République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB – Blida 1

Faculté des Sciences

Département d'Informatique

Mémoire du Master

Option : Ingénierie ***des Logiciels (IL)***

Thème :

Méta-heuristiques pour l'Optimisation du seuil dans une Politique d'Economie d'Energie dans les Capteurs Sans-Fil

Présenté par :

BOUKADER Younes.

KERROUCHI Zakaria.

Présentée et soutenue le…. Juillet 2023 devant le jury composé :

Dr. Toubaline Nesrine, Présidente. Université de Blida 1

Dr. Tobji Rachida, Examinatrice. Université de Blida 1

M^{me}. Boutoumi Bachira, Promotrice. Université Blida 1

M^{me}. Aroussi Sanaa, Promotrice. Université Blida 1

2022/2023

Remerciements

Tout d'abord, nous remercions dieu le tout puissant de nous avoir donné la force et la volonté d'entamer et de terminer ce mémoire.

Un grand merci à toutes nos familles surtout nos parents pour leur encouragement et leur suivi avec patience du déroulement de notre projet.

Nous tenons à remercier vivement nos encadreurs Mme Boutoumi Bachira et Mme Aroussi Sanaa pour leurs encadrements et leurs précieux conseils tout au long de ce travail. Leurs disponibilités, leurs expertises et leurs bienveillances ont grandement facilité notre progression, et nous lui sommes très reconnaissants.

Nous exprimons notre grand respect également aux membres du jury d'avoir accepté d'examiner et dévaluer ce travail.

Enfin, nous remercions tous nos amis Ahmed et Yazid qui nous partagé des moments d'entraides, de faiblesse, de souffrance et de joie et pour leur soutien indéfectible. Leur présence à nos côtés a été d'un grand réconfort et nous leur en sommes infiniment reconnaissants.

Merci infiniment

DEDICACE

C'est avec profonde gratitude et sincères mots, que nous dédions ce modeste travail de fin d'étude à nos chers parents, qui ont sacrifié leur vie pour notre réussite et nous ont éclairé le chemin par leurs conseils judicieux.

Nous espérons qu'un jour, nous pourrons leurs rendre un peu de ce qu'ils ont fait pour nous, que dieu leur prête bonheur et longue vie. Nous dédions aussi ce travail à nos frères et sœurs, nos familles, nos amis,

Tous nos professeurs qui nous ont enseigner et à tous ceux qui nous sont chers.

Younes & Zakaria

Résumé

Afin de minimiser la consommation d'énergie dans un Réseau de Capteurs Sans Fil (RCSF), plusieurs techniques de modélisation formelles ont été proposées dans la littérature, notamment la technique basée sur les modèles de files d'attente avec vacance et la politique N-vacance (N-Policy). La politique N-vacance, qui représente une approche de conception efficace pour diminuer la consommation d'énergie du nœud capteur, introduit un délai d'attente supplémentaires pour les paquets de données en attente dans le cas où le taux d'arriver est petit. Pour valider cette technique, nous utilisons généralement une heuristique aléatoire qui donne des valeurs aléatoires à ses paramètres de configuration tel que le seuil (N), la taille de buffer (K), le taux d'arrivé (λ) et le taux moyen de service (μ) des paquets. Cependant, cette heuristique aléatoire prend beaucoup temps et ne retourne pas des meilleurs résultats. Pour cela, nous proposons d'appliquer deux méta-heuristiques AG (Algorithme Génétique) et OEP (Optimisation par Essaim des Particules) pour résoudre ce problème d'optimisation multi-objectif. Ce dernier consiste à trouver les valeurs optimales S* = (N*, K*, λ*, μ*) permettant de minimiser la consommation d'énergie ainsi que de délai. Les résultats expérimentaux obtenus semblent prometteurs.

Mots clés : Réseaux de capteurs sans fil (RCSF), Économie d'énergie, N-vacance, optimisation multi-objectif, Seuil, Heuristique aléatoire, Méta-heuristiques, algorithme génétique (AG), Optimisation par essaim des particules (OEP).

Abstract

In order to minimize energy consumption in a Wireless Sensor Network (WSN), several formal modeling techniques have been proposed in the literature, notably the technique based on vacation queuing models with N-policy. The N-policy, which represents an efficient design approach for minimizing the energy consumption of the sensor node, introduces an additional delay for waiting data packets in the case where the arrival rate is small. To validate this technique, we typically use a random heuristic that gives random values to its configuration parameters such as threshold (N), buffer size (K), arrival rate (λ) and average packet service rate (μ). However, this random heuristic is time-consuming and does not return the best results. For this reason, we propose to apply two meta-heuristics GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) to solve this multi-objective optimization problem. The latter consists in finding the optimal values S* = (N*, K*, λ*, μ*) allowing to minimize energy consumption as well as latency. The experimental results obtained look promising.

Keywords: Wireless Sensor Networks (WSN), Energy saving, N-vacancy, multi-objective optimization, Threshold, Random heuristics, Meta-heuristics, Genetic Algorithm (GA), Particle Swarm Optimization (PSO).

ملخص

من أجل تقليل استهلاك الطاقة في شبكة الاستشعار اللاسلكية (RCSF)، تم اقتراح العديد من تقنيات النمذجة الرسمية في الأدبيات، على وجه الخصوص التقنية القائمة على نماذج قوائم الانتظار مع الشواغر (تقنية نمط طابور الشواغر) وسياسة N-Policy. تقدم سياسة N-Policy ، التي تمثل نهج تصميم فعال لتقليل استهلاك الطاقة لعقدة المستشعر مهلة (فترة انتظار) إضافية لانتظار حزم البيانات المعلقة في حال كان معدل الوصول صغيرا .للتحقق من صحة هذه التقنية، نستخدم عادة استدلالًا عشوائيًا يعطي قيمًا عشوائية لمعلمات التكوين الخاصة به مثل العتبة (N) وحجم المخزن المؤقت (K) ومعدل الوصول (λ) ومتوسط معدل الخدمة (μ) للحزم. ومع ذلك، فإن هذا الاستدلال العشوائي يستغرق وقتا طويلا ولا يعود بنتائج أفضل. لهذا، نقترح تطبيق اثنين من الاستدلال التلوي (AG) الخوارزمية الجينية و (PSO) تحسين سرب الجسيمات لحل مشكلة التحسين متعددة الأهداف هذه. يتكون الأخير من إيجاد القيم المثلى ($S^* = (N^*, K^*, \lambda^*, \mu^*)$) لتقليل استهلاك الطاقة وكذلك التأخير. النتائج التجريبية التي تم الحصول عليها تبدو واعدة.

الكلمات المفتاحية :شبكات الاستشعار اللاسلكية (RCSF)، توفير الطاقة، N-Policy، تحسين متعدد الأهداف، العتبة، الاستدلال العشوائي، الاستدلال التلوي، الخوارزمية الجينية (GA)، تحسين سرب الجسيمات (OEP).

Table des matières

Table des figures

Listes des tableaux

Listes des algorithmes

Listes des abréviations

AG : Algorithmes Génétiques.

CTMC : Chaine de Markov à Temps Continu.

EA: Algorithms Evolutionnaires.

FPGA: Field programmable Gate Array.

GPS: Global Positioning System.

HA: Heuristique Aléatoire.

LEACH: Low-Energy Adaptive Clustering Hierarchy.

MAC: Medium Access Control.

N-Policy : N-vacance.

OEP : Optimisation par Essaim de Particules

RAM: Random Access Memory.

ROM: Read Only Memory.

RSCF : Réseau Capteurs Sans Fil.

SB: Stations de Base.

WSN: Wireless Sensor Network.

FPGA: Field programmable Gate Array.

RTS: Request to Send.

CTS: Clear to Send.

SYNC: Synchronization.

ACK: Acknowledge.

INTRODUCTION GENERALE

La rénovation des technologies microélectroniques et de communication sans fil a engendré l'apparition de dispositifs miniaturisés dit nœuds capteurs, ces derniers ont l'aptitude de détecter des évènements qui se produisent dans leur zone de déploiement. L'ensemble de ces nœuds forme ce qu'on appelle le Réseau de Capteurs Sans Fil (RCSF) ou Wireless Sensor Network (WSN) qui a fait naissance à plusieurs applications.

Les RCSF suscitaient beaucoup d'enthousiasme dans la recherche scientifique en raison des ressources limités que disposent les capteurs en termes de puissance de transmission, de capacité de traitement, de stockage des données et d'énergie. D'où la préoccupation majeure est de prolonger la durée de vie du réseau à travers une bonne stratégie d'économie d'énergie. Notamment si l'endroit de surveillance est hostile car le rechargement des sources d'énergie des nœuds (généralement des batteries) après leur épuisement devient impossible.

Afin de minimiser la consommation d'énergie dans les RCSFs nœuds capteur, plusieurs techniques sont utilisées [1] [2] [3] . La technique des files d'attente avec vacance et la politique N-vacance (N-Policy) ou bien la politique à seuil est une technique utilisée pour économiser l'énergie du nœud capteur en réduisant le nombre de transition entre l'état oisif (idle) et l'état occupé (busy) [1] [2] [3]. La N-vacance contrôle le passage du nœud capteur de l'état oisif à l'état occupé. Le nœud qui est initialement dans l'état oisif passe à l'état occupé une fois le nombre de paquets en attente dans le buffer atteint le seuil N. Durant l'état occupé le nœud fait la transmission des paquets exhaustivement.

Pour valider cette technique, plusieurs tests ont été faits dans [3], en donnant des valeurs aléatoires aux paramètres de configuration tel que le seuil (N), la taille de buffer (K), le taux d'arrivé (λ) et le taux moyen de service (μ). Cependant, cette heuristique aléatoire prend beaucoup de temps et ne retourne pas des meilleurs résultats. C'est dans ce contexte que s'inscrit notre problématique qui consiste à utiliser d'autres méthodes approchées (notamment les méta-heuristiques) pour résoudre ce problème d'optimisation. Ce dernier est composé des variables de décision (N, K, λ, μ) dans le but est de trouver les valeurs optimales (la meilleure solution) $S^* = (N^*, K^*, \lambda^*, \mu^*)$ permettant de minimiser la consommation d'énergie ainsi que le délai dans les RCSF. En effet, cette politique N-vacance efficace en énergie, augmente considérablement le délai d'attente des paquets en attente dans le buffer dans les réseaux multi-sauts, parce que le délai d'attente augment avec l'augmentation du seuil N [1] [2] [3]. Pour cela, nous devons optimiser (minimiser) une fonction objective à deux critères (multi-objectif).

L'objectif principal du projet de fin d'étude est de concevoir une solution basée sur les méta-heuristique pour trouver les valeurs optimales des paramètres de configuration (N, K, λ, μ) de la techniques N-vacance qui minimise à la fois la consommation d'énergie et le délai dans les RCSFs, Nous allons appliquer deux méta-heuristique à base de population l'Algorithme Génétique (Genetic Algorithm AG) et l'optimisation par Essaim de Particules (Particle Swarm Optimization PSO) pour résoudre ce problème d'optimisation multi-objectif.

Le présent mémoire est organisé comme suit :

- Le premier chapitre qui représente notre état de l'art, est divisé en deux parties :
- La première partie est consacrée aux généralités sur les réseaux de capteurs sans fil comme par exemple la définition de ce domaine, la consommation d'énergie des capteurs, etc.
- La deuxième partie est dédiée aux différents concepts de bases liées aux problèmes d'optimisation et les méthodes de résolution des problèmes d'optimisations.
- Le deuxième chapitre s'intéresse à la politique des files d'attentes avec vacances pour économiser l'énergie dans les RCSF et à l'application des méta-heuristiques AG et PSO pour résoudre le problème d'optimisation associé.
- Le dernier chapitre est consacré à l'implémentation de notre solution et aux résultats expérimentaux obtenus.
- Enfin, nous terminons par une conclusion générale et quelques perspectives.

CHAPITRE 1 : ETAT DE L’ART

1.1 Introduction

Ce premier chapitre intitulé « état de l'art » résume les principales terminologies, concepts et notions liées à notre projet. Nous suggérons de le diviser en deux parties. La première partie est dédiée aux réseaux des capteurs sans fils et leurs problèmes, en particulier la consommation d'énergie. La deuxième partie présente les problèmes d'optimisation et leurs méthodes de résolution notamment les méta-heuristiques

---

Partie 1 : Généralités Sur les Réseaux de Capteurs Sans Fil

Un capteur sans fil est un petit dispositif électronique, capable de mesurer une valeur physique environnementale (température, lumière, pression, humidité, vibration, etc.), et de la communiquer à un centre de contrôle via une station de base. Chaque capteur assure les trois principales fonctions de base qui sont : l'acquisition de données, les traitements sur ces données et leurs communications aux stations de bases. [4]

Un nœud de capteur peut jouer le rôle de capteur et de routeur au même temps. Dans ce dernier cas, un nœud peut recevoir des données venant des autres nœuds et peut les retransmettre vers d'autres nœuds, avec ses propres données donc chaque capteur également appelé "nœud" est doté d'un circuit radio lui permettant de transmettre et de recevoir des informations via un médium sans fil [5]

1.2 Les composants d'un nœud capteur

Un nœud capteur est composé de plusieurs éléments ou modules correspondant chacun à une tâche particulière d'acquisition, de traitement, ou de transmission de données. Il comprend également une source d'énergie. Comme illustré dans la figure 1, un capteur est composé de quatre unités de base [6] :

- Unité d'acquisition : C'est une unité de capture ou d'acquisition des grandeurs physique tels que la température, humidité, vibrations, luminosité, etc. Cette unité contient un convertisseur (appelée CAN pour Convertisseur Analogique Numérique) qui transforme les données analogiques du capteur en grandeurs numériques compréhensibles par l'unité de traitement.

- Unité de traitement : Elle est composée de deux interfaces, une interface pour l'unité d'acquisition et une interface pour l'unité de communication. Cette dernière est le contrôleur principal du nœud capteur sans fil. Elle est généralement associée à des unités de stockage de type RAM (Random Access Memory), ROM (Read Only Memory) ou flash. Le choix de ces mémoires (taille et type) varie selon la plateforme utilisée et le type d'application du nœud capteur. Cette unité au point de vue matériel peut être de type microcontrôleur,

microprocesseur, ou encore FPGA (Field programmable GateArray) ou les deux à la fois et pour le point de vue logiciel l'unité de traitement intègre toutes les applications spécialement conçues pour le nœud capteur pour piloter les différents modules matériels. [6]

- **Unité de communication :** Cette unité est responsable de la connexion du nœud capteur au réseau afin qu'il puisse communiquer avec les autres nœuds via un médium sans fil. Elle peut être de type optique, ultrasons ou radiofréquence. En effet, le rôle de cette unité est de recevoir et transmettre les données.

– Batterie : Alimente les unités citées précédemment, et il existe des capteurs qui sont dotés d'autres composants additionnels : les systèmes de localisation comme (Global Positioning System) et un mobilisateur lui permettant le déplacement.

On peut trouver d'autres composants additionnels qui peuvent être implantés dans un capteur, on cite parmi eux :

- **Un système de localisation :**

Le nœud capteur peut être équipé d'un système de localisation spécifique (GPS pour les réseaux extérieurs). Il peut dans ce cas se localiser de manière autonome. Un nœud peut aussi se localiser au sein de son réseau par rapport à des balises.

- **Un système reconstitution de l'énergie :**

Il sert à reconstitue l'énergie consommée par un réapprovisionnement (grâce à une source externe : cellules solaires, température, vibration, etc.). C'est une technique de récupération d'énergie par l'énergie thermique, cinétique, etc. cette technique est utilisée pour des applications de réseau qui nécessitent une longue vie.

- **Un système de mobilité de nœud :**

Pour qu'un nœud puisse se déplacer dans le réseau, si ce dernier n'est pas rattaché à un appareil mobile, il lui faut un système de déplacement. Ce dernier permet de déplacer les nœuds pour accomplir ses tâches.

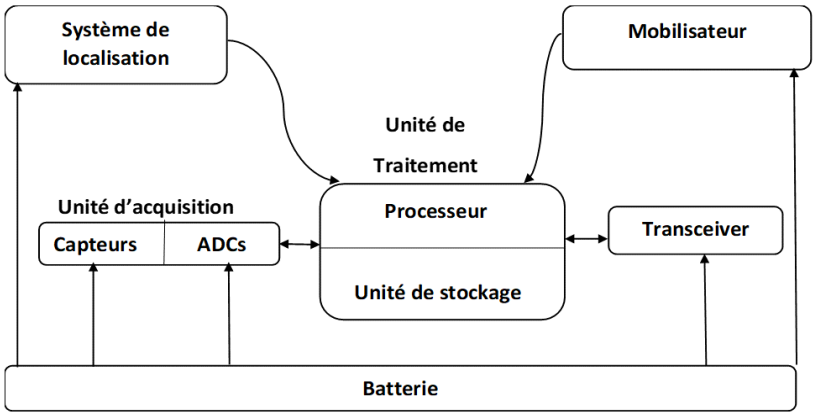


Figure 1 Architecture physique d'un nœud capteur [7]

1.3 Réseaux de Capteurs Sans Fil (RCSF) :

Les Réseaux de Capteurs Sans Fil (RCSF ou WSN "Wireless Sensor Network") résulte d'une fusion de deux pôles de l'informatique moderne : les systèmes embarqués et les communications sans fil. Un RCSF est composé d'un ensemble d'unités de traitements embarquées, communiquant via des liens sans fil. Le but général d'un RCSF est la collecte d'un ensemble de paramètres de l'environnement, telles que la température ou la pression de l'atmosphère, afin de les acheminer vers des points de traitement [8].

Comme représentés dans la figure 2, les réseaux de capteurs sans fils (sont formés d'un très grand nombre de capteurs sans infrastructure préalable. D'une part, un capteur surveille son environnement, capte et propage les données récoltées des capteurs appartenant à sa zone de couverture. Chaque capteur relayant l'information sur sa propre zone de couverture. D'autre part, ces nœuds sont reliés à une ou à plusieurs SB (Stations de Base) qui permettent l'interconnexion avec d'autres réseaux et la récupération des données. L'utilisateur du RCSF peut ainsi, adresser des requêtes aux autres nœuds capteurs du réseau en précisant le type de données requises et recueillir puis analyser les événements recensés par les capteurs sur la zone de couverture [5].

