des nœuds actifs avec une station puits ou encore l'existence d'une surveillance permanente sur une partie de la zone.



(a) Un réseau de capteurs déployé.

(b) Seule une partie des nœuds est nécessaire à la couverture de la zone.

Figure 2.5 – Ordonnancement d'activité avec le critère de la couverture de surface.

La figure 2.5(a) nous montre un réseau de capteurs déployé sur une zone à observer. Cette observation ne requiert pas une activité permanente de la part de tous les nœuds comme illustré sur la figure 2.5(b) où seul un sous-ensemble des capteurs suffit à une observation exhaustive de la zone.

De nombreux travaux dans le domaine des RCSF s'occupent des problèmes d'ordonnancement. Ils cherchent essentiellement des solutions pour éviter les interférences et les collisions, pour la gestion au canal, mais aussi pour la maximisation de la durée de vie ou la qualité de service. En voici deux exemples.

1. Scheduling based on Time Division Multiple Access (TDMA) scheme ;

2. Scheduling and coverage.

Scheduling based on TDMA scheme :

TDMA permet aux nœuds d'accéder au médium en affectant à chacun d'eux un time slot. Le problème d'ordonnancement concerne le calcul de ce time slot pour chaque nœud dans le réseau tel que deux nœuds voisins ne transmettent pas dans le même intervalle horaire. Le schéma d'accès TDMA est souvent utilisé pour assurer un Medium Access Control (MAC) sans collision, mais sa mise en œuvre dans les RCSF, principalement en raison des exigences de synchronisation, n'est pas facile. La majorité de ces problèmes sont NP-difficiles.

Scheduling and coverage

L'une des plus intéressantes applications de la théorie d'ordonnancement est la détermination du rapport cyclique des nœuds, qui vise à mettre les nœuds en sommeil périodiquement pour minimiser l'énergie consommée lors de l'écoute inactif. Dans un réseau de capteurs redondant avec un ensemble de cibles à surveiller, ce problème d'ordonnancement consiste à savoir comment regrouper les nœuds tel que chaque groupe (appelé couverture) couvre les cibles tout en maximisant la durée de vie du RCSF. Pour économiser l'énergie, on désactive des nœuds d'une couverture pendant une période de temps, si une autre couverture prévue pour la même zone est active. Ainsi, les couvertures sont activées d'une manière séquentielle. On peut aussi exiger un nombre de nœuds donné pour chaque couverture fixée pour assurer un seuil de tolérance aux pannes. Ce problème est un problème NP-difficile (Dhawan et Prasad, 2009).

2.2.4 Problèmes de Routage

Le routage est la méthode d'acheminement des données à la bonne destination à travers un réseau de connexion. Le but du routage est d'assurer un acheminement de coût minimal, la fiabilité des communications qui maintient le routage du trafic et garantit sa survie devant toute panne de lien ou de nœud (Misra et Enge, 2006). Le problème qui se pose dans le contexte des RCSF est l'adaptation de l'approche de routage utilisée avec le grand nombre de nœuds existants dans un environnement caractérisé par de modestes capacités de calcul, des réserves d'énergie et de capacité mémoire limitées. Il semble donc important que toute conception de protocole de routage doit étudier les problèmes importants tels que la tolérance aux fautes, l'utilisation optimale des ressources des nœuds, le passage à l'échelle et l'assurance d'une bonne qualité de service. Les protocoles de routage dédiés aux RCSF sont nombreux, mais ils sont, dans la plupart des cas, conditionnés par le type de l'application visée.

On peut distinguer les trois principaux types des protocoles de routage dans les RCSF, classés selon la structure du réseau, représentés dans la figure 2.6.

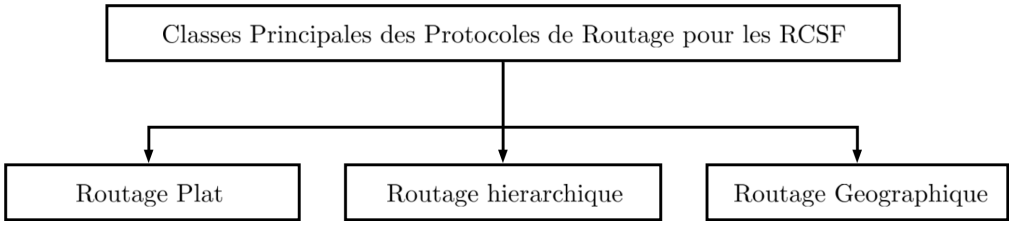


Figure 2.6 – Les classes principales des protocoles de routage

— Protocoles hiérarchiques : Ces protocoles consistent à former des réseaux dans lesquels le Sink (le niveau supérieur de la hiérarchie) est relié à un ou plusieurs autres nœuds, appelés Clusterheads, qui appartiennent à un niveau plus bas dans la hiérarchie (deuxième niveau) avec une liaison point à point. Aussi, chacun des nœuds du deuxième niveau aura également un ou plusieurs autres nœuds de niveau plus bas dans la hiérarchie (troisième niveau) reliés à lui avec une liaison point à point. Chaque ensemble de nœuds forme une sorte de motif (Cluster). motif (Cluster).

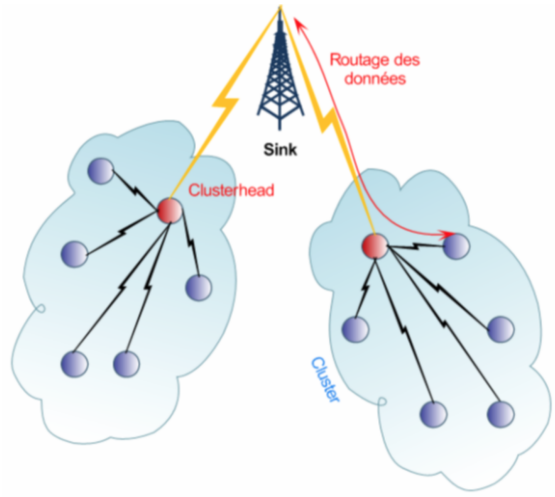


Figure 2.7 – Protocole hiérarchique

Parmi les protocoles hiérarchiques (Akkaya et Younis, 2005) on cite, le protocole Low-energy Adaptive Clustering Hierarchy (LEACH) et Cluster Based Routing Protocol (CBRP).

— Protocoles plats : Dans ces protocoles on considère que tous les nœuds sont égaux, ont les mêmes fonctions, et peuvent communiquer entre eux sans devoir passer par un nœud particulier ou une passerelle. Seul le Sink, est chargé de la collecte des données issues des différents nœuds capteurs. Au cas où la destination ne fait pas partie du voisinage de la source, les données seront transmises en utilisant les sauts multiples à travers les nœuds intermédiaires comme c'est illustré dans la figure 2.8.

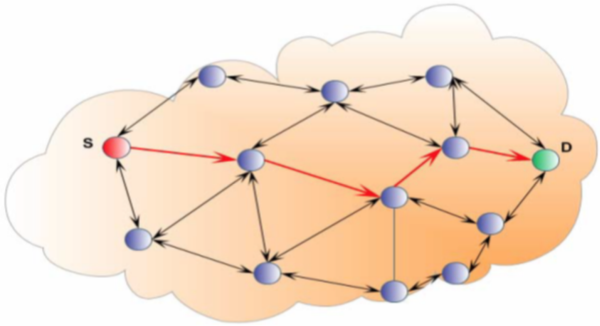


Figure 2.8 – Protocole plat

Parmi les protocoles plats (Akkaya et Younis, 2005) on peut citer le protocole Direct Diffusion, et le protocole Sensor Protocols for Information via Negotiation (SPIN).

— Protocoles géographiques : Contrairement aux autres protocoles, aucune information sur la topologie globale du réseau n'existe, car chaque nœud choisit le prochain saut parmi ses voisins en se basant sur la localisation de la destination, en utilisant Global Positioning System (GPS), ou par les mécanismes d'auto-localisation.

Par exemple le protocole Greedy Perimeter Stateless Routing (GPSR) (Akkaya et Younis, 2005), est un protocole géographique.

2.3 Méthodes de résolution des problèmes d'optimisation des RCSF :

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données. Étant donné l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (Recherche Opérationnelle (RO)) et en intelligence artificielle (Intelligence Artificielle (IA)). Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes) qui garantissent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

On retrouve, dans ce graphe, les principales distinctions opérées. On différencie, en premier lieu, l'optimisation combinatoire de l'optimisation continue. Pour l'optimisation combinatoire, on a recours aux méthodes approchées lorsqu'on est confronté à un problème difficile ; dans ce cas, le choix est parfois possible entre une heuristique "spécialisée", entièrement dédiée au problème considéré, et une métaheuristique.

Pour l'optimisation continue, on sépare sommairement le cas linéaire du cas non linéaire, où l'on retrouve le cadre de l'optimisation difficile ; dans ce cas, une solution pragmatique peut être de recourir à l'application répétée d'une méthode locale qui exploite,

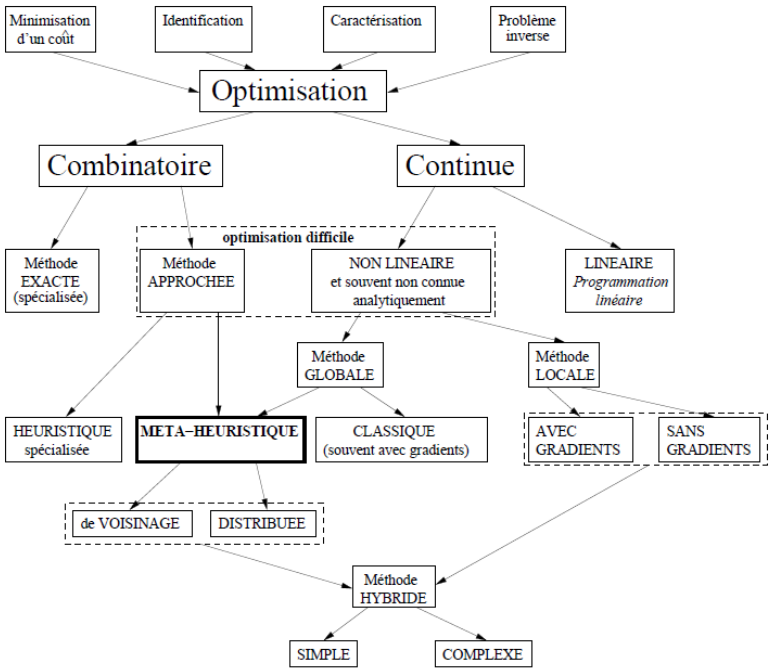


Figure 2.9 – Classification générale des méthodes d'optimisation.

ou non, les gradients de la fonction objectif. Si le nombre de minima locaux est très élevé, le recours à une méthode globale s'impose : on retrouve alors les métaheuristiques, qui offrent une alternative aux méthodes classiques d'optimisation globale, celles-ci requérant des propriétés mathématiques restrictives de la fonction objectif.

Parmi les métaheuristiques, on peut différencier les métaheuristiques "de voisinage", qui font progresser une seule solution à la fois (recuit simulé, recherche tabou, etc.), et les métaheuristiques "distribuées", qui manipulent en parallèle toute une population de solutions (algorithmes génétiques, etc.). Enfin, les méthodes hybrides associent souvent une métaheuristique et une méthode locale. Cette coopération peut prendre la simple forme d'un passage de relais entre la métaheuristique et la technique locale, chargée d'affiner la solution. Mais les deux approches peuvent aussi être entremêlées de manière plus complexe.

Citons quelques exemples de méthodes illustrant ces différentes catégories :

* Méthode exacte : cette méthode va rechercher l'optimum global. Pour cela, elle effectue, en général, une énumération des solutions de l'espace de recherche.
* Méthodes approchées :
  - Méthodes Heuristiques : elles sont développées pour résoudre un type de problème en particulier. Elles ne fonctionnent que sur le type de problème pour lequel elles ont été développées.
  - Méthodes Métaheuristiques : elles sont fondées sur une idée générale (le parallèle avec un processus naturel). Elles peuvent s'adapter à n'importe quel type de problème et donnent de bons résultats.
* Méthodes non-linéaires :
  - Globale :
    - Métaheuristiques : l'idée directrice sur laquelle repose une métaheuristique

est suffisamment générale pour être transposable facilement aux problèmes d'optimisation continus.
- Classiques : ces méthodes peuvent être considérées comme les méthodes "de base" de l'optimisation. Dans cette famille, on trouve la méthode de descente du gradient. Cette méthode, quand elle est appliquée à une fonction convexe, permet de trouver un optimum global.

— Locale :
- Avec gradient : ces méthodes exploitent le gradient d'une fonction pour effectuer la recherche de l'optimum. Quand cette technique est appliquée à un problème d'optimisation continue de forme générale, il n'est en général pas possible de trouver l'optimum global du problème. On obtient alors un optimum local.
- Sans gradient : cette famille regroupe une grande diversité de méthodes.

2.3.1 Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de séparation et évaluation progressive ou les algorithmes avec retour arrière. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Par exemple :

— Dijkstra : Pour optimiser quelques problèmes de Routage des RCSF (Gogu *et al.*, 2011) ;
— Spanning Tree : Pour optimiser des problèmes de Contrôle de topologie des RCSFGogu *et al.* (2011) ;
— Diagramme de Voronoi : Pour optimiser des problèmes de Couverture[8] et de Contrôle de topologie des RCSF (Gogu *et al.*, 2011) ;

Malgré les progrès réalisés (notamment en matière de la programmation linéaire en nombres entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante.

2.3.2 Méthodes approchées : Métaheuristiques

Les méthodes approchées constituent une alternative très intéressante, utilisées depuis longtemps, pour traiter les problèmes d'optimisation de grande taille. Au début cette mission a été consacrée au méthodes heuristiques, qui sont des algorithmes approchés permettant d'identifier en temps polynomial au moins une solution réalisable rapide, pas obligatoirement optimale. L'usage des heuristiques est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantit quant à la qualité de la solution calculée. Les heuristiques peuvent être classées en deux catégories :

— Méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue.
— Méthodes de fouilles locales qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage.

Des progrès importants ont été réalisés avec l'apparition d'une nouvelle génération de méthodes approchées puissantes et générales, appelées métaheuristiques. Une métaheuristique est constituée d'un ensemble de concepts fondamentaux (par exemple, la liste tabou et les mécanismes d'intensification et de diversification pour la métaheuristique tabou), qui permettent d'aider à la conception de méthodes heuristiques pour un problème d'optimisation. Ainsi les métaheuristiques sont adaptables et applicables à une large classe de problèmes. Les métaheuristiques sont représentées essentiellement par les méthodes de voisinage comme le recuit simulé et la recherche tabou, et les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution. Grâce à ces métaheuristiques, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classiques de plus grande taille et pour de très nombreuses applications qu'il était impossible de traiter auparavant.

Plusieurs problèmes dans les RCSF sont NP-difficiles nécessitant des méthodes de résolution performantes et robustes, telles que les métaheuristiques. Récemment, plusieurs métaheuristiques ont été utilisées pour optimiser des problèmes dans les RCSF, pour cela nous allons présenté plus de détails sur ces techniques dans ce qui suit .

Métaheuristiques inspirées de la nature

Les métaheuristiques utilisent des recherches stratégiques afin d'explorer plus efficacement l'espace de recherche, elles balancent sa recherche entre une focalisation sur les zones prometteuses et une exploration de nouvelles zones. Ces méthodes commencent par un ensemble de solutions initiales ou une population initiale, et après, elles examinent étape par étape une séquence de solutions pour atteindre, ou de s'approcher de, la solution optimale du problème. Les métaheuristiques ont plusieurs avantages par rapport aux algorithmes traditionnels. Les deux avantages les plus importants sont la simplicité et la flexibilité. Les métaheuristiques sont souvent simples à implémenter, pourtant, elles sont capables de résoudre des problèmes complexes avec la capacité de s'adapter à plusieurs problèmes d'optimisation du monde réel, à partir du domaine de la recherche opérationnelle, d'ingénierie vers l'intelligence artificielle. plusieurs exemples sont cités par Yang et Yang et al. dans (Yang, 2010a; Yang et Deb, 2013), Gandomi et al. dans (Gandomi *et al.*, 2011, 2012, 2013), et Yang et Gandomi dans (Yang *et al.*, 2013). Ces algorithmes sont très flexibles, et ils ont la capacité de traiter des problèmes avec des fonctions objectif de différentes propriétés, qu'elles soient continues, discrètes ou mixtes.

Dans la littérature, aucun algorithme n'est efficace pour la résolution d'une grande partie des problèmes d'optimisation combinatoire NP-difficiles. Le besoin de trouver rapidement une bonne solution (pas nécessairement la meilleure) à ces problèmes est le motif de développer différents algorithmes approximatifs comme les métaheuristiques (Blum et Roli, 2003a; Glover et Kochenberger, 2003; Talbi, 2009). En effet, les métaheuristiques ont prouvé leur potentiel et efficacité à résoudre une large variété de problèmes d'optimisation. Elles sont en principe simples à implémenter tout en traitant des problèmes complexes. Elles peuvent être adapter à un grand nombre de problèmes d'optimisation du monde

réel tout en gérant la diversité des fonctions objectif, soit ceux de caractère continu, discret ou mixte. Ce qui explique la faciliter de s'adapter avec un ensemble de paramètres simultanément.

Inspirées de la nature, les métaheuristiques utilisent des stratégies de recherche pour explorer l'espace de recherche et par la suite exploiter parfaitement les régions prometteuses de cet espace. Les métaheuristiques citées dans le reste de ce chapitre sont parmi les métaheuristiques inspirées de la nature les plus étudiées dans la littérature. Chacune de ces métaheuristiques est caractérisées par sa propre façon de résoudre les problèmes et sa source d'inspiration. Les métaheuristiques visent, toutes, à travers leur inspiration à adopter un concept important durant leur recherche des solutions optimales, qui est l'intensification et la diversification (I&D).

En citant l'intensification et la diversification, l'opposition et le conflit ne se présentent jamais. Elles sont conçues comme deux éléments où l'un exige l'existence de l'autre (avec un certain degré) pour augmenter la probabilité de trouver l'optimum global. Nous devons garder l'exploration de nouvelles régions, tout en encourageant l'investissement d'un peut de temps dans les régions prometteuses. Généralement, l'intensification et la diversification doivent être équilibrées intelligemment et dynamiquement. I&D se figurent comme des techniques (opérateurs, actions, ...) ou des stratégies au sein d'une métaheuristique (Yang, 2012).

Les métaheuristiques inspirées de la nature sont en principe reposées sur la notion d'intensification et de diversification, afin de sélectionner les potentielles meilleures solutions qui sont autour de la meilleure solution courante, ou randomiser la recherche vers des région loin de la meilleure solution dans le but de tomber sur des nouvelles solutions bien meilleures (ou qui mènent vers des solutions bien meilleures). Les métaheuristiques inspirées de la nature les plus citées dans la littérature sont les algorithmes génétiques, l'optimisation par essaims particulaires, l'optimisation par colonie de fourmis, l'algorithme des lucioles, l'optimisation par les colonies des abeilles, l'algorithme inspiré des chauve-souris, la recherche du coucou et encore d'autres métaheuristiques.

Genetic Algorithms (GA)

Les Algorithmes Génétiques "Genetic Algorithms (GA)" font partie de la famille des Algorithmes Évolutionnistes qui sont inspirés en principe du concept de sélection naturelle élaboré par Charles Darwin (Darwin, 1998). Ils sont conçus pour simuler une population d'individus, sur laquelle on applique différents opérateurs (croisement et mutation) et qui sont soumis à une sélection (figure 2.10). Ce processus est répété d'une génération à l'autre. Si la sélection s'effectue en se basant sur la fonction objectif, alors la population tend à s'améliorer. Le point fort de cet algorithme (et d'autre métaheuristiques) est qu'il ne nécessite pas une connaissance détaillée du problème : on peut représenter celui-ci par une boîte noire portant des entrées (les variables) et des sorties (selon les fonctions objectif). On ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties (Ghedira, 2007). Les algorithmes génétiques ne garantissent pas la solution optimale de l'espace de recherche. Mais, on peut constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes plus classiques, dans un même temps de calcul.

En 1975, John Holland a publié son livre "Adaptation in Natural and Artificial Systems" (Holland, 1975) pour mettre en évidence et expliquer rigoureusement les processus

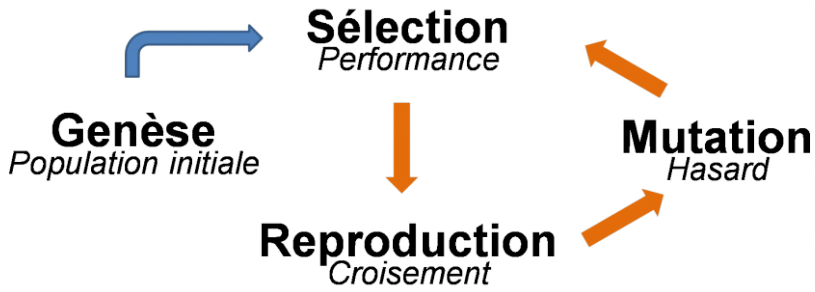


Figure 2.10 – Procédure générale des algorithmes génétiques.

d'adaptation des systèmes naturels et concevoir des systèmes artificiels qui utilisent ces mécanismes. Son algorithme génétique consiste à faire évoluer une population de chromosomes (représentent les solutions du problème étudié) vers une nouvelle population en utilisant une sélection naturelle et en appliquant ensuite les opérateurs inspirés de la génétique en l'occurrence le croisement et la mutation. Il maintient une population d'individus pour chaque génération. Certains individus (les parents) de la population sont sélectionnés et transformés avec des opérateurs génétiques afin de générer une nouvelle population (les enfants). Chaque individu est évalué par une fonction d'évaluation. Le processus s'arrêtera une fois qu'il aura trouvé une solution du problème ou quand un nombre maximal de générations aura été généré.

Les étapes clés des algorithmes génétiques sont la représentation des chromosomes qui nécessite l'encodage de la fonction d'optimisation comme des tableaux binaires ou des chaînes de caractères, la manipulation des chaînes par les opérateurs génétiques, et la sélection des individus en fonction de leur aptitude. Cela se fait souvent par la procédure suivante (Yang, 2010a) :

— Codage de la fonction objectif (d'optimisation).
— Définir le critère de la sélection.
— Population initiale des individus.
— Évaluer le coût de tous les individus de la population.
— Création d'une nouvelle population en effectuant le croisement et la mutation.
— Évolution de la population jusqu'à la vérification des critères d'arrêt.
— Décodage du résultat pour obtenir la solution du problème.

La notion d'intensification et diversification est adopté par les algorithmes génétiques. L'intensification consiste à pousser la recherche dans un sous-espace d'individus qui semble prometteur alors que la diversification permet de passer et découvrir de nouveaux sous-espaces de recherche dans le but d'améliorer la fonction objectif. Nous pouvons considérer en gros que l'opérateur de mutation fait plutôt de l'exploration et le croisement exploite les régions trouvées.

La formulation de la fonction objectif, l'utilisation de taille de la population, le choix des paramètres importants tels que le taux de croisement et de mutation, et les critères de sélection de la nouvelle population devraient être soigneusement effectuées. Après tout choix inapproprié, il sera difficile pour l'algorithme de converger, ou il produit simplement des résultats insignifiants. Malgré ces inconvénients, l'algorithme génétique reste un

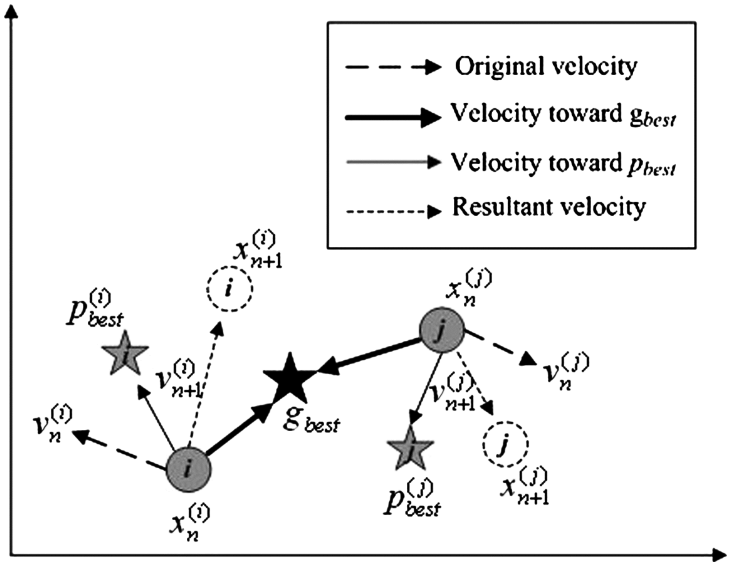


Figure 2.11 – Déplacement des particules dans l'espace de recherche (Wang *et al.*, 2010).

algorithme largement utilisé dans l'optimisation.

Particle Swarm Optimization (PSO)

En 1995, le psychologue social américain James Kennedy, et l'ingénieur Russell C. Eberhart ont introduit l'optimisation par essaims particulaires, "PSO" (Kennedy et Eberhart, 1995). Grosso modo, PSO est un algorithme d'optimisation inspiré de l'intelligence en essaim de poissons et d'oiseaux et même du comportement humain (Yang, 2010a). Le principe est de commencer par une population de solutions aléatoires et faire la recherche pour un optimum par la mise à jour des générations, et fouiller l'espace des solutions en suivant les optimums actuels. Chaque particule conserve la trace de ses coordonnées, dans l'espace de recherche, qui alimente (pBest) la meilleure solution réalisée jusqu'à maintenant. L'autre meilleure solution (lBest) suivie par les particules est la meilleure solution obtenue jusqu'à présent par toutes les particules voisines. Quand une particule prend toute la population comme ses voisines topologiques, la meilleure valeur est une meilleure solution globale (gBest).

L'optimisation par essaims de particules est une technique d'optimisation stochastique utilisant une population. elle se base sur une communication globale entre les particules de la population. Le premier principal avantage est qu'elle a un petit nombre de paramètres de configuration. Deuxièmement, l'implémentation est valable pour différents problèmes avec seulement quelques modifications. C'est pour ça qu'elle a été appliquée dans une vaste gamme de domaines de recherche et d'applications récentes. PSO considère les solutions dans l'espace de recherche comme des particules (figure 2.11), chaque particule se déplace avec une vitesse variable vers sa propre meilleure position $pBest$ dans le passé et la meilleure position globale $gBest$ selon les équations 2.1 :

$$X_{n+1}{}^{(i)} = X_n{}^{(i)} + V_{n+1}{}^{(i)}, \tag{2.1}$$

$$V_{n+1}{}^{(i)} = wV_n{}^{(i)} + c_1 r_1 (P_i - X_n^{(i)}) + c_2 r_2 (P_g - X_n^{(i)}), \tag{2.2}$$

où, dans l'itération $(n+1)$, X^i $(i = 1, 2, \ldots, m)$ est le $i^{ème}$ parmi m particules, qui se déplace avec la vitesse V^i. P_i et P_g sont respectivement $pBest$ et $gBest$. w, c_1 et c_2 représentent les paramètres qui sont sélectionnés, dans l'intervalle $[0, 1]$, pour contrôler le comportement de PSO. r_1 et r_2 sont aléatoirement générés dans $[0, 1]$.

La vitesse décroit lors du rapprochement de la particule vers $pBest$ et $gBest$. PSO a prouvé son efficacité face aux différents problèmes, mais la discussion est encore ouverte concernant son efficacité face aux problèmes combinatoires NP-difficiles. La plus grande contrainte est comment introduire le concept de vitesse dans un espace combinatoire. Plusieurs tentatives ont réussi de proposer une adaptation de PSO à un ensemble de problèmes d'optimisation combinatoire. Mais, lors de la transition de l'espace continu vers l'espace combinatoire, ces adaptations sont incapables de garder les propriétés entre les deux espaces. Cette contrainte est toujours rencontrée dans toutes les adaptations de la distance, la vitesse ou le déplacement dans l'espace continu vis-à-vis du combinatoire. Afin de contourner cette contrainte, plusieurs alternatives sont étudiées, comme l'approche de Shi et al. (Shi *et al.*, 2007), basée sur "Swap operator" développé par Wang et al. (Wang *et al.*, 2003). Les résultats des tests de cette approche sont comparés avec notre algorithme dans ce travail.

Ant Colony Optimization (ACO)

Les colonies de fourmis nous ont appris que les sociétés d'insectes, qui possèdent des mécanismes extrêmement simples, sont capables de réaliser en groupe des tâches complexes. La compréhension du comportement collectif des insectes sociaux a permis de mettre au point une nouvelle famille de méthodes de résolution de problèmes d'optimisation (Dorigo et Birattari, 2007). L'intelligence en essaim est une propriété des systèmes composés d'agents "non intelligents" avec des capacités individuelles limitées réalisant collectivement un comportement intelligent. Cette expression a été initialement utilisée dans le contexte des systèmes à robots cellulaires pour décrire l'auto-organisation des agents mécaniques simples grâce à l'interaction du plus proche voisin. Bonabeau, Dorigo et Theraulaz ont élargi la définition pour y inclure toute tentative de concevoir des algorithmes inspirés des comportements collectifs des colonies d'insectes sociaux et autres sociétés animales. Cela inclut les comportements de certaines fourmis, abeilles, guêpes, cafards, etc. Les solutions à de nombreux problèmes ont déjà été trouvées en utilisant les métaheuristiques basées sur le comportement des fourmis en quête de nourriture (Dorigo *et al.*, 2000).

L'optimisation par colonie de fourmis (ACO) est considérée comme une métaheuristique de construction de solutions qui s'inspire du comportement des fourmis afin d'optimiser la longueur de leur chemin entre la colonie et la source de nourriture (figure 2.12). Selon Dorigo et Di caro (Dorigo et Birattari, 2010), la version de base du ACO est conçue pour résoudre le problème du voyageur de commerce (Dorigo *et al.*, 1997). Lorsqu'une fourmi commence par un déplacement aléatoire, à partir d'une ville r à s, en passant par l'arc qui connecte ces deux villes, elle y dépose une quantité de phéromone. Les autres fourmis qui suivent cette fourmi auront le choix de considérer le phéromone déposé et

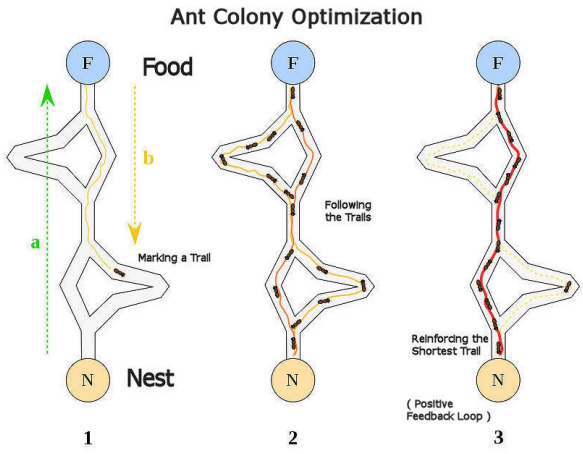


Figure 2.12 – Optimisation du chemin, par les fourmis, au cours des itérations (Esnault, 2012).

passer sur cet arc ou de prendre un arc aléatoirement selon la formule suivante :

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)]^\beta}{\sum_{u \in M_k}[\tau(r,u)].[\eta(r,u)]^\beta} \; if \; s \notin M_k \\ 0 \; otherwise \end{cases} \tag{2.3}$$

où $p_k(r,s)$ est la probabilité de choisir (par la fourmi k) le passage de la ville r vers la ville s, M_k est l'ensemble des villes visitées et mémorisées par la fourmi k, $\tau(r,s)$ est la quantité de phéromone déposé sur l'arc (r,s), $\eta(r,s)$ est la fonction heuristique, qui peut être l'inverse de la distance entre les villes r et s, β est un paramètre de poids relatif à l'importance de l'arc, utilisé par ACO pour promouvoir les arc des meilleures solutions, avec un contrôle centralisé. De l'autre côté les mauvaises décisions dans le passé sont oubliées au cours des itérations à l'aide de l'évaporation du phéromone (Colorni *et al.*, 1991).

