REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Batna 2

Faculté des Mathématiques et de l'Informatique

Département d'Informatique

THESE

En vue de l'obtention du diplôme de

Doctorat en Sciences en Informatique

Présentée par

Kamel BARKA

Une plateforme Middleware pour l'auto-adaptation des réseaux de capteurs sans fil hétérogènes

---

Soutenue publiquement le **08/06/2019** devant le jury formé de :

Dr. Seghir Rachid	**Président**	MCA à l'université de Batna 2
Pr. Azeddine BILAMI	**Rapporteur**	Prof. à l'université de Batna 2
Pr. Mamri Ramdane	**Examinateur**	Prof. à l'université de Constantine 2
Pr. Bitam Salim	**Examinateur**	Prof. à l'université de Biskra
Pr. Mezioud Chaker	**Examinateur**	Prof. à l'université de Constantine 2
Dr. Sedrati Maamar	**Examinateur**	MCA à l'université de Batna 2

Remerciements

Je remercie d'abord le Dieu tout-puissant de m'avoir donné la santé, la force et l'ambition de pouvoir faire ce travail.
Le travail réalisé tout au long de cette thèse n'a été possible qu'avec l'aide et le soutien de nombreuses personnes. Je profite de l'occasion qui m'est donnée pour leur exprimer toute ma gratitude.

Je tiens à remercier sincèrement et en premier lieu mon directeur de thèse qui a accepté de diriger mes travaux de thèse et m'a donné un encadrement de qualité tout au long de cette thèse. Il s'agit du Pr. **Azeddine Bilami**, Professeur au département d'informatique de l'université de Batna 2 et dirigeant du laboratoire Informatique L@STIC. Grâce à sa sympathie, sa franchise, sa disponibilité, ses idées, ses conseils et son encouragement, j'ai pu bien mener mes travaux de recherches et rédiger ce rapport de thèse.

Je remercie également tous les membres du jury de cette thèse, les rapporteurs et examinateurs qui ont accepté d'évaluer mes travaux de thèse. Je remercie en ce sens respectivement, Dr. **Seghir Rachid**, MCA à l'université de Batna 2, Pr. **Mamri Ramdane**, professeur à l'université de Constantine 2, Pr. **Bitam Salim**, professeur à l'université de Biskra, Pr. **Mezioud Chaker**, professeur à l'université de Constantine 2 et Dr. **Sedrati Maamar**, MCA à l'université de Batna 2.

Je tiens à remercier mes parents et tous les membres de ma famille qui m'ont apporté le soutien et le confort qui m'ont accompagné durant ce parcours.

Ce travail n'a pu atteindre ses objectifs sans la contribution de près ou de loin de plusieurs personnes auxquelles j'adresse mes chaleureux remerciements.

A vous tous, Merci !

Dédicaces

A mes chers parents, et ma grande famille,

A ma chère épouse,

A mon fils, Iyed Elhak !

Résumé

Au cours de ces dernières années, les réseaux de capteurs sans fil ont connu un intérêt croissant à la fois au sein de la communauté scientifique et industrielle en raison du large potentiel en termes d'applications offertes. Toutefois, les capteurs sont conçus avec d'extrêmes contraintes en ressources, en particulier la limitation de l'énergie. Nous avons récemment constaté l'apparition des nouvelles applications où nous avons besoin d'un nouveau composant du réseau appelé actionneur « super capteur ». Ces actionneurs disposent généralement d'une source d'énergie abondante, par conséquent ils sont riches en ressources (la capacité du traitement, du stockage, de puissance d'émission, etc…), ce type de réseau appelé réseau sans fil de capteurs et d'actionneurs (RCASF) se considère comme étant un réseau hétérogène. L'hétérogénéité est causée par la coexistence des nœuds capteurs à faibles ressources et des nœuds actionneurs riches en ressources. C'est dans ce contexte que se déroule cette thèse dans laquelle nous avons proposé des mécanismes d'auto-gestion et d'auto-contrôler s'inspirant d'un système biologie et aussi s'appuyant sur l'hétérogénéité.

Au début, nous proposons un nouveau Framework middleware bio-inspiré, qui assure l'auto-gestion d'un RCSF homogènes, appelé MONet. Ce Framework basé sur des agents mobiles et une approche bio-inspirée hybride (l'intelligence en essaim SI et les algorithmes génétique GA) dans lequel les agents acheminent les données captées vers la SB et leurs architectures sont optimisées à travers un GA afin d'améliorer leurs performances, à noter aussi que les comportements de ces agents, sont inspirés d'une colonie d'abeilles.

Par la suite, nous avons proposé aussi deux solutions pour la réduction de l'émission de phéromones par l'agent pendant son déplacement sur les nœuds, dont la première basée sur la caractéristique de l'égoïsme de l'agent et la deuxième basée sur un seuil d'émission. Les résultats des simulations montrent que la durée de vie du réseau est étendue de plus du tiers.

Finalement, nous nous sommes intéressés à l'hétérogénéité dans les RCSF à savoir RCASF (WSAN). Se basant sur l'idée que les ressources au niveau des nœuds actionneurs doivent être pleinement exploitées afin de réduire la charge de communication au niveau des nœuds capteurs, nous avons proposé une extension de MONet appelée BISSA. Qui permet de profiter de la puissance d'émission et de la capacité de stockage des actionneurs pour améliorer les performances du réseau. Comparée au MONet, cette extension réalise une prolongation de la durée de vie du réseau d'environ deux tiers, tout en améliorant les performances du réseau de plus de la moitié.

***Mots clés** : l'autonomie, les solutions bio-inspirées, MONet, BISSA, RCSF, RCASF (WSAN).*

Abstract

During the past few years, wireless sensor networks witnessed an increased interest in both the industrial and the scientific community due to the potential wide area of applications. However, sensors' components are designed with extreme resource constraints, especially the power supply limitation. We have recently seen the emergence of new applications where we need a new component of the network called actuator "Super Sensor". These actuator nodes are resource-rich devices with higher processing capabilities, transmission power and larger battery capacity. This type of network called Wireless Sensor and Actuator Networks (WSAN). this is a heterogeneous network. This heterogeneity is caused by the coexistence of sensor nodes with limited resources and actuator nodes with higher resources. It is in this context that this thesis takes place in which we proposed self-management and self-control mechanisms through biologically inspired computing and also based on heterogeneity.

first, we propose a new bio-inspired middleware framework, which ensures the self-management of a homogeneous WSN, called MONet. This framework is based on mobile agents (software agent) and a hybrid bio-inspired approach (i.e., swarm intelligence IS and genetic algorithm GA) in which the agents collect sensor data or detect an event on individual nodes, and carry sensor data to base stations. their architectures are optimized within a genetic algorithm to improve their performance, as well as the behaviors of these agents, they were inspired by an artificial Bee Colony

Secondly, we also proposed two solutions for reducing the pheromone emission by the agent during its migration on the nodes whose the first based on the selfishness characteristic of the agent and the second based on an emission threshold to control the level of cooperation among them. Simulation results show that the network lifetime is extended by more than one third.

Finally, we were interested in the heterogeneity in the WSN namely WSAN. Based on the idea that resource-rich nodes must be exploited to reduce the communication load level on low-power nodes, we proposed an extension of MONet called BISSA. BISSA uses the large transmit power and storage capacity of actuators to provide network performances. Compared to MONet, this extension extends the network lifetime about two-thirds, while improving network performances by more than half.

Keywords: *Autonomy, bio-inspired solutions, MONet, BISSA, WSN, WSAN.*

الملخص

في السنوات الأخيرة، اكتسبت شبكات الاستشعار اللاسلكية اهتماما متزايدا داخل المجتمع العلمي والصناعي بسبب الإمكانات الواسعة للتطبيقات المتاحة. ومع ذلك، تم تصميم أجهزة الاستشعار مع محدودية الموارد وخاصة كمية الطاقة جد محدودة. كما شهدنا مؤخرا ظهور تطبيقات جديدة حيث نحتاج فيها إلى مكون جديد للشبكة يسمى «المشغل ». عموما، هذه المشغلات لديها مصدر وفير للطاقة، وبالتالي فهي غنية بالموارد (قدرة المعالجة، التخزين، طاقة النقل، إلخ...)، هذا النوع من الشبكات يسمى شبكة اللاسلكية لاستشعار والمشغلات، وهذه هي شبكة غير متجانسة، بسبب تواجد فيها عقد استشعار ذات موارد محدودة وعقد المشغلات الغنية بالموارد. في هذا السياق، تدور هذه الأطروحة التي اقترحنا فيها آليات الإدارة الذاتية والتحكم الذاتي مستوحاة من نظام البيولوجيا وأيضاً تعتمد على عدم التجانس.

في البداية، نقترح إطارًا وسيطًا جديدًا مستوحى من الأنظمة البيولوجية، والذي يضمن الإدارة الذاتية لشبكات الاستشعار اللاسلكية المتجانس، يسمى MONet. هذا الإطار يعتمد على الوكلاء المتحركون وعلى نهج مختلط المستوحى من التقنيات الحيوية (ذكاء السرب والخوارزميات الجينية) حيث يقوم العملاء بنقل البيانات الملتقطة إلى القاعدة ويتم تعديل بنيتهم داخل خوارزمية جينية لتحسين أدائهم، وكذلك سلوكيات هؤلاء الوكلاء، مستوحاة من مستعمرة النحل.

بعد ذلك، اقترحنا أيضًا حلّين لتقليل من استهلاك الطاقة داخل MONet وذلك بخفض ارسال الفيروموناتمن طرف الوكيل أثناء تنقله على العقد. الحل الأول يستند على خاصية الأنانية الخاصة بالوكيل والثاني يستند على عتبة الارسال. تظهر نتائج المحاكاة أن مدة حياة الشبكة يمتد بأكثر من الثلث.

أخيراً، اهتمينا بعدم التجانس الشبكات، وبالتحديد في WSAN. واستناداً إلى فكرة أنه يجب استغلال الموارد الموجودة في العقد المشغلات من أجل تقليل عبء الاتصال في عقد الاستشعار، فقد اقترحنا امتدادًا لـ MONet يسمى BISSA. حيث ان BISSAتستغل سعة الإرسال وسعة التخزين للمشغلين لتحسين أداء الشبكة. وبالمقارنة مع MONet، فإن هذا الامتداد يساهم في تمديد مدة حياة الشبكة بحوالي الثلثين، مع تحسين أداء الشبكة بأكثر من النصف.

كلمات البحث: الاستقلالية، الحلول المستوحاة من الطبيعة،MONet، BISSA ، WSN، WSAN.

Table des matières

Liste des figures

Liste des tableaux

Glossaries des acronyms

ABC The Artificial Bee Colony algorithm

ACO Ant Colony Optimization

AIS Artificial Immune System

ANN Artificial neural network

AIS Artificial Immune System

BCO Bee Colony Optimization

BFO Bacterial Foraging Optimization

BISSA A Biologically-Inspired framework for Self-adaptation wireless Sensor and Actuator network

BSN Wireless body sensor networks

EGT Evolutionary Game Theory

GA Genetic Algorithm

MOGA l'algorithmes génétiques multi-objectifs

MONet Multi-objective Optimization for wireless sensor NETworks using a biological mechanism

MOP problème multi-optimisation

PCO pulse-coupled oscillations

PSO Particle Swarm Optimization

QMP Queen Mandibular Pheromone

SI Swarm Intelligence

SMA Système Multi-Agent

UWSN Underwater sensor networks

WiNS Wireless image Sensor Networks

WMSN Wireless Multimedia Sensor Networks

WSAN Wireless Sensor and Actuator Networks

WSN Wireless Sensor Network

WUSN Wireless Underground Sensor Network

WVSN Wireless Video Sensor Networks

RCAcSF Réseaux de Capteurs Acoustiques Sans Fil

RCASF Réseaux de Capteurs et Actionneurs Sans Fil

RCCSF Les Réseaux de Capteurs Corporels Sans Fil

RCISF Réseaux de Capteurs Image Sans Fil

RCMSF Réseau de Capteurs Multimédia Sans Fil

RCSF Réseau de Capteurs Sans Fil

RCSMSF Réseaux de Capteurs Sous-marins Sans Fil

RCVSF Réseaux de capteurs vidéo sans fil

RNA Réseaux de Neurones Artificiels

Introduction générale

Le contexte et les motivations

Les avancées technologiques durant ces dernières années sont accompagnées de conception d'objets de plus en plus petits, à faible coût, à faible ressources (énergie, communication, calcul), intelligents et autonomes, appelés les capteurs sans fil. Ces capteurs peuvent mesurer une distance, une direction, une vitesse, une vibration, etc. Ces capteurs disposent d'une antenne pour assurer la réception et la transmission des signaux. Ils disposent aussi d'un mini processeur et une capacité de stockage limitée pour (dé)coder les signaux et pour exécuter des protocoles de communications. Enfin, ces capteurs sont alimentés généralement par une source d'énergie limitée comme une batterie.

Les capteurs sont généralement déployés dans un environnement et sont attachés à une station de base appelée puits. L'objectif de ces nœuds capteurs est de récolter des grandeurs physiques (température, pression, humidité, luminosité, etc.) à partir de cet environnement et de les envoyer vers cette station de base. Tout d'abord, ces capteurs doivent coopérer entre eux pour s'auto-organiser et s'auto-configurer afin de constituer un réseau de capteurs sans fil ; RCSF (*WSN : Wireless Sensor Network*). Ce RCSF consiste généralement en un grand nombre de capteurs homogènes à faible ressources et au moins une station de base.

Malgré que les capteurs aient une grande autonomie dans la gestion de ses ressources, mais c'est très difficile pour un utilisateur de contrôler et de gérer manuellement des RCSF dynamiques dans lesquels leurs tailles (nombre de nœuds) sont toujours changeantes à cause d'ajout des nouveaux capteurs ou de la défaillance des capteurs ou des liens. Cette difficulté est augmentée surtout dans ces cas, le RCSF est un réseau dépasse l'échelle ou ils se sont déployés dans des environnements inaccessibles, hostiles et /ou instables. Pour cela les mécanismes qui facilitent ces opérations de minimisation des interventions de l'utilisateur dans la gestion du réseau (les capteurs), deviennent obligatoires. De plus, ces mécanismes doivent assurer l'efficacité énergétique parce que l'économie d'énergie représente l'un des grands défis à soulever pour le bon fonctionnement des réseaux de capteurs.

Bien que les RCSF soient utilisés dans plusieurs applications, nous constatons l'apparition des nouvelles applications où nous avons besoin d'un nouveau composant du réseau appelé actionneur ou super nœud. Ces actionneurs disposent généralement d'une source d'énergie abondante. Les fonctions de calcul et de communication au niveau des actionneurs profitent donc de cette richesse en énergie, par conséquent les capacités de calcul et de stockage ainsi que la puissance de transmission au niveau des nœuds actionneurs sont plus importantes que celles au niveau des nœuds capteurs [1]. Cette extension des réseaux de capteurs inclut les nœuds actionneurs qui ne sont pas considérés seulement comme responsables pour agir sur l'environnement mais d'un point de vue réseau, ils sont considérés comme des points de collecte locaux. Cette architecture est appelée réseau sans fil de capteurs et d'actionneurs (*WSAN Wireless Sensors and Actuators Networks*). La différence majeure entre les WSAN et les RCSF est que les WSAN sont des réseaux hétérogènes par nature. Pour cela nous avons

besoin des nouveaux mécanismes qui profitent de cette hétérogénéité entre les nœuds actionneurs et les nœuds capteurs avec une nouvelle stratégie dans laquelle les ressources disponibles au niveau des nœuds actionneurs sont pleinement exploitées afin de réduire la charge de communication et de traitement au niveau des nœuds capteurs et d'améliorer aussi les performances du réseau.

L'intelligence en essaim (*SI : Swarm intelligence*) est l'étude du comportement collectif des communautés d'individus sociaux, ce qui fournit des outils et des algorithmes métaheuristiques hautement efficaces qui traitent beaucoup de propriétés souhaitables et intéressantes appliquées dans les RCSF [2,3,4]. De plus, l'intelligence en essaim peut être vue comme des analogies entre les méthodes de calcul et les comportements biologiques des essaims dans lesquels une intelligence collective peut émerger. Les essaims considérés sont des colonies d'insectes sociaux, des troupeaux d'oiseaux et des bancs de poissons [5]. Chaque type d'essaim est constitué d'individus simples et autonomes, qui collaborent les uns avec les autres pour accomplir certaines tâches nécessaires à la survie de la colonie.

La tendance actuelle de la recherche vise, l'utilisation de solutions biologiques pour résoudre et optimiser différents aspects des systèmes artificiels, (i.e les problèmes de la vie réelle), elle a été façonnée dans un domaine important appelé l'informatique bio-inspirée (*the bio-inspired computing*) [6]. Dans le contexte du RCSF, la combinaison du rendement simple des individus et les comportements robustes, a été appliquée avec succès en identifiant et en modélisant certaines analogies entre les deux systèmes afin de former une solution unique de résolution de problèmes [7].

L'auto-contrôlement et l'auto-gestion dans les RCSF dynamique dans un contexte homogène ou hétérogène, sont des problèmes d'optimisation bien connus et bien étudiés pour le développement de nombreux algorithmes, protocoles et plateformes basés sur les approches bio-inspirées comme l'intelligence en essaim. Depuis plus d'une décennie, les solutions bio-inspirées attirent progressivement l'attention des chercheurs grâce à leur efficacité dans la résolution d'un certain nombre de problèmes d'optimisation dans divers domaines. L'auto-gestion en termes de l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation, ainsi que l'efficacité énergétique implique que la charge de travail sera répartie plus uniformément sur les composants du réseau et aussi doit équilibrer les niveaux d'énergie des nœuds en utilisant des solutions bio-inspirées hybride qui régissent nos contributions dans cette thèse.

Les contributions de thèse

La conception des plateformes d'auto-gestion et des protocoles à faible consommation d'énergie afin de contrôler et de prolonger la durée de vie du RCSF, a été largement étudiée dans la littérature, mais il reste un domaine de recherche actif. L'exploitation de l'hétérogénéité dans les réseaux sans fil comme WSAN, a une grande influence sur les performances globales du réseau et prolonger le plus possible sa durée de vie. Dans cette thèse, en utilisant les solutions bio-inspirées, nous visons à (concevoir) développer une plateforme middleware (Framework) dédiée au RCSF homogènes ou

hétérogènes afin qu'il puisse assurer les opérations de l'auto-gestion, et exploiter pleinement les ressources disponibles sur chaque nœud afin d'optimiser les performances du réseau et prolonger sa durée de vie. Pour atteindre cet objectif, nous avons réalisé les contributions suivantes :

1. Dans **la première contribution** : nous avons proposé un framework bio-inspiré dédié aux RCSF homogène, appelé MONet. Les mécanismes du MONet sont inspirés d'un système d'intelligence en essaim, à savoir « la colonie d'abeilles » afin d'assurer une grande autonomie dans la gestion du RCSF en termes de l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation tout à la fois. Le MONet est un système multi-agents dans lequel chaque agent « agent logiciel » réside dans un capteur afin qu'il capte et achemine les données vers une station de base. Comme les abeilles, les agents collaborent entes eux en mettant des phéromones sur les capteurs et optimisent leurs chemins de migration à travers un algorithme appelé ABC. Ces agents ont des structures génotypiques améliorable par un algorithme génétique implémenté dans un serveur. Cet algorithme évolue les agents périodiquement jusqu'à ce qu'il trouve le meilleur compromis entre leurs performances, à savoir la latence, le coût, le taux de réussite et le degré de l'agrégation de donnée.
2. Dans **la deuxième contribution** : bien que l'agrégation de données soit une technique qui diminue la consommation énergétique mais elle n'est pas suffisante. Pour cela nous avons proposé deux mécanismes qui minimisent les émissions des phéromones en sein du MONet et les plateformes similaires. Le premier est décentralisé basé sur la structure génotypique de l'agent, elle contient un gène, appelé « gène d'égoïste » qui détermine l'envie de l'agent d'émettre les phéromones ou non. Le deuxième est centralisé basé sur un seuil calculé et ajusté périodiquement par le serveur en fonction de l'entropie du RCSF. Chaque agent n'émet les phéromones que si la valeur du seuil dépasse la valeur de la concentration de phéromone correspondante au nœud qu'il se trouve actuellement. Les résultats des simulations montent que ces mécanismes réduisent la consommation énergétique de plus d'un tiers.
3. Dans **la troisième contribution** : nous avons proposé une extension du MONet, appelée BISSA. Cette extension dédiée au RCSF hétérogènes, à savoir les réseaux sans fil de capteurs et actionneurs. L'objectif principal de cette extension est d'exploiter l'hétérogénéité du réseau afin d'améliorer ses performances. Pour atteindre cet objectif nous avons développé une nouvelle architecture de l'agent qui lui permet de changer son rang selon le type du nœud afin qu'il exploite pleinement les ressources disponibles sur ce nœud. Les résultats de simulation montent que BISSA fonctionne mieux dans les réseaux de capteur sans fil, soit homogènes, soit hétérogènes, ainsi qu'il fournit une utilisation des ressources disponibles d'une manière équitable entre les nœuds, ce qui permet de prolonger la durée de vie du réseau d'environ deux tiers.

Organisation de la thèse

Cette thèse est organisée en cinq chapitres.

Nous commencerons par l'introduction générale, dans laquelle sont décrits le contexte de notre travail, la problématique soulevée et les objectifs de notre travail.

Le chapitre 1 présente une description d'un réseau de capteurs sans fil et ses caractéristiques spécifiques liées soit aux capteurs qui le constituent comme l'énergie, la portée de transmission, …etc. soit liées au réseau lui-même comme l'homogénéité, l'hétérogénéité, les plateformes middleware ...etc. Ce chapitre présente des généralités sur ce domaine de recherche en pleine évolution.

Le chapitre 2 présente les solutions bio-inspirées et leurs techniques. Ce chapitre est un état de l'art sur l'utilisation des techniques bio-inspirées pour aborder les défis dans les réseaux homogènes et les réseaux hétérogènes tout en mettant l'accent sur les quatre défis concernant l'auto-gestion.

Le chapitre 3 présente notre première contribution. Il s'agit de la proposition d'un framework middleware basé sur des agents et des mécanismes qui se sont inspirés de la colonie d'abeilles, appelé MONet. Ensuite, présente l'études détaillée de ses performances par simulations.

Le chapitre 4 présente notre deuxième contribution à savoir deux mécanismes qui assurent l'économie d'énergie dans le MONet et les plateformes similaires. Ces mécanismes basés sur la minimisation d'émissions des phéromones sans inflation sur les performances du réseau ainsi que des études détaillées de ses performances par simulations.

Dans le chapitre 5 on présente notre troisième contribution à savoir une extension du MONet pour les applications dans un réseau hétérogène permettant non pas seulement de détecter les nœuds riches en ressource mais aussi permettant d'exploiter ces ressources afin d'améliorer les performances du réseau.

Enfin, dans la conclusion générale, nous résumons les principaux apports, en présentant les différentes perspectives et directions des recherches futures.

Chapitre 1

Les réseaux de capteurs sans fil homogènes et hétérogènes

&

Les plateformes middleware

1.1 Introduction

Dans la dernière décennie, les réseaux de capteurs sans fil, ont attiré un grand nombre de chercheurs et d'industriels pour leur intérêt, et ont apporté des solutions dans un grand nombre d'applications. L'exécution de ces applications sur les RCSF est une tâche importante et difficile en raison de gap entre les exigences de l'application et les opérations qui ont été offert par le réseau. En outre, les RCSF sont typiquement hétérogènes. La raison essentielle en est les progrès scientifiques dans le domaine des réseaux où les meilleures technologies finissent par coexister sur le même réseau. Concevoir un logiciel pour des systèmes distribués est déjà une tâche qui n'est pas facile, et en ajoutant le problème d'hétérogénéité, cela devient beaucoup plus compliqué. D'où le besoin d'une couche intermédiaire entre l'application et le réseau qui gère ce problème, en plus des problèmes classiques des systèmes distribués, comme la transparence, la localisation, l'accès aux ressources partagés, etc. Cette couche s'appelle intergiciel, ou plus couramment : middleware. Dans ce chapitre nous commençons par introduire les réseaux RCSF homogènes et hétérogènes, leurs applications et leurs défis. Nous décrivons par la suite, la couche middleware, ses approches et les principes de conceptions d'approches middleware conçus pour RCSF.

1.2 Les réseaux de capteurs sans fil

1.2.1 Les modèles d'un nœud capteur

Un capteur sans fil est un petit dispositif électronique capable d'interagir avec l'environnement où il est déployé, de mesurer une valeur physique (température, lumière, pression, etc.), et de la communiquer à un centre de contrôle via une station de base (cf., la figure 1.1) [8].

Figure 1.1 Exemple d'un nœud capteur

Généralement, Un nœud est constitué d'unités essentielles et d'unités supplémentaires (optionnelles). Selon les unités et leurs capacités, qui sont constitués le nœud, on peut définir plusieurs modèles qui sont utilisées dans les RCSF. Dans même RCSF, on peut trouver les différents types de nœuds [9].

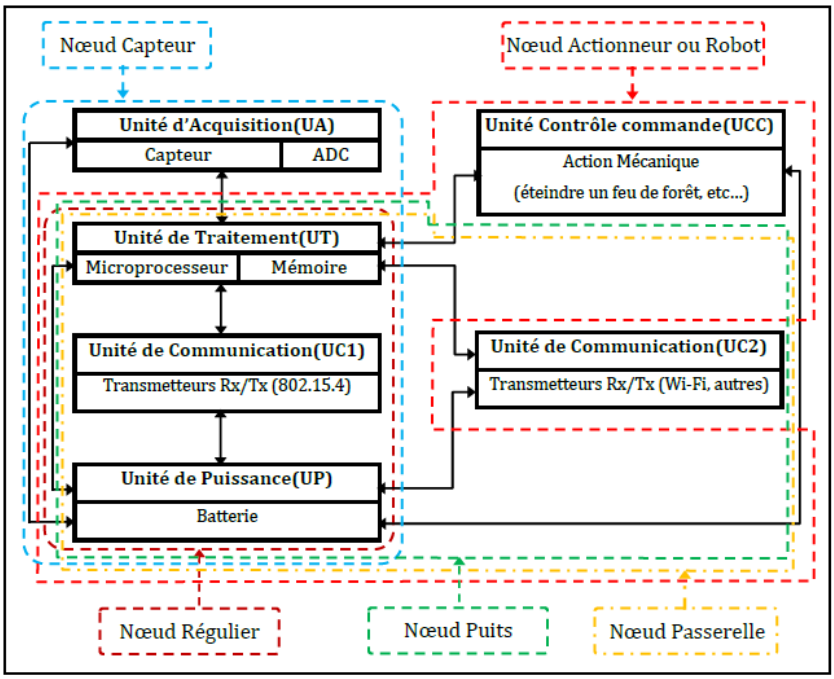

Figure 1. 2 Architecture des différents types de nœuds : régulier, capteur, robot, puits, passerelle [9]

Un nœud régulier : c'est un nœud basique doté d'une unité de transmission, d'une unité de traitement de données et une batterie limitée.

- L'unité de transmission de données est responsable de toutes les émissions et réceptions de données via un support de communication sans fil.
- L'unité de traitement de données est composée d'une mémoire, d'un microcontrôleur et d'un système d'exploitation spécifique (comme TinyOS). Elle est responsable du traitement des données en provenance ou au départ de l'unité de transmission.
- Ces deux unités sont alimentées par une batterie limitée comme le montre la Figure 1.2

Un nœud capteur ou nœud source : est un nœud régulier équipé d'une unité d'acquisition ou de détection.

- L'unité d'acquisition est généralement dotée d'un capteur ou plusieurs capteurs qui obtiennent des mesures analogiques (physiques et physiologiques) et d'un convertisseur Analogique/Numérique qui convertit l'information relevée en un signal numérique compréhensible par l'unité de traitement. En effet, le nœud capteur est un nœud à faible ressources.

Un nœud actionneur ou robot est un nœud régulier doté d'une unité lui permettant d'exécuter certaines tâches spécifiques comme des tâches mécaniques (se déplacer, combattre un incendie, piloter un automate, etc.). Comparés aux nœuds capteurs, les actionneurs sont des nœuds riches en ressources, ayant une grande capacité de calcul, de mémoire et une puissance de transmission importante.

Un nœud passerelle (ou Gateway) est un nœud régulier permettant de relayer le trafic dans le réseau sur le même canal de communication.

Un nœud puits est un nœud régulier doté d'un convertisseur série connecté à une seconde unité de communication (GPRS, Wi-Fi, WiMax, etc.). La seconde unité de communication fournit une retransmission transparente des données provenant de nœuds capteurs à un utilisateur final ou d'autres réseaux comme internet.

Selon le domaine d'application, ces modèles peuvent être équipé d'autres unités, tels qu'un système de localisation (GPS) pour déterminer sa position, ou bien un système générateur d'énergie (cellule photovoltaïque), ou encore un système mobile pour lui permettre de changer sa position ou sa configuration en cas de nécessité.

1.2.2 RCSF homogène

Les réseaux de capteurs sans fil RCSF [10] (*WSNs : Wireless Sensor Networks* en anglais) sont un type particulier des réseaux Ad-hoc, dans lesquels les nœuds sont des capteurs (*motes* ou *sensors* en anglais). Ces réseaux sont généralement constitués d'un grand nombre de capteurs qui sont autonomes, à faible coût, à faible ressources et déployés manuellement ou aléatoirement dans une zone géographique donnée, appelée zone à surveiller ou champ de captage, qui se communiquent entre eux via des liens radio pour le partage d'information et le traitement coopératif [11,12] Une fois déployés, ils s'auto-configurent et s'auto-organisent en un réseau sans fil multi-sauts (cf., la figures 1.3 et 1.4(a)). Dans ce type de réseau, chaque capteur est capable de récolter des grandeurs physiques à partir de l'environnement où ils sont déployés, de les traiter et enfin de les envoyer vers ses voisins (des autres capteurs) pour que ces derniers relaient ces informations jusqu'à arriver à un nœud spécifique dans le réseau, appelé stations de bases ou Puits (*Sink*). Cette dernière retransmet les données à la destination finale via internet ou par satellite. Dans la plupart des cas, les capteurs sont fixes déployé dans la zone. Cependant, ils peuvent aussi être mobiles et capables d'interagir avec l'environnement [13,14,15].

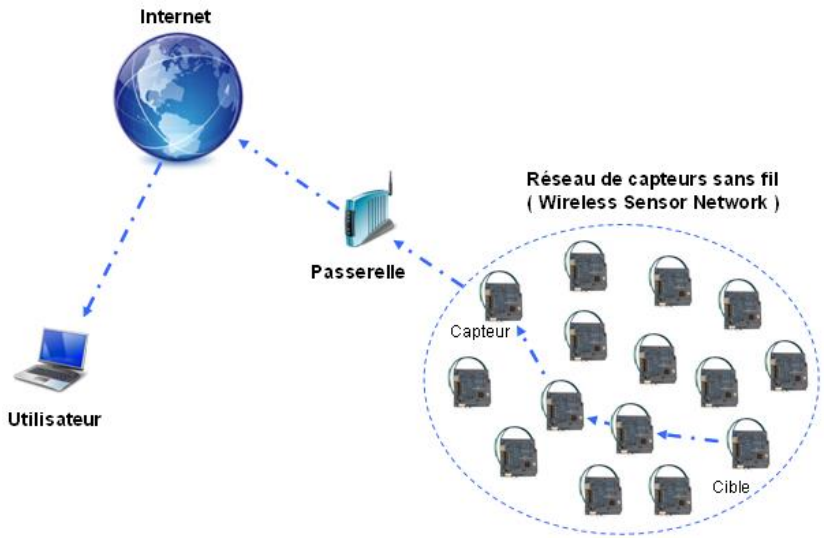

Figure 1. 3 Réseau de capteurs sans fil (RCSF).

La plupart des scénarii envisagés pour les réseaux homogènes RCSF contiennent un seul nœud puits et un grand nombre de nœuds capteurs homogènes. Le puits recueille des informations auprès des nœuds

capteurs, les analyse puis les traite. Les capteurs sont supposés des nœuds homogènes ayant les mêmes caractéristiques matérielles (même capacité énergétique, capacité de calcul, portée de communication, etc.).

1.2.3 RCSF hétérogène

Bien que les RCSF soient employés dans des nombreuses applications, il y a un nombre croissant d'applications qui nécessite l'utilisation d'un autre type de nœuds dans le réseau, ce dernier est le super-nœud. On parle alors d'hétérogénéité de réseau. Une étude faite par [16] a montré qu'un réseau hétérogène proprement déployé peut tripler le taux moyen de livraison et peut prolonger jusqu'à cinq fois plus la durée de vie du réseau. L'utilisation de super-nœuds dans les RCSF est donc perçue comme une voie possible pour faciliter la gestion et le passage à l'échelle du réseau, pour raccourcir les délais de transmission, mais aussi pour améliorer la connectivité et la durée de vie du réseau. Ils exécutent certaines tâches spécifiques comme l'agrégation et le relais des données, ou encore coordonnent les activités de nœuds capteurs, etc…, on peut classer les différentes RCSF hétérogène en trois catégories :

- **Réseau de capteurs sans fil multimodal (***multi-modal wireless sensor network)* [17]: est un réseau hétérogène à une structure généralement plat car il est constitué au moins de deux modèles de nœuds qui collectent les données de l'environnement où ils sont déployés (cf., la figure 1.4 (b)). On parle alors d'hétérogénéité de la composition, à travers les traiter nous pouvons prendre une vue multidimensionnelle du champ de captage, en général, ces modèles de nœuds ayant les même caractéristiques matérielles (même capacité énergétique, capacité de calcul, portée de communication, etc.). DARPA [17] comme exemples.
- **Réseau de capteurs sans fil multi-niveaux** (*multi-tier wireless sensor network*) [18]: est un réseau hétérogène à une structure hiérarchique ou multi-niveaux car il est constitué au moins de deux types de nœuds. On parle alors d'hétérogénéité des caractéristiques : les nœuds capteurs qui collectent les données de l'environnement où ils sont déployés, ceux-ci sont constitués le niveau bas, et les niveaux hauts sont constitués des super-nœuds qui ne sont pas responsables seulement d'agir sur l'environnement physique mais aussi, d'un point de vue réseau, sont responsables des autres opérations non traditionnelles (cf., la figure 1.4(c)), comment par exemple de recueillir les données récoltées par les nœuds capteurs. WVSN [19] comme exemple.
- **Réseau de capteurs sans fil multimodal multi-niveaux (***multi-tier multi-modal wireless sensor network*) [20,21]: est un réseau hybride constitué à partir des deux précédentes classes, le M2WSN est un réseau hétérogène à une structure multi-niveaux et est constitué au moins de deux types et deux modèles de nœuds. On parle alors d'hétérogénéité des caractéristiques et la composition : chaque type est constitué un niveau qui peut contient de différent modèle de nœuds (cf., la figure 1.4(c)). M2WSM peut avoir des avantages supplémentaires de RCSF (single-tier, unimodal) comme (la réduction le coût de construction, augmenter la fiabilité et la couverture et la fonctionnalité etc…), WSAN [22], WMSN [23] comme exemple.

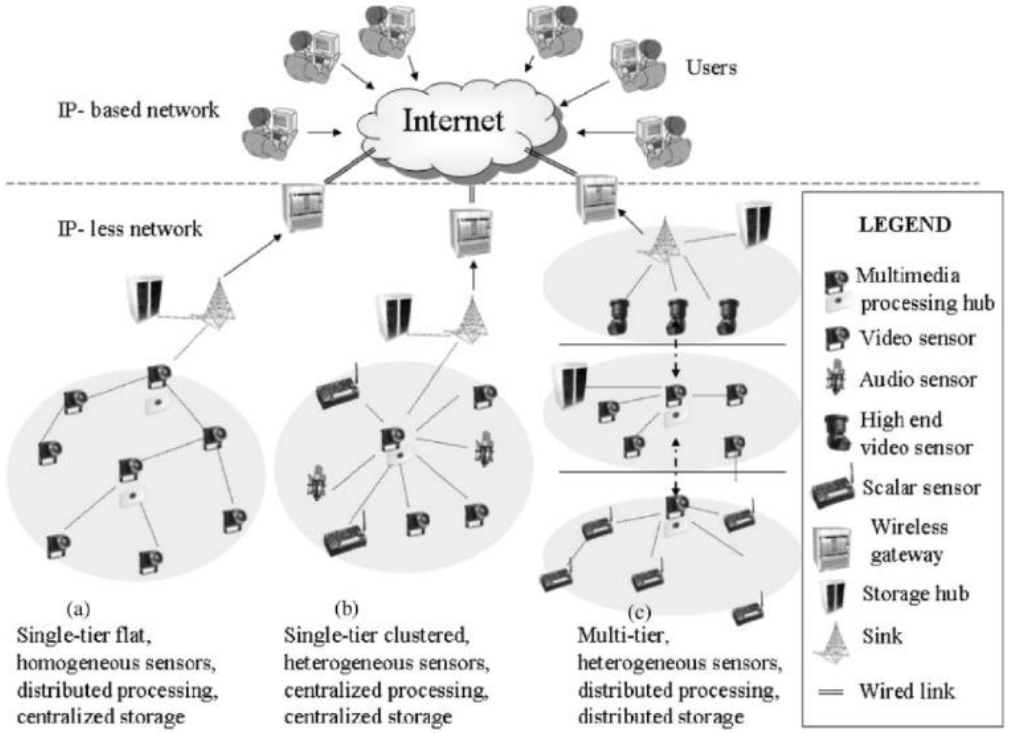

Figure 1. 4 Les différentes classes de RCSF (a)homogènes (b et c) hétérogènes [23]

1.2.4 Types de RCSF hétérogènes

Les RCSF courants sont déployés sur terre, sous terre et dans l'eau. Suivant l'environnement, un réseau de capteur fait face à différents challenges et contraintes. Il y a cinq types de base de réseaux de capteurs, comme les réseaux de capteurs et actionneurs RCASF, Les Réseaux de Capteurs Sous-marins Sans Fil (RCSMSF), Les Réseaux de Capteurs Sous-Terrain Sans Fil (RCSTSF), Les Réseaux de Capteurs Corporels Sans Fil (RCCSF) et réseau de capteurs multimédia sans fil (RCMSF). Il existe aussi des types hybrides comme les RCASF ou multimédia [24].

1.2.4.1 Réseaux de capteurs et d'actionneurs sans fil

Les RCSF fournissent des informations détaillées du monde physique grâce à des solutions de détection distribuées. En général, cette information est traitée au niveau du puits (*Sink*). Avec l'émergence d'actionneurs et de robots à faible coût, l'information recueillie à partir de l'environnement peut être utilisée pour déclencher une action sur cet environnement. Cela a conduit à l'émergence des Réseaux de Capteurs et d'Actionneurs Sans Fil (RCASF ; *WSAN : Wireless Sensor and Actuator Networks*) qui sont capables d'observer le monde physique, traiter localement quelques données pour la prise de décisions sur la base de ces observations, et effectuer des actions appropriées [25]. Un exemple important d'applications utilisant les RCASF est le cas d'extinction automatique des incendies : les capteurs qui détectent un déclenchement de feu dans leurs environnements font remonter cette information aux systèmes d'extinction. L'information devrait rapidement arriver aux actionneurs qui devraient prendre la décision afin que le feu puisse être facilement éteint avant qu'il ne devienne incontrôlable [26]. De même, des capteurs détecteurs de mouvement et de lumière répartis dans une maison peuvent détecter la présence de personnes. Selon l'identité et l'emplacement de

l'utilisateur, les capteurs peuvent commander les actionneurs appropriés pour exécuter des actions, en fonction des préférences de l'individu que la maison peut apprendre au fil du temps.

1.2.4.2 Réseaux de capteurs sous-marins sans fil

Les Réseaux de Capteurs Sous-marins Sans Fil (RCSMSF, *UWSN : Underwater sensor networks*) [27] sont envisagés pour permettre une grande variété d'applications telles que la collecte de données océanographiques, la surveillance de la pollution, l'exploration offshore, la prévention des catastrophes, la navigation assistée et la surveillance tactique. Plusieurs véhicules sans pilote ou sous-marins autonomes, équipés de capteurs sous-marins, peuvent être utilisés pour des applications d'exploration des ressources naturelles sous-marines et de collecte de données scientifiques, ou encore pour recueillir des données de surveillances stratégiques. Pour rendre ces applications viables, il est nécessaire de développer des schémas de communication adaptés à ces appareils sous-marins. Comme pour les exigences des réseaux de capteurs classiques, les nœuds sous-marins doivent posséder des capacités d'auto-configuration. Les RCSMSF peuvent contribuer à la surveillance de courants marins et les vents, l'amélioration des prévisions météo, la détection des changements climatiques, ainsi que la compréhension et la prévention de l'effet des activités humaines sur les écosystèmes marins. Les RCSMSF qui mesurent l'activité sismique à partir d'emplacements distants peuvent fournir des alertes aux tsunamis dans les zones côtières, ou étudier les effets des tremblements de terre sous-marins.

1.2.4.3 Réseaux de capteurs souterrains sans fil

Les Réseaux de Capteurs Sous-Terrain Sans Fil (RCSTSF ; *WUSN : Wireless Underground Sensor Network*) [28] sont constitués de nœuds sans fil qui fonctionnent sous la surface du sol. Ces dispositifs sont soit complètement enterrés sous un sol dense, ou placés dans un espace souterrain ouvert borné, comme les mines et les tunnels routiers/métro. Les RCSTSF promettent une grande variété de nouvelles applications. Par rapport aux réseaux de capteurs souterrains actuels, qui utilisent des méthodes de communication filaires pour le déploiement du réseau, les RCSTSF ont plusieurs mérites remarquables tels que la dissimulation, la facilité de déploiement, l'actualité des données, la fiabilité et la densité de la couverture. Dans les applications agricoles, des capteurs souterrains sont utilisés pour surveiller les conditions du sol en termes d'eau et de teneur en minéraux. Les capteurs sont également utilisés avec succès pour surveiller l'intégrité des infrastructures invisibles tels que la plomberie. Les glissements de terrain ainsi que les tremblements de terre sont suivis à l'aide de sismomètres enterrés. Une autre application possible est la surveillance de la qualité de l'air dans les mines de charbon souterraines. L'accumulation du méthane et de monoxyde de carbone est un problème dangereux qui peut conduire à des explosions et incendies dans la mine. La présence de ces gaz représente un grand danger aussi pour la santé des mineurs et doit être suivie en permanence. Les capteurs peuvent également être utiles dans la surveillance de l'état des structures souterraines d'un bâtiment, d'un pont ou d'un barrage. Cela pourrait permettre un suivi en temps réel de l'usure dans un bâtiment et de prévenir des incidents catastrophiques.

1.2.4.4 Réseaux de capteurs corporels sans fil

Les Réseaux de Capteurs Corporels Sans Fil (RCCSF, *BSN : Wireless body sensor networks*) sont une autre extension des RCSF qui, à leur tour, ont évolué au cours des dernières années en raison de l'innovation significative dans les micro-capteurs, le traitement embarqué, ainsi que les technologies sans fil [29]. Un RCCSF est constitué de plusieurs micro-nœuds interconnectés, de faible puissance, qui peuvent à l'intérieur du corps, sur le corps, ou près du corps, surveiller les patients et leurs constantes, donner des traitements et communiquer les données aux centres de surveillances. Les RCCSF ont reçu un énorme intérêt de la part des industriels et les différents acteurs du secteur de la santé, en raison de l'étendu des applications envisageables et commercialisables dans ce secteur, telle la télé-médecine, les soins de santé à distance, les sports et le divertissement. L'intérêt pour ces réseaux vient en majorité du fait que les systèmes de santé traditionnels n'ont pas été conçus pour répondre à l'énorme flux de patients et sont ainsi insuffisants dans les scénarios actuels. Les chercheurs parlent même d'arriver à un point où les individus n'auront même pas conscience du fait qu'ils étaient à un certain moment malades. Beaucoup de travaux de recherches ont porté sur les technologies de communications sans fil où le corps est le medium, nommées « communications On-Body » [30], et d'autres reliant le capteur corporel à un dispositif externe, caractérisées de « communications Off-Body » [31].

1.2.4.5 Réseaux de capteurs multimédias sans fil

Les progrès récents dans la technologie des CMOS ont permis le développement de modules de caméras qui pourront facilement être intégrés à des émetteurs-récepteurs bons marchés. En outre, les microphones ont longtemps été utilisés comme une partie intégrante de nœuds capteurs sans fil. L'interconnexion des sources multimédia avec des appareils de communication peu coûteux a favorisé la recherche dans la mise en réseau de capteurs multimédia sans fil (RCMSF, *WMSN : Wireless Multimedia Sensor Networks*) [32]. En conséquence, les RCMSF sont devenus l'objet de recherches dans une grande variété de domaines. Le type du capteur utilisé détermine le type de données collectées ainsi que la catégorie du réseau. Quand seuls les microphones sont utilisés le réseau est nommé Réseaux de Capteurs Acoustiques Sans Fil (RCAcSF). L'utilisation de caméras a permis l'émergence de la famille des RCVisSF, qui sont divises en Réseaux de Capteurs Image Sans Fil (RCISF, *WiNS : Wireless image Sensor Networks*) et Réseaux de capteurs vidéo sans fil (RCVSF, *WVSN : Wireless Video Sensor Networks*) [19].

1.2.5 Architecture de RCSF

Il existe deux types d'architectures pour les réseaux de capteurs sans fil, à savoir : l'architecture plate et l'architecture hiérarchique [33].

1.2.5.1 L'architecture plate

Dans l'architecture plate, à l'exception du nœud puits qui joue le rôle d'une passerelle et qui est responsable de la transmission de l'information collectée à l'utilisateur final. Tous les autres nœuds sont identiques, ils ont la même capacité en termes d'énergie et du calcul, et qui sont responsables de la collection les informations. Un capteur peut se communiquer avec le puit selon deux manières soit

en 1-saut (one-hop) ou soit en multi-sauts (multi-hop) [34]. La simplicité présentée dans cette architecture permet à une basse latence de communication. De plus, lorsque le réseau devient plus dense, le problème du passage à l'échelle se pose, notamment en ce qui concerne le routage. La figure 1.5 présente les architectures plates des RCSF.

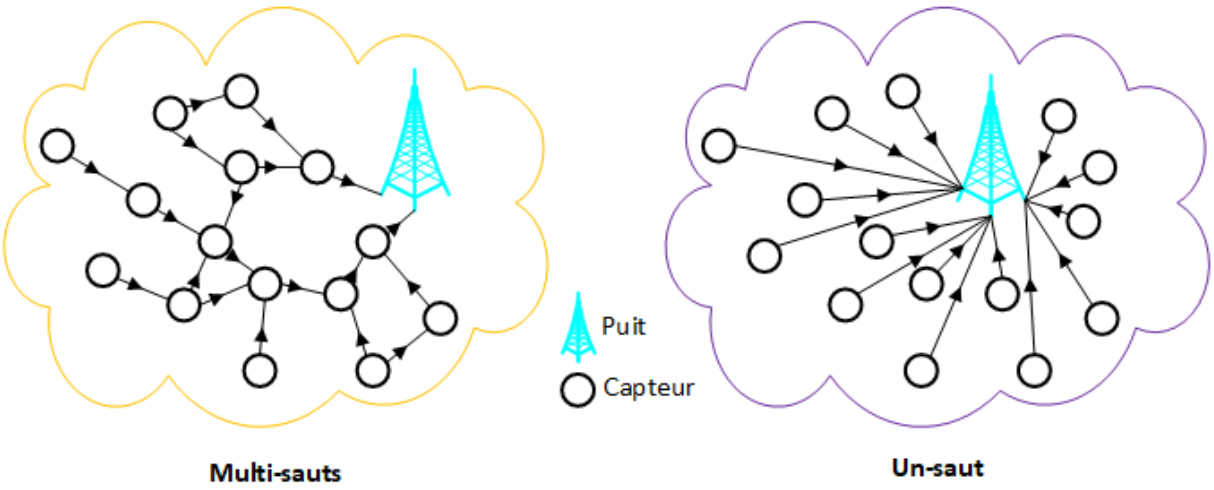

Figure 1. 5 Les architectures plates

1.2.5.2 L'architecture hiérarchique

Les architectures hiérarchiques sont utilisées dans les grands réseaux de capteurs [35]. Dans ces architectures, le réseau est divisé en plusieurs groupes (clusters) qui sont l'unité organisationnelle du réseau. Chaque cluster doit désigner un chef de groupe (*CH : Cluster Head*) qui est responsable de la coordination des capteurs sous sa responsabilité et agit comme passerelle vers un autre cluster. Le CH est responsable de l'agrégation et de la compression de toutes les données collectées afin de les acheminer vers le puits [36]. Cela permet de réduire la transmission des données dans le réseau. Cependant, en raison de la densité du réseau, il peut y avoir une haute latence dans les communications et de la consommation d'énergie plus élevée pour les CHs. La Figure 1.6 met en évidence les architectures en cluster pour les clusters à saut multiple et à saut unique.