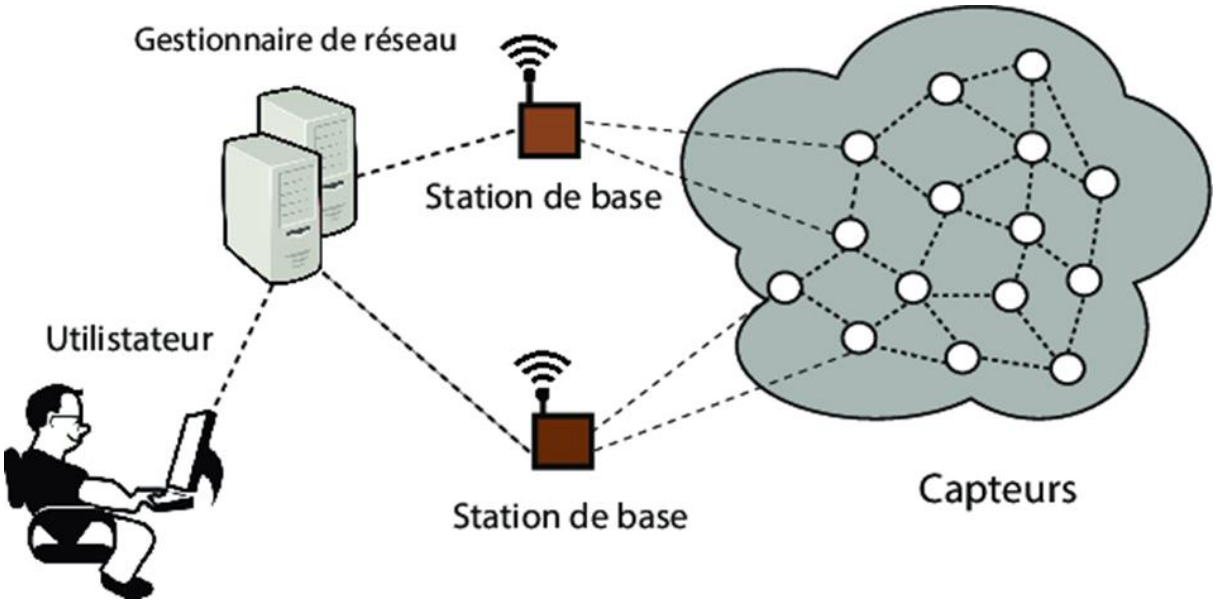


Figure 2 Schéma général d'un réseau capteur sans fil [9]

La miniaturisation du matériel et la multiplication des moyens de connexions associées à l'augmentation des capacités de calcul et de mémoire en informatique ont permis aux réseaux de capteurs d'exister, et cela à une échelle très large au point d'accomplir les tâches les plus complexes pour l'humain. Ainsi, on peut les retrouver désormais dans l'armement, le nucléaire, le sauvetage, la sauvegarde de l'environnement, la médecine, etc.

La recherche continue pour perfectionner le fonctionnement des futurs réseaux de capteurs. De nombreux travaux sont effectués pour résoudre les diverses problématiques auxquelles sont confrontés les capteurs comme la consommation d'énergie, connectivité et couverture du réseau, sécurité et confidentialité des données, fiabilité des mesures et qualité des données, l'emplacement, etc.

Le majeur problème réside souvent dans la gestion de l'énergie et de la consommation d'énergie. Dans ce qui suit, nous allons présenter les sources de cette problématique.

Dans notre travail, nous nous intéressons à ce problème, mais avant de le présenter, nous allons décrire les différentes architectures des RCSF dans la section suivante.

1.3.1 Architectures adoptées :

Les architectures dans les réseaux de capteurs dépendent essentiellement des applications et des techniques utilisées pour faire acheminer l'information des capteurs à la SB. Ce processus d'acheminement de l'information des capteurs vers la SB peut se faire selon deux formes [10]:

- **Une architecture à plat :**

Où les capteurs peuvent communiquer directement avec la SB (figure 3), ou en mode multi-sauts (figure4).

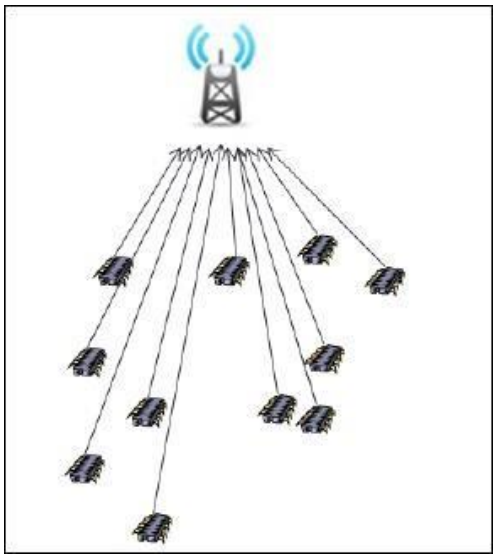


Figure 3 Transmission directe [10]

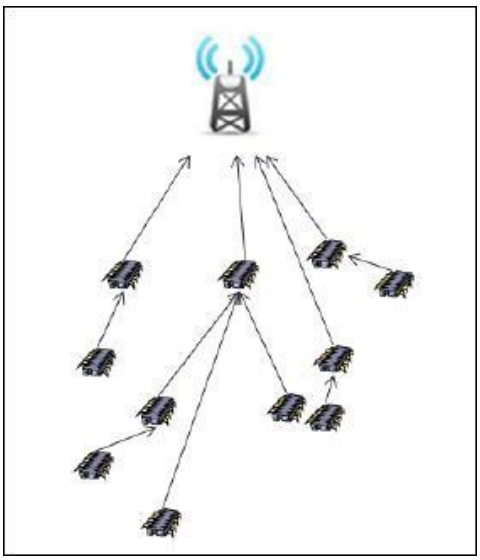


Figure 4 Transmission en multi-sauts [10]

- **Une architecture hiérarchisée :**

Qui consiste à regrouper les capteurs en groupes appelés clusters qui seront représentés par des clusters Head (CH) responsables de transmettre directement les données à la SB (Figure 5). Les réseaux de capteurs à base de cluster sont essentiellement séparés en deux types principaux : homogènes et hétérogènes, dans les réseaux homogènes tous les nœuds de capteurs sont uniques en termes d'énergie dans la batterie et de la complexité du matériel et tous les nœuds peuvent être un CH au tour de rôle tandis que dans les réseaux de capteurs hétérogènes constitué de nœuds à ressources non égales c.à.d. Certains nœuds ont de haute performance peuvent être CH.

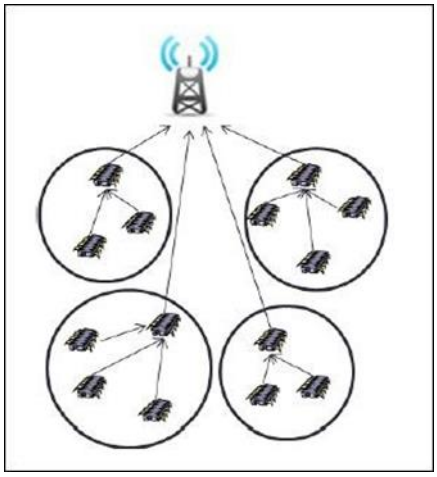


Figure 5 Clustering [10].

Il faut dire que l'architecture adoptée conditionne la répartition de la consommation d'énergie dans le réseau. En effet, dans le cas d'une transmission directe des données captées vers la SB, les capteurs éloignés de celle-ci vont plus rapidement manquer d'énergie. Tandis

que pour la transmission des données captées par sauts, les capteurs proches de la SB vont être vite manqué d'énergie car ils seront plus sollicités pour relayer les messages des autres. Au contraire, pour les architectures hiérarchisées, en changeant régulièrement les clusters-Head on obtient des réseaux où la consommation d'énergie est répartie équitablement.

1.4 Problème de consommation d'énergie :

La durée de vie d'un RCSF réseau de capteurs est généralement définie par le temps durant lequel le réseau est capable de maintenir assez de connectivité, couvrir la zone de captage ou garder son taux de perte de nœud inférieur à un certain niveau. La durée de vie d'un réseau de capteurs est donc liée à la durée de vie des nœuds, qui dépend principalement de la durée de vie de sa batterie. Cette dernière dépend des énergies consommées par les différents modules qui composent le nœud, par la technologie de la batterie et la façon de s'en servir.

1.4.1 Consommation d'énergie du capteur :

L'énergie consommée par un nœud capteur est due essentiellement aux opérations suivantes : la capture, le traitement et la communication de données.

- **Energie de capture :**

Est dissipée pour accomplir l'échantillonnage, le traitement de signal, la conversion analogique/numérique et l'activation de la sonde de capture. En général, l'énergie de capture représente un faible pourcentage de l'énergie totale consommée par un nœud. [11]

- **Energie de traitement :**

Est consommé dans la commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel (en exécutant un logiciel). Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement. En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication. [11]

- **Energie de communication :**

Est consommé par la réception, l'émission et l'écoute du canal. Cette énergie est déterminée par la quantité des données à communiquer et la distance de transmission, ainsi que par les propriétés physiques du module radio. L'émission d'un signal est caractérisée par sa puissance. Quand la puissance d'émission est élevée, le signal aura une grande portée et l'énergie consommée sera plus élevée. Notons que l'énergie de communication représente la portion la plus grande de l'énergie consommée par un nœud capteur [11].

L'histogramme présenté par la figure ci-dessous représente les portions d'énergie consommée par les différentes unités.

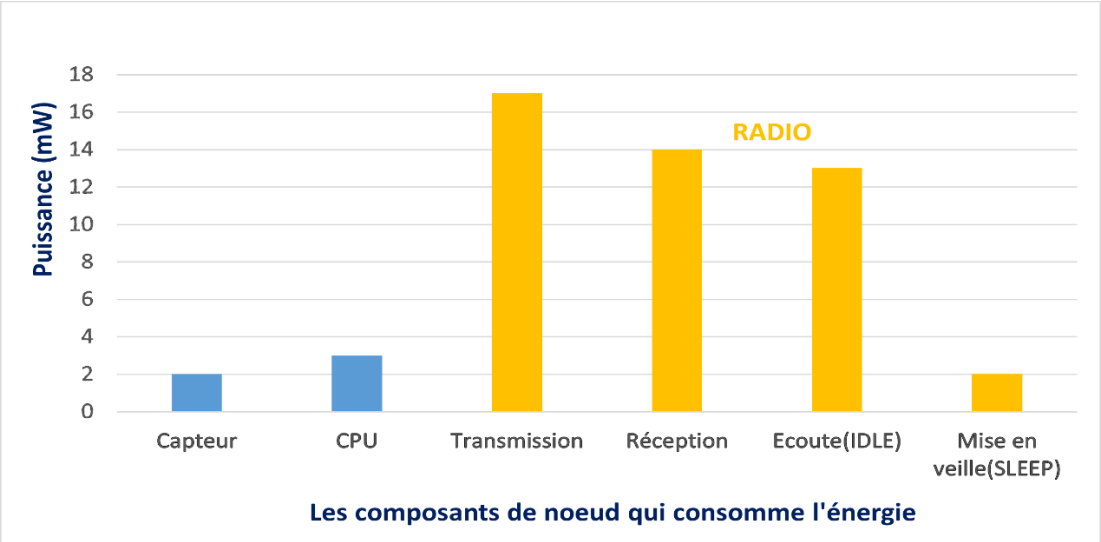


Figure 6 Consommation d'énergie en en acquisition, traitement et communication [11].

1.4.2 Sources de surconsommation d'énergie :

La conception d'un protocole MAC économisant l'énergie joue un rôle essentiel dans la réduction de la consommation d'énergie afin d'étendre la durée de vie des capteurs et du Réseau. Étant donné que les nœuds de capteurs sont supposés être disposés lorsqu'ils sont hors batterie, la conservation d'énergie est une exigence majeure dans la conception des protocoles de communication pour les réseaux de capteurs. Au cours de l'exploitation de RCSF, les principales sources de surconsommation d'énergie sont présentées dans les paragraphes suivants [12]:

- **État du module radio** : Le module radio est le composant du nœud qui consomme le plus d'énergie car c'est lui qui assure la communication entre les nœuds de capteurs. On distingue quatre modes des composants radio (transmetteur et récepteur) : actif, réception, transmission et sommeil. Il est aussi à noter que le passage fréquent du mode actif au mode sommeil peut avoir comme effet une consommation d'énergie plus élevée que de laisser le module radio en mode actif. Ceci est dû à la puissance nécessaire pour la mise sous tension du module radio. Cette énergie est appelée l'énergie de transition. Ce changement d'état du module radio doit être géré par un protocole de la couche MAC [12].

- **Les collisions :** Elles sont la première source de perte d'énergie. Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées. Elles se manifestent alors lorsqu'un nœud reçoit plus d'un paquet simultanément, tous les paquets qui causent la collision doivent être jetés et retransmis, leur retransmission consomme plus d'énergie. Tous les protocoles Mac avec contention intègrent un processus d'évitement de collisions. Les nœuds capteurs sont composés essentiellement par une seule antenne radio et partagent le même canal de transmission. Par ailleurs, l'émission simultanée des données provenant de différents nœuds capteurs peut amener à l'apparition des collisions et ainsi une éventuelle perte de l'information transmise. La retransmission des paquets perdus peut causer une perte significative de l'énergie [12].

- **L'écoute active/ inactive :** L'écoute inactive se produit lorsqu'un nœud écoute le canal pour une réception possible. C'est une source de gaspillage importante d'énergie dans les RCSF où il consomme habituellement 50 à 100% de l'énergie requise pour le processus de réception. L'écoute active du canal pour une telle réception de paquet qui ne sera pas reçue peut engendrer une grande perte de la capacité des nœuds en énergie. Il faut donc basculer les capteurs vers le mode sommeil le plus longtemps possible dans le but d'éviter ce problème [12].

- **La sur-écoute (Overhearing) :** Cette source de dissipation d'énergie se produit lorsqu'un nœud capteur gaspille de l'énergie en recevant un paquet destiné à une destination différente. La sur-écoute apparaît donc quand un capteur reçoit des paquets qui ne lui sont pas destinés. La sur-écoute conduit à une perte d'énergie additionnelle à cause de l'implication des autres capteurs dans la réception des données [12].

- **La surcharge (Overhead) :** Les paquets de données dans les RCSF sont généralement petits ; par conséquent, les en-têtes et les autres types de paquets (tels que les messages de contrôle, comme RTS, CTS, SYNC et ACK) impliquent un niveau élevé de gaspillage d'énergie. Plusieurs protocoles de la couche MAC fonctionnent par échange de messages de contrôle pour assurer différentes fonctionnalités : connectivité, signalisation, établissement de plan d'accès et évitement les collisions [12].

- **La surémission (Overemitting) :** La surémission est causée lorsque la livraison du message échoue en raison de l'inactivité du nœud de destination. Ce phénomène apparaît aussi quand un nœud capteur envoie les données à un destinataire qui n'est pas prêt à les recevoir. En effet, les messages envoyés sont considérés inutiles et consomment une énergie additionnelle.

- **La taille des paquets :** La taille des messages échangés dans le réseau a un effet sur la consommation d'énergie des nœuds émetteurs et récepteurs. Ainsi, la taille des paquets ne doit pas être ni trop élevée ni trop faible. En effet, si elle est petite, le nombre de paquets de contrôle (acquittement) générés augmente l'overhead. Dans le cas contraire, une grande puissance de transmission est nécessaire pour des paquets de grande taille [12].

1.5 Les techniques d'économies d'énergies dans les RCSFs:

Dans les réseaux de capteurs, la consommation d'énergie est très importante puisque généralement les capteurs sont déployés dans des zones inaccessibles. Ainsi, il est difficile de voir et impossible de remplacer les batteries après leur épuisement. De ce fait, la consommation d'énergie au niveau des capteurs a une grande influence sur la durée de vie du réseau. Nous présentons dans ce qui suit les différentes techniques utilisées pour minimiser cette consommation. Le schéma présenté par (la figure 7) donne un aperçu global de ces mécanismes.

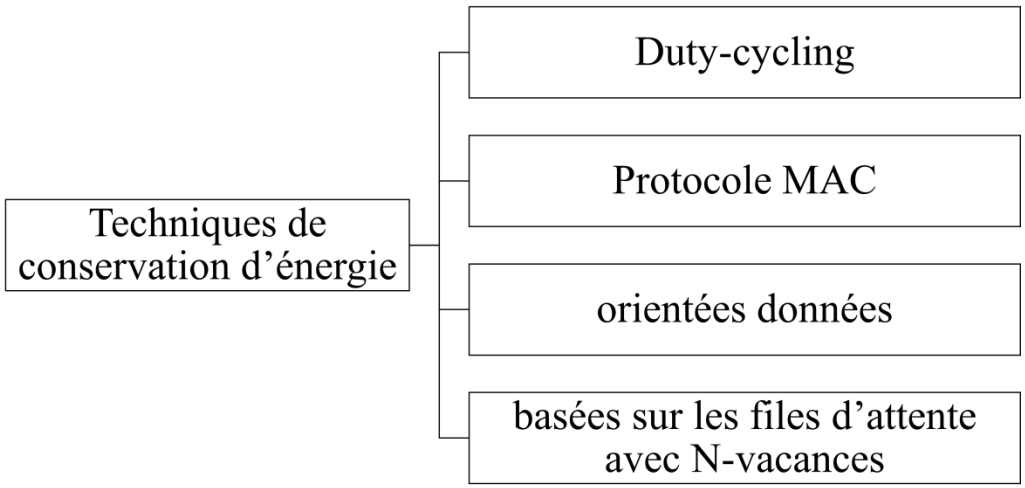


Figure 7 Classifications des techniques de conservation d'énergie.