Bee Colony Optimization (BCO)

Les algorithmes inspirés de la colonie des abeilles "Bee Colony Optimization (BCO)" ont commencé à émerger au cours des dix dernières années comme un outil prometteur et puissant. Ils ont été développés dans quelques années de manière indépendante par plusieurs groupes de chercheurs. En 2004 Nacrani et Tovey ont formulé "Honey Bees" (Nakrani et Tovey, 2004) pour allouer des ordinateurs entre différents clients sur les serveurs d'hébergement web. Vers l'année 2006, Basturk et Karaboga ont développé un algorithme de colonie d'abeilles artificielles "Artificial Bee Colony (ABC)" pour l'optimisation de fonctions numériques (Basturk et Karaboga, 2006).

La colonie des abeilles a pour objectif d'optimiser l'efficacité globale de la collecte de nectar. Les butineuses sont attribuées à différentes sources de nourriture de façon à maximiser l'apport total de nectar. La répartition des abeilles est en fonction de plusieurs facteurs tels que la richesse du nectar et la proximité de la ruche. Le point le plus important dans la danse des abeilles est leur force d'attirer l'attention des autres. Dans la colonie, les abeilles sont divisées en trois groupes :

— Les abeilles employées (butineuses),
— Les abeilles spectatrices (observatrices),
— Les scoutes (exploratrices),

Pour chaque source de nourriture il y a seulement une abeille employée. Le nombre d'abeilles employées est égal au nombre des sources de nourritures. L'abeille employée d'une source de nourriture abandonnée est forcée de devenir une "scoute" pour la recherche de nouvelles sources de nourritures de façon aléatoire. Les abeilles employées partagent des informations avec les abeilles spectatrices de telle sorte que les spectatrices ont la possibilité de choisir une source de nourriture. Contrairement à l'algorithme des abeilles qui a deux groupes d'abeilles (abeilles butineuses et observatrices), les abeilles dans ABC sont plus spécialisées (Yang, 2010a).

Après la localisation d'une source de nourriture, l'abeille retourne à la ruche et déclenche le processus de communication, sous forme de danses, avec le reste de la ruche. Les abeilles qui observent ont le choix de suivre les indices fournis de cette danse ou de les ignorer. Nous avons, donc, en principe deux facteurs qui affectent le butinage : le poids de la danse et la probabilité d'ignorer la danse et partir à une recherche aléatoire.

Ces deux facteurs (la force de la danse et la probabilité d'une exploration aléatoire de l'espace) sont présentés d'une manière simple par Guijano et Passino dans la version "Honey bee" (Quijano et Passino, 2010) par l'équation 2.4 de probabilité p_i d'une abeille observatrice.

$$p_i = \frac{w_i^j}{\sum_{i=1}^{n_f} w_i^j} \tag{2.4}$$

où w_i^j est le poids de la danse de l'abeille i dans l'itération $t = j$, n_f est le nombre des abeilles en butinage. Le nombre des abeilles observatrices est $N - n_f$ où N est le nombre total des abeilles. La probabilité d'exploration est définie sous forme d'une expression Gaussienne, comme suit :

$$p_e = 1 - p_i = e^{-\frac{w_i^2}{2\sigma^2}}, \tag{2.5}$$

telle que σ (fixée) est la volatilité de la colonie.

Cuckoo Search (CS)

La recherche du coucou Cuckoo Search (CS) est une métaheuristique développée récemment par Xin-She Yang et Suash Deb en 2009, initialement conçue pour résoudre les fonctions multimodales. Elle prend comme une base les idée suivantes :

— Chaque coucou pond un œuf à la fois et choisit un nid aléatoirement ;
— Un bon nid de bonne qualité peut passer vers une nouvelle génération ;
— Le nombre de nids hôtes est fixé, et un œuf posé par un coucou peut être découvert par l'oiseau hôte suivant une probabilité $p_\alpha \in [0, 1]$.

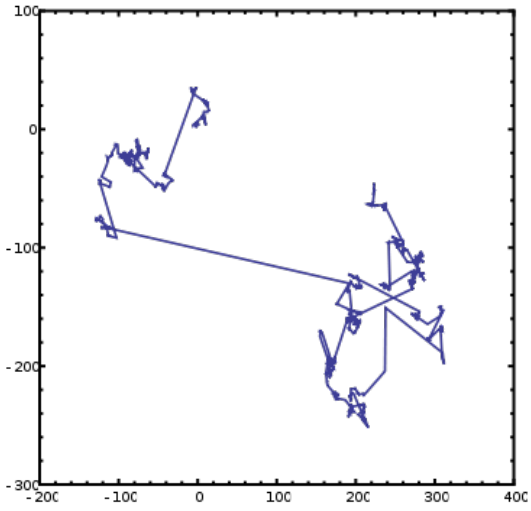


Figure 2.13 – Exemple de 1000 pas par les vols de Lévy en 2 dimensions.

Un coucou i génère une nouvelle solution $x_i^{(t+1)}$ via les vols de Lévy, selon l'équation (2.6)

$$x_i{}^{(t+1)} = x_i{}^{(t)} + \alpha \oplus Lévy(s, \lambda) \tag{2.6}$$

où α est la longueur du pas suivant la distribution des vols de Lévy montrée dans l'équation (2.7) :

$$Lévy(s, \lambda) \sim s^{-\lambda}, \quad (1 < \lambda \leq 3) \tag{2.7}$$

qui a une variance et une moyenne infinies (Yang et Deb, 2009). Ici, s est la taille du pas tiré de la distribution Lévy (figure 2.13).

Algorithme 2.1 Recherche du Coucou

Fonction objectif $f(x), x = (x_1, \ldots, x_d)^T$
Générer la population initiale de n nids $x_i (= 1, 2, \ldots, n)$
tantque (t <MaxCeneration) ou (le critère d'arrêt) **faire**
  Obtenir un Coucou aléatoirement par les vols de Lévy
  Évaluer sa qualité/fitness F_i
  Choisir un nid parmi n (soit j) aléatoirement
  si ($F_i < F_j$)(minimisation) **alors**
    Remplacer j par i
  finsi
  Une fraction (p_a) des mauvais nids est abandonnée et des nouveaux sont construits
  Garder les meilleures solutions (ou nids avec des solutions de qualité)
  Classer les solutions et trouver la meilleure actuelle
fin tantque
Post-processus des résultats et visualisation

L'algorithme de la recherche du coucou se résume autour des règles idéales suivantes :

— Chaque œuf du coucou dans un nid représente une solution.

— Chaque oiseau de coucou pondra un seul œuf à la fois, et choisira sont nid de façon "*aléatoire*". Donc, chaque individu de la population des coucous a le droit de

générer aléatoirement une seule nouvelle solution.
— Les meilleurs nids de meilleure qualité d'œufs nous mèneront vers les nouvelles générations. Ici, on a introduit implicitement la notion d'intensification ou la recherche autour des meilleures solutions.
— Certaines nouvelles solutions doivent être générées par les vols du Lévy autour de la meilleure solution obtenue jusqu'ici. Cela accélérera la recherche locale.
— Le nombre de nids hôtes est fixe, et l'œuf pondu par l'oiseau est découvert par l'hôte avec une probabilité $p_a \in [0, 1]$. Dans ce cas, l'oiseau hôte choisi de se débarrasser de l'œuf, ou d'abandonner le nid et de reconstruire un autre nid quelque part. Pour la simplification, cette dernière hypothèse sera approximée par la fraction p_a des n nids qui sont remplacés par des nouveaux (nouvelles solutions aléatoires).
— Une fraction importante des nouvelles solutions doivent êtres générées par randomisation vers des régions lointaines et dont les emplacements doivent être assez loin de la meilleure solution actuelle, ce qui fera que le système ne sera pas pris au piège dans un optimum local.
— Chaque nid peut contenir plusieurs œufs signifiants un ensemble de solutions.

Autres métaheuristiques

La présentation ci-dessus des métaheuristiques inspirées de la nature n'est pas exhaustive. Nous avons choisi les métaheuristiques les plus citées, les plus étudiées et les plus appliquées, car le but de cette présentation est la compréhension des méthodes et techniques utilisées dans les processus des inspirations des phénomènes et systèmes naturels. Ainsi, la façon de passer d'une version plus ancienne d'un algorithme à une autre plus récente et plus robuste. Nous citons entre autres "Bat-Inspired Algorithme" (Yang, 2010b) qui se base sur le comportement d'écho-location des microchiroptères, avec le taux d'émission et l'intensité des impulsions, "monkey search algorithm" (Mucherino et Seref, 2007) adoptant les notions ne montée et surveillance-saut des singes lors de leur déplacement sur les arbres, "Fruit Fly Optimization Algorithm" (Abidin *et al.*, 2010) inspiré du comportement des mouches de fruit durant leur recherche de nourriture, etc.

2.4 Routage basé sur des métaheuristiques inspirées de la nature

Le routage dans les RCSF diffère du routage dans les réseaux de communication traditionnels par l'absence d'infrastructure, la non fiabilité des liens, l'énergie limitée et la topologie dynamique,... (Al-Karaki et Kamal, 2004). D'autre part, ce problème a été qualifié comme étant un problème d'optimisation NP-difficile. Cette classe de problèmes ne peut être résolue par des méthodes exactes avec un temps de calcul qui nous permettrais d'avoir une solution dans des délais corrects. L'utilisation des métaheuristiques pour les problèmes d'optimisation combinatoire (Blum et Roli, 2003b) est donc un choix justifié, étant donné qu'elles fournissent de bonnes solutions dans un laps de temps négligeable par rapport à celui des méthodes exactes.

Espèces de la nature	**Comportement**	**Applications au Routage des RCSF**
Swarm 	Le comportement de l'essaim est un comportement collectif manifesté par des animaux de taille similaire, qui se regroupent ou migrent dans une certaine direction. Les essaims représentent un animal individuel qui suit trois règles : 1) Se déplacer dans la même direction que leurs voisins. 2) Rester toujours près de leurs voisins. 3) Se tenir à l'écart des collisions avec leurs voisins	(Guru *et al.*, 2005; Latiff *et al.*, 2007; Kuila et Jana, 2014; EL Ghazi et Ahiod, 2016)
Abeilles 	Combine efficacement la réplication et l'évasion pour que le réseau continue à fournir des données pendant une longue période lors d'une attaque de congestion.	(Karaboga et Basturk, 2007; Okdem *et al.*, 2011; Kavian *et al.*, 2013; Zheng et Luo, 2014)
Fourmis 	Le trait le plus caractéristique du comportement des fourmis est la sociabilité. Les fourmis n'agissent pas individuellement ; ils se comportent en fonction des besoins de la colonie et selon les rôles dictés par la caste dans laquelle ils sont nés.	(Zhang *et al.*, 2004; Okdem et Karaboga, 2009; Fathima et Sindhanaiselvan, 2013; El Ghazi *et al.*, 2014)
Termites 	Les termites passent d'une phase non coordonnée à une phase coordonnée seulement si leur densité est supérieure à la valeur de seuil.	(Zungeru *et al.*, 2012b,a; Lin *et al.*, 2012; Sharma *et al.*, 2013)
Les bancs de poisson 	Tout groupe de poissons qui s'est réuni collectivement dans une localité est appelé banc de poisson qui peuvent être structurés ou non structurés. Un banc de poisson non structuré est un groupe d'espèces de tailles mélangées qui se rassemblent au hasard près d'une ressource voisine telle que la nourriture ou les sites de nidification.	(Kuila et Jana, 2014; Yiyue *et al.*, 2012; Neshat *et al.*, 2014)
Coucou 	Le comportement des espèces de coucou consiste à déposer les œufs dans les nids d'autres oiseaux hôtes (d'autres espèces). Certains oiseaux hôtes peuvent engager un conflit direct avec les coucous intrus. Par exemple, si un oiseau hôte découvre que les œufs ne sont pas les tiens, il jette les œufs étrangers ou abandonne simplement son nid pour construire un nouveau nid ailleurs.	(Dhivya *et al.*, 2011b,a; Bhatti et Raina, 2014; Raj et Sumathi, 2014)

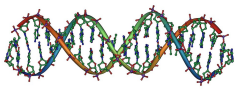
Génétiques 	Basé sur le comportement des gènes. Les systèmes génétiques sont des approches permettant de comprendre le flux d'informations biologiques qui sous-tend des traits complexes. Ils utilisent une gamme de méthodes expérimentales et statistiques pour intégrer quantitativement les phénotypes intermédiaires, tels que les transcrits, les protéines ou les niveaux de métabolites, dans les populations qui varient pour les traits d'intérêt.	(Hussain *et al.*, 2007; Rao et Kumar, 2011; Bayraklı et Erdogan, 2012; Shurman *et al.*, 2013; Nimbalkar *et al.*, 2015)

Tableau 2.1 – L'inspiration de la nature

Les chercheurs ont abordé les défis relevés par le routage dans les RCSF en adoptant plusieurs stratégies d'optimisation (Tableau 2.1). Il existe une gamme variée de méta-heuristiques inspirées de la nature utilisées pour optimiser le routage dans les réseaux de capteurs sans fil, nous citons entre autres :

Les algorithmes génétiques (GA)

Les algorithmes génétiques (GA) dans (Hussain *et al.*, 2007) où les auteurs introduisent un protocole de routage adaptatif qui prend en compte certains concepts des nœuds pour élire les cluster heads optimales. Les algorithmes génétiques sont appliqués pour optimiser les paramètres contrôlables du protocole proposé, afin de prolonger la durée de vie du réseau et optimiser la consommation énergétique.

Dans (Rao et Kumar, 2011), l'objectif fondamental des auteurs est de déterminer les clusters, la position des clusters head et d'optimiser la consommation d'énergie des nœuds et du réseau. Le réseau sans fil est représenté par des bits du chromosome : '0' représente un nœud membre et '1' représente un cluster head. Les opérateurs génétiques utilisés dans la méthode proposée sont la sélection, la mutation, la fonction fitness et le croisement. Comparée au protocole LEACH, l'approche proposée est meilleure en répartition uniforme des clusters head dans un réseau, en énergie résiduelle totale et en nombre de nœuds vivants.

Les auteurs dans (Bayraklı et Erdogan, 2012), ont proposé une méthode basée sur le clustering qui est similaire à LEACH et se compose de deux phases à savoir la phase de mise en place et la phase d'état stable. Dans la première phase, les nœuds membres choisissent eux-mêmes de rejoindre le Cluster Head (CH) le plus proche. Dans la deuxième phase, les données sont transmises au CH en utilisant la technique TDMA pour envoyer ces données à la station de base. Les clusters ne changent pas à chaque tour (un tour c'est l'exécution des deux phases) alors que le CH peut être changé si l'énergie résiduelle qui y reste est inférieure à l'énergie moyenne de tous les nœuds membres. Dans ce cas, le nœud qui a l'énergie résiduelle la plus élevée dans le groupe devient le nouveau CH et l'ancien CH devient un nœud membre. Le réseau peut être représenté génétiquement par des chromosomes, car le CH sera désigné par '1' et les nœuds membres seront désignés par

'0'. La population initiale est générée de manière aléatoire et chaque nœud est évalué pour son aptitude. Basé sur la forme physique, le chromosome le plus adapté est sélectionné pour le croisement et la mutation. Les principales limitations de cette approche consistent en absence du paramètre d'énergie dans la fonction fitness et la rotation des clusters head, alors que les clusters ne changent pas pendant toute la durée de vie du réseau.

Dans ce papier (Shurman *et al.*, 2013), les auteurs ont combiné les algorithmes génétiques avec la classification hiérarchique afin de minimiser la communication à longue distance entre les capteurs et le sink, et réduire la consommation d'énergie dans le réseau. Le fonctionnement de cette approche suit un enchainement qui commence par le déploiement d'un nombre de nœuds, la localisation de la station de base, puis la précision de la taille de la population (c'est-à-dire les chromosomes) et le nombre de cycles (c'est-à-dire de génération), ensuite l'application de l'algorithme génétique sur le RCSF. Enfin effectuer la classification hiérarchique qui consiste à former des groupes de cluster head. Chaque groupe a un super cluster head (le plus proche du Sink) chargé de rassembler les données venant des CHs de ce groupe et les transférer vers le nœud puits. Les résultats de la simulation montrent que l'algorithme proposé minimise la communication à longue distance et révèle des améliorations significatives en consommation énergétique.

Dans l'article (Nimbalkar *et al.*, 2015) les auteurs ont proposé une technique basée sur les GA appelée Trust based Energy Efficient Clustering using GA (TEECGA). Cette approche consiste à choisir le CH qui a l'énergie la plus élevée, et assurer une transmission fiable des données suivant le routage multi-sauts. Le TEECGA a pour but de prolonger la durée de vie du batterie et éventuellement la durée de vie du réseau. Selon les simulations réalisées, les résultats de TEECGA sont meilleurs comparativement à la méthode classique de clustering en considérant les paramètres suivants : l'énergie résiduelle, la distance entre les nœuds de capteurs, le nombre de nœuds capteurs, le nombre de CHs et la confiance.

L'optimisation par essaim de particule (PSO)

Guru et al. ont proposé quatre variantes de PSO, à savoir PSO avec poids d'inertie variant dans le temps (PSO-TVIW), PSO avec accélération variant dans le temps (PSO-TVAC), PSOTVAC hiérarchique (HPSO-TVAC) et PSO avec le mode étudiant superviseur (PSOSSM) dans (Guru *et al.*, 2005). Dans PSO-Time Varying Inertia Weight (PSO-TVIW), le poids d'inertie diminué linéairement à chaque itération. Dans PSO-Time Varying Acceleration Coefficients (PSO-TVAC), le poids d'inertie est constant et les constantes d'accélération varient linéairement à chaque itération. Dans Hierarchical Particle Swarm Optimizer with Time Varying Acceleration Coefficients (HPSOTVAC), la mise à jour des particules n'est pas influencée par la vitesse de l'itération précédente ; mais, la réinitialisation de la vitesse est faite quand la vitesse stagne dans l'espace de recherche. Enfin, dans Particle Swarm Optimisation with Supervisor-Student Model (PSO-SSM), l'équation de mise à jour PSO est modifiée, où le facteur de mouvement est constant. Une analyse comparative détaillée des quatre algorithmes pour un clustering optimal est présentée dans cet article (Guru *et al.*, 2005).

Dans (Latiff *et al.*, 2007), les auteurs présentent un protocole de clustering pour les réseaux de capteurs sans fil utilisant l'algorithme PSO. Ils proposent une nouvelle fonction de coût, dans l'objectif de minimiser simultanément la distance intra-cluster et d'optimiser la consommation d'énergie du réseau. Les performances de ce protocole sont comparées au protocole bien connu basé sur les clusters, LEACH (LowEnergy Adaptive Clustering Hierarchy) et LEACH-C, le dernier étant une version améliorée de LEACH. Les résultats

de la simulation montrent que le clustering basé sur PSO surpasse à la fois LEACH et LEACH-C en termes de durée de vie du réseau et de mise en place.

Le clustering et le routage avec consommation énergétique minimale sont deux problèmes d'optimisation bien connus. Ces problèmes ont été étudiés largement dans la littérature afin de prolonger la durée de vie des réseaux de capteurs sans fil. Dans (Kuila et Jana, 2014) les auteurs ont proposé deux algorithmes basés sur PSO : un algorithme de routage développé avec un schéma de codage de particules efficace et une fonction de fitness multi-objective. Un algorithme de clustering est présenté en considérant la conservation de l'énergie des nœuds par l'équilibrage de charge. Les algorithmes proposés sont comparés aux algorithmes existants pour démontrer leur supériorité en termes de durée de vie du réseau, de consommation d'énergie, de nombre de nœuds morts et de paquets livrés à la station de base.

L'optimisation par essaim de particule (PSO) qui est un algorithme d'optimisation simple et efficace dans la résolution du problème de routage. Dans (EL Ghazi et Ahiod, 2016) on décrit un protocole de routage basé sur PSO qui permet de traiter plusieurs métriques de qualité de services(QoS) et possède une capacité de recherche exceptionnelle. Le résultats de simulation ont montré l'efficacité de cette approche basée PSO comparativement à une autre basée sur ACO.

La recherche du coucou (CS)

Dans cet article (Dhivya *et al.*, 2011b), les auteurs ont mis l'accent sur les principales contraintes des RCSF qui sont la réduction de la consommation d'énergie et à la prolongation de la durée de vie du réseau. Ils ont proposé une technique d'optimisation basée sur la recherche de coucou CS. Dans cette approche, les nœuds avec une faible énergie sont chargés de détecter les événements et les nœuds ayant une grande énergie sont sélectionnés pour agir en tant que CH et communiquer avec la station de base. L'objectif des auteurs est d'équilibrer équitablement la consommation d'énergie entre les nœuds capteurs, en fonction de leur énergie résiduelle et de prolonger la durée de vie du réseau. D'après les résultats, la méthode proposée basée sur CS est bien meilleure en comparaison avec les méthodes traditionnelles.

Le but des auteurs dans (Bhatti et Raina, 2014) est d'améliorer la consommation énergétique, le débit et prolonger la durée de vie des RCSF. Ainsi ils ont proposé une nouvelle approche de routage opportuniste basée sur la recherche du coucou CS et la modification du protocole PEGASIS (Power-Efficient GAthering in Sensor Information Systems). Les résultats de l'approche proposée ont montré une amélioration significative dans la consommation d'énergie et de durée de vie du réseau par rapport aux algorithmes précédents.

Dans (Raj et Sumathi, 2014) les auteurs ont abordé le problème de routage dans les RCSF et ont mis l'accent sur les techniques informatiques bio-inspirées pour obtenir les meilleurs résultats en termes de consommation énergétique. Considérant les techniques bio-inspirées, les auteurs ont proposé un nouvel algorithme appelé EEEMRP (Enhanced Energy Efficient Multipath Routing Protocol) utilisant la recherche du coucou CS. L'algorithme de CS est utilisé pour trouver le chemin optimal afin d'acheminer les données vers la destination d'une manière efficace. EEEMRP est comparé avec le protocole de routage AOMDV suivant des paramètres tels que le débit, le taux de livraison des paquets, le délai et la consommation d'énergie. Les résultats ont montré que la consommation d'énergie et

le débit de l'EEEMRP sont bien meilleurs que ceux de l'AOMDV.

Autres métaheuristiques inspirées de la nature

Une approche intelligente pour la modélisation du routage dans les réseaux de capteurs sans fil a été présentée dans (Zheng et Luo, 2014). Un nouveau protocole de routage basé sur l'algorithme de colonie d'abeilles artificielles (ABC) a été proposé pour résoudre le problème du compromis entre l'énergie et le retard. Dans cette approche proposée, chaque source de nourriture représente un chemin possible entre un nœud source et un nœud destination. La source de nourriture ayant la valeur de nectar la plus élevée est une bonne solution selon l'évaluation de la fonction de fitness. Les résultats de simulation obtenus ont montré que ce protocole basé sur ABC peut réaliser un bon compromis entre le délai de routage et l'énergie.

L'algorithme de recherche d'harmonie Harmony Search (HS) utilisé par Zeng, B. and Dong, Y. dans (Zeng et Dong, 2016) afin de proposer un algorithme amélioré de routage efficace basé sur la recherche d'harmonie (Improved Harmony Search Based Energy-Efficient Routing (IHSBEER)) pour les RCSF. Plusieurs améliorations clés ont été proposées pour résoudre le problème : tout d'abord, le codage de la mémoire d'harmonie a été amélioré en fonction des caractéristiques de routage dans les RCSF. Deuxièmement, l'improvisation d'une nouvelle harmonie a été également améliorée. Troisièmement, une réduction de nombre de paramètres en éliminant le processus d'ajustement. Finalement, une stratégie de recherche locale efficace a été proposée pour améliorer la capacité de recherche, de manière à améliorer la vitesse de convergence et la précision de l'algorithme de routage.

Le choix du bon algorithme d'optimisation peut être crucial pour trouver les meilleures solutions dans un scénario d'optimisation donné. En fait, l'algorithme de colonie de fourmis (ACO) a été appliqué avec succès pour résoudre les problèmes de routage dans les RCSF (Okdem et Karaboga, 2009; Fathima et Sindhanaiselvan, 2013). Quelques exemples d'applications basées sur les fourmis sont :

Sensor-driven Cost-aware ant routing (SC), Flooded Forward ant routing (FF) et Flooded Piggy-backed ant routing (FP) (Zhang *et al.*, 2004) sont trois algorithmes de fourmis proposés pour le routage dans les RCSF, utilisant des fourmis agents qui cherchent le chemin le plus cours entre la source et la destination.

L'Adaptive ant-based Dynamic Routing (ADR) (Lu *et al.*, 2004), c'est algorithme de routage dynamique adaptatif pour les RCSF basé sur des fourmis biologiques simples qui explorent le réseau et apprennent les bonnes routes, en utilisant une nouvelle variante d'apprentissage par renforcement.

Les algorithmes Adaptive Routing (AR) et Improved Adaptive Routing (IAR) présentés dans (GhasemAghaei *et al.*, 2007) répond aux exigences du réseau de capteurs, y compris la consommation d'énergie, le taux de réussite et le délai.

En outre, nous avons proposé dans (El Ghazi *et al.*, 2014) un protocole de routage basé sur l'amélioration de l'algorithme de colonie de fourmis nommé IACOR, qui s'est révélé compétitif avec les approches de littérature.

Tous les algorithmes basés sur l'intelligence évolutive et le swarm nécessitent des paramètres de contrôle, ce qui affecte la performance de l'algorithme. Considérant ce fait, une nouvelle métaheuristique Teaching-Learning-Based Optimization (TLBO) (Rao *et al.*,

2011, 2012) a été développée récemment sans aucun paramètre d'algorithme. Nous avons utilisé cette méthode pour le routage dans les RCSF, ainsi nous avons proposé un nouveau protocole de routage efficace en consommation énergétique, simple en terme d'adaptation, d'implémentation et a d'autres avantages qui le qualifié bien meilleur que plusieurs protocoles basés sur des métaheuristiques inspirées de la nature.

2.5 Conclusion

Dans ce chapitre, nous avons présenté différents problèmes d'optimisation dans les RCSF, considérés comme fondamentaux. Problèmes liés à la couverture, le contrôle de la topologie et l'ordonnancement, qui nécessitent des algorithmes robustes est performants. Ensuite, nous avons mis l'accent sur les méthodes de résolution des problèmes d'optimisation combinatoires : méthodes exactes et méthodes approchées. Enfin, nous avons présenté le problème de routage dans les RCSF et un état de l'art sur les approches basé sur les métaheuristiques inspirées de la nature proposées pour ce problème.

Le but de ce chapitre est de montrer l'efficacité des métaheuristiques inspirées de la nature pour les RCSF et de contribuer dans ces deux domaines en choisissant un problème d'optimisation intéressant dans les RCSF qu'on peut résoudre par une métaheuristique. Parmi les problèmes majeurs des RCSF le problème de routage avec QoS représente le grand défi à relever. Ainsi, Nous avons proposé des approches basées sur des métaheuristiques pour ce problème qu'on va détailler dans les chapitres qui viennent.

Chapitre

3

PROTOCOLES DE ROUTAGE BASE SUR ACO ET PSO

Sommaire

3.1 Introduction

La résolution d'un problème d'optimisation consiste à utiliser un espace de recherche et à tenter de maximiser ou minimiser une fonction objective donnée. Les méthodes de résolution des problèmes d'optimisation sont principalement classées en deux catégories : les méthodes exactes (complètes) et les méthodes approchées (incomplètes). Les métaheuristiques appelées aussi méthodes de recherche globale font partie des méthodes approchées. Ces méthodes visant à résoudre des problèmes d'optimisation difficiles pour lesquels soit les méthodes exactes existantes ne fournissent pas de résultats optimaux soit aucune méthode exacte n'est, à ce jour découverte.

Le problème d'optimisation choisi comme objet d'étude dans ce travail est le problème de routage dans les RCSF qui est considéré un problème NP-difficile. Ce problème a reçu beaucoup d'attention dans la littérature, vue sa complexité, et ses contraintes contradictoires. Plusieurs métaheuristiques inspirées de la nature ont été adaptées à ce problème telles que les algorithmes génétiques, les algorithmes de colonies de fourmis, les algorithmes d'optimisation par essaim de particules ...

Dans ce chapitre, on présente en premier lieu l'amélioration d'une approche de routage basée sur l'algorithme de colonie de fourmis (ACO) avec des simulations et résultats comparatifs qui montrent la supériorité de notre propos vis-à-vis de l'approche originale. En

second lieu, on propose une comparaison de l'approche ACO à une autre approche basée sur PSO, et cela dans des RCSF statiques et mobiles.

3.2 Routage dans les RCSF basé sur ACO

Le problème de routage avec qualité de service, dans les réseaux de capteurs sans fil est un problème d'optimisation combinatoire NP-difficile qui a reçu beaucoup d'attention dans la littérature, vue sa complexité, et ses contraintes contradictoires. Plusieurs métaheuristiques inspirées de la nature ont été adaptées à ce problème, parmi les plus utilisées, les algorithmes de colonies de fourmis. Ainsi, nous citons quelques travaux de recherche basé sur cette métaheuristique :

Singh et al. (Singh *et al.*, 2005) ont proposé un algorithme basé sur les fourmis pour le routage dans les RCSF. Cependant, cet algorithme ne prend pas en compte les principales spécificités de structures des RCSF, y compris les questions liées à l'énergie. Zhang et al. (Zhang *et al.*, 2004) ont proposé également des algorithmes à base de fourmis pour les réseaux de capteurs. Leur étude comprend trois algorithmes de routage nommés SC, FF et FP. Ces derniers sont efficaces avec les paramètres initiaux de phéromone pour avoir un bon démarrage du système. Mais, les algorithmes SC et FF ne sont pas très efficaces dans la latence, alors qu'ils offrent une meilleure efficacité énergétique. De plus, l'algorithme FP, tout en offrant un taux de réussite élevé de la livraison de données, consomme beaucoup plus d'énergie par rapport aux algorithmes FC et FF. L'algorithme EEABR (Energy Efficient Ant Based Routing)(Camilo *et al.*, 2006), basé sur la métaheuristique ACO, est un autre algorithme de fourmi proposé pour maximiser la durée de vie des réseaux de capteurs. L'algorithme utilise une bonne stratégie compte tenu des niveaux d'énergie des nœuds et les longueurs des routes empruntées.