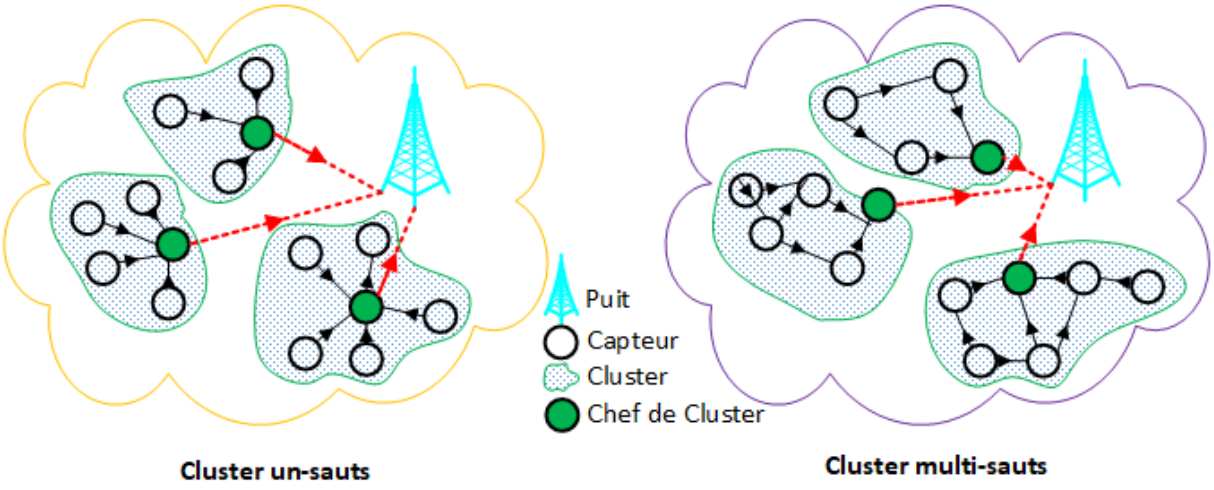

Figure 1. 6 Les architectures hiérarchiques

1.2.6 Domaine d'application

Les progrès récents dans la technologie des systèmes micro-électromécaniques (*Micro Electro-Mechanical Systems MEMS*) utilisés pour les capteurs, les communications sans fil, et l'électronique numérique ont permis le développement de capteurs très petits et intelligents. Ces avancées technologiques ont largement favorisé l'essor des RCSF pour de nombreuses applications et des contextes très variés. Les applications de RCSF peuvent être classées principalement en deux catégories, à savoir : la surveillance (*monitoring*) et la traçabilité (*tracking*) [37]. Dans la figure 1.7, nous résumons la classification des applications de RCSF.

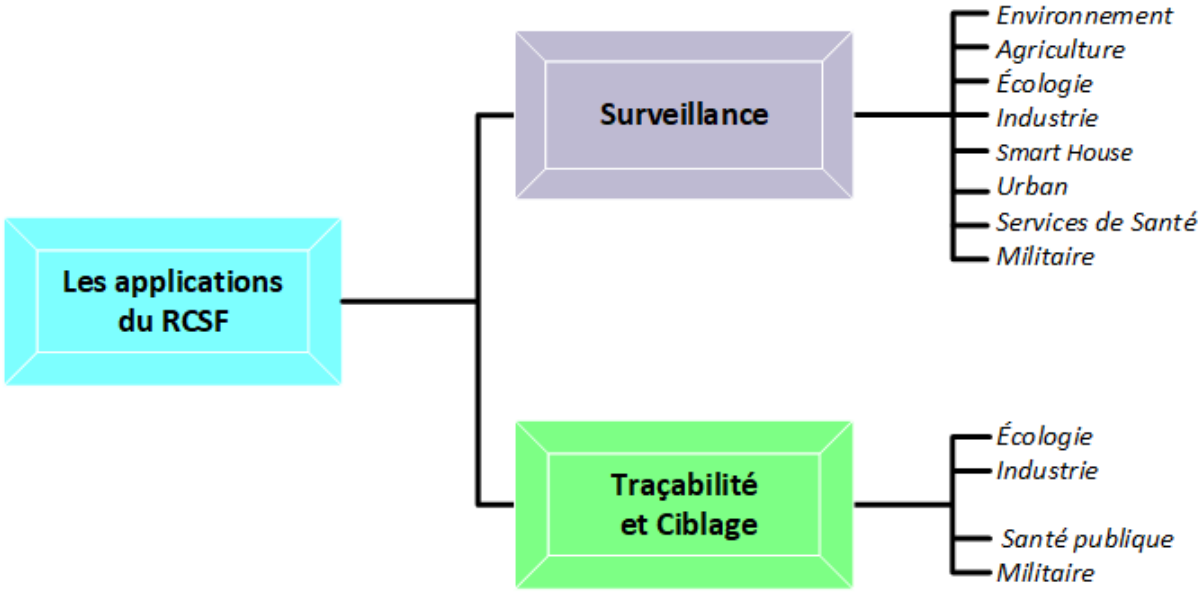

Figure 1. 7 Les classifications des applications des RCSF

1.2.6.1 La surveillance

La surveillance est utilisée pour analyser, superviser et contrôler attentivement les opérations d'un système ou d'un processus en temps réel [38]. Les applications de surveillance basées sur RCSF sont diverses. Ci-dessous certains d'entre eux sont brièvement présentés

1. *Environnement* : surveillance de la qualité de l'eau, des conditions météorologiques, de la pression, de la température, des vibrations, des phénomènes sismiques et des feux de forêt.
2. *Agriculture* : gestion de l'irrigation et surveillance de l'humidité.
3. *Écologie* : surveillance des animaux dans leur environnement naturel. Comme exemple, le projet présenté dans [39], étudie, en se basant sur un RCSF, les oiseaux de mer dans une réserve naturelle au Royaume-Uni.
4. *Industrie* : chaîne d'approvisionnement, surveillance de l'inventaire, procédés industriels et productivité. Comme exemple, la surveiller l'état d'un pont [40], de surveiller des installations oléoducs et gazoducs [41] et la vérification de la pression ou le débit circulant dans les tuyaux [42].
5. *Smart House* : surveillance de tout dispositif adressable dans la maison.
6. *Urban* : auto-identification, gestion de stationnement et transport.

7. *Santé* : surveillance d'opération chirurgicale, télésurveillance des patients à domicile [43]. Comme exemple, le projet Mercury [44] vise à surveiller les patients atteints de la maladie de Parkinson ou ceux qui souffrent d'épilepsie.
8. *Militaire* : détection d'intrusion. Comme exemple, le projet JBREWS (*Joint Biological Remote Early Warning System*) [45] qui a pour objectif de détecter et d'avertir les troupes sur l'utilisation d'armes biologiques, le projet WATS (*Wide Area Tracking System*) [46] consiste à utiliser un RCSF qui a pour objectif de détecter les rayons gamma et les neutrons dans le but de dépister des dispositifs nucléaires.

1.2.6.2 La traçabilité et ciblage

La traçabilité, la localisation et ciblage dans le RCSF sont généralement utilisés pour suivre un événement, une personne, un animal ou même un objet. Les applications existantes dans la traçabilité, la localisation et le ciblage peuvent être trouvées dans divers domaines.

1. *Industrie* : surveillance du trafic, détection de pannes.
2. *Écologie* : suivi de la migration des animaux dans divers domaines. Comme exemple, Le projet WildCENSE [47] consiste lui à suivre les déplacements des antilopes Nilgaud en Inde.
3. *Santé publique* : suivi des médecins et des patients dans un hôpital. Comme exemple, le projet CodeBlue [48] équiper des patients par des capteurs relevant des informations médicales telles que le niveau d'oxygénation et les pulsations cardiaques.
4. *Militaire* : un RCSF peut être déployé dans un champ de bataille ou une zone ennemie pour suivre, surveiller et localiser les mouvements de troupes ennemis. Comme exemple, un RCSF a été déployé dans la base militaire de MacDill (Air Force Base) à Tampa (Floride) pour détecter et suivre les mouvements d'objets mobiles intrus [49].

1.2.7 Les Défis du RCSF

Vu que les RCSF sont un type des réseaux Ad-hoc, donc les problèmes et les défis pour les réseaux Ad-hoc existent aussi dans les réseaux RCSF. De plus, le développement rapide de la technologie des capteurs sans fil est principalement dû aux différents besoins en termes d'applications. Le principal défi des RCSF est la conception d'un réseau en tenant compte d'une capacité énergétique limitée, contraintes de ressources, hétérogénéité, déploiement aléatoire et important, et un environnement dynamique et non contrôlé. Nous citions dans ce qui suit les défis critiques.

- La fiabilité et l'énergie : Permettre une communication fiable, tout en maintenant l'efficacité énergétique dans le réseau est un problème difficile dans RCSF puisque les capteurs sont très limités dans leur capacité de stockage d'énergie.
- Le débit : est une autre exigence importante de RCSF. En effet, la quantité de paquets reçus à la station de base permet d'évaluer la fiabilité de la livraison des données et donc le débit.
- La scalabilité : la capacité d'un RCSF à gérer un grand nombre de capteurs, qui est appelé la scalabilité, est une exigence critique dans la conception de réseaux de capteurs à grande échelle.

- La couverture du réseau, qui est une caractéristique importante dans les réseaux de capteurs, en particulier dans les réseaux RCSF nécessitant une haute disponibilité des informations collectées, est un autre problème difficile dans un RCSF statique.
- L'hétérogénéité : La coexistence de deux types de nœuds limités en ressources et de nœuds riches en ressources, introduit des nouveaux défis qui ne se posent pas dans les RCSF homogènes (au niveau des ressources disponibles sur les nœuds). Cette hétérogénéité est une question difficile exigeant la conception de nouveaux mécanismes qui prennent en compte cette caractéristique [50].
- Le routage et l'auto-organisation : Les applications visées par les RCSF sont diverses. Chaque application est susceptible d'exiger une solution de routage et de structuration différente [51,52]. Ainsi ces protocoles doivent être adaptés aux besoins de l'application tout en respectant les contraintes des ressources dont disposent les différents nœuds du réseau. Les principaux défis que rencontrent ces protocoles de routage et d'auto-organisation dans les RCSF sont [53]:

1.3 Les middleware

Les middleware (intergiciels) ont été largement utilisés avec les réseaux traditionnels. Cependant, Les intergiciels sont lourds en termes de mémoire et consomment beaucoup d'énergie, et par suite ils deviennent inadaptés pour les RCSF limités en termes d'énergie et de ressources comme la mémoire et la capacité du processeur. Les middleware conçus pour RSCF doivent obéir à certains principes de conception issus des particularités de ces réseaux pour obtenir un système qui fonctionne correctement. Nous identifions les principes suivants : gestion d'énergie et des ressources, traitement dans le réseau, support pour la QoS, connaissance de l'application, passage à l'échelle, topologie dynamique et tolérance aux fautes, adaptabilité, configurabilité et maintenance, intégration dans le monde réel, sécurité et hétérogénéité [54].

1.3.1 Définition :

Middleware est une couche logicielle qui se réside entre la couche d'application et le système d'exploitation (cf., la figure 1.8), qui réalisent les fonctions suivantes :

1. Cacher la *répartition*, c'est à dire le fait qu'une application est constituée de parties interconnectées s'exécutant à des emplacements géographiquement repartis ;
2. Cacher l'*hétérogénéité* des composants matériels, des systèmes d'exploitation et des protocoles de communication utilises par les différentes parties d'une application ;
3. Fournir des *interfaces* uniformes, normalisées, et de haut niveau aux équipes de développement et d'intégration, pour faciliter la construction, la réutilisation, le portage et l'interopérabilité des applications ;
4. Fournir un ensemble de *services* communs réalisant des fonctions d'intérêt général, pour éviter la duplication des efforts et faciliter la coopération entre applications.

Il comble la lacune entre les programmes de l'application et l'infrastructure matérielle et/ou logicielle de niveau inférieur. De plus, permettre et simplifie l'intégration des composants développés par plusieurs fournisseurs de technologie. Selon Miao-Miao Wang et al. [55], une solution complète de Middleware pour les RCSF doit inclure quatre composants principaux :

- *Les abstractions de programmation*, définissent l'interface du middleware au programmeur d'application.
- *Les services système*, fournissent les implémentations pour réaliser les abstractions.
- *Le support d'exécution*, sert d'extension au logiciel d'exploitation embarqué pour supporter les services du middleware.
- *Les mécanismes de QoS*, définissent les contraintes de qualité de service du système.

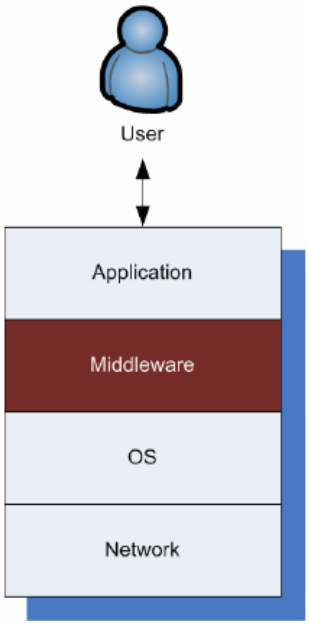

Figure 1.8 L'architecture de coche middleware dans un système distribué

L'objectif principal d'un middleware consiste à supporter le développement, la maintenance, le déploiement et l'exécution des applications utilisant les RCSF [56]. Ceci inclus entre autres des mécanismes pour la formulation des tâches complexes de capture de haut niveau, la communication de ces tâches au RCSF, la coordination des nœuds capteurs pour l'accomplissement d'une tâche et sa distribution aux nœuds capteurs individuels, la fusion de données. Des abstractions et des mécanismes appropriés pour traiter l'hétérogénéité des nœuds capteurs doivent également être fournis. Tous les mécanismes fournis par un middleware doivent respecter les principes de conception et les caractéristiques des RCSF qui se concentrent souvent autour de l'économie d'énergie, la robustesse et le passage à l'échelle. Un RCSF traite des données du monde réel, ainsi les concepts de temps et de localisation jouent un rôle beaucoup plus important que dans le cas des systèmes de calcul traditionnels. Par conséquent, la gestion du temps et la localisation doit être impérativement intégrée dans toute infrastructure de middleware conçue pour les RCSF.

1.3.2 Classification des approches middleware

Les intergiciels conçus pour les RCSF peuvent être classés selon les approches suivantes :

1.3.2.1 Approche base de données :

Dans cette approche, le RCSF est vu comme une base de données. Il peut être interrogé par des requêtes écrites en des langages similaires à SQL, comme s'il s'agit de tables relationnelles ordinaires. Cette approche est data-centric naturellement où l'information est centrée autour des données et non pas autour des identifiants des capteurs, par contre elle n'est pas adaptative ou configurable. SINA [57], Cougar [58] et TINYDB [59] sont des exemples d'intergiciels fondés cette approche.

1.3.2.2 Approche basée sur les évènements :

Dans cette approche, le middleware supporte les communications asynchrones, où les capteurs se réveillent quand un évènement surgit et peuvent s'endormir le reste du temps. Cette approche est adaptée aux RSCF qui génèrent un bas débit la plupart de temps, et où des évènements brusques puissent parvenir parfois, comme par exemple les applications de surveillance. Mires [60] et DSWare [61] sont des exemples d'intergiciels fondés cette approche.

1.3.2.3 Approche agents mobiles :

Dans cette approche, la station de base envoie un agent mobile à une région donnée pour visiter les nœuds un par un. Les données captées sont réduites et agrégées par l'agent mobile avant leur envoi à la station de base, ce qui économise de l'énergie et réduit l'utilisation de la bande passante. Comme les agents mobiles sont envoyés là où il faut, ce qui dépend de la topologie actuelle du réseau. Cette approche supporte le principe de topologie dynamique et de la tolérance aux fautes. Agilla [62] est un exemple d'intergiciel fondé sur cette approche.

1.3.2.4 Approche orientée application :

Cette approche favorise le contrôle du réseau par l'application à travers une architecture qui touche le niveau réseau, permettant ainsi aux programmeurs de régler le réseau suivant les besoins de l'application. Cette approche produit un middleware plus personnalisé et plus spécifique à une application donnée. Cela offre un compromis entre un middleware général et un middleware spécifique. MiLAN [63] et TinyCubus [64] sont des exemples d'intergiciels fondés cette approche.

1.3.2.5 Approche modulaire :

Cette approche consiste à construire une application à partir d'un ensemble de services implémentés sous forme de petits modules qui interagissent à travers leurs interfaces. Cette approche réduit la complexité du système pour l'utilisateur, puisque l'implémentation des services peut changer à n'importe quel moment alors que leurs interfaces restent intactes. La modularité des services facilite aussi la configurabilité et de légères mises à jour logicielles, ce qui économise beaucoup d'énergie. Impala [65] est un exemple d'intergiciel fondé sur cette approche.

1.3.2.6 Approche machine virtuelle :

Les machines virtuelles sont utilisées d'habitude pour la simulation de matériel, représentation de programmes intermédiaires ou interprétation de bytecode. Dans les RCSF, ces machines sont utilisées pour représenter un grand nombre d'applications à l'aide d'un ensemble simple d'instructions légères.

Une machine virtuelle offre un environnement sécurisé pour le système et interdit aux mauvais fonctionnements des applications d'affecter les nœuds. Maté [66] est un exemple d'intergiciel fondé sur cette approche.

1.3.3 Les principes de conception d'une approche middleware

Vu que les capacités d'un nœud capteur sont limités, la conception d'un middleware pour RCSF doit respecter beaucoup de principes pour fonctionner correctement dans un environnement RCSF. Chaque approche de middleware qui a été présentée dans la section précédente, peut soutenir certains principes de conception en fonction de sa nature de conception (cf., le tableau 1.1). Dans ce qui suit, nous présentons certains de ces principes [54] :

- *Data centricity* : un middleware conçu pour WSN devrait prendre en charge la centralisation des données en fournissant un routage centré sur les données et des interrogations au sein du réseau. Des approches telles que publier / souscrire et les modèles de type base de données peuvent convenir dans ce cas.
- *La Gestion de l'énergie et des ressources* : le middleware doit être léger, économe en énergie, gérer intelligemment des ressources limitées afin de fournir les services requis tout en maximisant la durée de vie de l'appareil.
- *In-network processing* : Le middleware devrait fournir des algorithmes d'agrégation de données et de compression de données donnant aux applications la capacité de choisir l'algorithme qui leur convient le mieux. En outre, il peut être utile de laisser l'application injecter ses propres fonctions de traitement en réseau dans le réseau.
- *Les supports de QoS* : Le middleware devrait fournir des mécanismes pour améliorer la probabilité de détection d'un évènement et le taux de fiabilité et aussi résoudre les problèmes de la couverture et du déploiement.
- *La Connaissance de l'application* : le middleware doit fournir à l'utilisateur la possibilité de choisir le contexte souhaité d'une application qu'il veut utiliser à un moment donné.
- *L'extensibilité* : Middleware doit supporter l'extensibilité (*Scalability*). En outre, il devrait fournir à l'utilisateur une interface conviviale qui permet d'ajouter, déplacer ou supprimer des nœuds du réseau
- *Topologie de réseau dynamique* : Le middleware doit tolérer les échecs ou les changements de topologie, en fournissant des routes différentes entre les nœuds.
- *Adaptabilité, Configurabilité et maintenabilité* : Les intergiciels conçus pour WSN devraient fournir des mécanismes adaptatifs aux applications, ce qui leur permet d'exprimer leurs besoins en fonction des changements survenus dans le réseau, et aussi doit avoir la capacité de faire des mises à jour facile aux logicielles et de la maintenance nécessaire aux nœuds de capteurs.
- *Hétérogénéité* : Un middleware devrait profiter l'hétérogénéité dans RCSF, afin de prolonger la durée de vie du réseau.

Principes de conception / **les catégories des middleware**	Database			Event-based		Tuple space	Application		Modular	Mobile Agents	Virtual Machine
	SINA	TinyDB	Cougar	Mires	DSWare	TinyLIME	MiLAN	TinyCubus	Impala	Agilla	Maté
Data-centricity	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
Gestion de l'énergie et des ressources	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Traitement en réseau	Y	Y	Y	Y	Y	Y	N	Y	N	Y	N
Support de QoS	N	N	N	N	Y-	N	Y	Y-	N	N	N
La Connaissance de l'application	N	N	N	N	N	N	Y	Y	Y-	Y	N
L'extensibilité	Y	N	Y-	Y-	N	N	N	N	Y	Y	N
Topologie de réseau dynamique	Y-	Y	N	N	Y	Y	Y	Y	Y-	Y	N
Adaptabilité	N	Y-	N	N	N	N	Y	Y	Y	N	N
Configurabilité et maintenabilité	N	N	N	N	N	N	N	Y	Y	N	Y
Modèle d'activité	Y	Y	N	Y	Y	N	N	Y	Y	Y	N
Intégration du monde réel	Y-	Y	Y	N	N	Y	N	N	Y	Y	Y
Sécurité	N	N	N	N	N	N	N	N	Y-	N	Y-
Hétérogénéité	N	Y	N	N	N	N	N	Y	Y-	N	Y

Tableau 1.1 Comparaison entre les middleware selon leurs principes de conception

1.4 Conclusion

Dans ce chapitre, nous avons présenté brièvement en premier lieu les concepts de base liés aux réseaux de capteurs sans fil, à savoir les différentes architectures d'un nœud capteur, la description d'un RCSF homogène et de RCSF hétérogène, et aussi les différentes familles des RCSF. Nous avons introduit en plus leurs architectures, leurs applications dédiées et leurs défis. Deuxièmement, nous avons introduit le middleware, leurs principes de conception et les différentes approches de middleware conçus pour RSCF, à savoir approche base de données, approche basée sur les évènements, approche agents mobiles, approche orientée application, approche modulaire, et approche machine virtuelle. Cette vue d'ensemble de ces approches nous permet de conclure que presque toutes les approches offertes la centralité les données et traitaient les problèmes de la consommation d'énergie et des ressources limitées, et presque aucun d'entre eux pris en charge l'autonomiste, l'optimisation, la (re)configurabilité, l'hétérogénéité, et le problème de sécurité. Dans le chapitre suivant, nous présentons les différentes solutions biologiques proposées dans la littérature portant sur l'autonomie, adaptation et l'optimisation dans les RCSF.

Chapitre 2

Les solutions bio-inspirées

2.1 Introduction

Comme nous venons de voir dans le chapitre précédent, les RCSF sont des réseaux autonomes, spontanés et multi-sauts, où les nœuds collaborent afin d'assurer le bon fonctionnement du réseau, et aussi leurs défis qui se sont liés aux caractéristiques de capteurs. D'autres défis sont liés à la gestion autonome « Self-Management » du RCSF. Le défi majeur est de concevoir des solutions efficaces qui permettent aux RCSF de se gérer d'une manière autonome sans l'intervention d'un utilisateur (superviseur), surtout, dans les cas les réseaux sont déployés dans des zones inaccessibles et/ou hostiles. Ceci nécessite donc que le réseau possède des fonctions suivantes [67]:

- L'auto-configuration (Self-configuration) : est la capacité d'adapter les paramètres de la configuration du réseau en fonction de son état et de son environnement.
- L'auto-optimisation (Self-optimization) : est la capacité de monitorer et d'optimiser l'utilisation des ressources limitées du réseau.
- L'auto-réparation (Self-Healing) : est la capacité de découvrir, d'identifier la cause et de réagir aux pannes et défaillances du réseau
- L'auto-protection (Self-protection) : est la capacité de reconnaître les intrusions et attaques et de s'en protéger.

En effet, ces fonctions se trouvent dans la nature, à savoir, les systèmes biologiques, écologique … donc, les solutions bio-inspirées sont progressivement appliquées dans les RCSF, comme l'intelligence en essaim (SI), le système Immunitaire Artificiel (AIS), les algorithmes heuristiques comme l'algorithme Génétique (GA) et les réseaux de neurones artificiel (NNA).

Dans ce chapitre nous commençons par introduire et définir les fonctions principales de l'auto-gestionnaire d'un système, nous présenterons les techniques bio-inspirés existantes dans la littérature. On s'intéressera par la suite aux méthodes adaptées au problème de l'auto-géstionnaire des RCSF, et en fin nous donnons un résumé sur les domaines d'application des solutions bio-inspirées dans RCSF.

2.2 Les fonctions principales, rôles et objectifs

2.2.1 L'auto-organisation

Un système est dit auto-organisé, s'il est organisé sans aucune entité externe et sans contrôle centralisé. Ainsi, les entités, qui constituent le système, doivent interagir localement avec d'autres entités d'une façon distribuée [68].

La notion d'auto-organisation a aussi trouvé sa place dans le domaine de la communication, des réseaux informatiques et des réseaux RCSF. Dans ce dernier les nœuds capteurs interagissent directement les uns avec les autres d'une manière distribuée avec un objectif commun. Ainsi, l'auto-organisation dans les RCSF est un processus duquel émerge une structure globale provenant seulement des interactions locales entre les nœuds du réseau [69]. Donc le résultat (rôle) de l'auto-organiser un réseau RCSF est création une topologie logique au-dessous de la topologie physique pour assurer des objectifs comme :

- Minimiser la consommation énergétique au niveau du réseau
- Améliorer les performances du réseau
- Partager et gérer les ressources disponibles dans le réseau

2.2.2 L'auto-configuration

L'auto-configuration est la capacité d'un système à déterminer de façon autonome la meilleure configuration d'un élément et modifier les paramètres de configuration sans aucune intervention humaine.

Les RCSF doivent être auto-configurés. Cela signifie que les nœuds doivent être autonomes et pouvoir se configurer pour s'adapter aux changements de l'environnement qu'ils sont déployés. C'est-à-dire chaque nœud doit faire les opérations suivantes :

- Obtenir toutes les informations nécessaires à sa connexion à un réseau RCSF.
- Chercher la meilleure configuration correspondante à son état actuelle.
- Modifier et s'adapter ses propres paramètres, (e.g. mette à jour la table de routage, ajuster le cycle d'activation et la période de sommet, etc.)

De plus, le réseau RCSF peut être statique, dynamique ou mobile, et les nœuds peuvent apparaître et disparaître fréquemment en raison des défaillances de nœuds, des dommages, de l'addition, de l'épuisement d'énergie ou de la disparition du canal. Pour permettre un fonctionnement continu et fiable du réseau, les nœuds doivent s'auto-configurer pour répondre à ces contraintes. Donc le résultat de l'auto-configurer un réseau RCSF est création une structure (ou modèle) souhaitable.

2.2.3 L'auto-optimisation

Le terme d'optimisation en informatique, désigne la procédure de recherche de la meilleure configuration pour laquelle un système informatique trouve ses performances optimales. L'auto-optimisation est donc, la capacité de rendre cette procédure de recherche de la meilleure configuration complétement automatique (i.e. sans intervention humaine).

Le réseau RCSF était un domaine fertile pour l'auto-optimisation, par exemple les paramètres de configuration du réseau doivent être automatiquement et continuellement adaptés au profil du trafic dans le réseau et au les conditions dynamiques de l'environnement où le réseau est déployé. L'auto-optimisation de la consommation en énergie a toujours été un domaine d'actualité.

Généralement, l'optimalité d'une caractéristique du système se mesure par un indice de performance à optimiser, c'est-à-dire à maximiser ou minimiser et cela en fonction de l'objectif de l'optimisation, on peut s'intéresser dans un réseau RCSF à minimiser la latence et le coût (la consommation d'énergie), maximiser le débit et le taux de réussite, etc... [70].

2.2.4 L'auto-réparation

Un système d'auto-réparation devrait se rétablir de l'état anormal (ou « malsain ») et revenir à l'état normatif (« sain »), et fonctionner tel qu'il était avant la perturbation [71,72]. Dans le réseau RCSF L'auto-réparation c'est un service de gestion qui découvre, diagnostique et réagit aux perturbations du réseau. Les composants d'auto-réparation détectent les opérations et les échecs incorrects et lancent

des actions correctives basées sur des stratégies définies pour récupérer le réseau ou un nœud. La récupération automatique des dommages améliore la disponibilité du service. La procédure de l'auto-réparation est une boucle qui contient trois étapes suivantes [73]:

- *Détection* : à cette étape, toute information suspecte reçue des nœuds est filtrée pour identifier les informations malveillantes ou les attaques
- *Diagnostique* : sur la base des attaques ou des menaces identifiées, l'analyse de la cause profonde est effectuée, puis un plan de récupération approprié est préparé selon les politiques prédéfinies (la base de règles)
- *Récupération* : les adaptations planifiées sont ensuite soigneusement appliquées au système de telle sorte que les contraintes des capacités du système soient satisfaites et que tout effet secondaire imprévisible soit évité.

2.2.5 L'auto-protection

L'auto-protection : anticiper, détecter, identifier et protéger contre les menaces (internes ou externes, accidentelles ou malveillantes) de n'importe où. En cas d'attaque, ce service exécute des routines de détection afin d'atteindre la sécurité.

L'auto-protection dans RCSF concerne l'utilisation de nœuds de capteurs pour fournir une protection à eux-mêmes, afin qu'ils puissent détecter et contrecarrer les attaques ciblant directement ces nœuds. Un RCSF est dit *P-auto-protection*, si à tout moment, au moins il y a *P* capteurs actifs qui peuvent surveiller tout nœud de capteur (actif ou non-actif) de ce RCSF. Un nœud actif est celui qui a la capacité d'effectuer des protections ; sinon il est appelé un capteur non-actif [74,75].

2.3 Les approches (techniques) bio-inspirées

2.3.1 Le Réseau de Neurones Artificiels

Le principe général des Réseaux de Neurones Artificiels (RNA, *ANN : Artificial neural network*) est à l'origine inspiré de certaines fonctions de base des neurones naturels du cerveau [76] (cf., la figure 2.1). Ces derniers sont apparus dans les années cinquante avec les premiers perceptrons, et sont utilisés industriellement depuis les années quatre-vingt [77].

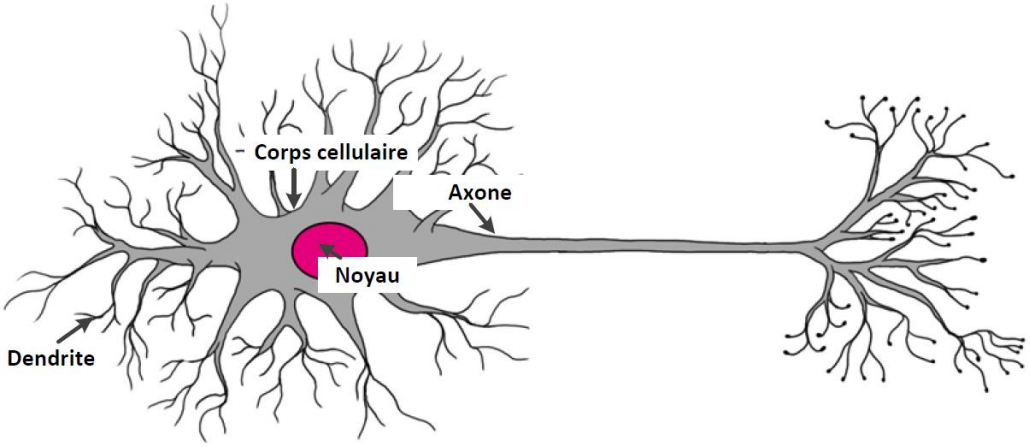

Figure 2.1 la structure d'un neurone biologique [76]

Les réseaux de neurones sont des outils très utilisés pour la classification, l'estimation, la prédiction et la segmentation. Ils sont issus de modèles bio-inspirés et sont constitués d'unités élémentaires (les neurones) organisées selon une architecture spécifique [77]. Les réseaux de neurones artificiels sont des modèles de calcul dont la conception est schématiquement inspirée du fonctionnement cérébral et des structures neuronales. Ils sont généralement optimisés par des méthodes d'apprentissage de type statistique qui leur permettent de prendre des décisions en s'appuyant davantage sur la perception que sur le raisonnement logique formel [78].

Un réseau de neurones artificiel est constitué de plusieurs neurones connectés via des liaisons leur permettant d'envoyer et de recevoir des signaux en provenance des neurones qui les précèdent. Chacune de ces connexions reçoit une pondération w_i qui détermine son impact sur les neurones qu'elle connecte. Chaque neurone dispose ainsi d'une entrée, qui lui permet de recevoir de l'information d'autres neurones, mais aussi d'une fonction d'activation φ et enfin d'une sortie comme le montre la Figure 2.2.

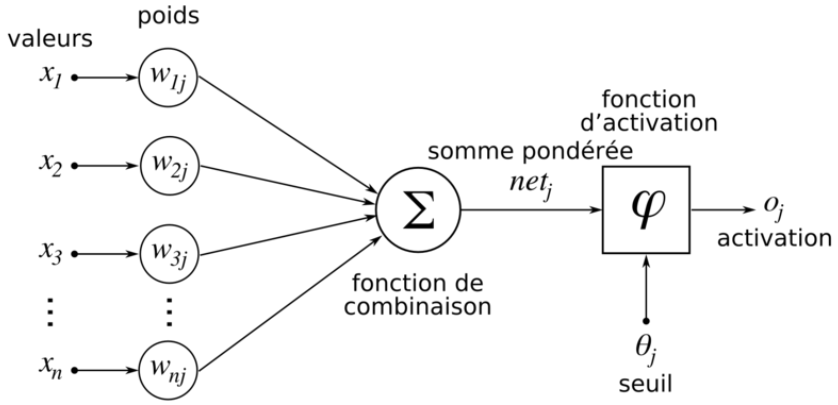

Figure 2.2 Modèle d'un neurone artificiel *j*

Selon, un neurone artificiel « *j* » est constitué d'un intégrateur effectuant la somme pondérée de toutes les entrées « x_{ij} ». Le résultat est ensuite calculé par la fonction de transfert (d'activation) φ qui fournit la sortie du neurone notée o_j. L'intégrateur délivrant la somme pondérée « net_j » des entrées x_{ij} peut être décrit comme suit :

$$net_j = \sum_{i=1}^{N} \left(\omega_{ij} x_i - \theta_j \right) \tag{1.1}$$

Le résultat net_j de la somme pondérée des entrées x_{ij} est appelé "niveau d'activation". Ce niveau varie en fonction de la valeur du seuil Θ_j (appelé aussi, biais du neurone). La fonction d'activation du neurone « φ » peut, alors, être représentée par l'équation suivante :

$$o_j = \varphi(net_j) = \varphi\left(\sum_{i=1}^{N} \left(\omega_{ij} x_i - \theta_j \right) \right) \tag{1.2}$$

Il existe plusieurs fonctions d'activation, les plus utilisées restent : seuil et linéaire. Les fonctions d'activation les plus connues sont comme suit :

- *Seuil* : l'équation qui définit cette fonction est comme suit :

$$o_j = \begin{cases} 0 & si \quad net_j \prec 0 \\ 1 & si \quad net_j \geq 0 \end{cases} \tag{1.3}$$

Il existe plusieurs variantes de cette fonction telle que le seuil symétrique.

- *Linéaire* : l'équation qui définit cette fonction est comme suit :

$$o_j = net_j \tag{1.4}$$

Il existe plusieurs variantes de cette fonction telles que : linéaire saturée, linéaire saturée symétrique et linéaire positive.

- *Sigmoïde* : l'équation qui définit cette fonction est comme suit :

$$o_j = \frac{1}{1+e^{-S}} \tag{1.5}$$

- *Tangente hyperbolique :*

$$o_j = \tanh(n) = \frac{\sinh(n)}{\cosh(n)} = \frac{e^n - e^{-n}}{e^n + e^{-n}} \tag{1.6}$$

2.3.2 Les Algorithmes Evolutionnaires

Les phénomènes physiques et biologiques ont inspiré un bon nombre d'algorithmes dans le domaine de l'optimisation combinatoire. Une multitude d'algorithmes bio-inspirés ont montré leur robustesse face à des problèmes complexes. Les algorithmes évolutionnaires (EA) sont l'un des exemples d'algorithmes bio-inspirés qui peuvent traiter des problèmes NP-Difficile [79]. Les EA sont basés sur la théorie de l'évolution proposée par Darwin en 1807. L'idée principale consiste à ce que l'apparition des espèces adaptées est une conséquence de deux phénomènes principaux :

(1) La sélection naturelle (les individus les plus adaptés survivent et se reproduisent).

(2) Plusieurs variations peuvent se produire sur le matériel génétique des espèces.

En termes d'optimisation, l'évolution se traduit par un processus itératif de recherche de l'optimum dans l'espace de recherche. D'après [80] on peut dire qu'il existe une multitude de variante au sein des algorithmes évolutionnaires, qui se sont développées de manière indépendante : La Programmation Evolutionnaire (EP) parue dans les années 60 [81], les Algorithmes Génétiques (GA) introduits par J. Holland [82] en 1975, ces derniers ont été promus par son élève D. E. Goldberg [83] dix ans plus tard (1989), Les Stratégies d'Evolution (SE) [84], et finalement la programmation génétique proposée par J.Koza dans ces deux articles [85,86].

Soit F une fonction à optimiser (minimiser-maximiser), définie sur un espace. Dans un vocabulaire d'EA on notera :

- La fonction F est appelée fonction de fitness ou fonction d'adaptation. Chaque individu dispose de sa propre valeur de fitness ou d'adaptation.

- Les points de l'espace de recherche sont appelés individus et peuvent être représentés par un ou plusieurs chromosomes.
- Un chromosome représente une solution ; il peut être codé par des valeurs binaires, réelles.
- Un ensemble *P* d'individus est appelé population
- Chaque itération de l'algorithme produit un nouvel ensemble de solutions appelé nouvelle génération
- Les individus de la Nième génération sont appelés parents et ceux de la (N+1)ieme génération sont appelés enfants
- À chaque génération les parents sont recombinés pour générer de nouveaux enfants (nouvelles solutions)

L'architecture typique d'un algorithme évolutionnaire est représentée dans la figure 2.3. Nous pouvons distinguer les étapes les plus importantes de l'algorithme citées auparavant (sélection, reproduction, évaluation). D'une manière générale, la population initiale (solutions de départ) est générée de manière aléatoire. Une fois ces individus générés, ils passent par une multitude d'opérations (sélection et reproduction) afin de trouver une solution à un problème donné. Les principales étapes d'un EA, à savoir : sélection, reproduction, évaluation, sont répétées jusqu'à l'atteinte d'un critère d'arrêt (ex. : nombre maximum de générations).

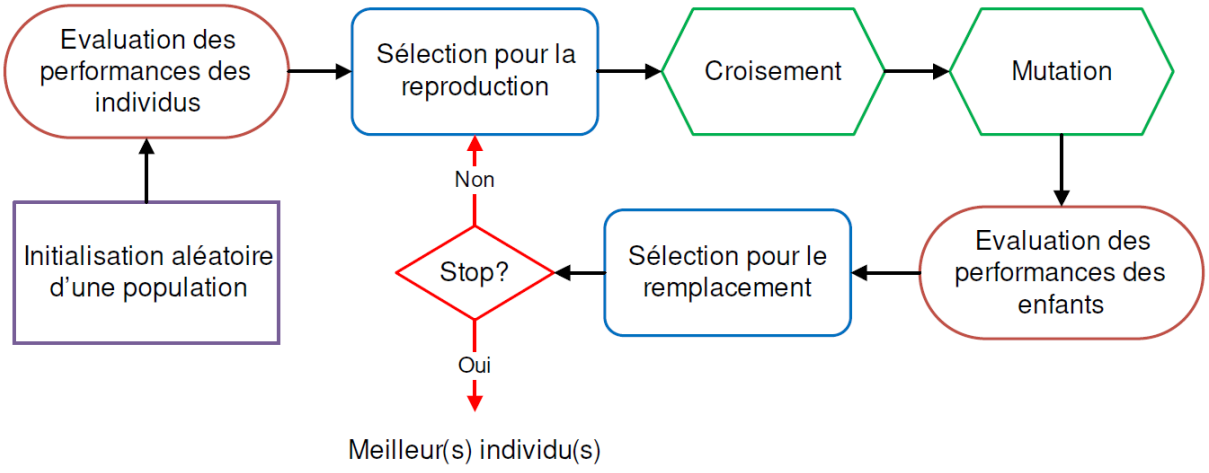

Figure 2.3 Principe d'un algorithme évolutionnaire (EA)

2.3.3 L'Intelligence en Essaim

L'intelligence en essaim (*SI : Swarm Intelligence*) est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [87]. Elle recouvre un ensemble d'algorithmes, à base de population d'agents simples (entités capables d'exécuter certaines opérations), qui interagissent localement les uns avec les autres et avec leur environnement. Ces entités, dont la capacité individuelle est très limitée, peuvent conjointement effectuer de nombreuses tâches complexes nécessaires à leur survie. Bien qu'il n'y ait pas de structure de contrôle centralisée qui dicte la façon dont les agents individuels devraient se comporter, les interactions locales entre les agents conduisent souvent à l'émergence d'un comportement collectif global et auto-organisée. Trois

exemples phares d'algorithmes de l'intelligence en essaim sont les algorithmes de colonies de fourmis, les algorithmes de colonies d'abeilles et les algorithmes d'optimisation par essaim particulaire [88].

2.3.4 Le système immunitaire artificiel

L'immunitaire biologique représente une source riche d'inspiration pour les chercheurs qui la puisent pour développer des applications artificielles destinées à résoudre des problèmes, ce qui conduit à l'émergence d'une nouvelle discipline, celle des systèmes immunitaires artificiels (*Artificial Immune System", AIS*) [89].

Le terme système immunitaire artificiel [90] s'applique à une vaste gamme de systèmes différents, notamment aux métaheuristiques d'optimisation inspirées du fonctionnement du système immunitaire. Dans la littérature, plusieurs définitions d'AIS existent comme dans [91] Les AIS sont des systèmes informatiques basés sur des métaphores du système immunitaire naturel, [89] les AIS sont des systèmes adaptatifs, inspirés de l'immunologie théorique et de l'observation des fonctions, des principes et des modèles immunitaires, afin d'être appliqués à la résolution de problèmes [92].

AIS s'inspire essentiellement des sélections opérées sur les lymphocytes, et des processus permettant la multiplication et la mémoire du système. En effet, ces caractéristiques sont capitales pour maintenir les propriétés auto-organisées du système. Cette approche est assez similaire à celle des algorithmes évolutionnaires. Elle peut également être comparée à celle des réseaux de neurones. Dans le cadre de l'optimisation, on peut considérer les AIS comme une forme d'algorithme évolutionnaire présentant des opérateurs particuliers. Par exemple, l'opération de sélection est basée sur une mesure d'affinité (i.e. entre le récepteur d'un lymphocyte et un antigène). La mutation s'opère, quant à elle, via un opérateur d'hyper-mutation directement issu de la métaphore.

Dans ce sens *De Castro et Timmis* proposent un cadre « *Framework* » pour l'ingénierie d'un AIS. Ce cadre repose sur les éléments suivants :

- *Une représentation pour les composants du système* : cette représentation sert à créer des modèles des organes, cellules et molécules immunitaires.
- *Un ensemble des mécanismes pour évaluer les interactions des composants aves l'environnement et entre eux :* l'environnement est généralement simulé par un ensemble de *stimul* d'entrée. Les interactions sont quantifiées par une ou plusieurs fonctions, nommées « *fonctions d'affinité* »
- *Des procédures d'adaptation* : qui gouvernent la dynamique du système, c'est-à-dire comment son comportement varie dans le temps

La famille des AIS peut être divisée en quatre types d'algorithmes : (1) les algorithmes de sélection négatifs (*Negative Selection Algorithm*) [93], (2) les réseaux immunitaires artificiels (*Immune Network Algorithm*) [94], (3) les algorithmes de sélection clonale (*Clonal Selection Algorithm*) [95], (4) la théorie du danger [96] et les algorithmes de cellules dendritiques [97]. Une discussion détaillée de ces algorithmes peut être trouvée dans [98].

2.4 L'analyse comparative aux techniques bio-inspirées

Tant que, l'efficacité de l'application une technique bio-inspirée (solution biologique) pour résoudre les divers problèmes du RCSF, dépend du type et de la gamme du problème. Donc le choix de la technique bio-inspirée approprié est très importante. Par exemple, le problème de routage est mieux résolu par l'intelligence en essaim (SI) car il traite des propriétés d'adaptatives, robustes, évolutives et des distributives, et aussi les algorithmes sont inclus dans SI, comme, ABC, ACO et PSO, qui sont des solutions centralisées (cf., la section suivante). D'autre part, le temps d'exécution, malgré l'AIS est un outil plus performant que AG, mais il traite de très peu de solutions dans RCSF, car les calculs de l'AIS exigent plus de temps d'exécution. Donc, par conséquent, si une technique est forte dans un aspect, elle peut être faible dans un autre aspect. Pour cette implication, la nature et les caractéristiques du problème ciblé doivent être soigneusement identifiées pour être résolue par la technique bio-inspirée approprié. Pour rendre le choix facile, il faut comparer et analyser les paradigmes des algorithmes de ces technique bio-inspirée, qui résolvent les défis de RCSF, les auteurs de [99] ont mis les paramètres d'évaluations, à savoir, les types de variables, les point de recherche, le temps d'exécution, les problèmes ciblés et la fonctionnalité. Dans le tableau 2.1 nous présentons les caractéristiques de base de quatre techniques bio-inspirées citées auparavant. Parfois, les chercheurs créent des technique hybride bio-inspirées qui se composent au moins de deux techniques afin d'assurer une meilleure solution et augmenter les performances, parmi ces techniques on peut citer comme des exemples, *Neuro-immune* [100], *Immune-swarm* [101], *Neurogenetic-swarm* [102] , *Swarm-fuzzy* [103] et *Genetic-neural* [104].

Paramètres d'évaluation	**(ANN)**	**AIS**	**GA**	**SI**
L'année de développement	1969	1986	1960	1990
Les types de variables	Mixtes	Continue et discrètes	Discrètes	Mixtes, continues et discrètes
Les points de recherche	Multipoint	Multipoint	Multipoint	Multipoint
La garantie de la solution	Rarement offre une solution complète	Le meilleur pour la solution variant dans le temps	Défini de manière favorable	Précis
Le temps d'exécution	Longue	Moyen	Moyen	Moyen
Les Problèmes ciblés	– Opt. combinatoire – Opt. multi-objectif	– Opt. combinatoire – Opt. multi-objectif – Opt. Continue	– Opt. combinatoire – Opt. globale – Opt. non linéaire	– Opt. combinatoire – Opt. continue – Opt. non linéaire
Fonctionnalités	– Apprentissage adaptatif – Auto-organisation – Tolérance de panne	– Fournit des outils de recherche locale et globale – Outil d'optimisation puissant – Auto-adaptatif et auto-apprentissage	– Solutions aux problèmes d'optimisation – Bonnes solutions globales	– Approche distributive – Auto-organisation – Contrôle décentralisé – Outil d'optimisation puissant

Tableau 2.1 Analyse comparative de diverses techniques bio-inspirées [6]

2.5 Les solutions bio-inspirées pour RCSF (l'état de l'art)

« Nous croyons que les défis rencontrés par les futures applications du réseau, telles que l'extensibilité (scalabilité), l'adaptabilité et la capacité de survivabilité/disponibilité, ont déjà été surmontés par des systèmes biologiques à grande échelle et que les futures applications du réseau bénéficieront en s'adaptant les principes et les mécanismes biologiques fondamentaux » [105].

Vu que les capacités du nœud capteur sont très limitées, on a besoin des techniques d'optimisation légères que l'on peut implémenter sur un nœud de capteur, bien que certaines techniques d'optimisation traditionnelles (e.g., les technique analytique) soient très efficaces, mais elles nécessitent une capacité de calcule très élevé, une capacité de traitement très élevé et un grand temps d'analyse, ce qui cause le problème d'épuisement de l'énergie du nœud plus rapidement [106]. Les méthodes d'optimisation bio-inspirées sont légers, ce qui en fait des alternatives efficaces aux méthode d'optimisations traditionnelles. De plus, la plupart des approches bio-inspirées visent à surmonter les défis de RCSF comme la scalabilité, l'adaptabilité et la survivabilité, etc. dans ce qui suit, nous donnerons un aperçu de l'état de l'art de solutions bio-inspirées et les domaines d'application dans RCSF.

Définition : *Les solutions bio-inspirées, aussi appelées solutions inspirées de la nature, sont des solutions métaheuristiques basées sur le comportement collectif des communautés sociales individuelles. Ces solutions fournissent des outils et des algorithmes efficaces qui traitent beaucoup de propriétés souhaitables et intéressantes appliquées dans les réseaux RCSF,* [2,3,12]*. Ces inspirations biologiques sont utilisées pour rendre les RCSF plus fiable, efficace, auto-organisés, auto-entretenues et auto-régulés* [4,107].

La solution bio-inspirée est un résultat de l'application l'une des techniques bio-inspirée citant dans la section précédente, ou l'application de l'hybridation de deux techniques, tel que les deux techniques sont bio-inspirées ou l'une bio-inspirée et l'autre non (e.g., hybride l'algorithme évolutionnaire par la logique floue [108]. Ipek Caliskanelli [109] a proposé une catégorisation des solutions bio-inspirées selon leurs inspirations biologiques et leurs domaines d'application dans l'informatique en général, cette catégorisation est basée sur le modèle de Meisel [110] comme l'illustre la Figure 2.4, ces solutions sont devisées en catégories, telles que la biologie cellulaire et développement, l'écologie et la biologie évolutive, l'immunologie, l'épidémiologie et la physiologie. Les inspirations biologiques sont présentées en sous-figure 2.4a, alors que les domaines d'application des techniques biologiques sont illustrés en sous-figure 2.4b.