1.5.1 La technique Duty-cycling :

Cette technique est principalement utilisée dans l'activité réseau. Le moyen le plus efficace pour conserver l'énergie est de mettre la radio de l'émetteur en mode veille (low-power) à chaque fois que la communication n'est pas nécessaire. Idéalement, la radio doit être éteinte dès qu'il n'y a plus de données à envoyer et ou à recevoir, et devrait être prête dès qu'un nouveau paquet de données doit être envoyé ou reçu. Ainsi, les nœuds alternent entre périodes actives et sommeil en fonction de l'activité du réseau. Ce comportement est généralement dénommé Duty-cycling. Un Duty-cycle est défini comme étant la fraction de temps où les nœuds sont actifs ITE. [13]

Un schéma sleep/wakeup peut être défini pour un composant donné (le module Radio) du nœud capteur. Les protocoles sleep/wakeup sont divisés en deux grandes catégories : à la demande, rendez-vous programmés.

Les protocoles à la demande utilisent l'approche la plus intuitive pour la gestion d'énergie. L'idée de base est qu'un nœud devrait se réveiller seulement quand un autre nœud veut communiquer avec lui. Le problème principal associé aux régimes à la demande est de savoir comment informer un nœud en sommeil qu'un autre nœud est disposé à communiquer avec lui. À cet effet, ces systèmes utilisent généralement plusieurs radios avec différents compromis entre énergie et performances (i.e. un radio à faible débit et à faible consommation pour la signalisation, et un radio à haut débit mais à plus forte consommation pour la communication de données). Le protocole STEM (Sparse Topology and Energy Management) par exemple utilise deux radios. [13]

Une autre solution consiste à utiliser une approche de rendez-vous programmés. L'idée est que chaque nœud doit se réveiller en mémé temps que ses voisins. Typiquement, les nœuds se réveillent suivant un ordonnancement de réveil et restent actifs pendant un court intervalle de temps pour communiquer avec leurs voisins. Ensuite, ils se rendorment jusqu'au prochain rendez-vous. [13]

1.5.2 Les techniques des protocoles du niveau MAC :

Plusieurs protocoles MAC pour les réseaux de capteurs sans fil ont été proposés, Nous nous concentrons principalement sur les questions de gestion d'énergie. La plupart d'entre eux mettent en œuvre un régime avec un faible duty-cycle pour gérer la consommation d'énergie.

Dans les protocoles MAC fondés sur la méthode TDMA (Time Division Multiple Access) le temps est divisé en trames (périodiques) et chaque trame se compose d'un certain nombre de slots de temps. A chaque nœud est attribué un ou plusieurs slots par trame, selon un certain algorithme d'ordonnancement. Il utilise ces slots pour l'émission/réception de paquets de/vers d'autres nœuds. Dans de nombreux cas, les nœuds sont regroupés pour former des clusters avec un cluster Head qui est chargé d'attribuer les slots de temps pour les nœuds de son cluster (par exemple : Bluetooth, LEACH). [14]

Les protocoles TDMA sont par nature efficaces en énergie, puisque les nœuds n'allument leur radio que lors de leurs propres slots et s'endorment le reste du temps.

Toutefois, dans la pratique, les protocoles TDMA ont plusieurs inconvénients qui compensent les avantages en termes d'économie d'énergie. En effet, dans un véritable réseau de capteurs, les changements de topologie sont fréquents (conditions variables du canal, défaillances de nœuds, . . .) et la répartition des slots peut être problématique donc dans de nombreux cas une approche centralisée peut être adoptée (LEACH). [14]

1.5.3 Techniques orientées données :

Généralement, les plans Duty-cycling ne tiennent pas compte des données prélevées par les nœuds. Par conséquent, des approches orientées données peuvent être utiles pour améliorer l'efficacité en énergie. En fait, la détection (ou prélèvement de données) affecte la consommation d'énergie de deux manières : [14]

• **Des échantillons inutiles :** les données échantillonnées ont souvent de fortes corrélations spatiales et/ou temporelle [15], il est donc inutile de communiquer les informations redondantes à la Station de Base. Un échantillonnage inutile implique une consommation d'énergie à son tour inutile. En effet, même si le coût de l'échantillonnage est négligeable, cela induit aussi des communications tout le long du chemin qu'emprunte le message.

La consommation électrique du module de détection : réduire la communication ne suffit pas lorsque le capteur est lui-même très consommateur.

• **Réduction des données** [14]**:** Réduire les données en terme de volume ou de nombre de paquets, dans le réseau peut avoir un impact majeur sur la consommation d'énergie due à la communication. Parmi les méthodes de réductions de données, nous trouvons le In-network processing qui consiste à réaliser de l'agrégation de données (par exemple, calculer la moyenne de certaines valeurs) au niveau des nœuds intermédiaires entre la source et le Sink. Ainsi, la quantité de données est réduite tout en parcourant le réseau vers le Sink. Une agrégation de données appropriée est spécifique à l'application. La compression de données peut être appliquée également pour réduire la quantité d'informations transmises par les nœuds sources. Ce régime implique l'encodage d'informations au niveau des nœuds qui engendrent des données, et le décodage au niveau du Sink.

1.5.4 Les techniques basées sur les modèles des files d'attentes avec vacances

Vue l'ampleur des applications émergentes des RCSF, l'évaluation de leurs performances est nécessaire. Une approche très importante et complémentaire pour l'analyse et l'évaluation de ces réseaux est d'utiliser des techniques de modélisation formelles qui fournissent des résultats exacts. Parmi ces techniques, nous trouvons la modélisation des nœuds capteurs avec des files d'attente avec vacances pour la conservation d'énergie des réseaux de capteur sans fil. Ces modèles ont fait l'objet de plusieurs travaux [1] et [16] [17] [18] [19] [20].

Initialement, la politique N-vacance (N-policy) a été présentée dans [21] par Yadin et Naoren. Cette politique a été utilisée pour contrôler le passage entre l'état de vacance et l'état de service d'une file d'attente avec vacance. Ensuite, elle a été utilisée comme politique pour contrôler le passage entre l'état idle et l'état busy du noeud capteur dans un RCSF pour économiser l'énergie [1] [3] [16] [17] [19] [20].

N-vacance permet la transition du nœud capteur de l'état idle à l'état busy lorsqu'il y a au moins N ($N \geq 1$) clients (paquets) en attente. Par souci de clarté, plusieurs ont abordées la politique N-vacance, pour simplifier leurs modèles [16] [17] ils ont considérés un système avec buffer infini. Dans d'autres travaux [1] [3] [20], les auteurs ont optés pour un modèle avec buffer fini.

Dans notre projet de fin d'étude, nous nous intéressons au modèle de Boutoumi et Gharbi réalisé dans [1**].** Dans ce modèle, un nœud capteur est modélisé par une file d'attente avec vacance et buffer fini de capacité K en utilisant les Chaines de Markov à Temps Continue (CTMC). Ce capteur alterne entre deux principaux modes (état) : sommeil (sleep) et actif (active). Lorsqu'il est en mode sommeil, le capteur ne peut pas interagir avec son environnement par conséquent, l'énergie consommé est très basse. Durant le mode actif, le capteur écoute le canal, génère des paquets, reçois et transmet les paquets de données de ces voisins donc l'énergie consommé est considérable. Ainsi, pendant la période actif le capteur transite entre l'état oisif où il peut générer et recevoir des paquets et l'état occupé où il peut générer, recevoir et émettre les paquets. Il est à noter également que l'énergie dissipée pour la transition entre l'état oisif et l'état occupé, génère une surconsommation d'énergie importante nommé l'énergie de transition (the switching energy).

Formellement, dans ce système, les paquets (relayés et détectés) arrivent selon le processus de poisson avec un taux moyen λ et les temps de service suivent une distribution exponentielle avec une moyenne de 1/ μ. Pour commencer, le nœud reste à l'état oisif (idle) pour conserver l'énergie, ce qui correspond à l'état de vacances d'un modèle de file d'attente avec vacance. Lorsqu'un nœud est en état oisif il peut produire des données de détection, collecter des paquets de données relayés et les stockés dans le buffer jusqu'à ce que le nombre de paquets dans la file d'attente atteigne le seuil fixé N. À ce même instant, il passe à l'état occupé. Lorsque le nœud est dans un état occupé, il est en mesure de créer des données de détection, ainsi que de recevoir et d'envoyer des paquets de manière exhaustive. Une fois le buffer devient vide, le nœud retourne à son état oisif. La figure ci-dessus montre le comportement du modèle proposé pour le capteur.

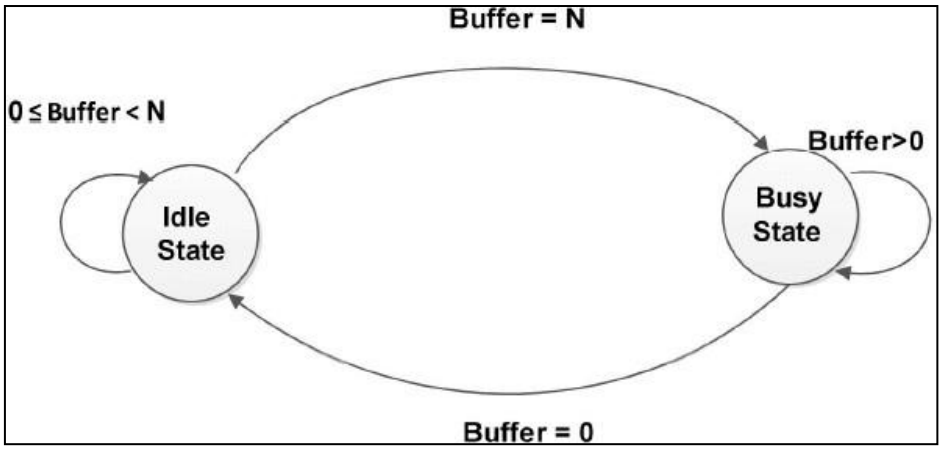


Figure 8 Diagramme d'état transition de la technique N-vacance [1]

Le comportement du capteur est modélisé par une CMTC à deux dimensions où chaque état est décrit par deux variables aléatoires (X(t), Y(t) : t≥0), tel que X(t) représente le nombre de paquet au niveau de buffer et Y(t) représente l'état de système [3].

Où :

- ✓ Y(t)=0 le serveur est à l'état oisif
- ✓ Y(t)=1 le serveur est à l'état occupée

L'espace d'état de la CMTC est donnée par : $\Omega = \{(i, j) : 0 \leq i \leq K, 0 \leq j \leq 1\}$

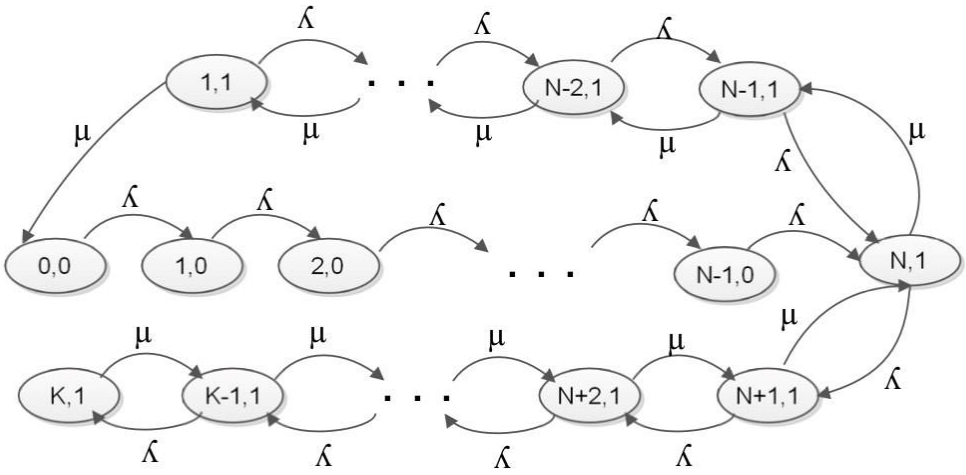


Figure 9 Le modèle CTMC avec N-Policy et un seul type de Traffic [3]

Pour valider ces modèles, plusieurs tests ont été faits dans [3] en donnant des valeurs aléatoires aux variables (N, K, λ, μ) mais de manière ordonnée (Ils ont fixés des entrées et variés d'autres entrées dans un ordre croissant ou décroissant). Cependant, cette méthode (appelée aussi heuristique) aléatoire prend beaucoup temps et ne retourne pas des meilleurs résultats. C'est dans ce contexte que s'inscrit notre problématique qui consiste à utiliser d'autres méthodes approchées (notamment les méta-heuristiques) pour résoudre ce problème d'optimisation. Ce problème d'optimisation composé des variables (N, K, λ, μ) permet de trouver la meilleure solution S* = (N*, K*, λ*, μ*) permettant de minimiser la consommation

d'énergie ainsi que de délai dans les RCSF. Dans la partie suivante, nous expliquons les concepts de base qui vont nous permettre de résoudre ce problème d'optimisation.

Partie 2 : Problèmes d'Optimisation.

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données.

2.1 Définitions

Un problème d'optimisation est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble SR deS représentant les solutions admissibles (réalisables) et une fonction objective (maximiser ou minimiser) qui assigne à chaque solution $s\in$ SR le nombre réel (ou entier) (s). Résoudre un tel problème consiste à trouver une solution $s*\in$ SR optimisant la valeur de la fonction de coût f. Une telle solution s* s'appelle une solution optimale ou un optimum global.

Formellement, un Problème d'Optimisation peut être défini comme suit :

- Un ensemble des variables de décision X
- Un ensemble de solutions S, chaque solution est composée d'un ou plusieurs variables de décision.
- Un ensemble de contraintes C (éventuellement vide) qui définit des conditions sur l'espace d'états que les variables doivent satisfaire, permettent en général de limiter l'espace de recherche (est l'ensemble des solutions réalisables).
- Un sous-ensemble SR de S représentant les solutions admissibles (ou réalisables) qui vérifient les contraintes C.

- Une fonction objectif (ou de coût) f qui assigne à chaque solution s ∈ SR une valeur f(s).
- Il s'agit de trouver une solution optimale (ou optimum global) s*∈ SR qui optimise (minimise ou maximise) la fonction objectif.

- ❖ Dans le cas de minimisation : $\forall s \in SR, f(s*) \leq f(s)$
- ❖ Dans le cas de maximisation : $\forall s \in SR, f(s*) \geq f(s)$

Résoudre un problème d'optimisation combinatoire nécessite l'étude des points suivants : [22]

- ✓ Définir l'ensemble des variables de décision X et des solutions « S »
- ✓ Exprimer l'ensemble des contraintes du problème « C » afin de définir l'ensemble des solutions réalisables « SR »,
- ✓ Exprimer la fonction objectif « f » à optimiser,
- ✓ Choisir la méthode de résolution à utiliser,

Bien qu'ils soient généralement faciles à formaliser/modéliser (les trois premiers points), les problèmes d'optimisation peuvent s'avérer très difficiles à résoudre.

2.2 Classification

Face à la résolution d'un problème d'optimisation, il est important de bien identifier à quelle catégorie ce problème appartient. En effet, les algorithmes développés sont conçus pour résoudre un type de problème donné et sont peu efficaces pour un type différent. La classification des problèmes d'optimisation peut se faire selon plusieurs critères [23].

• Classifier selon le type des variables de décision :

Dans certains cas, les variables de décision sont discrètes, le plus souvent sous la forme d'entiers ou de binaires, le problème d'optimisation est dit discret (l'optimisation discrète est également appelée optimisation combinatoire). En revanche, dans les problèmes d'optimisation continue, les variables peuvent prendre n'importe quelle valeur, ce sont des nombres réels (les variables). Les problèmes d'optimisation continue sont généralement plus simples à résoudre. Un problème d'optimisation est dit mixte si ces variables peuvent être à la fois discrètes et continues.

• Classifier selon le nombre de contraintes :

Il existe deux types de problèmes : les problèmes d'optimisation avec et sans contrainte, Il est important de bien distinguer les problèmes où des contraintes existent sur les variables de décision. Ces contraintes peuvent être simplement des bornes et aller jusqu'à un ensemble d'équations de type égalité et de type inégalité. Il est parfois possible d'éliminer une contrainte égalité par substitution dans la fonction objectif. Généralement, les problèmes avec contraintes sont plus compliqués à résoudre et utilisent des algorithmes dédiés.

• Classifier selon le nombre d'objectifs :

On distingue les problèmes d'optimisation mono-objectif ou multi-objectif, Les problèmes mono-objectifs sont définis par une unique fonction objectif. Le problème des objectifs multiples (multi-objectifs) se pose lorsqu'il s'agit de trouver un compromis entre plusieurs objectifs contradictoires. Il est éventuellement possible (mais pas nécessairement efficace) de reformuler un problème multi-objectif avec une seule fonction objectif sous forme d'une combinaison des différents objectifs ou en transformant des objectifs sous forme de contraintes.

Il est important de noter que la plupart des problèmes d'optimisation dans le monde réel sont multi-objectifs.

• Classifier selon la nature des fonctions du problème :

Nous disons que les problèmes d'optimisation sont linéaires si la fonction objectif et les contraintes sont linéaires, et pour les problèmes d'optimisation non linéaires si la fonction objectif et les contraintes sont représentées par des relations non linéaires.