Dans (Chen *et al.*, 2006), les auteurs ont proposé un protocole de routage amélioré basé sur une approche d'optimisation par colonies de fourmis pour les RCSF. Pour surmonter les défauts d'algorithmes de routage classique à base de fourmis, les auteurs ont adopté un nombre d'essais pour éviter l'impasse de l'algorithme ; ils ont ajouté des fourmis de recherche pour réduire le nombre d'essais, et ils ont introduit une stratégie de simulation globale pour mettre à jour la phéromone pour accélérer la convergence. D'autres améliorations exercent également un effet favorable dans cet algorithme. Les résultats de simulation ont montré qu'ils peuvent réduire le coût total de routage dans les réseaux de capteurs et le protocole est plus pratique, efficace et apte à être utilisé à grande échelle.

R. G. Aghaei et al. (GhasemAghaei *et al.*, 2007) ont présenté un nouveau système basé sur l'algorithme de fourmis pour le routage dans les RCSF. Ils ont proposé deux nouveaux algorithmes, l'algorithme de routage adaptatif AR (Adaptive Routing) et son amélioration IRA (Improved Adaptive Routing). Les algorithmes proposés sont en mesure de réagir, s'adapter et faire face à des changements dans le réseau. Les auteurs ont utilisé les caractéristiques d'apprentissage par renforcement pour les algorithmes. L'adoption de AR et IAR pour le routage dans un RCSF entraîne une faible consommation d'énergie, une haute efficacité énergétique, une faible latence et un taux de réussite élevé.

Un algorithme appelé BAOA (Broadcasting based on Ant colony system Optimization Algorithm) est développé par Ge *et al.* (2007). Les auteurs présentent la procédure de BAOA y compris six étapes lors de la recherche du chemin de diffusion optimisé dans les réseaux de capteurs et la conception BAOA avec UML. Ils simulent aussi BAOA avec le

langage C++ et les résultats montrent que la durée de vie de l'algorithme BAOA est plus longue et la consommation d'énergie totale est inférieure à celle de BIP (Wieselthier *et al.*, 2000).

Dans (Salehpour *et al.*, 2008), les auteurs proposent un algorithme de routage efficace à base de clusters pour les RCSF à grande échelle. La technique utilise deux niveaux de routage. Dans le premier niveau (intra-cluster), les membres du cluster envoient les données directement à leur CH. Dans le deuxième niveau (inter-cluster), les CHs utilisent l'algorithme d'optimisation par colonie de fourmis (ACO), qui est un paradigme inspiré de la biologie dédié à l'approche d'optimisation, pour trouver une route vers la SB. Comme seuls les CHs participent à l'opération de routage inter-cluster, la méthode peut fournir un bon fonctionnement plus efficace. Le retard de l'algorithme est minimisé en utilisant l'algorithme d'optimisation par colonie de fourmis avec clustering.

Guan *et al.* (2009) introduisent un nouvel algorithme de routage basé sur le système à colonie de fourmis (Ant Colony System). L'objectif de ce nouvel algorithme est de résoudre le problème d'énergie et contrôler la congestion sur le processus de routage dans les RCSF. Cet algorithme est capable d'atteindre un meilleur équilibre de la charge et prolonge la durée de vie du réseau. Dans cet algorithme, les auteurs combinent la phéromone libérée par les colonies de fourmis et l'énergie résiduelle. Cette combinaison constitue le facteur de contrôle de l'algorithme. En outre, les auteurs introduisent également le mécanisme de la concurrence entre les colonies de fourmis pour éviter la convergence simple. De cette façon, le nouvel algorithme contrôle efficacement la congestion de trafic du réseau et équilibre la consommation d'énergie pour les réseaux de capteurs.

Une façon de réduire la consommation d'énergie dans les réseaux de capteurs et donc de prolonger la durée de vie de ces réseaux est d'utiliser des algorithmes de clustering adaptatifs.Dans (Ziyadi *et al.*, 2009), les auteurs utilisent l'algorithme ACO-C (Ant Colony Optimization for Clustering) pour proposer un nouveau protocole de clustering efficace en terme d'énergie pour les réseaux de capteurs. En utilisant les fonctions de coûts appropriés, implémentés à la SB, leur algorithme minimise et distribue uniformément le coût des transmissions à longues distances et les données agrégées entre tous les nœuds.

Ru et Xu (2010) présentent un algorithme optimal basé sur les colonies de fourmis. Cet algorithme, nommé TORA, divise le réseau en différents types de régions. L'énergie consommée de chaque capteur peut être estimée à l'avance. La nouvelle conception de construction de facteur heuristique et la règle de mise à jour de phéromone peut doter les fourmis artificielles la capacité de détecter de manière adaptative l'état local de l'énergie dans le réseau et intelligemment approcher le modèle théorique dans le processus de construction de routage. Enfin, l'arbre de routage optimal proposé peut améliorer l'efficacité énergétique et la robustesse du réseau dans la collecte de données.

Ces études ont prouvé la performance des algorithmes de colonies de fourmis (Saleem *et al.*, 2010) dans la résolution de problème de routage avec QoS dans les RCSF. Nous citons entre autres l'article "Routing in Wireless Sensor Networks Using an Ant Colony Optimization (ACO) Router Chip" (Okdem et Karaboga, 2009) qui représente notre point de départ. Dans la suite de ce chapitre, Nous allons proposer une amélioration de l'approche ACO (Okdem et Karaboga, 2009), afin de concevoir une nouvelle plus performante.

3.3 Approche proposée basée sur ACO

L'approche présentée dans (Okdem et Karaboga, 2009) a montré les performances des algorithmes de colonies de fourmis pour le routage dans les RCSF cependant, les métaheuristiques restent un domaine relatif qui peut toujours donner plus, ce qui a fait de cette approche un point de départ pour nous. Dans cette section, nous allons présenter un protocole de routage pour les RCSF utilisant l'algorithme ACO afin d'optimiser les chemins de routage, en fournissant une méthode de transmission multi-path efficace et fiable même en cas de panne. La procédure de notre approche consiste à la détection d'événement, traitement des données (information sur l'évènement), construction des chemins de routage (de la source vers le sink), l'acquittement des paquets reçus et la retransmission des paquets non reçus.

3.3.1 Détection d'événements et traitement des données

Plusieurs nœuds deviennent sources en recevant des informations sur un évènement qui a eu lieu à proximité. Ces informations sont diffusées vers le sink à l'aide des nœuds voisins qui se comportent comme des répéteurs. Les données brutes sont les données relatives à un évènement qui contiennent des informations telles que l'identification de nœud source, l'identification de l'évènement, le temps et des données sur l'évènement. Le nœud source divise ces données brutes en N parties comme le montre la figure 3.1, où la taille de chaque partie est choisie selon la taille du tampon. Le nombre N représente aussi le nombre de fourmis qui vont transmettre les parties.

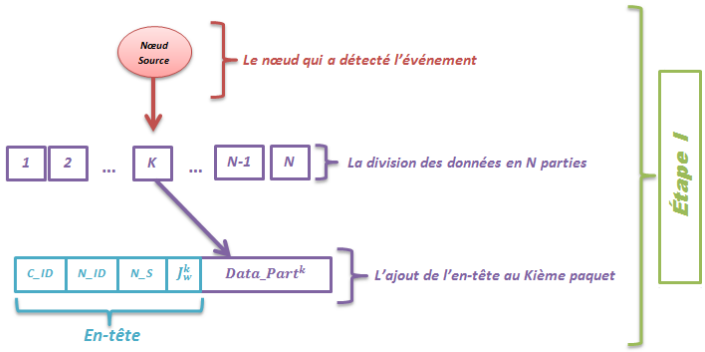


Figure 3.1 – Traitement des données brutes

Après avoir partagé les données brutes en parties, chaque partie est associée à des paramètres de routage pour construire un paquet prêt à transférer. Ces paramètres sont le code d'identification (donnée, erreur ou acquittement) C_ID, l'identifiant du nœud suivant (au quel le paquet est transféré) N_{ID}, numéro de paquet (représente aussi l'agent fourmi k), S_N le numéro de séquence, $J_w^k(t)$ qui contient le nombre de nœuds visités et la kème partie comme indiqué dans la figure 3.1. Les quatre premiers champs forment l'en-tête qui aide la station de base à combiner tous les paquets reçus pour reconstruire les données brutes.

3.3.2 Construction des chemins et envoie des données au sink

Quand un nœud source s a des informations à transférer au sink, il envoie des fourmis qui essaient de trouver des chemins avec un coût minimal. Chaque fourmi se déplace vers les nœuds répéteurs voisins ri qui sont choisis à base de la règle probabiliste, pour atteindre la destination finale d (Sink).

Dans le papier étudié (Okdem et Karaboga, 2009) la règle de décision probabiliste est définie comme suit :

$$P_k(s,r) = \begin{cases} \frac{[\tau(s,r)]^{\alpha}.[\eta(s,r)]^{\beta}}{\sum_{r \in R_s}[\tau(s,r)]^{\alpha}.[\eta(s,r)]^{\beta}} & \text{si k} \notin tabu^r \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

Où $\tau(s,r)$ est la valeur de phéromone, $\eta(s,r)$ est la valeur de l'heuristique liée à l'énergie, R_s est l'ensemble des nœuds voisins récepteurs. Pour le nœud r, $tabu^r$ est la liste des identités des paquets de données reçus auparavant. α et β sont respectivement, le paramètre qui contrôle le taux relatif de la trace de phéromone et l'information heuristique. Les traces de phéromone sont reliées aux liens. Chaque lien $l(s,r)$ a une valeur de piste $\tau(s,r) \in [0,1]$. Puisque la destination d est une station de base stable, le dernier nœud des chemins des fourmis est le même. L'information heuristique du nœud r est exprimée par l'équation 3.2

$$\eta(s,r) = \frac{(I - e_r)^{-1}}{\sum_{n \in R_s}(I - e_n)^{-1}} \tag{3.2}$$

Où I est l'énergie initiale, et e_r est le niveau d'énergie courant du nœud récepteur r. Cela permet de décider en fonction des niveaux d'énergie des nœuds voisins, ce qui signifie que le nœud ayant une énergie faible, a moins de chances d'être choisi. Les nœuds informent leurs voisins sur leurs niveaux d'énergie lorsqu'ils perçoivent un changement de leurs niveaux d'énergie. Le changement est plus probable après une participation active de l'écoute ou la transmission. Ainsi, ils peuvent sentir leur situation énergétique immédiatement après toute participation.

Dans l'ACO traditionnel, une variable spéciale M_k est stockée dans la mémoire de la fourmi pour retenir les places qu'elle a visité (les nœuds). Alors que dans l'équation 3.1, on stocke les identités des fourmis (un numéro de séquence) dans la mémoire du nœud visité, donc il n'est pas nécessaire d'utiliser les listes M_k dans les paquets durant la transmission, afin de diminuer la taille des données à transmettre et économiser l'énergie.

La deuxième étape illustrée par la figure 3.2 présente le cœur du protocole, où le nœud ayant un paquet à transmettre calcule les probabilités des nœuds voisins pour décider quel nœud sera la prochaine destination du paquet k. Le nœud choisi refait la même chose ainsi de suite jusqu'à trouver le Sink. L'approche dans (Okdem et Karaboga, 2009) a donné de très bon résultats relativement au protocole de routage Energy Efficient Ant- Based Routing (EEABR) proposé par T. Camilo et al. (Camilo *et al.*, 2006), mais ces résultats peuvent être améliorés, en ajoutant plus de précision aux choix. Particulièrement quand les probabilités sont égales, un nœud choisit au hasard le nœud suivant, donc il peut faire un mauvais choix et perd les données dans une zone non couverte, ou choisir un long

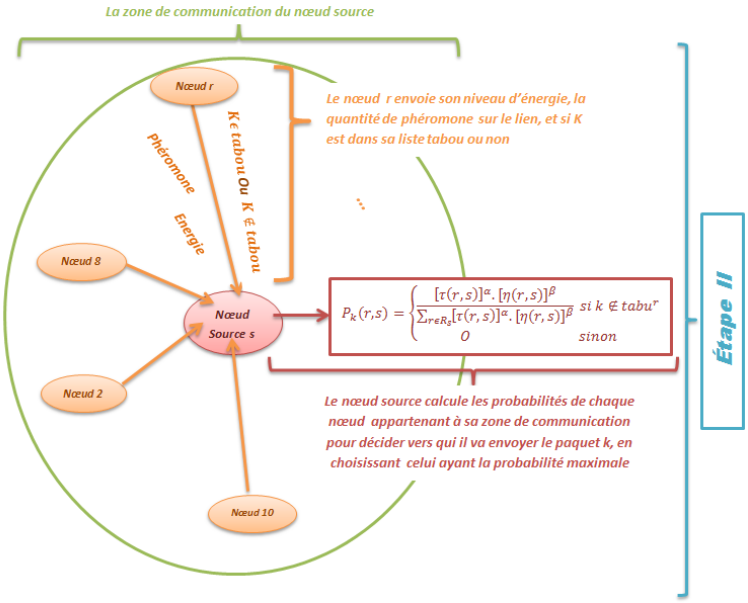


Figure 3.2 – Choix du nœud destination

chemin vers le sink. Par conséquent, de nombreux nœuds perdent l'énergie (juste parce que le choix était mauvais), le délai de livraison augmente et la durée de vie diminue. Pour réduire le nombre de mauvais choix, nous avons amélioré cette approche (Okdem et Karaboga, 2009), toute en gardant la même modélisation, nous avons rendu la règle de décision plus précise en ajoutant une nouvelle valeur heuristique δ (Voir figure 3.3). Ainsi, la nouvelle règle de décision probabiliste est la suivante (**equation 3.3**) :

$$P_k(s,r) = \begin{cases} \frac{[\tau(s,r)]^{\alpha}.[\eta(s,r)]^{\beta}.[\delta(s,r)]^{\gamma}}{\sum_{r \in R_s}[\tau(s,r)]^{\alpha}.[\eta(s,r)]^{\beta}.[\delta(s,r)]^{\gamma}} & \text{si k} \notin tabu^r \\ 0 & \text{sinon} \end{cases} \tag{3.3}$$

Le protocole amélioré se diffère principalement de celui de base dans l'étape deux, dans laquelle nous avons ajouté un nouveau paramètre, un indice de plus qui va aider les fourmis à choisir le chemin optimal.

Le paramètre δ (**equation 3.4**) est une deuxième valeur heuristique qui est reliée à la portée R_c des nœuds. Un nœud en communicant, l'information qui concerne l'existence du paquet k dans sa liste tabou, au nœud source ou au nœud ayant un paquet à transmettre, il envoie aussi une information sur le Sink.

$$\delta(s,r) = \begin{cases} \frac{E_r}{\sum_{i \in R_r} E_i} & \text{si Sink} \in R_c \\ v & \text{sinon} \end{cases} \tag{3.4}$$

Où E_r est l'énergie résiduelle du nœud r et v est une constante qui dépend de l'en-

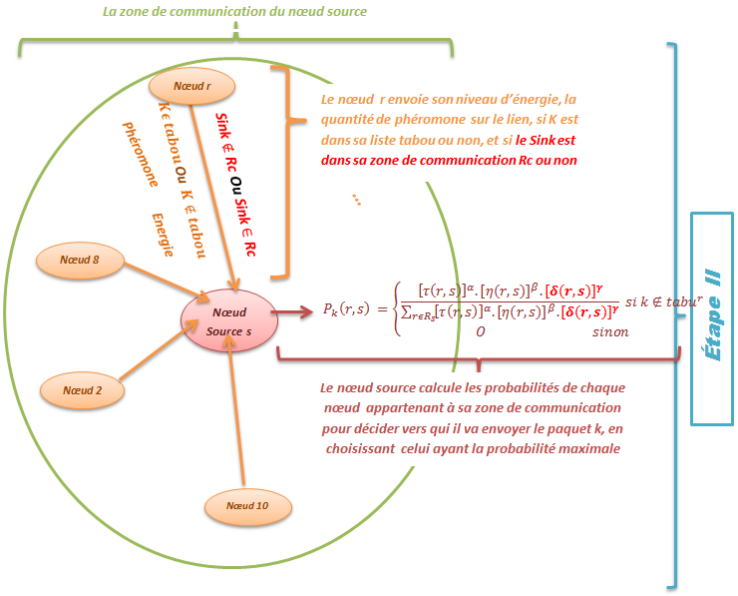


Figure 3.3 – Choix du prochain saut

vironnement de simulation. Un nœud r ayant le sink dans sa zone de couverture, doit informer les nœuds voisins sur ce détail, pour avoir plus de chances d'être choisis (parce qu'en choisissant ce nœud le paquet atteint le sink certainement). Lorsque le sink n'est pas dans le champ de r, seule l'énergie et phéromone sont pris en compte dans la règle probabiliste.

Cette valeur heuristique ajoutée nous a permis d'améliorer la métaheuristique ACO adaptée au protocole de routage, et par conséquent trouver des résultats meilleurs comparés à ceux obtenus par le protocole d'origine, principalement dans la consommation d'énergie, la durée de vie des RCSF et la fiabilité de transmission.

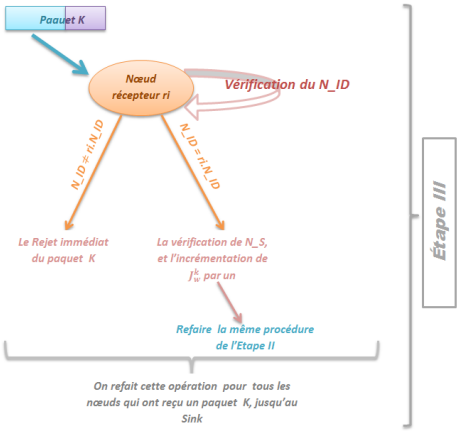


Figure 3.4 – Réception d'un paquet.

Dans la figure 3.4, on présente l'ensemble des traitements, effectués par un nœud répéteur quand il reçoit un paquet.

3.3.3 Acquittements et retransmission des paquets perdus

Le sink reçoit les données en parties de plusieurs chemins, pour assurer la fiabilité du transfert en cas de panne des routes principales. Afin d'éviter toute perte de paquet dans ces chemins, le sink diffuse des signaux d'acquittement, dés que la source et les nœuds voisins terminent leurs émissions. Le sink décide la valeur de phéromone à ajouter en évaluant J_w^k dans l'équation 3.5. Ensuite, il forme un signal d'acquittement comme le montre la figure 3.5, puis il l'envoie vers la source, en utilisant le chemin qui a été utilisé par le paquet associé. Cette opération est illustrée dans la figure 3.6.

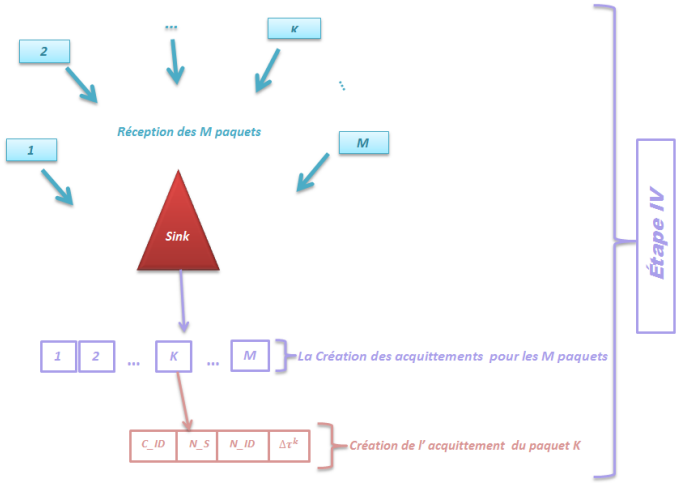


Figure 3.5 – Construction des acquittements.

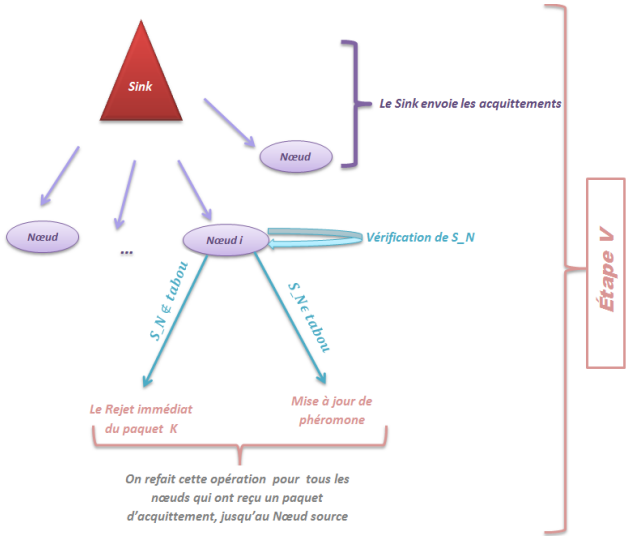


Figure 3.6 – Transmission des acquittements.

Dans la figure 3.5, le sink après la réception des paquets de données, il construit les paquets d'acquittements associés et dans la figure 3.6, on montre la transmission de ces acquittements du sink vers les nœuds voisins, ainsi que les traitements effectués au niveau d'un nœud récepteur.

$$\triangle\tau^k(t) = \frac{1}{J_w^k(t)} \tag{3.5}$$

Après que toutes les fourmis complètent leur tournée, chaque fourmi k dépose une quantité de phéromone $\triangle\tau^k(t)$ définie par l'équation 3.5, où $J_w^k(t)$ est la longueur de la tournée $w^k(t)$ faite par la fourmi k à l'itération t.

$$\tau(s,r)(t) \leftarrow \tau(s,r)(t-1) + \triangle\tau^k(s,r)(t) \qquad \forall l(s,r) \in w^k(t) \quad k = 1,\ldots,m \tag{3.6}$$

La quantité de phéromone dans chaque lien $(l(s,r))$ entre les nœuds est donnée par l'équation 3.6. Dans les RCSF, $J_w^k(t)$ représente le nombre total de nœuds visités par la fourmi k dans le tour w à l'itération t.

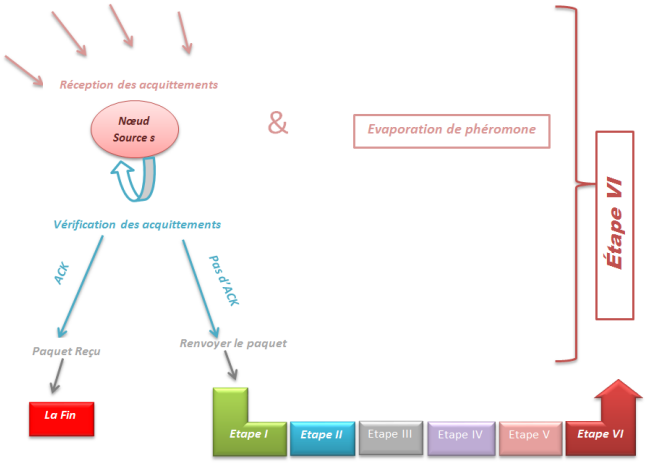


Figure 3.7 – Retransmission des paquets non-acquittés

L'étape finale est présentée par la figure 3.7 qui montre l'évaporation de phéromone et la retransmission des paquets non-acquittés.

Chaque nœud stocke les quantités de phéromone déposées sur les chemins vers ses nœuds voisins, dans sa mémoire. Après chaque tournée une quantité de phéromone $\triangle\tau^k$ est ajoutée au chemin visité par la fourmi k (la même quantité ajoutée à chaque $l(s,r)$ de ce chemin) cela en envoyant la fourmi k qui transfère le signal d'acquittement du paquet associé à partir du Sink vers le nœud source le long du même chemin. Pour contrôler la quantité de phéromone qui augmente en fonction de la longueur de tournée $J_w^k(t)$, l'évaporation du phéromone se fait aussi après chaque tour. Selon l'équation 3.7, on utilise un coefficient de contrôle $\rho \in [0,1]$ pour déterminer le taux d'évaporation de chaque tour

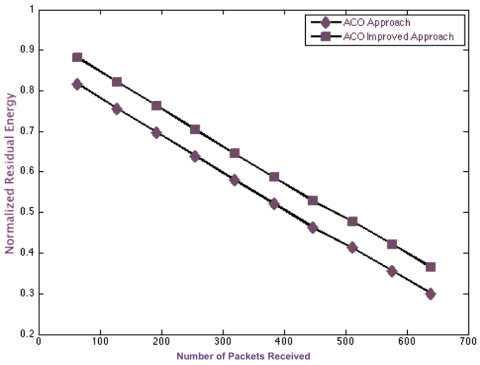
(Dorigo et Di Caro, 1999).

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t-1) \tag{3.7}$$

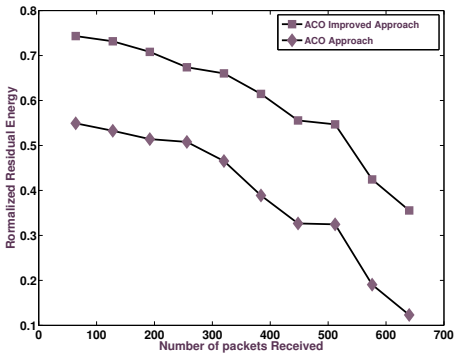
Dans les simulations les valeurs respectives sont : 1 pour α, 5 pour β et 0,5 pour ρ.

3.3.4 Résultats et comparaisons des approches ACO

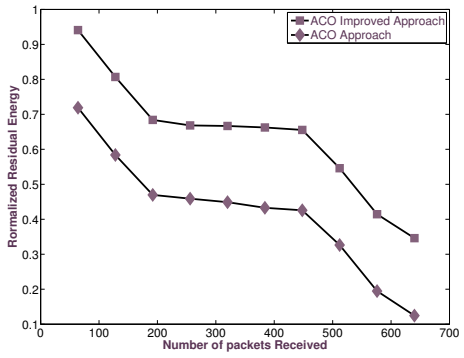
Dans cette section nous allons présenter les résultats de la simulation développée sous MATLAB, en utilisant un modèle de capteurs basé sur "First Order Radio Model" de Heinzelman et al. (Heinzelman *et al.*, 2000).



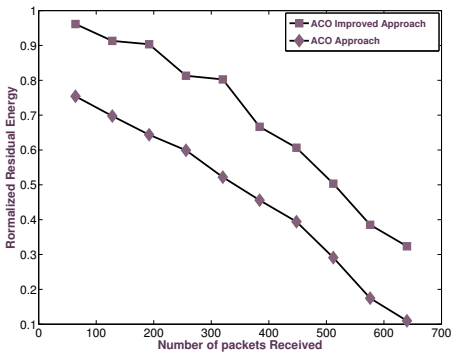
(a) 20 *noeuds*, *densité* 222x10^{-6}*noeuds*/m^2



(b) 40 *noeuds*, *densité* 160x10^{-6}*noeuds*/m^2



(c) 60 *noeuds*, *densité* 167x10^{-6}*noeuds*/m^2



(d) 80 *noeuds*, *densité* 222x10^{-6}*noeuds*/m^2

Figure 3.8 – Les résultats de différents RCSF

L'implémentation du protocole de routage ACO est basée sur la simulation de plusieurs RSCFs, de densités différentes, afin d'obtenir des résultats illustratifs divers dans la figure 3.8.

On déploie aléatoirement, un nombre de capteurs qui varie selon la surface de la zone de couverture : sur une surface de 200 x 200 m^2, on diffuse 10 nœuds, sur 300 x 300 m^2,

on diffuse 20 nœuds, sur 400 x 400 m^2, on jette 30 nœuds, sur 500 x 500m^2, 40 nœuds sont dispersés et finalement une surface de 600 x 600m^2 est surveillée, par plusieurs types des RCSF, composés respectivement de 50, 60, 70, 80 et 100 nœuds capteurs.

Le but de ce déploiement est de surveiller un phénomène statique, et acheminer les données vers le sink. Les emplacements de ce phénomène et de nœud Sink sont inconnus. Dans les mêmes conditions relatives au protocole d'origine nous avons conduit la simulation du protocole amélioré, et ce dans le but de comparer les résultats et confirmer l'efficacité de la nouvelle approche.

En utilisant des RCSF différents, dans les simulations, on a pu prouver que le protocole ACO amélioré , est bien meilleur vis-à-vis du protocole ACO original. L'approche proposée, donne des résultats meilleurs, comme illustre la figure 3.10, par la maximisation de la durée de vie du réseau et la minimisation de la consommation d'énergie, particulièrement pour les réseaux ayant une densité élevée.

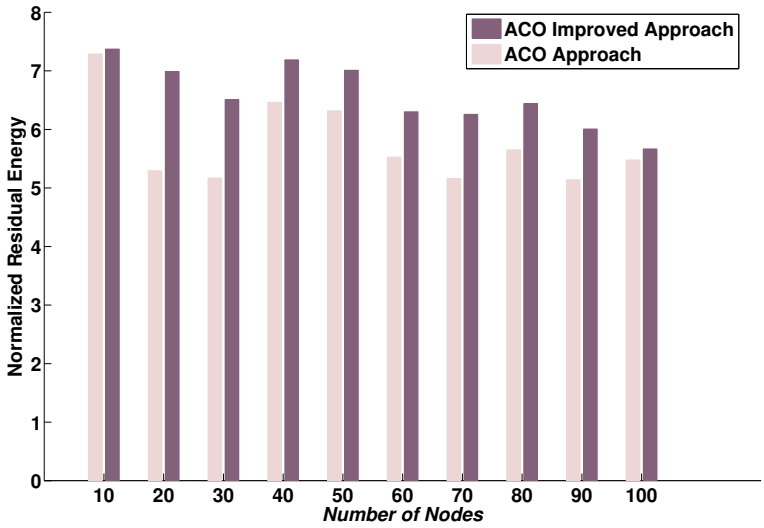


Figure 3.9 – Énergies résiduelles de différents RCSF, après la réception de 256 paquets

La figure 3.9 montre l'énergie résiduelle moyenne normalisée, après la réception de 256 paquets par le Sink, pour différents réseaux de capteurs ayant des nombres divers de nœuds.

Pour conclure cette partie, les métaheuristiques sont des méthodes d'optimisation efficaces et performantes. Ayant la possibilité de les adapter aux problèmes d'optimisation complexes et NP-difficiles et la possibilité de les améliorer on peut toujours trouver des résultats plus convaincants, et c'est le cas pour notre protocole de routage. La modification effectuée sur l'algorithme de colonies de fourmis, nous a permis d'atteindre des résultats meilleurs comparativement à ceux obtenus par le protocole de base. Les performances de notre protocole ont été prouvées à l'aide de simulation sous MATLAB.