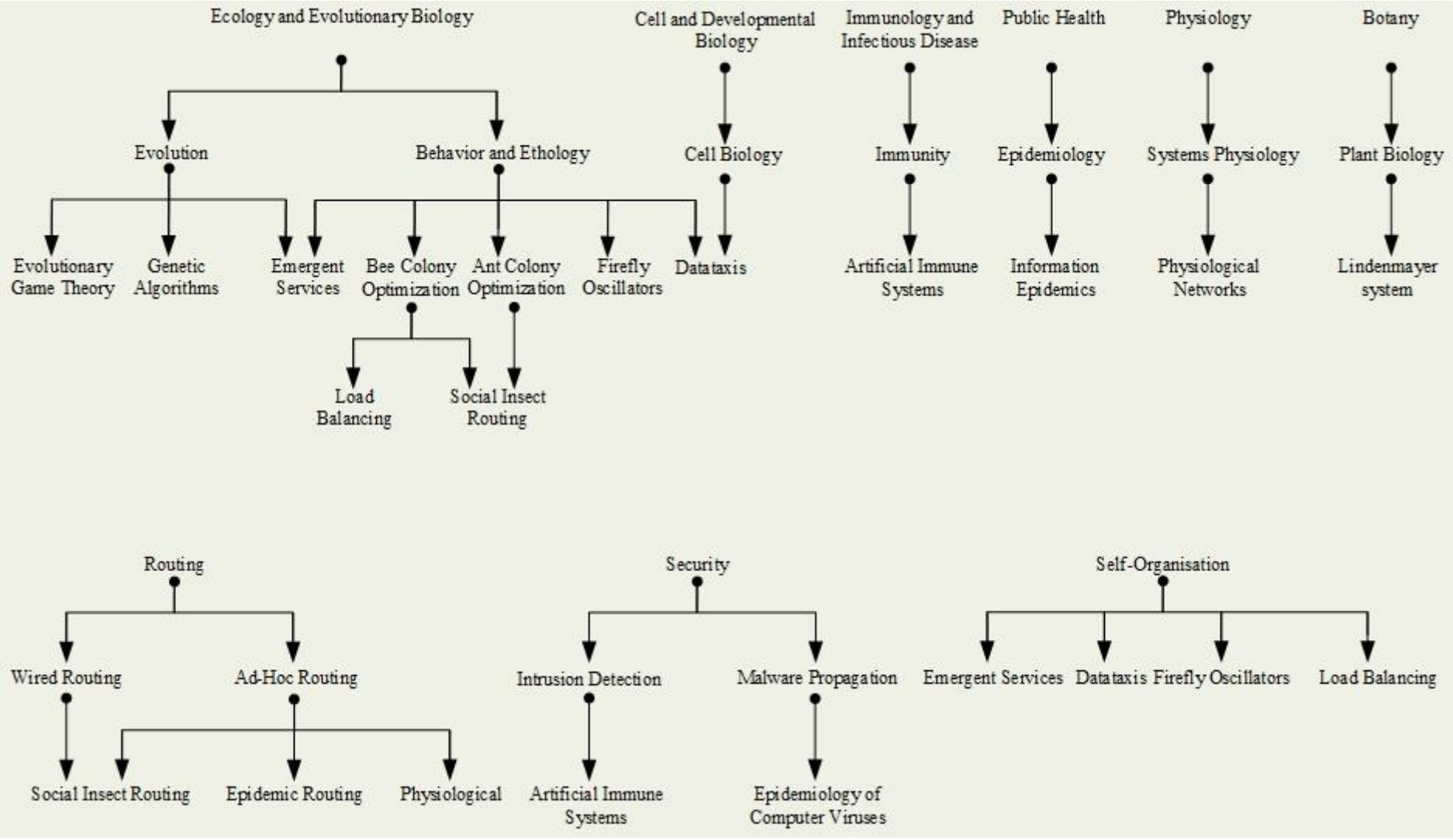

Figure 2.4 La classification de l'informatique biologique sur la base de, (a) l'inspiration, (b) le domaine d'application [110]

2.5.1 La théorie des jeux évolutionnaires

La théorie des jeux évolutionnaire (*EGT : Evolutionary Game Theory*) : est classée sous l'algorithme évolutionnaire, elle a été conçue par Borel en 1921 pour évaluer les problèmes multi-objectifs [111]. Cette dernière représente une optimisation multi-objective avec plusieurs décideurs, chacun contrôlant certaines variables de conception pour le problème donné [112]. La théorie des jeux est utilisée pour résoudre les problèmes de réseau RCSF dans les termes de l'efficacité énergétique [113,114], les problèmes de sécurité comme les attaques par déni de service (DoS) [115,116], la conservation d'énergie du routage [117-119], et quelques recherches se concentrent particulièrement sur l'équilibrage de charge [120,121].

2.5.2 Les algorithmes génétiques

Les algorithmes génétiques (*GA : Genetic Algorithm*) : ce sont les algorithmes évolutionnaires les plus connus et les plus répandus, ils sont des algorithmes d'optimisation inspirés de la théorie de l'évolution des espèces de Charles Darwin. En 1975, John Holland a introduit les algorithmes génétiques (GA), qui est basé sur une population d'individus dont chacun est une solution candidate du problème. Chaque solution doit être codée. Cette représentation codée est appelée chromosome, et est composée de gènes. Le degré d'adaptation d'un individu à l'environnement est exprimé par la valeur de la fonction objective correspondante. La taille de la population reste constante tout au long de l'algorithme génétique. La recherche de la solution est réglée par trois opérateurs qui sont appliqués successivement. La phase de coopération est gouvernée par un opérateur de sélection et un opérateur de croisement alors que la phase d'adaptation individuelle fait appel à un opérateur de mutation. La création d'une nouvelle génération est obtenue par itération de l'algorithme génétique qui va créer de nouveaux individus et en détruire d'autres (mécanisme de sélection naturelle) ce qui permet le renouvellement de la population (l'ensemble des solutions courantes). L'exploration de l'espace de recherche est alors réalisé par les opérateurs de mutation et assure la diversification des individus de la population (et donc des solutions). L'exploitation, quant à elle, est assurée par les opérateurs de croisement, qui recombinent les solutions, afin de les améliorer en conservant leurs meilleures caractéristiques [90].

Les algorithmes génétiques (GA) ont été étudiés sous divers aspects dans les RCSF : Zeng et al. [122] proposent une approche basée sur l'algorithme génétique, qui vise à améliorer le temps de réponse et à limiter la consommation d'énergie dans RCSF. Jin et al. [123] utilisent un algorithme génétique avec une fonction de fitness qui prend en compte la durée de vie du réseau ainsi que le temps nécessaire pour exécuter les ensembles de tâches. Cette approche dynamique équilibre la consommation d'énergie tout en prolongeant la durée de vie du réseau. Miorandi et al., [124] présentent également une approche génétique impliquant la mutation du génome et le croisement pour augmenter les performances des RCSF à grande échelle. Rossi et al. [125] ont proposé un algorithme efficace de génération de colonne basé sur les algorithmes génétiques afin de résoudre le problème de couverture de cibles et celui de la maximisation de la durée de vie dans les RCSF. Ting et al. [126] ont proposé une approche basée sur un algorithme mémétique combinant un algorithme génétique et le schéma

local de *Lamark* afin d'étendre la durée de vie dans les RCSF. Un algorithme génétique (GA) a été utilisé dans [127] pour assurer une couverture cible dans le déploiement de capteurs sans fil. Yildirim et al. [128] ont proposé un algorithme génétique qui permet de générer un déploiement optimal qui maximise la couverture de la zone, assure la connectivité entre les nœuds et les régions cibles sont couvertes pas au moins n nœuds. M.H. Alaiwy et al. [129] ont développé un algorithme génétique pour optimiser les délais de communication entre capteurs et actuateurs dans un schéma multi-acteurs (lorsque plusieurs actuateurs reçoivent la même information des capteurs) et optimiser aussi le coût de déploiement. Au-delà de ces algorithmes génétiques classiques, l'algorithmes génétiques multi-objectifs (MOGA) a également été utilisé dans les RCSF; par exemple, pour le routage [130-133], le placement des nœuds [134-138] et la gestion du cycle d'activation [139].

2.5.3 L'optimisation par colonies de fourmis

L'optimisation par colonies de fourmis (*ACO : Ant Colony Optimization*) est peut-être la meilleure branche analysée des algorithmes basés sur l'intelligence en essaim. L'ACO est basée sur l'observation du comportement collectif de la nourriture des fourmis. Marco Dorigo est à l'origine de l'ACO. Il estime que rechercher une source de nourriture est analogue à rechercher une solution dans un espace de recherche commun. Donc il appartient à la classe des métaheuristiques qui sont considérées comme des algorithmes d'approximation et à poser les bases d'un problème afin d'obtenir des solutions très efficaces de manière opportune. Nous citions dans ce qui suit quelques exemples qui sont résolus par l'utilisation d'ACO dans RCSF.

ARA (Ant colony Routing Algorithm) [140] c'est un L'algorithme de routage basé sur ACO, il vise à réduire la surcharge de routage pour les réseaux ad hoc mobiles (MANETs) [141]. L'algorithme permet la découverte de la route, la maintenance et le traitement des échecs pour augmenter le taux de transmission réussi tout en réduisant le trafic réseau, mais ne considère pas la consommation d'énergie. De même, AntHocNet [142] ne considère pas le problème de l'énergie mais propose un routage de réseau efficace avec un taux de transmission élevé et un temps de retard court en fournissant des chemins de routage alternatifs en cas d'échec de route.

Le problème de la consommation d'énergie est teint en compte dans LTAWSN [143], où il utilise des paramètres spéciaux dans sa fonction de compétence pour réduire la consommation d'énergie des nœuds du réseau, ce qui permet de prolonger la durée de vue du RCSF. Aussi EEABR (*Energy Efficient Ant Colony Based Routing*) [144] vise à prolonger la durée de vie du réseau en augmentant le taux de communication réussi avec une approche centralisée. Où il considère seulement la longueur du chemin et son niveau d'énergie, car il est toujours préférable de sélectionner le chemin le plus court avec un niveau d'énergie élevé ; mais l'utilisation un mécanisme de contrôle centralisé limite l'utilisation de cette technique proposée. Une extension de EEABR est nommée AMQRA (*Ant Colony Multiple QoS Constrained Routing*) [145] qui est proposée afin d d'améliorer le taux de livraison de paquets et de réduire le délai de bout en bout. AMQRA détermine le chemin en se basant sur une fonction, appelée fonction de coût, ce qui permet de réduire le délai et le taux de perte de communication tout en augmentant la QoS et l'utilisation optimale de la bande passante.

Enfin, il existe des autres protocoles basés sur l'ACO, qui rendent le routage plus efficace, à savoir, ARO [146], ANT-E [147], EARA [148], PERA [149] et FACO [150], et la plupart de ces protocoles concernent de la réduction des couts et d'augmenter le taux de livraison, mais ne pas tenir en compte la consommation d'énergie sauf ARO et FACO.

2.5.4 L'algorithme des colonies d'abeilles artificielles

L'algorithme des colonies d'abeilles artificielles (*ABC : The Artificial Bee Colony algorithm, BCO : Bee Colony Optimization*) est un algorithme métaheuristique basé sur une intelligence en essaim développé pour l'optimisation numérique. L'ABC a été introduit par Karaboga et al. [151,152], et qui est basé sur une modélisation minimale des comportements intelligentes de la recherche de nourriture par les abeilles. Dans ce modèle, chaque solution représente une source de nourriture potentielle dans l'espace de recherche (i.e. l'environnement) et la qualité de la solution correspond à la qualité de la position alimentaire. Les abeilles artificielles (i.e. agents) cherchent à exploiter les sources de nourriture dans l'espace de recherche. Dans leur comportement intelligent, les abeilles utilisent deux concepts fondamentaux : *l'auto-organisation* et *la division de travail*.

- **Auto-organisation** : un ensemble des mécanismes dynamiques qui établissent les règles de bases concernant l'interaction entre les abeilles. Les quatre propriétés fondamentales sur lesquels repose l'auto-organisation des abeilles dans la ruche sont :
  1. *Rétroaction positive* (*Positive feedback*) : Comme la quantité de nectar des sources de nourritures augmente, le nombre de leurs visites par les abeilles augmente aussi.
  2. *Rétroaction négative* (*Negative feedback*) : Le processus d'exploitation des sources de nourriture pauvres est arrêté par les abeilles.
  3. *Fluctuations* (*Fluctuations*) : Les scouts mènent un processus de recherche aléatoire pour découvrir des nouvelles sources de nourriture.
  4. *Interactions multiples* : Les abeilles partagent leurs informations sur les sources de nourriture avec les abeilles réceptrices (*Onlookers*) sur la piste de danse.
- **Division de travail** : Différentes tâches sont exécutées simultanément par des agents spécialisés

Le modèle proposé qui décrit le comportement intelligent des abeilles se compose de trois éléments essentiels : les sources de nourriture, les abeilles employées et les abeilles non employées et il définit deux modes principaux du comportement : le recrutement d'une source de nectar et l'abandon d'une source [153].

- *Sources de nourriture* : La valeur d'une source de nourriture dépend de nombreux facteurs tels que sa proximité de la ruche, sa richesse, le gout de son nectar ou la concentration de son énergie et la facilité d'extraction de cette énergie.
- *Abeilles employées* : Elles sont associées à des sources de nourriture particulière dont elles en sont actuellement exploitantes, elles transportent avec elles et partagent avec une certaine probabilité des informations à propos de cette source, sa direction, sa distance de la ruche et la rentabilité.

- *Abeilles non employées* : Elles cherchent sans cesse des sources de nourriture à exploiter. Il existe deux types des abeilles non employées : les scouts qui recherchent dans l'environnement entourant la ruche pour des nouvelles sources de nourriture et les *onlookers* qui attendant dans la ruche et choisissent une source de nourriture par le bais des informations partagées par les abeilles employées

Nombreux sont les domaines d'application des algorithmes d'abeilles dans RCSF, citons quelques-unes :

Karaboga et al., ont inspiré d'une colonie d'abeilles des mécanismes pour développer des protocoles dans lesquels les paquets sont traités comme des agents biologiques [154,155]. *Bee-Sensor-C* [156], un protocole de routage multi-chemin basé sur le cluster dynamique et le comportement de recherche de nourriture d'abeilles. Il est une extension de *BeeSensor* pour équilibrer la consommation d'énergie et améliorer la scalabilité.

BiSNET [157] est un middleware inspiré de colonie d'abeilles, qui définit une architecture de réseau de capteurs qui permet aux nœuds de capteurs d'adapter de façon autonome leurs cycles d'activation pour l'efficacité énergétique et la réactivité de la transmission de données, pour autoréparer collectivement (c.-à-d. détecter et éliminer) les faux positifs dans leurs données captées.

MONSOON [158] est un Framework à consiste d'un ensemble d'agents biologiques (i.e., des abeilles artificielles ou agent) qui peut tenter de satisfaire simultanément plusieurs objectifs éventuellement contradictoires (comme la latence, l'efficacité énergétique et le succès de la livraison) dans un RCSF. La Nina [159], ELNino [160] sont des Framework basant sur MONSOON pour l'auto-gestion d'un RCSF.

2.5.5 L'optimisation par essaim de particules

L'optimisation par Essaim Particulaire (OEP, ou *PSO : Particle Swarm Optimization* en Anglais) s'inspire du comportement social des animaux évoluant en essaim tel que les bancs de poissons par exemple. Cette méthode a été proposée par Kennedy et Eberhart en 1995 [161]. Au départ, les deux chercheurs voulaient simuler la capacité des oiseaux à voler d'une façon synchrone et leur aptitude à changer brusquement de direction tout en gardant une formation optimale. Le modèle qu'ils ont proposé a ensuite été étendu en un algorithme simple et efficace d'optimisation [162]. On observe dans ce type d'organisme un déplacement relativement complexe du groupe, alors que, chaque individu dispose d'une connaissance strictement locale de sa situation dans l'essaim, et a une intelligence très limitée [163]. Grâce à une connaissance de sa propre position dans l'essaim et des informations sur la vitesse et la position de ses voisins, un individu peut décider de son propre déplacement. Ces déplacements s'effectuent en employant des règles simples telles que : "aller à la même vitesse que les autres", "se déplacer dans la même direction" ou encore "rester proche de ses voisins", qui maintiennent la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. Ce comportement collectif propre à ce type d'animaux qui vivent en essaims, et qui se base sur l'analyse de l'environnement et du voisinage pour effectuer des choix optimaux, constitue alors, une méthode d'optimisation par l'observation des tendances des individus voisins.

Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modère par ses propres vécus [164]. Cette méthode d'optimisation a prouvé son efficacité pour résoudre plusieurs problèmes d'optimisation dans les RCSF. Cette section mentionne brièvement certaines de ces applications.

Tabu-PSO [165] est une méthode hybride qui est basée sur POS et l'algorithme de recherche Tabou, elle détermine le chemin optimal dans le routage pour améliorer la durée de vie du réseau et aussi l'efficacité énergétique, par conséquent, la fiabilité et la latence. MOPSO [166] est un algorithme d'optimisation multi-objectifs basée sur POS, qui utilise la dominance de Pareto pour obtenir les meilleures solutions locales et globales de chaque particule, afin de concevoir une trajectoire efficace de l'énergie pour les stations de bases (SB)mobiles. Pos-SinkPath [167] est un algorithme qui détermine le chemin optimal pour le mouvement d'une station de base dans la zone à surveiller pour maximise la couverture. Les auteurs de [168] ont proposé un algorithme de clustering basé sur PSO pour RCSF avec une base de station mobile (BS), il utilise PSO pour construire les claustre virtuelle pendant le processus de routage et aussi sélectionner le chef de cluster (CH) en fonction de son niveau énergétique et sa position. Cet algorithme réduit la consommation d'énergie et la durée de vie du réseau est prolongée. EBC-PSO [169] est un protocole de clustering utilisant PSO pour équilibrer la consommation énergétique dans un RCSF à grande échelle afin de maximiser la durée de vie du réseau. PSO-Traffic [170] est un algorithme de planification topologique basée sur POS pour déterminer la meilleure allocation de l'énergie aux nœuds, pour obtenir une couverture maximale avec un coût minimal. Les auteurs de [171] ont proposé un algorithme de déploiement de nœud basé sur des nœuds mobiles, pour couvrir totalement la zone à surveiller. Cet algorithme utilise POS pour calculer la nouvelle position de déplacement des nœuds capteurs mobiles, afin d'éviter le problème de trou de couverture. S-POS [172] est une méthode qui détermine l'emplacement d'un nœud selon les autres nœuds pour maximiser la couverture. PSO-Loc [173] et PSO-Iterative [174] sont des méthodes qui estiment les coordonnées de N nœuds dans un réseau contient M nœuds déployés dans une zone.

2.5.6 Les oscillations de luciole

Les oscillations de luciole visent généralement à développer des mécanismes d'auto-organisation pour permettre la robustesse sur un système complexe massivement distribué en particulier pour la synchronisation d'horloge. Cette méthode s'inspire de la synchronisation des oscillations d'essaim de luciole, elle est basée sur PCO (pulse-coupled oscillations) des oscillations couplées par impulsions, dont le modèle mathématique de base a été proposé par Richmond [175]. Bien que beaucoup d'autres modèles mathématiques ont été récemment développés sur des osculations de lucioles, mais les chercheurs ont commencé à utiliser des osculations de luciole pour améliorer l'auto-organisation basée sur le modèle M&S de Strogatz et Mirollo [176]. En effet, cette méthode utilisant pour la synchronisation d'horloge dans RCSF comme dans ces exemples.

Allen et al., ont proposé un algorithme appelé RFA (*Reachback Firefly Algorithm*) [177], qui améliore l'horodatage de la couche MAC et le mécanisme de réponse. Cet algorithme est amélioré par Cui et Wang dans [178], en considérant les retards de messages (*message delays*) avec une fenêtre de sensibilité tardive au schéma de *reachback firing* (RFA avec LSW). En négligeant les messages de

retard, la charge du réseau et le taux de couverture ont été optimisés. De même, en 2012, Sun et al. [179] ont développé leur synchronisation de luciole pour le clustering dans RCSF basé sur RFA. L'activation des clusters virtuels et la synchronisation des clusters entraînent individuellement une atténuation dans la charge du réseau. Les auteurs de [180] ont proposé un algorithme de synchronisation inspiré des lucioles basé sur un modèle de phase discrète multi-échelle qui peut optimiser le compromis de performance entre la scalabilité du réseau et la capacité de synchronisation dans un RCSF complexe. Le processus de synchronisation peut être considéré comme une transition d'état de Markov, ce qui assure la stabilité de cet algorithme par rapport au modèle M&S et à RFA.

2.5.7 Le système immunitaire artificiel

Vu que la définition de l'AIS dans la section 2.2.4 et selon le modèle de Meisel (cf., la figure 2.3), l'AIS est répertorié sous l'immunologie et les maladies infectieuses. Il est inspiré du système immunitaire humain ou mammifère. En effet, cette méthode est utilisée dans RCSF pour la sécurité et la détection d'anomalies, à cause, sa sensibilité à la détection des changements environnementaux et à l'identification des agents étrangers ou infectieux. Parmi ces travaux, on peut citer :

SASHA [181] implémente un mécanisme de détection de faute autoréparer (self-Healing) pour les données capturées défectueuses. Dans [182,183] les auteurs ont présenté leurs travaux sur la détection des anomalies pour les réseaux mobiles basés sur l'AIS, où ils ont utilisé un mécanisme basé sur l'AIS pour améliorer l'auto-apprentissage et l'adaptation sur un algorithme de routage appelé DSR, cette amélioration augmente le taux de transmission réussie et réduit les paquets perdus.

DNRS [184] est une structure de système immunitaire artificiel qui vise à limiter la consommation d'énergie tout en conservant la fiabilité de la détection des événements en changeant la fréquence du signal des nœuds. En permettant un changement de tension dynamique, DNRS réduit la consommation d'énergie de manière autonome.

WAIS [185] est un algorithme de diagnostic de fautes basé sur l'AIS pour les réseaux RCSF, qui permet à chaque nœud de réaliser le diagnostic. Cet algorithme améliore les mesures de performance, comme la latence moyenne de diagnostic, la précision de détection, le taux de fausse alarme par rapport aux algorithmes classiques.

DAWA [186] est un protocole de défense contre l'attaque de trou de ver dans le MANET, qui utilise la logique de floue et l'AIS. Et aussi est une optimisation au protocole AODV. DAWA surpasse les autres solutions existantes en termes, de taux de faux négatifs, de taux de faux positifs, de taux de détection, de taux de livraison de paquets, de taux de perte de paquets.

2.5.8 Le système de Lindenmayer

Le système de Lindenmayer ou L-système est inspiré de la plante (botanique) et initialement modélisé par A. Lindenmayer en 1968 [187], où il modélise le processus de développement et de prolifération de plantes ou de bactéries. Plus tard, les modèles mathématiques sont combinés par des algorithmes génétiques, ont commencé à être utilisés pour le mécanisme d'auto-organisation, d'auto-réparation et d'auto-adaptation. Ponnusamy et al. [188] ont utilisé le mécanisme d'auto-réparation inspiré de la

botanique. Ils ont proposé un algorithme de routage économe en énergie, la nature dynamique de cet algorithme permet d'éviter les trous sur les chemins de routage.

Nous avons présenté différentes catégories des solutions bio-inspirées et donné quelques exemples de chaque catégorie ainsi que leurs domaines d'application. Cependant, il existe d'autres catégories et de nombreux travaux importants que nous n'avons pas pu présenter dans cette section en détail. Le tableau 2.2 résume les domaines d'application de chaque catégorie biologique et fournit quelques autres travaux intéressants dans la littérature.

La solution bio-inspirée	**Domaine d'application dans RCSF**	**Référence**
La théorie des jeux évolutionnaire (EGT)	Utilisé pour améliorer la sécurité et réduire la consommation d'énergie	[115-121]
Les algorithmes génétiques (GA)	Utilisé pour optimiser les paramètres, le routage dynamique et maximiser la durée de vie.	[189-195]
ACO	Utilisé pour améliorer l'efficacité énergétique et la QoS dans le routage	[140,143-145,147-150]
ABC et BCO	Utilisé pour améliorer l'efficacité énergétique, la QoS dans le routage et l'équilibrage de charge	[196-200]
POS	Utilisé pour améliorer l'efficacité énergétique et la localisation, maximiser la couverture, minimiser le temps de transaction	[201-207]
Les oscillations de luciole	Utilisé pour améliorer l'auto-organisation et la synchronisation d'horloge	[177-179, 208-216]
AIS	Utilisé pour identifier les molécules étrangères et produire des cellules bénéfiques pour éliminer les molécules étrangères	[101,181,184,217-222]
Datataxis	Utilisé pour permettre de la couverture d'une grande zone et maximiser la collecte de données	[223-225]
Info. Epidemics	Utilisé pour maximiser le taux de transmission de message réussi et réduire la latence	[226-231]
ANN	Utilisé pour améliorer la sécurité, le routage, efficacité énergétique, détection de défaut et auto-configuration	[232-237]

Tableau 2.2 Les solutions bio-inspirées et leurs domaines d'application dans le RCSF

2.6 Conclusion

Dans ce chapitre nous avons introduit les défis du RCSF, que nous traiterons dans ce travail, tels que l'autonomie, l'auto-organisation, l'auto-configuration, l'auto-optimisassions et l'auto-réparation. Puis, nous avons présenté les approches bio-inspirées existantes dans la littérature, à savoir, le réseau de neurone artificiel, les algorithmes évolutionnaires, l'intelligence en essaim et le système immunitaire artificiel. Par la suite, nous avons présenté l'état de l'art des solutions bio-inspirées qui s'inspirent de la nature pour résoudre les défis des RCSF, où nous avons présenté pour chaque solution bio-inspirée, son approche, ses mécanismes et donner quelques exemples de ses domaines d'applications dans RCSF. Enfin nous conclurons que les solutions bio-inspirées ont commencé à être utilisées couramment car elles imitent la robustesse de la nature dans une variété de différents aspects des RCSF.

Chapitre 3

MONet : Développement et évaluation

3.1 Introduction

Précédemment, nous avons présenté un état de l'art sur les solutions bio-inspirées qui s'inspirent de la nature pour résoudre les défis des réseaux de capteurs sans fil, nous nous focalisons sur les défis concernant l'autonomie du RCSF. Récemment, la recherche dans ce domaine a été très fructueuse avec de nombreuses propositions et améliorations de ces solutions bio-inspirées. Cependant, elles ne peuvent pas être applicables dans tous les cas. Par ailleurs, la recherche dans les réseaux de capteurs est ouverte pour de nouvelles idées afin d'optimiser encore les solutions existantes pour obtenir des meilleures performances.

Dans ce chapitre nous présenterons notre première contribution de recherche dans le domaine d'autonomie d'un RCSF. Il s'agit d'un nouveau Framework middleware nommé MONet, inspiré d'un système biologique (i.e., la colonie d'abeilles). En effet, notre Framework utilise des mécanismes similaires à qui se trouvent dans la nature pour auto-gérer un RSCF, dans l'objectif de réduire au maximum les interventions externes, soit de l'utilisateur, soit des autres logiciels. A la fin, nous allons évaluer les performances de notre contribution en termes d'auto-organisation, d'auto-configuration, d'auto-optimisassion et d'auto-réparation du RCSF. L'évaluation des performances sera conduite à travers le simulateur PowerTOSSIM [238].

3.2 MONet: Une optimisation multi-objectifs pour les réseaux de capteurs sans fil utilisant un mécanisme biologique.

3.2.1 Définition

MONet (***M**ulti-objective **O**ptimization for wireless sensor **NET**works using a biological mechanism*) [239] est un système multi-agent bio-inspiré (SMA) dans lequel chaque agent mobile collecte les données sur le nœud capteur où il se trouve, et les transporte à une SB. Ces agents interagissent entre eux par un mécanisme nommé Stigmergie qui permet à un agent de communiquer indirectement, diverses informations, avec d'autres agents, ils vont y marquer leurs déplacements (réussite et échec) d'un nœud à un autre en mettant des phéromones sur les voisins de leurs nœuds de départ. De façon autonome l'agent utilise l'algorithme d'optimisation par colonie d'abeilles artificielle (ABC : *Artificial Bee Colony*) pour déterminer et optimiser le chemin qu'il doit parcourir pendant sa migration vers la SB, en appliquant l'algorithme génétique (GA) sur les agents qui sont arrivés à la SB pour améliorer les nouveaux agents qui construiront ce système, ce qui leur permettra de s'adapter aux conditions de leur environnement.

3.2.2 Le système multi-agent utilisé

3.2.2.1 Les agents

La littérature propose beaucoup de définitions différentes pour la même notion d'agent comme par exemple dans, [240-242]. Cependant, les auteurs de ces articles ont vu qu'un agent est une entité autonome évoluant dans un environnement, et agissant dans l'objectif d'atteindre des buts. Cette définition s'applique particulièrement bien aux agents physiques tels que les robots qui peuvent au

travers de capteurs et d'effecteurs interagir avec leur environnement. Il existe une seconde catégorie d'agents pour lesquels les notions d'environnement, de ressources, et d'interactions sont plus floues. Ce sont les agents logiciels, et nous allons maintenant présenter le concept d'agent logiciel, et nous intéresser plus particulièrement à la notion de mobilité pour un agent logiciel.

3.2.2.2 Les agents logiciels

Par définition, un agent mobile est un agent logiciel ou virtuel, n'ayant par conséquent pas de représentation physique. Un agent logiciel est constitué de code s'exécutant dans un environnement informatique. La différence entre un agent logiciel et un simple programme informatique est mince. Toutefois, il est assez simple de voir qu'un agent informatique est plus évolué qu'un programme informatique. Là où un programme s'exécute sur une machine et produit un résultat, un agent logiciel est capable de s'adapter aux conditions changeantes de l'environnement (ressources CPU, mémoire disponible, bande passante, etc. …) pour réaliser le but qu'il s'est ou qui lui a été fixé.

L'adaptation d'un agent logiciel passe par des capacités de *réplication*, de *terminaison*, de *communication* et de *déplacement*. Derrière ces quatre termes se cachent les propriétés suivantes :

- **Réplication** : un agent logiciel peut décider de se dupliquer sur un nœud pour, par exemple, assurer au moins l'existence d'un agent dans le nœud.
- **Terminaison** : un agent logiciel peut décider d'arrêter son exécution sans une intervention extérieure.
- **Communication** : un agent logiciel est capable de communiquer par l'intermédiaire de messages avec d'autres agents logiciels.
- **Déplacement** : un agent logiciel possède la capacité de se déplacer dans l'environnement dans lequel il est situé (cf., la section 3.3.2).

3.2.2.3 Systèmes multi-agents

Au travers des différentes définitions apportées dans la littérature comme dans [240,242,243], nous pouvons clairement considérer qu'un Système Multi-Agents (SMA) comme un ensemble d'agents autonomes qui interagissent entre eux, selon certaines relations, et se regroupes à l'aide de relations organisationnelles dans un environnement partagé. Certes, la notion d'environnement est primordiale dans un système multi-agents mais cet environnement est décomposé en deux parties. L'environnement social qui est composé des autres agents du système et l'environnement physique au sein duquel les agents évoluent.

3.2.2.4 Systèmes Multi-Agents bio-inspirés

Les concepteurs de SMA ont souvent puisé leur inspiration dans le domaine biologique. Beaucoup de travaux ont été inspires par les animaux et plus particulièrement par leur comportement et les organisations régissant différentes espèces. Les types d'animaux les plus étudiés par la communauté sont les abeilles, les fourmis et les oiseaux. Comme nous venons de voir quelques travaux portant sur ces thématiques dans le chapitre 2, les abeilles et les fourmis ont notamment été étudiés pour leur comportement et leur organisation sous la forme de société animale. Les oiseaux ont été étudiés pour

leur capacité à se déplacer en groupe durant leur vol. Dans notre travail, nous avons utilisé un SMA s'inspirant de la colonie des abeillés.

3.2.3 Modélisation du RCSF

La conception du RCSF est similaire au certains principes et mécanismes des systèmes biologiques, comme autonomiste, adaptabilité, optimalité, *self-healing*. Les abeilles sont l'un des animaux sociaux les plus étudiés et beaucoup de leurs comportements ont déjà été appliqués aux RCSF. Vu que les abeilles agissent de manière autonome, influencées par les conditions locales et les interactions locales avec les autres abeilles. Une colonie d'abeilles peut être composée d'un nombre considérable d'abeilles car toutes leurs activités sont effectuées sans aucun contrôle centralisé, et s'adapte aux conditions environnementales dynamiques. Par exemple, lorsque la quantité de miel dans une ruche est insuffisante, donc nombreuses abeilles quittent la ruche pour récolter du nectar ou du pollen des fleurs, dès que la ruche est pleine de miel, les abeilles se reposent dans la ruche. Les abeilles se communiquent entre elles par un système de communication phéromonal, où la reine diffuse une information centrale, appelée la phéromone royale (*QMP : Queen Mandibular Pheromone*) pour réguler l'homéostasie et le développement de la colonie, la phéromone de *Nasonov* : cette phéromone volatile est produite lors d'une position particulière de l'abeille, qui ventile la phéromone avec ses ailes pour orienter les autres abeilles de la colonie comme par exemple à l'entrée de la ruche [244] ou pendant l'essaimage [245] et en cas de risques et de menaces, les abeilles gardiennes émettent une phéromone d'alarme à l'entrée de la ruche, ce qui incite alors d'autres abeilles à se déplacer à l'entrée pour protéger la ruche, IPA (l'IsoPentyl 5 Acetate) [246] est une phéromone d'alarme qui provoque rapidement un comportement défensif (alerte, recrutement). Ces mécanismes phénoménaux ont un rôle majeur dans la régulation des tâches dans la colonie d'abeilles.

La colonie d'abeilles	**RCSF**
Les composants	
La reine	La station de base
Les fleurs	Les nœuds de capteurs
Les abeilles ouvrières	Les Agents (abeille artificiel)
La communication	
La phéromone *QMP*	La phéromone de station de base (SB)
Les phéromones *Nasonov* et *BEP*	La phéromone de migration
Les phéromones *IPA* et *EO*	La phéromone d'alerte

Tableau 3.1 Corrélation entre la colonie d'abeilles et les RCSF.

La structure et le comportement de chaque abeille sont très simples. Cependant, un groupe d'abeilles présente de manière autonome des caractéristiques de système souhaitables telles que l'autonomiste, reconfiguration, la scalabilité, l'adaptabilité et self-Healing, à travers des comportements collectifs et des interactions entre les abeilles. Sur la base de cette observation, on peut estimer que si les applications des RCSF sont conçues selon certains principes et mécanismes biologiques, elles peuvent être en mesure d'atteindre les exigences des RCSF citées auparavant (e.g., auto-organisation, auto-

configuration, auto-optimisation et etc…). En effet, nous allons modéliser un RCSF comme une colonie d'abeilles pour simplifier l'utilisation des mécanismes d'abeilles sur les RCSF. L'idée est illustrée dans le tableau 3.1.

3.2.4 L'architecture de MONet

Comme nous avons vu dans la section précédente, MONet est un Framework qui se compose de deux parties (cf., la figure 3.1), un système multi-agent (SMA) et un outil d'optimisation. Le système est composé des éléments suivants :

- Un environnement local, appelé *MONet-runtime* (MONet-R), est le lieu où un agent évolue, observe, interagit. Il est de type totalement observable, qui permet à un agent de posséder l'ensemble des connaissances du monde dans lequel il est.
- Un ensemble d'objets. Ces objets sont situés, c'est-`a-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans MONet-R. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
- Un ensemble d'*agents logiciel*, qui sont des objets particuliers, lesquels représentent les entités actives du système.
- Un ensemble de relations, appelé *les phéromones* qui unissent des agents entre eux.
- Un ensemble d'opérations, appelé *les comportements* permettant aux agents de percevoir, produire, consommer, transformer et manipuler des objets de système.

Ici nous proposons de situer dans un environnement global (MONet-R) composé d'objets modifiables, des agents logiciel unis entre eux par des relations (phéromones). Dans cette architecture c'est clairement le couple agent/environnement qui est au cœur d'un SMA. Tandis que la deuxième partie, c'est un outil d'optimisation au SMA, appelé MONet-Server (MONet-S) qui se base sur les principes du problème multi-optimisation (MOP) et utilise un algorithme génétique multi-objectifs (e.g., MOGA) afin d'améliorer les agents du SMA.

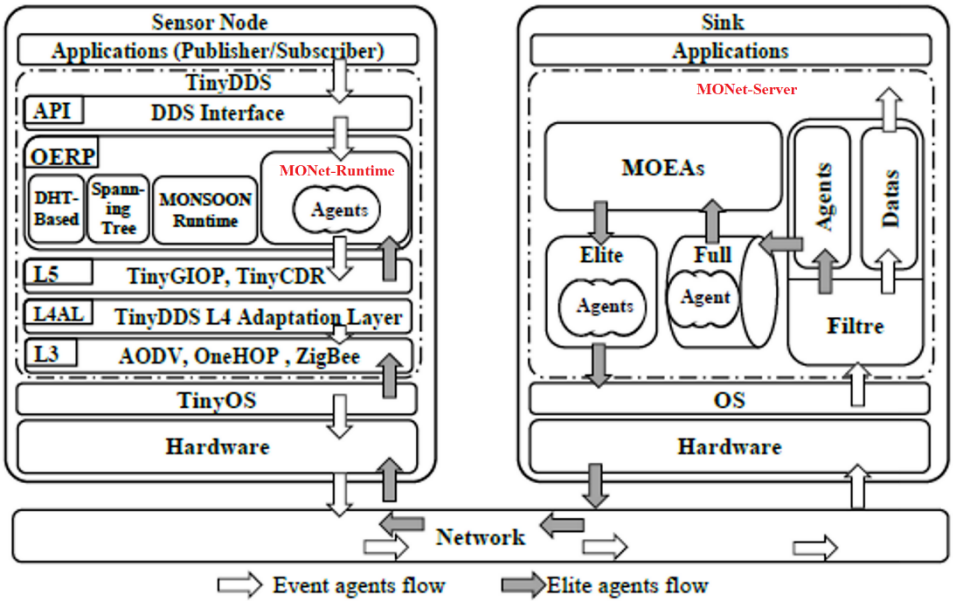

Figure 3.1 l'architecture de MONet.

3.2.5 Le fonctionnement de MONet

Généralement, le MONet-runtime fonctionne au-dessus du système d'exploitation (e.g., TinyOS [247]) dans chaque nœud capteur. Alors que le MONet-server fonctionne comme un serveur central, qui se situe dans la station de base si elle a des ressources qui satisfassent ses besoins, sinon le processus s'exécute dans une machine terminale (e.g., PC) qui est reliée avec la SB à travers une liaisons (e.g., internet). Quand un nœud capteur capte des données, le MONet-R encapsule ces données dans son agent local, puis lui permet de se déplacer vers la SB à travers le réseau RCSF, chaque agent a sa propre structure génétique (i.e., un ensemble des gènes) qui détermine le temps et la manière d'invoquer ses comportements, par exemple, comment faire pour se déplacer vers le prochain nœud. Ces gènes, peuvent être améliorés itérativement par le MONet-S. Notamment, dès qu'un agent arrive à une BS, celle-ci l'envoie au MONet-S, lequel évalue l'ensemble d'agents qui lui sont arrivés selon leurs performances qui ont été estimés par les agents pendent leurs migrations vers la SB (e.g., la latence), y compris les paramètres de QoS. Le résultat de cette évaluation est un ensemble d'agents élites ayant les meilleures performances. Le MONet-S injecte ces agent élites dans le RCSF afin d'améliorer les caractéristiques comportementales des prochaines générations des agents, de sorte que dans chaque nœud capteur, le MONet-R permet à son propre agent d'effectuer les opérations génétique (i.e. le croissement et la mutation) avec l'un des agents élites, et ainsi de suite, le nouvel agent (i.e., le fils) répète le même cycle de vie de ses parents, la figure 3.2 donne un aperçu de processus de MONet.

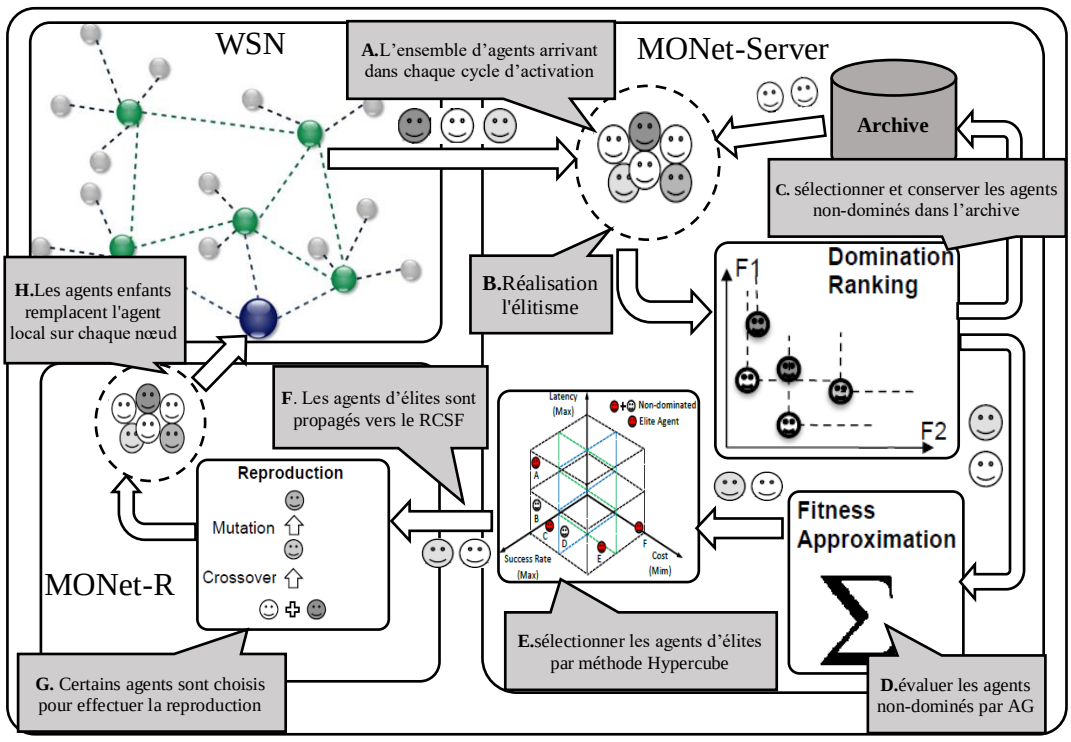

Figure 3.2 Un aperçu du processus de MONet

3.3 MONet-Runtime

Habituellement, le SMA a un seul propre environnement, dans notre cas, le SMA a un environnement global qui se compose d'un ensemble des environnements locaux. A savoir, le MONet-R dans chaque nœud capteur, représente un environnement local et tous ces environnements existant dans les nœuds qui construisent le RSCF, représentant ainsi l'environnement global du SMA. Il y a plusieurs services qui sont assurés par le MONet-R, comme le contrôle du cycle d'activation (*duty cycle*) du nœud, de plus, il assure les trois services suivants nécessaires aux agents :

- Les services des comportements d'agent : chaque agent exécute plusieurs comportements et chaque comportement est une implémentation de plusieurs services, quand il est exécuté, il invoque les services correspondances.
- Les services de maintenance des nœuds voisins : est un ensemble des services qui permet de créer, supprimer et mettre à jour la table de phéromone de nœuds voisins. Par exemple, quand il reçoit une phéromone de son nœud voisin déjà existant dans la table, donc la mise à jour de sa concentration, se fait automatiquement. Ces services permettent à l'agent d'émettre les phéromones aux nœuds voisins pendent sa migration à la SB.
- Le service de gestion les phéromones : est un ensemble des services qui gèrent les phéromones comme le contrôle d'évaporations des phéromones, etc.

Dans ce qui suit nous présentons les éléments qui constituent le MONet-R.

3.3.1 La Structure de l'Agent

Dans le MONet-R, l'agent c'est un agent logiciel qui est constitué des attributs, un corps et des comportements.

- **Les attributs** : est un ensemble d'informations qui décrivent l'agent. Il comprend, les gènes, le niveau d'énergie, les données captées, son temps de capture et l'identificateur du nœud sur lequel les données ont été captées.
- **Le Corps** : est un ensemble de fonctions pour la collection, le transport et le traitement les données captées (e.g., une fonction qui décide de rejeter les données captées ou elle les envoie à une SB) et aussi on peut implémenter des fonctions qui font des actions sur le nœud, donc ces fonctions déterminent les fonctionnalités de l'agent.
- **Les Comportements** : sont des fonctions particulières qui implémentent les actions inhérentes à tous les agents. Ces fonctions permettent à l'agent d'interagir avec son environnement selon les conditions détectées sans aucune intervention des autres agents, du système, des stations de base et/ou de l'utilisateur.

3.3.2 Les Comportements de l'Agent

Dans MONet-R, chaque agent implémente les comportements suivants :

- **Gain d'énergie** : est le premier comportement que l'agent effectue, où dès qu'il capture des données, le MONet-R lui alloue une valeur constante de l'énergie, cette énergie est une valeur

logique et n'est pas physique, qui est utilisée par l'agent pendent sa migration pour contrôler sa durée de vie dans le RCSF.

- **Dépenses d'énergie :** est le comportement qui contrôle la consommation énergétique de l'agent. Chaque agent consomme une quantité constante de l'énergie physique pour utiliser les ressources qui sont disponibles sur le nœud (e.g., CPU, radio), il dépense aussi l'énergie qu'il a gagné du premier comportement pour effectuer ses comportements, et la valeur de l'énergie consommée par chaque comportement est fixée pour tous les agents et appelée ECIB (*Energy Cost to Invokes Behavior*), et après chaque invocation d'un comportement, l'énergie de l'agent est décrémentée de ECIB.
- **La mort** : est le dernier comportement que l'agent effectue dans sa vie, lequel est invoqué, soit par l'agent lui-même ce cas est appelé « le suicide », soit par MONet-R ce cas est appelé « le meurtre ». Dans le premier cas, une fois que l'énergie de l'agent est épuisée, il se suicide à cause de la faim. Ceci évite qu'un agent reste se déplacer à l'infini s'il existe un problème dans RCSF. Dans le deuxième cas, après chaque invocation du comportement de reproduction (cf., le paragraphe ci-dessous), le MONet-R doit tuer les agents qui effectuent ce comportement. Dans les deux cas, lorsqu'un agent meurt, l'environnement local libère toutes les ressources qui lui sont attribuées, et dans en cas où, tous les agents sont morts au même temps, alors le MONet-R doit créer un agent au hasard afin d'assurer la contrainte de SMA (i.e., dans tout moment au moins un agent se trouve dans MONet-R) comme cela est illustré par le diagramme de la figure 3.3.

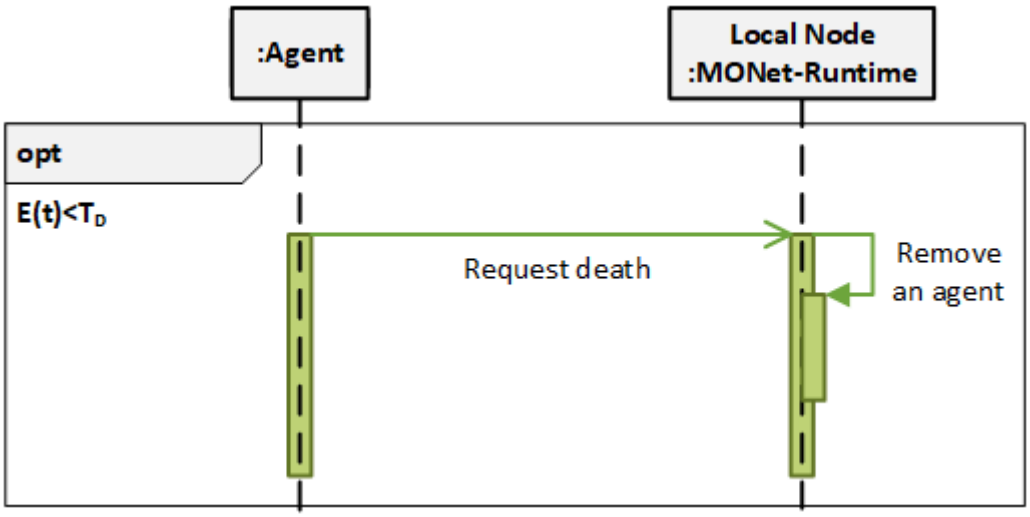

Figure 3.3 Le diagramme de séquence de comportement de mort

- **Réplication** : est le deuxième comportement que l'agent effectue dans sa vie, quand un agent veut transporter les données captées à la SB, il doit créer une copie de lui-même, appelée l'agent enfant. L'un d'eux reste dans le nœud et l'autre transporte les données à la SB. Cette opération coûte à l'agent parent la moitié de son énergie qu'il donne à l'agent enfant comme l'illustre la figure 3.4. L'objectif de ce comportement est d'assurer la contrainte de SMA (i.e., à tout moment au moins un agent se trouve dans MONet-R).