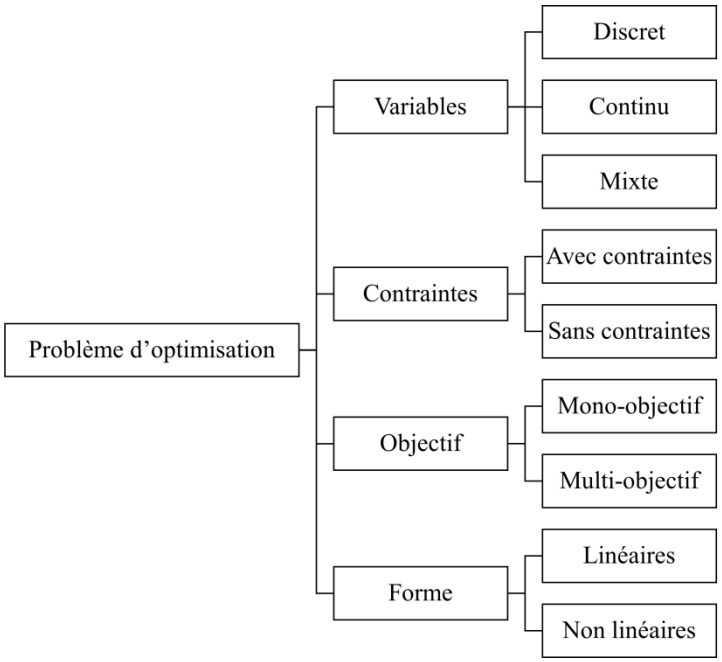


Figure 10.Classification des problèmes d'optimisation [23]

2.3 Les méthodes de résolution :

La résolution d'un problème d'optimisation consiste à parcourir l'espace de recherche (l'ensemble des instances pouvant être prises par les variables du problème) afin d'en extraire une solution optimale parmi un ensemble fini de solutions, d'une taille souvent très grande telle que son énumération exhaustive est une tâche fastidieuse. Selon le type de la recherche, on distingue deux grandes classes des méthodes de résolution, le schéma ci-dessous représente cette classification :

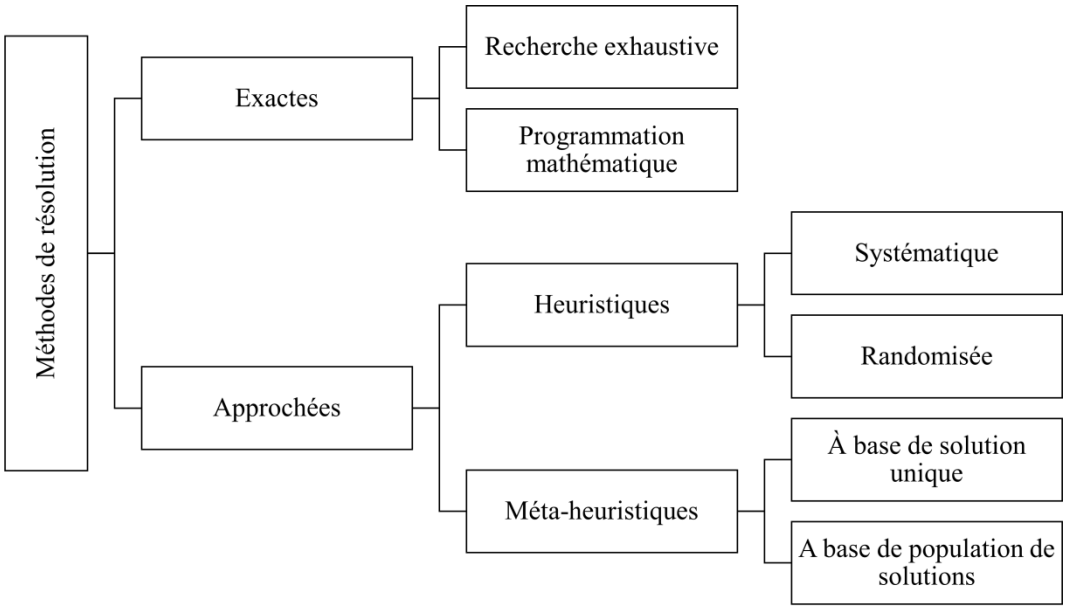


Figure 11.Classification de méthodes de résolution de problèmes d'optimisation

2.3.1 Méthodes exactes (complètes)

Elles se basent généralement sur une recherche complète de l'espace des solutions afin de trouver une solution optimale. Il y a deux types [22] :

- **Méthodes de recherche exhaustive :** qui énumèrent toutes les solutions possibles pour trouver la meilleure solution. Cela peut être réalisé à l'aide de techniques telles que les algorithmes de Branch and Bound ou back-tracking.
- **Méthodes de programmation mathématique :** qui utilisent des modèles mathématiques pour formuler le problème d'optimisation sous forme d'un programme linéaire, d'un programme en nombres entiers ou d'autres formes spécifiques. Les solveurs mathématiques, tels que le simplexe ou les solveurs de programmation en nombres entiers, sont utilisés pour résoudre ces modèles.

Les méthodes exactes sont connues par le fait qu'elles garantissent l'optimalité de la solution mais elles demandent des coûts de recherche (temps de calcul et espace mémoire) souvent prohibitifs qui augmentent avec la taille de l'instance du problème traité.

2.3.2 Les méthodes approchées (incomplètes)

Elles se basent généralement sur une recherche incomplète de l'espace des solutions afin de trouver une bonne solution (pas forcément optimale) dans un temps raisonnable. La majorité des problèmes d'optimisation, sont des problèmes NP-difficiles et donc ils ne possèdent pas à ce jour un algorithme exacte efficace, i.e. de complexité polynomiale, valable de trouver la solution optimale en un temps raisonnable. Ceci a motivé les chercheurs à développer de nombreuses méthodes approchées :

- La recherche s'est d'abord orientée vers des heuristiques spécifiques aux problèmes,

• elle s'est progressivement intéressée aux méthodes plus générales, c'est à dire les méta-heuristiques.

2.3.2.1 Les heuristiques

Selon [24], Une heuristique est : « une méthode qui aide à découvrir la solution d'un problème en faisant des conjectures plausibles mais faillible de ce qui est la meilleure chose à faire. » Les heuristiques disposent d'une simplicité et donc d'une rapidité dans leur exécution plus élevée que les algorithmes classiques. S'appliquant à un ensemble particulier la recherche des faits se voit simplifiée et accélérée (moins de possibilité), d'où une analyse des situations améliorées. Mais une méthode heuristique trop simplifiée ou au contraire trop générale peut conduire à des biais cognitifs, générant des erreurs de décision. L'utilisation de plus de ces éléments simples (les heuristiques) afin de créer des éléments plus complexes (les méta-heuristiques) permet donc de réduire considérablement l'ensemble de recherche global de l'algorithme.

Il existe 2 catégories de heuristiques [22]:

• **Heuristique systématique** qui explore le voisinage pour choisir la solution suivante:

- Meilleur voisin : choisir la solution voisine qui est évaluée le mieux.
- Premier voisin : choisir la première solution voisine qui améliore la fonction objectif.
- Heuristique multi-étages : choisir d'abord un sous-ensemble des solutions du voisinage ensuite choisir dans cet ensemble une solution qui améliore le mieux la fonction objectif

• **Heuristique randomisée** qui est caractérisée par un choix aléatoire de la prochaine solution :

- Amélioration aléatoire : choisir au hasard une solution voisine et se déplacer vers cette solution si elle améliore la fonction objectif.
- Heuristique Métropolis : choisir au hasard une solution voisine de la solution courante et se déplacer vers cette solution si elle n'accroît pas la fonction objectif. Sinon, elle est acceptée avec une certaine probabilité, déterminée de façon différente selon chaque programme.

L'une de leur caractéristique principale et à première vue défaut, dont hérite également les méta-heuristiques, est qu'ils peuvent dans certains cas ne pas proposer de solution optimale au problème. Mais au résultat s'y approchant d'assez près pour qu'il soit considéré comme correct, on parle alors de garantie de performance.

2.3.2.2 Méta-heuristiques

Le terme méta-heuristique, introduit par Glover en 1986 [25], est dérivé de la composition de deux mots grecs :

- heuristique qui vient du verbe heuriskein (euriskein) et qui signifie "trouver" ;
- meta qui est un suffixe signifiant "au-delà", et indique un changement de niveau, un passage à un niveau "supérieur" pour étudier ou manipuler des informations de niveau inférieur.

Plusieurs définitions ont été proposées pour expliquer clairement ce qu'est une méta-heuristique. Aucune de ces définitions n'est universellement reconnue. Une définition est donnée dans [26]: « Une méta-heuristique est formellement définie comme un processus de génération itératif qui guide une heuristique sous-jacente en combinant intelligemment différents concepts pour exploiter et explorer l'espace de solutions, et où des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions près de l'optimum. »

Une méta-heuristique généralement, n'est pas propre à un problème précis, mais adaptable et applicable à une large classe de problèmes. Elles sacrifient la garantie d'optimalité ou d'approximation avec en contrepartie l'espoir de trouver très rapidement de bonnes solutions. Pour adapter une méta-heuristique à un problème d'optimisation particulier, il est nécessaire de modéliser adéquatement un certain nombre d'ingrédients :

- L'ensemble S des solutions ; cet ensemble définit le domaine dans lequel la recherche d'un optimum va s'effectuer.
- la fonction objectif f à minimiser (ou à maximiser) ; cette fonction induit une topologie sur l'espace de recherche, avec des montagnes (mauvaises solutions), des vallées (régions autour d'un minimum local), etc.
- le voisinage d'une solution, ce concept indique les déplacements possibles dans S. L'exploration de S consiste à parcourir un chemin qui se dirige à chaque pas d'une solution vers une solution voisine.
- L'opérateur de combinaison ; utilisé dans le cadre des méthodes évolutives, cet opérateur permet de générer de nouvelles solutions à partir des solutions présentes dans la population courante.

a. Propriétés :

Les propriétés fondamentales des méta-heuristiques ont été résumées dans [27]:

1. Elles sont des stratégies qui permettent de guider la recherche d'une solution optimale

2. Leur but visé est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales.

3. Les techniques qui constituent leurs algorithmes vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.

4. Elles sont en général non-déterministes et ne donnent aucune garantie d'optimalité.

5. Elles peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.

6. Leurs concepts de base peuvent être décrit de manière abstraite, sans faire appel à un problème spécifique.

7. Elles peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.

8. Elles peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum pour mieux guider la suite du processus de recherche.

9. Elles sont, au moins pour partie, stochastiques : cette approche permet de faire face à l'explosion combinatoire des possibilités ;

10. Elles sont inspirées par des analogies : avec la physique (recuit simulé, diffusion simulée...), avec la biologie (algorithmes évolutionnaires, recherche tabou...) ou avec l'éthologie (colonies de fourmis...) ;

11. Elles partagent aussi les mêmes inconvénients : les difficultés de réglage des paramètres de la méthode, et le temps de calcul élevé.

Les méta-heuristiques reposent finalement sur un ensemble commun de concepts peu nombreux : la mémoire, qui sauvegarde l'information recueillie par l'algorithme, l'intensification, qui tente d'améliorer la pertinence des informations disponibles, au moyen de recherches locales, et la diversification, qui vise à accroître la quantité de ces informations, en explorant de nouvelles régions de l'espace de recherche.

b. Classification :

Selon le nombre de solutions, on peut distinguer deux classes des méta-heuristiques :

- **Méta-heuristique à base d'une solution unique**
  Elle lance la recherche avec une seule solution et essaye d'améliorer sa qualité au cours des itérations. On trouve la recherche locale simple (descente), recuit simulé, recherche tabou
- **Méta-heuristique à base d'une population de solutions**
  Elle lance la recherche avec une population de solutions et essaye d'améliorer leurs qualités au cours des itérations dans le but de fournir la ou les meilleures solutions trouvées**.** On trouve les algorithmes génétiques, les algorithmes de colonies de fourmis, essaim de particules.

Dans notre travail, nous nous sommes orientés vers les algorithmes génétiques et l'optimisation par essaim des particules. Dans ce qui suit, nous décrivons ces méta-heuristiques qui seront détaillées formellement dans le chapitre suivant.

2.4 L'algorithme génétique (AG)

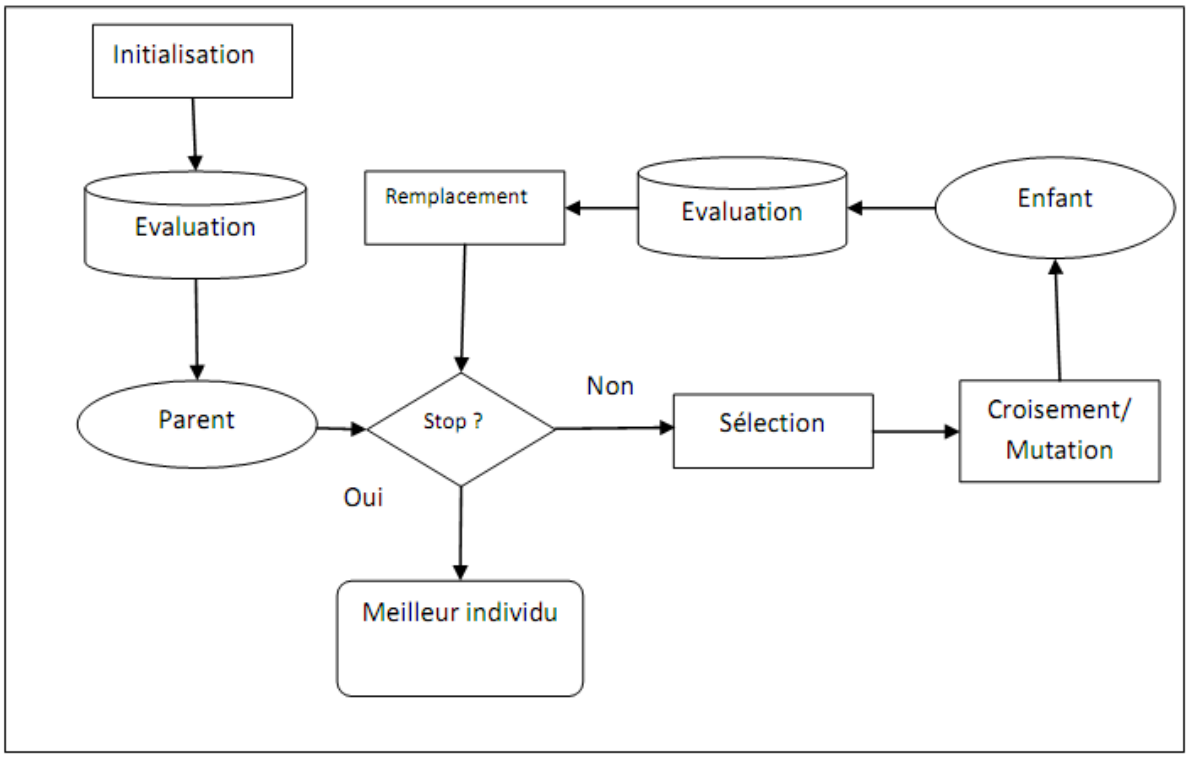
Les algorithmes génétiques (Genetic Algorithm GA) sont des algorithmes d'optimisation inspirés de la théorie de l'évolution des espèces de Charles Darwin [28]. Leur processus de recherche de solutions à un problème donné imite celui des êtres vivants dans leur évolution. Il utilise le même vocabulaire que celui de la biologie et la génétique classique, on parle donc de :

- Gène : est un ensemble de symboles représentant la valeur d'une variable. Dans la plupart des cas, un gène est représenté par un seul symbole (un bit, un entier, un réel ou un caractère).
- Chromosome : est un ensemble de gènes, présentés dans un ordre donné de manière qui prend en considération les contraintes du problème à traiter.

- Individu : est composé d'un ou de plusieurs chromosomes. Il représente une solution possible au problème traité.
- Population : est représentée par un ensemble d'individus (i.e. l'ensemble des solutions du problème).
- Génération : est une succession d'itérations composées d'un ensemble d'opérations Permettant le passage d'une population à un autre individu, population et génération.

L'algorithme génétique fait évoluer une population composée d'un ensemble d'individus pendant un ensemble de génération jusqu'à ce qu'un critère d'arrêt soit vérifié. Le passage d'une population à une autre est réalisé grâce à des opérations d'évaluation, de sélection, de reproduction (croisement et mutation) et de remplacement.

L'algorithme commence la recherche avec un ensemble d'individus. A chaque itération de la procédure de recherche, les meilleurs individus sont sélectionnés pour survivre et se reproduire. La sélection des individus est fondée sur leurs qualités qui sont mesurées à partir d'une fonction appelée « fonction objectif ou fonction fitness ». Ensuite, les individus (appelés parents) sont sélectionnés pour subir des opérateurs de croisement et de mutation permettant la génération d'une autre population d'individus (appelés enfants). Les individus de la nouvelle population seront évalués pour remplacer une partie des individus de la population courante. L'algorithme 1 illustre les étapes du processus de recherche de l'algorithme génétique.



Algorithme 1 Algorithme Génétique [29]

a. Opérateurs

Le processus de recherche de l'algorithme génétique est fondé sur les opérateurs suivants :

- Un opérateur de codage des individus : Il permet la représentation des chromosomes représentant les individus.

- Un opérateur d'initialisation de la population : Il permet la production des individus de la population initiale. Malgré que cet opérateur ne s'intervienne qu'une seule fois et au début de la recherche, mais il joue un rôle non négligeable dans la convergence vers l'optimum global. En fait, le choix de la population initiale peut rendre la recherche de la solution optimale du problème traité plus facile et plus rapide.
- Un opérateur de sélection : Il permet de favoriser la reproduction des individus qui ont les meilleures fitness (i.e. les meilleures qualités).
- Un opérateur de croisement : Il permet l'échange des gènes entre parents (deux parents en général), pour créer un ou deux enfants en essayant de combiner les bonnes caractéristiques des parents. Le but de cet opérateur est la création de nouveaux individus en exploitant l'espace de recherche.
- Un opérateur de mutation : Il consiste à modifier quelques gènes des chromosomes des individus, dans le but d'intégrer plus de diversité au sein du processus de la recherche.
- Un opérateur d'évaluation : Il permet de valoriser la qualité des individus, en se basant sur la fonction « objectif » (la fonction fitness) qui permet de calculer la qualité de chaque individu.