3.4 ACO et PSO pour le routage dans les RCSF

Les réseaux de capteurs sans fil sont généralement composés de nœuds statiques déployés dans une zone de surveillance. Cependant, en raison des changements de conditions et de l'environnement hostile, un RCSF immobile pourrait faire face à plusieurs problèmes tels que, le problème de couverture, le problème de trous, la connectivité de capteurs, etc. Ainsi, l'introduction de la mobilité à certains ou à tous les nœuds D'un RCSF, peut améliorer ses capacités et sa flexibilité. Cette mobilité suit un modèle conçu décrivant le mouvement des capteurs, leur évolution, leur vitesse et leur accélération au fil du temps. Il peut s'agir d'un modèle aléatoire ou contrôlé. Cette partie, présente une comparaison de notre approche ACO détaillée dans la partie précédente à une autre approche basée sur la métaheuristique inspirée de la nature, optimisation par essaim de particules (Particle Swarm Optimisation (PSO)). La comparaison inclut l'évaluation de l'impact de mobilité sur ces deux protocoles basés sur ACO et PSO.

Après l'efficacité dont la métaheuristique ACO a fait preuve pour le problème de routage dans les réseaux de capteurs sans fil, les chercheurs ont orienté leurs attention vers l'optimisation par essaim de particules. Ainsi on a proposé une approche basé sur PSO pour le routage dans les RCSF.

3.4.1 Particle Swarm Optimization

L'optimisation par essaim de particules (PSO) développée par Eberhart et Kennedy en 1995 (Eberhart *et al.*, 1995)est une technique d'optimisation basée sur une population de solutions ; il a inspiré du comportement social des interactions dans l'essaim d'oiseaux ou les bancs de poissons qui tentent d'atteindre une destination inconnue. PSO est une collection de particules ou (agents) dispersés dans un espace à N dimensions et qui cherchent la meilleure (optimum) solution en mettant à jour les générations. Par rapport à d'autres heuristiques, PSO présente quelques particularités intéressantes, comme l'efficacité est due à la collaboration plutôt qu'à la compétition. Chaque particule i est caractérisée par les informations suivantes :

- Sa position xi qui représente la solution au problème.
- Sa vitesse vi
- La meilleure position personnelle $pbest$
- La meilleure position globale $gbest$ trouvée par les particules voisines.

Le degré d'optimalité est mesuré par une fonction fitness (aptitude) définie par l'utilisateur. Chaque particule utilise donc l'information dont il dispose par rapport à sa propre histoire et à celle de ses voisins afin de déterminer la direction qu'il doit prendre ainsi que sa vitesse de déplacement. Les vitesses et les positions sont mises à jour selon les deux équations suivantes :

$$v_{i+1} = \omega v_i + \eta_1 rand()(Pbesti - x_i) + \eta_2 rand()(Gbest - x_i) \tag{3.8}$$

$$x_{i+1} = x_i + v_{i+1} \tag{3.9}$$

Où v_i est la vitesse de la particule i. x_i est la position actuelle de particules i. $Pbesti$ est la meilleur position de particules i et $Gbest$ est la meilleure position du groupe.

Le mouvement de chaque particule est influencée par sa meilleur position locale et globale dans l'espace de recherche. Cela devrait déplacer l'essaim vers les meilleures solutions. L'algorithme 3.1 décrit la procédure PSO.

Algorithme 3.1 Optimisation par Essaim de Particules

Une population de particules qui ont des positions et des vitesses aléatoires de D dimensions dans l'espace de recherche
tantque Condition d'arrêt non atteinte **faire**
  pour chaque particule i **faire**
    Adapter la vitesse de particule utilisant l'équation 3.8
    Mettre à jour la position de particule utilisant l'équation 3.9
    Évaluer par la fonction fitness
    si $f(x_i) < f(Pbesti)$ **alors**
      $Pbesti \leftarrow x_i$
    finsi
    si $f(x_i) < f(Gbesti)$ **alors**
      $Gbesti \leftarrow x_i$
    finsi
  fin pour
fin tantque

Le PSO est un outil d'optimisation basé sur une population d'individus. Il a été prouvé que PSO peut donner de bons résultats pour les problèmes d'optimisation combinatoires discrets et continus avec une fonction mono-objective ou multi-objectives.

3.4.2 Routage dans les RCSF et PSO

Le routage c'est l'exploration de tous les chemins disponible entre le nœud source et le nœud de destination. L'algorithme de routage basé sur PSO à pour principe l'exploration stochastique pour découvrir de nouveau chemin. On obtient cette propriété stochastique en utilisant des tables de routage qu'attribuent des paramètres de QoS des nœuds. D'après l'évaluation des mesures de QoS synthétique, les paquets de données suivent le saut avec la plus haute qualité de synthèse des évaluations du rendement. L'évaluation des métriques de QoS synthétique est identique à la fonction objective du routage.

Pour rechercher le chemin optimal entre le nœud source et le nœud destination, l'initialisation est une étape nécessaire, comme décrit dans (Zhang et Xu, 2006).

Les chemins initiaux sont formés en employant deux agents : forward et backward agent (Zhang et Xu, 2006). L'agent forward se déplace pour déterminer le nœud suivant sur le chemin, crée et maintient la table de routage. En même temps, il recueille de nombreuses informations sur les nœuds dans le chemin. L'agent backward est créé par le nœud de destination et se déplace de la destination vers le nœud source.

Dans les RCSF, la recherche du meilleur chemin entre la source et la destination, consiste à chercher la cible optimale dans tout l'espace de solution, qui est similaire au recherche de solution basé sur la coopération des essaim de particules. L'initialisation d'essaim de particules (chemins) obtenue est exprimée par
$\{x_1, \ldots, x_i, \ldots, x_m\}$.

Chaque chemin x_i est un potentiel meilleur chemin de routage, Noté *Pbesti* quand x_i est modifié. *Gbest* est le meilleur chemin global dans tous les chemins. Supposons que x_i se compose de k nœuds $\{n_s, \ldots, n_k \ldots, n_{sink}\}$. Afin de trouver le meilleur chemin, nous mettons en œuvre l'algorithme PSO dont l'équation 3.8 a été remplacée par certaines permutations entre *Pbesti* et x_i, *Gbesti* et x_i comme décrit dans (Zhang et Xu, 2006).

Comme l'équation 3.8 ne peut être appliquée telle qu'elle est, les auteurs dans (Zhang et Xu, 2006) ont proposé une méthode pour modifier xi. Certains nœuds dans xi sont aussi dans *Pbesti*. Il est raisonnable de remplacer un nœud $n_s$dans le chemin xi par un nœud np_s sélectionné de *Pbesti*, ce qui va rendre xi plus proche de *Pbesti*. Le nœud np_s diffuse un agent forward, si n_{s-x} et n_{s+y} $(x, y >= 1)$ reçoivent l'agent direct, le chemin brisé est réparé et le nouveau chemin xi est recréé. Sinon, nous devrions sélectionner un autre nœud dans *Pbesti* au lieu de np_s. De la même méthode on redéfini la formule *Gbest* de l'équation 3.8.

L'algorithme de routage proposé utilisant PSO est exprimé comme suit :

Algorithme 3.2 Routage basé sur PSO

Le nœud source initialise la population en utilisant les deux agents : forward et backward.

tantque Nbr_iteration < nombre d'itérations maximal **faire**
  pour Chaque particule xi **faire**
    Évaluer xi en utilisant la fonction objective
    si $f(x_i) < f(Pbesti)$ **alors**
      $Pbesti \leftarrow x_i$
    finsi
    si $f(x_i) < f(Gbesti)$ **alors**
      $Gbesti \leftarrow x_i$
    finsi
    Améliorer xi suivant la redéfinition des equations 3.8 et 3.9 (Zhang et Xu, 2006)
  fin pour
  Nbr_iteration ++
fin tantque
Garder le chemin final.

Un RCSF est présenté par un graphe pondéré $G(V, E)$, V est un ensemble de nœuds dans le RCSF. S'il y a m nœuds en réseau, $V = \{n_1, n_2 n_3 \ldots n_m\}$. on note $l(i, j)$ deux voisins i et j, $P(a, b)$ le chemin entre a et b. E est un ensemble de chemin $E = P(a, b)\{a \in V, b \in V$ Il existe un chemin entre a et b $\}$. Les métriques de qualité de services (QoS) pour $l(i, j)$ sont le délai $D(i, j)$, l'énergie résiduelle $E(i, j)$ et les métrique de QoS synthétiques telles que le rapport de bruit de signal (SNR) et le rapport d'efficacité de bande passante (BWER), exprimé en $SSNR(i, j)$, $SBWER(i, j)$. Lorsqu'un chemin $P(a, b)$ est formé, les mesures QoS pour $P(a, b)$ sont exprimées en équations suivantes :

$$D_{p(a,b)} = \sum_{l(i,j)\in P(a,b)} D(i,j) \rightarrow min \tag{3.10}$$

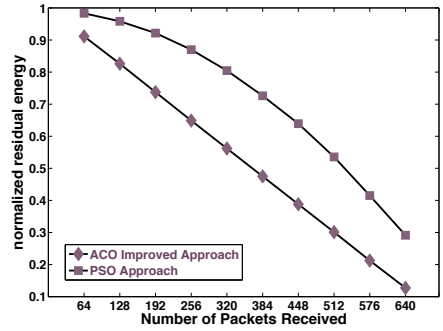
$$E_{p(a,b)} = \sum_{l(i,j)\in P(a,b)} E(i,j) \rightarrow max \tag{3.11}$$

$$S_{p(a,b)} = \sum_{l(i,j)\in P(a,b)} S_{SNR}(i,j) + S_{BWER}(i,j) + \cdots \rightarrow max \tag{3.12}$$

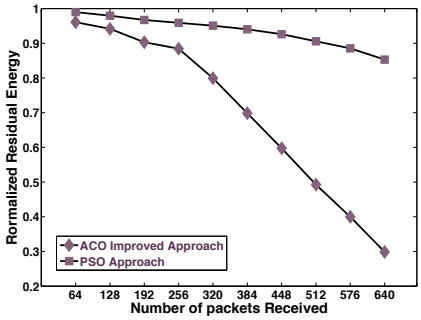
3.4.3 ACO et PSO pour le routage dans les RCSF statiques

En effectuant de nombreuses simulations des deux approches PSO et ACO, en utilisant Matlab et les mêmes conditions d'expérimentation, nous avons utilisé le modèle de capteurs basé sur "First Order Radio Model" de Heinzelman et al. (Heinzelman *et al.*, 2000).

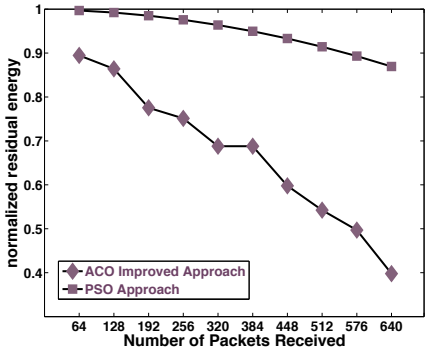
Dans l'objectif de comparer l'approche basée sur PSO et celle basée sur ACO, en utilisant un déploiement aléatoire de différents nombres de nœuds dans des zones de surface variées, nous simulons la transmission de données d'un nœud source à une station de base connue. La meilleure approche apparaît dans les résultats présentés dans la Figure 3.10.



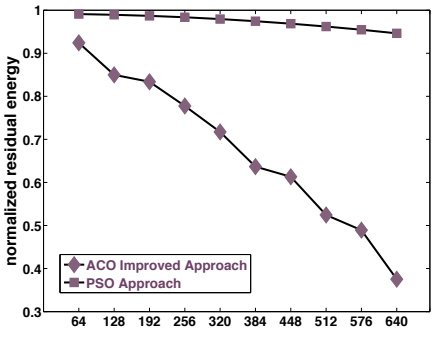
(a) 10 *noeuds*, *densité is* 225x10^{-6}*noeuds*/m^2



(b) 50 *noeuds*, *densité is* 222x10^{-6}*noeuds*/m^2



(c) 30 *noeuds*, *densité is* 187.5x10^{-6}*noeuds*/m^2



(d) 70 *noeuds*, *densité is* 222x10^{-6}*noeuds*/m^2

Figure 3.10 – Les résultats de différents RCSF

La figure 3.10 montre que PSO est supérieur à ACO en considérant l'énergie résiduelle. Ainsi, la consommation d'énergie est minimisée et la durée de vie de WSN est maximisée.

Afin de confirmer l'efficacité de notre proposition, nous avons simulé la transmission de 256 paquets dans différentes zones de couverture où nous avons déployé au hasard plusieurs nœuds (de 10 à 100 nœuds). Les résultats indiqués dans la figure 3.11 représentent l'énergie résiduelle.

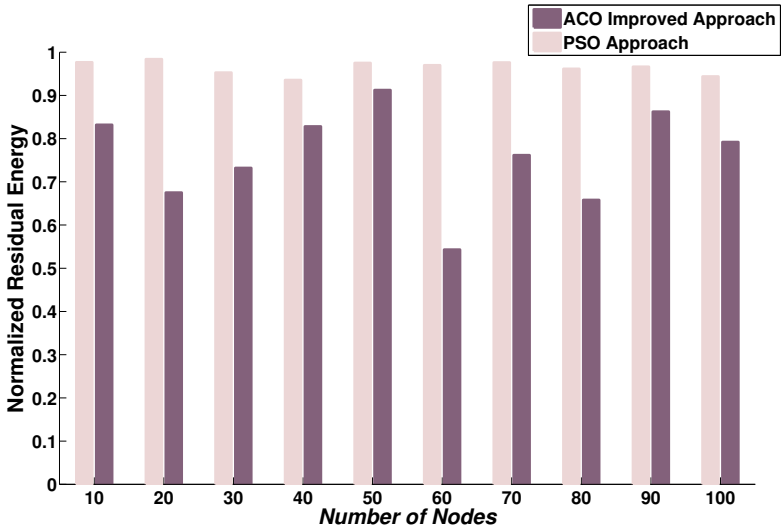


Figure 3.11 – L'énergie Résiduelle pour différents RCSF

Les résultats trouvés prouvent l'efficacité de PSO pour les problèmes de routage, en termes de consommation d'énergie et de durée de vie du réseau de capteurs sans fil.

3.4.4 ACO et PSO pour le routage dans les RCSF mobiles

Afin d'évaluer les performances d'un protocole, ce dernier doit être testé sous un modèle de mobilité des nœuds. Un tel modèle doit être capable d'imiter le mouvement des nœuds tel que souhaité dans un certain scénario. D'où la nécessité d'utiliser des modèles synthétiques qui ont pour rôle d'exprimer le comportement des nœuds sans l'usage de traces réelles. Plusieurs modèles sont définis dans la littérature afin de simuler le comportement des nœuds. Le modèle le plus utilisé pour l'évaluation des protocole est Random Waypoint Mobility. Nous allons donc étudier ce modèle de mobilité afin d'avoir une vue sur son fonctionnent et évaluer les deux protocoles de routage basés sur ACO et PSO.

The Random Waypoint Mobility Model

Ce modèle était d'abord utilisé par Johnson et Maltz (Santi, 2012) dans l'évaluation du protocole de routage DSR et était ensuite raffiné par les mêmes auteurs. Le Random Waypoint est pour modéliser tous les scénarios dans lesquelles, les nœuds se déplacent vers une destination, prennent un repos en arrivant, avant de se déplacer vers une autre

destination et ainsi de suite. Dans ce modèle chaque nœud choisit aléatoirement, comme destination un point de coordonnées (x, y) dans la surface de simulation, et une vitesse entre 0 et $Vmax$. Le nœud voyage vers la destination choisie avec la vitesse choisie. A l'arrivée, le nœud prend un temps de repos avant de choisir à nouveau une nouvelle destination et une nouvelle vitesse pour répéter le même processus. Des études ont été faites sur ce modèle puisqu'il est le modèle le plus utilisé dans les simulations dû à la facilité de son implémentation. Certaines études ont traité l'initialisation de ce modèle et le temps de convergence des simulations dans le cas où les nœuds commencent par prendre un temps de repos. La figure 3.12 présente le déplacement d'un nœud utilisant le Random Waypoint.

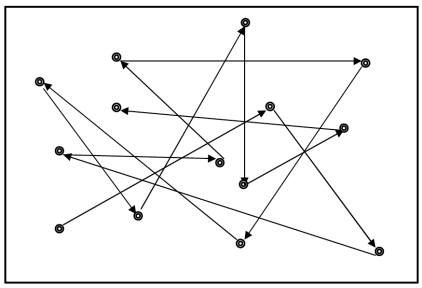


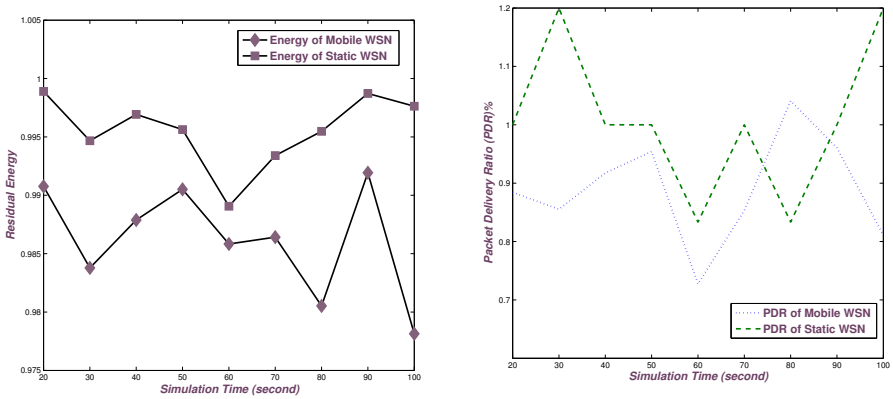
Figure 3.12 – Random waypoint model

Figure 3.12 montre un exemple de modèle de déplacement de P en utilisant le modèle de mobilité random waypoint commençant à un point aléatoire et en passant par une vitesse uniformément choisie entre 0 et 10 ms.

Le Random Waypoint, dans sa forme courante, n'atteint pas un état d'équilibre, mais plutôt que la vitesse diminue sans interruption pendant que la simulation progresse, ce qui peut fausser les résultats. Basés sur les analyses faites, les auteurs proposent une solution simple qui est de choisir une valeur strictement positive pour la vitesse minimale. Les chercheurs ont montré que la vitesse moyenne peut prendre plus que 1000 secondes du temps de simulation pour converger, si la vitesse minimale est petite. Les auteurs montrent comment implémenter un générateur de modèle de mobilité équilibré pour le Random Waypoint, vu que, si les valeurs initiales de la position et de la vitesse sont choisies d'une distribution stationnaire, la convergence est immédiate. Dans l'article (Bettstetter *et al.*, 2004), on présente une analyse mathématique de quelques propriétés stochastiques du Random Waypoint et on donne un arrangement plus profond du comportement de ce modèle comme par exemple l'effet que les nœuds tendent à se déplacer au centre de la surface de simulation. On remarque que le Random Waypoint est proche du Random Walk à la différence près que la destination choisie est toujours un point intérieur à la surface de simulation, ce qui élimine tout effet de bord.

L'approche PSO dans les RCSF mobiles

L'évaluation de l'approche PSO dans les RCSF mobiles consiste à un déploiement aléatoire de différents nombres de nœuds afin de couvrir une zone à surface variée.

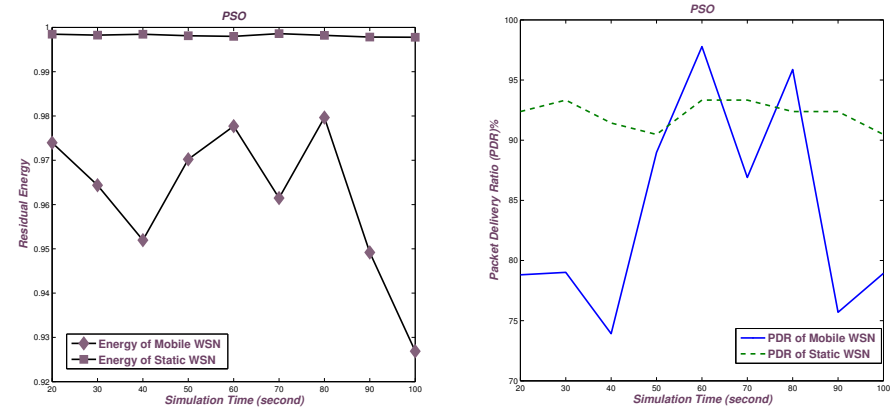


(a) Énergie Résiduelle (b) PDR

Figure 3.13 – RCSF de 50 *noeuds* : PSO

D'après les **Figures 3.13** et **3.14** Il parait évident que le taux de livraison de données que dans les RCSF mobiles est meilleur que dans les RCSF statiques. Par contre la consommation d'énergie devrait être minimisée.

Après l'évaluation de l'impact du RWP sur l'approche de PSO et l'approche d'ACO, dans la section suivante, nous comparons les résultats afin de voir laquelle est la meilleur avec la mobilité.

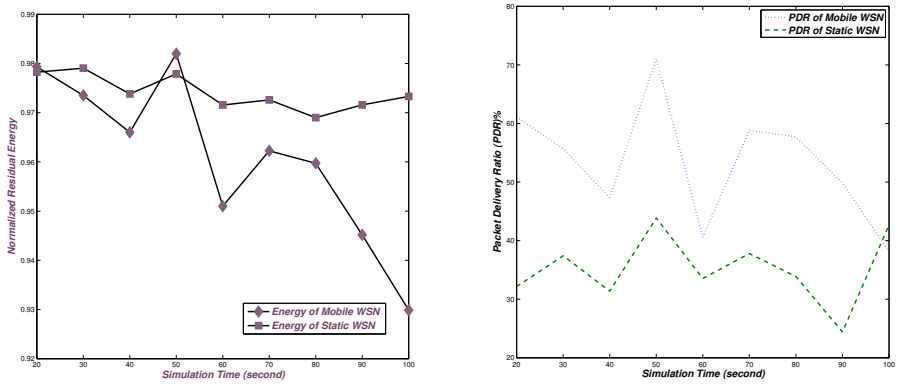


(a) Énergie Résiduelle (b) PDR

Figure 3.14 – RCSF de 100 *noeuds* : PSO

L'approche ACO dans les RCSF mobiles

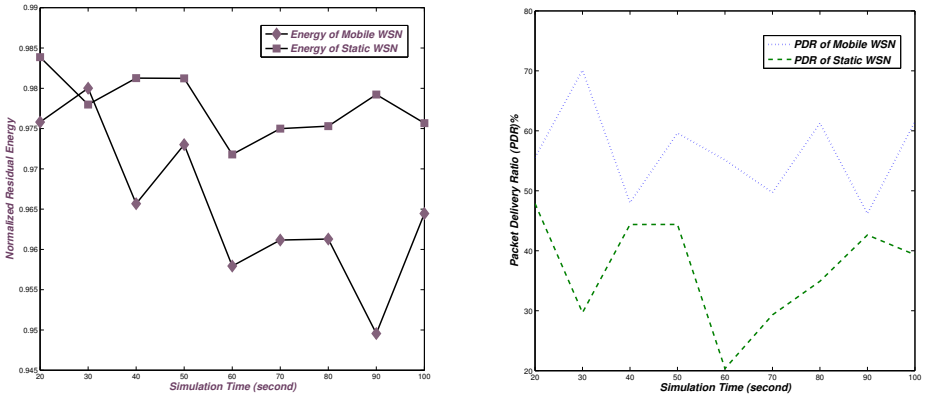
Nous avons évalué l'approche ACO dans les RCSF statiques et mobiles, on a trouvé que les modèles de mobilité peuvent affecter considérablement les performances du protocole.



(a) Énergie résiduelle (b) PDR

Figure 3.15 – L'énergie Résiduelle pour un RCSF de 100 *noeuds* : ACO

Les **Figures 3.15** et **3.16** présentent les résultats au cours des différents temps de simulation et nombre de nœuds déployés dans des domaines variés. Ainsi, montrent que l'énergie résiduelle du RCSF statique est plus élevée que le RCSF dynamique (ceci s'explique par la mobilité des nœuds).



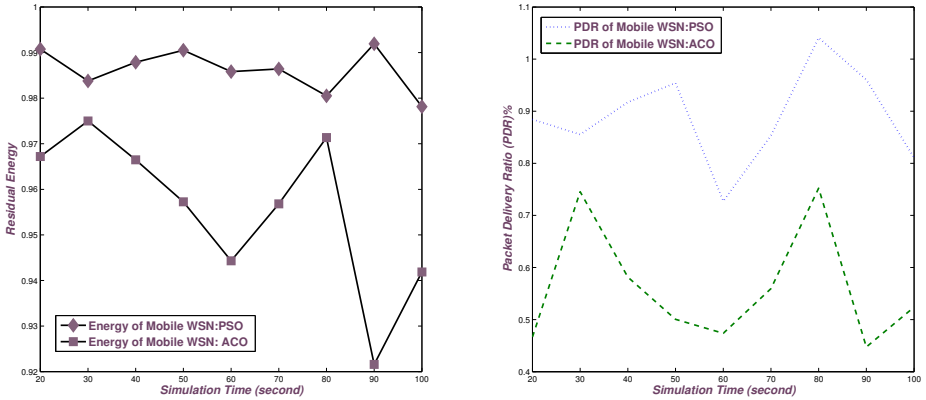
(a) Énergie résiduelle (b) PDR

Figure 3.16 – L'énergie Résiduelle pour un RCSF de 200 *noeuds* : ACO

A partir des résultats (**Figures 3.15** et **3.16**), dans le cas de capteurs stationnaires, les paquets sont perdus, ce qui peut être dû au manque de nœuds voisins actifs ou à la mort des nœuds dans le chemin. En cas de capteurs dynamiques, le changement de topologie suivant le modèle RWP donne aux paquets plus de chance d'atteindre la destination.

Comparaison des approches ACO et PSO dans les RCSF mobiles

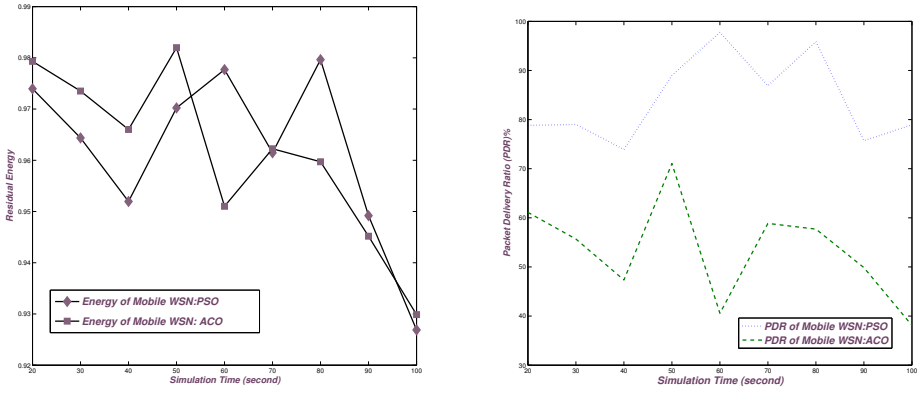
En utilisant les mêmes conditions d'expérimentation et les mêmes paramètres, les protocoles de routage PSO et ACO sont implémentés dans les RCSF mobiles basés sur le modèle RWP. Les résultats trouvés sont affichés dans **Figures 3.17** et **3.18**.



(a) Énergie résiduelle (b) PDR

Figure 3.17 – RCSF de 50 *noeuds* : PSO et ACO

Selon les résultats dans **Figures 3.17** et **3.18**, nous pouvons dire que l'approche PSO donne de bons résultats dans les RCSF mobiles par rapport à l'ACO en terme de La consommation d'énergie et la PDR.



(a) Énergie résiduelle (b) PDR

Figure 3.18 – RCSF de 100 *noeuds* : PSO et ACO

Cependant, lorsque le nombre de capteurs augmente jusqu'à 100 nœuds, nous avons constaté que le protocole basé sur ACO, à un certain temps de simulation, offre moins de

consommation d'énergie que celui de PSO (voir **Figure 3.18**). En dépit de ce fait, le PSO reste plus bien en considérant le taux de livraison des paquets (PDR).

À partir des résultats trouvés, nous déduisons que la mobilité basée sur le modèle RWP affecte les comportements des protocoles de routage. L'approche PSO fournit de meilleurs résultats dans les RCSF statiques, dans les RCSF mobiles, les résultats de l'approche ACO pourraient être meilleurs surtout lorsque le nombre de nœuds augmente.

3.5 Conclusion

Les RCSF traditionnels statiques ont plusieurs limitations pour la gestion de plusieurs situations lorsque les conditions du réseau changent. L'introduction de la mobilité aux RCSF peut améliorer considérablement la capacité du réseau et ainsi le libérer de ses limitations.

Dans ce chapitre, nous présentons un nouveau travail, où en premier lieu nous avons proposé une amélioration d'un protocole de routage basé sur ACO. En second lieu, nous avons analysé l'impact du modèle de mobilité random waypoint sur les métaheuristiques les plus utilisés pour le routage : PSO et ACO. En modifiant le nombre de nœuds, le temps de simulation, la longueur et Largeur de la zone de simulation.

Les résultats des expériences montrent que, en termes de taux de livraison des paquets et de consommation d'énergie, l'approche basée sur le PSO offre les meilleurs résultats que l'approche d'ACO. Pour les deux approches, le modèle RWP affecte la consommation énergétique, cependant, il améliore le PDR.

Notre approche basée sur ACO a montré des limitations en efficacité par rapport à l'approche basée sur PSO, cela nous a motivé à concevoir un nouveau protocole de routage dans les RCSF utilisant une nouvelle métaheuristique présenté en détail dans le chapitre suivant.

Chapitre

4

PROTOCOLE DE ROUTAGE BASE SUR TLBO

Sommaire

4.1 Introduction

Le problème de routage dans les RCSF est considéré comme un problème d'optimisation NP-difficile dans plusieurs cas. Plusieurs protocoles de routage sans basés sur des métaheuristiques inspirées de la nature, telles que ACO et PSO. Malgré les bons résultats fournis par ces métaheuristiques, leur adaptation au problème et leur paramétrage reste un peu difficile.Dans ce contexte, nous avons proposé une nouvelle approche pour faire face à ces difficultés, en se basant sur une métaheuristique récente appelée :Teaching-Learning-Based Optimization (TLBO). TLBO est une méthode robuste qui ne nécessite aucun paramétrage et est composée de deux phases essentielles : l'enseignant et l'étudiant. L'approche proposée Teaching-Learning-Based Optimization based Routing (TLBOR) permet d'obtenir une consommation d'énergie plus faible ainsi une durée de vie des RCSF prolongée, selon les résultats comparatifs.