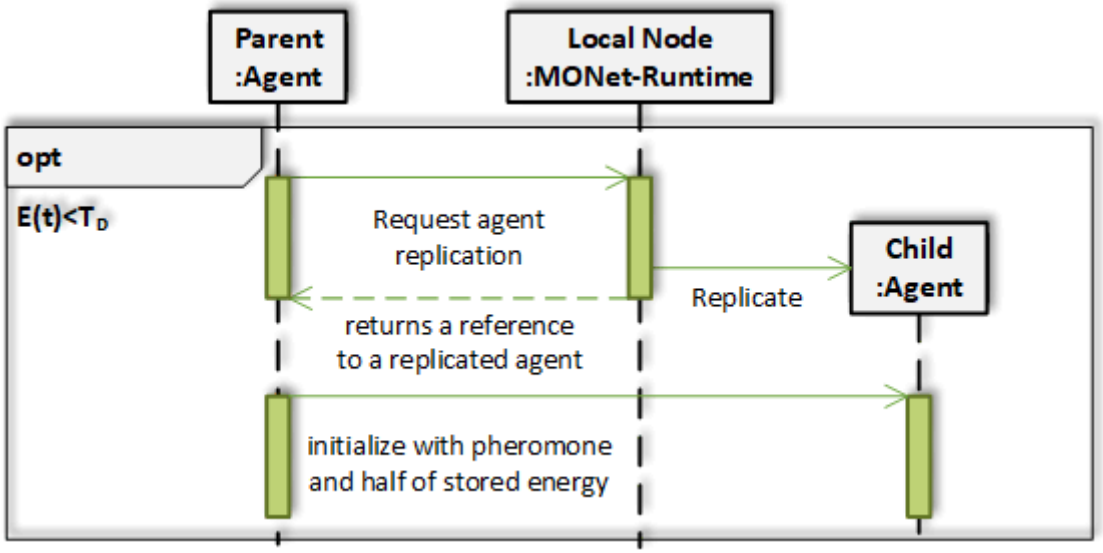

Figure 3.4 Le diagramme de séquence de comportement de réplication

– **Swarming** : Ce comportement s'effectue entre au moins deux agents migrants. Quand plusieurs agents se rencontrent dans un nœud et s'ils acheminent vers la même SB, alors ils se fusionnent dans un seul agent qui hérite la structure génétique de l'un d'eux et agrège leurs données captées, à condition que sa taille totale ne dépasse pas la taille maximale de paquet. Afin qu'il augmente le swarming entre les agents, chaque agent utilise un mécanisme qui se base sur la probabilité de swarming (P_S), celle-ci permet à un agent d'attendre une période de temps (T_w) dans le nœud ou de migrer immédiatement. Sachant que les nœuds situés à proximité de la SB reçoivent un grand nombre d'agents migrants vers la SB par rapport aux nœuds situés loin de cette dernière. Pour cette raison nous avons défini cette probabilité en fonction de la concentration de phéromone de la SB.

$$P_S = 1 - \frac{\rho_a}{\rho_n} \tag{3.1}$$

Avec, $\rho_a$est une valeur allouée à l'agent par MONet-R, elle est égale à la valeur de la concentration de la phéromone BSP de son nœud source. ρ_n représente la valeur de la concentration de la phéromone BSP du nœud actuel où il se situe. Chaque agent avant de se déplacer à sa prochaine destination, il doit calculer l'équation 3.1 et n'attends les autres agents que si la valeur de P_S est très élevée, comme l'illustre la figure 3.5. L'objectif de ce comportement est de minimiser le nombre d'agents migrants à la SB, par conséquence, la consommation d'énergie est aussi minimisée, néanmoins, l'utilisation aveugle de ce comportement peut augmenter la latence parce que l'agent s'arrête plusieurs fois dans les nœuds intermédiaires pendant sa migration.

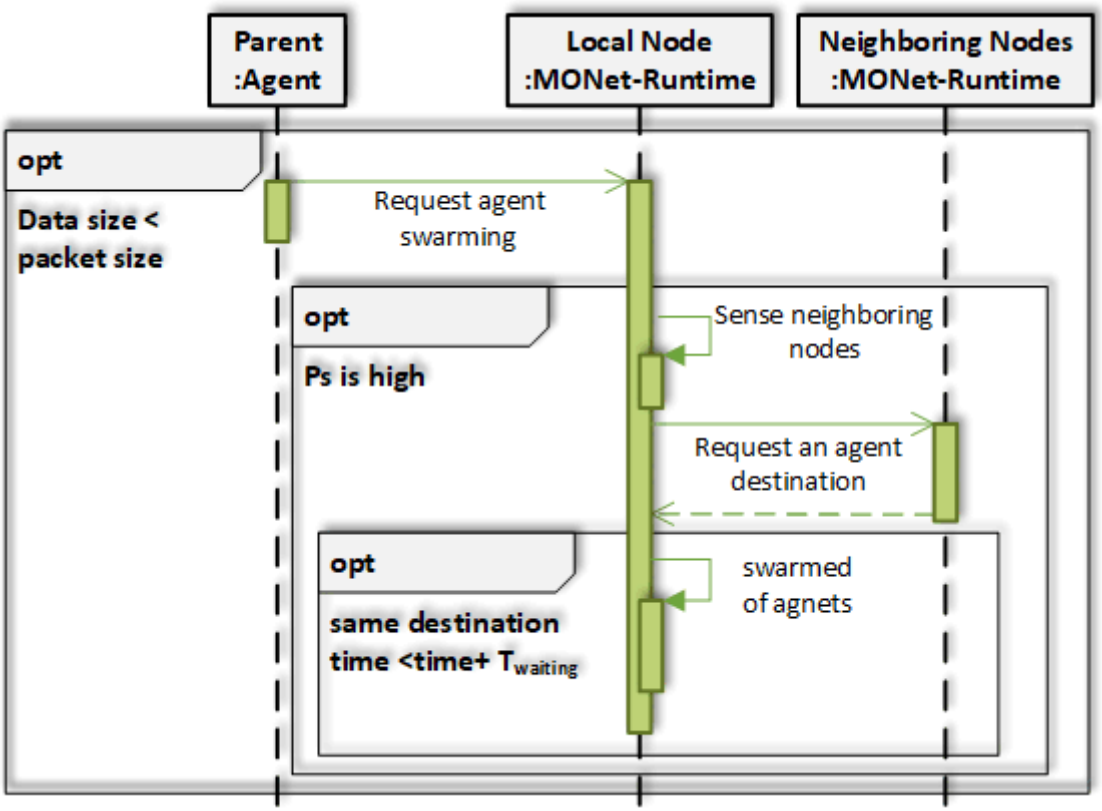

Figure 3.5 Le diagramme de séquence de comportement de Swarming

- **Le déplacement** : est le comportement qui permet à l'agent de se déplacer de nœud local à un nœud voisin. Afin qu'un agent transporte ses données captées à la SB, il effectue ce comportement qui lui détermine le nœud de destination suivant en se basant sur trois types de phéromones disponibles dans la table de phéromone du nœud actuel, à savoir, la phéromone de station de base, de migration et d'alerte.

    1. **Phéromone de station de base (BSP):** Cette phéromone similaire à la phéromone QMP de reine des abeilles. Chaque SB (reine) diffuse périodiquement une information centrale appelée la phéromone BSP vers les nœuds du réseau afin d'attirer fortement les agents (les abeilles). Ces nœuds interagissent avec les autres nœuds du RCSF et propagent la BSP dans tout le réseau (la colonie) indiquant la présence et l'influence de la SB (la reine), tel que le BSP se propage dans le RCSF de nœuds les plus proches aux nœuds plus lointains à la SB. Une fois qu'un nœud reçoive la BSP, il décrémente sa concentration puis l'envoie vers ses voisins et ainsi de suite, si un nœud a reçu la même phéromone avec plusieurs concentrations différentes, alors il garde la plus haute valeur. À l'aide de cette phéromone, les agents peuvent s'orienter vers les SBs qui existent aux environs et ils doivent toujours se déplacer du nœud à faible concentration vers le nœud à haute concentration de BSP (i.e., l'inverse du principe du gradient de concentration).

2. **Phéromone de migration (MP) :** Cette phéromone similaire à la phéromone de *Nasonov* des abeilles. Chaque agent (abeille) diffuse la phéromone MP pour attirer les autres agents de façon organisée pour suivre sa trace de migration. Cette phéromone est produite lorsqu'un agent s'est déplacé vers le nœud de destination avec succès. L'agent disperse la MP dans tous ses nœuds voisins seulement, chaque MP indiquant l'ID du nœud actuel et sa propre concentration ρ_{MP}.
3. **Phéromone d'alerte (AP) :** Cette phéromone similaire à la phéromone d'IPA des abeilles, dans la nature, IPA aide les abeilles dans le ciblage des piqûres, mais l'AP aide les agents pour l'évitement des nœuds temporairement faibles. Lorsqu'un agent ne peut pas se déplacer dans un délai vers le nœud de destination, il doit donc diffuser la phéromone AP pour avertir les autres agents. Les échecs de migration peuvent se produire en raison d'épuisement de la batterie des nœuds, les dommages physiques, ainsi que des pannes de liaison en raison d'interférences et la congestion. Comme dans MP, l'agent disperse l'AP dans toutes ses nœuds voisins seulement, chaque AP indiquant l'ID du nœud qu'un agent n'a pas pu se déplacer vers ainsi que sa propre concentration ρ_{AP}.

Une fois qu'un nœud reçoive une phéromone (SBP, MP ou AP), son environnement local (i.e., MONet-R) fait automatiquement la mise à jour de la table de phéromones (PT), où il met la nouvelle valeur de concentration de cette phéromone dans la case correspondante à l'ID de nœud indiqué par cette phéromone.

Du fait du dynamisme du RCSF (i.e. ajoute des nouveaux nœuds), et les agents ont besoin de la phéromone SBP afin de se déplacer vers la SB. Pour cela SBP est une phéromone permanente, et il doit donc se rediffuser dans chaque cycle d'activation, et aussi pour éviter le cas, où les agents passent (ou évitent) un nœud déjà référencé de MP (ou AP), les phéromones MP et AP doivent être temporaire. Pour cela nous fixons le taux d'évaporation de ces deux phéromones en fonction du nombre de cycle d'activation. Nous pouvons donc définir ce qui suit:

Définition 1: *Soient ρ la concentration d'une phéromone référençant le nœud x et η le nombre total de cycle d'activation nécessaire pour l'évaporation d'une phéromone. Nous avons le taux d'évaporation des phéromones* :

$$\varepsilon = \frac{\rho_x}{\eta} \tag{3.2}$$

Ainsi après chaque cycle d'activation d'un nœud les concentrations de phéromones MP et AP existants dans sa table PT sont décrémentes de $\boldsymbol{\varepsilon}$. De cette manière il ne reste aucune trace d'une phéromone au bout d'un temps équivalent à $\boldsymbol{\eta}$ cycle.

Pour effectuer son prochain déplacement, le comportement de migration d'un agent consiste alors à choisir un nœud possédant la valeur plus élevée de l'équation 3.3.

$$\mathrm{Max}\left[\left\{WS_j, \mathrm{j} \in \mathrm{V}_X\right\}\right] \tag{3.3}$$

Où la valeur WS est calculée pour chaque nœud voisin de *x* comme suite :

$$WS = w_1 \frac{\rho_{BSP} - \min(\rho_{BSP})}{E(\rho_{BSP})} + w_2 \frac{\rho_{MP} - \min(\rho_{MP})}{E(\rho_{MP})} + w_3 \frac{\rho_{AP} - \min(\rho_{AP})}{E(\rho_{AP})} \tag{3.4}$$

Avec : *x* : le nœud local de l'agent

V_x : l'ensemble du voisinage du nœud *x*

ρ_{BSP} (resp. ρ_{MP} *et* ρ_{AP}) : la valeur actuelle de la concentration de BSP (resp. MP et AP)

min(ρ_{BSP}) (resp. *min*(ρ_{MP}) *et min*(ρ_{AP})) : la valeur minimale de toutes les concentrations de BSP (resp. MP et AP) dans la table PT du nœud x.

E(ρ_{BSP}) (resp. *E*(ρ_{MP}) *et E*(ρ_{AP})) : L'étendue des concentrations de BSP (resp. MP et AP) est la différence entre sa valeur la plus élevée et sa valeur la moins élevée dans la table PT du nœud x.

w_1,w_2 *et* w_3 : sont les poids (coefficients) de l'équation 3.4 et aussi représentent la structure génétique de chaque agent. Chaque poids prend l'une de ces valeurs {-1,0,1}.

Enfin, ce comportement est considéré comment étant le comportement principal de l'agent. Nous pouvons donc proposer la définition suivante :

Définition 2 : *Soient x et y deux nœuds voisins, y possédant la valeur plus élevée de WS et* V_i *l'ensemble de voisinage du nœud i. Un agent marque son déplacement d'un nœud x vers un nœud y, si ce dernier est succès alors il dépose le phéromone MP sur* V_y *sinon il dépose le phéromone AP sur* V_x*. On note* ρ_{MP} *(resp.* ρ_{AM}*) la concentration de phéromone MP (resp. AM).*

Algorithme 1 : *le déplacement un agent sur un nœud x*
Entrées : *l'agent d'un nœud x, Vx le voisinage d'un nœud x*
Sorties : *Un nœud y où se déplacer*
PasVoisinage ← *faulse*
Tant que *PasVoisinage* == *faulse* **faire**
  pour chaque $y \in V_X$ **faire**
    Calculer la valeur de WS pour y
  Choisir $y \in V_X$ *avec sa valeur plus élevée de WS* ;
  Se déplacer sur le nœud y;
  Si *son déplacement est succès* **alors**
    Construction des voisinage y V_y ;
    Envoyer la phéromone MP référençant y vers Vy ;
    PasVoisinage← *true* ;
  Sinon
    Incrémenterr$_{AP}$ *pour y*
    Envoyer la phéromone AP référençant y vers Vx ;

Fin

Après la définition 2, nous pouvons donc proposer l'algorithme 1 qui permet à un agent de se déplacer dans l'ensemble du réseau RCSF et d'en atteindre n'importe quelle SB. Ici un agent choisit sa prochaine destination en fonction de l'équation 3.3 parmi le voisinage du nœud qu'il se situe. Dans le cas où le déplacement est réussi, l'agent envoie la phéromone MP vers son voisinage, sinon il envoie la phéromone AP. voir aussi la figure 3.6 qui illustre en détail ce comportement.

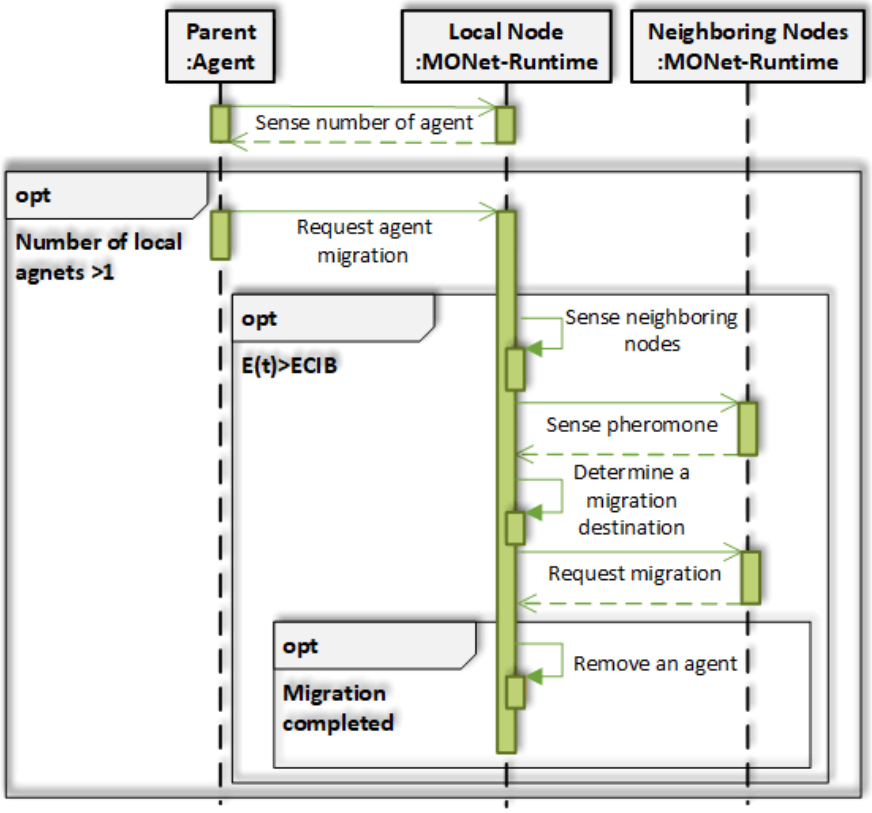

Figure 3.6 Le diagramme de séquence de comportement de déplacement

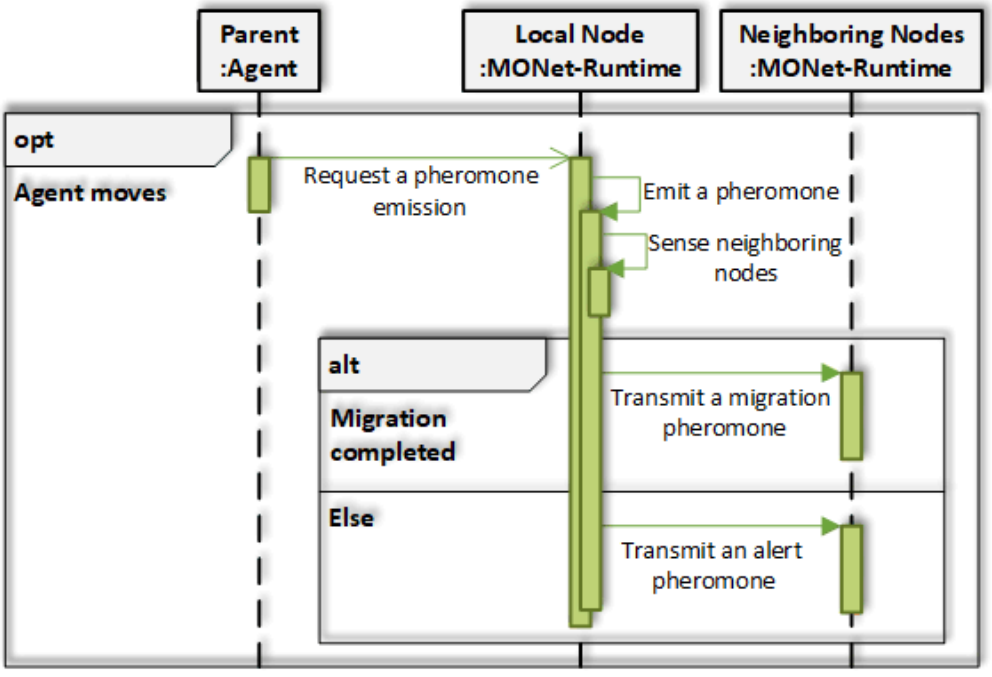

Figure 3.7 Le diagramme de séquence de comportement d'émission d'une phéromone

- **Stigmergie** : est un comportement imbriqué dans le comportement du déplacement, où une fois qu'un agent effectue le comportement du déplacement, il l'effectue alors immédiatement. Ce comportement est considéré aussi comme étant un système de communication entre les agents. Le but de ce comportement est qu'un agent informe les autres agents du statut de son déplacement (succès ou échec) par la diffusion de la phéromone appropriée vers le voisinage immédiat de nœud qu'il se situe, comme l'illustré par la figure 3.7. Chaque agent acquis ces informations, peut décider de suivre ou d'éviter la trace de cet agent. En conséquence, le taux de réussite et la latence peuvent être améliorées et ceci permet d'optimiser la consommation énergétique.
- **Reproduction** : est un comportement d'optimisation et est destiné à faire évoluer les agents qui se trouvent dans les nœuds du réseau. Celui-ci est effectué entre deux agents appelés *les parents*, pour produire un agent appelé *le fils* dont hérite la structure génétique de ses parents, une moitié de chaque père. Afin d'assurer l'optimisation de l'agent fils, le MONet-R tue les parents, et laisse agent fils qui jouera le même rôle de ses parents, mais avec des bonnes performances (cf., la figure 3.8).

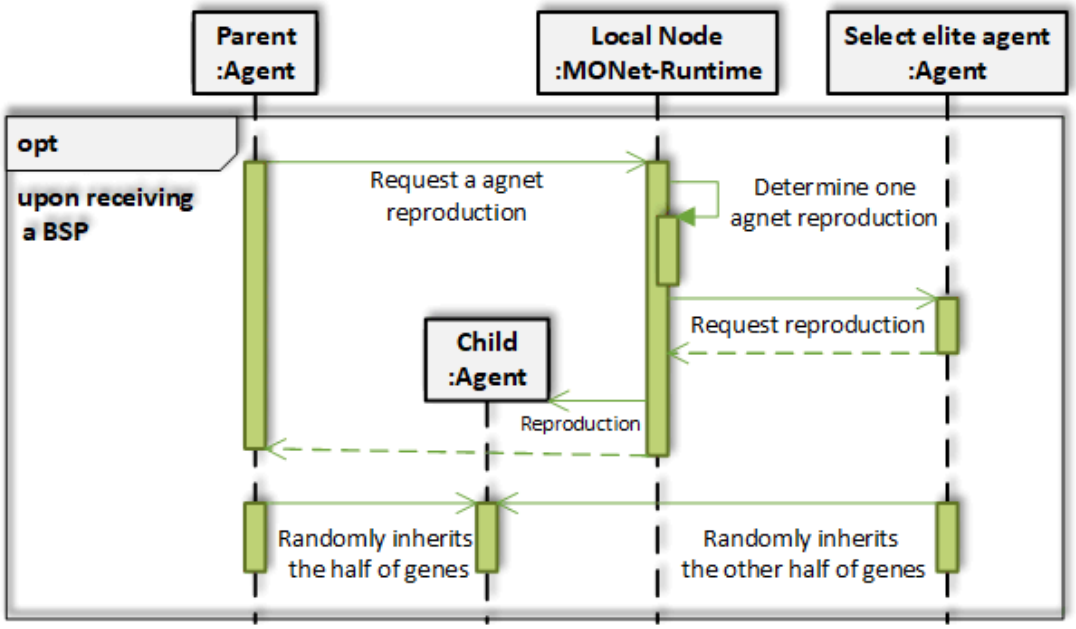

Figure 3.8 Le diagramme de séquence de comportement de reproduction

3.3.3 Les caractéristiques comportementales de l'agent

Chaque agent possède un ensemble d'attributs qui le distingue des autres, les gènes sont un sous-ensemble des attributs, qui sont responsable de déterminer les caractéristiques comportementales de l'agent lesquelles se divisent en deux type « *héritées* et *acquises* » :

- **Les caractéristiques acquises** : sont les caractéristiques que l'agent acquiert de son environnement, dans le MONet-R, chaque agent acquiert une caractéristique qui lui permet de déterminer la façon d'invoquer son comportement de Swarming, le gène qui la représente est le rang de l'agent (ρa), où il prend une valeur entière.

- **Les caractéristiques héritées** : sont les caractéristiques que l'agent hérite de ses parents, où chaque agent hérite l'ensemble de coefficients {$w1$, $w2$, $w3$} de l'équation 3.4 comme des gènes de ses parents et la valeur de chaque coefficient est de {-1,0,1}. Le triple (w_1, w_2, w_3) détermine la catégorie de l'agent i.e., comment l'agent exécute le comportement du déplacement, ceci génère quatre catégories (*Directional, Chemotaxis, Detour et Sidestep*) comme illustre dans le tableau 3.2.

Les gènes de l'agent			**La catégorie**	**Les caractéristiques**
w_1	w_2	w_3		
?	**0**	**0**	Directional	– L'agent se déplace à la SB à travers le plus court chemin – Il se base sur la phéromone BSP et néglige MP et AP. ☑ Il réduit le nombre de saut et par la suite la latence ☒ Les nœuds dans le plus court chemin consomment beaucoup plus d'énergie que les autres (risque de rupture du réseau)
?	**+1**	?	Chemotaxis	– L'agent peut suivre les traces de passage des autres agents sur les nœuds – Il se base beaucoup plus sur la phéromone MP ☒ Ces traces peuvent être le plus court chemin.
?	**-1**	?	Detour	– L'inverse de *chemotaxis* – L'agent essaye d'éviter les traces de passage récent des autres agents sur les nœuds. ☑ Il permet de distribuer la consommation énergétique sur les nœuds du réseau (éviter de rupture du réseau). ☒ La latence est aussi augmentée.
?	?	**-1**	Sidestep	– C'est un cas particulier de *détour*. – L'agent essaye d'éviter les traces de passage des autres agents sur les nœuds. – Et aussi éviter de se déplacer vers les nœuds qui ont été référencé par les phéromones AP. ☑ Il permet de réduire le nombre de déplacement échoués (la consommation énergétique) et il augmente le taux de réussite ☒ La latence est aussi augmentée.

Tableau 3.2 Les caractéristiques de chaque catégorie de l'agent

3.4 Le MONet- serveur

MONet-Server est un outil d'optimisation qui se base sur les principes du problème multi-optimisation (MOP) et implémente aussi bien un algorithme génétique multi-objectifs permettant aux agents du SMA d'évoluer leurs propres caractéristiques comportementales héritées (i.e., leurs gènes hérités), afin qu'ils s'adaptent aux mieux avec leur environnement global. MONet-S fonctionne comme un serveur central qui évalue les agents pour sélectionner un ensemble d'agents, appelé agents élites qui possèdent les meilleurs gènes, ces agents élites seront utilisés par MONet-R pour qu'il engendre une nouvelle génération d'agents du SMA. Cette évaluation des agents s'effectue en fonction des objectifs opérationnels de chaque agent et aussi en fonction des contraintes de l'utilisateur s'il y en a, où MONet-S calcule la fonction de fitness de chaque agent et l'agent ayant la valeur la plus basse est

dominé par l'agent ayant la valeur la plus haute. Tout d'abord, nous présentons les aspects théoriques de l'optimisation multi-objectif dans la section suivante.

3.4.1 Optimisation multi-objectif

Dans sa forme la plus générale, un problème d'optimisation s'écrit mathématiquement :

$$(P)\begin{cases}\min\limits_{x} f(x) = (f_1(x), f_2(\mathrm{x}), \ldots, f_p(x)) \\ s.q. \begin{vmatrix} g_j(x) \le 0 & \forall j \in \{1,\ldots,G\} \\ h_k(x) = 0 & \forall k \in \{1,\ldots,H\} \\ x_i^l \le x_i \le x_i^u & \forall i \in \{1,\ldots,n\} \end{vmatrix}\end{cases} \tag{3.5}$$

Où $f(x) = (f_1(x), f_2(\mathrm{x}), \ldots, f_p(x))$ représente les différents objectifs du problème, $x = (x_1, x_2, \ldots, x_n)$ désigne les variables d'optimisation. Le vecteur de variables $\vec{x}$ peut contenir des variables continues, binaires, discrètes ou encore des permutations. Il comprend les variables de positionnement, ainsi que les variables des attributs. Chaque variable x_i est comprise entre deux bornes x_i^l et x_i^u. Les fonctions g_j et h_k expriment respectivement les contraintes d'inégalités et d'égalités du problème. Ces contraintes sont comprises dans ces fonctions et sont gérées de différentes manières suivant le type de modélisation choisi.

Dans le cadre de problèmes multi-objectifs, il n'y a pas une solution optimale unique, mais un ensemble de solutions optimales. La résolution de tels problèmes consiste à trouver l'ensemble de ces solutions optimales. Ces dernières sont appelées solutions efficaces. Dans cet ensemble, aucune des solutions n'est meilleure qu'une autre, aucune n'étant systématiquement inférieure aux autres sur tous les objectifs.

Un problème d'optimisation multi-objectif (*MO*) consiste à minimiser simultanément un ensemble de *p* fonctions objectifs :

$$\min_{x \in E} f(x) = \min_{x \in E}(f_1(x), f_2(\mathrm{x}), \ldots, f_p(x)) \tag{3.6}$$

Où *E* désigne l'espace des solutions réalisables. Chaque solution *x* a pour image dans l'espace des objectifs (R^p) un point *y* : $y = f(x)$ avec $y \in \mathbb{R}^p$.

Dans le contexte multi-objectif, on utilise le principe de dominance pour comparer deux points.

Définition 3 (Définition de la dominance)

Une solution $x \in E$ *domine* $x' \in E$ *si elle vérifie :*

$$\begin{vmatrix} \forall i \in \{1,\ldots,p\},\ f_i(x) \le f_i(x') \\ \exists j \in \{1,\ldots,p\},\ f_j(x) \prec f_j(x') \end{vmatrix} \tag{3.7}$$

On note cette relation de dominance $x \le x'$.

Cette définition permet d'introduire le concept de Pareto-optimalité

Définition 4 (Définition de Pareto-optimal)

Une solution ***x*** *est dite efficace ou Pareto-optimale si elle n'est dominée par aucune autre solution appartenant à* ***E****. L'ensemble de ces solutions sont également appelées solutions non dominées.*

Définition 5 (Front de Pareto) : *L'image des solutions efficaces forme dans l'espace des objectifs un ensemble de points non dominés, communément appelé front de Pareto.*

3.4.2 Les Objectifs Opérationnels vs les contraintes

Chaque agent, pendent son déplacement de son nœud source à la SB, estime plusieurs objectifs contradictoires. En général, ces objectifs sont liés à la quantité de données capturées, la qualité de données (i.e., nouveauté ou ancienneté) et la consommation d'énergie. Dans notre cas, nous nous intéressons à quatre objectifs qui sont : la latence (L), le coût (C), le taux de réussite (S) et le degré d'agrégation des données (D), où le taux de réussite et le degré d'agrégation des données sont liés au rendement de données, la Latence est liée à la qualité de données et le Coût est lié à la consommation d'énergie.

Le MONet s'évertue d'évoluer les agents afin de minimiser la latence et le coût, et aussi de maximiser le taux de réussite et le degré d'agrégation des données. Ainsi il offre à l'utilisateur la possibilité d'associer à chaque objectif opérationnel une ou plusieurs contraintes suivant lesquelles représentes les exigences de QoS, (e.g., la valeur de la latence doit être inférieure que à une valeur prédéfinir) et tous les agents qui ne satisfissent pas les contraintes ne sont pas tenus en compte dans l'opération de reproduction. Dans ce qui suit, nous définissons ces objectifs.

- **La latence (L)** : désigne le temps nécessaire à un agent pour se déplacer de nœud source à la station de base à travers le réseau. Elle est mesurée sous la forme d'un rapport de temps de déplacement de l'agent à la distance physique entre la station de base et le nœud source. La latence représente le paramètre de QoS.

$$L = \frac{t_{arrivé} - t_{stamp}}{\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2}} \tag{3.8}$$

  Avec, t_{stamp} : le temps de capture des données transportées sur l'agent, $t_{arrivé}$: le temps d'arrivé de cet agent à la SB, (x_i,y_i) et (x_s,y_s) sont respectivement les coordonnées du nœud source et la station de base. On suppose que le MONet-S connaît les coordonnées de chaque nœud avec une méthode de localisation (e.g., GPS).

- **Le coût (C)** : représente le rapport de l'énergie qui est consommée par l'agent pendant son déplacement de nœud source à la SB à l'énergie qui sera consommée par le même agent s'il se déplace sans échec vers la même SB sur le plus court chemin (il peut que ce plus court chemin est virtuel). Sachant que tous les agents pour faire un seul saut, ils consomment une valeur fixe de l'énergie, donc ce rapport est seulement mesuré en fonction du nombre de sauts et est comme suit :

$$C = \frac{N_{hop}}{1 + floor\left[\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2} \,/\, r\right]} \tag{3.9}$$

Avec, N_{hop} : est le nombre total de sauts, y compris les sauts échoués, qui ont été effectués par l'agent jusqu'il arrive à la SB, r : la portée de communication radio de chaque nœud et *floor* est la partie entière.

- **Le taux de réussite (S)** : est mesuré comme le rapport du nombre de sauts réussies (N_{succ}) au nombre total de sauts, y compris les sauts échoués (N_{fail}), le taux de réussite représente les paramètres de fiabilité de QoS.

$$S = \frac{N_{succ}}{N_{succ} + N_{fail}} \tag{3.10}$$

- **Le degré d'agrégation des données (D)** : est mesuré comme étant le nombre de données captées qu'un agent transporte simultanément. Dans le cas où, un agent a déjà effectué le comportement de *swarming*, alors au-moins deux données de sources différentes sont fusionnées dans celui-ci. Grâce ce processus, la consommation d'énergie est réduite car le nombre des agents qui se déplacent vers SB dans le réseau est réduit. Cependant, cela peut augmenter la latence car les agents doivent attendre dans les nœuds des autres agents.

3.4.3 L'optimisation des agents

Comme nous l'avons mentionné auparavant, les agents estiment des objectifs contradictoires en fonction de leurs gènes, le problème d'optimisation des agents dans RCSF afin qu'ils donnent les meilleures performances est un problème NP-Difficile [248]. Par conséquent, ce problème ne pourrait pas être résolu grâce à des méthodes exactes, mais résolu par des méthodes non-conventionnelles telles que les algorithmes évolutionnaires [249]. Ce problème peut être décrit comme suit.

Soit un RCSF constitué de nœuds capteurs, chaque nœud capteur possède un agent qui doit assurer au-moins deux tâches essentielles : collecter des données, et les transporter à station de base la plus proche. Ceci n'est possible que si les phéromones sont disponibles sur les nœuds (au-moins BSP). Les données ainsi collectées ne sont pas forcements acheminés à la SB sur le plus court chemin. Ceci est déterminé en fonction des gènes de l'agent (w_1,w_2,w_3), la diversité dans ces gènes, influe sur l'estimation des objectifs opérationnels (L, C, S, D) des agents. Ces objectifs sont généralement des objectifs contradictoires qui sont aussi influés par le contexte et les conditions du réseau.

L'objectif de notre travail est de trouver les gènes (i.e., les agents) qui estiment des compromis entre la minimisation de la latence (L) et le coût (C) et la maximisation du taux de réussite (S) et le degré de l'agrégation de données (D) c'est-à-dire les gènes s'adaptant aux conditions de l'environnement. Ceci n'est possible que si l'on utilise des méthodes de résolution multi-objectifs, nous avons déjà présenté l'état de l'art sur les travaux effectués dans cette optique (cf., le chapitre 2). Par la suite, nous allons présenter l'algorithme que nous utilisons dans notre travail.

3.4.4 Méthode utilisée

Notre travail peut être exprimé par un problème d'optimisation multi-objectifs qui a été prouvé comme étant NP-Difficile. Afin de résoudre un tel problème nous avons retenu cinq algorithmes évolutionnaires : *Genetic Algorithms* (GA), *Non-Dominated Sorting Genetic Algorithm* (NSGA-II) [250], *Strength Pareto Evolutionary Algorithm* (SPEA-II) [251],*Pareto Envelope based Selection* (PESA-II) [252], et *Dierential Evolution* (DE) [253]. Ces algorithmes sont, à notre connaissance, très utilisés pour trouver des solutions proches de l'optimal dans un délai minimal lors de la résolution de problèmes NP-Difficile comme le problème d'optimisation des agents du RCSF. De plus, l'optimisation des agents du RCSF est un problème multi-objectifs NP-difficile, ce qui valide le choix d'une méthode sélectionnée. Cet algorithme est caractérisé par trois phases distinctes, qui sont : (1) Sélection et évaluation, (2) Reproduction, (3) Classement des solutions et remplacement. Donc nous choisissons l'algorithme génétique (AG).

3.4.5 L'algorithme génétique (AG)

L'algorithme génétique est une stratégie adaptative et méthode d'optimisation globale. Cet algorithme s'inspire de la génétique de population, ainsi que, la compréhension de la structure mendélienne (chromosome, gènes, allèles, . . .) et ses mécanismes (croisement et mutation) [254]. Cet algorithme opère en quatre parties (opérations) distinctes : *Evaluation, Sélection, Croisement,* et *Mutation.* Dans chaque génération de l'algorithme, les parents sont sélectionnés pour entreprendre plusieurs opérations (*croisement* et *mutation*). Chaque opération est réalisée avec une probabilité prédéfinie. Dans notre cas, nous avons implémenté les parties de cet algorithme dans deux environnements déférents, grâce à cette technique son temps d'exécution est réduit, car ses deux parties sont exécutées en parallèles. Nous avons implémenté dans chaque composant de MONet une partie comme suit :

Dans **MONet-S**, la phase de sélection, qui détermine le nombre de fois qu'un individu participe à la reproduction en une seule génération de l'algorithme. Les individus ayant les meilleures valeurs de fitness seront fréquemment sélectionnes par rapport aux autres. Les individus sélectionnés sont appelés *parents*, et sont choisis pour participer à la phase de reproduction. Nous adoptons une sélection par tournoi, pour choisir les individus participants à la phase de reproduction.

Dans **MONet-R**, le croisement, qui consiste à l'application d'opérations avec une certaine probabilité P_c aux individus précédemment sélectionnés. La probabilité de croisement définit la proportion de la population (nombre de parents) qui va être utilisée par un opérateur de croisement. Dans les premières générations de l'algorithme, l'opérateur de croisement joue un rôle d'exploration de l'espace de recherche [255]. Ceci est vrai, car l'opération de croisement génère de nouvelles solutions qui se trouvent dans des espaces inexplorés auparavant. Les nouvelles solutions générées par les opérations de reproduction et évaluées, sont ensuite classées et une phase de remplacement de la population est entreprise. Dans cette phase, les meilleures solutions sont sélectionnées afin de remplacer la population de la génération précédente. Il existe une multitude de méthodes de croisement qui sont adaptées à des problèmes spécifiques. À partir des travaux effectués dans cite, nous pouvons dire qu'il existe 21 types de croisement. Ces types de croisement différent dans la manière de découpage et

d'assemblage des chromosomes (parents) pour trouver de nouvelles solutions (enfants). Nous utilisons dans notre algorithme un croisement à un point. Ce type de croisement sera expliqué en détails dans les sections suivantes.

Dans **MONet-R**, aussi l'autre opération de reproduction et l'étape de *mutation* avec une plus petite probabilité notée P_m appelée taux de mutation. Cet opérateur introduit une perturbation à la solution, ce qui maintient une diversité au sein de la population. L'opérateur de mutation évite la convergence prématurée à un optimum local. Nous intégrons dans notre algorithme une méthode de mutation binaire qui sera détaillée dans les sections suivantes.

3.5 Adaptation de l'algorithme génétique au problème d'optimisation du RCSF.

Afin de résoudre le problème d'optimisation des agents dans le RCSF, nous avons procédé à l'adaptation de l'algorithme génétique qui a été présenté précédemment. Ceci offre une facilité d'implémentation et s'adaptent aisément aux problèmes binaires. Son plus grand défaut reste la création d'opérateurs adaptés qui conduisent à l'efficacité de l'algorithme. Dans un ordre chronologique nous pouvons exprimer ces opérateurs comme suit :

- Premièrement, il faut formuler le problème d'optimisation c-à-d définir le vecteur de décisions et le vecteur de fonctions objectifs.
- Deuxièmement, il faut définir une représentation adéquate des solutions appelées "Chromosomes".
- Une fois le codage de la solution défini, nous procédons à la création de la population initiale. En général, cette partie est réalisée d'une manière aléatoire.
- Après que la population initiale créée, des opérateurs de reproduction sont appliqués afin de simuler l'évolution naturelle.

3.5.1 Formulation du problème :

Dans notre travail les individus de l'algorithme génétique sont les agents, où chaque agent est composé de trois (3) gènes qui représentent les coefficients de l'équation 3.4, ce qui nous permet de représenter le vecteur de décisions « *X* » comme suit :

$$\overrightarrow{X} = \begin{pmatrix} x_1 = w_1 \\ x_2 = w_2 \\ x_3 = w_3 \end{pmatrix} \tag{3.11}$$

Comme nous avons défini auparavant (cf., la section 3.4.2) le problème d'optimisation comme étant un problème de quatre (4) objectifs qui sont la latence (L), le coût (C), le taux de réussite (S) et l'agrégation de données (D), avec les deux premiers à minimiser et les autres à maximiser. Nous pouvons alors représenter le vecteur de fonctions objectifs *f(X)* comme suit :

$$f(\vec{x}) = \begin{cases} \min\ f_i(x) & i = 1, 2 \\ \max\ f_i(x) & i = 3, 4 \end{cases} \tag{3.12}$$

Tel que :

$$f(\vec{x}) = \begin{pmatrix} \min f_1 = \min(\mathrm{L}) \\ \min f_2 = \min(\mathrm{C}) \\ \max f_3 = \max(\mathrm{S}) \\ \max f_4 = \max(\mathrm{D}) \end{pmatrix} \tag{3.13}$$

Nous avons $\max(f) = \min(-f)$ donc :

$$f(\vec{x}) = \begin{pmatrix} \min f_1 = \min(\mathrm{L}) \\ \min f_2 = \min(\mathrm{C}) \\ \min f_3 = \min(-\mathrm{S}) \\ \min f_4 = \min(-\mathrm{D}) \end{pmatrix} \tag{3.14}$$

3.5.2 Codage du chromosome

Dans notre cas les variables de décision sont des variables binaires (i.e., une suite de nombres binaires composées de zéros et des uns). Nous pouvons les représenter par un vecteur de taille fixe qui est égal six (6) bits tel que chaque gène hérité (w_1, $w_2$ou w_3) est composé de deux (2) bits, dont le premier pour indiquer son signe (1 : signe négatif et 0 : signe positif) et le deuxième pour sa valeur, il prend une valeur de {-1, 0, +1}. Nous avons utilisé ce codage, car il facilite la création des opérateurs de mutation et de croisement. Un exemple d'individu (agent) est représenté dans la figure 3.9.

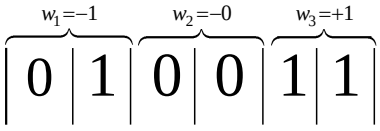

Figure 3.9 Exemple d'une solution (individu)

3.5.3 Création de la population initiale

En général, cette étape est réalisée lors du démarrage des algorithmes évolutionnaires. Elle permet de générer un grand nombre de solutions réalisables, tout en maintenant une diversité suffisante dans la population, permettant d'éviter le piège des optima-locaux. Le but principal de cette phase est de trouver un équilibre entre diversité et solutions de bonnes qualités.

Dans notre cas, la population initiale est créée pendant le déploiement des nœuds capteurs du RCSF, où au sien de chaque nœud capteur un agent est aléatoirement crée. Nous avons opté pour une génération aléatoire des agents. Ce choix a été motivé par le fait que cette manière de générer la population ne consomme pas trop de temps lors de l'exécution sur des grandes instances. Générer aléatoirement les agents permet de créer une diversité de la population tout en ne gardant que des solutions réalisables. La création de la population initiale est comme suit : la population initiale est composée de tous les agents, qui se déplacent à travers les nœuds du RCSF et arrivent au MONet-S à traves la SB. Un exemple de population initiale avec cinq (05) individus est représenté dans le tableau 3.3.

$Agent_1$	0 1 0 1 0 1
$Agent_2$	1 1 0 1 0 0
$Agent_3$	0 0 0 1 1 1
$Agent_4$	0 1 1 0 0 1
$Agent_5$	1 1 0 0 0 1

Tableau 3.3 Exemple d'une population initiale de cinq (05) individus

3.5.4 L'évaluation et la sélection des agents

Comme nous avons indiqués auparavant, le but principal de MONet-S est la sélection des agents parents (ou élites) qui sont utilisés par le MONet-R dans la phase reproduction. Cela est réalisée avec deux opérateurs l'*évaluation* et la *sélection.*

3.5.4.1 L'évaluation des agents

Notre travail consiste à l'application de l'optimisation multi objectif par l'algorithme génétique (comme MOGA) et l'approche de *Pareto,* plusieurs agents sont obtenus, les meilleurs se situent sur la frontière de Pareto (*solutions efficaces*), ils portent le rang « 1 », le reste des agents sont appelés solutions réalisables (solution moins désirables), ils portent un rang différent de « 1 » ce qui permet de regrouper tous les agents de population dans deux groupes (*non-dominés, dominés*). Cela est réalisée avec une fonction appelée fonction d'évaluation ou fonction de *fitness*. Dans notre cas, elle représente le rang de chaque agent (critère de sélection), tel qu'un rang égal « 1 » est mieux qu'un agent de rang égale à « 2 » et ainsi de suite. Donc ce regroupement de population se base sur le concept de *dominance,* appelé *Rang de Fonseca et Fleming*[1] qui est définit comme suite :

Selon l'équation 3.14, notre problème est un problème de minimisation multi-objectifs donc : On dit qu'un agent A **domine** un agent B, et on note $A \leq B$ s'il vérifie la définition 3 c.-à-d. ce qui suit :

$$\begin{cases} L_a \leq L_b \\ C_a \leq C_b \\ -S_a \leq -S_b \\ -D_a \leq -D_b \\ \exists x, x'; \mathrm{x} \in \{L_a, \mathrm{C}_a, S_a, D_a\}, \mathrm{x}' \in \{L_b, \mathrm{C}_b, S_b, D_b\},\ x \prec x' \end{cases} \tag{3.15}$$

La figure 3.10 illustre en détail l'opérateur de dominance. Au début, il doit extraire les objectifs qui sont estimés par chaque agent (ligne1), puis faire une comparaison entre leurs objectifs. Dire que,**A domine B**, si tous les objectifs de l'agent A sont inférieurs ou égaux aux objectifs de l'agent B et il y a au-moins un d'eux, qui est strictement inférieur (boucle 2). Cependant, B **domine A**, si tous les objectifs de l'agent B sont inférieurs ou égaux aux objectifs de l'agent A et il y a au-moins un d'eux, qui est strictement inférieur (boucle 3) et dans les autres cas, les agents **A et B non-dominés**.

Enfin, tous les agents non-dominés de la population courante sont dits de rang « 1 », qui vont approcher le *front de Pareto* de notre problème.

[1]Le rang d'un agent est alors défini comme le nombre d'agent dominés plus 1. C'est le rang utilisé dans l'algorithme MOGA

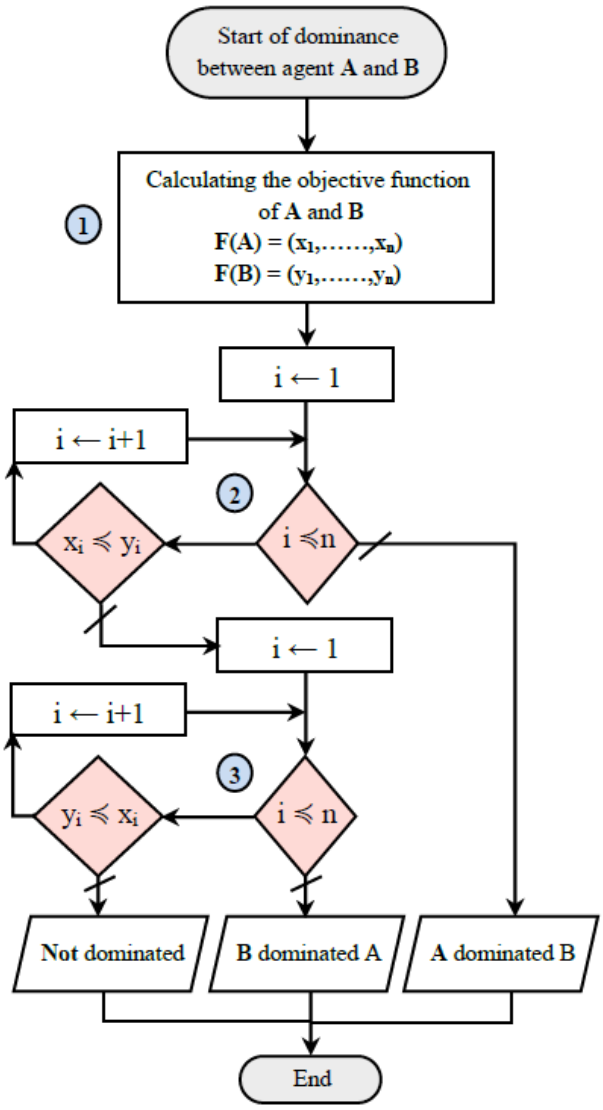

Figure 3.10 Principe de fonctionnement de l'opérateur de dominance.

3.5.4.2 La sélection des agents

Généralement, le front de Pareto est représenté par l'ensemble des solutions efficaces, cependant, il peut contenir un grand nombre d'agents, ceci ne permet pas de l'injecter dans le RCSF, donc nous avons besoin d'un mécanisme qui permet de sélectionner un sous-ensemble du front de Pareto avec, un cardinal très petit et ses éléments doivent assurer la diversité (*phénotypique* et *génotypique)*, c.-à-d. l'objectif premier de la sélection est de sélectionner les agents à fort potentiel de bonnes qualités et d'éliminer les autres tout en conservant la taille de la population. Ce sous-ensemble est appelé « agents *élites* » ou les *parents*. On peut alors définir une fonction de sélection, qui se base sur l'évaluation précédemment établie, nous allons identifier statistiquement les meilleurs agents de la population. Dans la littérature, plusieurs méthodes ont été élaborées, les plus connues étant la méthode de sélection proportionnelle aussi connue sous le nom de sélection par *roulette*. Cette dernière a été améliorée pour donner la méthode de sélection universelle stochastique. Cependant, d'autres méthodes existent, comme la sélection par *tournoi* qui utilise des comparaisons par paire de solutions pour sélectionner les meilleurs individus. C'est la technique la plus utilisée lors d'optimisation de

problèmes sous contraintes, où l'ensemble des valeurs des contraintes est regroupé dans un indice de violation de contraintes.

Dans notre cas, Nous adoptons une sélection par hypercube, pour choisir les agents élites du front de Pareto. Ceci est effectué dans l'espace des objectifs ; un espace hypercube en quatre (4) dimensions dont les axes représentent quatre objectifs (L,C,S,D). Chaque axe est divisé entre la valeur d'objectif la plus haute et la plus basse des agents du front de Pareto de sorte que l'espace contiendra de petits cubes (cf., la figure 3.11). Chaque agent non-dominé est tracé dans l'espace objectif en fonction des valeurs de ses objectifs. Si plusieurs agents sont tracés dans le même cube, un seul agent doit être sélectionné au hasard en tant qu'un agent élite. Si aucun agent n'est tracé dans un cube, alors aucun agent élite n'est sélectionné de ce cube. Cette méthode de sélection est conçue pour s'assurer la diversité des agents élites, ce qui permet d'éviter la convergence prématurée de l'algorithme c'est-à-dire elle sert à éviter la convergence vers un optimum local et améliorer aussi leur adaptabilité même aux conditions de RCSF imprévues.