En outre des différents opérateurs permettant de guider la recherche par l'algorithme génétique, ce dernier nécessite un certain nombre de paramètres de base, sur lesquels dépendent les différents opérateurs cités en dessus. Ces paramètres doivent être fixés à l'avance, ils jouent un rôle très important dans la performance de l'algorithme. On parle de :

- La taille de la population : Elle représente le nombre d'individus de la population. S'il est trop grand, le processus de recherche demande un coût de recherche élevé, que ce soit en termes d'espace mémoire ou du temps de calcul nécessaires. Cependant, s'il est trop petit, l'algorithme risque d'être tombé dans le cas de la convergence prématurée à cause du manque de la diversité au sein de la population. Il est préférable donc de choisir une taille moyenne en prenant en considération l'instance du problème à traiter.
- La probabilité de croisement : Elle représente la probabilité d'échange de patrimoine (i.e. les gènes) entre deux individus (ou plus). Plus elle est grande, plus elle permet la génération de nouveaux enfants qui peuvent être meilleurs que leurs parents.
- La probabilité de mutation : Elle est en général faible, dans le but d'échapper aux possibilités de modifications radicales des solutions, particulièrement, des solutions de bonnes qualités qui ne nécessitent que peu d'amélioration pour passer aux solutions optimales.
- Le nombre maximum de génération : Ce paramètre peut jouer le rôle d'un critère d'arrêt. Il peut construire un obstacle pour l'algorithme. En fait, il peut empêcher les différents opérateurs d'aboutir à la meilleure solution s'il est trop petit. Comme il peut engendrer un temps de calcul prohibitif dans le cas où il est trop grand. Ainsi, le choix de sa valeur peut se baser sur des tests préliminaires.

b. Avantages et limites

Les algorithmes génétiques présentent plusieurs avantages tels que :

- La simplicité de l'approche.
- La possibilité de paralléliser l'algorithme.
- La facilité d'implémentation.

- La flexibilité : peut être facilement modifié pour d'autres problèmes.
- Il gère les problèmes d'optimisation multi-objectifs et multi-modales.
- Il permet une bonne exploration de l'espace de recherche.

D'autre part, il existe des limites pour cet algorithme tels que :

- Le problème de représentation de la solution
- L'ajustement de différents paramètres : taille de population, taux de mutation, …
- Son exécution qui est lente par rapport à d'autres méthodes bio-inspirées.
- Sa convergence prématurée
- Il ne peut pas garantir des temps de réponse constants

2.5 L'optimisation par essaim de particules (OEP)

L'Optimisation par Essaims de Particules « OEP » (Particle Swarm Optimization PSO) est inspirée du comportement social des animaux évoluant en essaim. En effet, chez certains groupes d'animaux, comme les oiseaux migrateurs et les bancs de poissons, on peut observer des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu n'a qu'une intelligence limitée et une connaissance locale de sa situation dans l'essaim. Un individu de l'essaim n'a pour connaissance que la position et la vitesse de ses plus proches voisins. Chaque individu utilise donc, non seulement, sa propre mémoire, mais aussi l'information locale de ses voisins pour décider de son propre déplacement. 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 » sont des exemples de comportements qui suffisent à maintenir la cohésion de l'essaim, et qui permettent la mise en œuvre de comportements collectifs complexes et adaptatifs.

a. Fonctionnement

Les auteurs d'OEP (Eberhart et Kennedy) [30] se sont inspirés de ces comportements socio-psychologiques pour créer l'algorithme d'optimisation par essaim de particules. Un essaim de Formellement, un essaim de particules survole l'espace de recherche, à la recherche de l'optimum global, les particules représentent des solutions potentielles au problème d'optimisation. Le déplacement d'une particule est influencé par les trois composantes suivantes (voir figure 12) :

- Une composante physique : la particule tend à suivre sa direction courante de déplacement.
- Une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée en exploitant sa propre expérience dans l'essaim.
- Une composante sociale : la particule tend à se à se diriger vers le meilleur site déjà atteint par ses voisins, en exploitant l'expérience de ses voisins.
-

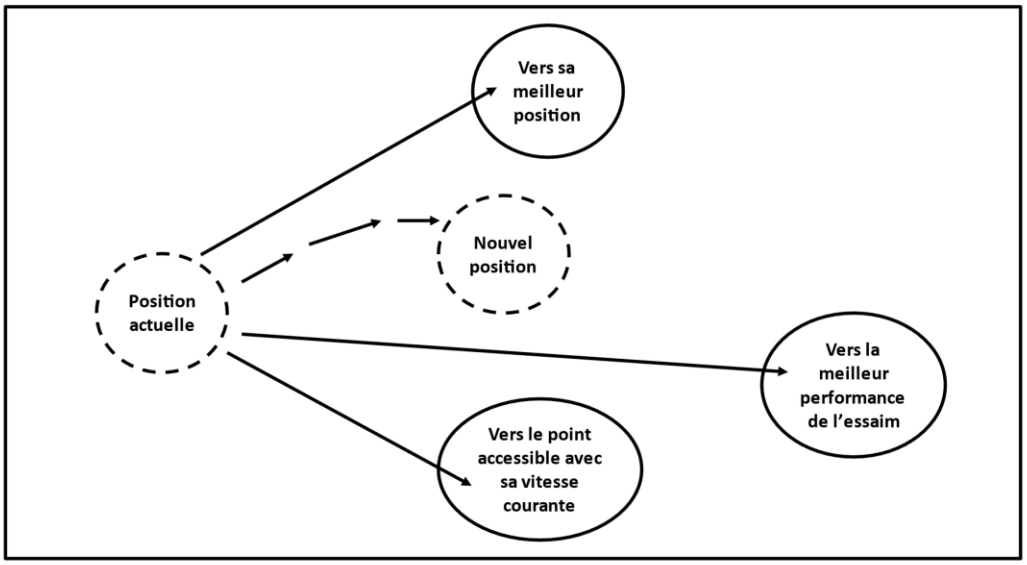
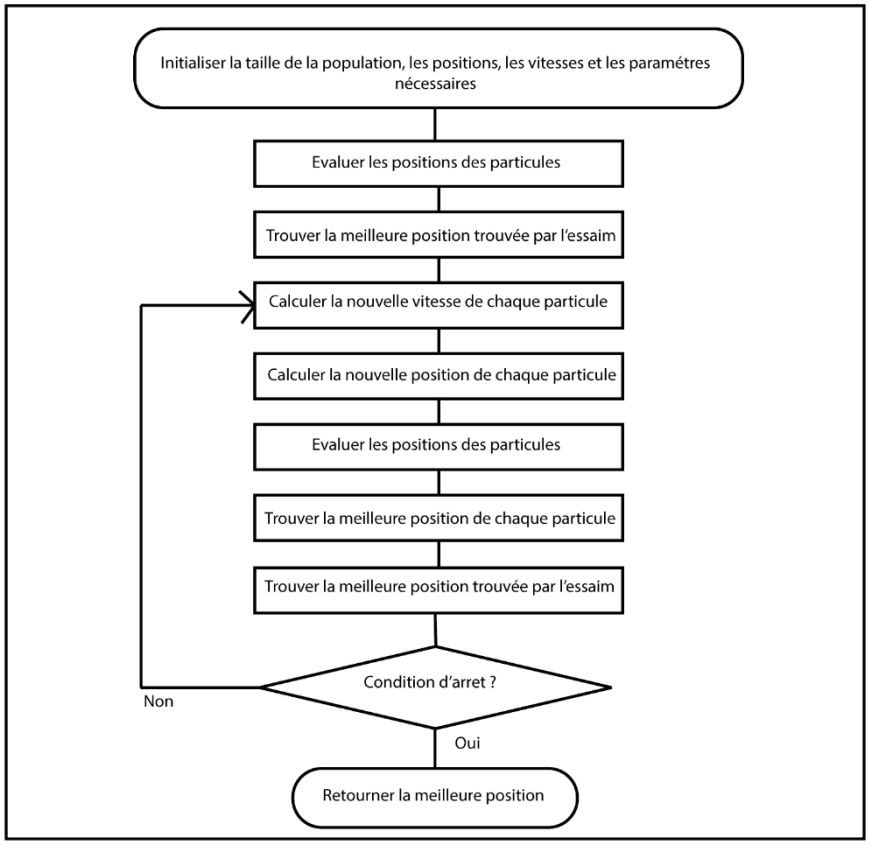


Figure 12.Déplacement d'une particule [30]

Au début de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également une position et une vitesse aléatoires. Ensuite, à chaque itération (Figure 12) :

- Chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance, c'est-à-dire la meilleure position par laquelle elle est déjà passée et elle a tendance à retourner vers cette position.
- Chaque particule est capable d'interroger un certain nombre de ses voisins (ses informatrices) et d'obtenir de chacune d'entre elles sa propre meilleure performance. Autrement dit, chaque particule est informée de la meilleure position connue au sein de son voisinage et elle va tendre à aller vers cette position
- Chaque particule choisit la meilleure des meilleures performances dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence. Une fois la meilleure informatrice détectée, la modification de la vitesse est une simple combinaison linéaire de trois tendances, à l'aide de coefficients de confiance :
    - la tendance « aventureuse », consistant à continuer selon la vitesse actuelle,
    - la tendance « conservatrice », ramenant plus ou moins vers la meilleure position déjà trouvée,
    - la tendance « panurgienne », orientant approximativement vers la meilleure informatrice.



Algorithme 2 Algorithme optimisation de l'essaim de particules [31]

Le voisinage constitue la structure du réseau social des particules. Sa topologie définit avec qui chacune des particules va pouvoir communiquer. Il existe de nombreuses combinaisons dont les plus utilisées sont [32] :

- Topologie en étoile (Fig. 13(a)) : Chaque particule est reliée à toutes les autres. L'optimum du voisinage est l'optimum global.
- Topologie en anneau (Fig. 13(b)) : Chaque particule est reliée à n voisines immédiates (en général, n = 3). Elle tend à se déplacer vers la meilleure dans son voisinage local.
- Topologie en rayon (Fig. 13(c)) : Une particule "centrale" est connectée à toutes les autres. Seule cette particule centrale ajuste sa position vers la meilleure, si cela provoque une amélioration l'information est propagée aux autres.

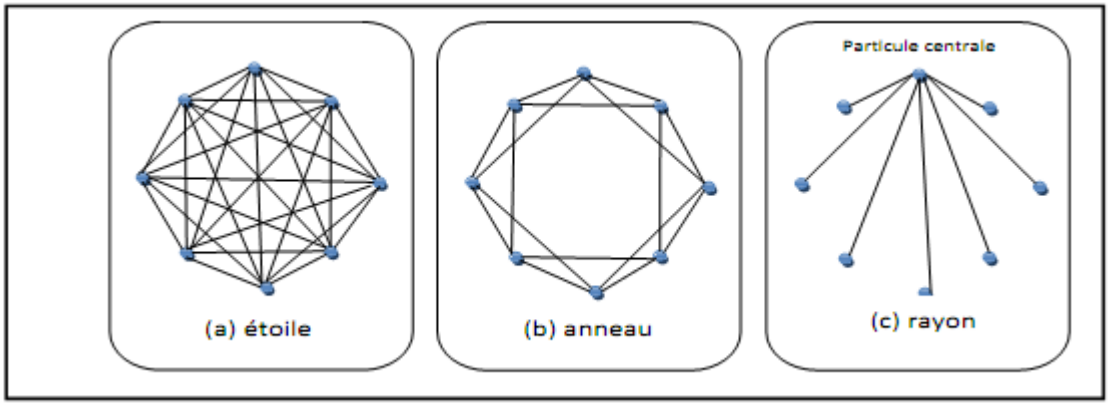


Figure 13 Trois topologies différentes [32]

b. Avantages et limites

L'algorithme OEP présentent plusieurs avantages tels que :

- Basé sur l'intelligence et peut être appliquée dans différents domaines de recherche et d'ingénierie.
- Facile et simple à programmer, quelques lignes de code suffisent dans n'importe quel langage évolué.
- Robuste vis-à-vis au réglage paramètres : un mauvais choix de paramètres dégrade les performances, mais n'empêche pas d'obtenir une solution.
- Basé sur la coopération plutôt que sur la compétition : une particule actuellement médiocre mérite d'être conservée, car c'est peut-être justement elle qui permettra le succès futur.
- Caractérisé par un très bon comportement, si multiples optima.
- OEP ne contient pas d'opérations de calcul basé la mutation ou le croisement, la recherche de solution est guidée par la vitesse de la particule. Durant plusieurs générations, seules les particules qui sont bien adaptés peuvent transmettre l'information aux autres, ce qui permettra une convergence rapide de la recherche ;
- Le point fondamental de la technique OEP, est le moyen de communication utilisé par les particules afin de s'échanger leur pbest.
- Simple, chaque particule est soumise à des règles de déplacement très simples, qui mènent cependant l'essaim à converger rapidement vers un optimum. C'est d'ailleurs un des points forts de cette méthode.

Cependant, l'OEP souffre du problème de convergence précoce : il peut se retrouver bloqué dans un optimum local, mais la modification de la topologie du réseau social permet souvent de résoudre ce problème ; De plus, il ne peut pas être utilisé en dehors des problèmes d'optimisation.

2.6 Conclusion

Dans ce chapitre nous avons procédé à l'étude des réseaux de capteurs sans fil. Nous avons posé les briques de base et fédéré quelques concepts nécessaires à la compréhension de

nos problématiques dans la suite de ce manuscrit. Puis, nous avons expliqué les concepts de base des problèmes l'optimisation et la classification des différentes méthodes utilisées dans leur résolution. Enfin, nous avons présenté brièvement les méta-heuristiques AG et OEP que nous allons utiliser dans notre travail pour résoudre le problème d’optimisation d’énergie dans les réseaux de capteurs. Cette contribution fait l’objet du chapitre suivant.

CHAPITRE 2 : Contribution

1.Introduction

Dans la section suivante, nous présentons les travaux qui concernent l'utilisation des méthodes formelles pour l'évaluation des performances des RCSFs. Ensuite nous proposons d'utiliser une heuristique pour obtenir une solution réalisable et de l'améliorer avec deux méta-heuristiques : Algorithme génétique et Essaim de particules.

2. Notre problème d'optimisation

L'objectif principal de notre projet de fin d'étude est de concevoir une solution basée sur les méta-heuristique pour trouver les valeurs optimales des paramètres de configuration de la technique N-vacance (proposé dans [3]) tel que le seuil (N), la taille de buffer (K), le taux moyen d'arrivé (λ) et le taux moyen de service (μ) des paquets. Ces valeurs optimales (N*, K*, λ*, μ*) permettent de minimiser la consommation d'énergie ainsi que de délai dans les RCSFs. En effet, la technique N-vacance efficace en énergie, augmente considérablement le délai d'attente dans les réseaux multi-sauts, parce que le délai d'attente augment avec l'augmentation du seuil N [1] [3]. Pour cela, nous devons optimiser (minimiser) une fonction objectif à deux critères (multi-objectif).

Formellement, notre problème d'optimisation est défini comme suit :

- Un ensemble des variables de décision X = {N, K, λ, μ} où N et K sont des naturels et λ, μ sont des réels.
- Un ensemble de solutions S = (N, K, λ, μ) qui est infini (dénombrable).
- Un ensemble de contraintes C où
  - K est compris entre 10 et 50
  - N est compris entre 1 et K-1
  - λ est entre 0.25 et 5
  - μ est entre 5 et 10 de type réel
  - K doit être strictement supérieur à N
  - μ doit être strictement supérieur à λ
- Un sous-ensemble SR de S représentant les solutions admissibles (ou réalisables) qui vérifient les contraintes C.
- Une fonction objectif f qui assigne à chaque solution s $\in$SR une valeur f(s). Cette fonction prend deux critères : l'énergie (E) et le délai (D). Cette fonction multi-objectif est transformée en une fonction mono-objectif en calculant une somme pondérée entre l'énergie standardisé (E) et le délai standardisé (D) tel que :
  - la standardisation permet d'éliminer les unités des mesures des données. Elle consiste à opérer une double transformation de centrage et de réduction sur les variables quantitatives pour comparer entre eux [33]. Pour se faire, nous devons calculer la moyenne de l'énergie et du délai ainsi que leurs écart-type. Pour chaque valeur de l'énergie et du délai, nous soustrairons la moyenne et nous divisons par l'écart-type comme indiqué dans les équations (9) et (10).
  - F(s) = a1 * E (s) + a2 * D(s)
  - Les équations du calcul d'énergie et de délai sont données dans le tableau 1

- Il s'agit de trouver une solution optimale (ou optimum global) s*∈ SR qui minimise la fonction objectif : ∀s ∈ SR, f (s∗) ≤ f (s)

Tableau 1 Calcul de la fonction objectif

Rôle	**Equation**
Calculer la durée moyenne d'une période d'état oisif $\bar{I}$	$\bar{I}=N/\lambda$……. (Équation 1)
Calculer le nombre moyen de paquets dans un nœud de capteur $\bar{Q}$	$\sum_{j=0}^{1}\sum_{i=0}^{k}\pi_{i,j}\ .i+\sum_{i=1}^{N-1}\pi_{i,1}\ .i.$ …….(Équation 2)
Calculer la durée moyenne d'une période d'état occupé B	$\bar{B}=\bar{Q}/\mu$…….(Équation 3)
Calculer la durée moyenne d'un cycle $\bar{C}$	$\bar{C}=\bar{I}+B$…….(Équation 4)
Calculer le débit en réception des paquets $\bar{\lambda}$	$\bar{\lambda}=\lambda\,.\left(\sum_{j=0}^{1}\sum_{i=0}^{K-1}\pi_{i,j}\ +\sum_{i=1}^{N-1}\pi_{i,1}\right)$ …….(Équation 5)
Calculer le délai moyen d'attente des paquets $\bar{W}$	$\bar{W}=\bar{Q}/\bar{\lambda}$ …….(Équation 6)
Calculer la probabilité d'être à l'état oisif P_I	$P_I=\sum_{0}^{N-1}\pi_{i,0}$…….(Équation 7)
Calculer la consommation d'énergie moyenne à un nœud capteur EC	$EC\ =EC_i\ .P_i+EC_b\ .(1-P_i)+\frac{EC_s}{\bar{C}}+\bar{Q}\,.E_{ch}$ …….(Équation 8) où $P_B=1-P_i$ La probabilité d'état occupée ECi : La consommation d'énergie de l'unité de transmission à l'état idle. ECb : La consommation d'énergie de l'unité de transmission à l'état busy. ECs : La consommation d'énergie du changement de l'état de capteur entre réception et transmission. E_{ch} : La consommation d'énergie de transmission des paquets.
Calculer l'énergie centré réduit	$E=\frac{\lvert Ec-Ec_{Moy}\rvert}{Ecartype(EC)}$…….(Équation 9)
Calculer le délai centré réduit	$D=\frac{\lvert W-W_{Moy}\rvert}{Ecartype(w)}$…….(Équation 10)

En résumé, notre problème d'optimisation est un problème

- Mixte car les variables N et K sont discrètes et les variables λ, μ sont continues.
- Avec contrainte : des bornes et deux équations inégalité sur les variables de décision.
- Multi-objectifs que nous avons reformulé en un problème mono-objectif
- Linéaire car la fonction objectif et les contraintes sont linéaires.