Le reste de ce chapitre est organisé comme suit. La section 2 présente la metaheuristique Teaching-Learning-Based Optimization (TLBO). La section 3 décrit l'adaptation du TLBO au problème de routage dans les RCSF. La section 4 montre l'évaluation de performance de notre algorithme. Enfin, la section 5 conclusion de ce travail.

4.2 Teaching-Learning-Based Optimization (TLBO)

Teaching-Learning-Based Optimization (TLBO) est une nouvelle métaheuristique proposée par Rao et al. qui s'est inspiré de l'influence de l'enseignant et l'interaction des étudiants (Rao *et al.*, 2011). Cette méthode a surpassé plusieurs métheuristiques connues dans la littérature en utilisant les instances de benchmark (Rao *et al.*, 2012). TLBO a été adaptée aux différents problèmes tels que le problème de routage multidiffusion QoS (Naik *et al.*, 2013) et le problème de répartition optimale de la puissance réactive (Ghasemi *et al.*, 2015).

elle est composé de deux parties fondamentales : la phase de l'enseignant et la phase de l'étudiant. La figure 4.1 décrit en détail le processus de TLBO.

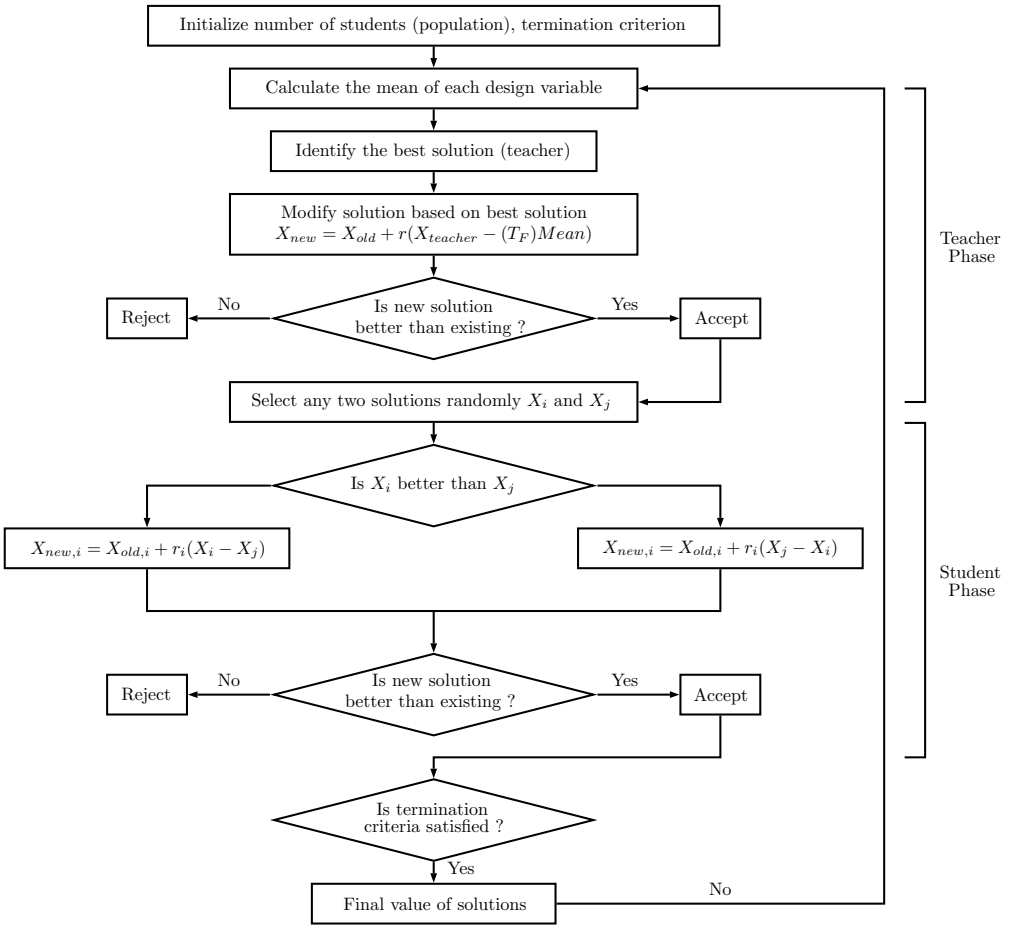


Figure 4.1 – Teaching-Learning-Based Optimization (TLBO) (Rao *et al.*, 2011).

4.2.1 Teacher Phase

TLBO comme d'autre métaheuristique inspirées de la nature est une métaheuristique à population. La population initiale est un groupe d'étudiants et les matières étudiées sont

des variables de conception. Après l'évaluation de l'ensemble de la population à l'aide de la «fitness» la meilleure solution est considérée comme un enseignant. Dans cette phase, un enseignant essaie d'améliorer la moyenne de la classe des étudiants en se basant sur la différence entre ses connaissances et celles des étudiants. (Voir Equation 4.1 et 4.2)

$$X_{new} = X_{old} + Difference_D \tag{4.1}$$

La différence entre la moyenne des étudiants et l'enseignant est calculé suivant cette équation :

$$Difference_D = r(X_{teacher} - T_F Mean) \tag{4.2}$$

Où $Mean$ est la moyenne, $X_{teacher}$ est la note d'enseignant, r un nombre aléatoire dans [0,1] et T_F un facteur d'enseignant qui est une valeur décidée au hasard avec une probabilité égale à 1 ou 2, calculé en utilisant l'équation 4.3.

$$T_F = round[1 + rand(0, 1)]. \tag{4.3}$$

4.2.2 Learner Phase

Au cours de la deuxième phase (phase d'apprentissage ou phase d'étudiant), les étudiants essaient d'améliorer leurs connaissances par l'interaction entre eux. En formant des pairs d'étudiant au hasard ainsi, leur niveau augmente selon le processus ci-dessous :

Dans la i^{eme} itération, nous choisissons aléatoirement un étudiant X_j, tel que $i \neq j$, et nous mettons à jour le X_i suivant l'équation 4.4.

$$\begin{cases} X_{new,i} = X_{old,i} + r_i(X_i - X_j), & si\ f(X_i) < f(X_j) \\ X_{new,i} = X_{old,i} + r_i(X_j - X_i), & si\ f(X_j) < f(X_i) \end{cases} \tag{4.4}$$

Pour bénéficier des avantages de TLBO dans les RCSF, nous proposons un nouveau protocole de routage basé sur cette métaheuristique.

4.3 Approche proposée basée sur TLBO

Les RCSF sont connus par la stricte contrainte énergétique et les possibilités limitées de renouvellement de batterie. Il est donc important d'optimiser la consommation d'énergie pour le routage, afin de prolonger autant que possible la durée de vie du réseau. Dans cette section, nous proposons un nouveau protocole de routage pour les RCSF, Teaching-learning-based optimization based routing (TLBOR). TLBOR est un nouveau protocole qui n'est pas centralisé, qu'on doit installer dans chaque nœud du réseau. Son processus commence par l'initialisation de la population de chemins, puis la recherche du chemin optimal à l'aide de l'algorithme TLBO, et finalement l'envoie des données à travers ce chemin. L'adaptation de l'algorithme TLBO au problème de routage dans les RCSF est

détaillée comme suit.

4.3.1 Population initiale

Le nœud source envoie un message de broadcast à ses voisins afin de recueillir des informations relatives à certains chemins qui mènent au sink (voir Fig 4.2). Une fois les paquets de requêtes reçus, les nœuds vérifient leurs table de routage. Si un chemin existe, le nœud source reçoit directement cette information. Sinon, les nœuds récepteurs envoient à leurs tour des broadcasts à leurs voisins, et ainsi de suite.

Le nœud source initialise la population, en fonction du nombre de chemins vers le sink reçus. On note la population des chemins par $P = \{p_1, \ldots, p_i, \ldots, p_m\}$ où chaque chemin p_i est de la forme $p_i = \{N_s, \ldots, n_k, \ldots, n_{sink}\}$.

L'initialisation de la population est une tâche accomplie par un ensemble de nœuds du réseau. Par un exemple (voir Fig 4.2), un nœud source S détectant un événement envoie une requête à ses voisins, nœud 1, nœud 8, nœud 2 et nœud r. Un voisin r vérifie sa table de routage, au cas où un ou plusieurs chemins vers le sink sont stockés, r informe le nœud source S. Si non, le nœud r demande de la même façon que S à ses voisins des chemins menant au sink, les voisins de r se comportent de la même manière que le nœud r et ainsi de suite. Après un temps d'attente déterminé, tous les paquets de réponse sont reçus par le nœud source S, qui collecte l'information et génèrent une population initiale par un nombre aléatoire de chemins.

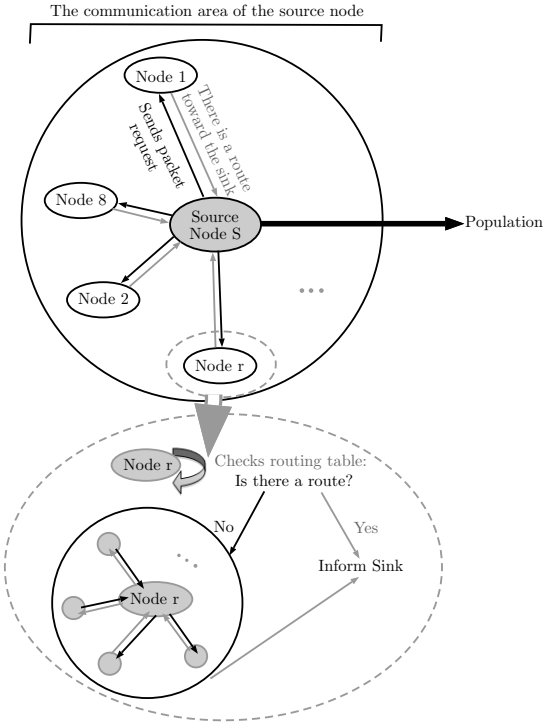


Figure 4.2 – Initialisation de population.

La figure 4.2 montre comment le nœud source s envoie une demande à ses nœuds

voisins et comment un nœud voisin r réagit après la réception de cette demande, il informe s qu'au moins un chemin existe ou cherche à son tour des chemins possible vers le sink en demandant aux voisins.

4.3.2 Division des données

Afin de gérer la bande passante, le nœud source divise les données brutes en N parties. Les données brutes contiennent des informations telles que l'identification de l'événement, l'heure et les données relatives à l'événement détecté. Avant le transfert, chaque partie est associée aux paramètres de routage. Ces paramètres sont l'identification du nœud suivant N_ID, le numéro d'ordre N_S et le chemin vers le sink $Path$ comme montré dans la figure 4.3.

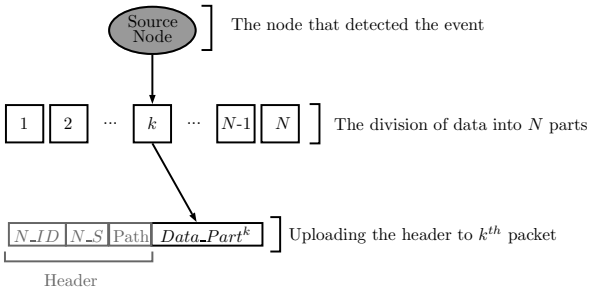


Figure 4.3 – Data division.

Quand la transmission des données est accomplie, le sink combine les parties reçues pour former les données brutes initiales.

4.3.3 Implémentation de TLBO

Nous appliquons l'algorithme TLBO aux problèmes de routage, suivant les cinq étapes ci-dessous :

Étape 1 :

L'initialisation des paramètres d'optimisation considérés pour le problème de routage et la définition de la fonction objectif.

La taille de la population : (Pn) dépend du nombre de nœuds dans le réseau et le temps d'attente. Le nœud source attend un temps TTL (Time To Live) les requêtes provenant des voisins afin de former un nombre de chemins (du nœud source au sink), ce nombre est considéré comme la taille de la population.

Le critère de terminaison : Si le même chemin est revisite plusieurs fois pendant un certain nombre d'itérations, ce chemin est considéré comme la solution. Ainsi, on quitte la boucle d'exécution

Le nombre de variables de conception : Dans le problème de routage, les variables de conception sont discrètes, alors le nombre de ces variables peut être choisi de façon aléatoire selon l'ensemble de toutes les valeurs discrètes candidates dans la population.

La fonction objectif : On utilise le modèle d'énergie approximative "First Order Radio Model" de Heinzelman et al. (Heinzelman *et al.*, 2000)(voir la figure 4.4). Pour envoyer et recevoir un message, les besoins énergétique sont formulés comme suit :

- Pour envoyer k bits à un récepteur distant par d mètres, l'émetteur consomme :

$$E_{Tx}(k,d) = (E_{elec} \times k) + (\epsilon_{amp} \times k \times d^2)$$

- Pour recevoir k bits, le récepteur consomme :

$$E_{Rx}(k) = E_{elec} \times k$$

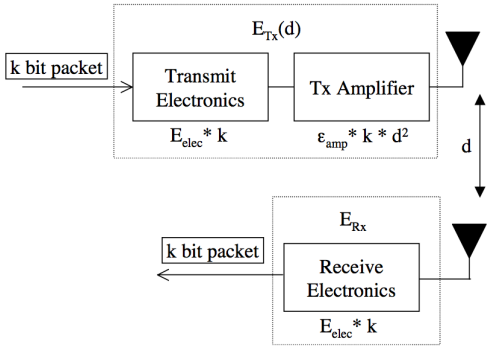


Figure 4.4 – First Order Radio Model (Heinzelman *et al.*, 2000).

Supposons que le chemin $X = (n_s, \ldots, n_k, \ldots, \ldots, n_{sink})$ contient L nœuds, l'énergie consommée pour transmettre un message de $k-bits$ le long de ce chemin est calculé comme suit :

$$E(X) = 2 \times (L-1) \times E_{elec} \times k + \epsilon_{amp} \times k \times \sum_{i=1}^{L-1} d_{i,i+1}^2 \tag{4.5}$$

Où E_{elec}= $50nJ/bit$, ϵ_{amp}=$100nJ/bit/m^2$ sont respectivement l'énergie de transmission et d'amplification électroniques. $d_{i,i+1}^2$ est le carré de la distance euclidienne entre le nœud i et le nœud $i+1$. Le modèle de fonction objectif qui peut équilibrer la consommation d'énergie et le délai de transmission d'un chemin X autant que possible est indiqué dans l'équation 4.6.

$$f(X) = E(X) \times \frac{L}{E_{Min}} \times E_{Avg} + D(X) \tag{4.6}$$

Où E_{Min} est l'énergie résiduelle minimale d'un nœud dans le chemin X, E_{Moy} est l'énergie résiduelle moyenne de tous les nœuds en X et $D(X)$ le délai de transmission des données par le chemin X.

Étape 2 :

La population s'exprime comme suit :

$$P = \begin{pmatrix} (n_s, n_{1,1}, n_{1,2}, \ldots, n_{1,q_1}, n_{sink}) \\ (n_s, n_{2,1}, n_{2,2}, \ldots, n_{2,q_2}, n_{sink}) \\ \vdots \\ (n_s, n_{m,1}, n_{m,2}, \ldots, n_{m,q_m}, n_{sink}) \end{pmatrix}$$

Où les tailles des chemins ne sont pas nécessairement les mêmes. On note q_i la longueur du chemin ayant l'indice i.

Les valeurs objectives sont respectivement les suivantes :

$$\begin{pmatrix} f_1 \\ \vdots \\ f_i \\ \vdots \\ f_m \end{pmatrix}$$

Étape 3 : (Teacher phase)

Dans cette phase, il faut calculer la moyenne de chaque colonne dans la population P. En fait, cela ne peut pas être exprimé analytiquement dans le routage. Par conséquent, nous devons redéfinir quelques opérations mathématiques de base afin d'utiliser les équations de TLBO. Nous proposons d'utiliser l'Edge Recombination Operator (ERO) pour faire face aux espaces discrets. ERO a été développé pour construire une génération qui hérite autant d'informations que possible de la structure des parent (Whitley *et al.*, 1989).

Dans la figure 4.5, On présente un exemple de RCSF composé de 80 nœuds, un nœud source S qui détecte un événement et on suppose que la population initiale est :

$$pop = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{pmatrix} = \begin{pmatrix} (S, 48, 46, 43, 61, 35, Sink) \\ (S, 48, 46, 25, 24, 18, Sink) \\ (S, 78, 40, 23, 21, 35, Sink) \\ (S, 78, 43, 61, 35, Sink) \\ (S, 43, 61, 35, Sink) \\ (S, 43, 24, 18, Sink) \end{pmatrix}$$

La moyenne de la population *pop* est un chemin X_{Mean} résultant de ERO (voir algorithm 4.1) entre tous les chemins de la population.

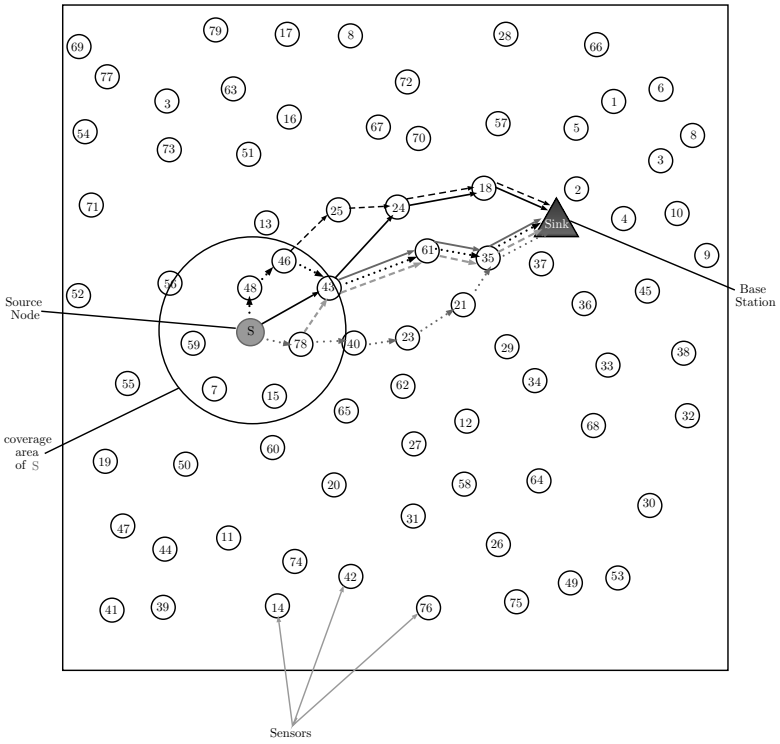


Figure 4.5 – Exemple

Algorithme 4.1 Edge Recombination Operator

```
Parent 1, Parent 2, ENFANT = Vide
Générer une liste de voisins
X = le premier nœud d'un parent aléatoire.
tantque l'enfant n'est pas plein, faire
  Ajoute X à ENFANT
  Supprimer X des listes des voisins
  si La liste des voisins de X est vide alors
    Z = nœud aléatoire pas déjà dans ENFANT
  sinon
    Détermine le voisin de X qui a le moins de voisins
    S'il y a égalité, choisissez un au hasard
    Z = nœud choisi
  finsi
  X = Z
fin tantque
```

Exemple d'Edge Recombination Operator :
Ayant les deux solutions parent suivantes :

Parent 1	S	48	46	43	61	35	Sink
Parent 2	S	48	46	25	24	18	Sink

Nœud	Voisins		
S	48		
48	S	46	
46	48	43	25
43	46	61	
61	43	35	
35	61	Sink	
25	46	24	
24	25	18	
18	24	Sink	

Tout d'abord, nous choisissons au hasard le premier nœud d'un parent.

ENFANT	S

Ensuite, après avoir supprimer S de toutes les listes des voisins, nous voyons que 48 est le seul voisin dans la liste de S. Donc,

ENFANT	S	48

Ensuite, après avoir supprimer 48 de toutes les listes des voisins, 46 est le seul voisin dans la liste de 48. Donc,

ENFANT	S	48	46

Ensuite, après avoir supprimer 46 de toutes les listes des voisins, 43 et 25 ont tous deux un seul voisin, donc on choisi au hasard un entre les deux :

ENFANT	S	48	46	43

Ensuite, après avoir supprimer 43 de toutes les listes des voisins, on voit que 61 est le seul voisin dans la liste de 43. Donc,

ENFANT	S	48	46	43	61

Ensuite, après avoir supprimer 61 de toutes les listes des voisins, le voisin de 61 est 35 :

ENFANT	S	48	46	43	61	35

Ensuite, après avoir supprimer 35 de toutes les listes des voisins, 35 n'a qu'un seul voisin qui est le Sink :

ENFANT	S	48	46	43	61	35	Sink

L'utilisation de la technique de croisement ERO doit soumettre à quelques contraintes liées au RCSF. Le premier nœud dans un enfant est obligatoirement le nœud source. On cas d'appartenance du Sink à la liste des voisins, On le choisi et on arrête le processus.

Après la permutation ERO entre tous les chemins de la population *pop*, la moyenne est $X_{Mean} = (S, 78, 40, 23, 21, 35, Sink)$

L'enseignant est la meilleure solution pour cette itération :
$X_{teacher} = min_{1<=i<=m}(f(X_i))$
Selon l'exemple $X_{teacher} = (S, 43, 61, 35, Sink)$

La différence est exprimée dans l'équation 4.2. Dans ce cas (les espaces discrets) la différence est présentée par l'intersection entre deux chemins.
$X_{teacher} \cap X_{Mean} = Difference_D = \{35\}$

En utilisant cette différence, une nouvelle solution est calculée X_{new}. La différence est un nombre de nœuds qui remplacera d'autres nœuds dans l'ancienne solution X_{old} afin de l'améliorer. X_{new} est acceptée si sa valeur objective est supérieure à celle de X_{old}.

Étape 4 : (Learner phase)

Sélectionnant un X_j aléatoire de la population où $i \neq j$. Dans les espaces discrets, le X_{new} est le résultat de la combinaison ERO de X_i et X_j.

$X_{new} = ERO(X_i, X_j)$

Étape 5 :

Si le critère de terminaison est atteint, arrêtez le processus, sinon passez à l'étape 3.

4.3.4 Envoi des informations sur l'événement

Le nœud source envoie les parties de données au Sink à travers le chemin optimal. Lorsqu'un ou plusieurs nœuds s'épuisent ou disparaissent du chemin, le dernier nœud ayant des informations construit une nouvelle route en utilisant la procédure de TLBO (de la même manière que le nœud source). Une fois la livraison réalisée, la station de base rassemble les paquets reçus, et forme donc les données brutes. Si une partie est manquante, le nœud source reçoit un paquet d'avertissement du sink sur le même chemin. La retransmission du paquet manquant, suit la même procédure décrite ci-dessus.

4.3.5 Algorithme de routage basé sur TLBO

L'algorithme utilisé pour optimiser le problème de routage dans les RCSF est présenté par algorithm 4.2)

Algorithme 4.2 TLBOR Pseudo-code

g 0 ;
Initialize_population(P, pop_size)
tantque $g \neq num\_gen$ **faire**
pour $i = 1 \rightarrow pop\_size$ **faire**
(Teacher Phase)
$X_{Mean} \leftarrow Calculate\_mean\_vector(P)$
$X_{teacher} \leftarrow Best\_solution(P)$
$Difference = X_{teacher} \cap X_{Mean}$
$X_{New} \leftarrow Replace\_nodes(X_{i,old}, Difference)$
$Evaluate(X_{i,new})$
si $X_{i,New}$ *meilleur que* $X_{i,old}$ **alors**
$X_i \leftarrow X_{i,New}$
finsi

(Learner Phase)
$j \leftarrow Random(pop\_size)$ ou $j \neq i$
$X_{i,new} = Edge\_Recombination\_Operator(X_i, X_j)$
$Evaluate(X_{i,new})$
si $X_{i,New}$ *meilleur que* $X_{i,old}$ **alors**
$X_i \leftarrow X_{i,New}$
finsi
fin pour
$g \leftarrow g + 1$
fin tantque

Le pseudocode proposé pour le routage dans RCSF utilisant TLBO contient certaines fonctions, qui sont décrites en détail comme suit :

Initialize_population() : Cette fonction va initialiser la population, un ensemble de chemins collecter aléatoirement après un temps TTL (Fig 4.2). *Calculate_mean_vector*() une fonction qui calculera la moyenne de la population à l'aide d'edge recombination operator entre tous les chemins de la population (voir l'algorithme ERO 4.1). *Best_solution*() celle-ci vérifiera la meilleure solution dans la population qui a la valeur maximale de la fonction objective. *Replace_nodes*() cette fonction remplacera certains nœuds du chemin par les nœuds dans la $Difference$, dans le cas où cela est possible en considérant la zone de couverture pour générer une nouvelle solution. La fonction *Evaluate*() évalue une solution en utilisant la fonction objective.

Edge_Recombination_Operator() celle-là considérera un chemin aléatoire de la population pour en créer un nouveau basé sur la combinaison ERO.

4.4 Expérimentations et résultats

Cette section évalue les performances du protocole de routage TLBOR proposé suivant plusieurs simulations et comparaisons. Pour garantir une marge de crédibilité des résultats obtenus, nous considérons les mêmes conditions et environnement pour tous les protocoles mis en œuvre.

4.4.1 Données expérimentales

On a implémenté les protocoles de routage dans MATLAB et C++ sous un système d'exploitation Windows de 64 bits. Les expériences sont effectuées sur un ordinateur avec $Intel(R)\ i5\ 3.20GHz\ CPU$ et $4GB$ de RAM. Les paramètres utilisés dans les simulations sont indiqués dans le Tableau 4.1.

Paramètres	**Valeur**
No. de nœuds	10, 20, 30, 40, ..., 90,100 et 200
Surface de Simulation	$300m^2$ $400m^2$, $500m^2$, $600m^2$ et $1000m^2$
Topologie	Topologie Plate
Type du Réseau	Réseau Statique
Type de Déploiement	déploiement aléatoire
Modèle d'énergie	First Order Radio Model
L'énergie Initiale d'un Nœud	10, 20 et 30 Joules
No. de simulations	15

Tableau 4.1 – Paramètres des simulations

Comme le montre la Figure 4.6, le nombre de nœuds dans ces scénarios variait de 10 à 100, avec une incrémentation de 10. Les zones de simulation sont modifiées pour changer les densités des réseaux. Les cercles représentent les nœuds capteurs et les triangles représentent le sink.

Afin de vérifier les performances des protocoles en termes d'efficacité, certain mesures peuvent être utilisées pour les comparer. Tels que la variation des nombres de nœuds, la zone de couverture et le nombre de paquets.

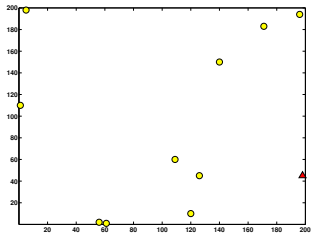
Nous avons effectué des expériences approfondies suivant trois cas, afin de faire une comparaison complète entre notre protocole proposé TLBOR et d'autres protocoles.

1. Cas 1 : Pour tous les scénarios, le niveau d'énergie de tous les nœuds est fixé à $10J$.
2. Cas 2 : On utilise des RCSF composés de 100 et de 200, avec une répartition uniforme de trois niveaux d'énergie de $10J$, $20J$ et $30J$.
3. Cas 3 : Nous avons appliqué le A-test qui est une mesure de supériorité stochastique permettant de classer les échantillons en fonction de leurs valeurs.

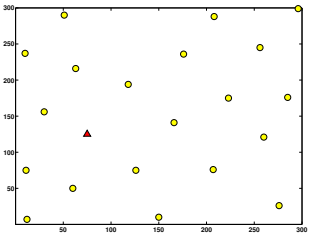
Pour évaluer l'efficacité du protocole de routage proposé, nous considérons les mêmes conditions utilisées dans les expérimentations. Tous les événements sont considérés au même endroit et tous les paquets sont envoyés à partir du même nœud source pour les cinq algorithmes et dans chaque scénario. Nous avons comparé notre approche TLBOR aux protocoles de routage suivants : L'approche ACO (El Ghazi *et al.*, 2014), l'approche PSO (EL Ghazi et Ahiod, 2016), l'approche IHSBEER (Zeng et Dong, 2016) et le protocole AODV (Perkins *et al.*, 2003).

4.4.2 Premier cas étudié

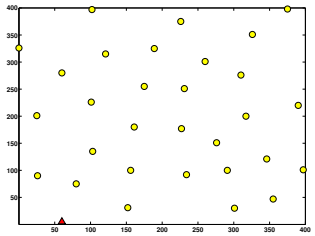
Dans ce premier cas, le niveau d'énergie est fixe pour tous les nœuds dans tous les scénarios, alors que le nombre de paquets envoyé change. Pour chaque protocole de routage



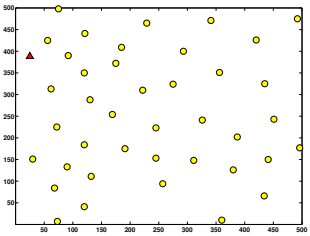
(a) 10 *noeuds*



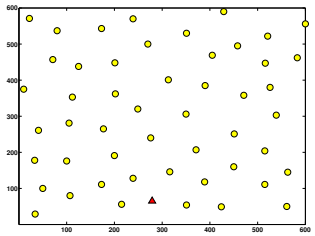
(b) 20 *noeuds*



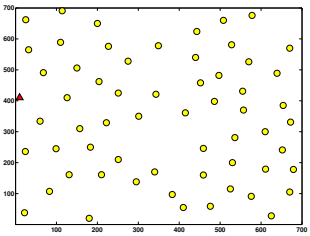
(c) 30 *noeuds*



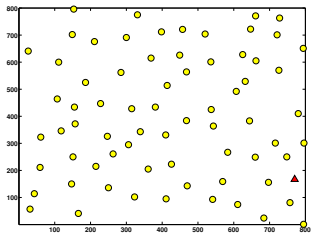
(d) 40 *noeuds*



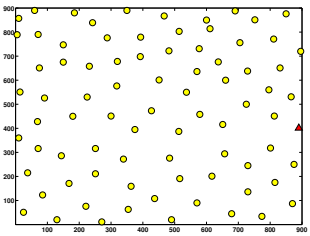
(e) 50 *noeuds*



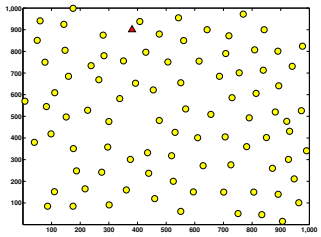
(f) 60 *noeuds*



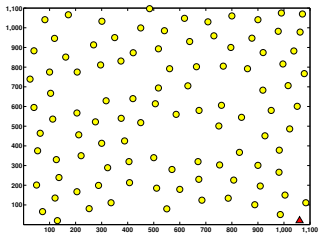
(g) 70 *noeuds*



(h) 80 *noeuds*



(i) 90 *noeuds*



(j) 100 *noeuds*

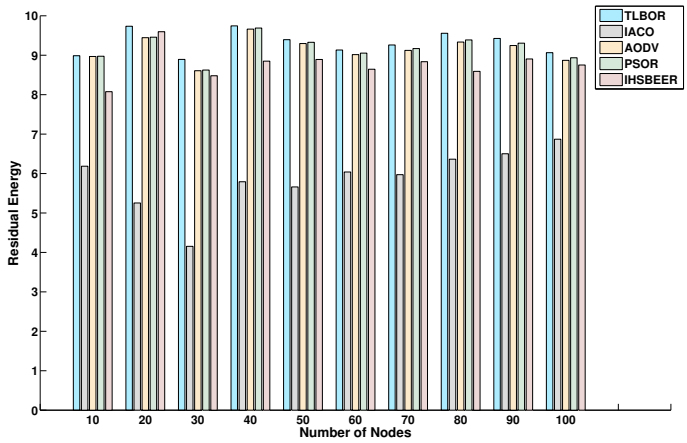
Figure 4.6 – Scénarios

utilisé, nous avons simulé la transmission de 600 et 1000 paquets suivant les 10 scénarios de la Figure 4.6.