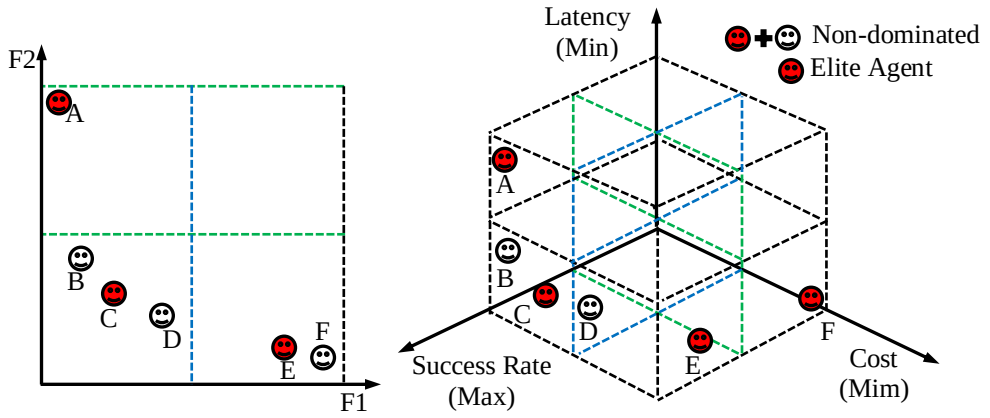

Figure 3.11 un exemple de hypercube, Détermine les agent élite pour un problème de **(a)** minimisation bi-objectif, **(b)** minimisation /maximisation tri-objectif, à partir des agents non-dominés trouvés.

La figure 3.11 illustre deux exemples de la méthode d'hypercube, la sous-figure 3.11 (a) présente un problème bi-objectif (coût et latence). Chaque objectif (axe) est divisé en deux intervalles, ceux qui créent quatre carrés au total, donc le nombre maximum d'agents élitaires est quatre. La sous-figure 3.11 (b) présente un problème tri-objectif (taux de réussite, coût et latence), lesquels créent huit cubes au total. Donc, le nombre maximum d'agents élitaires est huit. Dans les deux exemples, six agents non dominés (de A à F) sont tracés. Dans le deuxième exemple, trois agents (B, C et D) sont tracés dans le cube inférieur gauche, tandis que les trois autres agents (A, E et F) sont tracés dans trois cubes différents. À partir du cube inférieur gauche, un seul agent est sélectionné au hasard en tant qu'un agent élitaire (e.g., C). A, E et F sont sélectionnés comme agents élitaires car ils sont dans des cubes différents. Donc l'ensemble d'agents élite est composé de {A,C,E,F}. Alors que dans le premier exemple, les agents élites sont {A,C,E} car les agents (B, C et D) sont tracés dans le même carré et aussi (E et F) sont tracés dans un autre carré et A est tracé à part dans un autre carré.

La figure 3.12 illustre comment la sélection élite se produit au niveau du MONet-S dans chaque cycle d'activation. Au premier, le serveur collecte tous les agents qui lui arrivent via les SBs du réseau RCSF (ligne 1). Afin d'assurer l'élitisme qui est destiné à améliorer la vitesse d'évolution des agents la population initiale est constituée de ces agents arrivants et des agents d'élites récemment élus (ligne 2), l'opération de dominance est effectuée sur tous les agents de la population initiale, résultant un ensemble d'agents non-dominés (i.e., le front de Pareto) (boucle 3). Finalement, il utilise la méthode hypercube pour choisir parmi ces agents non-dominés quelques agents comme des agents élites qu'il envoie à la SB (lignes 4 et 5). Cette dernière encapsule les gènes de ces agents élites dans sa propre phéromone qu'elle diffuse vers les nœuds capteurs du RCSF. Lorsqu'un nœud capteur reçoit la phéromone de SB, il doit alors faire l'opération de la reproduction (croisement et mutation), ce que nous allons expliquer dans la section suivante.

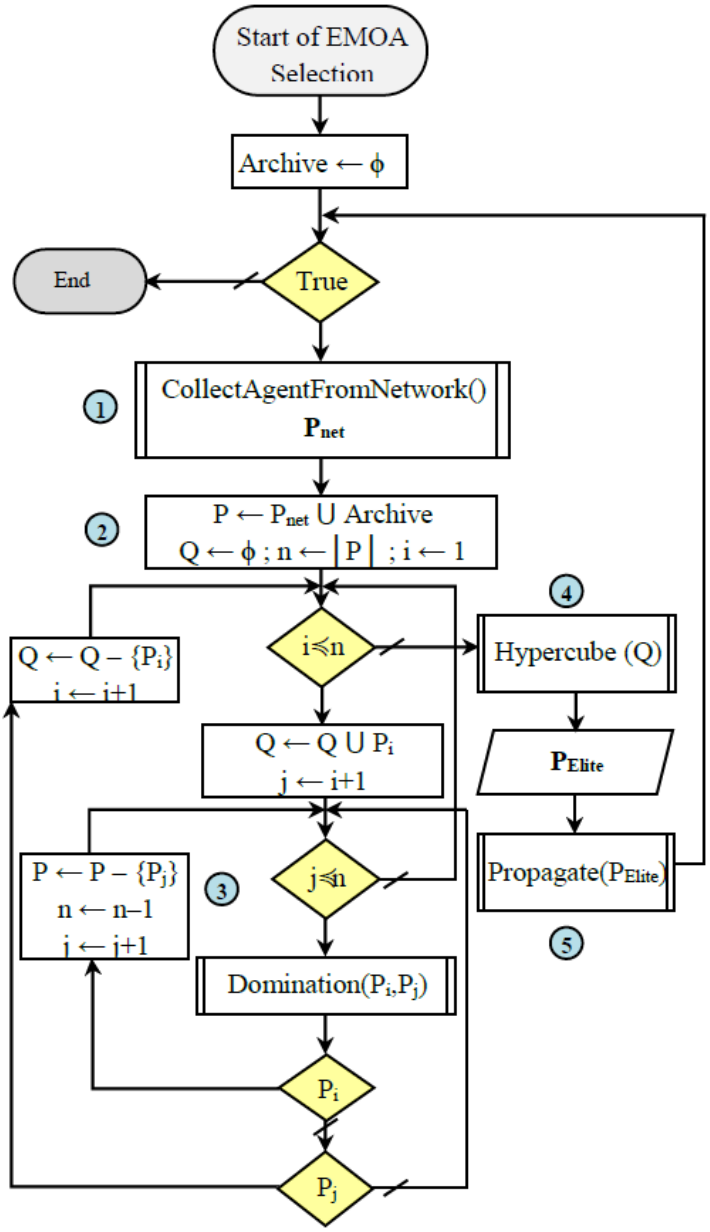

Figure 3.12 Principe de fonctionnement de l'opérateur de sélection.

3.5.5 La reproduction (variation) des agents :

Dès qu'un nœud capteur reçoive la phéromone BSP, son agent local doit invoquer le comportement de reproduction, où il choisit un agent de l'ensemble d'agents élites, qui est le plus proche de lui, afin qu'il représente un parent. Ce choix s'est effectué en se basant sur la distance génotypique entre l'agent local et l'agent élite, où l'agent qui génère la plus petite distance sera choisi. La distance génotypique est mesurée selon l'équation (3.16) suivante :

$$D(l,e) = \sqrt{\sum_{i=1}^{3} \left(w_i^e - w_i^l \right)^2} \tag{3.16}$$

Où l et ℓ représentent l'agent local et l'agent élite, w_i^l et w_i^e représentent leurs gènes respectivement.

En général, on peut trouver deux opérateurs principaux (croisement et mutation) :

3.5.5.1 L'opérateur de croisement

L'opérateur de croisement permet d'intensifier la population d'agents au sien du RCSF. Généralement, le croisement consiste à appliquer un processus avec une certaine probabilité (qui s'appelle aussi le taux de croisement $P_c \in [0, 1]$) aux deux agents parents sélectionnés auparavant, ce taux de croisement définit la proportion de chaque agent parent dans la population qui va être utilisée par un opérateur de croisement. Il permet le mélange des gènes de deux agents parents, résultant d'une ou plusieurs agents (généralement deux) enfants selon l'opérateur choisi. Les trois opérateurs très utilisés dans la littérature sont : le croisement en un point, le croisement multi-points et le croisement uniforme. Dans notre algorithme, nous avons utilisé le croisement en un point qui est définit comme suit :

- **Le croisement en un point**: permet de mélanger les données (les gènes) de deux individus (agents) tout en gardant une certaine cohérence. La première étape consiste à choisir le point k, celui-ci étant aléatoirement fixé entre 2 et N-1 où n est la taille de la configuration (i.e., la structure génotypique de l'agent) dans notre cas N=6. Soit deux configurations choisies X et X', la création des nouvelles configurations Y et Y' se fait de la manière suivante :

$$Y_i = \begin{cases} X_i & \forall i \le k \\ X_i^{'} & \forall k \le i \le N \end{cases}$$
$$Y_i^{'} = \begin{cases} X_i^{'} & \forall i \le k \\ X_i & \forall k \le i \le N \end{cases} \tag{3.17}$$

  Nous avons opté ce type de croisement parce que c'est un opérateur s'adapte bien au codage binaire que nous avons utilisé. La figure 3.13 représente le croisement en un point.

  Après cette opération, on sélectionne l'un ou l'autre de l'agent enfants pour le conserver dans le nœud (pour éviter les conflits entre les agents, il faut conserver toujours un seul agent dans le nœud), et les autres agents sont détruite par le MONet-R.

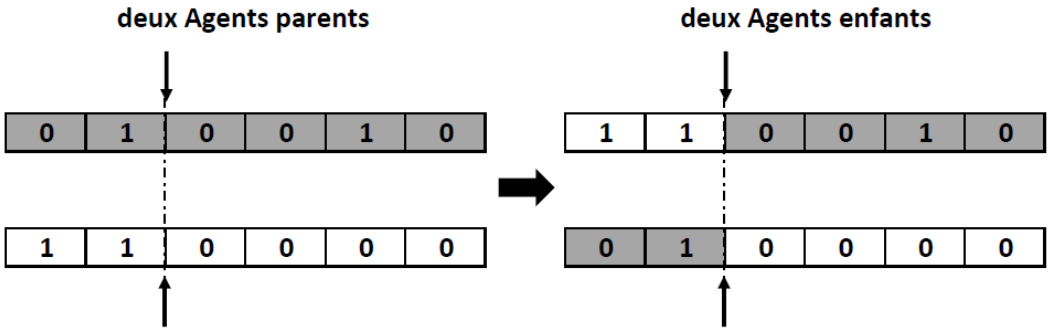

Figure 3.13 Exemple de croisement entre deux agents parents en un point

3.5.5.2 Operateur de mutation

Une fois l'étape de croisement effectuée, l'opérateur de mutation est appliqué avec une faible probabilité (Pm) appelée aussi taux de mutation. Cet opérateur introduit une perturbation dans la solution. Ceci a pour vocation de diversifier la population d'agent et explorer l'espace de recherche. Aussi, l'opérateur de mutation évite de converger vers des optima locaux. Cet opérateur se sert donc de la randomisation pour effectuer des changements aveugles. Nous avons opté pour l'utilisation d'une méthode binaire de mutation. La valeur d'un *bit* de la structure génotypique, choisi au hasard dans l'intervalle [1, N], **est changeante** au 0 s'il était 1, et 1 sinon, comme indiquée par l'équation (3.18) suivante.

$$\left|\begin{array}{l} \forall i \le N \\ tirage = U(0,1) \\ x_i = \begin{cases} x_i & \text{si } tirage \succ P_m \\ 1 - x_i & \text{si } tirage \le P_m \end{cases} \end{array}\right. \qquad (3.18)$$

Où *tirage* est un tirage au sort, qui est calculé par la loi uniforme $U(0,1)$ entre 0 et 1, x_i est la valeur de *bit* choisi. Ce type de mutation reste, le plus adapte aux variables binaires. Un exemple de mutation est représenté par la figure 3.14.

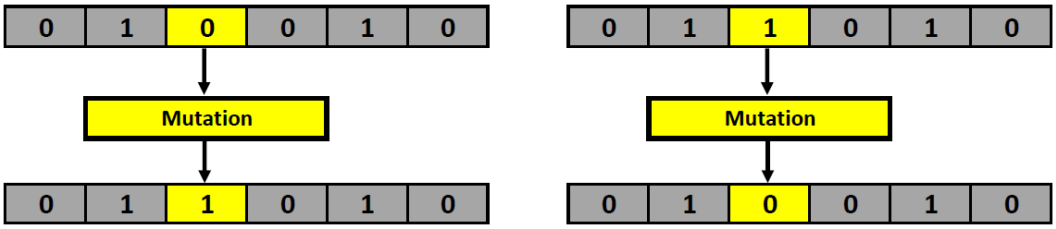

Figure 3.14 Deux exemples de mutation

3.6 Le développement et la simulation

3.6.1 Le développement

Comme nous l'avons mentionné auparavant, le MONet se compose de deux parties distinguées, ce qui nous permet des développements plus simples, plus rapides et séparés. Ceci est fait en utilisant deux

différents langages de programmations, qui sont JAVA et *nes*C [256]. Dans ce qui suit nous présentons les éléments principaux de l'implémentation de chaque partie.

3.6.1.1 L'implémentation de MONet-server

Nous avons implémenté les éléments de MONet-S en JAVA. En effet, le MONet-S constitue d'ensemble de classes et d'interfaces, chacune a pour un objectif et contient un ensemble de méthode, ces interfaces et ces classes importantes, sont comme suites :

- L'interface *Agent*, définit le comportement de la classe agent, sans son implémentation, c'est un ensemble des méthodes abstraites nécessaire à manipuler l'agent comme la fonction qui récupère l'identificateur de l'agent et du nœud source, les valeurs d'objectifs et etc.
- L'interface *Objective*, définit le comportement de la classe de chaque objectif, contient toutes les méthodes nécessaires pour les manipuler comme la fonction qui récupère les valeurs d'objectifs et fait la comparaison entre les objectifs de même type, et etc.
- L'interface *Engine*, définit le comportement qui permet de recevoir les agents du réseau, sans son implémentation.
- L'interface *Network*, définit le comportement qui contient toutes les méthodes qui permettent d'obtenir les agents du réseau, et d'injecter les agent élites dans la SB, et etc.

Nous avons implémenté le comportement de l'interface *Agent* dans une classe appelée *AgentS.class* et pour chaque objectif de l'agent, nous avons créé une classe avec le même nom (e.g., *Latancy.class*) , qui est une implémentation de l'interface *Objectve.class*. L'implémentation de la méthode qui collecte les agents du réseau et la méthode qui injecte les agent élite dans le réseau, a été effectué dans la classe appelée *AgentCollectionNet.class*. En fin, nous avons implémenté l'opérateur d'*évaluation* et de *sélection* de GA dans la classe *MONetEngine.class*. la figure 3.15 Illustre l'interface utilisateur de MONet-serveur.

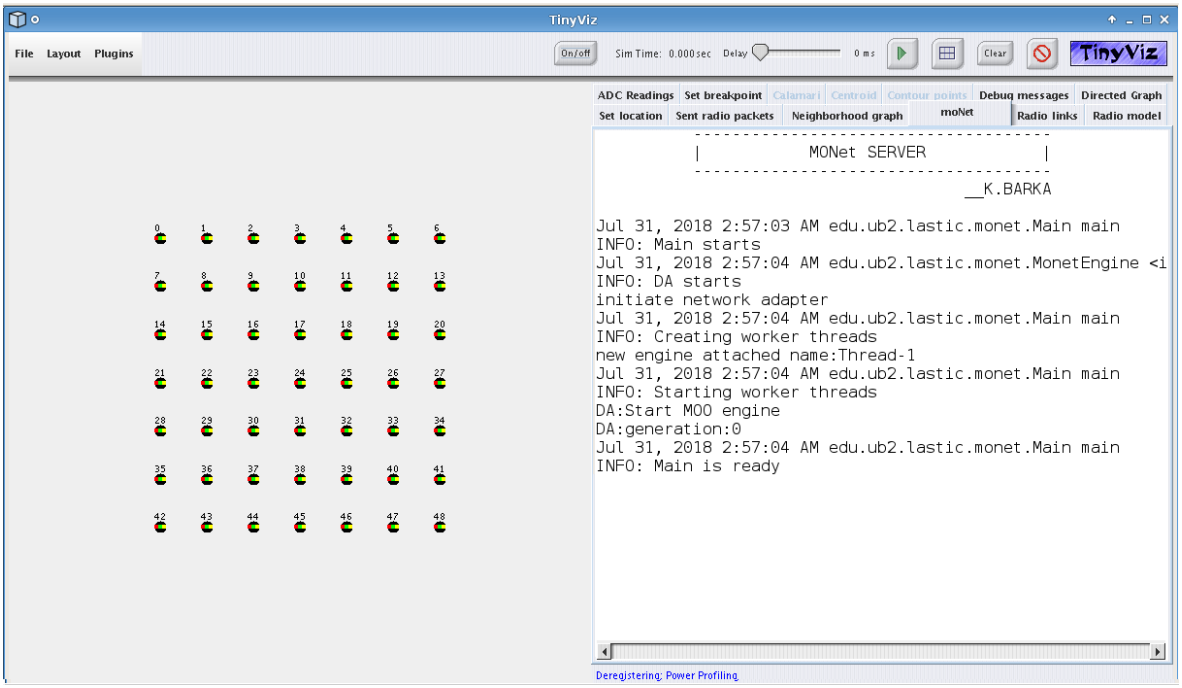

Figure 3.15 l'interface utilisateur de MONet-S sous TinyViz

3.6.1.2 L'implémentation de MONet-runtime

MONet-runtime est un Framework middleware qui opère au niveau du système d'exploitation du nœud capteur, dans notre cas, est le TinyOS, pour cela nous l'avons développé en langage nesC. Dans MONet-R deux concepts importants, sont l'agent et les phéromones. Dans ce qui suit, nous présentons leurs explications.

L'implémentation de l'agent : Comme nous l'avons mentionné dans la section 3.3.1, que l'agent est un concept logiciel ce que nous implémentons avec la structure qui est illustré par la figure 3.16 Où :

- `Id_A` : ce champ est un identifiant de l'agent (le numéro de l'agent).
- `Type (4 bits)` : indique le type de l'agent, ce champ est réservé à une utilisation future (maintenant type = 0)
- `Status(4bits)` : il s'agit de l'état actuelle de l'agent, nous citons six états (inconnu, normal, nouveau, mort, en mouvement et déplacé).
- `Energy_Level` : ce champ indique la valeur restante de l'énergie logique qui est acquis par l'agent (cf., la section 3.3.2)
- `Source_Id` : l'identificateur de nœud source de l'agent
- `Prev_Hop` : l'identificateur du dernier nœud expédiant de l'agent
- `Genotypic (8 bits)` : ce champ indique les trois valeurs de gènes hérités (w_i), chaque gène est représenté dans 2 bits et le 2 bits de poids faible sont réservés à une utilisation future.
- `Total_length` : ce champ indique le nombre d'octets de l'agent, son en-tête compris.
- `Timestamp` : ce champ indique le temps auquel les données ont été capturé par l'agent.
- `Hop_Count` : il s'agit du nombre de sauts qu'un agent a effectué jusqu'à l'instant, y compris les sauts échoués.
- `Trans_Count`: il s'agit du nombre de transmission qu'un agent a effectué jusqu'à l'instant.
- `Data_Count` : il s'agit du nombre de données qu'un agent transporte à la fois.
- `Option (8 bits)` : ce champ est réservé à une utilisation future. Il occupe un octet afin d'obtenir une entête agent multiple de 32 bits. La valeur des bits d'option est 0.
- `Data` : ce champ contient les données captées.

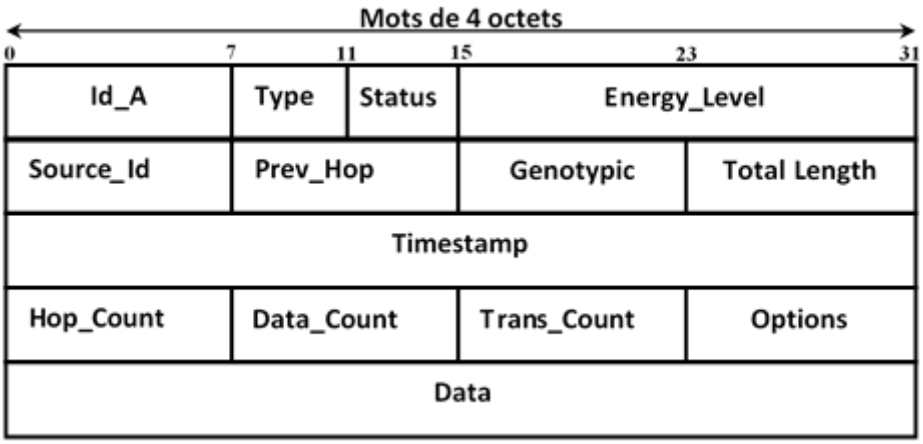

Figure 3.16 la structure de l'agent mobile

L'implémentation de la phéromone :**la phéromone** est un petit message que nous implémentons avec la structure illustrée par la figure 3.17. Pour les tailles des phéromones MP et AP sont fixées (32 bits) et celle de la phéromone BSP est variable selon le nombre d'agents élites (dans notre cas la taille maximale est 64 bits). Où :

- *Type_Ph*: ce champ indique le type de phéromone, malgré que nous avons actuellement trois types de phéromones, cependant il est codé sur 3 bits, ceci pour une utilisation future, où :
    - 0 :la phéromone de station de base BSP
    - 1 :la phéromone de migration MP
    - 2 : la phéromone d'alerte AP
- *TTL_Ph* (5 bits) : il s'agit d'une valeur initialisée par l'émetteur (l'agent ou SB) et qui est décrémentée de 1 à chaque fois que la phéromone passe dans un nœud capteur. Si le *TTL_Ph* arrive à la valeur 0, la phéromone est détruite, ce champ est codé sur 5 bits, ce qui permet de spécifier des valeurs initiales de 1 à 31.
    - La valeur initiale de 1 est utilisée par l'agent pour s'assurer que les phéromones MP et AP ne sont envoyées qu'à ses voisins immédiats (un seul saut).
    - La valeur initiale de 31 (l'infini) est utilisée par la SB pour désactiver le rôle de ce champ afin d'assurer la propagation horizontalement de sa propre phéromone BSP dans tout le RCSF (en cas où le réseau dépasse l'échelle).
    - Les autres valeurs sont utilisées, soit par l'agent, soit par la SB pour limiter le nombre de nœud qui renvoient la phéromone

  Ce mécanisme assure la destruction des phéromones qui se perdent sur le réseau. Ainsi ces phéromones perdues n'encombrent pas indéfiniment le réseau (boucle à l'infini).
- *Concentration* : il s'agit de la valeur de concentration de la phéromone. Si la phéromone est BSP, alors après chaque passage par un nœud ce champ est décrémenté. La valeur initiale est généralement fixée par la SB.
- *Source_Id* : l'identificateur de nœud source de la phéromone.
- *Opt* : ce champ joue deux roules selon le type de la phéromone où :
    - Si la phéromone est MP (resp. AP) alors ce champ indique l'identifiant du nœud de destination vers lequel un agent s'est déplacé (resp., du nœud auquel un agent n'a pas pu se déplacer).
    - Si la phéromone est BSP alors ce champ est divisé en deux sous-champs de même taille, dont le premier indique le nombre d'agents élites c'est-à-dire la langueur de champ GEA en octet, et l'autre indique la valeur de l'entropie récente (cf., l'équation 3.19)
- *GEA* : ce champ contient les gènes de tous les agent élites, il concerne seulement la phéromone BSP.

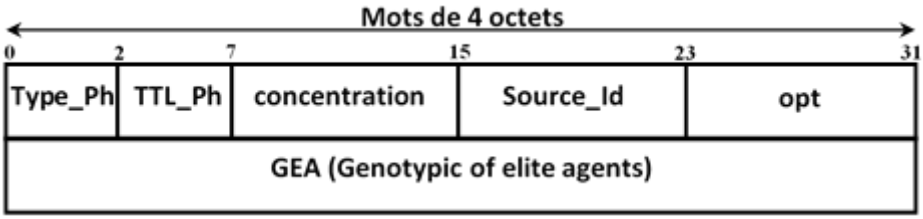

Figure 3.17 la structure générale d'une phéromone

3.6.2 Simulations et discussion des résultats

3.6.2.1 L'environnement de simulation

Afin d'évaluer l'efficacité du MONet, en termes des performances, d'auto-organisation, d'auto-configuration, d'auto-optimisation et de self-healing, nous avons opté pour des simulations qui ont été effectuées à l'aide du simulateur PowerTOSSIM avec le système d'exploitation TinyOS qui est dédié à la simulation des RCSF.

Toutes les simulations sont effectuées sur un RCSF qui se compose de 49 nœuds capteurs et une SB. Les nœuds sont uniformément déployés dans la zone observée de 180x180 m^2, et forment une grille 7x7 où la distance diagonale entre deux nœuds voisins est de 30 mètres, et la SB est positionnée à un des coins de la zone (ex., le coin nord-ouest), qui se connecte au MONet-S via un port de connexion. Durant la simulation, la SB et tous les nœuds capteurs restent immobiles. Chaque nœud capteur ne peut communiquer qu'avec ses voisins à l'horizontale et à la verticale, donc le nombre maximal de voisinage d'un nœud est de quatre, comme c'est illustré dans la figure 3.18.

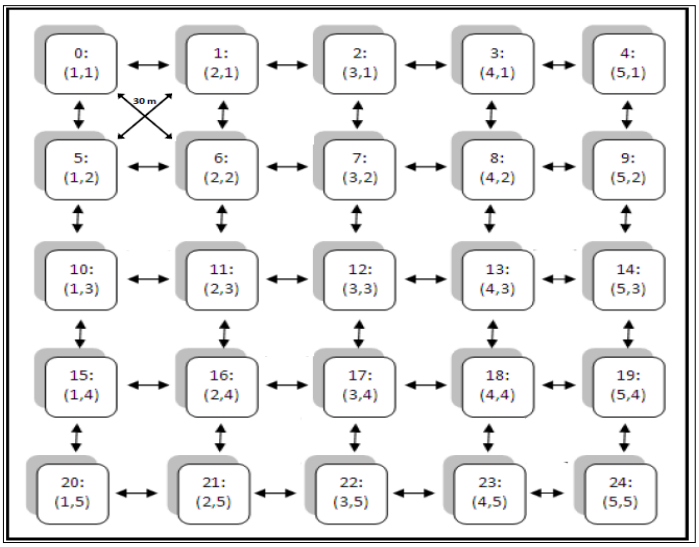

Figure 3.18 La topologie du réseau simulé (une grille 5x5)

Au début, tous les nœuds capteurs sont configurés par défaut et leurs périodes de sommeils sont initialisées à une minute. La conception de MONet peut permettre à l'agent, soit de collecter les données périodiquement, soit de détecter un évènement. Dans nos simulations nous avons opté à la deuxième, où nous avons implémenté l'évènement comme la différence entre les données captées (la valeur de la température) dans le cycle d'activation actuel et précédent, si la valeur de la différence est supérieure à un seuil, alors l'agent détecte un évènement. Tous les paramètres de notre simulation sont résumés dans le tableau 3.4 ci-dessous :

Paramètres	Valeurs
La surface du réseau	180m^2
Le nombre de nœuds	49
La localisation de la SB	(0.0)
L'énergie initiale des nœuds	2J …
La taille du paquet de données	255 octets
La taille d'agent	30 octets
La taille de phéromones BSP, MP et AP	8, 4 et 4 octets
Le taux métabolique (m)	1
Le coefficient (γ)	0.25
Le seuil de réplication initial (T_R)	25
La probabilité de croisement (P_c)	0.5
La probabilité de mutation initiale (P_m)	0.2
La consommation d'énergie dans l'état d'écoute	10 mA
La consommation d'énergie à l'état de diffusion	25 mA
La consommation d'énergie à l'état de veille	5 mA
Le période de sommeil initiale	1 min
Le période de sommeil	1< SP<5 min
Le cycle d'activation	5 min

Tableau 3.4 Paramétrage des simulations

3.6.2.2 Métriques et paramétrages des simulations

Afin de mesurer les quatre performances de MONet citées auparavant, nous utiliserons les métriques qui ne sont pas classiques dont :

- La première est l'*entropie*. Cette métrique mesure le degré de **l'auto-organisation**. Où elle indique à quel point les valeurs des objectifs qui sont estimées récemment par les différents agents, sont similaires. En général, sa valeur se situe entre 0 et 1. 0 (valeur basse) signifiant que les objectifs produits, sont plus similaires, cette dernière est calculée de la manière suivante :

$$E = \sum_{i \in S} \frac{n_i}{N} \log_2 \left(\frac{n_i}{N} \right) \tag{3.19}$$

  Où S est l'ensemble de tous les cubes qui sont formés par la méthode de sélection hypercube dans l'espèce objectif (cf., la section 3.5.4.2), ***ni*** est le nombre d'agents tracés dans le cube ***i*** et ***N*** le nombre total d'agents.

- La deuxième représente *les moyennes des valeurs d'objectifs* : Ces métriques mesurent **l'auto-optimisation** du réseau, qui signifie que nous recherchons à optimiser les performances des agents (i.e., {L,C,S,D}) dans le réseau (i.e., la génération récente) jusqu'à ce que celles-ci convergent vers l'optimum. Le meilleur cas correspond au cas où les valeurs de L et C soient basses et celle de S et D soient élevées. Elles sont calculées de la manière suivante :

$$\overline{O_i} = \frac{\sum_{a \in A} O_i(a)}{|A|} \tag{3.20}$$

Où $\overline{O_i}$ représente la moyenne de l'objectif $i \in \{L, C, S, D\}$, $|A|$ est la cardinalité de l'ensemble d'agents arrivés à la SB durant le cycle d'activation récent, $O_i(a)$ est la valeur de l'objectif i de l'agent a.

- La troisième est *le seuil de réplication de l'agent* : cette métrique mesure le **self-healing** d'un nœud, qui signifie que son agent local ne transporte que les données captées régulièrement et essaye de détecter et d'éliminer les fausses données positives qui apparaissent pendant un ou plusieurs cycles d'activations successifs. Dans ce cas, le seuil est plus élevé donc le mauvais fonctionnement du nœud. Ce seuil est calculé comme suit :

$$T_R(t) = (1-\gamma)T_R(t') + \gamma(e(t') + (s \times m)) \tag{3.21}$$

Où $T_R(t)$ (resp. $T_R(t')$) représente le seuil de réplication récent (resp., précédant), $e(t')$ est la valeur de l'énergie logique de l'agent dans le cycle précédent, s est la valeur absolue de la différence entre les valeurs de données captées dans les deux cycles successifs , m est un coefficient métabolique qui dépend de l'environnement, $m \in [0\ 1]$. γ est une valeur constante pour contrôler la sensibilité de TR au changement de données captées s, c.-à-d. si la valeur de γ est faible, alors les changements significatifs correspondent seulement à s, ce qui va influer sur le changement de T_R.

- En ce qui concerne **l'auto-configuration** qui est expliqué par l'ajustement de la *période du sommeil* entre sa valeur minimale et maximale, de plus l'adaptation de la *probabilité de mutation* P_m en fonction des conditions actuelles du réseau, où la P_m est mesurée comme suite :

$$P_m = \sqrt{1-(1-E)^2} \tag{3.22}$$

Où E est la valeur de l'entropie de cycle d'activation récent.

Lorsqu'un nœud reçoit la phéromone BSP, son agent local ajuste sa période du sommeil actuelle (*SP :Sleep Periode*) entre la valeur minimale (SP_{min}) et la maximale (SP_{max}) qui sont prédéfinies. Cet ajustement à la SP s'effectue en fonction de la concentration de deux phéromones MP et AP correspondante à ce nœud. Il est calculé comme suit :

$$SP_j = \begin{cases} SP_{\min} & \text{si}, \sum P_{t,j} \succ 0 \\ SP_{\max} & \text{sinon} \end{cases} \tag{3.23}$$

Où $\sum P_{t,j}$ est la somme de la concentration des phéromones MP et AP correspondantes au nœud j. Si cette somme est supérieure strictement à 0, alors la probabilité qu'un autre agent choisit le nœud j comme sa nouvelle destination, est très élevée, donc sa SP doit être dans la valeur minimale.

3.6.2.3 Evaluation de performances du MONet

L'auto-configuration et self-Healing

La figure 3.19 montre la configuration et réparation (Healing) de façon autonome du nœud (48), tout d'abord, au début de la simulation, tous les nœuds capteurs, y compris nœud (48), consomment certaine quantité de l'énergie pour émettre et/ou recevoir le message "*hello*" afin de découvrir ses nœuds voisins pour construire un RSCF.

Nous avons configuré le modèle qui génère la valeur de température au sien du nœud (48) selon trois cas de changements suivent : *fixe, pas à pas et oscillatoire*.

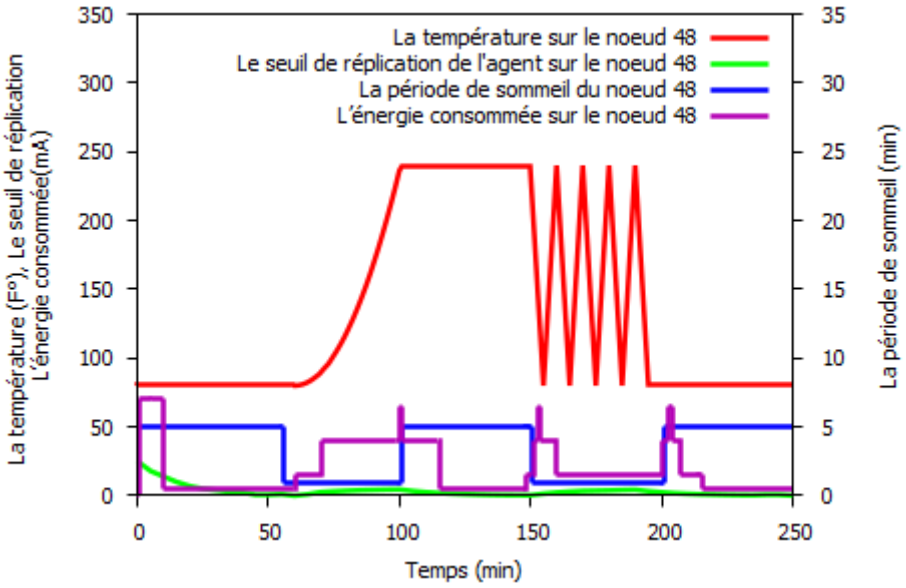

Figure 3.19 L'auto-configuration, l'auto-réparation et l'énergie consommée dans un nœud

Premier cas, la température est fixe : de 0 min à 60 min, la température au niveau de nœud 48, est fixe à 80°F, c'est-à-dire qu'il n'y a pas de changement significatif dans la lecture du nœud capteur (donc pas d'évènement) et aussi ses concentrations de phéromones MP et AP, sont égaux à 0, ce qui permet à l'agent d'ajuster la période de sommeil jusqu'à la valeur maximale (i.e., auto-configuration). Par conséquence, la valeur d'énergie consommée est devenue à bas niveau. MONet-R considère la stabilité de la température pendant cette longue durée comme des lectures fausses, ce qui exige d'aborder cette situation en réduisant pas à pas la valeur de seuil de réplication, ce qui permet à l'agent de détecter le plus petit changement de la température (i.e., salf-healing).

Deuxième cas, la température change pas à pas : de 60 min à 100min, la température augmente pas à pas, de 80° F jusqu'au 240° F. Où dans chaque cycle d'activation l'agent détecte un événement selon équation 3.21, donc le nœud devient en état active et ses concentrations de phéromones MP et AP, sont déférentes de 0, donc l'agent à ajuster la période de sommeil jusqu'à la valeur minimale. Par conséquence, la valeur d'énergie consommée est augmentée aussi, et le seuil de réplication reste presque fixé car l'agent acquise d'énergie de MONet-R qui le permet à fixer ce seuil. Et de 100 à 150 min le nœud est dans le premier cas.

Troisième cas, la température oscillante : de 150 à 200, la température est rapidement oscillée entre 80°F et 240°F, une fois que la température commence de se balancer, l'agent correspondant détecte un événement, car la différence est largement entre 240 et 80° F, ce qui augmente rapidement le seuil de réplication afin d'éviter la détection des faux événements car le MONet-R considère la répétition de ces données de cette façon est illogique. En fin après 200 min, la température est fixée à 80° F, le seuil de réplication de l'agent diminue brusquement, ce qui permet à l'agent de détecter un autre événement.

L'auto-organisation et l'auto-optimisation

Dans les simulations suivantes, nous avons configuré le modèle qui génère la valeur de température au sien de chaque nœud de la façon qui permet à l'agent de détecter un seul évènement, au début de chaque cycle d'activation. La figure 3.20 illustre les valeurs moyennes d'objectifs que tous les agents estiment dans chaque cycle de simulation, l'auto-optimisation est représentée par ces moyennes, et aussi illustre l'entropie produite dans chaque cycle, qui représente l'auto-organisation de réseau.

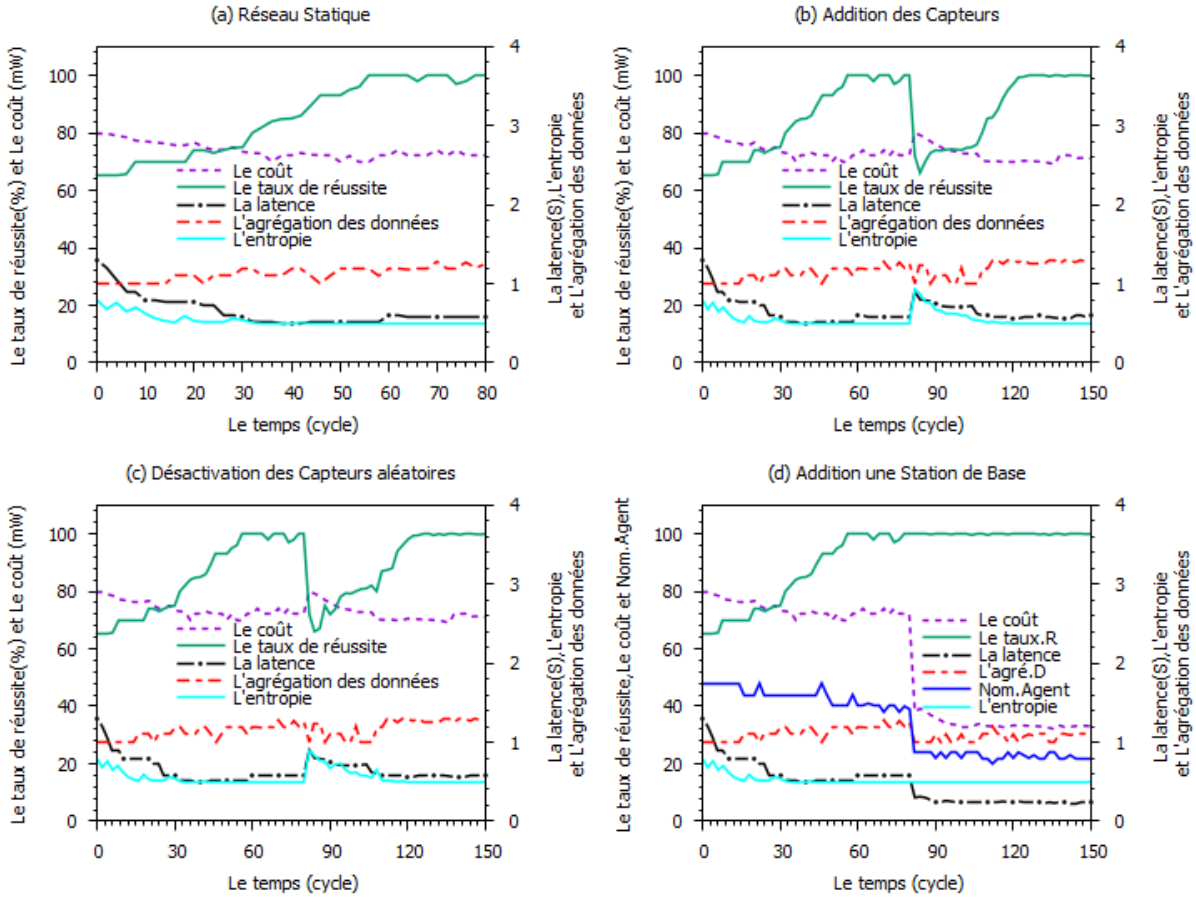

Figure 3.20 Les performances et l'entropie

La sous-figure 3.20 (a) illustre des résultats de simulation un réseau statique dans lequel les nœuds capteur et la SB ne tombent jamais en panne. Au début de la simulation les agents ont estimé des mauvaises performances (i.e., les valeurs d'objectifs) en raison de la randomisation de leurs structures génotypiques et le manque les phéromones sur le nœud, surtout l'AP et la MP. Ces performances se

sont améliorées pas à pas jusqu'à ce qu'elles soient convergées[2] vers le 60ième cycle (*tick*). Cette amélioration est due au fait que les agents s'adaptent au contexte du réseau, où ils reconfigurent leurs structure génotypes de façon autonome par l'utilisation des opérateurs génétiques (croisement et mutation). Et aussi au fil du temps, les phéromones deviennent disponibles sur tous les nœuds, ce qui leur permet de se déplacer avec une bonne orientation.

Les sous-figures 3.20 (b) à (c) illustrent des résultats de simulations d'un réseau dynamique, ce qui expliquent comment les agents réagissent à chaque changement dans le réseau, ceci se produit à la 80ième cycle. En général, une fois que le changement se produit, les performances se dégradent.

Dans la sous-figure 3.20 (b), une fois que les 14 nœuds sont ajoutés de manière aléatoire sur la zone observée, les performances se dégradent, surtout le taux de réussite, parce que les agents sur les nouveaux nœuds possèdent initialement des structures génotypiques aléatoires, ce qu'ils ne permettent pas de se déplacer efficacement vers la SB. De plus, les phéromones ne sont pas disponibles sur ces derniers lorsqu'ils sont déployés, ce qui influe sur les déplacements des autres agents. Cela peut être observé à l'augmentation du coût et de latence. Dans la sous-figure 3.20(c), il obtient les mêmes valeurs quand nous avons désactivé 14 nœuds qui sont sélectionnés de manière aléatoire. Les performances se dégradent, pace que certains agents essayent de se déplacer vers les nœuds désactivés qui sont référencés par les phéromones MP, ce qui augmente le nombre de déplacements échoués. Par conséquent, le coût et la latence sont augmentés. Néanmoins, une fois que les performances se dégradent en raison de ces changements dans le réseau, les agents s'adaptent progressivement à ces changements, ce qui permet d'améliorer leurs performances nouvellement convergées.

Dans la sous-figure 3.20 (d), après le déploiement d'une autre SB au coin nord-est de la zone observée, les performances s'améliorent sauf que l'agrégation de donnés qui se dégrade un peu. Pace que le réseau est divisé en deux sous réseaux où l'agent choisi de se déplacer à la SB la plus proche, donc la latence et le coût sont diminués presque à la moitié. Le nombre d'agents[3] qui arrivent à une SB est diminué aussi à la moitié ce qui signifie que la taille de chaque sous réseau est la moitié de la taille du réseau global, ce qui a conduit à minimiser l'agrégation de données.

En ce qui concerne l'auto-organisation qui est mesurée par la notion de l'entropie (cf., la figure 3.20), où après chaque changement dans le réseau, la valeur de l'entropie est augmentée, ce qui signifie que les agents sont en désordre et le réseau n'est pas bien organisé, puis que sa valeur diminue pas à pas, jusqu'à ce qu'elle atteigne le niveau le plus bas. Dans ce cas le réseau devient stable et bien organiser et les performances des agents sont convergées.

Ces résultats de simulations montrent que le MONet a la capacité de configurer, d'organiser et d'optimiser le RCSF de façon autonome qu'il permet aux agents d'autoconfigurer leurs structures génotypiques selon les conditions de chaque changement dynamique du réseau, ce qui permet d'optimiser leurs performances rapidement.

[2]La différence entre deux valeurs successives de la même performance, est presque nulle

[3] Les données agrégées dans un agent, sont comptées comme un seul agent, bien qu'elles soient captées par plusieurs agents.

3.7 Conclusion

Dans ce chapitre, nous avons présenté la conception et le développement un nouveau Framework, appelé *MONet*, qui est un Framework middleware dédié au réseau RCSF, pour déduire ses mécanismes nous nous sommes inspirés d'un système biologique (ex. la colonie d'abeilles) afin d'assurer l'autonomie du RCSF, en termes de l'organisation, la (re)configuration, l'optimisation et l'auto-réparation. Le MONet consiste en deux parties, une partie implémentée dans le nœud capteur, appelé *MONet-Runtime*, c'est une couche middleware qui fonctionne en haut du système d'exploitation (e.g, TinyOS) dans chaque nœud, et qui se base sur deux concepts principaux, *l'agent mobile* et *les phéromones*. L'agent a pour but de collecter et transporter les données du nœud vers la SB, alors que les phéromones sont des messages de communication indirecte entre les agents à travers l'environnement. L'autre partie, appelé *MONet-Server*, qui est un outil d'évaluation des agents, qui se base sur une fonction de *fitness* afin de sélectionner les agents d'élites. *MONet-Runtime* effectue périodiquement les opérateurs génétiques (*croisement* et *mutation*) sur les agents (locaux et élites) afin d'optimiser la prochaine génération d'agents dans le RCSF.

Par la suite, nous avons présenté les résultats d'évaluation de performances du MONet. Les résultats obtenus confirment que le MONet donne au RCSF une grande autonomie, ainsi qu'il permet aux agents d'évoluer et d'adapter leurs structures génotypiques aux conditions dynamiques du RCSF de manière auto-configurante, auto-optimisante et auto-réparatrice en recherchant les compromis optimaux entre leurs objectifs contradictoires. Dans le chapitre qui suit, nous allons introduire notre deuxième contribution, qui consiste en un mécanisme qui minimise la consommation énergétique dans MONet et les Framework similaires.

Chapitre 4

MONet : La Consommation énergétique

4.1 Introduction

Ce chapitre est dédié à nos contributions de recherche dans le domaine d'économie d'énergie dans MONet, et les framework bio-inspirés similaires, qui se basent sur les phéromones comme un moyen de communication. Sachant que la majorité de l'énergie d'un nœud est consommée pendant l'émission ou la réception des messages afin d'économiser l'énergie, il faut minimiser le nombre de messages transmis dans le réseau. En effet, le MONet utilise l'auto-réparation (self-Healing) pour minimiser les messages mais pas les phéromones. Pour cela, nous avons développé deux mécanismes dont l'objectif est de réduire autant que possible le nombre d'émission de phéromones sans influer négativement sur la communication entre les agents et aussi sans compromettre les performances du réseau.
Dans ce chapitre, nous allons tout d'abord présenter la problématique. Après, nous allons décrire nos solutions proposées et évaluer les résultats des simulations.

4.2 La problématique

Comme nous avons mentionné dans le chapitre 3, les agents dans le MONet se basent sur les phéromones pour déterminer les BSs les plus proches, ainsi que les chemins de migration vers ces BSs, comme la plupart des plateformes inspirées de la nature, à savoir BiSNET, El Niño et La Niña. Selon les études qui ont été effectuées dans [17], la consommation énergétique est réduite à environ 33% dans les nœuds quand les phéromones sont disponibles sur ceux-ci. Cependant, la propagation de phéromones par l'agent de manière aveugle gaspille beaucoup d'énergie. Comme le MOSOON dans BiSNET, par exemple, un agent doit émettre une phéromone en chaque saut d'un nœud à autre, même dans le cas où le saut a échoué. Il devrait donc émettre plusieurs phéromones pour transmettre un paquet du nœud source à la BS. Par exemple, dans la figure 4.1, pour que l'agent sur le nœud A transporte des données captées vers la SB via des nœuds intermédiaires (B, C et D), dans le meilleur cas, il doit diffuser au moins quatre phéromones MP et peut-être une phéromone AP. Pour réduire l'énergie consommée dans les nœuds capteurs, nous ne pouvons pas totalement abandonner les phéromones, mais nous pouvons minimiser leur propagation.
Un autre inconvénient dans le MONSOON, est que la structure génotypique de l'agent est limitée (les agents ont des structures très proches). Par conséquent, ils se déplacent vers la BS presque sur les mêmes chemins, ce qui peut conduire à une augmentation de la consommation d'énergie des nœuds qui constituent ces chemins, comme l'illustre la figure 4.2. Les nœuds qui se situent à proximité de la BS consomment beaucoup plus d'énergie que les autres nœuds. De plus, les nœuds qui se situent sur la diagonale de la zone observée (i.e., les plus courts chemins), consomment plus d'énergie que les nœuds qui se situent sur les bords.
Pour aborder ces défis, nous avons implémenté deux mécanismes dans MONet, dont le premier est une approche décentralisée qui se base sur l'égoïsme entre les agents et la deuxième est une approche centralisée qui se base sur un seuil, afin de minimiser la propagation des phéromones sans influer sur la coopération entre les agents. De plus, le MONet essaye d'assurer la diversification des structures

génotypiques d'agents afin d'équilibrer la consommation énergétique sur tous les nœuds dans le réseau. Dans ce qui suit, nous allons présenter ces approches.