3. Méthodes de résolution

Selon la classification de notre problème d'optimisation, plusieurs méthodes approchées peuvent être utilisées pour le résoudre. Dans [3], Boutoumi et Gharibi ont déjà utilisé une heuristique aléatoire qui passe par trois étapes (Algorithme 3 :

- étape 1 : génération aléatoire des paramètres de configuration en utilisant une fonction mathématique aléatoire
- étape 2 : Calcul de l'énergie et de délai
- étape 3 : enregistrement des résultats

Initialisation	• Lire n (nombre d'itération)
Etape 1	• K ← Rand (1, 50); N ← Rand (1, K-1); λ ← Rand (0,25, 5); μ ← Rand (5, 10) • Si (K > N) et (μ > λ) aller à l'étape 2; Sinon aller à l'étape 1
Etape 2	• (1) Calculer la durée moyenne d'une période d'état oisif $\bar{I}$ • (2) Calculer le nombre moyen de paquets dans un nœud de capteur $\bar{Q}$ • (3) Calculer la durée moyenne d'une période d'état occupé B • (1) + (3) → (4) Calculer la durée moyenne d'un cycle $\bar{c}$ • (5) Calculer le débit en réception des paquets λ • **(2) + (5) → (6) Calculer le délai moyen d'attente des paquets W** • (7) Calculer la probabilité d'être a l'état oisif PI • **(2) + (4) + (7) → (8) Calculer la consommation d'énergie moyenne à un nœud capteur EC**
Etape 3	• Enregistrer les résultats (6) et (8) • Si le nombre d'itération n'est pas atteint, aller à l'étape 1; Sinon, fin de l'heuristique

Algorithme 3. Heuristique Aléatoire

Ces étapes sont répétées plusieurs fois (n étant le nombre d'itération qui est donné au début). Cependant, cette heuristique fait une recherche aléatoire, accepte et enregistre toute solution réalisable trouvée. Elle n'est pas guidée vers une solution optimale car elle ne calcule pas la fonction objectif. Dans notre solution, nous proposons de guider la recherche vers l'optimum globale en utilisant les méta-heuristiques à base de de population suivantes : l'Algorithme Génétique (Genetic Algorithm, GA) et l'optimisation par essaim de Particles (Particle Swarm Optimization, PSO). Car, ces méta-heuristiques peuvent aider à élargir l'espace de recherche d'une heuristique et permettre d'optimiser davantage les performances d'une heuristique en explorant l'espace de recherche de manière plus exhaustive et intelligente pour aider à trouver des solutions de meilleure qualité ou à atteindre des résultats plus proches de l'optimalité et d'éviter les solutions sous-optimales et d'obtenir de meilleures performances par rapport à une heuristique.

4. Application de l'algorithme génétique (AG)

L'application de AG pour le problème de minimisation de l'énergie et de délai commence par définir le codage convenable des individus de la population. Chaque individu (une solution réalisable à notre problème) est représenté par un chromosome qui est composé de quatre gènes où chaque gène exprime un paramètre (N, K, λ et μ). Etant donné que ces paramètres sont

continus, nous optons pour un codage réel qui représente les solutions par des suites de type réel, ex. [30, 17, 0.26, 6.53].

Ensuite, nous devons définir l'opérateur de la sélection qui s'intervient dans deux étapes de chaque itération :

- Au début de chaque itération pour sélectionner les individus parents qui vont se reproduire entre eux.
- A la fin de chaque itération pour sélectionner les individus enfants qui vont remplacer les individus parents dans le but de créer une nouvelle population en respectant la taille de la population pour ne pas entrainer une explosion démographique de la population.

La sélection des individus dépend essentiellement de leurs qualités (fitness). En effet, les meilleurs individus sont sélectionnés pour se reproduire, survivre et donc remplacer les moins performants.

Puis, dans la phase de reproduction, nous avons opté pour :

- Le croisement à 2 points qui consiste à choisir deux points de coupure, puis échanger les fragments situés entre ces deux points comme le montre la figure suivante :

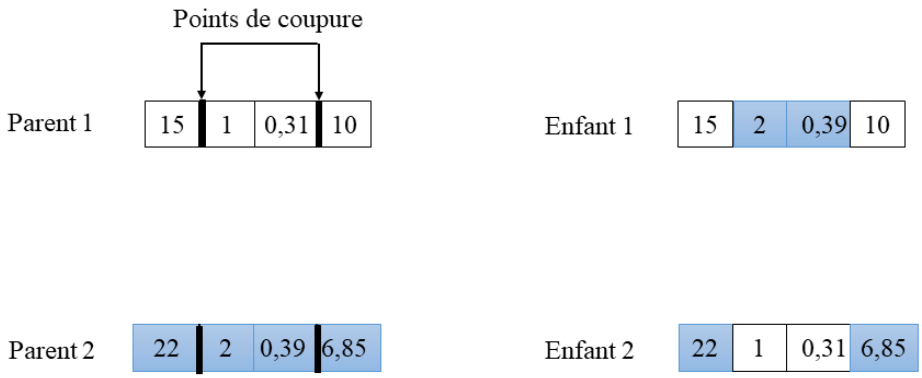


Figure 14 Croissement 2-points

- La mutation quant à elle consiste à modifier les propriétés d'un *individu*. Etant donné que nos gènes sont de type mixte et bornée alors nous ne pouvons pas insérer ou échanger ou reverser les gènes. Ainsi, nous avons opté pour multiplier la valeur d'un seul gène par un nombre aléatoire tout en respectant les contraintes.

Après croisement et mutation des individus, le nombre d'individus de la population augmente et la taille de cette dernière dépasse sa limite. En fait, la phase de reproduction permet de créer une nouvelle population composée de deux groupes d'individus : parents et enfants. La phase de remplacement permet de décider quels sont les individus qui vont représenter la nouvelle population. Dans notre algorithme, nous allons sacrifier par les individus de mauvaise qualité, autrement dit, nous allons remplacer un parent de mauvaise qualité par un enfant de bonne qualité.

L'algorithme 4 représente l'ensemble des étapes de l'algorithme génétique.

Etape 1: initialisation	1. Définir le nombre maximum d'itération (max) et la taille de la population (Np) 2. Initialiser les paramètres nécessaires (M, pc, pm, ,,); 3. Initialiser une population de Np individus ; **chaque individu doit vérifier les contraintes C.** 4. Evaluer les Np individus ;
Etape 2: Boucle	5. Tant que (le nombre maximum d'itérations n'est pas atteint) faire a. Utiliser l'opérateur de sélection pour sélectionner M individus ; b. Appliquer l'opérateur de croisement sur les M individus avec la probabilité Pc ; c. Appliquer l'opérateur de mutation sur les individus avec la probabilité Pm ; **chaque individu doit vérifier les contraintes C.** d. Utiliser l'opérateur d'évaluation pour évaluer les enfants obtenus ; e. Utiliser l'opérateur de sélection pour remplacer quelques individus parents par des individus enfants ;
Etape 3: Fin	6. Enregistrer la meilleure solution (individu) trouvée

Algorithme 4 Déroulement de l'algorithme AG

5. Application de l'optimisation par essaim de particules (OEP)

L'application de l'OEP pour le problème de minimisation de l'énergie et du délai commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. Un essaim est composé d'un ensemble de particules qui représentent chacune une solution dans un espace de recherche de dimension D (D = 4). Chaque particule i de l'essaim i $\in \{1, \dots, N_p\}$ où Np est la taille de l'essaim) est modélisée par son vecteur de position courante xi = (xi1, xi2, xi3, xi4) = (N, K, λ, μ) et son vecteur vitesse de déplacement vi= (vi1, vi2, vi3, vi4). La qualité de sa position est déterminée par la valeur de la fonction objectif f en ce point xi. F(xi) = 0.5 * E(xi) + 0.5 * D(xi) Chaque particule mémorise la meilleure position par laquelle elle est déjà passée, que l'on note pbesti= (pbesti1, pbesti2, pbesti3, pbesti4), ainsi que la meilleure position atteinte par ses particules voisins l'essaim, notée gbest = (gbest1, gbest2, gbest3, gbest4). Chaque particule va se déplacer entre les deux itérations t et t+1, en fonction de sa vitesse et des deux meilleures positions qu'elle connaît (la sienne et celle de ses voisins[1]) suivant les deux équations suivantes [34]:

$$V_{id}(t+1)= c0\ V_{id}(t)+c1\ r1(P_{bestid}(t) - X_{id}(t))+c2\ r2\ (g_{bestid}(t) - X_{id}(t)) \quad \text{.....(Équation 11)}$$

$$X_{id}(t+1)= X_{id}(t) +V_{id}(t+1) \quad \text{.....(Équation 12)}$$

Où

- c0 est en général une constante appelée, coefficient d'inertie. Il est choisi entre 0 et 1 tout en sachant que plus il est proche de 1 plus l'algorithme favorisera l'exploration de l'espace des solutions plutôt que l'exploitation et la convergence vers les bonnes solutions remarquées.
- c1 et c2 sont deux constantes, appelées coefficients d'accélération ou d'attraction,
- r1 et r2 sont deux nombres aléatoires tirés uniformément dans [0,1[

[1] Nous choisissons la topologie de voisinage social global (topologie en étoile). La particule est ainsi attirée vers la meilleure particule (l'optimum global)

- à chaque itération, et pour chaque dimension, xid (t) et vid (t) représentent respectivement la position et la vitesse de la particule i sur la dimension d à l'instant t,
- Pbestid (t) et Gbestd(t) représentent respectivement la meilleure position obtenue par la particule i et la meilleure position trouvée par l'essaim sur la dimension d à l'instant t.
- c0 vid (t) correspond à la composante physique du déplacement. Le paramètre c0 contrôle l'influence de la direction de déplacement sur le déplacement futur.
- c1 r1 (pbestid (t) - xid (t)) correspond à la composante cognitive du déplacement. Le paramètre c1 contrôle le comportement cognitif de la particule.
- c2 r2 (gbestd (t) - xid (t)) correspond à la composante sociale du déplacement. Le paramètre c2 contrôle l'aptitude sociale de la particule.

Le pseudo algorithme d'OEP est présenté ci-dessous.

Etape 1: initialisation	1. Définir le nombre maximum d'itération (max) et la taille pour l'essaim (Np) 2. Initialiser aléatoirement la position (Xi), la meilleure position (Pbesti←Xi) et la vitesse (Vi) pour chaque particule i. **Chaque position doit vérifier les contraintes C.** 3. Evaluer les positions des particules (calculer F(xi)) et calculer la meilleure position dans l'essaim (Gbest) 4. Initialiser les coefficients d'inertie et d'attraction (c0, c1, c2)
Etape 2: Boucle	5. Tant que le nombre maximum d'itérations n'est pas atteint) faire Pour chaque particule i =1 à Np faire a. Changer sa vitesse en fonction de (Équation 11). b. Se déplacer à une nouvelle position en utilisant (Équation 12). c. Evaluer sa performance F en utilisant sa position courante *X*i et la comparer avec ➢ Sa meilleure performance personnelle: Si F(Xi) < F(Pbesti) alors Pbesti←Xi ➢ La meilleure performance de l'essaim: Si F(Xi) < F(Gbest) alors Gbest←Xi
Etape 3: Fin	6. Enregistrer la meilleure solution (position) trouvée

Algorithme 5 Déroulement de l'algorithme OEP

6. Conclusion

Dans ce chapitre, nous avons présenté notre contribution pour résoudre le problème d'optimisation d'énergie et de délai dans les RCSF qui se base sur les méta-heuristiques AG et OEP. Nous avons présenté comment ces algorithmes sont appliqués pour trouver les valeurs optimales des paramètres de la politique N-vacance. Dans le prochain chapitre, nous allons faire une étude expérimentale et nous discuterons les résultats obtenus.

CHAPITRE 3 : Réalisation, Tests et Résultats

1.Introduction

Ce chapitre est principalement consacré aux expériences réalisées dans le cadre de notre travail. Nous commençons par décrire les outils utilisés pour le déploiement de notre solution. Ensuite, nous présentons et expliquons les résultats obtenus après avoir appliqué les algorithmes AG et OEP pour trouver les valeurs optimales de la techniques N-vacance.

2. Environnement de développement

Afin de déployer notre solution, nous avons utilisés deux ordinateurs portables (PC) qui ont les caractéristiques suivantes :

Tableau 2 Les caractéristiques des 2 PC utilisés

	PC1	PC2
Marque	Lenovo	HP
Processeur	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz	ntel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
RAM	8,00 Go	4,00 Go
Système d'exploitation	Windows 10 professionnel 64 bits	Windows 10 professionnel 64 bits

Sur ces PCs, nous avons installé le langage de programmation Python[2] et l'environnement MATLAB[3] avec l'environnement Microsoft Visual Studio Code 2023[4].

Pour connecter Python avec MATLAB, nous avons suivi ces étapes :

- ✓ Installer MATLAB Engine API pour Python en utilisant la commande pip ou un script d'installation Python setup.py.
- ✓ Importer le module matlab.engine dans le script Python.
- ✓ Démarrer une session MATLAB en appelant la fonction matlab.engine.start_matlab().
- ✓ Appeler des fonctions MATLAB depuis Python en utilisant la syntaxe suivante : matlab.engine.eval('commande MATLAB').

[2] Python [https://www.python.org/] est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées au typage dynamique et à la liaison dynamique, le rendent très attrayant pour le développement rapide d'applications.

[3] L'environnement MATLAB (MATrix LABoratory) [https://www.mathworks.com] est une plate-forme de programmation conçue spécifiquement pour les ingénieurs et les scientifiques afin d'analyser et de concevoir des systèmes et des produits qui transforment notre monde. Le cœur de MATLAB est le langage MATLAB, un langage matriciel permettant l'expression la plus naturelle des mathématiques computationnelles.

[4] Microsoft Visual Studio Code [https://code.visualstudio.com/] est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages et runtimes (tels que C++, C#, Java, Python, PHP, Go, .NET).

- ✓ Appeler des fonctions Python depuis MATLAB en utilisant la syntaxe suivante : python (‘commande Python’).

Python dispose d'un écosystème riche et dynamique avec une vaste gamme de bibliothèques et de frameworks qui facilitent le développement des méta-heuristiques comme la bibliothèque DEAP qui est un nouveau cadre de calcul des métaheuristique à base de population évolutif pour le prototypage et le test rapides d'idées. Elle cherche à rendre les algorithmes implicites et les structures de données transparentes. Le tableau suivant résume les principaux modules et classes DEAP que nous avons utilisé.

Tableau 3 Les modules et les classes DEAP utilisés

	Description	Méta-heuristique
Module base	Il contient les éléments de base qui sont utilisés pour construire des algorithmes à base de population tels que les individus, les particules, les populations et les fonctions de fitness.	AG, OEP
Module Creator	Il est utilisé pour définir de nouveaux types d’objets tels que les individus, les particules et les fonctions de fitness.	AG, OEP
Module Tools	Il contient des fonctions utiles pour la création d’algorithmes évolutifs tels que la sélection, la mutation et le croisement.	AG
Boîte à outils toolbox	Elle contient des fonctions et des classes pour la création d’algorithmes évolutifs. Elle est utilisée pour stocker les fonctions de sélection, de mutation et de croisement ainsi que les paramètres de l’algorithme.	AG,OEP

3. Implémentation des méta-heuristiques

En utilisant DEAP, nous avons implémenté AG comme suit :

- Créer des nouvelles classes :
  - **creator.create("FitnessMin", base.Fitness, weights=(-0.5,-0.5))** permet de créer un nouvel objet de type FitnessMin qui est une sous-classe de la classe Fitness du module base. Les poids (-0.5,-0.5) sont utilisés pour calculer la somme pondérée de la fonction objectif fitness (F(s) = 0.5 * E (s) + 0.5 * D(s)). Le signe (-) pour indique qu’il s’agit de problème de minisation.
  - **creator.create("Individual", list, fitness=creator.FitnessMin)** permet de créer un nouvel objet de type Individual qui est une sous-classe de la classe list. Cet objet a un attribut fitness qui est un objet de type FitnessMin.
  - **toolbox.register("individualCreator", tools.initCycle, creator.Individual,(toolbox.attr_intk, toolbox.attr_intN,toolbox.attr_fltlamb,toolbox.attr_fltmu), n=1)** permet d’enregistrer la fonction individualCreator dans la boîte à outils toolbox. Elle est

utilisée pour créer des individus pour l'algorithme génétique. Elle utilise la fonction initCycle du module tools pour initialiser les valeurs des individus, La valeur de n est définie à 1 pour créer un seul individu.

- **toolbox.register("population", tools.initRepeat, list, toolbox.individualCreator)** permet d'enregistrer la fonction population dans la boîte à outils toolbox, elle est utilisée pour créer une population d'individus, Elle utilise la fonction initRepeat du module tools pour initialiser les valeurs des individus. Les valeurs sont choisies à partir de la fonction individualCreator de la boîte à outils.
- **toolbox.register("select", tools.selTournament, tournsize=3)** permet d'enregistrer la fonction « select » dans la boîte à outils toolbox, Elle utilise la fonction selTournament du module tools pour effectuer la sélection. Le paramètre tournsize est utilisé pour définir la taille du tournoi, L'opérateur de sélection des individus pour la reproduction de la génération suivante : chaque individu de la génération actuelle est remplacé par le 'fittest' (meilleur) de trois individus tirés au sort de la génération actuelle.
- **toolbox.register("mate", tools.cxTwoPoint)** permet d'enregistrer la fonction « mate » dans la boîte à outils toolbox, elle est utilisée pour effectuer le croisement des individus. Elle utilise la fonction cxTwoPoint du module tools pour effectuer le croisement, Le croisement à deux points est un cas spécifique d'une technique de croisement à N points où deux points de croisement sont choisis au hasard à partir des chromosomes parents. Les bits entre les deux points sont échangés entre les organismes parents.