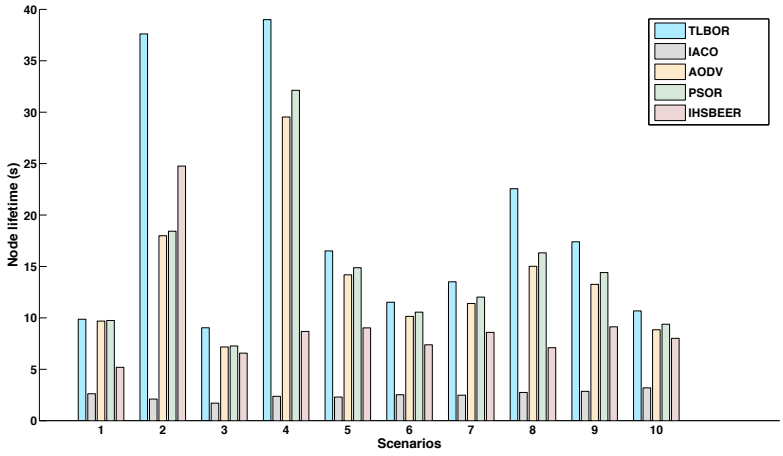
La Figure 4.7 et 4.8 montrent l'énergie résiduelle moyenne des nœuds, étant la métrique utilisée pour évaluer les performances des protocoles.

la durée de vie d'un nœud Lt est calculée suivant l'équation 4.7 proposé par les auteurs dans Sosedka et Shostko (2014). Tel que E_0 est l'énergie du batterie et P la puissance consommée par le périphérique.

$$Lt = \frac{E_0}{P} \tag{4.7}$$



(a) *Resultats d'Energie Residuelle*

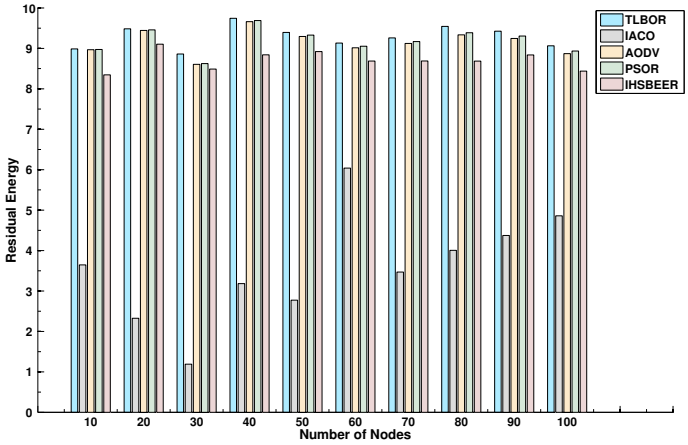


(b) *Durée de Vie d'un Noeud*

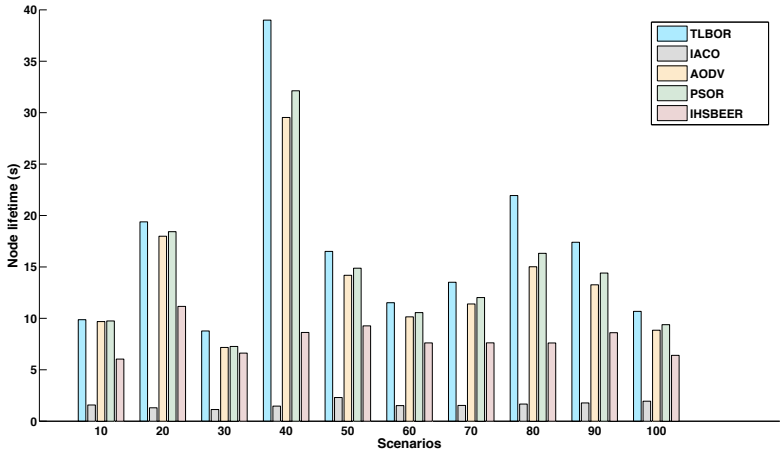
Figure 4.7 – Résultats de simulation pour différents RCSF (600 paquets))

Après la réception de 600 paquets, les résultats pour différents RCSF sont présentés

par la Figure 4.7. Nous pouvons déduire que TLBOR donne les meilleurs résultats dans la grande majorité des scénarios. Une énergie résiduelle supérieure signifie une faible consommation d'énergie, ainsi montre que TLBOR peut économiser beaucoup plus d'énergie que IACOR, PSOR, IHSBEER et AODV, pour transmettre un paquet au sink.



(a) *Resultats d'Energie Residuelle*



(b) *Durée de Vie d'un Noeud*

Figure 4.8 – Résultats de simulation pour différents RCSF (1000 paquets)

La Figure 4.8 montre l'énergie résiduelle après la réception de 1000 paquets, ainsi nous pouvons voir que les résultats obtenus par TLBOR sont meilleurs que ceux obtenus par PSOR, IACOR, IHSBEER et AODV. L'énergie résiduelle affecte directement la durée de vie du réseau, donc un RCSF avec une énergie résiduelle maximale a automatiquement une durée de vie plus longue. D'après les résultats, il est très clair que TLBOR fonctionnera mieux pour prolonger la durée de vie du réseau que les autres protocoles de routage dans tous les scénarios, comme le montre la Figure 4.8.

4.4.3 Deuxième cas étudié

Dans ce cas, nous comparons le comportement de notre protocole TLBOR aux protocoles de routage IACOR, PSOR, IHSBEER et AODV dans deux RCSF composés de 100 et 200 nœuds. Dans les mêmes conditions d'expérimentation, nous simulons la transmission d'un nombre varié de paquets de 64 à 640 paquets d'un pas de 64. On peut voir à partir de la Figure 4.9 qu'en général même en augmentant le nombre de paquets, le protocole TLBOR donne les meilleurs résultats.

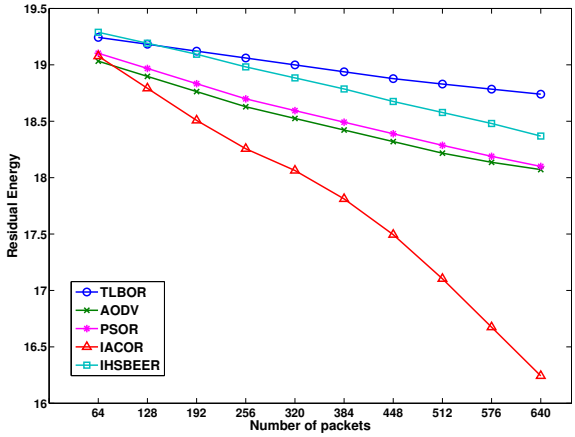


Figure 4.9 – Résultats de simulation pour un RCSF de 100 nœuds

Pour être certain que notre approche basée sur TLBO est performante pour les problèmes à grande échelle, nous avons effectué des expériences supplémentaires sur un RCSF de 200 nœuds. Le TLBOR présenté dans la figure 4.10 par des ciercles bleus, surpasse clairement tous les autres protocoles de routage. À partir des résultats obtenus, nous pouvons déduire que notre approche basée sur TLBO est la meilleure pour économiser l'énergie d'un RCSF ainsi que pour prolonger sa durée de vie.

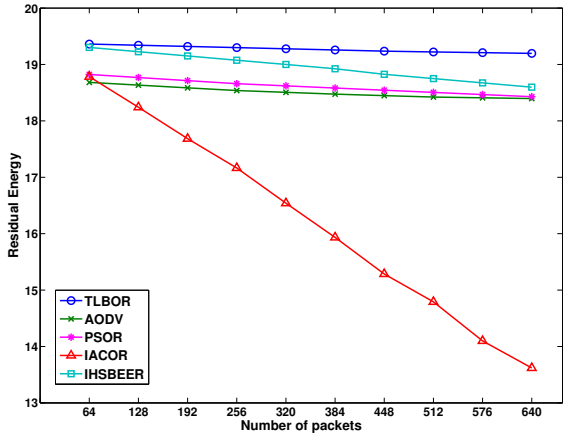


Figure 4.10 – Résultats de simulation pour un RCSF de 200 nœuds

De la figure 4.10, il est clair que notre protocole de routage basé sur TLBO, optimise l'énergie dans les RCSF mieux que le protocole PSOR, IACOR, IHSBEER et AODV.

4.4.4 Troisième cas étudié

En effectuant de nombreuses simulations, nous avons montré que le protocole proposé (TLBOR) est meilleur que les protocoles de routage mentionnés ci-dessus. L'énergie résiduelle pour notre approche est la plus élevée ainsi, la consommation d'énergie est réduite et la durée de vie d'un RCSF est optimisée surtout lorsque les densités sont élevées. Afin de vérifier l'efficacité de notre proposition, nous avons appliqué le A-test (la A-mesure Vargha-Delaney (Vargha et Delaney, 2000)).

La A-mesure Vargha-Delaney nous indique en moyenne à quelle fréquence une technique dépasse l'autre. Le A-test renvoie une valeur entre 0 et 1, lorsque la A-mesure est exactement 0.5, il n'y a pas de différence entre les deux techniques. Lorsque la A-mesure est inférieure à 0.5, la deuxième technique est plus performante, sinon la première technique l'est. Plus la valeur d'A-mesure est proche de 0.5, plus la différence entre les techniques est faible ; Plus elle est loin de 0.5 plus la différence est importante.

Dans notre cas, les échantillons représentent l'énergie résiduelle de chaque algorithme en 10 excusions indépendantes. La valeur d'A-test représente la probabilité qu'une observation sélectionnée au hasard d'un échantillon soit meilleure qu'une observation sélectionnée au hasard de l'autre échantillon. La valeur 1.000 montre que le protocole de routage TLBOR proposé dépasse les autres protocoles et cela dans tous les scénarios étudiés.

TLBOR vs / **Scenarios**	**IACOR**	**PSOR**	**IHSBEER**	**AODV**
Sc. 10 nœuds	1.000	1.000	1.000	1.000
Sc. 20 nœuds	1.000	1.000	1.000	1.000
Sc. 30 nœuds	1.000	1.000	1.000	1.000
Sc. 40 nœuds	1.000	1.000	1.000	1.000
Sc. 50 nœuds	1.000	1.000	1.000	1.000
Sc. 60 nœuds	1.000	1.000	1.000	1.000
Sc. 70 nœuds	1.000	1.000	1.000	1.000
Sc. 80 nœuds	1.000	1.000	1.000	1.000
Sc. 90 nœuds	1.000	1.000	1.000	1.000
Sc. 100 nœuds	1.000	1.000	1.000	1.000

Tableau 4.2 – A-test pour 10 scénarios

La table 4.2 montre le A-test d'une comparaison par paire entre TLBOR et d'autres algorithmes de littérature dans chaque scénario.

Selon les résultats des expériences, nous pouvons constater que le protocole TLBOR a toujours présenté les meilleurs résultats suivant toutes les métriques. Nous avons conclu que TLBOR peut optimiser la consommation d'énergie d'un RCSF et prolonger sa durée de vie mieux que IACOR, PSOR, IHSBEER et AODV.

4.4.5 Discussion

Plusieurs algorithmes d'optimisation nécessitent des paramètres ce qui affectent les performances de la méthode. Le PSOR nécessite des facteurs d'apprentissage, une variation de poids et une valeur maximale de vélocité ; IHSBEER nécessite un taux de considération de la mémoire d'harmonie, un taux d'ajustement de hauteur et un certain nombre d'improvisations ; IACOR nécessite des paramètres comme la valeur de phéromone le taux d'évaporation. Contrairement à d'autres méthodes d'optimisation, TLBOR ne nécessite aucun paramètre d'algorithme, ce qui simplifie énormément son implémentation. En tant que PSOR, TLBOR utilise la meilleure solution de l'itération pour changer la solution courante dans la population, augmentant ainsi le taux de convergence. TLBOR ne divise pas la population, elle utilise deux phases différentes : "phase d'enseignant" et "phase d'apprenant". TLBOR utilise la valeur moyenne de la population pour mettre à jour la solution.

Le TLBOR proposé dans ce chapitre est un nouveau protocole de routage utilisant la métaheuristique TLBO dans un espace de recherche discret, cela par la redéfinition des équations en se basant sur l'edge recombination operator (ERO). Cette approche a montré une efficacité en termes de consommation d'énergie et d'extension de la durée de vie des RCSF, comme le montrent les résultats des expériences ci-dessus. Ces résultats satisfaisants ont été obtenus puisque TLBO calcule toujours des chemins optimales possible.

Notre algorithme TLBOR a dépassé les autres algorithmes de littérature : basés sur des essaims (PSO, ACO et HS) ou non basés sur des essaims (AODV) dans tous les scénarios 4.6. L'approche TLBOR est très efficace en termes de consommation d'énergie et de durée de vie des RCSF selon les résultats présentés dans les Figures 4.7, 4.8, 4.9, 4.10 et la Table 4.2 qui résume les résultats comparatifs et montre le pourcentage de succès dans chaque cas étudié. La performance de TLBOR est due aux raisons suivantes :

— L'absence de paramètres dans TLBO a considérablement simplifié son adaptation au problème de routage étudié ;
— Les opérateurs que nous utilisions pour redéfinir des équations dans TLBO telles que l'opérateur de recombinaison de Bord (ERO) et l'intersection d'ensembles ont joué un rôle important dans la performance de l'algorithme ;

4.5 Conclusion

Le routage dans les RCSF a introduit de nombreux défis par rapport au routage de données traditionnel dans les réseaux câblés. Ce chapitre présente un nouveau protocole de routage utilisant une nouvelle méthode d'optimisation basée sur la philosophie du processus d'enseignement-apprentissage. Cette approche TLBOR assure une optimisation de la consommation d'énergie, ce qui prolonge la durée de vie du réseau. En effectuant des expériences dans les mêmes conditions de simulation, le protocole TLBOR est comparé à plusieurs protocoles de routage dans les RCSF tels que les approches basées sur ACO, PSO et IHSBEER et le protocole AODV. Généralement, les résultats montrent que notre protocole TLBOR est meilleur en termes de consommation d'énergie et de durée de vie du réseau.

Après la validation théorique, nous avons passé à la validation pratique. Une application de surveillance a été mise en œuvre sur la plateforme Arduino au sein de laboratoire

LRIT. Cette phase d'implémentation est encours et le travail est avancement. Nous avons mis en Annexe des détails sur l'environnement de travail et état d'avancement.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Synthèse

Lors de cette thèse, nous avons effectué une étude approfondie et critique sur les réseaux de capteurs et leurs principaux problèmes d'optimisation : de déploiement, de couverture, de topologie, d'ordonnancement et de routage. Parmi ces problèmes, nous sommes pleinement intéressés par les problèmes de routage et comment les résoudre à l'aide des métaheuristiques inspirées de la nature. Ainsi nous avons proposé des solutions pour offrir un meilleur pris en compte des ressources énergétiques du réseau.

La première contribution, basée sur les algorithmes de colonie de fourmis, développe une version améliorée dans le but de fournir plusieurs chemins pour la transmission efficace des données en cas des erreurs ou défaillance des nœuds, et ce afin de conserver la durée de vie du réseau de capteurs au maximum, tout en assurant une transmission efficace de données. L'idée de base de cette contribution est l'interprétation des principales composantes de l'algorithme de colonie de fourmis, ce qui a permet d'un côté de modifier l'algorithme et le réadapter au problème de routage dans les réseaux de capteur, et de l'autre côté résoudre ce problème d'une manière plus efficace.

La deuxième contribution consiste à comparer le protocole proposé qui est basé sur ACO à un autre protocole basé sur PSO. Ces comparaisons sont élaborées sur un ensemble de scénarios dans deux types de RCSF statiques et mobiles. Nous avons étudié l'impacte de la mobilité sur les deux approches (en utilisant le modèle d'évaluation random waypoint) afin d'évaluer l'influence qu'apporte la mobilité à un réseau et à quel point cela peut affecter l'efficacité de communication, de routage et de transmission d'information.

Dans une troisième contribution, nous avons proposé un nouveau protocole de routage dans les RCSF basé sur une métaheuristique inspirée de la nature : TLBO, initialement conçue pour les problèmes d'optimisation continus. Nous avons adapté cette méthode au problème de routage qui est un problème discret, et cela en effectuant quelque modification et redéfinition des équations de TLBO. L'utilisation d'opérateur ERO a ajouté une touche spéciale sur le fonctionnement du protocole, d'où la supériorité qu'a montrée notre approche TLBOR vis-à-vis des autres protocoles de la littérature en terme de consommation énergétique, fiabilité, délai de transmission et la prolongation de la durée de vie des RCSF. Les simulations, les expérimentations et les A-test ont prouvé ce propos dans plusieurs scénarios avec un changement de paramétrage et d'environnement.

A la fin de ce travail, nous avons pu élaborer une étude de cas, une application utilisant de vrais capteurs dans le but de valider notre travail et voir le comportement de notre protocole de routage dans le cas réel. Les capteurs sont à base de carte Arduino, détectant la température et l'humidité pour un système de surveillance. Une description d'environnement de travail et d'état d'avancement est présentée en Annexe.

Perspectives

Les réseaux de capteurs constituent un domaine de recherche très vaste. Ils ont de nombreuses perspectives d'application dans des domaines très variés. Il reste encore de nombreux problèmes à résoudre dans ce domaine afin de pouvoir les utiliser dans des conditions réelles. Nous proposons d'améliorer notre approche TLBOR pour qu'elle soit applicable dans plusieurs types de réseaux de capteurs sans fil ayant d'autres contraintes, telles que la mobilité des nœuds, ou l'utilisation de plus qu'un seul collecteur (Sink),...

Nous proposons également de développer notre approche et prendre la couverture en considération pour atteindre un réseau plus fiable. Il serait intéressant également de proposer un nouvel algorithme de routage basé sur l'hybridation composée de l'algorithme teaching-learning based optimization et d'autres métaheuristiques, vue la capacité d'optimisation prouvée par cette méthode. Enfin, utilisé la métaheuritique TLBO dans la résolution des autres problèmes d'optimisation dans les RCSF ainsi étendre le champ d'utilité de cette technique.

Liste des publications

Revues internationales

Asmae EL GHAZI, Belaïd Ahiod, « Energy Efficient Teaching-Learning-Based Optimization for the Discrete Routing Problem in Wireless Sensor Networks », ***Applied Intelligence***, Springer, pp 1-15, 2017.

Asmae EL GHAZI, Belaïd Ahiod and Mohammed Abbad, « TLBO-Based Routing Approach for Wireless Sensor Networks », ***International Journal of Intelligent Engineering and Systems***, pp 94-103, 2018.

Asmae EL GHAZI, Belaïd Ahiod and Mohammed Abbad, « Routing in Wireless Sensor Networks using Ant Colony Optimization : A Brief Survey », ***en cours***.

Conférences internationales

Asmae EL GHAZI, Belaïd Ahiod, « Improved ant colony optimization routing protocol for wireless sensor networks », ***Networked Systems (NETYS)***, Springer International Publishing, pp 246–256, 2014.

Asmae EL GHAZI, Belaïd Ahiod, « Particle swarm optimization compared to ant colony optimization for routing in wireless sensor networks », ***Proceedings of the Mediterranean Conference on Information & Communication Technologies (MedICT)***, Springer International Publishing, pp 221–227, 2015.

Asmae EL GHAZI, Belaïd Ahiod, « Impact of random waypoint mobility model on PSO based routing protocol for wireless sensor networks », ***The International conference Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)***, IEEE, 2016.

Asmae EL GHAZI, Belaïd Ahiod, « Impact of random waypoint mobility model on ant-based routing protocol for wireless sensor networks », ***The International Conference on Big Data and Advanced Wireless Technologies (BDAW)***, ACM, 2016.

Asmae EL GHAZI, Zineb AARAB, Belaïd Ahiod « Context-Aware Routing Protocol based on PSO for Mobile WSN », ***The 3rd International Conference***

on Cloud Computing Technologies and Applications (CloudTech), IEEE, 2017.

Bibliographie

Abidin, Z., Ngah, U., Arshad, M. et Ping, O. (2010). A novel fly optimization algorithm for swarming application. *In Robotics Automation and Mechatronics (RAM)*, pages 425–428. IEEE.

Akkaya, K. et Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349.

Akyildiz, I. F., Melodia, T. et Chowdhury, K. R. (2007). A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960.

Akyildiz, I. F., Pompili, D. et Melodia, T. (2004). Challenges for efficient communication in underwater acoustic sensor networks. *ACM Sigbed Review*, 1(2):3–8.

Akyildiz, I. F. et Stuntebeck, E. P. (2006). Wireless underground sensor networks : Research challenges. *Ad Hoc Networks*, 4(6):669–686.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. et Cayirci, E. (2002a). A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. et Cayirci, E. (2002b). Wireless sensor networks : a survey. *Computer networks*, 38(4):393–422.

Al-Karaki, J. N. et Kamal, A. E. (2004). Routing techniques in wireless sensor networks : a survey. *Wireless communications*, 11(6):6–28.

Andersen, T. et Tirthapura, S. (2009). Wireless sensor deployment for 3d coverage with constraints. *In Networked Sensing Systems (INSS), 2009 Sixth International Conference on*, pages 1–4. IEEE.

Arun, C. et Sudha, K. L. (2012). Agricultural management using wireless sensor networks-a survey. *In 2nd International Conference on Environment Science and Biotechnology (IPCBEE), Singapore.*

Bai, X., Kumar, S., Xuan, D., Yun, Z. et Lai, T. H. (2006). Deploying wireless sensors to achieve both coverage and connectivity. *In Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 131–142. ACM.

Basturk, B. et Karaboga, D. (2006). An artificial bee colony (abc) algorithm for numeric function optimization. *In IEEE swarm intelligence symposium*, pages 12–14.

Bayrakli, S. et Erdogan, S. Z. (2012). Genetic algorithm based energy efficient clusters (gabeec) in wireless sensor networks. *Procedia Computer Science*, 10:247–254.

Bettstetter, C., Hartenstein, H. et Pérez-Costa, X. (2004). Stochastic properties of the random waypoint mobility model. *Wireless Networks*, 10(5):555–567.

Bhatti, G. K. et Raina, J. P. S. (2014). Cuckoo based energy effective routing in wireless sensor network. *International Journal of Computer Science and Communication Engineering*, 3(1):92–95.

Blum, C. et Roli, A. (2003a). Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

Blum, C. et Roli, A. (2003b). Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35:268–308.

Bouabdellah, K., Noureddine, H. et Larbi, S. (2013). Using wireless sensor networks for reliable forest fires detection. *Procedia Computer Science*, 19:794–801.

Camilo, T., Carreto, C., Silva, J. et Boavida, F. (2006). An energy-efficient ant-based routing algorithm for wireless sensor networks. *Ant Colony Optimization and Swarm Intelligence*, pages 49–59.

Campo, E., Estève, D. et Chan, M. (2012). Conception d'un habitat adapté pour l'aide à l'autonomie des personnes âgées. *Les cahiers de l'année gérontologique*, pages 1–8.

Chen, D. et Varshney, P. K. (2004). Qos support in wireless sensor networks : A survey. *In International conference on wireless networks*, volume 233, pages 1–7.

Chen, G., Guo, T.-D., Yang, W.-G. et Zhao, T. (2006). An improved ant-based routing protocol in wireless sensor networks. *In Collaborative Computing : Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, pages 1–7. IEEE.

Cheng, X., Narahari, B., Simha, R., Cheng, M. X. et Liu, D. (2003). Strong minimum energy topology in wireless sensor networks : Np-completeness and heuristics. *IEEE transactions on mobile computing*, 2(3):248–256.

Chong, C. et Kumar, S. P. (2003a). Sensor networks : Evolution, opportunities, and challenges. *Proceedings of the IEEE, Vol. 91, No 8.*

Chong, C.-Y. et Kumar, S. P. (2003b). Sensor networks : evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

Colorni, A., Dorigo, M. et Maniezzo, V. (1991). Distributed optimization by ant colonies. *In Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France.

Darwin, C. (1998). The origin of the species.

Dhawan, A. et Prasad, S. K. (2009). A distributed algorithmic framework for coverage problems in wireless sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 24(4):331–348.

Dhivya, M., Sundarambal, M. et Anand, L. N. (2011a). Energy efficient computation of data fusion in wireless sensor networks using cuckoo based particle approach (cbpa). *International Journal of Communications, Network and System Sciences*, 4(04):249.

Dhivya, M., Sundarambal, M. et Vincent, J. O. (2011b). Energy efficient cluster formation in wireless sensor networks using cuckoo search. *In International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 140–147. Springer.

DHT22 (2014). *DHT22 Datasheet*. distribuée par l'intégrateur Sparkfun.

Dorigo, M. et Birattari, M. (2007). Swarm intelligence. *Scholarpedia*, 2.

Dorigo, M. et Birattari, M. (2010). Ant colony optimization. *In Encyclopedia of Machine Learning*, pages 36–39. Springer.

Dorigo, M., Bonabeau, E. et Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16:851–871.

Dorigo, M. et Di Caro, G. (1999). Ant colony optimization : a new metaheuristic. *In Proceedings of the 1999 Congress on Evolutionary Computation CEC 99*, pages 1–8. IEEE.

Dorigo, M., Gambardella, L. M. *et al.* (1997). Ant colonies for the travelling salesman problem. *BioSystems*, 43(2):73–82.

Douglas, S. M., Bachelet, I. et Church, G. M. (2012). A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070):831–834.

Eberhart, R. C., Kennedy, J. *et al.* (1995). A new optimizer using particle swarm theory. *In Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY.

EL Ghazi, A. et Ahiod, B. (2016). Particle swarm optimization compared to ant colony optimization for routing in wireless sensor networks. *In Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*, pages 221–227. Springer.

El Ghazi, A. et Ahiod, B. (2016). Random waypoint impact on bio-inspired routing protocols in wsn. *In Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2016 7th International Conference on*, pages 326–331. IEEE.

El Ghazi, A. et Ahiod, B. (2017). Energy efficient teaching-learning-based optimization for the discrete routing problem in wireless sensor networks. *Applied Intelligence*, 47(3):585–606.

El Ghazi, A., Ahiod, B. et Ouaarab, A. (2014). Improved ant colony optimization routing protocol for wireless sensor networks. *In Networked Systems*, pages 246–256. Springer.

Esnault, A. (2012). *Energy-Aware Distributed Ant Colony Based Virtual Machine Consolidation in IaaS Clouds*. Thèse de doctorat, INRIA-IRISA Rennes Bretagne Atlantique, équipe Myriads.

Fathima, K. et Sindhanaiselvan, K. (2013). Ant colony optimization based routing in wireless sensor networks. *International Journal of Advanced Networking & Applications*, 4(4).

Gandomi, A. H., Talatahari, S., Yang, X.-S. et Deb, S. (2012). Design optimization of truss structures using cuckoo search algorithm. *The Structural Design of Tall and Special Buildings*.

Gandomi, A. H., Yang, X.-S. et Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, 89(23-24):2325 – 2336.

Gandomi, A. H., Yang, X.-S. et Alavi, A. H. (2013). Cuckoo search algorithm : a metaheuristic approach to solve structural optimization problems. *Engineering with Computers*, 29(1):17–35.

Garcia, M., Catalá, A., Lloret, J. et Rodrigues, J. J. (2011). A wireless sensor network for soccer team monitoring. *In Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–6. IEEE.

Ge, F., Wang, Y., Wang, Q. et Kang, J. (2007). Energy efficient broadcasting based on ant colony system optimization algorithm in wireless sensor networks. *In Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 129–133. IEEE.

GhasemAghaei, R., Rahman, M. A., Gueaieb, W. et El Saddik, A. (2007). Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. *In Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pages 1–6. IEEE.

Ghasemi, M., Taghizadeh, M., Ghavidel, S., Aghaei, J. et Abbasian, A. (2015). Solving optimal reactive power dispatch problem using a novel teaching–learning-based optimization algorithm. *Engineering Applications of Artificial Intelligence*, 39:100–108.

Ghedira, K. (2007). *Optimisation combinatoire par métaheuristiques*.

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

Glover, F. et Kochenberger, G. A. (2003). *Handbook of metaheuristics*. Springer.

Gogu, A., Nace, D., Dilo, A. et Mertnia, N. (2011). Optimization problems in wireless sensor networks. *In Complex, Intelligent and Software Intensive Systems (CISIS), International Conference on*, pages 302–309. IEEE.

Gosnell, T., Hall, J., Jam, C., Knapp, D., Koenig, Z., Luke, S., Pohl, B., Schach von Wittenau, A. et Wolford, J. (1997). Gamma-ray identification of nuclear weapon materials. Rapport technique, Lawrence Livermore National Lab., Livermore, CA (US).

Guan, X., Guan, L., Wang, X. G. et Ohtsuki, T. (2009). A novel routing algorithm based on ant colony system for wireless sensor networks. *In Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–5. IEEE.

Gupta, H., Zhou, Z., Das, S. R. et Gu, Q. (2006). Connected sensor cover : Self-organization of sensor networks for efficient query execution. *IEEE/ACM Transactions on Networking (ToN)*, 14(1):55–67.

Guru, S., Halgamuge, S. et Fernando, S. (2005). Particle swarm optimisers for cluster formation in wireless sensor networks. *In Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on*, pages 319–324. IEEE.

Han, X., Cao, X., Lloyd, E. L. et Shen, C.-C. (2010). Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656.

Heidemann, J., Li, Y., Syed, A., Wills, J. et Ye, W. (2005). Underwater sensor networking : Research challenges and potential applications. *Proceedings of the Technical Report ISI-TR-2005-603, USC/Information Sciences Institute*.

Heinzelman, W. R., Chandrakasan, A. et Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. *In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 1–10. IEEE.

Holland, J. (1975). Adaptation in natural and artificial systems.

Hussain, S., Matin, A. W. et Islam, O. (2007). Genetic algorithm for energy efficient clusters in wireless sensor networks. *In ITNG*, pages 147–154.

IRIS (2011). *IRIS Datasheet*. distribuée par l'intégrateur MEMSIC.