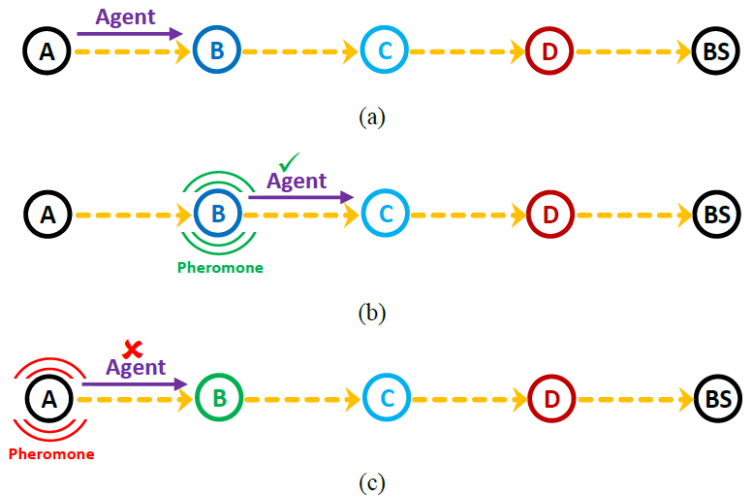

Figure 4.1 La dissémination des phéromones dans MONSOON **(a)** l'agent se déplace du nœud A vers le nœud B, le déplacement est **(b)** réussi, l'agent émet la phéromone MP **(c)** échoue, l'agent émet la phéromone AP.

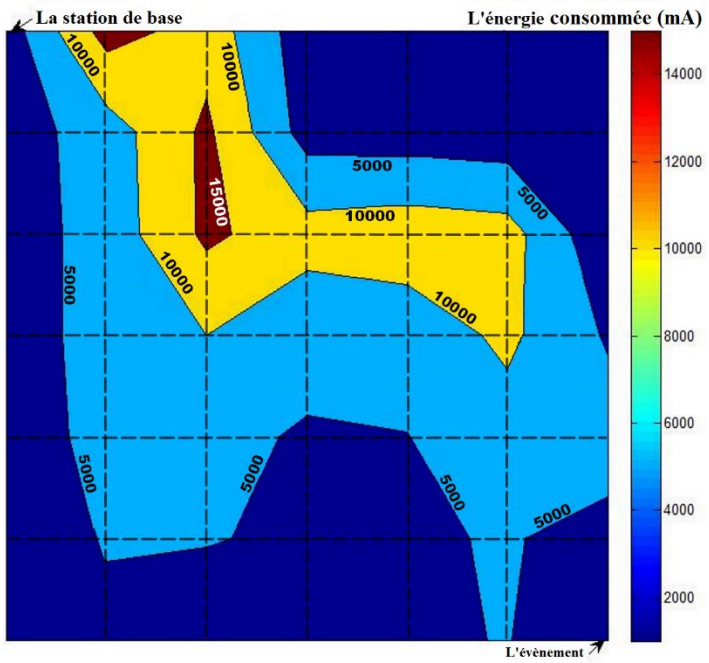

Figure 4.2 L'énergie consommée par chaque nœud capteur dans le réseau avec BiSNET/e[17]

4.3 L'approche décentralisée

L'idée de cette approche est de donner à chaque agent la capacité de décider de manière autonome d'émettre les phéromones ou non. Cette décision est prise par l'agent en fonction de son envie. S'il a une grosse envie de propagation les phéromones, alors il les émet et vice versa. Cette envie ne change pas avec le temps. Afin que nous réalisions cette idée, nous avons restructuré la structure génétique d'agent, où elle est devenue composée de quatre gènes au lieu de trois gènes (cf., la section 3.5.2),

comme illustré sur la figure 4.3. Le nouveau gène est appelé le gène égoïste (*selfish gene*) et a pour le rôle de déterminer la caractéristique de la propagation des phéromones de l'agent.

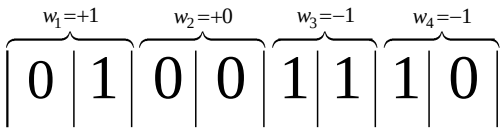

Figure 4.3 la nouvelle structure génotypique d'un agent dans MONet

L'implémentation de cette nouvelle structure est très simple dans le champ `Genotypic` de la structure de l'agent, où nous utilisons le deux bits faible poids, qui ont été réservés (cf., la section 3.6.1.2) pour indiquer la valeur du gène égoïste $w_4 \in \{-1,0,+1\}$. De cette manière, la taille de l'agent a été maintenue.

En ce qui concerne les comportements de l'agent, ils sont restés inchangés sauf le comportement de *Stigmergie* que nous avons adapté comme suit :

Chaque agent prend la décision d'émettre des phéromones ou non en se basant sur la probabilité d'égoïsme ($P_{selfish}$) et la valeur de w4 qui indique le type d'égoïsme de l'agent. Où si un agent est égoïste et sa valeur de $P_{salfish}$ est très élevée, alors il n'émet pas de phéromones afin de sauver son énergie, et par conséquent, sauver l'énergie du nœud. Selon la valeur du gène égoïste w_4, trois catégories d'agents sont identifiées :

- *Catégorie non-égoïsme* : les agents de cette catégorie, pouvant émettre les phéromones dans tous les cas. Ce type d'agent consomme pendant son déplacement de son nœud source à la SB une quantité d'énergie qui est définie par l'équation 4.1 suivante :

$$E_T = N \times E_{trans} + (N-1) \times E_{Ph} \tag{4.1}$$

  Où E_T représente la quantité totale d'énergie consommée par un agent lors de son déplacement vers la SB. E_{trans} représente la quantité d'énergie consommée en chaque saut. On suppose que sa valeur est fixée. E_{Ph} est la quantité d'énergie consommée pour émettre une seule phéromone et aussi est une valeur fixée et N est le nombre total de sauts, qui a été effectué par l'agent jusqu'à ce qu'il soit arrivé à la SB, y compris les sauts qui ont échoué.

- *Passif-égoïsme* : c'est l'égoïsme absolu, les agents de cette catégorie ne mettent jamais les phéromones, ce qui permet de réduire la consommation énergétique au niveau des nœuds qu'ils ont traversé pendant leur déplacement vers la SB, parce que la valeur de E_{Ph} dans l'équation 4.1 est nulle. Toutefois, à cause du manque de phéromones, les autres agents peuvent prendre de mauvaises décisions comme le montre l'équation 3.4. Par exemple, si un agent égoïste n'émet pas de phéromone AP lorsqu'il ne peut pas se déplacer à un nœud, les autres agents peuvent essayer de se déplacer à ce nœud, ce qui augmente la consommation énergétique à cause du grand nombre de déplacements échoués. Par conséquent, le taux de réussite et la latence sont dégradés.

– *Positif-égoïsme :* les agents de cette catégorie n'émettent les phéromones que dans le cas de la disparition des phéromones correspondantes. De cette manière l'agent minimise la possibilité d'émettre des phéromones, ce qui permet de réduire la consommation énergétique au niveau de certains nœuds qu'il a traversé, sans dégrader le taux de réussite et la latence. La quantité d'énergie consommée par cet agent est entre la première et la deuxième catégorie. La valeur de l'égoïsme d'un agent de cette catégorie est calculée selon les concentrations de phéromones correspondantes au nœud où il se situe, par l'équation 4.2 suivante :

$$\begin{cases} S_i = w_4 \times P_{selfish} \\ P_{selfish} = 1 - \left(P_i \middle/ P_{\max} \right) \end{cases} \quad (4.2)$$

Où P_i représente la concentration d'une phéromone, soit MP dans le cas de déplacement réussi, soit AP dans l'autre cas, P_{max} représente la valeur maximale correspondante à ce P_i sur le nœud *i*. $P_{selfish}$ la probabilité d'égoïsme. La valeur de poids dans l'équation 4.2 (w_4 = {-1, 0, +1}) détermine la catégorie de l'agent. Si un agent a une valeur positive pour w_4 (i.e., l'égoïsme positif), donc il calcule (S_i) pour son nœud local *i* selon chaque cas (MP ou AP). Si la valeur de (S_i) est très élevée, l'agent n'émet aucune phéromone et vice versa. Pour les autres valeurs de w_4 l'agent néglige la valeur de (S_i).

Vu que la nouvelle structure génotypique de l'agent, consiste en quatre gènes ($w1$, $w2$, $w3$, $w4$) ce qui permet de prolonger l'espace des solutions réalisables, de sorte qu'il devient contenant 81 au lieu de 27 solutions possibles (cf., la section 3.3.3), cette extension de l'espace des solutions, assure la diversification des agents qui se trouvent dans le réseau, ce qui permet de distribuer la consommation énergétique sur les nœuds.

4.3.1 Evaluation les performances

Afin d'évaluer l'efficacité énergétique du MONet et l'influence de l'égoïsme de l'agent sur les performances (l'auto-organisation, l'auto-configuration) et surtout la distribution de la consommation énergétique sur tous les nœuds dans le RCSF, des simulations ont été effectuées avec la même configuration de la section 3.6.2 et aussi les mêmes paramètres. La seule différence est la méthode de calcul de la période de sommeil, qui est calculée par l'équation 4.3 suivante :

$$SP_j = \begin{cases} \left(SP_{\max} \middle/ \sum_{t=1}^{3} P_{t,j} \right), & si \sum P_{t,j} \succ 0 \; et \; S_j \neq 0 \\ \left(SP_{\max} \right) & si \sum P_{t,j} = 0 \; et \; S_j \neq 0 \\ \left(SP_{\min} \right) & \text{sinon} \end{cases} \quad (4.3)$$

Où $\sum P_{t,j}$ est la somme de la concentration des phéromones MP et AP correspondantes au nœud *j*. S_j représente l'égoïsme de l'agent local de nœud j, Si la valeur de S_j est égale à 0, alors l'agent émet les

phéromones, donc la probabilité qu'un autre agent choisi le nœud *j* comme sa nouvelle destination, est très élevée, sa SP doit être dans la valeur minimale.

4.3.1.1 Impacts de l'agent égoïste sur un nœud

Afin de déterminer les impacts des agents égoïstes sur l'auto-configuration, l'auto-réparation et la consommation énergétique au niveau d'un nœud, nous avons répété la simulation qui est expliquée dans la section 3.6.2.3.1, mais nous avons changé la configuration du nœud (48) dans lequel son agent local est devenu toujours égoïste. Les résultats de cette simulation sont illustrés par la figure 4.4. On remarque que ces résultats sont presque similaires à ceux de la figure 3.19, sauf dans ce cas où l'énergie consommée est plus grande du fait que l'agent ajuste la période de sommeil à la valeur minimale (1 min) (cf., l'équation 4.3).

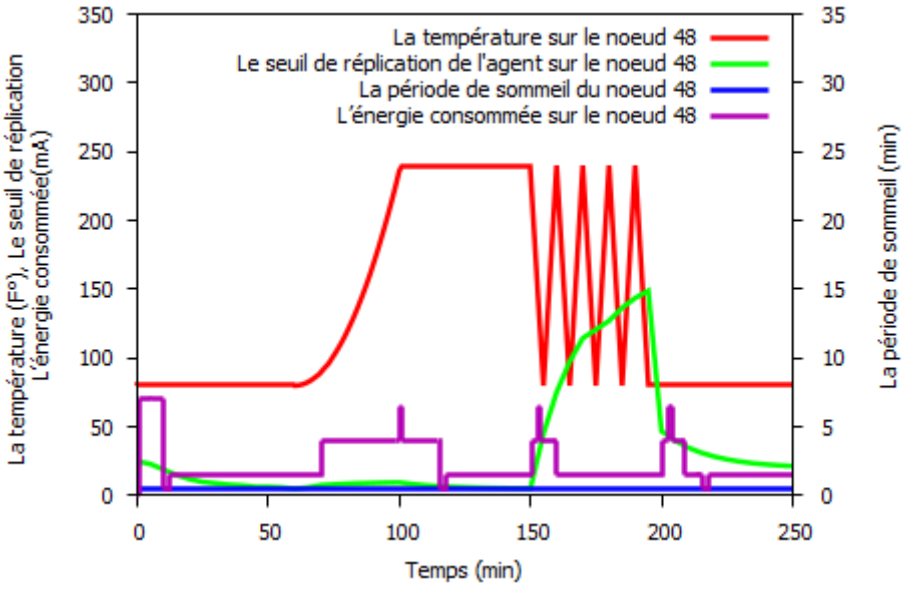

Figure 4.4 L'auto-configuration, l'auto-réparation et l'énergie consommée dans un nœud avec un agent égoïste

4.3.1.2 Impacts des agents égoïstes sur l'efficacité énergétique dans le réseau

Dans la section précédente, nous avons vu les impacts de l'agent égoïste sur la consommation énergétique au niveau d'un nœud. Maintenant, nous déterminons les impacts des agents égoïstes sur l'efficacité énergétique au niveau de tout le réseau. Les figures 4.5, 4.6 et 4.7 montrent la quantité d'énergie consommée par chaque nœud dans le réseau selon les trois configurations suivantes, l'égoïsme négatif (w_4=-1), l'égoïsme positif (w_4=+1) et l'hybride (w_4={-1,0,+1}), respectivement . La figure 4.5 illustre les résultats de simulation avec la configuration « *le passif-égoïsme* », c'est-à-dire que durant la simulation aucun agent n'émet une phéromone (tous les agents sont égoïstes). On remarque que certains nœuds, surtout les nœuds qui se situent à proximité de la BS et aussi qui se trouvent en centre du réseau, consomment beaucoup plus d'énergie que les autres (les intersections des lignes représentent les locations des nœuds), parce que quand les phéromones MP et AM ne sont pas disponibles au niveau des nœuds, les agents deviennent *Directional* (i.e., ils ne se déplacent que sur les plus courts chemins) (cf., la section 3.3.3). Cela augmente le risque qu'un réseau ne se décompose en petits clusters isolés et que certains nœuds ne peuvent plus communiquer avec la BS.

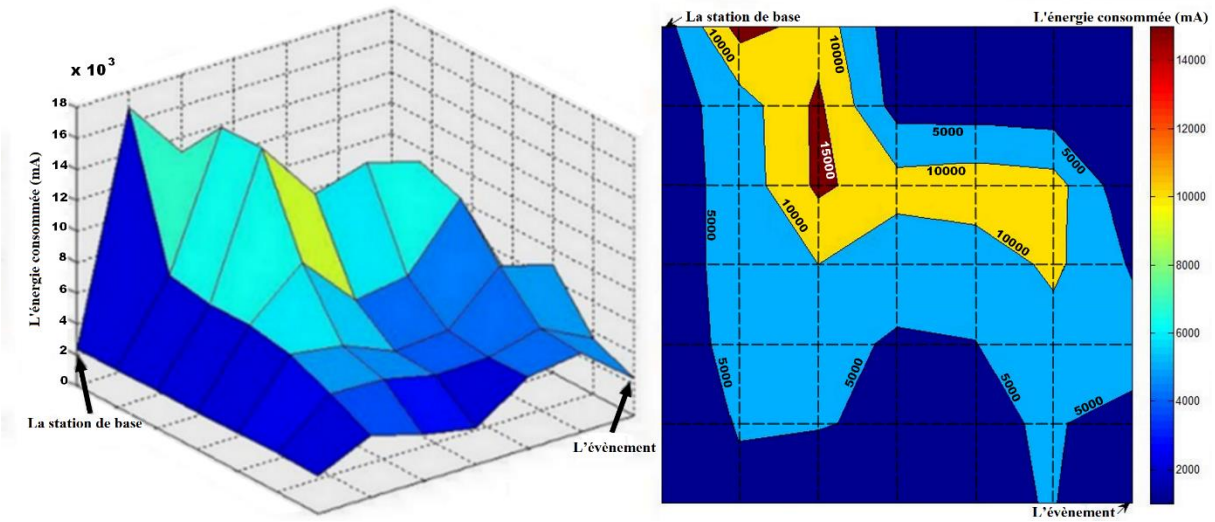

Figure 4.5 la consommation énergétique des nœuds avec l'égoïsme négatif

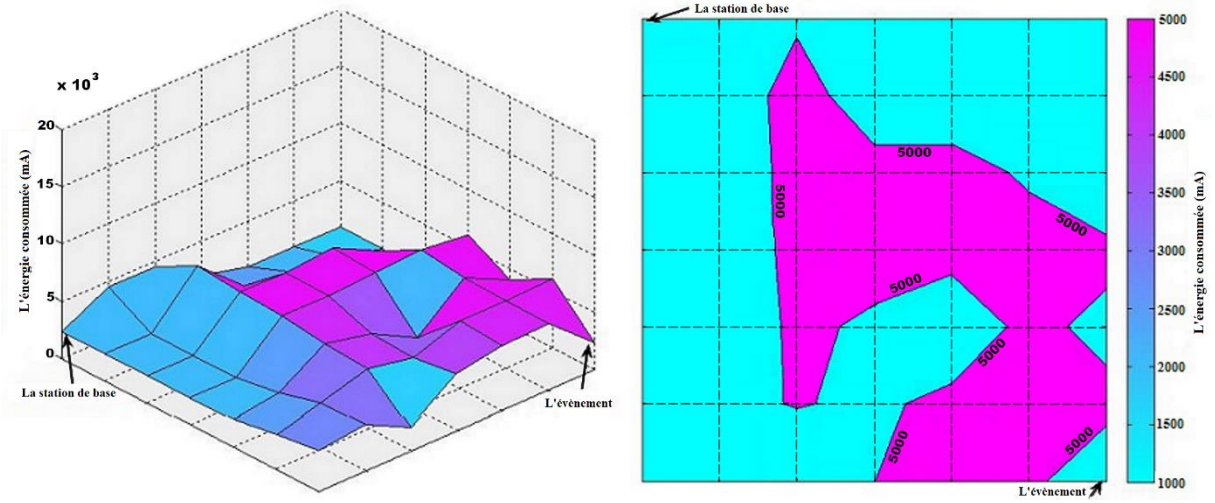

Figure 4.6 la consommation énergétique des nœuds avec l'égoïsme positif

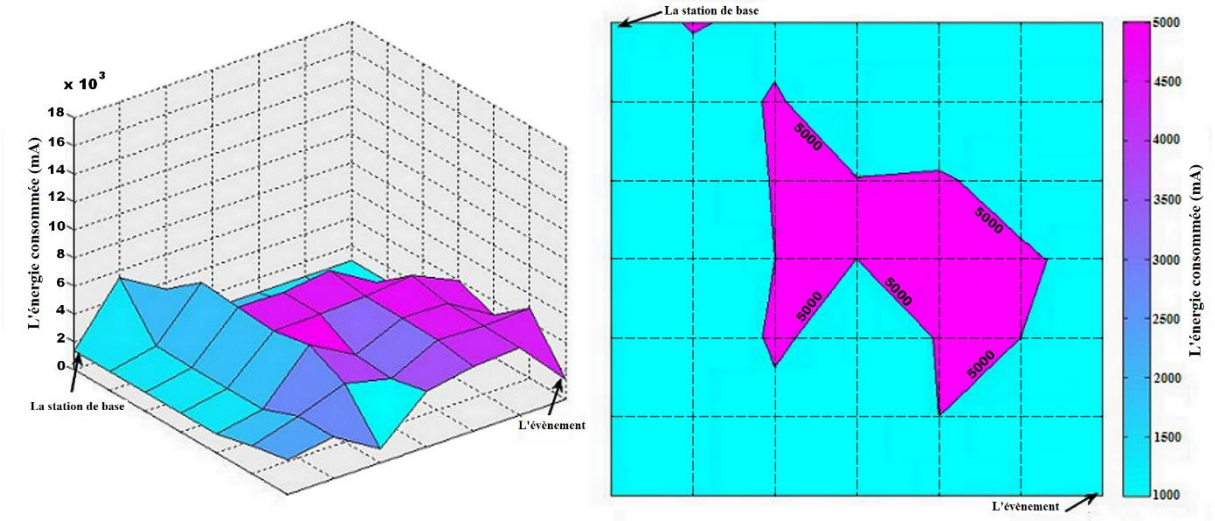

Figure 4.7 la consommation énergétique des nœuds avec l'égoïsme hybride

Par contre, avec la configuration « *le positif-égoïsme* » figure 4.6, où les agents n'émettent les phéromones que si nécessaire. Chaque nœud consomme une petite quantité d'énergie. En général, ces quantités sont similaires, car au début de la simulation tous les agents égoïstes dans le réseau émettent

les phéromones, et avec le temps la propagation des phéromones est réduite car les phéromones deviennent disponibles sur les nœuds, ce qui permet de réduire la consommation d'énergie sur les nœuds vers 50% et 20% de la consommation totale. Les résultats dans la configuration hybride sont meilleurs comme il est illustré sur la figure 3.7. Ces résultats montrent que MONet a la possibilité de réduire et de distribuer la consommation d'énergie sur les nœuds en utilisant le mécanisme d'égoïsme.

4.3.1.3 L'impact de l'agent égoïste sur la distribution de consommation énergétique

Pour évaluer l'impact de l'agent égoïste sur la consommation énergétique au niveau des nœuds capteurs du RCSF, dans chaque cas mentionné dans la section précédente. Où après la fin de chaque simulation (le temps de simulation est le même dans les trois configurations), nous avons regroupé tous les nœuds dans des ensembles selon la quantité d'énergie consommée par chaque nœud ; dans un ensemble, tous les nœuds qui consomment une quantité entre 0 et 2 Ah, sont regroupés dans un autre ensemble les nœuds qui consomment une quantité entre 2 Ah et 4 Ah, et ainsi de suite jusqu'à ce que nous regroupions tous les nœuds. La figure 4.8 illustre le nombre de nœuds dans chaque ensemble résultant pendant chaque configuration. Comme on constate sur cette figure, avec les configurations « égoïsme hybride » et « égoïsme positif », les nœuds sont regroupés dans trois ensembles contigus (le quatrième ensemble contient un petit nombre de nœuds, on peut le négliger), mais avec les configurations « égoïsme négatif », les nœuds sont regroupés dans dix ensembles. Ce qui signifie que quand les phéromones sont disponibles sur les nœuds, MONet peut équitablement distribuer la consommation énergétique sur tous les nœuds, ce qui permet d'éviter la rupture du fonctionnement réseau et de prolonger sa durée de vie.

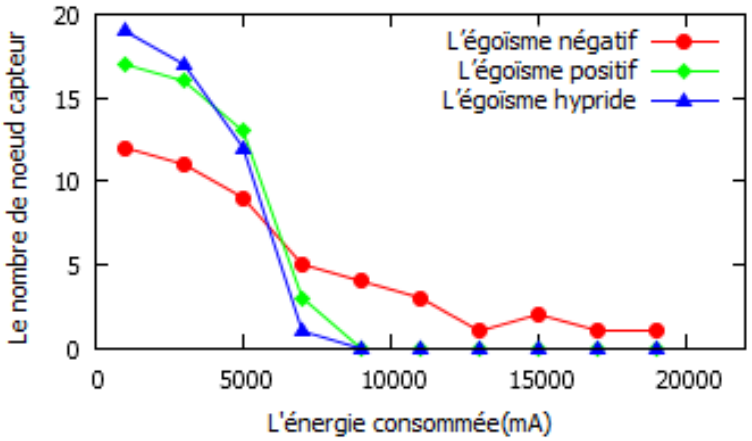

Figure 4.8 le regroupement des nœuds capteurs selon la consommation énergétique

4.3.1.4 L'impact de l'agent égoïste sur la durée de vie du réseau RSCF

Afin d'évaluer l'impact de l'agent égoïste sur la durée de vie du RCSF. Nous avons mesuré l'énergie résiduelle des nœuds capteurs toutes les 2 heures pendant toute la durée de la simulation des trois configurations décrites précédemment, afin de calculer le nombre total des nœuds morts (épuisement de toutes les réserves énergétiques), au fur et à mesure de l'avancement de la simulation. La figure 4.9, présente les résultats que nous avons obtenus. On remarque que dans la configuration « l'égoïsme négatif », pour 10 nœuds sur 49, les réserves énergétiques sont épuisées totalement pendant les 2 premières heures, et pendant la quatrième heure, tous les nœuds sont morts. Dans les autres

configurations, les capacités énergétiques de deux-tiers des nœuds sont épuisées pendant la sixième heure et tous les nœuds sont morts après la septième heure. Donc l'utilisation de l'égoïsme hybride et positif contribue à prolonger la durée de vie du réseau.

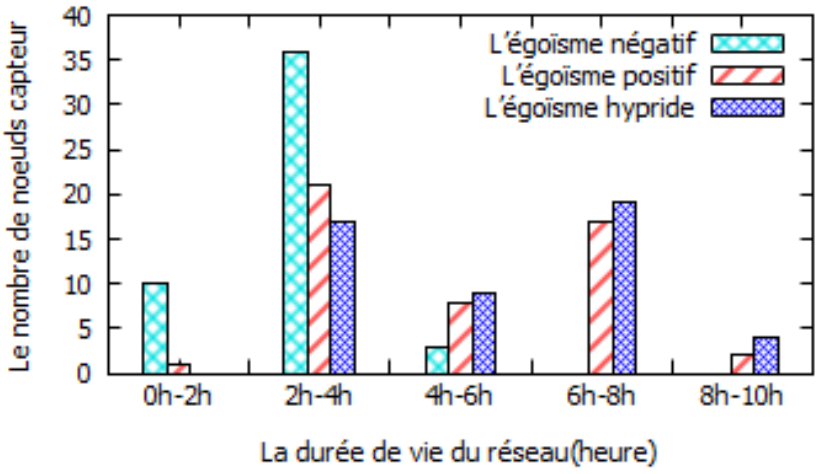

Figure 4.9 Histogramme de la durée de vie du réseau

4.3.1.5 L'émission de phéromones MONet vs MONSOON

Dans cette section, nous avons effectué des simulations pour faire une étude comparative en termes de réduction de l'émission de phéromones et d'économie d'énergie, entre notre proposition MONet et MONSOON. Les résultats que nous avons obtenus sont représentés dans figure 4.10 qui à son tour illustre les moyennes des émissions de phéromones (*APE : the average pheromone emissions*), qui sont produites par les agents pendant un intervalle de temps équivalant à un cycle d'activation du nœud. Nous avons gardé la même configuration utilisée dans la simulation pour la figure 4.8.

Au début de ces simulations, cas de MONSOON, les agents ne peuvent pas se déplacer efficacement vers la SB à cause du manque de phéromones sur chaque nœud du RCSF. Cela peut augmenter le nombre de déplacements échoués qu'un agent effectue. Par conséquent, l'émission de phéromone AP est devenue élevée, ce qui produit une grande valeur d'APE, c'est plus de 12 phéromones/cycle. Cependant, au fil de temps les phéromones deviennent disponibles sur les nœuds (i.e., chaque nœud a été référencé par au moins une phéromone, soit MP ou soit AP), ce qui permet aux agents de diminuer graduellement leurs émissions de phéromones, surtout (AP) et les valeurs d'APE convergent vers $50^{ième}$ cycle et sa valeur devient 9 phéromones/cycle. Ceci est identique avec MONet, mais les valeurs d'APE produites dans ce cas sont inférieures d'environ 30% que celles produites dans le cas de MONSOON, parce que les agents passif-égoïsmes n'émettent jamais des phéromones, aussi les agents positif-égoïsmes n'émettent des phéromones qu'en cas de besoin, ce qui retarde la convergence des valeurs d'APE autour de dix cycles.

Afin d'évaluer l'influence de cette réduction des émissions de phéromones sur les performances des agents, nous avons opté comme une métrique comparative des valeurs moyennes de latence que les agents produisent dans chaque configuration de la figure 4.10. Au début de la simulation, nous remarquons que dans les deux configurations (MONet ou MONSOON), les valeurs moyennes de latence sont élevées car la structure génotypique de chaque agent est initialisée aléatoirement sur son

nœud local, aussi les phéromones ne sont pas disponibles sur tous les nœuds lorsqu'ils sont déployés pour la première fois. Pour cela, les agents ne peuvent pas prendre la décision de déplacement appropriée vers ces nœuds. Mais dans MONSOON, les valeurs moyennes de latence sont améliorées plus rapidement que dans MONet parce que dans le MONSOON les agents émettent beaucoup plus de phéromones que dans MONet. Cela accélère la disponibilité des phéromones sur les nœuds (MONSOON) et permet aux agents d'adapter leurs structures génotypiques pour qu'ils puissent se déplacer vers la SB à travers le plus court chemin. Les nœuds MONet n'ont pas besoin de faire cela (i.e. les agents suivent plusieurs chemins de migration et pas seulement le plus court). Toutefois, les valeurs moyennes de latence deviendront égales après 60^{ième} cycle dans les deux configurations.

Enfin, nous pouvons conclure que la réduction des émissions de phéromones offre les avantages suivants

- économise l'énergie des nœuds sans affecter la coopération entre les agents et les paramètres de qualité de service (par exemple, la latence).
- varie les chemins de migration des agents à la BS, et distribue la consommation d'énergie sur plus de nœuds, évitant ainsi que le réseau soit séparé en des clusters isolés.

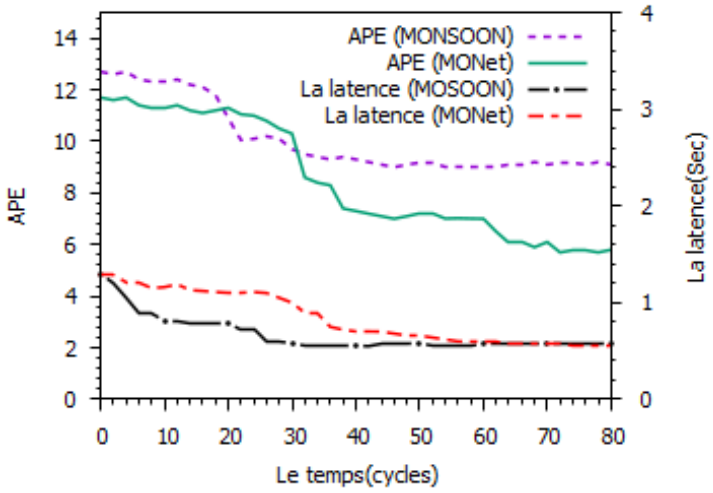

Figure 4.10 MONet vs MONSOON : APE et la latence

4.4 L'approche centralisée

L'idée de cette approche est de contrôler le niveau de coopération entre les agents à travers un mécanisme central qui se base sur un seuil, appelé le seuil d'émission de phéromones (*the pheromone emission threshold*), pour réduire le nombre de phéromones qu'un agent émet sur chaque nœud, et par conséquent, l'économie de la consommation énergétique de nœuds du réseau, sans influer les performances comme la fiabilité des données. Ce seuil est automatiquement ajusté par le MONet-serveur en fonction des conditions dynamiques du réseau, ou prend des valeurs prédéterminées par l'utilisateur.

Afin de réaliser cette idée dans MONet, nous avons effectué des optimisations sur les comportements de l'agent et exactement dans le comportement de *Stigmergie*, où nous avons associé une contrainte à chaque type de phéromone, en se basant sur le seuil d'émission de phéromones, de sorte que chaque

agent ne peut émettre aucune phéromone avant que la concentration des phéromones correspondant à son nœud local devienne inférieure au seuil d'émission de phéromone (T_P). Les autres comportements sont restés inchangés.

4.4.1 Le seuil d'émission de phéromone et l'entropie

En ce qui concerne le seuil d'émission de phéromone, le MONet-serveur l'ajuste en fonction de l'entropie E (cf., la section 3.6.2.2). Cette dernière indique la similarité des objectifs des différents agents arrivant à la SB, lorsque celle-ci est faible, les objectifs sont très similaires. En d'autres termes, les agents se sont adaptés aux conditions actuelles du réseau, c'est-à-dire que ces agents peuvent se déplacer efficacement vers la SB grâce à la disponibilité suffisante des phéromones sur tous les nœuds. Dans ce cas, le MONet-serveur minimise la valeur de seuil (T_p), afin de réduire le nombre d'émissions de phéromones, et vice-versa. La valeur de l'entropie est mesurée par l'équation 3.19, le MONet-serveur utilise le champ `Opt` de phéromone SBP pour envoyer la valeur récente de l'entropie vers les agents dans le réseau.

Lorsque l'agent reçoit la nouvelle valeur de l'entropie, il ajuste la valeur de T_p de son nœud local selon l'équation 4.4 suivante :

$$T_P = C_{\max} \times \sqrt{1-\left(1-E/E_{\max}\right)^2} \tag{4.4}$$

T_P est la nouvelle valeur du seuil d'émission de phéromone. C_{max} indique la concentration maximale de phéromone. E (E_{max}) est la valeur actuelle (resp., maximale) d'entropie E.

4.4.2 Evaluation les performances

Dans cette section, nous avons gardé les mêmes paramètres et les mêmes configurations cités dans la section 4.3.2. Concernant le seuil d'émission des phéromones (T_P), deux scénarios sont possibles. Dans le premier, la valeur de T_P est fixée aux valeurs suivantes 50%, 75% et 90%. La valeur du deuxième scénario est calculée par l'équation 4.4.

4.4.2.1 Impacts du seuil fixe sur les émissions de phéromones

La figure 4.11 illustre les moyennes d'émission de phéromones (*APE : the average pheromone emissions*) que le MONet produit durant chaque cycle de simulation (*the tick*) pour le réseau RCSF, selon les quatre scénarios : avec la valeur de T_P est fixée à 100%, ceci est équivalent au cas en BiSNET/e, 90%, 75% et 50% de la valeur maximale de la concentration de phéromone.

Comme dans la section 4.3.2.5, au début des simulations, et dans tous les cas, les agents ne peuvent pas se déplacer efficacement vers la SB en raison du manque ou l'existence insuffisante de phéromones sur chaque nœud du RCSF. Cela peut augmenter le nombre de déplacements échoués qu'un agent effectue. Par conséquent, l'émission de phéromone AP, est devenue élevée, ce qui produit une grande valeur d'APE, c'est plus de 11 phéromones/cycle. Cependant, au fil du temps les phéromones deviennent disponibles sur les nœuds, ce qui permet aux agents de diminuer graduellement leurs émissions de phéromones, surtout (AP) et les valeurs d'APE convergent vers 50^{ième} cycle et sa valeur devient 9 phéromones/cycle pour la valeur T_P être égale au 100% ou 90% et 6

phéromones/cycle pour T_P être égale au 75% ou 50%. Nous pouvons donc conclure que la minimisation de la valeur seuil (T_P) conduit à réduire les émissions des phéromones. Toutefois, l'exagération à cela, peut entraîner des fluctuations (chute) dans les performances des agents.

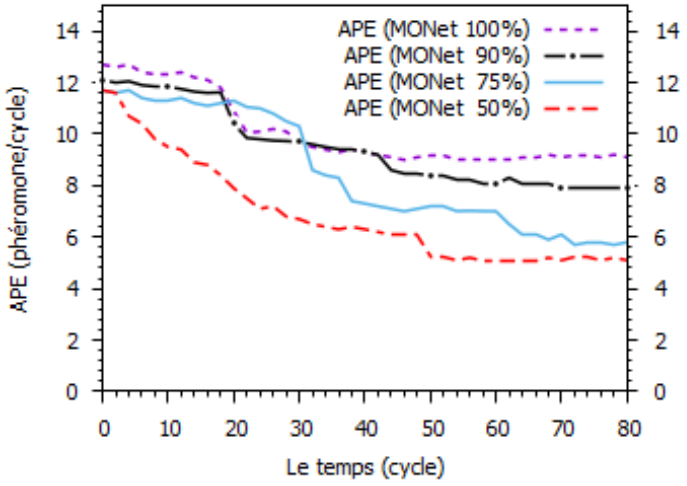

Figure 4.11 MONet : APE avec la valeur de TP est fixé.

4.4.2.2 Impacts du seuil ajusté sur les émissions de phéromones

La figure 4.12 illustre les valeurs de seuil d'émission des phéromones (T_P) et les moyennes d'émission des phéromones (APE) dans chaque cycle ; où la valeur de T_P est ajustée par le MONet-serveur selon l'équation 4.4. On remarque que la valeur de l'APE se détériore rapidement au début de la simulation. Elle continue à se diminuer graduellement jusqu'à ce qu'elle converge vers 50^{ième} cycle. L'APE diminue quand la valeur du T_P diminue, où la valeur de ce dernier se raccorde à la situation du RCSF. Si le réseau est instable comme au début, la valeur de T_P est élevée dès que les agents s'adaptent aux conditions du réseau (i.e., l'entropie devient nulle), la valeur de T_P est devenue basse.

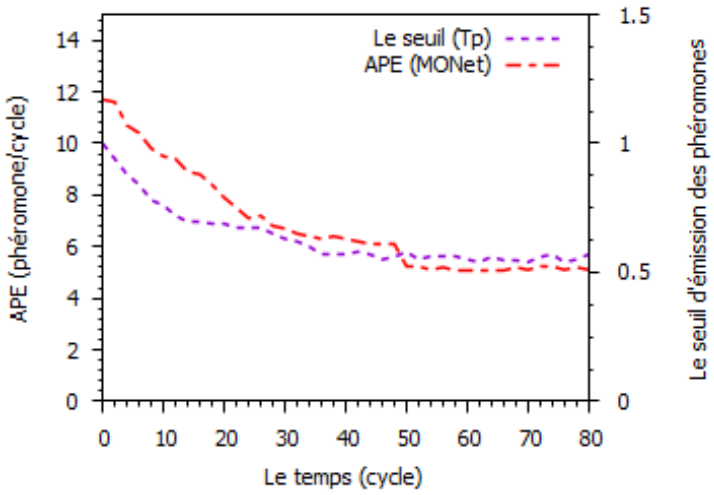

Figure 4.12 MONet : APE avec la valeur de T_P ajustée.

4.4.2.3 Les performances des agents

Afin de comparer MONet avec un mécanisme d'adaptation bio-inspiré, appelé El Nina [159], qui est conçu pour réduire le nombre de transmissions de données en utilisant l'égoïsme des agents, et pas les émissions de phéromones MP et AP pour réduire la consommation d'énergie.

La figure 4.13 illustre les valeurs moyennes d'objectifs que les agents produisent avec MONet et El Nina. Dans les deux cas, toutes les valeurs d'objectifs s'améliorent et convergent vers la 50^{ième} cycle. Dans MONet, la sous-figure 4.13 (a) comparé à El Nina sous-figure 4.13 (b), les valeurs d'objectifs s'améliorent un peu lentement et fluctuent généralement moins. Par exemple, dans la sous-figure 4.13 (a), il faut 50 cycles (*ticks*)de simulation pour que le taux de réussite atteigne 30% alors qu'il ne prend que 40 cycles sur la sous-figure 4.13 (b), mais le coût moyen est d'environ 2,0 transmissions par saut (trans/hop) alors qu'il est inférieur que 1,80 trans/hop dans la sous-figure 4.13(a). Cela signifie que MONet peut réduire le nombre d'émissions de phéromones qu'El Nina.

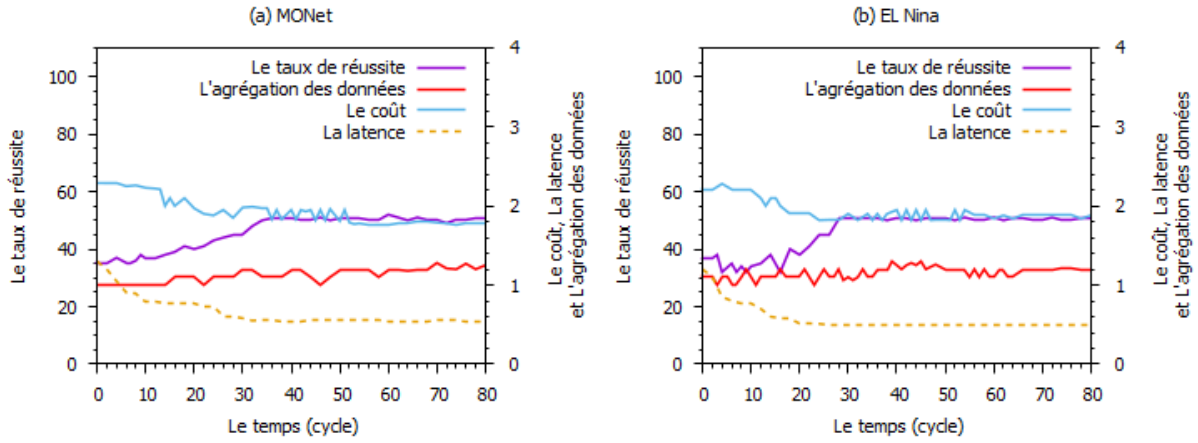

Figure 4.13 Les performances avec (a) MONet (b) EL Nina

4.5 Conclusion

Prenant l'efficacité énergétique comme objectif principal, nous avons proposé deux mécanismes dédiés à la gestion efficace d'énergie. Notre idée de base est de minimiser l'émission de phéromones le plutôt possible sans influer la coopération entre les agents et sans compromettre les performances de réseau. Le premier est un mécanisme décentralisé qui se base sur la nouvelle caractéristique de l'agent, appelée l'égoïsme, qui détermine la nature de l'agent en termes de sa capacité à envoyer des phéromones (i.e., la coopération) pendant son déplacement ou non. Il y a trois catégories : agent passif-égoïste (non coopératif), agent positif-égoïste (coopératif si nécessaire) et agent non-égoïste (coopératif). Le but principal de ce mécanisme est de trouver le meilleur compromis entre ces trois catégories d'agents dans le réseau, afin de minimiser l'émission de phéromones. Le deuxième est un mécanisme centralisé, qui se base sur un seuil calculé par le MONet-serveur, et chaque agent n'envoie pas les phéromones seulement si les concentrations de ces phéromones dans son nœud local, sont inférieures à ce seuil. Les deux mécanismes diminuent le nombre d'émission de phéromones d'environ un tiers, et distribuent aussi la consommation énergétique de façon équitable sur la plupart des nœuds du réseau, ce qui permet de prolonger sa durée de vie. Dans le chapitre qui suit, nous allons introduire notre troisième contribution, qui consiste en une extension de MONet pour exploiter les capacités disponibles sur les réseaux de capteurs sans fil hétérogènes.

Chapitre 5

BISSA : Extension de MONet pour les RCSF hétérogènes

5.1 Introduction

Plusieurs études et approches ont conçu des plateformes (framework) middleware pour les réseaux de capteurs sans fil (RCSF) au même titre que notre proposition MONet (cf., le chapitre 3). Néanmoins, la plupart d'entre eux ont supposé une homogénéité au niveau des nœuds. En d'autres termes, ils supposent que les caractéristiques de nœuds constituants le réseau sont similaires, ce qui est en contradiction avec la réalité de ces réseaux. En effet, au fil de temps le réseau RCSF devient un réseau hétérogène à cause de la différence dans la quantité d'énergie résiduelle dans chaque nœud ce qui influe sur leurs puissances de transmission, en d'autres termes le RCSF était à l'origine hétérogène causé par l'existence de différents types de nœuds avec des caractéristiques différentes, comme par exemple les réseaux de capteurs et actionneurs (WSAN) dont les nœuds actionneurs disposant généralement d'une source d'énergie abondante. L'existence de cette richesse en ressources peut considérablement dégrader les performances du réseau avec MONet qui ne considère pas ce type de ressources, c'est-à-dire qu'il ne peut pas donc profiter de cette richesse en énergie : par conséquent les capacités de calcul et de stockage ainsi que la puissance de transmission au niveau des actionneurs sont plus importantes.

De notre point de vue, la richesse en énergie et la puissance de transmission au niveau des actionneurs doivent être considérés car ils peuvent être efficaces dans l'assurance de la connectivité du réseau et de la prolongation sa durée de vie. Par ailleurs, ils ouvrent de nouvelles opportunités pour améliorer les performances du réseau. En effet, l'utilisation de ces portées de communication, qui sont généralement des portées à longue distance, nous permet de réduire le nombre de sauts pour atteindre la destination finale (i.e., la SB), d'augmenter ainsi le taux de livraison et d'exploiter de plus la diversité des portées de communication et des routes dans le réseau.

Dans cette perspective et afin de tirer profit de ces ressources, nous proposons une extension de MONet nommée BISSA pour la collecte des données ou/et la détection d'événements dédié aux WSAN. Cette extension tire profit des portées de communications, des capacités d'énergie et de stockage, permet d'assurer un taux de réussite élevé tout en réduisant significativement le nombre de messages dupliqués ainsi que d'assurer une latence et un coût faible tout en réduisant le nombre de sauts de bout-en-bout.

Dans ce chapitre, nous commençons par présenter notre proposition BISSA. Nous évaluons par la suite les performances de notre proposition.

5.2 BISSA : un Framework bio-inspiré pour l'auto-adaptation des réseaux de capteurs et d'actionneurs sans fil

Vu que BISSA (*A **B**iologically-**I**nspired framework for **S**elf-adaptation wireless **S**ensor and **A**ctuator network*) est une extension du MONet pour les réseaux de capteurs sans fil hétérogènes, en conséquence elle doit se composer de trois parties comme MONet (cf., la section 3.2.4), un système multi-agent, BISSA-Runtime (BISSA-R) fonctionne au-dessus du système d'exploitation dans chaque

nœud capteur ou actionneur, et BISSA-serveur (BISSA-S) fonctionne comme un serveur central. La seule différence entre eux c'est que dans le BISSA, il y a deux types de BISSA-R, l'un dédié au nœuds à faibles ressources comme les capteurs, nommé BISSA-Rs et l'autre dédié au nœud riches en ressources comme actionneurs, nommé BISSA-Ra, mais les deux types partagent des services et des comportements communs (cf., la figure 5.1). Cette nouvelle architecture de BISSA-R exige deux types d'agents, un agent capteur (SA *: sensor agent*) pour les nœuds capteurs et un agent actionneur (AA : *actuator agent*) pour les nœuds actionneurs. La relation entre les deux types d'agents est l'agent «SA» peut devenir un agent «AA» par l'acquisition d'un nouveau comportement et des actions et vice-versa (cf., la section ci-dessous).

Lorsqu'un agent « SA » détecte un évènement ou collecte des données sur le nœud capteur local (i.e., BISSA-Rs), les transporte saut pas saut à travers les nœuds capteur et/ou actionneur intermédiaire jusqu'à une station de base. Lors de ce déplacement, l'agent essaye d'arriver au nœud actionneur le plus proche (i.e., BISSA-Ra), quand cela arrive, il change son range de «SA» à «AA» afin d'augmenter son saut au maximum (dans notre cas, la portée de transmission de nœud actionneur est plus grande par rapport aux capteurs) et aussi acquérir des actions appropriée. Chaque agent, soit «SA», soit «AA» contient un ensemble de données appelées la structure génotypique et un code (programme) qui est interprété par le BISSA-R. Cette structure génotypique gouverne les comportements de l'agent, par exemple, comment invoquer les comportements de l'agent tels que l'émission de phéromones, la réplication et le déplacement (cf., la section suivante). Cela est amélioré par BISSA-S qui évalue l'agent en fonction de ses objectifs (performance) comme coût et latence.

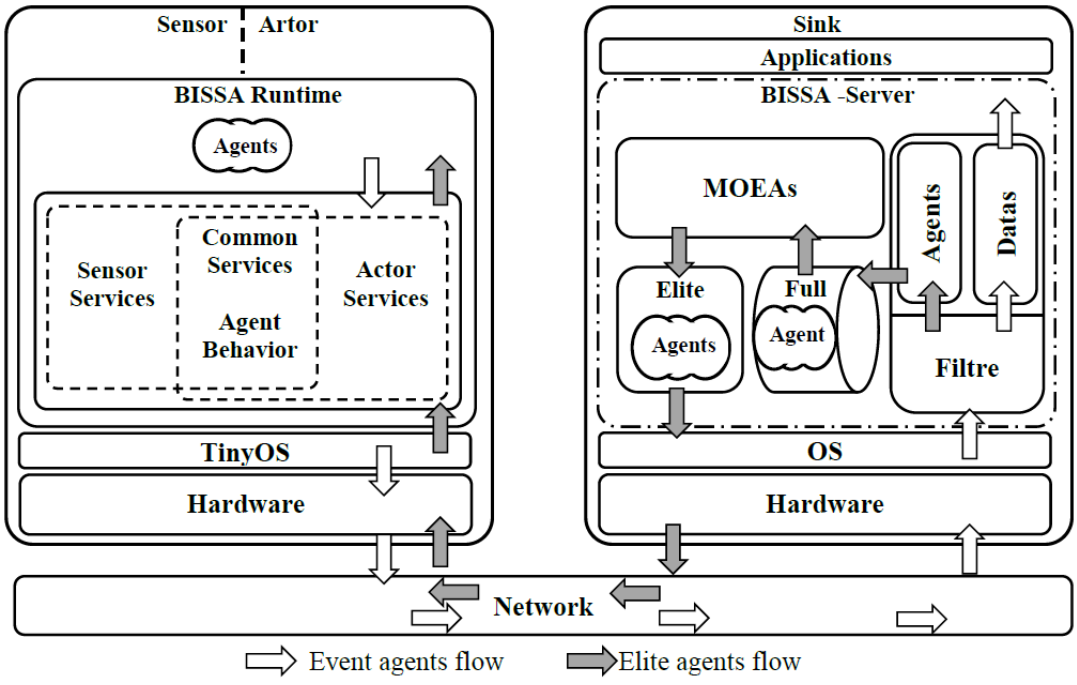

Figure 5.1 l'architecture de BISSA.

5.2.1 L'agent dans BISSA

Après le déploiement dans le WSAN, chaque agent « SA » se trouve seulement dans un nœud capteur (pas dans un nœud actionneur), les structures génotypiques de ces agents sont générées aléatoirement. Dans BISSA, chaque agent « SA » consiste des attributs, un corps et des comportements et chaque agent « AA » est une extension de l'agent « SA » en termes d'actions supplémentaires. Cette architecture est expliquée dans la section 3.3.1, mais avec les suppléments suivants.