- Redéfinir l'opération de mutation : puisque les opérateurs de mutation disponibles dans le module DEAP ne sont pas adaptés pour notre problématique (des variables mixtes bornées), nous avons défini une nouvelle opération de mutation qui consiste à prendre un chromosome et modifier un gène de ce chromosome (le chromosome est composé de 4 champs d'où la probabilité de mutation dans un chromosome est 0.25).

De même, nous avons implémenté OEP en utilisant DEAP comme suit :

- Créer des nouvelles classes :
  - **creator.create("FitnessMin", base.Fitness, weights=(-0.5,-0.5))** permet rée un nouvel objet de type FitnessMin qui est une sous-classe de la classe Fitness du module base.
  - **creator.create("Particle", list, fitness=creator.FitnessMin, speed=list,smin=None, smax=None, best=None)** permet de crée une classe appelée Particle qui hérite de la classe list de Python. Elle définit les attributs fitness, speed, smin, smax et best. L'attribut fitness est défini comme une instance de la classe FitnessMin. Les autres attributs sont définis comme des listes vides.
  - **toolbox.register("particle", generate, size=4, pmin=[10, 1, 0.25, 5.0], pmax=[30, None, 5.0, 10.0], smin=-3, smax=3)** permet d'enregistrer la fonction « particle » qui crée une particule en utilisant la fonction generate pour initialiser les valeurs des particules : size, pmin, pmax, smin et smax.
  - **toolbox.register("swarm", tools.initRepeat, list, toolbox.particle)** permet d'enregistrer la fonction swarm pour initialiser un essaim de particules. Elle utilise la

fonction initRepeat du module tools pour initialiser les particules avec la fonction particle.

- **toolbox.register("update", updateParticle, phi1=2.0, phi2=2.0)** permet d'enregistrer la fonction « update » pour mettre à jour les particules de l'algorithme. Les paramètres phi1 (C1) et phi2 (C2) sont utilisés pour régler l'importance de la mémoire personnelle et de la mémoire globale dans la mise à jour des particules.

- Redéfinir les fonctions suivantes :
  - « def generate » pour créer une particule et initialiser ses attributs, à l'exception de l'attribut best, qui ne sera défini qu'après évaluation.
  - « update Particle() » pour calculer d'abord la vitesse puis limiter les valeurs de vitesse entre smin et smax, et enfin calculer la nouvelle position des particules.

4. Tests

Pour effectuer nos tests, nous avons pris les configurations suivantes ;

- **Configuration des paramètres du problème d'optimisation :**

Comme mentionné précédemment, l'énergie et le délai doivent être standardisés à chaque itération car ils possèdent des unités de mesures différentes ainsi ils ont des moyennes et des variances trop différentes. Avant de calculer la fonction objectif, nous devons appliquer la standardisation (centration-réduction) de l'énergie et du délai pour pouvoir les comparer (voir l'équation 9 et 10 du chapitre précédent). Pour se faire, nous devons calculer la moyenne de l'énergie et du délai ainsi que leurs écart-type. Nous avons utilisé l'échantillon généré par l'heuristique aléatoire (HA) du [3] pour calculer ces valeurs, ainsi nous fixons la moyenne de l'énergie à 423,127048, la moyenne du délai à 2,71471631, l'écart type de l'énergie à 146,535836 et l'écart type du délai à 1,75690995.

D'autre part, en raison de la simplicité, nous avons pris les poids pondérés de la fonction objectif égal à 0.5.

Tableau 4 Les valeurs prises pour le problème d'optimisation

Problème d'optimisation	Valeurs
Contraintes	K entre 1 et 30, N entre 1 et k-1, λ entre 0.25 et 5, μ entre 5 et 10.
Fonction objectif	a1 = a2 = 0.5
Normalisation d'énergie	Moy = 423,127048 écart type = 146,535836
Normalisation de délai	Moy = 2,71471631 écart type = 1,75690995

- **Configuration des méta-heuristiques :** faute du temps, nous avons pris arbitrairement les valeurs suivantes :

Tableau 5 Les valeurs prises pour le AG

Paramètres	Valeur
Nombre maximum d'itération (max)	20
Taille de la population (Np)	50
La taille de la sélection M	Aléatoire entre 1 et 50
Probabilité de mutation	De chaque chromosome (MUTPB) = 0.2 De chaque gène = 0.25
Probabilité de croisement	CXPB = 0.5

Tableau 6 Les valeurs prises pour le OEP

Paramètres	Valeur
Nombre maximum d'itération (max)	20
La taille d'essaim (Np)	50
coefficient d'inertie (C0)	1
coefficient d'attraction (C1 et C2)	2

5. Résultats

Dans cette section, nous allons présenter les résultats des trois méthodes (AG, OEP, HA) sous forme d'un tableau qui contient tous les résultats de cinq (5) meilleurs tests obtenus :

5.1 L'énergie :

5.1.1 HA :

Ce tableau contient la consommation moyenne de l'énergie des 5 meilleurs testes obtenue qui concerne l'Heuristique Aléatoire. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient l'Energie et la 6émé le temps d'exécution.

Tableau 7 Les résultats de l'énergie de HA

	K	N	λ	μ	EC	Temps d'exécution
Test 1	20	9	1	5	191,706853	10 m
Test 2	20	10	1	5	191,14726	10 m
Test 3	20	11	1	5	191,146265	10 m
Test 4	20	12	1	5	191,563684	10 m
Test 5	20	13	1	5	192,302609	10 m

5.1.2 AG :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme génétique. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient l'Energie et la 6émé le temps d'exécution.

Tableau 8 Les résultats de l'énergie de AG

	K	N	∧	μ	EC	Temps d'exécution
Test 1	50	4	0.25	10	87.451107	3h:36m:50.203821s
Test 2	10	5	0.25	10	86.227805	3h:54m:43.368271s
Test 3	10	7	0.25	10	86.977950	3h:24m:59.028475s
Test 4	10	8	0.25	10	88.151016	3h:17m:07.530217s
Test 5	10	6	0.25	10	86.248031	4h:44m:00.384246s

5.1.3 OEP :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme optimisation de l'essaim de particules. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient l'Energie et la6émé le temps d'exécution.

Tableau 9 Les résultats de l'énergie de OEP

	K	N	∧	μ	EC	Temps d'exécution
Test 1	26	5	0.25	10	86.227805	5h:37m:15.183309s
Test 2	29	5	0.25	10	86.227805	2h:32m:56.746929s
Test 3	29	5	0.25	10	86.227805	2h:39m:27.635484s
Test 4	33	5	0.25	10	86.227805	2h:34m:32.248662s
Test 5	30	5	0.25	10	86.227805	2h:50m:13.266681s

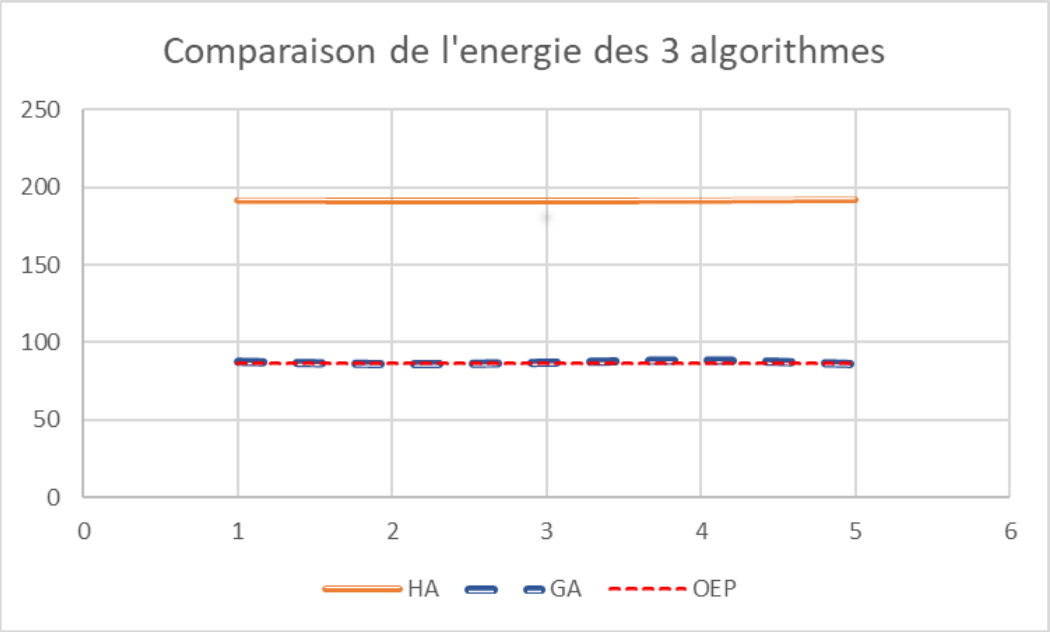


Figure 15 Comparaison de l'énergie des algorithmes pour les tests 1 à 5

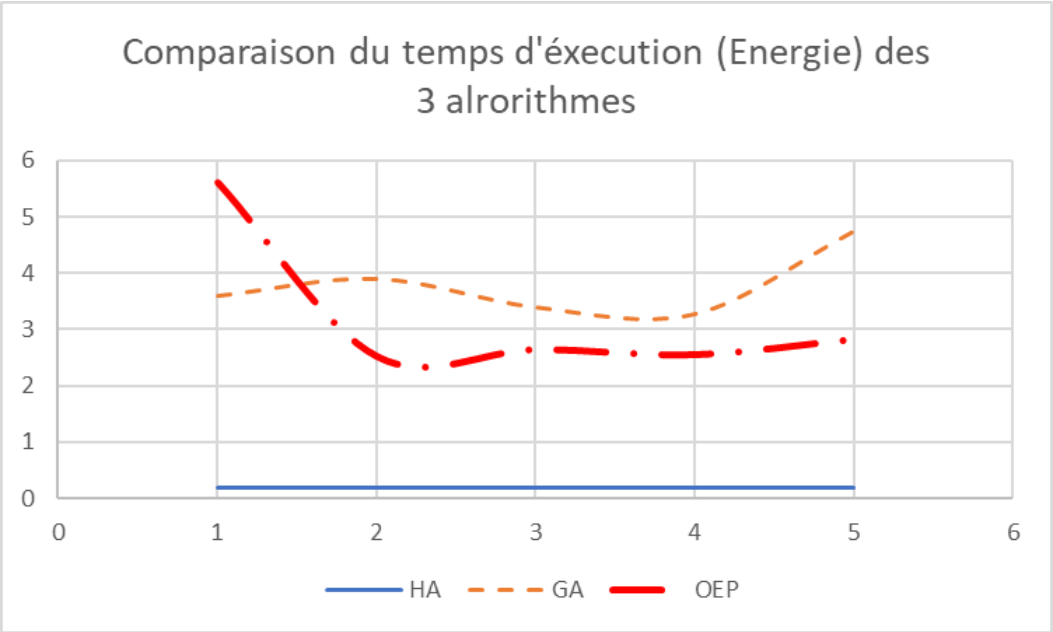


Figure 16 Comparaison du temps d'exécution (énergie) des algorithmes pour les tests 1 à 5

D'après les tableaux 7, 8 et 7, nous notons que nos résultats obtenues par les algorithmes à base de méta-heuristiques AG et OEP par rapport à l'énergie sont meilleurs que ceux obtenus par l'algorithme heuristique HA. En contre partie, l'algorithme HA présente un temps d'exécution inférieur aux autres algorithmes (AG et OEP), ceci est dû au cout induis par l'algorithme de calcul de l'énergie. AG et OEP sont des méta-heuristiques à base de population de solution, lancer l'algorithme de calcul de l'énergie 50 fois dans chaque itération augmente considérablement le temps d'exécution.

Les figures 15 et 16 illustre graphiquement ces comparaisons.

5.2 Le Délai :

5.2.1 HA :

Ce tableau contient le délai d'attente moyen des 5 meilleurs testes obtenue qui concerne l'Heuristique Aléatoire. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient le délai d'attente et la6émé le temps d'exécution.

Tableau 10 Les résultats du délai de HA

	K	**N**	**Λ**	**μ**	**W**	**Temps d'exécution**
Test 1	20	2	1.5	5	0,61904733	10 m
Test 2	20	2	3	5	0,66647966	10 m
Test 3	20	1	3	5	0,49985364	10 m
Test 4	20	2	1	5	0,75	10 m
Test 5	10	2	3	5	0,65176607	10

5.2.2 AG :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme génétique. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient le délai d'attente moyen et la6émé le temps d'exécution.

Tableau 11 Les résultats du délai de AG

	K	N	∧	μ	W	Temps d'exécution
Test 1	50	1	0.25	10	0.102564	3h:46m:57.461408s
Test 2	50	1	0.25	10	0.102564	3h:50m:09.908629s
Test 3	50	1	0.25	10	0.102564	4h:21m:31.851864s
Test 4	11	1	0.25	10	0.102564	3h:15m:27.014551s
Test 5	45	1	0.25	10	0.102564	3h:00m:04.704301s

5.2.3 OEP :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme optimisation de l'essaim de particules. Les 1ére colonnes contient les entrées de cet algorithme, la 5émé contient le délai d'attente moyen et la6émé le temps d'exécution.

Tableau 12 Les résultats du délai de OEP

	K	N	∧	μ	W	Temps d'exécution
Test 1	18	1	0.25	10	0.102564	2h:38m:33.644514s
Test 2	30	1	0.25	10	0.102564	2h:40m:32.972756s
Test 3	38	1	0.25	10	0.102564	2h:47m:14.254118s
Test 4	35	1	0.25	10	0.102564	2h:38m:13.210234s
Test 5	18	1	0.25	10	0.102564	2h:32m:53.143091s

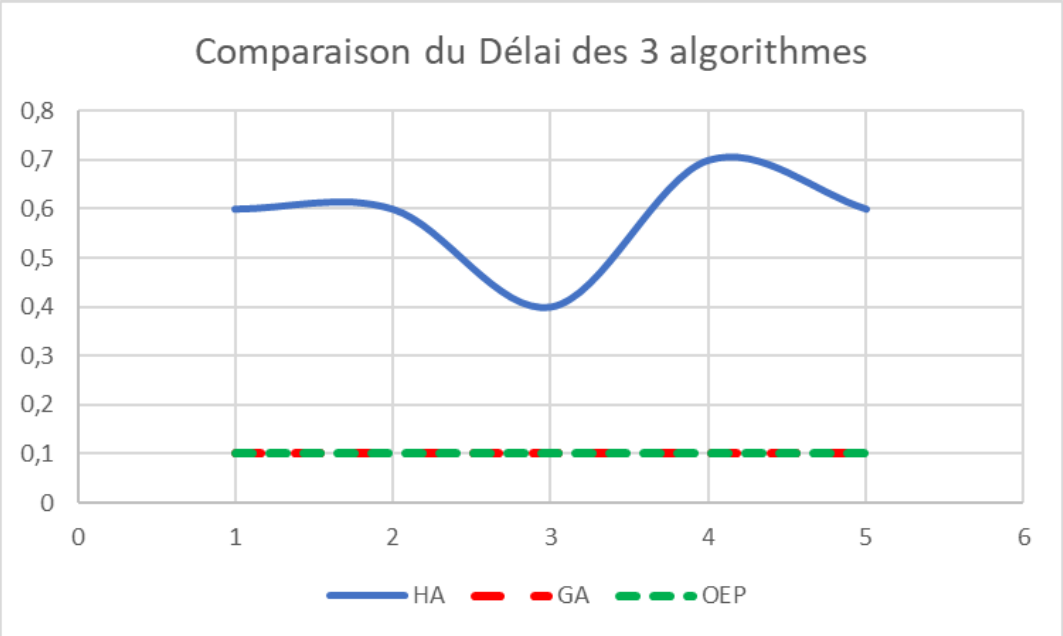


Figure 17 Comparaison du délai des algorithmes pour les tests 1 à 5

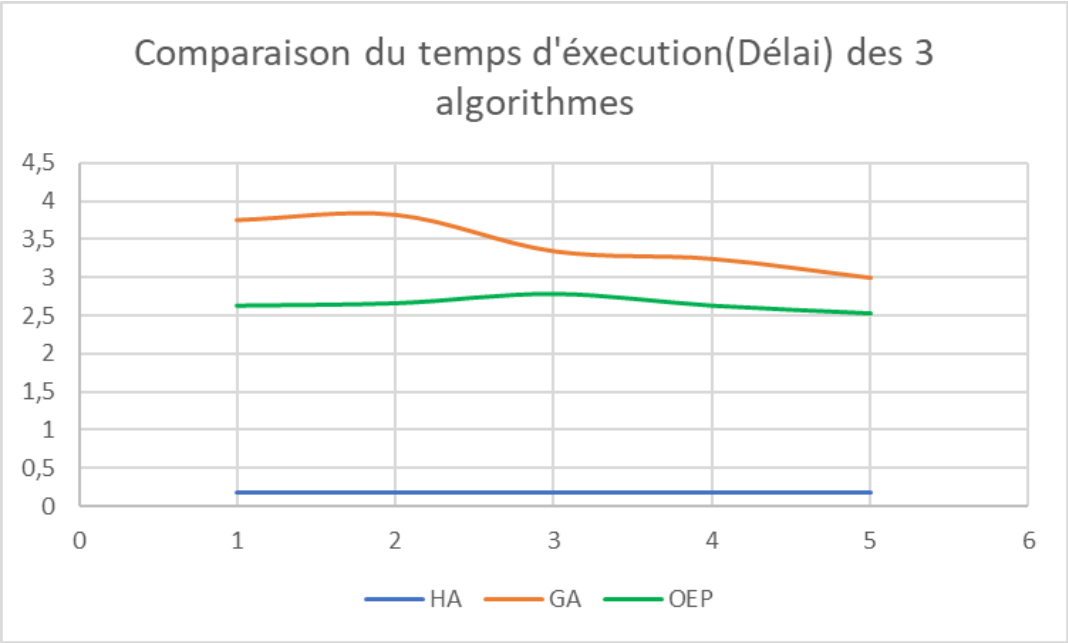


Figure 18 Comparaison du temps d'exécution (délai) des algorithmes pour les tests 1 à 5

D'après les tableaux 10, 11 et 12, nous notons que nos résultats obtenus par les algorithmes à base de méta-heuristiques AG et OEP par rapport au délai sont meilleurs que ceux obtenus par l'algorithme heuristique HA. En contre partie, l'algorithme HA présente un temps d'exécution inférieur aux autres algorithmes (AG et OEP), ceci est dû au cout induis par l'algorithme de calcul de l'énergie. AG et OEP sont des méta-heuristiques à base de population de solution, lancer l'algorithme de calcul de l'énergie 50 fois dans chaque itération augmente considérablement le temps d'exécution.