Karaboga, D. et Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization : artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471.

Karl, H. et Willig, A. (2007). *Protocols and architectures for wireless sensor networks*. John Wiley & Sons.

Kavian, Y. S., Rashedi, A., Mahani, A. et Ghassemlooy, Z. (2013). Routing and wavelength assignment in optical networks using artificial bee colony algorithm. *Optik-International Journal for Light and Electron Optics*, 124(12):1243–1249.

Ke, W.-C., Liu, B.-H. et Tsai, M.-J. (2011). The critical-square-grid coverage problem in wireless sensor networks is np-complete. *Computer Networks*, 55(9): 2209–2220.

Kennedy, J. et Eberhart, R. (1995). Particle swarm optimization. *In Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE.

Kuila, P. et Jana, P. K. (2014). Energy efficient clustering and routing algorithms for wireless sensor networks : Particle swarm optimization approach. *Engineering Applications of Artificial Intelligence*, 33:127–140.

Latiff, N. A., Tsimenidis, C. C. et Sharif, B. S. (2007). Energy-aware clustering for wireless sensor networks using particle swarm optimization. *In Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pages 1–5. IEEE.

Li, M. et Liu, Y. (2007). Underground structure monitoring with wireless sensor networks. *In Proceedings of the 6th international conference on Information processing in sensor networks*, pages 69–78. ACM.

Lin, M.-S., Leu, J.-S., Yu, W.-C., Li, K.-H. et Wu, J.-L. C. (2012). Tbra : Termites based routing algorithm in 3d wireless sensor networks. *In Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, pages 1–5. IEEE.

Liu, C., Wu, K., Xiao, Y. et Sun, B. (2006). Random coverage with guaranteed connectivity : joint scheduling for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(6):562–575.

Lu, Y., Zhao, G. et Su, F. (2004). Adaptive ant-based dynamic routing algorithm. *In Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 3, pages 2694–2697. IEEE.

Misra, P. et Enge, P. (2006). Global positioning system : Signals, measurements and performance second edition. *Massachusetts : Ganga-Jamuna Press*.

MRF24J40 (2010). *MRF24J40*. distribuée par l'intégrateur Microchip Technology Inc.

MTS400 (2011). *MTS400 Datasheet*. distribuée par l'intégrateur MEMSIC.

Mucherino, A. et Seref, O. (2007). Monkey search : a novel metaheuristic search for global optimization. *In Data Mining, Systems Analysis, and Optimization in Biomedicine(AIP Conference Proceedings Volume 953)*, volume 953, pages 162–173. American Institute of Physics, 2 Huntington Quadrangle, Suite 1 NO 1, Melville, NY, 11747-4502, USA.

Naik, A., Parvathi, K., Satapathy, S. C., Nayak, R. et Panda, B. (2013). Qos multicast routing using teaching learning based optimization. *In Proceedings of International Conference on Advances in Computing*, pages 49–55. Springer.

Nakrani, S. et Tovey, C. (2004). On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior*, 12(3-4):223–240.

Nekoogar, F., Dowla, F. et Spiridon, A. (2004). Self organization of wireless sensor networks using ultra-wideband radios. *In Radio and Wireless Conference, 2004 IEEE*, pages 451–454. IEEE.

Neshat, M., Sepidnam, G., Sargolzaei, M. et Toosi, A. N. (2014). Artificial fish swarm algorithm : a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, pages 1–33.

Nimbalkar, N. B., Das, S. S. et Wagh, S. J. (2015). Trust based energy efficient clustering using genetic algorithm in wireless sensor networks (teecga). *International Journal of Computer Applications*, 112(9).

Okdem, S. et Karaboga, D. (2009). Routing in wireless sensor networks using an ant colony optimization (aco) router chip. *Sensors*, 9:909–921.

Okdem, S., Karaboga, D. et Ozturk, C. (2011). An application of wireless sensor network routing based on artificial bee colony algorithm. *In Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 326–330. IEEE.

Perkins, C., Belding-Royer, E. et Das, S. (2003). Ad hoc on-demand distance vector (aodv) routing. Rapport technique.

Pister, K. S. (1999). Smart dust : Wireless networks of millimeter-scale sensor nodes. *Highlight Article in 1999 Electronics Research Laboratory Research Summary*.

Pompili, D., Melodia, T. et Akyildiz, I. F. (2006). Deployment analysis in underwater acoustic wireless sensor networks. *In Proceedings of the 1st ACM international workshop on Underwater networks*, pages 48–55. ACM.

Quijano, N. et Passino, K. M. (2010). Honey bee social foraging algorithms for resource allocation : Theory and application. *Engineering Applications of Artificial Intelligence*, 23(6):845–861.

Raj, D. A. A. et Sumathi, P. (2014). Enhanced energy efficient multipath routing protocol for wireless sensor communication networks using cuckoo search algorithm. *Wireless Sensor Network*, 6(04):49.

Rao, D. S. et Kumar, B. R. (2011). Performance evaluation of genetic based dynamic clustering algorithm over leach algorithm for wireless sensor networks. *Network*, 100:100m.

Rao, R. V., Savsani, V. J. et Vakharia, D. (2011). Teaching–learning-based optimization : a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315.

Rao, R. V., Savsani, V. J. et Vakharia, D. (2012). Teaching–learning-based optimization : an optimization method for continuous non-linear large scale problems. *Information Sciences*, 183(1):1–15.

Rebai, M., Snoussi, H., Khoukhi, L. et Hnaien, F. (2013). Linear models for the total coverage problem in wireless sensor networks. *In Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on*, pages 1–4. IEEE.

Ru, H. et Xu, G. (2010). Swarm intelligence-inspired adaptive routing construction in wsn. *In Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1–5. IEEE.

Saleem, K., Fisal, N., Baharudin, M. A., Ahmed, A. A., Hafizah, S. et Kamilah, S. (2010). Ant colony inspired self-optimized routing protocol based on cross layer architecture for wireless sensor networks. *WSEAS Transactions on Communications*, 9(10):669–678.

Salehpour, A.-A., Mirmobin, B., Afzali-Kusha, A. et Mohammadi, S. (2008). An energy efficient routing protocol for cluster-based wireless sensor networks using ant colony optimization. *In Innovations in Information Technology, 2008. IIT 2008. International Conference on*, pages 455–459. IEEE.

Santi, P. (2012). *Mobility models for next generation wireless networks : ad hoc, vehicular and mesh networks*. John Wiley & Sons.

Shakkottai, S., Srikant, R. et Shroff, N. B. (2005). Unreliable sensor grids : Coverage, connectivity and diameter. *Ad Hoc Networks*, 3(6):702–716.

Sharma, K., Monga, H. *et al.* (2013). Improved termite hill routing protocol using aco wsn. *In Computer Science and Engineering Conference (ICSEC), 2013 International*, pages 365–370. IEEE.

Shi, X., Liang, Y., Lee, H., Lu, C. et Wang, Q. (2007). Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information Processing Letters*, 103(5):169–176.

Shurman, M. M., Al-Mistarihi, M. F., Mohammad, A. N., Darabkh, K. A. et Ababnah, A. A. (2013). Hierarchical clustering using genetic algorithm in wireless sensor networks. *In Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*, pages 479–483. IEEE.

Singh, G., Das, S., Gosavi, S. V. et Pujar, S. (2005). Ant colony algorithms for steiner trees : An application to routing in. *Recent developments in biologically inspired computing*, page 181.

Sosedka, J. et Shostko, I. (2014). Calculation method for power consumption and lifetime of nodes in wsn. *In Problems of Infocommunications Science and Technology, 2014 First International Scientific-Practical Conference*, pages 116–117. IEEE.

Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons.

Tian, D. et Georganas, N. D. (2005). Connectivity maintenance and coverage preservation in wireless sensor networks. *Ad Hoc Networks*, 3(6):744–761.

Toumpis, S. et Tassiulas, L. (2006). Optimal deployment of large wireless sensor networks. *IEEE Transactions on Information Theory*, 52(7):2935–2953.

Vargha, A. et Delaney, H. D. (2000). A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.

Wang, K.-P., Huang, L., Zhou, C.-G. et Pang, W. (2003). Particle swarm optimization for traveling salesman problem. *In Machine Learning and Cybernetics, 2003 International Conference on*, volume 3, pages 1583–1585. IEEE.

Wang, X., Wan, W., Zhang, X. et Yu, X. (2010). Annealed particle filter based on particle swarm optimization for articulated three-dimensional human motion tracking. *Optical Engineering*, 49(1):017204–017204–11.

Wang, X. et Wang, S. (2011). Hierarchical deployment optimization for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 10(7):1028–1041.

Whitley, L. D., Starkweather, T. et Fuquay, D. (1989). Scheduling problems and traveling salesmen : The genetic edge recombination operator. *In ICGA*, volume 89, pages 133–40.

Wieselthier, J. E., Nguyen, G. D. et Ephremides, A. (2000). On the construction of energy-efficient broadcast and multicast trees in wireless networks. *In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 585–594. IEEE.

Yang, X.-S. (2010a). *Engineering Optimization : An Introduction with Metaheuristic Applications*. Wiley, USA.

Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74.

Yang, X.-S. (2012). *Swarm-based metaheuristic algorithms and no-free-lunch theorems*. INTECH Open Access Publisher.

Yang, X.-S., Cui, Z., Xiao, R., Gandomi, A. H. et Karamanoglu, M. (2013). *Swarm intelligence and bio-inspired computation : theory and applications*. Newnes.

Yang, X. S. et Deb, S. (2009). Cuckoo search via lévy flights. *In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE.

Yang, X.-S. et Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 40:1616–1624.

Yiyue, W., Hongmei, L. et Hengyang, H. (2012). Wireless sensor network deployment using an optimized artificial fish swarm algorithm. *In Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 2, pages 90–94. IEEE.

Yuanyuan, Z., Jia, X. et Yanxiang, H. (2006). Energy efficient distributed connected dominating sets construction in wireless sensor networks. *In Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 797–802. ACM.

Zeng, B. et Dong, Y. (2016). An improved harmony search based energy-efficient routing algorithm for wireless sensor networks. *Applied Soft Computing*, 41:135–147.

Zhang, X.-h. et Xu, W.-b. (2006). Qos based routing in wireless sensor network with particle swarm optimization. *In Agent Computing and Multi-Agent Systems*, pages 602–607. Springer.

Zhang, Y., Kuhn, L. D. et Fromherz, M. P. (2004). Improvements on ant routing for sensor networks. *Lecture notes in computer science*, 3172:154–165.

Zhao, Q. et Gurusamy, M. (2008). Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(6):1378–1391.

Zheng, W. et Luo, D. (2014). Routing in wireless sensor network using artificial bee colony algorithm. *In Wireless Communication and Sensor Network (WCSN), 2014 International Conference on*, pages 280–284. IEEE.

Zhou, Z., Das, S. et Gupta, H. (2004). Connected k-coverage problem in sensor networks. *In Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 373–378. IEEE.

Zhu, C., Zheng, C., Shu, L. et Han, G. (2012). A survey on coverage and connectivity issues in wireless sensor networks. *Journal of Network and Computer Applications*, 35(2):619–632.

Ziyadi, M., Yasami, K. et Abolhassani, B. (2009). Adaptive clustering for energy efficient wireless sensor networks based on ant colony optimization. *In Communication Networks and Services Research Conference, 2009. CNSR'09. Seventh Annual*, pages 330–334. IEEE.

Zungeru, A. M., Ang, L.-M. et Seng, K. P. (2012a). Performance of termite-hill routing algorithm on sink mobility in wireless sensor networks. *In International Conference in Swarm Intelligence*, pages 334–343. Springer.

Zungeru, A. M., Ang, L.-M. et Seng, K. P. (2012b). Termite-hill : Performance optimized swarm intelligence based routing algorithm for wireless sensor networks. *Journal of Network and Computer Applications*, 35(6):1901–1917.

ANNEXE

Introduction

Ce chapitre présente une étude de l'application de gestion d'un réseau de capteurs sans fil que nous avons réalisé. Dans cette partie nous allons étudier tour à tour les spécificités de cette application, son comportement ainsi que l'ensemble des étapes de l'implémentation. Pour atteindre nos objectifs nous avons réalisé l'intégralité du travail d'implémentation de l'application sur une plateforme matérielle.

Avec les différentes phases composant notre travail d'implémentation et qui sont :

- la recherche des différentes caractéristiques du matériel ;
- la compréhension des outils de développement, le codage et le débogage ;
- la réalisation des différentes tâches de notre application afin d'obtenir une solution comportant l'ensemble des fonctionnalités souhaitées ;
- l'expérimentation de la solution ;
- l'exploitation des résultats obtenus et l'évaluation de notre application, plus particulièrement en terme de consommation d'énergie.

Les objectifs de l'application étudiée sont découpés en deux parties, dont le but de la première partie est de construire un réseau basé sur la topologie plate qui comporte un ensemble de nœuds capteurs connectés entre eux. Ce réseau doit être géré de manière à ce que ses entités soient organisées de manière automatique où chaque nœud doit connaître ses voisins. Les données transitent dans le réseau depuis le nœuds source jusqu'à le sink. La deuxième partie consiste à gérer les données collectées par la station de base en les exploitant dans une interface utilisateur pour une bonne visualisation et surveillance du comportement de notre réseau.

Choix d'une solution matérielle

La première étape en vue d'une implémentation est de s'intéresser au support physique de notre application, ainsi nous devons choisir une solution matérielle pour les nœuds de notre réseau de capteurs sans fil. Dans ce cadre, il existe sur le marché, un large éventail de matériels disponibles proposés par différents fabricants et intégrateurs (Crossbow Technology, Memsic, Arduino, etc), cependant notre Laboratoire LRIT dispose d'une plateforme matérielle de type Memsic et d'une autre d'Arduino.

L'objectif ici n'est pas d'en faire une liste exhaustive mais de pouvoir choisir une des solutions les plus adéquates pour notre application tout en tenant compte de nos contraintes (temps de livraison, compatibilité logicielle, etc). Le tableau 4.3 résume les caractéristiques des solutions envisagées.

	Memsic (IRIS & MTS400)	**Arduino (Mrf24j40MA & DHT22)**
Microcontrôleur et fréquence	Atmel ATMega 1281	Atmel SAM3X8E ARM cortex M3-MCU
Fréquence du microcontrôleur	2.4 GHz	2.4 GHz
Débit du module radio	250 Kb/s	250 Kb/s
Portée radio Indoor	50m	mrf24j40MA : 38m
Portée radio Outdoor	300m	mrf24j40MA : 120m
Compatibilité 802.15.4	Oui	Oui
Mémoire flash	512 Ko	512 Ko
Mémoire Vive	128 Ko (RAM)	96 Ko (SRAM)
Mémoire EEPROM	4 Ko	-
Intervalle de capture	**Temp** : -10≈60 Celsius **Hum** : 0-90%	**Temp** : -40≈80 Celsius **Hum** : 0-100%

Tableau 4.3 – Caractéristiques des plateformes de capteurs sans fil existantes dans LRIT IRIS (2011)MTS400 (2011)MRF24J40 (2010)DHT22 (2014)

- Les fréquences des microcontrôleurs sont exprimées en Méga Hertz ;
- La capacité des différentes mémoires est exprimée en Kilo Octet ;
- La mémoire EEPROM (Electrically-Erasable Programmable Read-Only Memory) est une mémoire ré-inscriptible dans une certaine limite et destinée à conserver le code exécutable, du nœud lors de sa mise hors tension ;
- La mémoire flash est une variante de la mémoire EEPROM destinée à la sauvegarde des données du nœud lors de sa mise hors tension ;
- Les mémoires RAM/SRAM (Random Access Memory & Static Random Access Memory) sont des mémoires vives utilisées pour l'exécution du code des nœuds, leurs contenu est effacé lors de la mise hors tension du nœud, SRAM est plus rapide et chère que la mémoire RAM ;
- Les débits des modules radio sont exprimés en Kilo Bits par Seconde ;

— Compatibilité 802.15.4 signifie si le module radio est compatible avec le standard 802.15.4 (Zigbee) ;

- Les portées radio sont exprimées en mètre ;
- L'intervalle de température est exprimé en Degré Celsius et celui d'humidité en pour-cent.

Après avoir pris connaissance des caractères des deux solutions matérielles existantes, la plateforme Arduino a été choisie. Cette plateforme a été originalement conçue pour fournir une solution simple, économe en énergie et capable d'intégrer différents capteurs du milieu environnant (capteurs de température, de lumière, d'hu-

midité, etc) contrairement à la plateforme Memsic qui ne peut pas être modifiable et d'une autre part la plateforme memsic est commercialisée déjà avec une interface utilisateur simplifiée, contrairement à Arduino, de ce fait nous avons décidé de créer une pour notre plateforme Arduino afin de faciliter la tâche aux membres de notre laboratoire qui souhaitent travailler sur cette plateforme dans le futur.

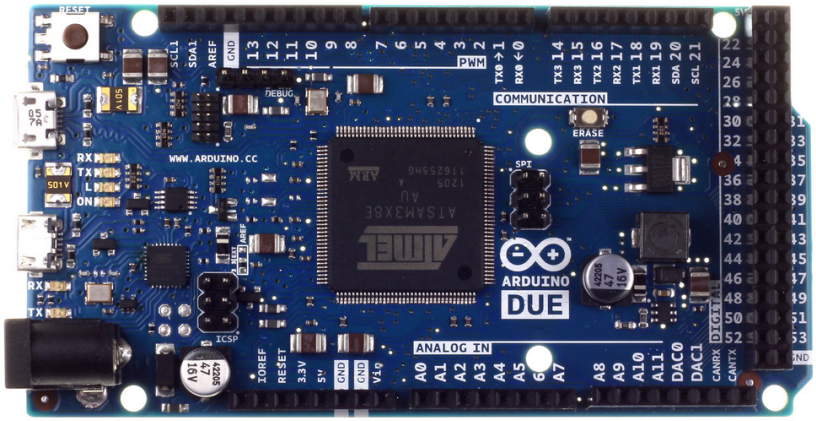


Figure 4.11 – Structure d'une carte Arduino Due

Les figures 4.11 4.12 suivante présente l'architecture générale de la solution Arduino.

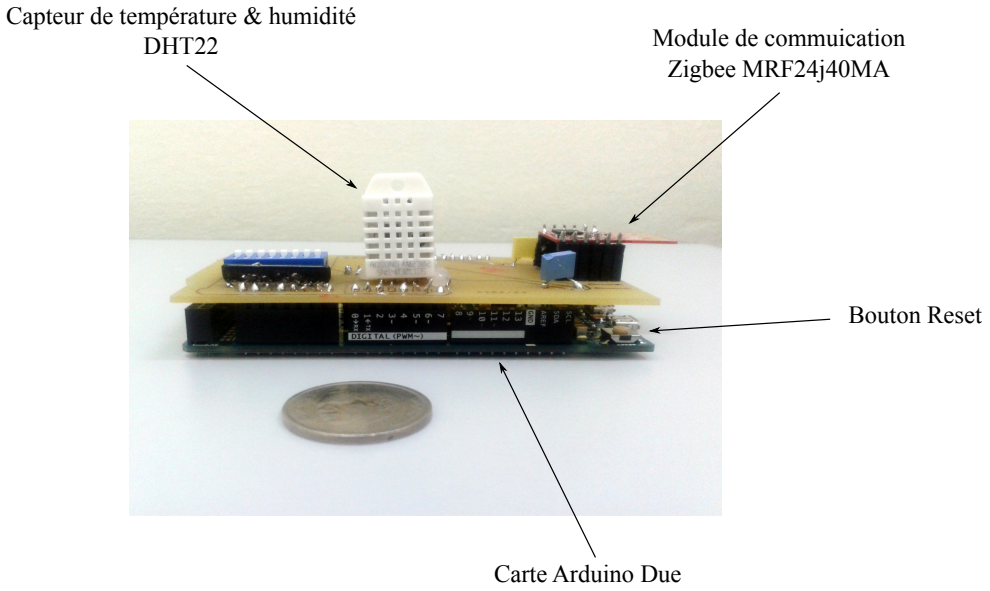


Figure 4.12 – Structure d'un nœud capteur ARDUINO

Environnement logiciel

Cette section du chapitre décrit les logiciels qui vont nous permettre d'exploiter la solution matérielle que nous avons choisi : la plateforme de capteurs Arduino. ainsi que son interfaçage avec l'utilisateur.

Le logiciel de programmation des modules Arduino est une application Java, libre et multi-plateforme, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (USB dans notre cas). Dans Arduino le langage de programmation utilisé est le C++, il lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties.

L'application Arduino nous permet de créer et éditer un programme (appelé sketch) qui sera compilé puis téléversé sur la carte Arduino. Ainsi, lorsque nous apportons des changements sur le code, ces changements ne seront effectifs qu'une fois le programme est téléversé sur la carte.

L'interface du logiciel Arduino est simple en utilisation, elle comporte les éléments suivants :

1. un menu permettant de créer de nouveaux sketchs, de les sauvegarder ,et de gérer les préférences du logiciel et les paramètres de communication avec votre carte Arduino ;
2. une barre d'actions qui sert à enregistrer le programme, le compiler et le téléverser sur la carte ;

3. le moniteur série qui affiche l'information qui est envoyée par la carte Arduino vers l'application (par le câble USB) et vis versa ;
4. un ou plusieurs onglets correspondant aux sketchs ;
5. une fenêtre de programmation ou le code est écrit ;
6. une console qui affiche les informations et erreurs de compilation et de téléversement du programme.

Généralement, un programme Arduino est composé de trois parties fondamentales :

— La partie déclaration des variables, des constantes et appel aux bibliothèques (optionnelle) ;
— La partie initialisation et configuration des entrées/sorties : la fonction setup() ;
— La partie principale qui s'exécute en boucle (programmation des interactions et comportements) : la fonction loop() ;

Pour faire le monitoring de notre réseau via l'application que nous avons réalisé, il faut tout d'abord extraire les données provenant de la station de base depuis le port série. Pour faire cela nous avons utilisé le langage Python car il dispose d'un module nommé PySerial qui encapsule l'accès au port série.

Pour la conception de l'interface de notre tableau de bord, nous avons opté pour les langages PHP, HTML5 et css3, ces derniers sont des nouveaux standards qui offrent aux utilisateurs un large éventail de fonctionnalités, une structuration beaucoup plus propre ainsi qu'ils sont compatibles sur les terminaux mobile.

Nous avons également utilisé le langage JavaScript pour que notre application s'interacte en temps réel où plus spécifiquement nous avons utilisé les bibliothèques suivantes :

— **MorrisChart** : Pour les graphes de température, d'humidité et de voltage ;
— **SigmaJs** : Pour afficher la topologie de notre réseau.

Implémentation de l'application

Comme nous l'avons déjà expliqué, notre implémentation se compose de deux parties, la première réside dans la structuration du réseau et la seconde consiste à la réalisation d'un tableau de bord pour surveiller notre réseau de capteurs sans fil.

Pour la première partie, nous avons développé un code source qui est constitué de trois fichiers aux rôles et objectifs bien déterminés. La figure 4.13 illustre les liaisons entre les différents fichiers de code.

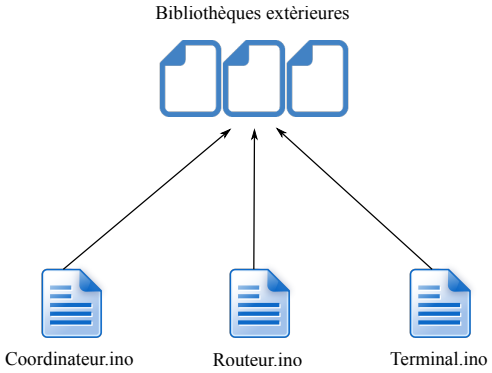


Figure 4.13 – Structure générale du code de notre réseau

— **Coordinateur.ino :** Ce fichier regroupe l'ensemble des fonctions exécutées par la station de base ;
— **Routeur.ino :** Ce fichier contient l'ensemble des instructions faites par un nœud routeur ;
— **Terminal.ino :** Ce fichier contient le code correspondant a un nœud terminal(nœud source).

Les bibliothèques extérieures sont les différentes librairies auxquelles ces trois fichiers font appel. Dans notre implémentation nous faisons appel à trois librairies :

— **SPI :** pour le transfert de données ;
— **DHT :** pour les capteurs de température et d'humidité ;
— **MRF :** pour la communication radio.

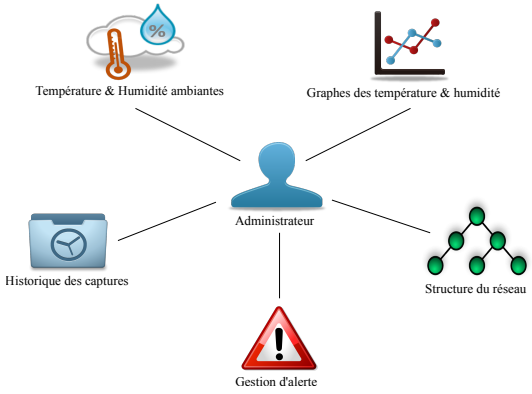


Figure 4.14 – Principales fonctionnalités de notre tableau de bord

Pour la seconde partie, nous avons conçu un ensemble de fichiers afin de répondre aux tâches déjà spécifiées. Ici la figure 4.14 présente l'ensemble des fonctionnalités de cette partie.

Déploiement et résultats de l'application

L'expérimentation de notre application de surveillance a été réalisée au sein de notre laboratoire de manipulation LRIT à la Faculté des Sciences de Rabat, où les différentes pages de l'interface sont :

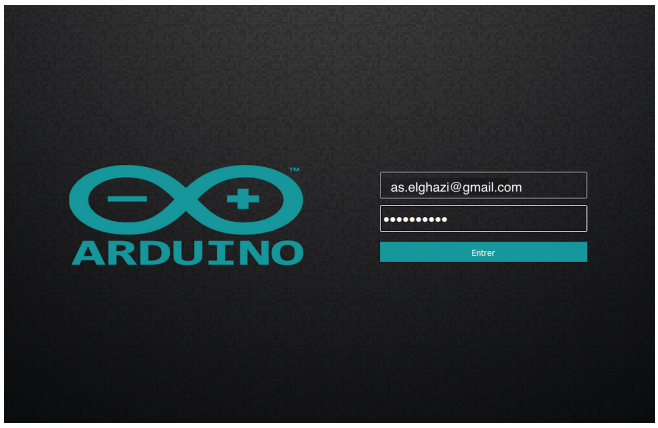


Figure 4.15 – Page d'authentification

La page de connexion : C'est la première page de notre application, l'utilisateur est amené à entrer son adresse mail ainsi que son mot de passe afin de s'authentifier dans le système.

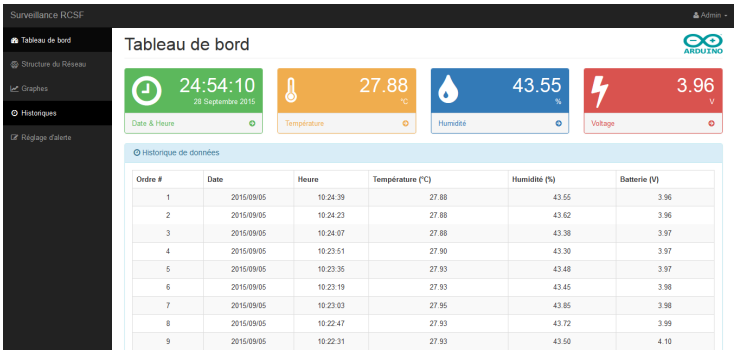


Figure 4.16 – Page d'accueil

La page d'accueil : Cette page permet de visualiser les valeurs les plus récentes de température, d'humidité et de voltage reçues depuis notre réseau. Cette interface d'accueil est également composée d'un menu en haut et d'un autre à gauche, celui du haut permet un accès à la boite émail de l'utilisateur, avec la possibilité de se déconnecter, alors que le menu à gauche sert à naviguer entre les différentes pages de notre application.

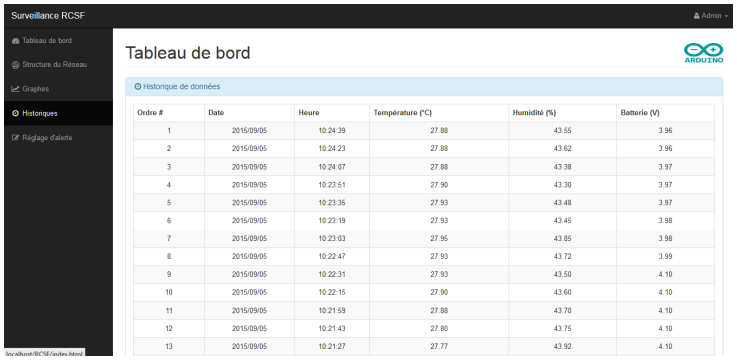


Figure 4.17 – Historique des captures

La page d'historiques : Cette page sert à lister une trace de toutes les captures que notre réseau a acquis depuis qu'il a démarré.

Figure 4.18 – Graphes des températures, humidités et voltage

La page des graphes : Cette page permet de visualiser en temps réel les variations des grandeurs physiques captées par les nœuds constituant notre réseau. Une fois qu'une nouvelle valeur est reçue elle est ajouté automatiquement au graphes flottants.

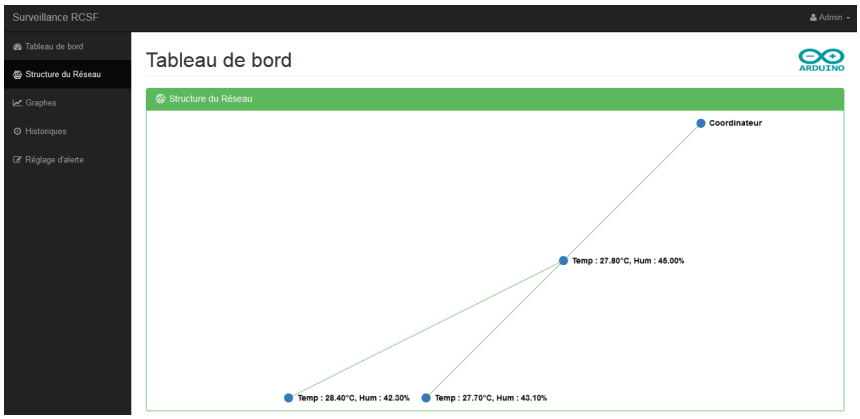


Figure 4.19 – Topologie du réseau construit

La page de topologie : cette page permet de surveiller la structure sous jacente du réseau à tout moment, y compris les valeurs captées par chaque nœud du réseau.