- **Les attributs** : nous avons ajouté le rang d'agent comme un nouvel attribut, qui détermine le type de l'agent « SA » ou « AA ».
- **Le Corps** : nous avons ajouté un nouvel ensemble de fonctions appelée les *actions*, qui concerne les agents « AA ». Ces actions sont définies par l'utilisateur selon le cas d'utilisation, par exemple, une action qui permet d'allumer et d'éteindre les sirènes en cas d'incendie.
- **Les Comportements** : nous avons ajouté un nouveau comportement pour les agents « SA » et « AA », appelé « *changement de rang* ». Chaque agent invoque ses comportements à l'ordre suivant dans chaque cycle d'activation (cf., la figure 5.2).

Dans cette partie nous avons gardé les mêmes définitions des comportements sauf pour les comportements suivants que nous redéfinissons comme suit :

- **Swarming:** Chaque agent peut se joindre (ou se fusionner) aux autres agents sur un nœud intermédiaire (capteur ou actionneur) à condition que ces agents se déplacent vers la même SB. Plusieurs agents deviennent un seul agent. Sur chaque nœud actionneur, un agent « AA » peut attendre une période du temps (At_w) pour l'arrivée des autres agents au nœud actionneur actuel, il peut se joindre avec les agents « AA » ou « SA ». Aussi sur chaque nœud capteur, un agent « SA » fait la même chose, mais il peut attendre une période du temps (St_w, avec $St_w<At_w$), à l'exception qu'il ne peut pas se joindre avec les agents « SA ». Dans les deux cas, si plusieurs agents se sont rencontrés sur le même nœud et ils s'achemineront vers la même SB ou le même nœud actionneur, alors ils se fusionnent en un seul agent qui agrège toutes leurs données captées, en tenant compte de la taille maximale du paquet.
- **Stigmergie** : nous avons défini deux nouveaux types de phéromones concernant le nœud actionneur. Donc cinq types de phéromones doivent être disponibles sur chaque nœud local, à savoir : les phéromones de station de base (BSP), de nœud actionneur (ACP), d'isolement (IP), de migration (MP) et d'alerte (AP). Chaque phéromone a sa propre concentration, qui diminue de moitié à chaque cycle d'activation, et disparaît quand sa concentration atteint zéro.

  Les définitions de nouvelles phéromones sont ci-dessous et les autres restent inchangées (cf., la section 3.3.2).

- **Phéromone de nœud actionneur (ACP)** : Cette phéromone similaire à la phéromone BSP de la station de base. Chaque nœud actionneur diffuse périodiquement une information centrale appelée la phéromone ACP vers les nœuds du réseau afin d'attirer fortement les agents. Ces phéromones jouent le même rôle de phéromones BSP.
- **Phéromone d'isolement (IP)** : Cette phéromone similaire à la phéromone AP, mais l'IP aide les agents dans l'évitement des nœuds actionneurs temporairement isolés. Lorsqu'un nœud actionneur ne peut pas communiquer avec un autre nœud actionneur, il doit donc diffuser la phéromone IP vers les nœuds capteurs afin d'avertir les agents afin qu'ils évitent de s'y déplacer. Chaque phéromone IP indiquant l'ID du nœud actionneur isolé et sa propre concentration.

– **Le déplacement** : en utilisant les phéromones BSP et ACP disponibles sur les nœuds capteurs, les agents peuvent trouver les SBs ou les nœuds actionneurs qui se situent à proximités et se déplacent vers l'un d'eux en grimpant sur un gradient de concentration de phéromones correspondantes. Au cas où l'agent détecte une phéromone IP, il lui faut donc éviter de se déplacer vers le nœud actionneur qui est indiqué par cette phéromone. En effet, chaque agent lors de son déplacement dans le RCSF vers la SB or le nœud actionneur, il doit choisir un nœud intermédiaire comme sa prochaine destination en se basant sur les concentrations du cinq phéromones citées auparavant.

Tout d'abord, chaque agent « SA » sur un nœud capteur, examine l'équation 5.1, afin qu'il détermine, s'il lui faut se déplacer à la SB ou à un nœud actionneur, dans ce dernier cas, c'est lequel ?

$$Max\left[\left\{\rho_{BSP}^{j}, \rho_{ACP}^{i,j} - \rho_{IP}^{i,j} \text{ où } i \in E_a\right\}\right] \tag{5.1}$$

L'agent sur le nœud capteur « *j* » fait une comparaison entre la valeur de la concentration de phéromone BSP correspondante au nœud « *j* » (ρ_{BSP}^{j}) et toutes les valeurs qui résultent de la différence entre la valeur da la concentration de phéromone ACP ($\rho_{ACP}^{i,j}$) et IP ($\rho_{IP}^{i,j}$) correspondantes à chaque nœud actionneur « *i* » qui appartient à l'ensemble de nœuds actionneur (E_a) qui se situent à la proximité de nœud « *j* » (i.e., leurs propres phéromones sont disponibles sur le nœud « *j* »). L'agent choisit le nœud actionneur (y compris la SB) qui produit la valeur la plus élevée.

Deuxièmement, chaque agent « SA » (resp., AA) examine l'équation 5.2 pour déterminer le prochain nœud capteur (resp., nœud actionneur) vers lequel il se déplacera.

$$Sw_j = \sum_{t=k}^{2+k} w_t \frac{\rho_t^j - \rho_t^{\min}}{\rho_t^{\max} - \rho_t^{\min}} \tag{5.2}$$

Un agent « SA » (resp., « AA ») calcule cette somme (Sw_j) pour chaque nœud voisin « j » (resp., seulement le nœud actionneur) et se déplace vers le nœud pour lequel la somme est la plus élevée. Avec *t* désigne le type de phéromone; de ρ_1^j à ρ_4^j représentent les concentrations de phéromone ρ_{BSP}^j, ρ_{MP}^j, ρ_{AP}^j et ρ_{ACP}^j respectivement, où cette dernières correspondante au nœud actionneur qui a été choisi selon l'équation 5.1 sur le nœud « j ». k indique l'étape de démarrage de calcul, si l'agent « SA » choisit de se déplacer vers un nœud actionneur, alors k=2, sinon k=1 (resp., dans le cas d'agent « AA » toujours k=1). $\rho_t^{\min}$ et $\rho_t^{\max}$ représentent la valeur minimale et maximale de la concentration correspondante pour chaque type de phéromone existant dans la table de phéromones PT du nœud « j », respectivement.

En ce qui concerne l'ensemble de coefficients {w_1, w_2, w_3, w_4}de l'équation 5.2, que l'agent hérite de ses parents. Ces coefficients régissent la façon dont les agents exécutent leurs comportements du déplacement. Chaque agent quand il est « AS », il utilise le sous-ensemble {w_1, w_2, w_3}, dès qu'il devienne « AA », il utilise le sous-ensemble {w_2, w_3, w_4}. Cela est expliqué bien détail dans la section 3.3.3.

- **Changement de rang** : lorsqu'un agent est arrivé à un nœud actionneur qui n'est pas isolé, il doit promouvoir son rang de « SA » à « AA », puis continue son déplacement vers la SB à travers les nœuds actionneur intermédiaires. Cependant, au cas où un agent « AA » s'est déplacé au nœud actionneur isolé, alors il doit dégrader son rang de « AA » à « SA », puis continue son déplacement vers la SB à travers les nœuds capteur intermédiaires, ainsi qu'il doit émettre la phéromone IP, si ce nœud actionneur n'a pas déjà été référencé pas un autre agent, et ainsi de suite. Ce comportement est effectué par la mise à jour du champ `Type` de la structure de l'agent mobile (cf., la section 3.6.1.2), où la valeur 0 pour l'agent « SA » et 1 pour l'agent « AA »

5.2.1.1 La Séquence des comportements de l'agent

La figure 5.2 illustre la séquence de comportements que chaque agent effectue sur un nœud capteur et un nœud actionneur pendant chaque cycle d'activation. Premièrement, quand un agent SA capte les données, il s'acquiert une quantité constante d'énergie qu'il utilise pour invoquer les comportements, dès que l'agent la consomme totalement, donc il meurt (cette situation nommé la famine). En effet, dans chaque cycle d'activation, l'agent SA se réplique si la valeur de l'énergie acquise est suffisante pour invoquer le comportement de réplication. Chaque agent se déplace saut par saut vers la SB à travers les nœuds intermédiaires (y compris les actionneurs), dans ces derniers l'agent doit attendre certain temps afin d'invoquer, si possible, le comportement *Swarming* avec d'autres agents. Lorsqu'un agent veut invoquer le comportement de déplacement, tout d'abord, il choisit selon l'équation 2.333 un nouveau nœud auquel il se déplacera. Si le déplacement d'un agent à sa nouvelle destination, est effectué avec succès, il émit la phéromone de migration, sinon il émit la phéromone d'alerte vers les nœuds voisins seulement. Une fois que les agents SA arrivent aux nœuds actionneurs qui ne sont pas isolés, ils changent leurs rangs de SA à AA, et vice-versa.

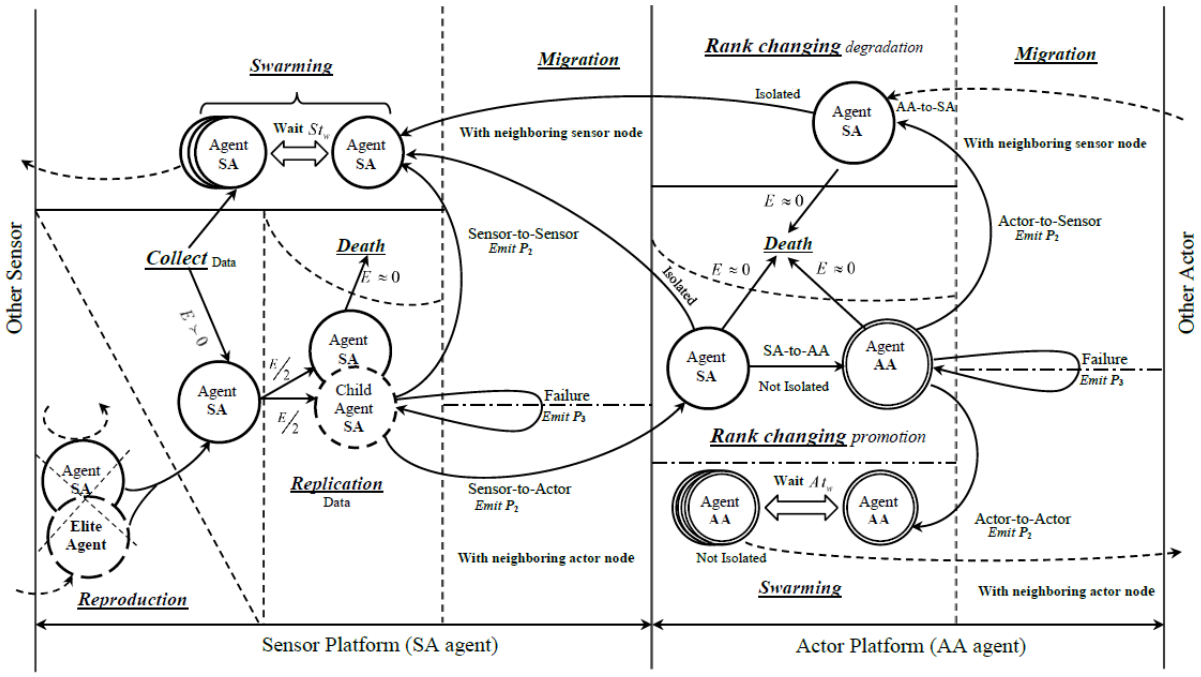

Figure 5.2 La séquence de comportement d'un agent au sein de BISSA.

5.2.1.2 La structure génotypique d'un agent dans BISSA

Chaque agent porte ses propres paramètres, en tant qu'une structure génotypique, qui régit l'invocation de ses comportements dans le nœud sur lequel il réside. Dans BISSA, la structure génotypique de l'agent se compose d'un paramètre et des quatre gènes : le délai d'attente (St_w ou At_w) concerne le comportement de *swarming*, c'est un paramètre acquis et les quatre coefficients {w_1, w_2, w_3, w_4} de l'équation 5.2, ce sont des gènes hérités, la valeur de St_w et d'At_w, est toujours positive et est calculée de façon aléatoire et la valeur de chaque coefficient w_t est de {-1,0,1}. Cette structure est représentée comme dans la figure 4.3, mais le gène w_4 correspond à la phéromone ACP.

5.2.1.3 Les Objectifs vs les contraintes

Dans cette partie, nous avons gardé les mêmes définitions des objectifs cités dans la section 3.4.2, à part le coût que nous définissons comme suit :

- **Coût (C)** : représente la quantité d'énergie qui est consommée par l'agent « *i* » pendant son déplacement de nœud source « j » à la SB. Il se mesure comme suit :

$$C_i = \frac{E_{sense} + \left(N_{tx} \times \left(E_{tx} \times S_i + E_{radio} \right) \Big/ \sqrt{\left(x_j - x_s \right)^2 + \left(y_j - y_s \right)^2} \right)}{N_{data}} \qquad (5.3)$$

Où E_{sense}, E_{tx}et E_{radio} représentent respectivement la consommation d'énergie nécessaire (en mW) pour collecter les données du nœud capteur, transmettre un *bit* de données entre deux

nœuds voisins et fonctionner le circuit radio dans un nœud. Ces constantes sont obtenues à partir de [29]. N_{tx} est le nombre total de transmissions capteur-à-capteur et capteur-à-actionneur, qui sont effectués par l'agent « *i* » pendant son déplacement à la SB, y compris ses transmissions réussies et échouées, ainsi que les transmissions de ses phéromones de migration et d'alerte, mais sans tenir compte des transmission actionneur-à-actionneur (parce que la capacité énergétique d'un actionneur est illimitée). S_i est la taille de l'agent « *i* » en bit. N_{data} est le nombre total de données transportées par un agent « *i* ».

5.2.2 BISSA-Runtime et BISSA-serveur

BISSA-runtime (BISSA-Rc ou BISSA-Ra) interprète le code du programme de l'agent et effectue ses actions, de sorte qu'il offre un ensemble de services pour les agents en cours d'exécution sur le nœud local (cf., la figure 5.1). Par exemple, il implémente les comportements de l'agents comme services réutilisables. Ainsi qu'il permet de maintenir et de contrôler un ensemble de nœuds voisins existant dans la portée de communication du nœud local (i.e., le routage de table) et gère les phéromones émises sur le nœud local. De plus, chaque plate-forme est responsable du contrôle de la fonctionnalité des nœuds locaux.

La différence entre BISSA-Rc et BISSA-Ra réside dans la mise en œuvre et la conception de chaque comportement de l'agent, c'est-à-dire la façon d'invoquer ces comportements, où dans.

- **La conception** : l'architecture de chaque agent consiste de huit comportements (cf., la section 5.2.1), où les résultats de l'invocation d'un comportement par l'agent au sein du BISSA-Rc ou au sein du BISSA-Ra ne sont pas nécessairement les mêmes. Par exemple dans BISSA-Rc, au cas où l'agent effectue le comportement « *la mort* » et si tous les agents sont morts au même temps, alors il doit créer un nouvel agent au hasard, mais dans BISSA-Ra, il ne fait rien.
- **La mise en œuvre** : il y a des comportements qui ne sont invoqués que soit dans BISSA-Rc, soit dans BISSA-RA. Par exemple, dans BISSA-Rc, l'agent peut invoquer tous les comportements sauf le comportement « *changement de rang* », mais dans BISSA-Ra, l'agent ne peut pas invoquer les comportements suivants « *gain d'énergie* », « *réplication* » et « *reproduction* ».

Afin d'utiliser les ressources des nœuds actionneurs qui existent dans le réseau de la meilleure façon possible, nous avons besoin d'utiliser un mécanisme qui permet d'optimiser les structures génotypiques des agents afin qu'ils s'adaptent aux conditions dynamiques du réseau, ce mécanisme nommé BISSA-serveur (BISSA-R), ce qui est le MONet-R avec quelques modifications simples concernant la nouvelle architecture de l'agent seulement. En ce qui concerne les opérations génétiques (croisement et mutation), nous avons utilisé les mêmes opérations expliqués dans la section 3.5.5, où pour la détermination de l'agent élite qui est similaire à l'agent local comme un agent parents, se fait selon l'équation 3.16, tout en tenant compte du quatrième gène « w_4».

5.3 Evaluation les performances

Afin d'évaluer l'efficacité énergétique du BISSA et aussi en termes d'une meilleure exploitation de l'hétérogénéité qui se trouve dans les réseaux comme WSAN, afin d'améliorer l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation de ce type de réseau, ainsi que pour assurer la bonne distribution de la consommation énergétique sur tous les nouds dans le réseau, des simulations ont été effectuées avec la même configuration de la section 3.6.2 et aussi les mêmes paramètres.

5.3.1 Résultats de simulation dans WSAN (SANET)

5.3.1.1 Les performances de l'agent

La figure 5.3 illustre les valeurs moyennes d'objectifs que tous les agents (SA et AA) estiment dans chaque cycle de simulation (*tick*) d'un réseau WSAN (SANET) dynamique.

La sous-figure 5.3 (a) illustre les résultats de simulation d'un RCSF statique, avant et après l'addition des nœuds actionneurs, où au 80^{ieme} cycle le RCSF devient WSAN (SANET) après qu'y fut ajouté cinq (5) nœuds actionneurs à des positions prédéterminées comme {(2,2);(2,7);(5,5);(7,2);(7,7)} dans le réseau. Toutes les performances se dégradent brusquement, parce que les agents essaient de se déplacer vers le nœud actionneur le plus proche, au lieu de se déplacer vers la BS. En outre, les phéromones ne sont pas suffisamment disponibles sur ces nœuds actionneurs, ce qui ne permet pas aux agents de continuer leurs déplacements vers la BS de façon efficace, mais les agents améliorent les valeurs de tous leurs objectifs d'une façon plus rapides que dans le RCSF (cf., la figure 3.20), et ces objectifs convergent à nouveau au 108ème cycle. Il est à noter que dans WSAN (après le 80ème cycle), les valeurs moyennes du coût moyen et de la latence sont inférieures à ceux du RCSF (avant le 80ème cycle), parce que la consommation d'énergie pour transmettre un agent de nœud actionneur à un autre actionneur, n'est pas comptée car leurs capacités énergétiques sont illimitées (cf., l'équation 5.3), et le temps d'arrivée d'un agent à une BS est diminuée parce que la portée de communication du nœud actionneur est plus grande que celle du nœud capteur, ce qui permet également de réduire le nombre de sauts effectués par l'agent de son nœud source à la SB et aussi les valeurs moyennes de l'agrégation des données sont plus élevées presque elles ont doublées car beaucoup des agents arrivent simultanément au même nœud actionneur, ce qui les permet de s'agréger.

Les sous-figures 5.3 (b) à (c) illustrent des résultats de simulations d'un réseau WSAN (SANET) dynamique, ce qui explique comment les agents réagissent à chaque changement dans le réseau, ceci se produit à la 150^{ième} cycle. En général, une fois que le changement se produit, les performances se dégradent.

Dans la sous-figure 5.3 (b), une fois que les 25 nœuds sont ajoutés de manière aléatoire sur la zone observée, la même chose s'est produite comme dans la sous-figure 3.20 (b). Comparé aux résultats de sous-figures 3.20 (b), la sous-figures 5.3 (b) montre que les valeurs moyennes des performances, se dégradent légèrement parce que les nouveaux agents essaient de se déplacer vers les nœuds actionneurs les plus proches, donc c'est un impact partiel. Ensuite, les agents améliorent rapidement les valeurs de toutes leurs performances qui convergent à nouveau au 170^{ième} cycle. La même chose

s'est produite si nous ajoutons simultanément des nœuds capteurs et des nœuds actionneurs au réseau, comme l'illustre la sous-figure 5.3 (c).

La sous-figure 5.3 (d) montre les performances des agents lorsqu'un nœud actionneur tombe en panne. Au 150^{ième} tique, nous avons désactivé le nœud actionneur qui se situe à la position (5,5), c'est-à-dire qui se trouve au centre du réseau WSAN. C'est le pire des cas car tous les autres nœuds actionneurs deviennent isolés. En conséquence, les valeurs moyennes des objectifs des agents, se dégradent brusquement parce que certains agents « AS » se déplacent vers des nœuds actionneur isolés ou défaillants et certains agents « AA » essayent de se déplacer vers des nœuds actionneurs défaillants. Si nous comparons la sous-figure 5.3 (a) à la sous-figure 5.3 (c), les valeurs d'objectifs s'améliorent un peu lentement, on remarque que les valeurs de la latence et de l'agrégation de données, sont pires que celles qui se sont produites avant le 150^{ième} cycle, et elles sont presque identiques dans le cas du RCSF parce que le nombre de sauts actionneur-à-actionneur (transmission) devient nul ce qui augmente également le nombre de sauts acteur-à-capteur, et aussi l'agrégation de données est désactivée dans la plupart des nœuds actionneurs, où le taux de réussite est diminué à la valeur la plus basse au 150ème cycle, parce qu' à ce moment le réseau est divisé en quatre sous-réseaux. Mais le réseau a guéri de manière autonome de cette perturbation en se basant sur les nœuds capteurs pour créer de nouveaux liens indirects entre des nœuds actionneurs isolés (i.e., les nœuds capteurs jouent le rôle d'un pont qui relie les nœuds actionneurs isolés) ce qui permet aux agents de se déplacer à nouveau vers la BS.

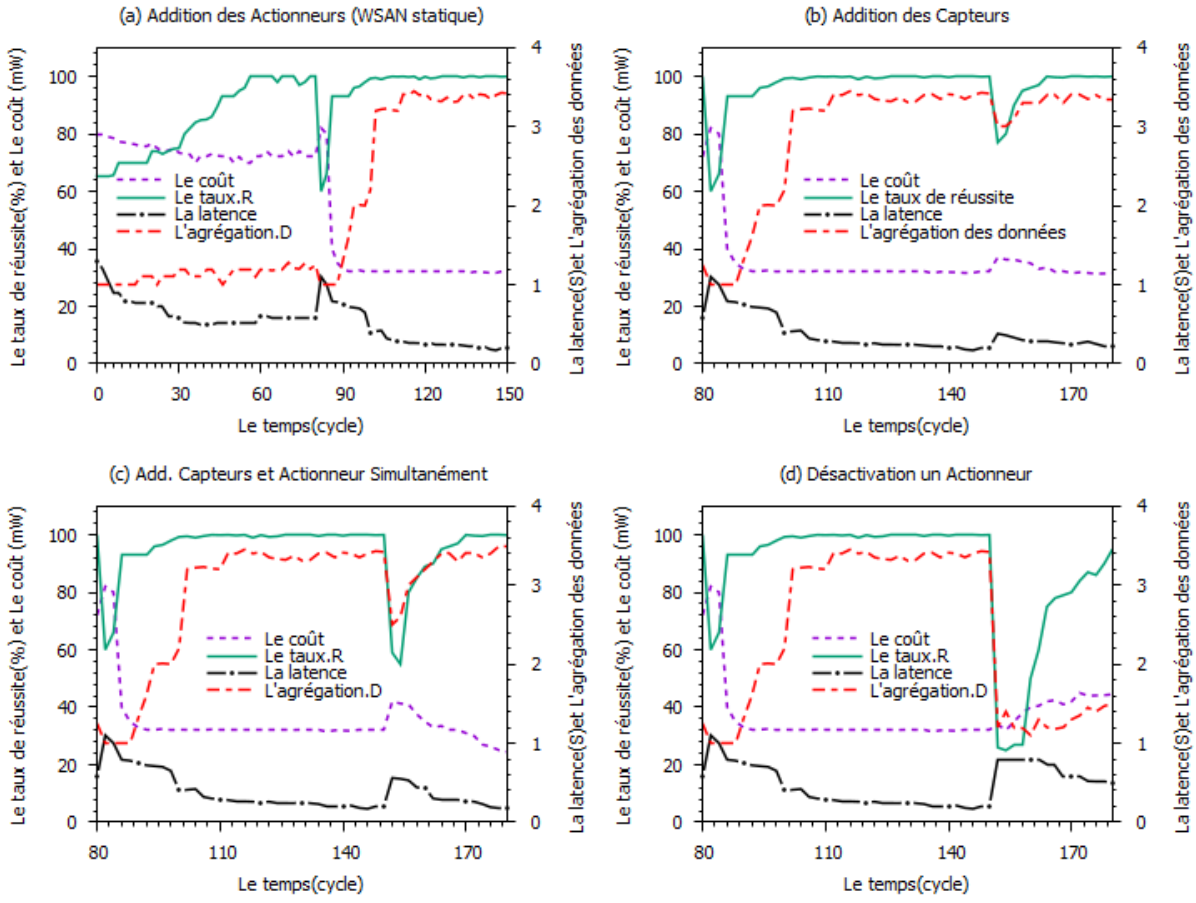

Figure 5.3 les performances des agents dans WSAN avec BISSA

La figures 5.3 démontre que le BISSA permet aux agents de s'adapter de façon autonome et plus rapidement aux changements dynamiques dans le RCASF et d'optimiser leurs performances en évoluant leurs objectifs. Ainsi que le BISSA exploite les liens asymétriques entre les nœuds actionneurs et les nœuds capteurs (i.e., différentes portées de communication dans le réseau) afin de réparer les perturbations qui se sont produites dans le RCASF.

5.3.1.2 La consommation d'énergie

Les figures 5.4 et 5.5 illustrent la consommation d'énergie de chaque nœud capteur et de chaque nœud actionneur dans deux configurations différentes (scénario). La figure 5.4 illustre la consommation d'énergie de chaque nœud dans le réseau avec le scénario d'addition de nœuds actionneurs, comme dans la figure 5.3 (a). La sous-figure 5.4 (a) montre que les nœuds actionneurs consomment beaucoup plus d'énergie que les nœuds capteurs, ainsi que certains nœud actionneur, en particulier les nœuds actionneurs qui se situent au centre du réseau et à la proximité de BS, ils consomment beaucoup plus d'énergie que les autres nœuds actionneurs. Les sous-figures 5.4 (de b à c) montrent que presque tous les nœuds capteurs consomment des quantités d'énergie faibles et convergentes.

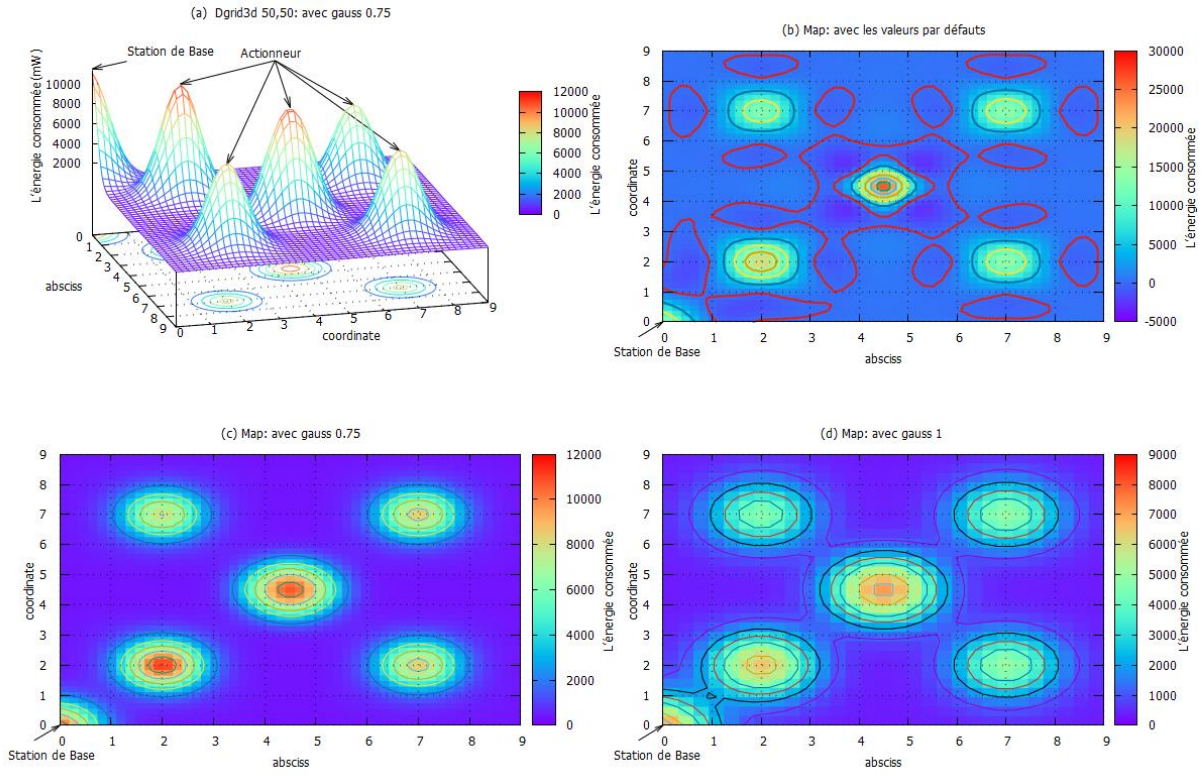

Figure 5.4 Répartition de la dissipation d'énergie dans WSAN avec BISSA

Dans la figure 5.5, le scénario est identique à celui de la figure 5.4, mais le nœud actionneur au centre du réseau est désactivé (i.e., il tombe en panne) comme dans la sous-figure 5.3 (d). La sous-figure 5.5 (a) montre que certains nœuds capteurs, en particulier ceux qui se situent au centre du réseau, consomment beaucoup plus d'énergie que les autres nœuds capteurs. Cela signifie que les ressources d'énergie des nœuds capteurs situés au milieu du réseau s'épuisent plus vite que les autres nœuds comme l'illustre la sous-figure 5.5 (de b à c). Ce qui augmente le risque que le réseau se sépare en plusieurs sous réseaux, et que certains nœuds capteurs ne puissent pas communiquer avec la BS (sur

ces figures, chaque intersection de deux lignes de grille représente un nœud capteur ou un nœud actionneur).

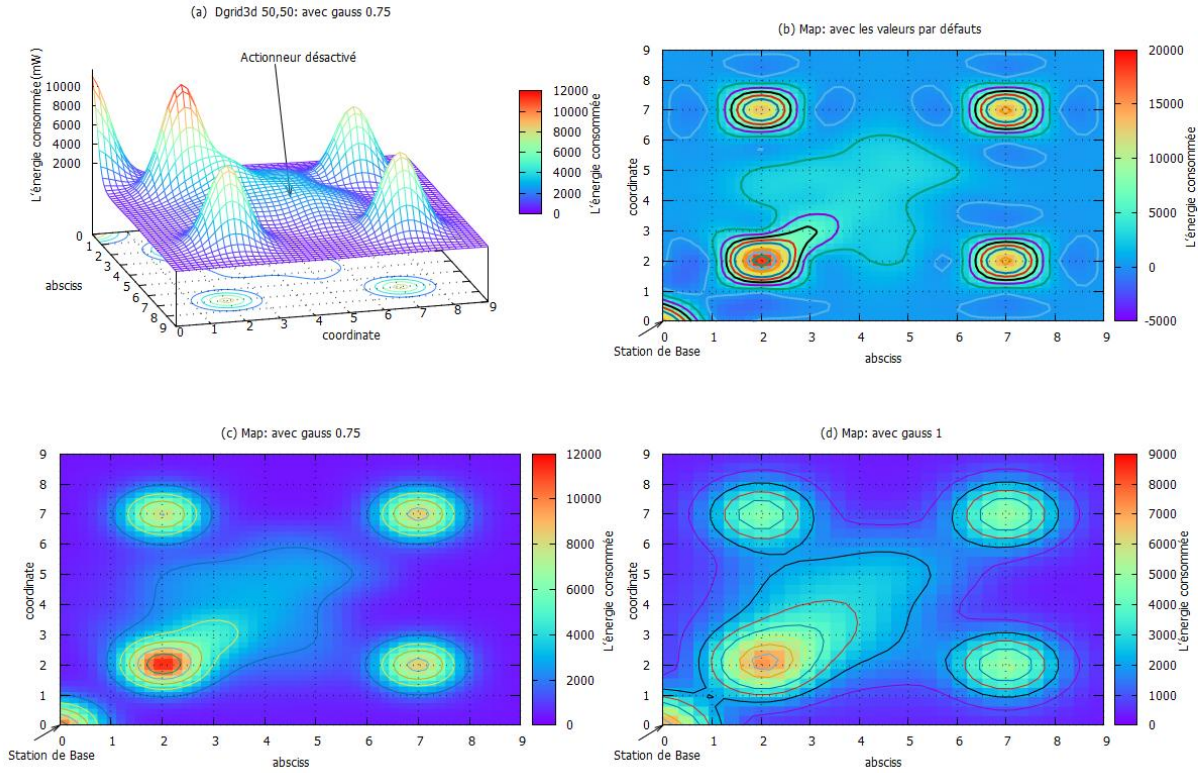

Figure 5.5 Répartition de la dissipation d'énergie dans WSAN (BISSA) avec des nœuds actionneurs isolés.

5.4 Conclusion

Dans ce chapitre, nous avons proposé une extension à notre framework MONet, appelée BISSA, qui convient à tous les types d'applications des réseaux de capteur sans fil hétérogènes comme le réseau de capteurs et actionneurs sans fil (WSAN) dans lequel il existe une grande hétérogénéité entre les nœuds capteurs et les nœuds actionneurs, en termes de portée de communication et de capacité de traitement et d'énergie. Le BISSA se base sur des mécanismes bio-inspirés et de nouvelle architecture de l'agent mobile afin d'aborder plusieurs problèmes dans les réseaux hétérogènes, à savoir l'auto-configuration, l'auto-organisation, l'auto-optimisation et l'auto-réparation. Dans notre proposition, les problèmes d'économiser de l'énergie et d'équilibrer la consommation énergétique sont abordés avec une nouvelle stratégie dans laquelle les ressources disponibles sur chaque nœud du réseau sont pleinement exploitées. Notre approche consiste à produire une nouvelle architecture de l'agent basée sur la structure génotypique et les comportements afin d'améliorer les performances du réseau et de garantir le coût et la latence les plus faibles possibles et le taux de réussite le plus haut possible. Les résultats de l'évaluation montrent que notre Framework middleware BISSA fonctionne mieux dans les réseaux de capteurs sans fil, qu'ils soient homogènes, ou hétérogènes, ainsi qu'il présente une bonne solution pour réduire la consommation énergétique et réparer de façon autonome les perturbations qui se sont produites dans le réseau.

Conclusion générale

Conclusion générale

Bilan

Les réseaux de capteurs sans fil ont un large potentiel avec diverses applications pratiques et utiles. Cependant, il existe encore beaucoup de problèmes qui doivent être abordés pour un fonctionnement efficace de ces réseaux dans des applications réelles. Parmi les problèmes fondamentaux et importants dans ces réseaux de capteurs, nous pouvons mentionner l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation, ainsi que l'efficacité énergétique. Dans cette thèse, nous nous sommes intéressés aux réseaux de capteurs sans fil hétérogènes. Nous avons pris comme un cas d'étude les réseaux de capteurs et actionnaire sans fil qui sont principalement des réseaux hétérogènes et dynamiques. L'hétérogénéité est causée par la coexistence des nœuds capteurs à faibles ressources et des nœuds actionneurs riches en ressources. Ces derniers devraient être utilisés de manière différenciée par le réseau. C'est dans ce contexte que se déroule cette thèse dans laquelle nous avons développé une plateforme middleware (un Framework) bio-inspirée qui traite les problèmes cités auparavant, premièrement cette plateforme dédiée au réseau de capteurs homogènes. Ensuite, nous avons développé une extension de cette plateforme s'appuyant sur l'hétérogénéité du réseau de capteurs et actionnaires afin d'assurer l'exploitation de ressource de nœuds actionnaire. Ces travaux ont fait l'objet de plusieurs publications qui sont présentés dans l'annexe Publications.

Dans un premier lieu, nous nous sommes intéressés aux premières problématiques dans les réseaux de capteurs sans fil qui sont l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation, ainsi que l'efficacité énergétique dans un contexte homogène. Se basant sur l'idée que les systèmes biologiques ont des mécanismes qui résolvent ces problèmes, ainsi que la plupart des plateformes ou protocoles existants ne les traitent pas tous à la fois, nous avons proposé, dans le chapitre 3, une plateforme middleware dédiée au RCSF homogènes, appelée MONet. Dans cette nouvelle proposition, ses mécanismes s'inspirant d'un système biologique, à savoir « la colonie d'abeillés », où les abeilles sont représentées par des agent mobile « des agent logiciel » qui acheminent les données captées vers la station de base, basé sur le multi-sauts. Ces agents sont autonomes et ont un ensemble de comportements qui invoquent les actions qu'ils peuvent effectuer dans un comportement, appelé « stigmergie », qui gère la communication entre eux en mettant des phéromones sur les nœuds, ainsi qu'ils ont des structures génotypiques renouvelables et améliorables au sein d'un algorithme génétique implémenté dans un serveur du MONet. En effet, nous avons montré, en utilisant le simulateur PowerTOSSIM, que le MONet assure l'auto-organisation, l'auto-configuration, l'auto-optimisation et l'auto-réparation dans les réseaux homogènes.

Ensuite, nous avons présenté, dans le chapitre 4, une variante de notre proposition pour répondre au problème d'efficacité énergétique. Cette variante basée sur deux mécanismes dont le premier c'est un mécanisme décentralisé qui se base sur l'égoïsme de l'agent, cette caractéristique détermine les cas

dans lesquels l'agent émet les phéromones pendant son déplacement ou non. Il y a trois catégories, à savoir, l'agent passif-égoïste n'envoie jamais de phéromones, l'agent positif-égoïste les envoie si nécessaire et l'agent non-égoïste doit toujours les envoyer. Le but principal de ce mécanisme, est de trouver le meilleur compromis entre ces trois catégories d'agents dans le réseau, afin de minimiser l'émission de phéromones. Le deuxième est un mécanisme centralisé qui se base sur un seuil calculé par le serveur de MONet et chaque agent n'envoie les phéromones que si ce seuil dépasse les concentrations de phéromones indiquées dans le nœud sur lequel il se situe actuellement. Les deux mécanismes diminuent le nombre d'émission de phéromones d'environ un tiers, et aussi distribuent la consommation énergétique de façon équitable sur la plupart des nœuds de réseau ce qui permet de prolonger sa durée de vie.

Enfin, nous nous sommes intéressés à l'exploitation de la richesse en ressources des nœuds spéciaux qui constituent avec les nœuds capteurs des réseaux hétérogènes, cette richesse ouvre de nouvelles opportunités pour améliorer les performances du réseau. En effet, nous avons proposé, dans le chapitre 5, une extension du MONet pour la collecte des données et/ou la détection des évènements dédiés aux WSANs appelée BISSA. Cette extension a pour objectif d'assurer l'exploitation des ressources qui sont disponibles sur les nœuds actionnaire comme la puissance de transmission qui est plus élevée que celle dans les nœuds capteurs ce qui permet d'améliorer le taux de réussite et de diminuer le coût tout en réduisant significativement le nombre de messages dupliqués et de minimiser la latence tout en réduisant le nombre de sauts de bout-en-bout, ainsi qu'elle tire profit la capacité de stockage de nœuds actionnaire, permet aussi d'assurer un degré d'agrégation de données élevé sans effets secondaire sur les autres performance du réseau. Les résultats de l'évaluation montrent que notre proposition BISSA permet d'améliorer les performances de réseau d'environ une moitié par rapport au MONet et de fonctionner mieux dans les réseaux de capteur sans fil, soit homogènes, soit hétérogènes, ainsi qu'il présente une bonne solution pour réduire la consommation énergétique et réparer de façon autonome les perturbations qui se sont produites dans le réseau.

Perspectives

L'interface utilisateur graphique (GUI)

Afin de faciliter l'utilisation du MONet (resp., BISSA), nous proposons de développer une interface utilisateur graphique, cette interface basée sur l'emploi d'éléments graphiques tels que les fenêtres, les icônes et les menus, qui visent la simplicité d'emploi de MONet et qui créent un environnement de travail convivial. Elle peut consister en plusieurs fenêtres dont la première pour afficher les données captées, la deuxième pour afficher les valeurs moyennes des performances en temps réel et troisième permet à l'utilisateur de saisir les paramètres de serveur comme les contraintes et les paramètres de GA, ainsi qu'un ensemble de boutons qui permet à l'utilisateur de contrôler le serveur comme un bouton de démarrage et d'arrêt et un bouton pour installer les fichiers du MONet-runtime sur les capteurs… etc.

La sécurité

Les réseaux RCSF peuvent manipuler des données sensibles et/ou confidentielles. Or, une caractéristique des RCSF est qu'ils sont déployés dans des environnements hostiles et/ou ouverts. Cette caractéristique oblige l'utilisation des techniques pour garantir l'intégrité des données circulant dans le réseau. Cependant, MONet (resp. BISSA) n'assure pas la sécurité, donc nous aurons besoin de développer un mécanisme qui assure la sécurité de façon autonome, appelé l'auto-protection. Cette auto-protection, soit un mécanisme bio-inspiré qui doit être léger et se base sur la phéromone de station de base, soit un algorithme de cryptographie traditionnel. En effet, ces dernières sont très coûteuses et nécessitent une grande puissance de calcul. Ce qui exige de s'appuyer sur les nœuds riches en ressources dans les réseaux pour servir comme point d'exécution d'algorithme de sécurité complexe. Ces nœuds riches en ressources peuvent jouer le rôle de distributeur de clé de sécurité de façon autonome.

L'exploitation d'autres types d'hétérogénéité

Nous nous sommes intéressés dans chapitre 5 à l'hétérogénéité au niveau des puissances de transmission et aussi à l'hétérogénéité au niveau des capacités de stockage des nœuds constituant le réseau. Les autres types d'hétérogénéité ouvrent aussi des opportunités et méritent d'être étudiés. Les nœuds avec des ressources énergétiques illimitées et des capacités de calcul et de mémoire importantes doivent être plus sollicités. Ces nœuds riches en ressources peuvent jouer le rôle de points de traitement de données locales, ce qui minimise le nombre de messages transmués dans le réseau. Ils peuvent aussi jouer le rôle de nœuds permettant de décider et d'affecter les actions appropriées aux nœuds dans leurs voisinages.

Le déploiement des actionneurs

Nous avons vu dans le chapitre 5 que l'influence des positions des actionneurs sur les performances du réseau est très importante. Donc nous aurons besoin d'un mécanisme, qui détermine soit les meilleures positions des actionneurs dans le cas où les actionneurs sont fixes, soit optimise les postions des actionneurs dans le cas où les actionneurs sont mobiles. Ce problème est un problème d'optimisation multi-objectifs (MOP). En utilisant le même algorithme d'optimisation multi-objectif qui est implémenté au niveau de serveur afin de résoudre ce problème, où nous définissons la fonction objective comme une fonction fitness qui indique le minimum de la distance de transmission effective totale entre tous les nœuds actionneurs et la SB sous quelques contraintes comme le nombre de nœuds hétérogènes (les actionneurs) qui peuvent être déployés, la portée de communication de l'actionneur et du capteur, la couverture, …etc.

Liste des publications

- **Kamel Barka, Azeddine Bilami and Samir Gourdache**, (2017) "MONet: A framework for self-adaptive energy-aware middleware for dynamic wireless sensor network", *International Journal of Pervasive Computing and Communications*, Vol. 13 Issue: 4, pp.345-369, https://doi.org/10.1108/IJPCCD-17-00009

- **Samir Gourdache, Azeddine Bilami and Kamel Barka**. (2019). A Framework for Spectrum Harvesting in Heterogeneous Wireless Networks Integration. *Journal of King Saud University-Computer and Information Sciences.*

- **Samir Gourdache, Azeddine Bilami and Kamel Barka**,(2018)"Spectrum harvesting for heterogeneous wireless networks integration", *Wireless Networks,* pp.1-17https://doi.org/10.1007/s11276-018-1822-0

- **Lyamine GUEZOULI, Kamel BARKA and Souheila BOUAM**, (2018) *"* Towards Mobile Nodes Collaboration to Ensure the Receipt of Supervisory Data in Wireless Sensor Networks". *In: The 3rd IEEE Conference on Smart Cities and Innovative Systems, Marrakech, Morocco on 21st - 24th October 2018 (SCIS'18).*

- **Lyamine GUEZOULI, Kamel BARKA, Souheila BOUAM and Abdelmadjid ZIDANI**, (2018). 'A variant of Random WayPoint mobility model to improve routing in Wireless Sensor Networks'. *International Journal of Information and Communication Technology* (*IJITC*). Vol. 13 Issue: 4, DOI: 10.1504/IJICT.2018.10012666.

- **Guezouli Lyamine, Barka, Kamel, Bouam, Souheila**, **Djamila, BOUHTA and Souha AOUTI** (2017) "Mobile sensor nodes collaboration to optimize routing process-based mobility model."*In: Wireless Networks and Mobile Communications (WINCOM), 2017 International Conference on*. IEEE, p. 1-5.DOI: 10.1109/WINCOM.2017.8238161.

Bibliographie

[1] I. F. Akyildiz et I. H. Kasimoglu, «Wireless sensor and actor☆ networks: research challenges» *Ad hoc networks,* vol. 2, n° 14, pp. 351-367, 2004.

[2] A. M. Zungeru, L.-M. Ang et K. P. Seng, «Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison» *Journal of Network and Computer Applications,* vol. 35, n°15, pp. 1508-1536, 2012.

[3] M. Saleem, G. A. Di Caro et M. Farooq, «Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions,» *Information Sciences,* vol. 181, n°120, pp. 4597-4624, 2011.

[4] F. Dressler et O. B. Akan, «Bio-inspired networking: from theory to practice,» *IEEE Communications Magazine,* vol. 48, n° 111, pp. 176-183, 2010.

[5] D. Karaboga et B. Akay, «A survey: algorithms simulating bee swarm intelligence,» *Artificial intelligence review,* vol. 31, n° 11-4, pp. 61-85, 2009.

[6] S. Jabbar, R. Iram, A. A. Minhas, I. Shafi, S. Khalid et M. Ahmad, «Intelligent optimization of wireless sensor networks through bio-inspired computing: survey and future directions,» *International Journal of Distributed Sensor Networks,* vol. 9, n° 12, p. 421084, 2013.

[7] F. Dressler, Self-organization in sensor and actor networks, John Wiley & Sons, 2008.

[8] L. Cobo Campo, «Gestion de la qualité de service et planification optimale de réseaux de capteurs multimédia sans fil,» Thèse de doctorat,Ecole Polytechnique de Montréal, 2011.

[9] C.-T. Kone, «Conception de l'architecture d'un réseau de capteurs sans fil de grande dimension,» Thèse de doctorat,Université Henri Poincaré-Nancy I, 2011.

[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam et E. Cayirci, «Wireless sensor networks: a survey,» *Computer networks,* vol. 38, n° 14, pp. 393-422, 2002.

[11] N. Labraoui, M. Gueroui et L. Sekhri, «On-off attacks mitigation against trust systems in wireless sensor networks,» *{IFIP International Conference on Computer Science and its Applications_x000D_,* pp. 406-415, 2015.

[12] A. Ari, A. Gueroui, B. O. Yenke et N. Labraoui, «Energy efficient clustering algorithm for wireless sensor networks using the ABC metaheuristic,» *Computer Communication and Informatics (ICCCI), 2016 International Conference on,* pp. 1-6, 2016.

[13] E. T. Fute et E. Tonye, «Modelling and self-organizing in mobile wireless sensor networks: Application to fire detection,» *International Journal of Applied Information Systems, IJAIS, New York, USA,* vol. 5, n° 13, 2013.

[14] S. A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie et J. Ma, «Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing,» *In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW'07.,* pp. 113-120, 2007.

[15] R. Loomba, R. de Frein et B. Jennings, «Selecting energy efficient cluster-head trajectories for collaborative mobile sensing,» *Global Communications Conference (GLOBECOM), 2015 IEEE,* pp. 1-7, 2015.

[16] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu et S. Singh, «Exploiting heterogeneity in sensor networks,» *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE,* vol. 2, pp. 878-890, 2005.

[17] P. Boonma et J. Suzuki, «BiSNET: A biologically-inspired middleware architecture for self-managing wireless sensor networks,» *Computer networks,* vol. 51, n° 116, pp. 4599-4616, 2007.

[18] P. Kulkarni, D. Ganesan, P. Shenoy et Q. Lu, «SensEye: a multi-tier camera sensor network,» *Proceedings of the 13th annual ACM international conference on Multimedia,* pp. 229-238, 2005.

[19] V. Jelicic, M. Magno, D. Brunelli, V. Bilas et L. Benini, «An energy efficient multimodal Wireless Video Sensor Network with eZ430--RF2500 modules,» *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on,* pp. 161-166, 2010.

[20] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic et others, «VigilNet: An integrated sensor network system for energy-efficient surveillance,» *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, n° 11, pp. 1-38, 2006.

[21] C. E. R. Lopes et L. B. Ruiz, «On the development of a multi-tier, multimodal wireless sensor network for wild life monitoring,» *Wireless Days, 2008. WD'08. 1st IFIP*, pp. 1-5, 2008.

[22] B. Romdhani, «Exploitation de l'hétérogénéité des réseaux de capteurs et d'actionneurs dans la conception des protocoles d'auto-organisation et de routage,» 2012.

[23] J. Molina, J. M. Mora-Merchan, J. Barbancho et C. Leon, «Multimedia data processing and delivery in wireless sensor networks,» *Wireless Sensor Networks: Application-Centric Design*, 2010.

[24] O. Alaoui Fdili, «Optimisation multicritères de la qualité de service dans les réseaux de capteurs multimédia sans fil,» Thèse de doctorat, Université de Valenciennes, 2015.

[25] I. F. Akyildiz et M. C. Vuran, Wireless sensor networks, vol. 4, John Wiley & Sons, 2010.