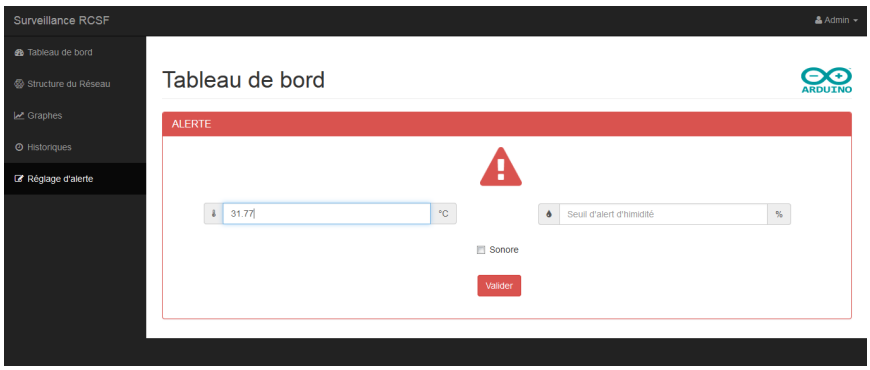


Figure 4.20 – Réglage du seuil de déclenchement d'alerte

La page d'alerte : cette page sert à régler les seuils de déclenchement d'alarme de température ainsi que d'humidité.

NB : Nous avons choisi un seuil égal à 31.77, afin de tester le déclenchement d'alerte dans notre application.

Si une valeur dépasse ou égale ces seuils est détectée, une alerte sera déclenchée en affichant un message prioritaire au sein de notre interface (voir la figure 4.21) ainsi qu'un mail est envoyé à la boite de l'administrateur(voir la figure 4.22).

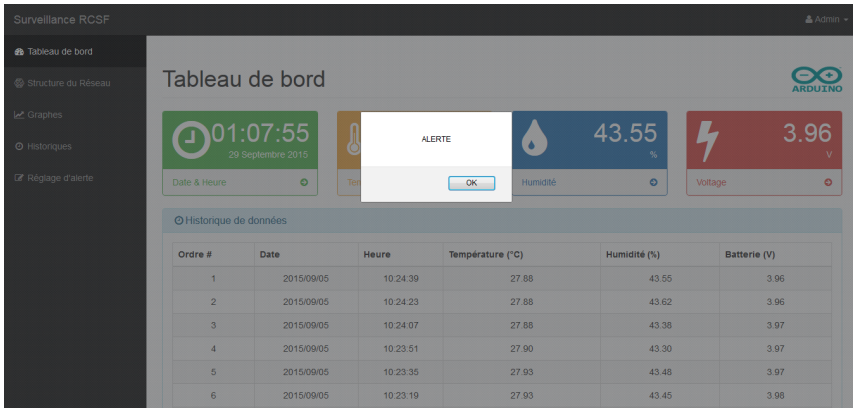


Figure 4.21 – Message d'alerte au niveau de l'interface

Figure 4.22 – Émail d'alerte

Résultats de TLBOR dans l'application

La Figure 4.23 montre les étapes utilisées pour optimiser le problème de routage dans les RCSF.

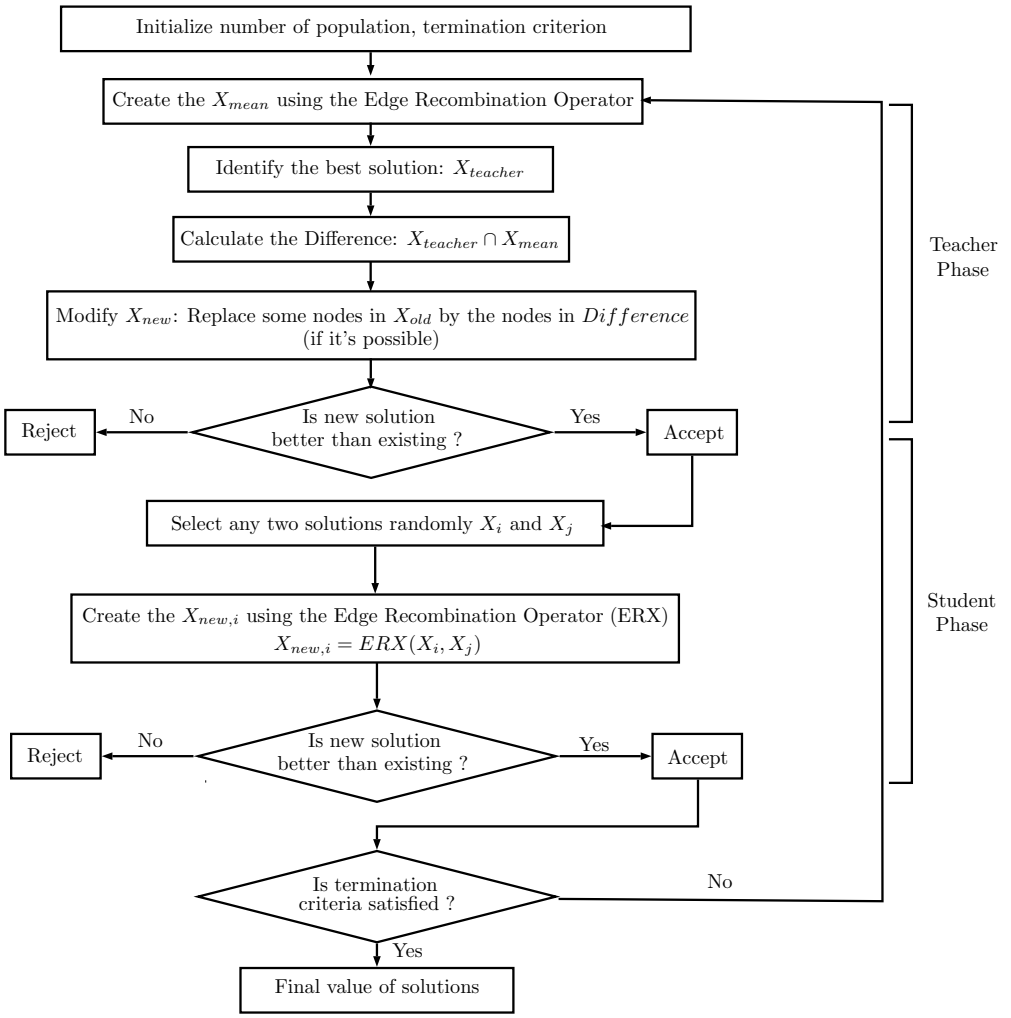


Figure 4.23 – Flowchart de l'algorithme de routage proposé (TLBOR)

L'implémentation de notre protocole de routage TLBOR est encours de réalisation. Nous avons pu programmer des fonctions et tester sur les capteurs Arduino : fonction d'initialisation de population, fonction de différence et fonction de meilleur solution. Nous sommes en phase de programmation de l'algorithme ERO 4.1, nous avons rempli les tables de voisinage et commencer la génération du chemin enfant.

```
/* Fonction d'Initialisation de Population */

#include <SPI.h>
#include <mrf24j.h>
#include <dht.h>

// Defines if the module is the base or the terminal
#define BASE true

dht DHT;

#define DHT22_PIN 7
```

```
13
15 const int pin_reset = 6;
   const int pin_cs = 52; // default CS pin on ATmega8/168/328
17 const int pin_interrupt = 5; // default interrupt pin on ATmega8/168/328

19 Mrf24j mrf(pin_reset, pin_cs, pin_interrupt);

21 //////

23 unsigned long current_time;
   unsigned long  event_time;
25 unsigned long  TTL=3000;
   int j=0;
27 String str="";
   String population[200];
29 char buf[40];


31
   void setup() {
33
     SPI.setBitOrder(MSBFIRST) ;
35   SPI.setDataMode(SPI_MODE0);
     SPI.begin();
37   Serial.begin(115200);



39


41   mrf.reset();
     mrf.init();
43


45   mrf.set_pan(0x1234);
     mrf.address16_write(602);
47   mrf.set_bufferPHY(true);
     attachInterrupt(pin_interrupt, interrupt_routine, CHANGE); // interrupt 0
       equivalent to pin 2(INT0) on ATmega8/168/328
49   interrupts();
   }
51 void interrupt_routine() {
     mrf.interrupt_handler(); }
53
   void loop() {
55          mrf.check_flags(&handle_rx, &handle_tx); }

57 void handle_rx() {
      Serial.println(" RECPETION NV PACKET  \n");
59    //to test if it is a QUESTION packet
        if (mrf.get_rxbuf()[9] == 81 ) {
61        //if timeout is elapsed
          delay(2000);
63        Serial.println(" ENVOI DU chemin  \n");
          String str="R:14/15/1/50/S";
65        str.toCharArray(buf,40);
          Serial.println(buf);
67        mrf.send16(601, buf
      }
69 }

71 void handle_tx() {
     if (mrf.get_txinfo()->tx_ok) {
73     Serial.println(" TX went ok, got ack \n");
```

```
  } else {
    Serial.print(" TX failed after \n ");
    Serial.print(mrf.get_txinfo()->retries);
    Serial.println(" retries\n");
  }
}
```

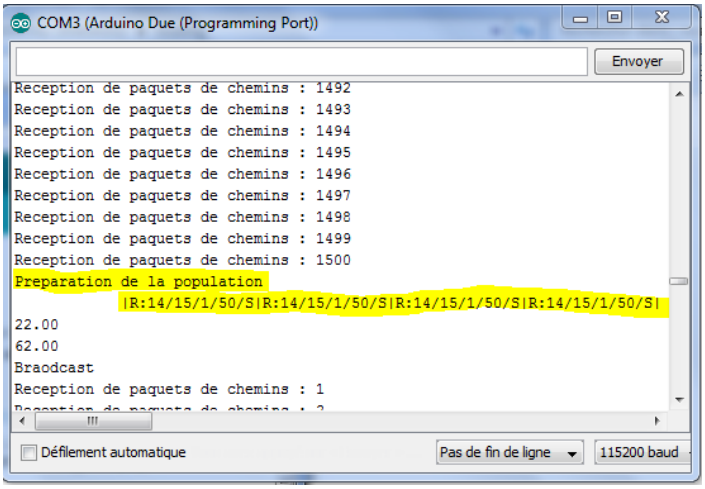


Figure 4.24 – Test d'initialisation de population.

La figure 4.24 montre les résultats de test de la fonction d'initialisation de population. Les capteurs Arduino ont communiqué entre eux les informations nécessaire ainsi, la préparation de la population a été effectué avec succès.

```
/* Fonction de Meilleure Solution */

void setup() {

  Serial.begin(115200);

}

void loop() {
 String population[10]={"S:0.5;48:0.8;46:0.4;43:0.44;61:0.46;35:0.2;Sink:0",
                        "S:0.5;48:0.8;46:0.4;25:0.12;24:0.1;18:0.45;Sink:0",
                        "S:0.5;78:0.55;40:0.33;23:0.20;21:0.28;35:0.2; Sink
    :0"};
  best_solution(population);
  delay(10000);
}

float best_solution(String Population[]) {

  float cost_tab[20];

  char charParent[60];
```

```
23   float maxim = 0;

25   int index = 0;

27   for (int i = 0 ; i < Count(Population); i++) {

29     Population[i].toCharArray(charParent, 60);

31     cost_tab[i] =  Objective_function(charParent);
       Serial.print("le cout est : ");  Serial.println(cost_tab[i]);
33

35     if (cost_tab[i] > maxim) {
         maxim = cost_tab[i];
37       index = i;
       }
39   }

41   Serial.print("le cout maximal est : ");  Serial.println(cost_tab[index]);
     Serial.print("Best solution est : ");  Serial.println(Population[index]);
43
}
45
float Objective_function(char* parent) {
47
  char* pch1;
49  String parent_node[20];
  float parent_energy[20];
51  int m = 0;
  int i = 0;
53  int j = 0;
  char delimiters[] = ":; ";
55  float cost = 0;
  float L = 0;
57  float E_min = 0;
  float E_avg = 0;
59  float E_x = 0;
  //split P1 into separated nodes
61  pch1 = strtok (parent, delimiters);
  while (pch1 != NULL) {
63
    //split imbrique ici pour extraire lenergie
65    parent_node[m] = pch1;
    m++;
67    pch1 = strtok (NULL, delimiters);

69  }

71  for (i = 0, j = 1; j < m - 1; i++, j = j + 2) {
      parent_energy[i] = parent_node[j].toFloat();
73    Serial.println(parent_energy[i]);
  }
75

77  E_x = sum(parent_energy, i);
  L = float(i);
79  E_min = minim(parent_energy, i);
  E_avg = average(parent_energy, i);
81
  Serial.print("L        ");  Serial.println(L);
83
  Serial.print("E_Min:     ");    Serial.println(E_min);
```

```
85
  Serial.print("E_avg:    ");   Serial.println(E_avg);
87
  Serial.print("E_x:    ");   Serial.println(E_x);
89
  cost =  E_x * (L / E_min) * E_avg;
91
  return cost;
93}


95
float minim(float tab[], int size) {
97  int i = 1;
  float minim = 99;
99  while (i < size) {
    minim = min(tab[i], minim);
101   i++;
  }
103 return minim;
}
105
float average(float tab[], int size) {
107 int i = 0;
  float sum = 0;
109 float avg = 0;

111 while (i < size) {
    sum +=  tab[i];
113   i++;
  }
115 avg = sum / size;
  return avg;
117}

119float sum(float tab[], int size) {
  int i = 0;
121 float sum = 0;

123 while (i < size) {
    sum +=  tab[i];
125   i++;
  }
127 return sum;
}
129
int Count(String tab[]) {
131 int i = 0;
  int count = 0;
133 while (tab[i] != NULL) {

135   count++;
    i++;
137 }

139 return count;
}
```

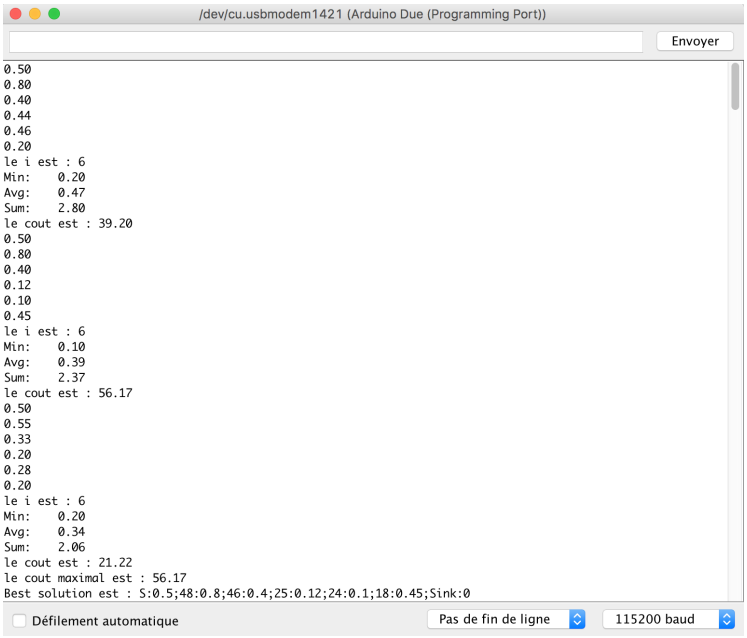


Figure 4.25 – Test de la fonction de meilleur solution.

Afin de vérifier la fonction qui évalue les chemins et sélection le meilleur, nous l'avons tester sur les capteurs Arduino. La figure 4.25 montre une capture des résultats du meilleur chemin sélectionné.

```
/* Fonction de Difference */
void setup() {

  Serial.begin(115200);
}

void loop() {
 String population[10]={"S:0.5;48:0.8;46:0.4;43:0.44;61:0.46;35:0.2;Sink:0",
                        "S:0.5;48:0.8;46:0.4;25:0.12;24:0.1;18:0.45;Sink:0",
                        "S:0.5;78:0.55;40:0.33;23:0.20;21:0.28;35:0.2; Sink
    :0"};
  Serial.print("Les elements en commun sont : ");
  Serial.println(difference(population[0], population[1]));
  delay(10000);
}
String difference (String Xteacher, String Xmean) {
  char CharXteacher[60];
  char CharXmean[60];

  //split the 1st parent
  Xteacher.toCharArray(CharXteacher, 60);
  char* pch1;
  String parent_node[20];
  String nodes[20];
  int m = 0;
  int i = 0;
  int j = 0;
  char delimiters[] = ":;";
```

```
29   //split P1 into separated nodes
     pch1 = strtok (CharXteacher, delimiters);
31   while (pch1 != NULL) {
       parent_node[m] = pch1;
33     m++;
       pch1 = strtok (NULL, delimiters);
35   }
     for (i = 0, j = 0; j < m - 1; i++, j = j + 2) {
37     nodes[i] = parent_node[j];
       // Serial.println(nodes[i]);
39   }

41   //split the 2nd parent
     Xmean.toCharArray(CharXmean, 60);
43   char* pch2;
     String parent_node1[20];
45   String nodes1[20];
     int m1 = 0;
47   int i1 = 0;
     int j1 = 0;
49
     pch2 = strtok (CharXmean, delimiters);
51   while (pch2 != NULL) {
       parent_node1[m1] = pch2;
53     m1++;
       pch2 = strtok (NULL, delimiters);
55   }
     for (i1 = 0, j1 = 0; j1 < m1 - 1; i1++, j1 = j1 + 2) {
57     nodes1[i1] = parent_node1[j1];
       // Serial.println(nodes1[i1]);
59   }

61   //La recherche des elements en commun
     i = 0;
63   String difference = "";
     while (nodes[i] != NULL) {
65
       if (find_element(nodes[i], nodes1, 20) == true) {
67
         difference.concat(nodes[i]);
69       difference.concat(" ");

71     }
       i++;
73   }
     return difference;
75 }

77 boolean find_element(String val, String tab[], int len ) {
     for (int i = 0; i < len ; i++) {
79     if (val == tab[i]) {
         return true;
81       break;
       }
83   }
   }
```

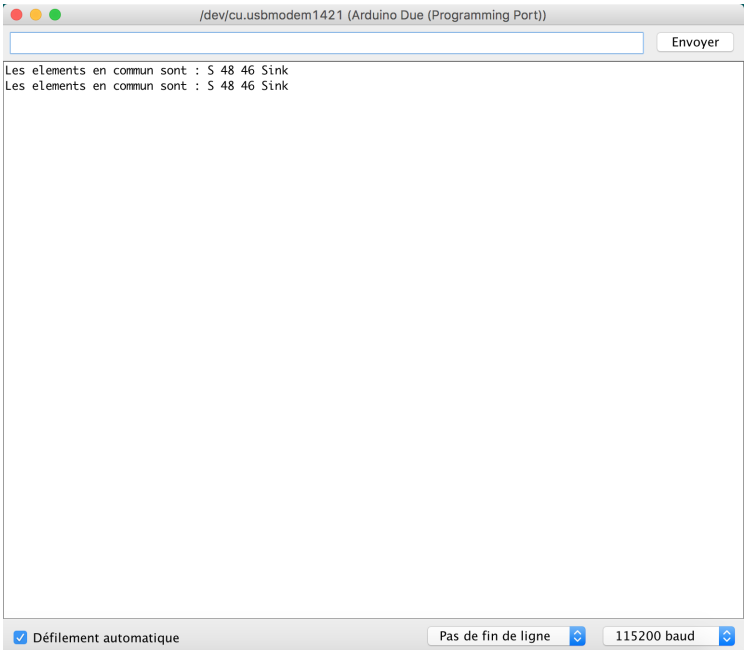


Figure 4.26 – Test de la fonction de différence.

La fonction de différence d'écrite dans le chapitre précédent a bien fonctionné sur les capteurs Arduino, la figure 4.26 montre un exemple de test.

```
  /* Fonction ERO   */
//
//  #include <SPI.h>
//  #include <mrf24j.h>
//  #include <dht.h>

  // Defines if the module is the base or the terminal
  #define BASE true

//  dht DHT;

//  #define DHT22_PIN 7
//  const int pin_reset = 6;
//  const int pin_cs = 52; // default CS pin on ATmega8/168/328
//  const int pin_interrupt = 5; // default interrupt pin on ATmega8/168/328
//
//  Mrf24j mrf(pin_reset, pin_cs, pin_interrupt);
//
  //////
  void setup() {
//    SPI.setBitOrder(MSBFIRST) ;
//    SPI.setDataMode(SPI_MODE0);
//    SPI.begin();
//
    Serial.begin(115200);
//    mrf.reset();
//    mrf.init();
//
//    mrf.set_pan(0x1234);
//    // This is _our_ address
```

```
//    mrf.address16_write(601);
//
    // uncomment if you want to receive any packet on this channel
    //mrf.set_promiscuous(true);

    //uncomment if you want to enable PA/LNA external control
   // mrf.set_palna(true);

    // uncomment if you want to buffer all PHY Payload
//    mrf.set_bufferPHY(true);
//
//    attachInterrupt(pin_interrupt, interrupt_routine, CHANGE); //
   interrupt 0 equivalent to pin 2(INT0) on ATmega8/168/328
//    interrupts();
  }
//
//  void interrupt_routine() {
//    mrf.interrupt_handler(); // mrf24 object interrupt routine
//
//  }
  void loop() {
  //
  //  data [0]='28';
  //  data [1]='/';
  //  data [2]='26';
  //  data [3]='/';
  //  data [4]='24';
  //  data [5]='/';
  //  data [6]='22';
  //  data [7]='/';
  //  data [8]='20';
  //
  //
  //    for (int i = 0; i < sizeof(data); i++) {
  //      Serial.println(data[i]);
  //    }
  //
  //Serial.println("************************************");
  //
  char* pch;
  char* temp[20];
  char data[20];
  char data1[20];

    data [0]='S';
    data [1]=':';
    data [2]='4';
    data [3]='8';
    data [4]=':';
    data [5]='4';
    data [6]='6';
    data [7]=':';
    data [8]='4';
    data [9]='3';
    data [10]=':';
    data [11]='6';
    data [12]='1';
    data [13]=':';
    data [14]='3';
    data [15]='5';
    data [16]=':';
    data [17]='i';
```

```
92      data [18]='k';

94      data1 [0]='S';
        data1 [1]=':';
96      data1 [2]='4';
        data1 [3]='8';
98      data1 [4]=':';
        data1 [5]='4';
100     data1 [6]='6';
        data1 [7]=':';
102     data1 [8]='2';
        data1 [9]='5';
104     data1 [10]=':';
        data1 [11]='2';
106     data1 [12]='4';
        data1 [13]=':';
108     data1 [14]='1';
        data1 [15]='8';
110     data1 [16]=':';
        data1 [17]='i';
112     data1 [18]='k';

114     ERO(data, data1);

116       delay(20000);

118 }

120 void ERO(char* P1, char* P2){

122   char* pch1;
      String P1_nodes[8];
124   char* pch2;
      String P2_nodes[8];
126   String Neighbouring[20][5];
      int exist=1;
128   int m=0;
      int i=0;
130   int j=0;
      int n=0;
132   int min1;
      int min2;
134
         //split P1 into separated nodes
136      pch1 = strtok (P1, ":");
         while (pch1 != NULL)
138      {
           P1_nodes[m] = pch1;
140        m++;
           pch1 = strtok (NULL, ":");
142      }

144        // initialiser l indice de la des lignes de la matrice de voisinage
     "Neighbouring"
         m=0;
146
         //parcours du 1er chemin
148      i=0;

150      while(P1_nodes[i]!=(NULL,"ik")){

152      int n=0;
```

```
154         Neighbouring[m][0] = P1_nodes[i];

156         n++;

158         if (i==0){
                Neighbouring[m][n]=P1_nodes[i+1];
160             //la derniere case de chaque ligne se refere sur le nombre de
    voisins de chaque ligne
                Neighbouring[m][4]=1;
162
            } else {
164
                Neighbouring[m][n]=P1_nodes[i-1];
166             n++;
                Neighbouring[m][n]=P1_nodes[i+1];
168             Neighbouring[m][4]=2;
                }
170       m++;
          i++;
172         }

174       //split P2 into separated nodes
          n=0;
176       pch2 = strtok (P2, ":");
          while (pch2 != NULL)
178       {
            P2_nodes[n] = pch2;
180         n++;
            pch2 = strtok (NULL, ":");
182       }

184        //parcours du 2eme chemin
    i=0;
186
    while(P2_nodes[i]!=(NULL,"ik")){
188
    j=0;
190
    while(Neighbouring[j][0] != NULL){
192
                // tester si le
194             if (P2_nodes[i] == Neighbouring[j][0]){

196                 exist=1;

198                 //recuperer la valeur de n depuis la derniere case de la
    ligne
                   // n = Neighbouring[j][4].toInt() + 1;
200
                // Si il est en debut du chemin P2
202             if (i==0){

204             //tester si le nouveau voisin a ajouter a un id existe deja
                  if ( Neighbouring[j][1]!= P2_nodes[i+1] && Neighbouring[j][2
    ] != P2_nodes[i+1] && Neighbouring[j][3] != P2_nodes[i+1] ){
206                     Neighbouring[j][Neighbouring[j][4].toInt()+1]=P2_nodes
    [i+1];
                        Neighbouring[j][4]= Neighbouring[j][4].toInt()+1;
208
                        }
210
```

```
                } else {

                    //tester si le nouveau voisin a ajouter a un id existe deja
                if ( Neighbouring[j][1]!= P2_nodes[i-1] && Neighbouring[j][2] != P2_nodes[i-1] && Neighbouring[j][3] != P2_nodes[i-1] ){

                        Neighbouring[j][Neighbouring[j][4].toInt()+1]=P2_nodes[i-1];
                        Neighbouring[j][4]= Neighbouring[j][4].toInt()+1;

                    }

                if ( Neighbouring[j][1]!= P2_nodes[i+1] && Neighbouring[j][2] != P2_nodes[i+1] && Neighbouring[j][3] != P2_nodes[i+1] ){

                        n++;
                        Neighbouring[j][Neighbouring[j][4].toInt()+1]=P2_nodes[i+1];
                        Neighbouring[j][4]= Neighbouring[j][4].toInt() +1;
                  }
                }
                   // si l id existe deja sortir de la boucle interne Neighbouring
                    break;

            } else {

                    exist=0;
              }
        j++;
        }
        // si le Id n'existe pas deja dans la matrice
        if(exist==0){

              int n=0;

              Neighbouring[m][0] = P2_nodes[i];

              n++;

              if (i==0){

                    Neighbouring[m][n]=P2_nodes[i+1];
                    Neighbouring[m][4]=1;


              } else {

                    Neighbouring[m][n]=P2_nodes[i-1];
                    n++;
                    Neighbouring[m][n]=P2_nodes[i+1];
                    Neighbouring[m][4]=2;
              }
                    m++;
          }
      i++;
    }
          Serial.println("");
          Serial.println("Debut de voisignage");

              i=0;
```

```
                while(Neighbouring[i][0]!=NULL){
268
                  for(j=0; j<5; j++){
270
                       Serial.print(Neighbouring[i][j]);
272                    Serial.print(" ");
                       }
274
                       Serial.println(" ");
276                    i++;
               }
278                      Serial.println("Fin de voisignage");
}
```

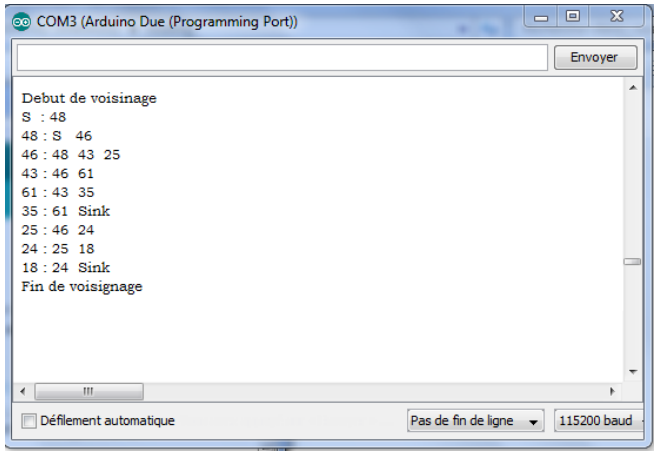


Figure 4.27 – Résultat initial.

Dans la figure 4.27, on trouve une capture du code Arduino de l'algorithme ERO, ainsi que les résultats initiales des tables de voisinage.

Conclusion

Dans ce chapitre, nous avons présenté une application de surveillance et son implémentation au sein du LRIT. Il s'agit de déployer un RCSF qui s'organise automatiquement en une topologie plate, afin de surveiller la température et l'humidité dans le environnement d'expérience. L'application a été mise en œuvre suivant ces étapes décrites dans le chapitre :

Nous avons présenté les différentes plateformes matériel de test existantes dans notre laboratoire LRIT, La comparaison entre eux nous a mener à choisir celle d'Arduino, car elle est simple et économe en énergie.

Nous avons également expliqué, le choix de l'environnement logiciel, qui peut se

résumer par l'utilisation des langages : C pour Arduino, Python pour la communication en série, PHP/HTML5/CSS3 pour le développement de l'interface utilisateur.

Nous avons présenter l'architecture générale de notre implémentation. cette architecture décrit l'ensemble des fichiers de notre code source Arduino, ainsi que les différentes fonctionnalités que l'interface utilisateur doit assurer.

Finalement, en section nous avons présenté notre état d'avancement en implémentation de notre protocole de routage TLBOR dans l'application réelle.

Résumé

Dans les Réseaux de Capteurs Sans Fil (RCSF), plusieurs problèmes d'optimisation sont considérés NP-difficiles, tels que le problème de couverture, d'ordonnancement et de routage. De tels problèmes nécessitent des solutions de bonne qualité dans un délai acceptable, d'ou l'utilisation des métaheuristiques. Dans cette thèse, nous nous intéressons particulièrement au routage, puisque ce problème présente un des grands défis des RCSF. Dans cet ordre d'idées, nous avons proposé des protocoles de routage basés sur différentes métaheuristiques inspirées de la nature. Les travaux effectués durant cette thèse ont mené à trois contributions principales :

La première contribution consiste à améliorer une approche basée sur l'algorithme de colonie de fourmis (Ant Colony Optimization (ACO)) afin de concevoir une nouvelle. Comparée à l'approche de base, la nouvelle approche ACO proposée a minimisé la consommation énergétique et a prolongé la durée de vie d'un RCSF, selon les résultats trouvés.

La deuxième contribution se focalise sur des études comparatives entre l'approche ACO et une autre basée sur l'optimisation par essaim de particules (Particle Swarm Optimization (PSO)). Ainsi, nous avons évalué l'impact de la mobilité aléatoire sur ces deux protocoles, dans des RCSF composés de nœuds dynamiques suivant le modèle de mobilité Random Waypoint (RWP).

Dans la troisième contribution, nous avons proposé un nouveau protocole de routage basé sur une nouvelle métaheuristique nommée Teaching Learning Based Optimization (TLBO) développée récemment pour les problèmes d'optimisation continus. Nous avons pu adapter cette métaheuristique au problème de routage qui est un problème discret. L'adaptation est basée sur la redéfinition des équations tout en gardant l'aspect général de la méthode qui se base sur la relation enseignant-étudiants et les interactions entre les étudiants.

Afin de valider nos travaux effectués et tester les performances de notre protocole de routage vis-à-vis des autres protocoles dans des simulations réelles, nous sommes en cours de réaliser une application de surveillance réelle utilisant un RCSF composé de capteurs Arduino.