[26] P. Kulakowski, E. Calle et J. L. Marzo, «Sensors-actuators cooperation in wsans for fire-fighting applications,» *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*, pp. 726-732, 2010.

[27] I. F. Akyildiz, D. Pompili et T. Melodia, «Underwater acoustic sensor networks: research challenges,» *Ad hoc networks*, vol. 3, n° 13, pp. 257-279, 2005.

[28] I. F. Akyildiz et E. P. Stuntebeck, «Wireless underground sensor networks: Research challenges,» *Ad Hoc Networks*, vol. 4, n° 16, pp. 669-689, 2006.

[29] G.-Z. Yang et G. Yang, Body sensor networks, vol. 1, London: Springer, 2006.

[30] P. S. Hall, Y. Hao, Y. I. Nechayev, A. Alomainy, C. C. Constantinou, C. Parini, M. R. Kamarudin, T. Z. Salim, D. T. Hee, R. Dubrovka et others, «Antennas and propagation for on-body communication systems,» *IEEE Antennas and Propagation Magazine*, vol. 49, n° 13, pp. 41-58, 2007.

[31] C. Hertleer, L. Van Langenhove, H. Rogier et L. Vallozzi, «Off-body communication for protective clothing,» *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pp. 301-304, 2009.

[32] I. F. Akyildiz, T. Melodia et K. R. Chowdury, «Wireless multimedia sensor networks: A survey,» *IEEE Wireless Communications*, vol. 14, n° 16, 2007.

[33] J. Zheng et A. Jamalipour, Wireless sensor networks: a networking perspective, John Wiley & Sons, 2009.

[34] S. Katiyar et D. Prasad, «A comprehensive survey of data processing approaches,» *International Journal of Computer Applications*, vol. 126, n° 111, 2015.

[35] X. Xu, R. Ansari, A. Khokhar et A. V. Vasilakos, «Hierarchical data aggregation using compressive sensing (HDACS) in WSNs,» *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, n° 13, p. 45, 2015.

[36] M. Kumar et K. Dutta, «A survey of security concerns in various data aggregation techniques in wireless sensor networks},» *Intelligent Computing, Communication and Devices*, pp. 1-15, 2015.

[37] J. Yick, B. Mukherjee et D. Ghosal, «Wireless sensor network survey,» *Computer networks*, vol. 52, n° 112, pp. 2292-2330, 2008.

[38] A. Ari, A. Gueroui, N. Labraoui et B. O. Yenke, «Concepts and evolution of research in the field of wireless sensor networks,» *International Journal of Computer Networks & Communications*, vol. 7, n° 11, pp. 81-98, 2015.

[39] T. Naumowicz, R. Freeman, A. Heil, M. Calsyn, E. Hellmich, A. Brandle, T. Guilford et J. Schiller, «Autonomous monitoring of vulnerable habitats using a wireless sensor network,» *Proceedings of the workshop on Real-world wireless sensor networks*, pp. 51-55, 2008.

[40] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser et M. Turon, «Wireless sensor networks for structural health monitoring,» *Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 427-428, 2006.

[41] Y. Guo, F. Kong, D. Zhu, A. S. Tosun et Q. Deng, «Sensor placement for lifetime maximization in monitoring oil pipelines,» *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 61-68, 2010.

[42] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline et M. Csail, «PIPENET: A wireless sensor network for pipeline monitoring,» *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pp. 264-273, 2007.

[43] C. R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. Der Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick et others, «Wireless sensor networks for home health care,» *21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW'07.*, vol. 2, pp. 832-837, 2007.

[44] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, M. Welsh et others, «Mercury: a wearable sensor network platform for high-fidelity motion analysis,» *In Sens 09 : Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, vol. 9, pp. 183-196, 2009.

[45] M. Brown, «Users guide developed for the jbrews project,» 1999.

[46] T. Gosnell, J. Hall, C. Jam, D. Knapp, Z. Koenig, S. Luke, B. Pohl, A. Schach von Wittenau et J. Wolford, «Gamma-ray identification of nuclear weapon materials,» 1997.

[47] V. R. Jain, R. Bagree, A. Kumar et P. Ranjan, «wildCENSE: GPS based animal tracking system,» *International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008.*, pp. 617-622, 2008.

[48] D. Malan, T. Fulford-Jones, M. Welsh et S. Moulton, «Codeblue: An ad hoc sensor network infrastructure for emergency medical care,» *International workshop on wearable and implantable body sensor networks*, vol. 5, 2004.

[49] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda et others, «A line in the sand: a wireless sensor network for target detection, classification, and tracking,» *Computer Networks*, vol. 46, n° 15, pp. 605-634, 2004.

[50] F. Xia, «QoS challenges and opportunities in wireless sensor/actuator networks,» *Sensors*, vol. 8, n° 12, pp. 1099-1110, 2008.

[51] D. E. Boubiche et A. Bilami, «Cross layer intrusion detection system for wireless sensor network,» *International Journal of Network Security & Its Applications*, vol. 4, n° 12, p. 35, 2012.

[52] M. Naidja et A. Bilami, «A dynamic self-organising heterogeneous routing protocol for clustered WSNs,» *International Journal of Wireless and Mobile Computing*, vol. 12, n° 12, pp. 131-141, 2017.

[53] T. Watteyne, A. Molinaro, M. G. Richichi et M. Dohler, «From manet to ietf roll standardization: A paradigm shift in wsn routing protocols,» *IEEE Communications Surveys & Tutorials*, vol. 13, n° 14, pp. 688-707, 2011.

[54] W. Masri, «Dérivation d'exigences de Qualité de Service dans les Réseaux de Capteurs Sans Fil basés sur TDMA,» 2009.

[55] M.-M. Wang, J.-N. Cao, J. Li et S. K. Dasi, «Middleware for wireless sensor networks: A survey,» *Journal of computer science and technology*, vol. 23, n° 13, pp. 305-326, 2008.

[56] K. Romer, O. Kasten et F. Mattern, «Middleware challenges for wireless sensor networks,» *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, n° 14, pp. 59-61, 2002.

[57] C. Srisathapornphat, C. Jaikaeo et C.-C. Shen, «Sensor information networking architecture,» *Parallel Processing, 2000. Proceedings. 2000 International Workshops on*, pp. 26-30, 2000.

[58] P. Bonnet, J. Gehrke et P. Seshadri, «Towards sensor database systems,» *Mobile Data Management*, pp. 3-14, 2001.

[59] S. R. Madden, M. J. Franklin, J. M. Hellerstein et W. Hong, «TinyDB: an acquisitional query processing system for sensor networks,» *ACM Transactions on database systems (TODS)*, vol. 30, n° 11, pp. 122-173, 2005.

[60] E. Souto, G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa et C. Ferraz, «A message-oriented

middleware for sensor networks,» *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pp. 127-134, 2004.

[61] S. Li, Y. Lin, S. H. Son, J. A. Stankovic et Y. Wei, «Event detection services using data service middleware in distributed sensor networks,» *Telecommunication Systems*, vol. 26, n° 12-4, pp. 351-368, 2004.

[62] C.-L. Fok, G.-C. Roman et C. Lu, «Mobile agent middleware for sensor networks: An application case study,» *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 382-387, 2005.

[63] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho et M. A. Perillo, «Middleware to support sensor network applications,» *IEEE network*, vol. 18, n° 11, pp. 6-14, 2004.

[64] P. J. Marron, D. Minder, A. Lachenmann et K. Rothermel, «TinyCubus: An Adaptive Cross-Layer Framework for Sensor Networks,» *it-Information Technology*, vol. 47, n° 12, pp. 87-97, 2005.

[65] T. Liu et M. Martonosi, «Impala: A middleware system for managing autonomic, parallel sensor systems,» *ACM Sigplan Notices*, vol. 38, n° 110, pp. 107-118, 2003.

[66] P. Levis et D. Culler, «Maté: A tiny virtual machine for sensor networks,» *ACM Sigplan Notices*, vol. 37, n° 110, pp. 85-95, 2002.

[67] K. Roussel, «Evaluation et amélioration des plates-formes logicielles pour réseaux de capteurs sans-fil, pour optimiser la qualité de service et l'énergie,» 2016.

[68] B. Romdhani, «Exploitation de l'hétérogénéité des réseaux de capteurs et d'actionneurs dans la conception des protocoles d'auto-organisation et de routage,» INSA de Lyon, 2012.

[69] F. Dressler, «A study of self-organization mechanisms in ad hoc and sensor networks,» *Computer Communications*, vol. 31, n° 113, pp. 3018-3029, 2008.

[70] E. H. Bendahmane, «Introduction de fonctionnalités d'auto-optimisation dans une architecture de selfbenchmarking,» Université de Grenoble, 2012.

[71] A. G. Ganek et T. A. Corbi, «The dawning of the autonomic computing era,» *IBM systems Journal*, vol. 42, n° 11, pp. 5-18, 2003.

[72] D. Ghosh, R. Sharman, H. R. Rao et S. Upadhyaya, «Self-healing systems—survey and synthesis,» *Decision support systems*, vol. 42, n° 14, pp. 2164-2185, 2007.

[73] A. E. Hassanien, T.-H. Kim, J. Kacprzyk et A. I. Awad, Bio-inspiring Cyber Security and Cloud Services: Trends and Innovations, vol. 70, Springer, 2014.

[74] D. Wang, J. Liu et others, «Self-protection for wireless sensor networks,» *26th IEEE International Conference on Distributed Computing Systems, 2006. ICDCS 2006.*, pp. 67-67, 2006.

[75] Y. Wang, X.-Y. Li et Q. Zhang, «Efficient self protection algorithms for static wireless sensor networks,» *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pp. 931-935, 2007.

[76] M. Arif, «Fusion de données; ultime étape de reconnaissance de formes: application à l'identification et à l'authentification,» Université de Tours, 2005.

[77] H. B. Demuth, M. H. Beale, O. De Jess et M. T. Hagan, Neural network design, Martin Hagan, 2014.

[78] J. Dong, L. Zhao et L. Zhang, «Face recognition based on neural network ensemble and feature fusion,» *2013 International Conference on Information Science and Technology (ICIST)*, pp. 59-62, 2013.

[79] M. A. Benatia, «Optimisation multi-objectives d'une infrastructure réseau dédiée aux bâtiments intelligents,» INSA de Rouen, 2016.

[80] M. Yagoubi, «Optimisation évolutionnaire multi-objectif parallèle: application à la combustion Diesel,» Université Paris Sud-Paris XI, 2012.

[81] L. J. Fogel, A. J. Owens et M. J. Walsh, «Artificial intelligence through simulated evolution,» 1966.

[82] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with

applications to biology, control, and artificial intelligence, MIT press, 1992.

[83] D. E. Goldberg, «Messy Genetic Algorithms: Motivation Analysis, and First Results,» *Complex systems,* vol. 4, pp. 415-444, 1989.

[84] I. Rechenberg, «Evolutionsstrategie-Optimierung technisher Systeme nach Prinzipien der biologischen Evolution,» 1973.

[85] R. John, «Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection,» 1992.

[86] R. John, Koza .Genetic Programming II, Automatic Discovery of Reusable Subprograms, MIT Press, Cambridge, MA, 1992.

[87] E. Bonabeau, M. Dorigo et G. Theraulaz, Swarm intelligence: from natural to artificial systems, Oxford university press, 1999.

[88] I. Boussaid, «Perfectionnement de métaheuristiques pour l'optimisation continue,» Université Paris-Est, 2013.

[89] L. N. De Castro et J. Timmis, «An artificial immune network for multimodal function optimization,» *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02.,* vol. 1, pp. 699-704, 2002.

[90] L. SAID, «Méthodes bio-inpirées hybrides pour la résolution de problèmes complexes,» Thèse de doctorat, Université Constantine 2, 2013.

[91] J. Timmis, «Artificial immune systems: A novel data analysis technique inspired by the immune network theory,» University of Wales, 2000.

[92] S. Darmoul, «Etude de la contribution des systèmes immunitaires artificiels au pilotage de systèmes de production en environnement perturbé,» Université Blaise Pascal-Clermont-Ferrand II; Université du 7 Novembre à Carthage, 2010.

[93] S. Forrest, A. S. Perelson, L. Allen et R. Cherukuri, «Self-nonself discrimination in a computer,» *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on,* pp. 202-212, 1994.

[94] N. K. Jerne, «Towards a network theory of the immune system,» *Annals of Immunology,* vol. 125, pp. 373-389, 1974.

[95] L. N. De Castro et F. J. Von Zuben, «Learning and optimization using the clonal selection principle,» *IEEE transactions on evolutionary computation,* vol. 6, n° 13, pp. 239-251, 2002.

[96] U. Aickelin et S. Cayzer, «The danger theory and its application to artificial immune systems,» *arXiv preprint arXiv:0801.3549,* 2008.

[97] J. Greensmith, U. Aickelin et S. Cayzer, «Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection,» *International Conference on Artificial Immune Systems,* pp. 153-167, 2005.

[98] D. Dasgupta, S. Yu et F. Nino, «Recent advances in artificial immune systems: models and applications,» *Applied Soft Computing,* vol. 11, n° 12, pp. 1574-1587, 2011.

[99] R. Iram, M. I. Sheikh, S. Jabbar et A. A. Minhas, «Computational intelligence based optimization of energy aware routing in WSN,» *Proceedings of the World Congress on Engineering and Computer Science,* vol. 1, 2001.

[100] A. Jabbari et W. Lang, «Advanced bio-inspired plausibility checking in a wireless sensor network using Neuro-immune systems: autonomous fault diagnosis in an intelligent transportation system,» *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on,* pp. 108-114, 2010.

[101] K. Saleem, N. Fisal, M. S. Abdullah, A. Zulkarmwan, S. Hafizah et S. Kamilah, «Proposed nature inspired self-organized secure autonomous mechanism for WSNs,» *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on,* pp. 277-282, 2009.

[102] S. Mishra et S. K. Patra, «Short term load forecasting using neural network trained with genetic algorithm & particle swarm optimization,» *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on,* pp. 606-611, 2008.

[103] X. Cui, T. Hardin, R. K. Ragade et A. S. Elmaghraby, «A swarm-based fuzzy logic control mobile sensor network for hazardous contaminants localization,» *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on,* pp. 194-203, 2004.

[104] M. Gao et J. Tian, «Wireless sensor network for community intrusion detection system based on improved genetic algorithm neural network,» *Industrial and Information Systems, 2009. IIS'09. International Conference on,* pp. 199-202, 2009.

[105] M. Wang et T. Suda, «The bio-networking architecture: A biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications,» *Proceedings. 2001 Symposium on Applications and the Internet.,* pp. 43-53, 2001.

[106] R. V. Kulkarni et G. K. Venayagamoorthy, «Particle swarm optimization in wireless-sensor networks: A brief survey,» *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 41, n° 12, pp. 262-267, 2011.

[107] F. Dressler, «A study of self-organization mechanisms in ad hoc and sensor networks,» *Computer Communications,* vol. 31, n° 113, pp. 3018-3029, 2008.

[108] A. Michael et H. Takagi, «Dynamic control of genetic algorithms using fuzzy logic techniques,» *Proceedings of the Fifth International Conference on Genetic Algorithms,* pp. 76-83, 1993.

[109] I. Caliskanelli, «A bio-inspired load balancing technique for wireless sensor networks,» Thèse de doctorat, University of York, 2014.

[110] M. Meisel, V. Pappas et L. Zhang, «A taxonomy of biologically inspired research in computer networking,» *Computer Networks,* vol. 54, n° 16, pp. 901-916, 2010.

[111] R. W. Dimand et M. A. Dimand, «The early history of the theory of strategic games from Waldegrave to Borel,» *History of Political Economy,* vol. 27, n° 1 Supplement, pp. 15-27, 1992.

[112] R. T. Marler et J. S. Arora, «Survey of multi-objective optimization methods for engineering,» *Structural and multidisciplinary optimization,* vol. 26, n° 16, pp. 369-395, 2004.

[113] J. Byers et G. Nasser, «Utility-based decision-making in wireless sensor networks,» *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing,* pp. 143-144, 2000.

[114] M. Felegyhazi, J.-P. Hubaux et L. Buttyan, «Cooperative packet forwarding in multi-domain sensor networks,» *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on,* pp. 345-349, 2005.

[115] J. M. McCune, E. Shi, A. Perrig et M. K. Reiter, «Detection of denial-of-message attacks on sensor network broadcasts,» *Security and Privacy, 2005 IEEE Symposium on,* pp. 64-78, 2005.

[116] A. Agah, K. Basu et S. K. Das, «Preventing DoS attack in sensor networks: a game theoretic approach,» *Communications, 2005. ICC 2005. 2005 IEEE International Conference on,* vol. 5, pp. 3218-3222, 2005.

[117] D. A. Miller, S. Tilak et T. Fountain, «" Token" equilibria in sensor networks with multiple sponsors,» *Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on,* pp. 5-pp, 2005.

[118] R. Kannan et S. S. Iyengar, «Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks,» *IEEE Journal on Selected Areas in Communications,* vol. 22, n° 16, pp. 1141-1150, 2004.

[119] V. Garth et P. Niki, «Evolution of Cooperation in Multi-Class Wireless Sensor Networks,» *In Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN),* pp. 489-495, 2007.

[120] J. Yuan et W. Yu, «WSN11-1: Distributed cross-layer optimization of wireless sensor networks: A game theoretic approach,» *In Proceedings of the IEEE Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE,* pp. 1-5, 2006.

[121] N. Sadagopan, M. Singh et B. Krishnamachari, «Decentralized utility-based sensor network design,» *Mobile Networks and Applications,* vol. 11, n° 13, pp. 341-350, 2006.

[122] Z. Zeng, A. Liu, D. Li et J. Long, «A highly efficient DAG task scheduling algorithm for wireless sensor networks,» *In Proceedings of The 9th International Conference for Young Computer Scientists, 2008. ICYCS 2008. ,* pp. 570-575, 2008.

[123] Y. Jin, D. Wei, A. Gluhak et K. Moessner, «Latency and energy-consumption optimized task allocation in wireless sensor networks,» *Wireless Communications and Networking Conference (WCNC), 2010 IEEE,* pp. 1-6, 2010.

[124] D. Miorandi, L. Yamamoto et P. Dini, «Service Evolution in Bio-Inspired Communication Systems,» *ITSSA,* vol. 2, n° 11, pp. 51-60, 2006.

[125] A. Rossi, A. Singh et M. Sevaux, «Lifetime maximization in wireless directional sensor network,» *European Journal of Operational Research,* vol. 231, n° 11, pp. 229-241, 2013.

[126] C.-K. Ting et C.-C. Liao, «A memetic algorithm for extending wireless sensor network lifetime,» *Information Sciences,* vol. 180, n° 124, pp. 4818-4833, 2010.

[127] A. N. Njoya, W. Abdou, A. Dipanda et E. Tonye, «Evolutionary-based wireless sensor deployment for target coverage,» *In 2015 11th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), ,* pp. 739-745, 2015.

[128] K. S. Yildirim, T. E. Kalayci et A. Ugur, «Optimizing coverage in a k-covered and connected sensor network using genetic algorithms,» *Proceedings of the 9th WSEAS international conference on evolutionary computing,* pp. 21-26, 2008.

[129] M. H. Alaiwy, F. H. Alaiwy et S. Habib, «Optimization of actors placement within wireless sensor-actor networks,» *Proceedings of 12th IEEE Symposium on Computers and Communications, 2007. ISCC 2007. ,* pp. 179-184, 2007.

[130] R. Rajagopalan, C. K. Mohan, P. Varshney et K. Mehrotra, «Multi-objective mobile agent routing in wireless sensor networks,» *The 2005 IEEE Congress on Evolutionary Computation, 2005.,* vol. 2, pp. 1730-1737, 2005.

[131] R. Rajagopalan, P. K. Varshney, K. G. Mehrotra et C. K. Mohan, «Fault tolerant mobile agent routing in sensor networks: A multi-objective optimization approach,» *Proc. of IEEE Upstate New York Workshop on Communication and Networking,* 2005.

[132] F. Xue, A. Sanderson et R. Graves, «Multi-objective routing in wireless sensor networks with a differential evolution algorithm,» *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, 2006. ICNSC'06.,* pp. 880-885, 2006.

[133] H. Sin, J. Lee, S. Lee, S. Yoo, S. Lee, J. Lee, Y. Lee et S. Kim, «Agent-based framework for energy efficiency in wireless sensor networks,» *World Academy of Science, Engineering and Technology,* vol. 46, pp. 305-309, 2008.

[134] D. B. Jourdan et O. L. de Weck, «Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility,» *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III,* vol. 5403, pp. 565-576, 2004.

[135] R. Rajagopalan, P. K. Varshney, C. K. Mohan et K. G. Mehrotra, «Sensor placement for energy efficient target detection in wireless sensor networks: A multi-objective optimization approach,» *Conference on information sciences and systems,* 2005.

[136] A. Raich et T. Liszkai, «Multi-objective genetic algorithm methodology for optimizing sensor layouts to enhance structural damage identification,» *Proc. of Int'l Workshop on Structural Health Monitoring,* pp. 650-657, 2003.

[137] J. Jia, J. Chen, G. Chang et Z. Tan, «Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithm,» *Computers & Mathematics with Applications,* vol. 57, n° 111-12, pp. 1756-1766, 2009.

[138] G. Molina, E. Alba et E.-G. Talbi, «Optimal Sensor Network Layout Using Multi-Objective Metaheuristics,» *J. of Universal Computer Science,* vol. 14, n° 115, pp. 2549--2565, 2008.

[139] E. Yang, A. T. Erdogan, T. Arslan et N. H. Barton, «Multi-objective evolutionary optimizations of a space-based reconfigurable sensor network under hard constraints,» *Soft Computing,* vol. 15, n° 11, pp. 25-36, 2011.

[140] M. Gunes, U. Sorges et I. Bouazizi, «ARA-the ant-colony based routing algorithm for MANETs,» *Proceedings of the International Conference on Parallel Processing Workshops,* pp. 79-85, 2002.

[141] S. Maamar, A. Lamia, G. Leila et B. Azeddine, «Etude des performances des protocoles de

routage dans les réseaux mobiles ad-hoc,» *In 4th International Conférence on Computer Integrated Manufacturing CIP'2007,* 2007.

[142] G. Di Caro et M. Dorigo, «AntNet: Distributed stigmergetic control for communications networks,» *Journal of Artificial Intelligence Research,* vol. 9, pp. 317-365, 1998.

[143] A. Mohajerani et D. Gharavian, «An ant colony optimization based routing algorithm for extending network lifetime in wireless sensor networks,» *Wireless Networks,* vol. 22, n° 18, pp. 2637-2647, 2016.

[144] T. Camilo, C. Carreto, J. S. Silva et F. Boavida, «An energy-efficient ant-based routing algorithm for wireless sensor networks,» *International Workshop on Ant Colony Optimization and Swarm Intelligence,* pp. 49--59, 2006.

[145] Y. Liu, H. Zhang, Q. Ni, Z. Zhou et G. Zhu, «An effective ant-colony based routing algorithm for mobile ad-hoc network,» *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on,* pp. 100-103, 2008.

[146] X.-M. Hu et J. Zhang, «Ant routing optimization algorithm for extending the lifetime of wireless sensor networks,» *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on,* pp. 738-744, 2010.

[147] S. Sethi et S. K. Udgata, «The efficient ant routing protocol for MANET,» *International Journal on Computer Science and Engineering,* vol. 2, n° 107, pp. 2414-2420, 2010.

[148] M. Arif et T. Rani, «ACO based routing for MANETS,» *arXiv preprint arXiv:1205.1604,* 2012.

[149] J. S. Baras et H. Mehta, «A probabilistic emergent routing algorithm for mobile ad hoc networks,» *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks,* pp. 10-pages, 2003.

[150] M. Goswami, R. Dharaskar et V. Thakare, «Fuzzy ant colony based routing protocol for mobile ad hoc network,» *Computer Engineering and Technology, 2009. ICCET'09. International Conference on,* vol. 2, pp. 438-444, 2009.

[151] D. Karaboga, «Artificial bee colony algorithm,» *scholarpedia,* vol. 5, n° 13, p. 6915, 2010.

[152] D. Karaboga et B. Basturk, «On the performance of artificial bee colony (ABC) algorithm,» *Applied soft computing,* vol. 8, n° 11, pp. 687-697, 2008.

[153] A. Ari, «Bio-inspired Solutions for Optimal Management in Wireless Sensor Networks,» Thèse de doctorat, Université Paris-Saclay, 2016.

[154] B. Akay et D. Karaboga, «Parameter tuning for the artificial bee colony algorithm,» *International Conference on Computational Collective Intelligence,* pp. 608-619, 2009.

[155] D. Karaboga et B. Basturk, «A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,» *Journal of global optimization,* vol. 39, n° 13, pp. 459-471, 2007.

[156] X. Cai, Y. Duan, Y. He et J. a. L. C. Yang, «Bee-sensor-C: an energy-efficient and scalable multipath routing protocol for wireless sensor networks,» *International Journal of Distributed Sensor Networks,* vol. 11, n° 13, p. 976127, 2015.

[157] P. Boonma et J. Suzuki, «BiSNET: A biologically-inspired middleware architecture for self-managing wireless sensor networks,» *Computer networks,* vol. 51, n° 116, pp. 4599-4616, 2007.

[158] P. Boonma et J. Suzuki, «MONSOON: A coevolutionary multiobjective adaptation framework for dynamic wireless sensor networks,» *Proceedings of the 41st Annual Hawaii International Conference on System Sciences,,* pp. 497-497, 2008.

[159] P. Boonma et J. Suzuki, «La Nina: an evolutionary noise-aware optimisation framework in self-adaptive publish/subscribe middleware for wireless sensor networks,» *International Journal of Autonomous and Adaptive Communications Systems,* vol. 4, n° 12, pp. 180-201, 2011.

[160] P. Boonma et J. Suzuki, «Accelerated evolution: a biologically-inspired approach for augmenting self-star properties in wireless sensor networks,» *Transactions on Computational Science XV,* pp. 108-129, 2012.

[161] E. Russell C et Y. Shi, «Particle swarm optimization: developments, applications and resources,» *evolutionary computation, 2001. Proceedings of the 2001 Congress on,* vol. 1, pp. 81-86, 2001.

[162] A. Dutot et D. Olivier, «Optimisation par essaim de particules Application au problème des n-Reines,» *Laboratoire Informatique du Havre, Université du Havre,* p. 8, 2002.

[163] Y. Cooren, «Perfectionnement d'un algorithme adaptatif d'optimisation par essaim particulaire: application en génie médical et en électronique,» Université Paris-Est, 2008.

[164] G. CALAS, «Optimisation par essaim particulaire,» *Une,* vol. 3, n° 13, 2009.

[165] K. Vijayalakshmi et P. Anandan, «A multi objective Tabu particle swarm optimization for effective cluster head selection in WSN,» *Cluster Computing,* pp. 1-8, 2018.

[166] A. Kaswan, V. Singh et P. K. Jana, «A novel multi-objective particle swarm optimization based energy efficient path design for mobile sink in wireless sensor networks,» *Pervasive and Mobile Computing,* 2018.

[167] C. Mendis, S. M. Guru, S. Halgamuge et S. Fernando, «Optimized sink node path using particle swarm optimization,» *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on,* vol. 2, pp. 5-pp, 2006.

[168] J. Wang, Y. Cao, B. Li, H.-j. Kim et S. Lee, «Particle swarm optimization based clustering algorithm with mobile sink for WSNs,» *Future Generation Computer Systems,* vol. 76, pp. 452-457, 2017.

[169] S. Jha et G. P. Gupta, «Energy Balanced Clustering Protocol Using Particle Swarm Optimization for Wireless Sensor Networks,» *International Conference on Information and Communication Technology for Intelligent Systems,* pp. 33-41, 2017.

[170] J. Hu, J. Song, M. Zhang et X. Kang, «Topology optimization for urban traffic sensor network,» *Tsinghua Science & Technology,* vol. 13, n° 12, pp. 229-236, 2008.

[171] J. Wang, C. Ju, H.-j. Kim, R. S. Sherratt et S. Lee, «A mobile assisted coverage hole patching scheme based on particle swarm optimization for WSNs,» *Cluster Computing,* pp. 1-9, 2017.

[172] P. N. Ngatchou, W. L. Fox et M. A. El-Sharkawi, «Distributed sensor placement with sequential particle swarm optimization,» *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE,* pp. 385-388, 2005.

[173] A. Gopakumar et L. Jacob, «Localization in wireless sensor networks using particle swarm optimization,» *Proceedings of the IET International Conference on Wireless, Mobile and Multimedia Networks,* pp. 227-230, 2008.

[174] R. V. Kulkarni, G. K. Venayagamoorthy et M. X. Cheng, «Bio-inspired node localization in wireless sensor networks,» *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on,* pp. 205-210, 2009.

[175] C. A. Richmond, «Fireflies flashing in unison,» *Science,* vol. 71, n° 11847, pp. 537-538, 1930.

[176] R. E. Mirollo et S. H. Strogatz, «Synchronization of pulse-coupled biological oscillators,» *SIAM Journal on Applied Mathematics,* vol. 50, n° 16, pp. 1645--1662, 1990.

[177] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh et R. Nagpal, «Firefly-inspired sensor network synchronicity with realistic radio effects,» *Proceedings of the 3rd international conference on Embedded networked sensor systems,* pp. 142-153, 2005.

[178] L. Cui et H. Wang, «Reachback firefly synchronicity with late sensitivity window in wireless sensor networks,» *Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on,* pp. 451-456, 2009.

[179] Y. Sun, Q. Jiang et K. Zhang, «A clustering scheme for Reachback firefly synchronicity in wireless sensor networks,» *Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on,* pp. 27-31, 2012.

[180] C. Hao, P. Song, C. Yang et X. Liu, «Testing a Firefly-Inspired Synchronization Algorithm in a Complex Wireless Sensor Network,» *Sensors,* vol. 17, n° 13, p. 544, 2017.

[181] T. Bokareva, N. Bulusu et S. Jha, «Sasha: Toward a self-healing hybrid sensor network architecture,» *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on,* pp. 71-78, 2005.

[182] S. Sarafijanovic et J.-Y. Le Boudec, «An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal, and memory detectors,» *International Conference on Artificial Immune Systems,* pp. 342-356, 2004.

[183] S. Sarafijanovic et J.-Y. Le Boudec, «An artificial immune system approach with secondary response for misbehavior detection in mobile ad hoc networks,» *IEEE Transactions on Neural Networks,* vol. 16, n° 15, pp. 1076-1087, 2005.

[184] B. Atakan et O. B. Akan, «Immune system based distributed node and rate selection in wireless sensor networks,» *Bio-Inspired Models of Network, Information and Computing Systems, 2006. 1st,* pp. 1-8, 2006.

[185] S. Mohapatra et P. M. Khilar, «Artificial immune system based fault diagnosis in large wireless sensor network topology,» *Region 10 Conference, TENCON 2017-2017 IEEE,* pp. 2687-2692, 2017.

[186] S. Jamali et R. Fotohi, «DAWA: Defending against wormhole attack in MANETs by using fuzzy logic and artificial immune system,» *The Journal of Supercomputing,* vol. 73, n° 112, pp. 5173-5196, 2017.

[187] A. Lindenmayer, «Mathematical models for cellular interactions in development I. Filaments with one-sided inputs,» *Journal of theoretical biology,* vol. 18, n° 13, pp. 280-299, 1968.

[188] V. Ponnusamy, A. Hudaya et A. G. Downe, «A biologically inspired energy efficient intrusion detection system,» *Computer & Information Science (ICCIS), 2012 International Conference on,* vol. 2, pp. 729-735, 2012.

[189] J. S. Kumar et E. B. Raj, «Genetic algorithm based multicast routing in wireless sensor networks—A research framework,» *International Journal of Engineering and Innovative Technology (IJEIT) ,* vol. 2, pp. 240-246, 2012.

[190] A. P. Bhondekar, R. Vig, M. L. Singla, C. Ghanshyam et P. Kapur, «Genetic algorithm based node placement methodology for wireless sensor networks,» *Proceedings of the international multiconference of engineers and computer scientists,* vol. 1, pp. 18-20, 2009.

[191] K. P. Ferentinos et T. A. Tsiligiridis, «Adaptive design optimization of wireless sensor networks using genetic algorithms,» *Computer Networks,* vol. 51, n° 14, pp. 1031-1051, 2007.

[192] D. B. Jourdan et O. L. de Weck, «Layout optimization for a wireless sensor network using a multi-objective genetic algorithm,» *Vehicular technology conference, 2004. VTC 2004-Spring. 2004 IEEE 59th,* pp. 2466-2470, 2004.

[193] J. Zhou, Q. Cao, C. Li et R. Huang, «A genetic algorithm based on extended sequence and topology encoding for the multicast protocol in two-tiered WSN,» *Expert Systems with Applications,* vol. 37, n° 12, pp. 1684-1695, 2010.

[194] S. Yang, H. Cheng et F. Wang, «Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks,» *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 40, n° 11, pp. 52-63, 2010.

[195] X.-M. Hu, J. Zhang, Y. Yu, H. S.-H. Chung, Y.-L. Li, Y.-H. Shi et X.-N. Luo, «Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks,» *IEEE transactions on evolutionary computation,* vol. 14, n° 15, pp. 766-781, 2010.

[196] D. Karaboga et B. Basturk, «Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems,» *LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing,* vol. 4529, pp. 789-798, 2007.

[197] D. Karaboga et B. Akay, «A modified artificial bee colony (ABC) algorithm for constrained optimization problems,» *Applied soft computing,* vol. 11, n° 13, pp. 3021-3031, 2011.

[198] N. Bacanin et M. Tuba, «Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators,» *Studies in Informatics and Control,* vol. 21, n° 12, pp. 137-146, 2012.

[199] D. Karaboga et C. Ozturk, «A novel clustering approach: Artificial Bee Colony (ABC) algorithm,» *Applied soft computing,* vol. 11, n° 11, pp. 652-657, 2011.

[200] B. Akay et D. Karaboga, «Solving integer programming problems by using artificial bee colony algorithm,» *Congress of the Italian Association for Artificial Intelligence,* pp. 355-364, 2009.

[201] T.-P. Hong et G.-N. Shiu, «Allocating multiple base stations under general power consumption by the particle swarm optimization,» *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE,* pp.

23-28, 2007.

[202] T. Wimalajeewa et S. K. Jayaweera, «Optimal power scheduling for correlated data fusion in wireless sensor networks via constrained PSO,» *IEEE Transactions on Wireless Communications,* vol. 7, n° 18, pp. 3608-3618, 2008.

[203] H. Guo, K.-S. Low et H.-A. Nguyen, «Optimizing the localization of a wireless sensor network in real time based on a low-cost microcontroller,» *IEEE Transactions on Industrial Electronics,* vol. 58, n° 13, pp. 741-749, 2011.

[204] K. Low, H. Nguyen et H. Guo, «Optimization of sensor node locations in a wireless sensor network,» *Natural Computation, 2008. ICNC'08. Fourth International Conference on,* vol. 5, pp. 286-290, 2008.

[205] J. Li, K. Li et W. Zhu, «Improving sensing coverage of wireless sensor networks by employing mobile robots,» *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on,* pp. 899-903, 2007.

[206] X. Wang, S. Wang et J.-J. Ma, «An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment,» *Sensors,* vol. 7, n° 13, pp. 354-370, 2007.

[207] K. K. Veeramachaneni et L. A. Osadciw, «Dynamic sensor management using multi-objective particle swarm optimizer,» *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2004,* vol. 5434, pp. 205-217, 2004.

[208] A. Rowe, D. Goel et R. Rajkumar, «Firefly mosaic: A vision-enabled wireless sensor networking system,» *Real-time systems symposium, 2007. RTSS 2007. 28th IEEE international,* pp. 459-468, 2007.

[209] S. Nandy, M. Karmakar, P. Sarkar, A. Das, A. Abraham et D. Paul, «Agent based adaptive firefly back-propagation neural network training method for dynamic systems,» *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on,* pp. 449-454, 2012.

[210] R. Falcon, X. Li, A. Nayak et I. Stojmenovic, «A harmony-seeking firefly swarm to the periodic replacement of damaged sensors by a team of mobile robots,» *Communications (ICC), 2012 IEEE International Conference on,* pp. 4914-4918, 2012.

[211] N. Pari, A. Kailas et M. Nogueira, «Bio-inspired time synchronization for cognitive radio ad hoc networks,» *Globecom Workshops (GC Wkshps), 2012 IEEE,* pp. 980-985, 2012.

[212] X. Liu et S. Zhou, «Evaluation of several time synchronization protocols in WSN,» *Information Science and Management Engineering (ISME), 2010 International Conference of,* vol. 1, pp. 488-491, 2010.

[213] A. Tyrrell, G. Auer et C. Bettstetter, «Fireflies as role models for synchronization in ad hoc networks,» *Proceedings of the 1st international conference on Bio-inspired models of network, information and computing systems,* pp. 1-7, 2006.

[214] S. Merkel, C. W. Becker et H. Schmeck, «Firefly-inspired synchronization for energy-efficient distance estimation in mobile ad-hoc networks,» *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International,* pp. 205-214, 2012.

[215] A. Tyrrell et G. Auer, «Imposing a reference timing onto firefly synchronization in wireless networks,» *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th,* pp. 222-226, 2007.

[216] J. Yackovich, D. Mosse, A. Rowe et R. Rajkumar, «Making wsn tdma practical: Stealing slots up and down the tree,» *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on,* vol. 1, pp. 41-50, 2011.

[217] D. DasGupta, «An overview of artificial immune systems and their applications,» *Artificial immune systems and their applications,* pp. 3-21, 1993.

[218] B. Atakan et O. B. Akan, «Immune system-based energy efficient and reliable communication in wireless sensor networks,» *Advances in Biologically Inspired Information Systems,* pp. 187-207, 2007.

[219] T. Lim, H. Lau, J. Timmis et I. Bate, «Immune-inspired self healing in wireless sensor networks,» *International Conference on Artificial Immune Systems,* pp. 42-56, 2012.

[220] K. Saleem et N. Fisal, «Bio-inspired self-organized secure autonomous routing protocol for

WSN,» *RF and Microwave Conference, 2008. RFM 2008. IEEE International,* pp. 417-421, 2008.

[221] A. Nikdel, S. M. Jameii et H. Noori, «A novel scheduling mechanism based on artificial immune system for communication between cluster head and cluster members in WSNs,» *International Journal of Information and Electronics Engineering,* vol. 2, n° 13, p. 333, 2012.

[222] O. Engin et A. Doyen, «A new approach to solve hybrid flow shop scheduling problems by artificial immune system,» *Future generation computer systems,* vol. 20, n° 16, pp. 1083-1095, 2004.

[223] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, P. Lio et K.-W. Lee, «Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment,» *Ad Hoc Networks,* vol. 7, n° 14, pp. 725-741, 2009.

[224] V. Pappas, D. Verma, B.-J. Ko et A. Swami, «A circulatory system approach for wireless sensor networks,» *Ad Hoc Networks,* vol. 7, n° 14, pp. 706-724, 2009.

[225] N. Haddadou, A. Rachedi et Y. Ghamri-Doudane, «Advanced diffusion of classified data in vehicular sensor networks,» *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International,* pp. 777-782, 2011.

[226] T. Kontos, E. Zaimidis, C. Anagnostopoulos, S. Hadjiefthymiades et E. Zervas, «An adaptive epidemic information dissemination scheme with cross-layer enhancements,» *Computers and Communications (ISCC), 2011 IEEE Symposium on,* pp. 230-235, 2011.

[227] J. Yoo, S. Choi et C.-K. Kim, «The capacity of epidemic routing in vehicular networks,» *IEEE Communications Letters,* vol. 13, n° 16, pp. 459-461, 2009.

[228] J. Xue, J. Li, Y. Cao et J. Fang, «Advanced PROPHET routing in delay tolerant network,» *Communication Software and Networks, 2009. ICCSN'09. International Conference on,* pp. 411-413, 2009.

[229] P. Hui, J. Crowcroft et E. Yoneki, «Bubble rap: Social-based forwarding in delay-tolerant networks,» *IEEE Transactions on Mobile Computing,* vol. 10, n° 111, pp. 1576-1589, 2011.

[230] K. A. Harras, K. C. Almeroth et E. M. Belding-Royer, «Delay tolerant mobile networks (dtmns): Controlled flooding in sparse mobile networks,» *International Conference on Research in Networking,* pp. 1180-1192, 2005.

[231] G. Theodorakopoulos, J.-Y. Le Boudec et J. S. Baras, «Selfish response to epidemic propagation,» *IEEE Transactions on Automatic Control,* vol. 58, n° 12, pp. 363-376, 2013.

[232] R. V. Kulkarni et G. K. Venayagamoorthy, «Neural network based secure media access control protocol for wireless sensor networks,» *Neural Networks, 2009. IJCNN 2009. International Joint Conference on,* pp. 1680-1687, 2009.

[233] R. V. Kulkarni, G. K. Venayagamoorthy, A. V. Thakur et S. K. Madria, «Generalized neuron based secure media access control protocol for wireless sensor networks,» *Computational intelligence in miulti-criteria decision-making, 2009. mcdm'09. ieee symposium on,* pp. 16-22, 2009.

[234] J. Barbancho, C. Leon, J. Molina et A. Barbancho, «Giving neurons to sensors. QoS management in wireless sensors networks.,» *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on,* pp. 594-597, 2006.

[235] J. Podpora, L. Reznik et G. Von Pless, «Intelligent real-time adaptation for power efficiency in sensor networks,» *IEEE Sensors Journal,* vol. 8, n° 112, pp. 2066--2073, 2008.

[236] A. I. Moustapha et R. R. Selmic, «Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection,» *IEEE Transactions on Instrumentation and Measurement,* vol. 57, n° 15, pp. 981-988, 2008.

[237] H. He, Z. Zhu et E. Makinen, «A neural network model to minimize the connected dominating set for self-configuration of wireless sensor networks,» *IEEE Transactions on Neural Networks,* vol. 20, n° 16, pp. 973-982, 2009.

[238] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen et M. Welsh, «Simulating the power consumption of large-scale sensor network applications,» *Proceedings of the 2nd international conference on Embedded networked sensor systems,* pp. 188-200, 2004.

[239] K. Barka, A. Bilami et S. Gourdache, «MONet: A framework for self-adaptive energy-aware middleware for dynamic wireless sensor network,» *International Journal of Pervasive Computing and Communications,* vol. 13, n° 14, pp. 345-369, 2017.

[240] M. Wooldridge, An introduction to multiagent systems, John Wiley & Sons, 2009.

[241] S. J. Russell et P. Norvig, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited,, 2016.

[242] J. Ferber, «Les systèmes multi-agents: un aperçu général,» *Techniques et sciences informatiques,* vol. 16, n° 18, 1997.

[243] N. R. Jennings, «On agent-based software engineering,» *Artificial intelligence,* vol. 117, n° 12, pp. 277-296, 2000.

[244] S. FWL, «A scent organ in the bee,» *British Bee Journal,* p. 142, 1901.

[245] R. Morse et R. Boch, «Pheromone concert in swarming honey bees (Hymenoptera: Apidae),» *Annals of the Entomological Society of America,* vol. 64, n° 16, pp. 1414-1417, 1971.

[246] R. Boch, D. Shearer et B. Stone, «Identification of iso-amyl acetate as an active component in the sting pheromone of the honey bee,» *Nature,* vol. 195, n° 14845, pp. 1018-1020, 1962.

[247] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer et others, «TinyOS: An operating system for sensor networks,» *Ambient intelligence,* pp. 115-148, 2005.

[248] D. He, G. Mujica, J. Portilla et T. Riesgo, «Modelling and planning reliable wireless sensor networks based on multi-objective optimization genetic algorithm with changeable length,» *Journal of Heuristics,* vol. 21, n° 12, pp. 257-300, 2015.

[249] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63, Citeseer, 1999.

[250] K. Deb, A. Pratap, S. Agarwal et T. Meyarivan, «A fast and elitist multiobjective genetic algorithm: NSGA-II,» *IEEE transactions on evolutionary computation,* vol. 6, n° 12, pp. 182-197, 2002.

[251] E. Zitzler, K. Deb et L. Thiele, «Comparison of multiobjective evolutionary algorithms: Empirical results,» *Evolutionary computation,* vol. 8, n° 12, pp. 173-195, 2000.

[252] D. W. Corne, J. D. Knowles et M. J. Oates, «The Pareto envelope-based selection algorithm for multiobjective optimization,» *International conference on parallel problem solving from nature,* pp. 839-848, 2000.

[253] R. Storn et K. Price, «Differential evolution--a simple and efficient heuristic for global optimization over continuous spaces,» *Journal of global optimization,* vol. 11, n° 14, pp. 341-359, 1997.

[254] J. rownlee, Clever algorithms: nature-inspired programming recipes, Jason Brownlee, 2011.

[255] Y. Collette et P. Siarry, Multiobjective optimization: principles and case studies, Springer Science & Business Media, 2013.

[256] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer et D. Culler, «The nesC language: A holistic approach to networked embedded systems,» *Acm Sigplan Notices,* vol. 49, n° 14, pp. 41-51, 2014.

Une plateforme Middleware pour l'auto-adaptation des réseaux de capteurs sans fil hétérogène

Résumé

Au cours de ces dernières années, les réseaux de capteurs sans fil ont connu un intérêt croissant à la fois au sein de la communauté scientifique et industrielle en raison du large potentiel en termes d'applications offertes. Toutefois, les capteurs sont conçus avec d'extrêmes contraintes en ressources, en particulier la limitation de l'énergie. Nous avons récemment constaté l'apparition des nouvelles applications où nous avons besoin d'un nouveau composant du réseau appelé actionneur « super capteur ». Ces actionneurs disposent généralement d'une source d'énergie abondante, par conséquent ils sont riches en ressources (la capacité du traitement, du stockage, de puissance d'émission, etc…), ce type de réseau appelé réseau sans fil de capteurs et actionneurs (RCASF) se considère comme étant un réseau hétérogène. L'hétérogénéité est causée par la coexistence des nœuds capteurs à faibles ressources et des nœuds actionneurs riches en ressources. C'est dans ce contexte que se déroule cette thèse dans laquelle nous avons proposé des mécanismes d'auto-gestion et d'auto-contrôler s'inspirant d'un système biologie et aussi s'appuyant sur l'hétérogénéité.

Au début, nous proposons un nouveau Framework middleware bio-inspiré, qui assure l'auto-gestion d'un RCSF homogènes, appelé MONet. Ce framework basé sur des agents mobiles et une approche bio-inspirée hybride (l'intelligence en essaim SI et les algorithmes génétique GA) dans lequel les agents acheminent les données captées vers la SB et leurs architectures sont optimisées à travers un GA afin d'améliorer leurs performances, à noter aussi que les comportements de ces agents, sont inspirés d'une colonie d'abeilles.

Par la suite, nous avons proposé aussi deux solutions pour la réduction de l'émission de phéromones par l'agent pendant son déplacement sur les nœuds, dont la première basée sur la caractéristique de l'égoïsme de l'agent et la deuxième basée sur un seuil d'émission. Les résultats des simulations montrent que la durée de vie du réseau est étendue de plus du tiers.

Finalement, nous nous sommes intéressés à l'hétérogénéité dans les RCSF à savoir RCASF (WSAN). Se basant sur l'idée que les ressources au niveau des nœuds actionneurs doivent être pleinement exploitées afin de réduire la charge de communication au niveau des nœuds capteurs, nous avons proposé une extension de MONet appelée BISSA. BISSA permet de profiter de la puissance d'émission et de la capacité de stockage des actionneurs pour améliorer les performances du réseau. Comparée au MONet, cette extension réalise une prolongation de la durée de vie du réseau d'environ deux tiers, tout en améliorant les performances du réseau de plus de la moitié.

***Mots clés** : l'autonomie, les solutions bio-inspirées, MONet, BISSA, RCSF, RCASF (WSAN).*

Abstract

During the past few years, wireless sensor networks witnessed an increased interest in both the industrial and the scientific community due to the potential wide area of applications. However, sensors' components are designed with extreme resource constraints, especially the power supply limitation. We have recently seen the emergence of new applications where we need a new component of the network called actuator "Super Sensor". These actuator nodes are resource-rich devices with higher processing capabilities, transmission power and larger battery capacity. This type of network called Wireless Sensor and Actuator Networks (WSAN). this is a heterogeneous network. This heterogeneity is caused by the coexistence of sensor nodes with limited resources and actuator nodes with higher resources. It is in this context that this thesis takes place in which we proposed self-management and self-control mechanisms through biologically inspired computing and also based on heterogeneity.

first, we propose a new bio-inspired middleware framework, which ensures the self-management of a homogeneous WSN, called MONet. This framework is based on mobile agents (software agent) and a hybrid bio-inspired approach (i.e., swarm intelligence IS and genetic algorithm GA) in which the agents collect sensor data or detect an event on individual nodes, and carry sensor data to base stations. their architectures are optimized within a genetic algorithm to improve their performance, as well as the behaviors of these agents, they were inspired by an artificial Bee Colony

Secondly, we also proposed two solutions for reducing the pheromone emission by the agent during its migration on the nodes whose the first based on the selfishness characteristic of the agent and the second based on an emission threshold to control the level of cooperation among them. Simulation results show that the network lifetime is extended by more than one third.

Finally, we were interested in the heterogeneity in the WSN namely WSAN. Based on the idea that resource-rich nodes must be exploited to reduce the communication load level on low-power nodes, we proposed an extension of MONet called BISSA. BISSA uses the large transmit power and storage capacity of actuators to provide network performances. Compared to MONet, this extension extends the network lifetime about two-thirds, while improving network performances by more than half.

***Keywords:** Autonomy, bio-inspired solutions, MONet, BISSA, WSN, WSAN.*