Les figures 17 et 18 illustre graphiquement ces comparaisons.

5.3 La Fonction Objectif :

5.3.1 HA :

Ce tableau contient tous les résultats de 5 meilleurs testes obtenue qui concerne l'Heuristique Aléatoire, Les 1ére colonnes contient les entrées de cet algorithme et le 5émé et le 6émé contient l'Energie et le délai et le 7émé et le 8éme contient l'Energie normalisée et le délai normalisé et la dernière colonne contient les valeurs de la fonction objectives.

Tableau 13 Les résultats de la fonction objective de HA

	K	**N**	**ʌ**	**µ**	**EC**	**W**	**E**	**D**	**Fct objectif**	**Temps d'exécution**
Test 1	20	14	3	5	193,035714	3,75	1,57020522	0,58926395	1,07973458	10 m
Test 2	20	15	3	5	191,706853	4,25	1,57927373	0,87385452	1,22656412	10 m
Test 3	20	13	3	5	191,14726	4,75000001	1,58309254	1,15844509	1,37076882	10 m
Test 4	20	16	3	5	191,146265	5,25000005	1,58309933	1,44303568	1,51306751	10 m
Test 5	20	12	3	5	191,563684	5,75000025	1,58025075	1,72762636	1,65393856	10

5.3.2 AG :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme génétique, Les 1ére colonnes contient les entrées de cet algorithme et le 5émé et le 6émé contient l'Energie et le délai et le 7émé et le 8éme contient l'Energie normalisée et le délai normalisé et la dernière colonne contient les valeurs de la fonction objectives.

Tableau 14 Les résultats de la fonction objective de AG

	K	**N**	**ʌ**	**µ**	**EC**	**W**	**E**	**D**	**Fct objectif**	**Temps d'exécution**
Test 1	50	19	3.17	5.05	422.89	3.37	0.00155944	0.37356209	0,187561	3h:58m:51.354036s
Test 2	10	4	2.76	6.32	423.61	0.82	0.00332372	1.07671241	0,540018	3h:37m:06.826908s
Test 3	41	7	3.85	7.73	422.58	1.03	0.00368367	0.95495137	0,479318	3h:19m:12.093560s
Test 4	50	45	3.88	7.35	423.30	5.95	0.00118499	1.84552221	0,923354	3h:20m:14.056841s
Test 5	18	12	4.77	8.63	423.05	1.40	0.0005195	0.74298265	0,371751	3h:08m:17.155404s

5.3.3 OEP :

Ce tableau contient les résultats de 5 tests qui concerne l'algorithme optimisation de l'essaim de particules, Les 1ére colonnes contient les entrées de cet algorithme et le 5émé et le 6émé contient l'Energie et le délai et le 7émé et le 8éme contient l'Energie normalisée et le délai normalisé et la dernière colonne contient les valeurs de la fonction objectives.

Tableau 15 Les résultats de la fonction objective de OEP

	K	**N**	**ʌ**	**µ**	**EC**	**W**	**E**	**D**	**Fct objectif**	**Temps d'exécution**
Test 1	10	9	4.58	8.69	422.75	1.09	0.00067936	0.92102341	0,460851	2h:32m:37.781539s
Test 2	37	21	5	8.59	396.64	2.43	0.00010014	0.24837028	0,124235	2h:33m:33.026826s
Test 3	28	1	1.05	7.02	425.11	0.16	0.01749283	1.44995267	0,733723	2h:34m:46.935583s
Test 4	36	10	5	9.85	423.19	1.10	0.00039513	0.91556204	0,457979	2h:37m:33.472420s
Test 5	27	19	3.15	5	423.38	3.39	0.00201649	0.38535802	0,193687	2h:33m:29.344581s

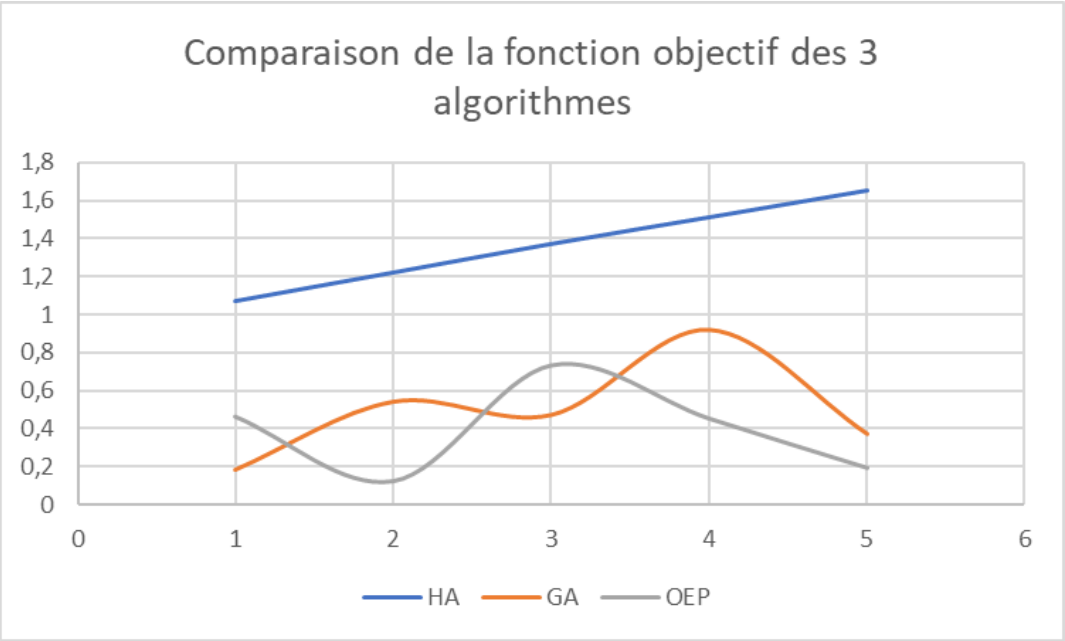


Figure 19 Comparaison de la fonction objective des algorithmes pour les tests 1 à 5

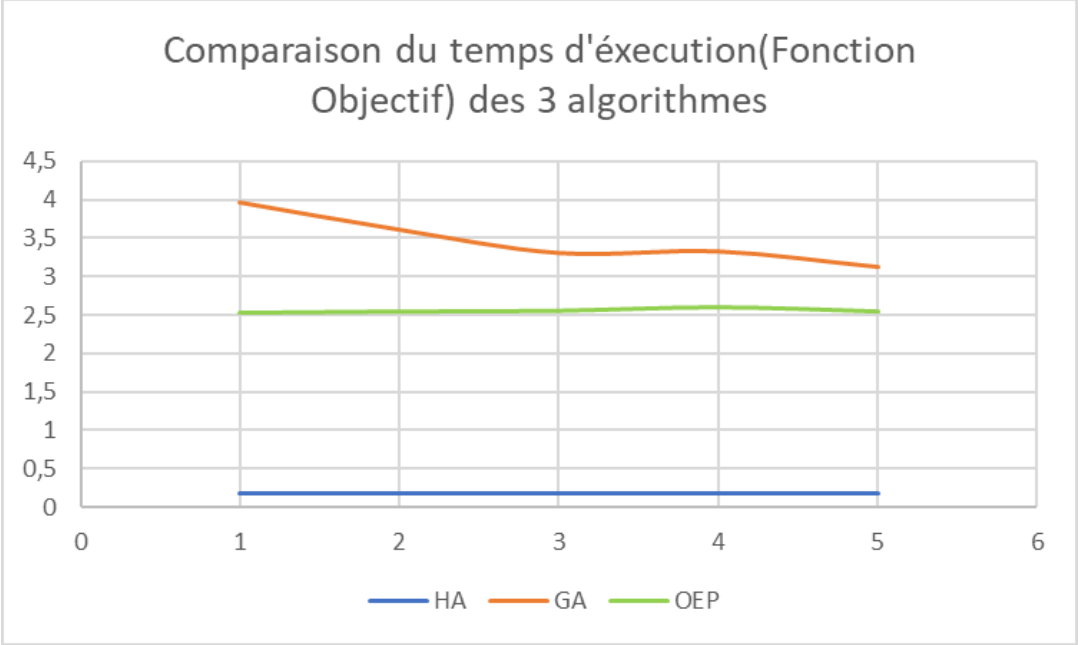


Figure 20 Comparaison du temps d'exécution (fonction objective) des algorithmes pour les tests 1 à 5

A travers les tableaux 13, 14 et 15 on peut dire que les résultats de la fonction objective pour minimiser l'énergie et le délai de HA sont meilleurs par rapport aux deux autres algorithmes AG et OEP. Les résultats de AG et OEP sont proches puisque les deux algorithmes sont des méta-heuristiques à base de population de solutions qui ont le même principe. De même, pour le temps d'exécution HA donne un temps d'exécution inférieur à AG et OEP.

Les figures 19 et 20 montre une représentation graphique des résultats des tableaux 13, 14 et 15.

6. Conclusion

Dans ce chapitre nous avons présenté l'environnement de développement ainsi que l'implémentation de HA, AG et OEP. Ensuite en ce qui concerne les résultats obtenus, en termes de l'Energie et en terme de délai et en terme de l'Energie et le délai(multi-objectives) le AG et OEP (les méta-heuristiques) elle est meilleur par rapport HA, Enfin ces résultats montrent l'intérêt d'utiliser les algorithmes de méta-heuristiques pour résoudre les problèmes d'optimisation dans les RCSF, Et dans le domaine des RCSF, l'utilisation de AG et OEP est recommandée pour l'optimisation d'économie d'énergie et le délai des nœuds capteurs.

.

Conclusion Générale& Perspectives

Dans la majorité des réseaux de capteur sans fils (RCSF), les nœuds capteurs opèrent avec des batteries irremplaçables. De ce fait, leurs sources d'énergie sont très limitées, et la durée de vie devient très courte à cause de l'épuisement d'énergie des nœuds capteurs. Une approche très importante pour optimiser la consommation d'énergie est d'utiliser des techniques de modélisation formelles basées sur les files d'attentes avec vacance et la politique N-vacance (N-Policy). Le problème de ces techniques est de trouver les valeurs optimales de leurs paramètres de configuration afin d'optimiser la consommation d'énergie. Ce problème d'optimisation peut être résolu par plusieurs méthodes approchées. Notre travail consistait à appliquer deux méta-heuristiques AG et OEP afin de minimiser la consommation d'énergie ainsi que de délai dans les RCSF.

Pour atteindre cet objectif, nous avons commencé par une étude bibliographique autour des RCSF où nous avons met l'accent sur le problème de la surconsommation d'énergie et des techniques existantes pour le résoudre. Ensuite, nous avons fait un état de l'art sur les problèmes d'optimisation et leurs méthodes de résolution. Cet état de l'art nous a permis de définir formellement notre problème d'optimisation d'énergie et d'appliquer les deux méta-heuristiques AG et OEP pour le résoudre. Cette contribution a été expliquée dans le chapitre 2 avant d'être mise en place et testée dans le chapitre 3. Les résultats obtenus montrent l'efficacité de ces deux algorithmes par rapport à l'heuristique aléatoire déjà utilisée.

Néanmoins, parmi nos perspectives nous envisageons :

- D'améliorer les résultats obtenus en raffinant les paramètres de configuration de la fonction objectif (poids de la somme pondérée), de l'algorithme génétique (taille de la population, probabilité de croisement, de mutation, de sélection……) et de l'algorithme par essaim de particules (taille de population, coefficient d'inertie, coefficient d'attraction……).
- De standardiser les données de la fonction multi-objectif (pour éliminer leur unité de mesure), en utilisant la population de l'itération actuelle.
- D'utiliser d'autres méthodes méta-heuristiques afin d'évaluer les performances des nœuds des capteurs et de les rendre plus efficaces.
- D'utiliser d'autres techniques de files d'attente avec N-vacances.

Bibliographies

[1] B. BOUTOUMI and N. GHARBI, “An Energy Saving and Latency Delay Efficiency Scheme for Wireless Sensor Networks Based on GSPNs”, The Fourth (4th) International Conference on Control, Decision and Information Technologies (CoDIT’17), Barcelone, Espagne, 5-7 April 2017.

[2] B. BOUTOUI and N. GHARBI, “Two Thresholds Working Vacation Policy for Improving Energy Consumption and Latency in WSNs”, The 13th International Conference on Queueing Theory and Network Applications (QTNA 2018) , Tsukuba, Japan, July 25-27, 2018.

[3] Boutoumi, B., & Gharbi, N. (2023, March). N-policy Priority Queueing Model for Energy and Delay Minimization in Wireless Sensor Networks Using Markov Chains. In 2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS) (pp. 1-6). IEEE.

[4] M. LEHSAINI, «Diffusion et couverture basées sur le clustering dans les réseaux de capteurs: application à la domotique. » , thèse de doctorat en informatique, Université A.B Tlemcen Faculté des Sciences pour l'Ingénieur && Université de Franche-Comté U.F.R Sciences et Techniques École Doctorale SPIM, 2009.

[5] Dessales, D. (2011). Conception d'un réseau de capteurs sans fil, faible consommation, dédié au diagnostic in-situ des performances des bâtiments en exploitation (Doctoral dissertation, Poitiers).

[6] Randrianarisaina, A. (2015). Modélisation de la consommation d’énergie En vue de la conception conjointe (matériel/logiciel) des applications embarquées. Application aux réseaux de capteurs sans fil (wsn) (Doctoral dissertation, UNIVERSITE DE NANTES).

[7] I. F. Akyilidiz, W. Su, Y. Sankarasubramaniam, and E.I. Cayirci "A survey on sensor networks". IEEE Communications magazin,Vol 40, No.8,pp. 102-116, August 2002.

[8] TOUAT, F., TOUAT, N. (2015). Application de détection de présence par réseau de capteurs sans fil Simulation dans l’environnement TinyOS. Thèse de Master, Université Mouloud MAMMERI de Tizi-Ouzou.

[9] YAHIAOUI, Y., ZAIDI, H. (2018). Evaluation des Performances d'un Mécanisme d'Economie d'Energie d'un Réseau de Capteurs Sans Fil. Thèse de Master, Université: USDB-Blida1.

[10] Jorio, A. (2015). Le Clustering basé sur la Classification Spectrale pour l'Optimisation d'Energie dans les Réseaux de Capteurs Sans Fil Homogènes.

[11] Sofiane, M. O. A. D. (2007). Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil. Master en informatique, Université: IFSIC-Rennes1, 2008.

[12] BOUAZZI, I. (2018). Optimisation d'accès au medium afin d'économiser de l'énergie dans les réseaux de capteurs sans fils(Doctoral dissertation, Université de Monastir).

[13] Curt Schurgers, Vlasios Tsiatsis, and Mani B. Srivastava. STEM: Topology management for energy efficient sensor networks. In Proceedings of the IEEE Aerospace Conference, volume 3, pages 7889, Big Sky, Montana, USA, 2002.

[14] Rahim KACIMI « Techniques de conservation d'énergie pour les réseaux des capteurs sans fil >> Doctorat de l'université de Toulouse. Septembre 2009.

[15] Mehmet C. Vuran, Özgür B. Akan, and Ian F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. Computer Networks, 45(3):245259, 2004.

[16] Jiang, F.C., Huang, D.C., Yang, C.T. and Wang, K.H.: Mitigation Techniques for the Energy Hole Problem in Sensor Nettworks Using N-policy M/G/1 Queuing Models. The International Conference on Frontier Computing, Theory, Technologies and Applications (IET '2010),pp.281-286 (2010).

[17] Jiang, F.C., Huang, D.C., Yang, C.T., Leu, F.Y.: Lifetime Elongation for Wireless Sensor Network Using Queue-Based approaches. Supercomputing, vol. 59(3), pp. 1312-1335 (2012).

[18] Huang, D.C., Tseng, H.C., Deng, D.J., Chao, H.C.: A Queue-Based Prolong Lifetime Methods for Wireless Sensor Node. Computer Communications, vol. 35(9), pp. 1098-1106 (2012).

[19] Huang, D.C. and Lee, J.H.: A Dynamic N Threshold Prolong Lifetime Method for Wireless Sensor Nodes. Mathematical and Computer Modelling, vol. 57(11), pp. 2731-2741 (2013).

[20] Nidhi, M., Goswami, V.: A Randomized-Policy Queueing Method to Prolong Lifetime of Wireless Sensor Networks. The 3rd International Conference on Advanced Computing, Networking and Informatics (ICACNI '2016). pp. 347-357 (2016).

[21] Yadin, M., and Naor, P.:Queueing systems with a removable service station. the Operational Research Society, vol. 14(4), pp.393-405 (1963).

[22] Aroussi Sana. Cours de Heuristiques et Méta-Heuristiques niveau M1. Master. USDB. 2022

[23] HADDI, H., AMEUR, N. (2022). Développement d'une méthode pour le problème de tournées de véhicules multi-dépôts. Thèse de Master. Université: USDB-Blida1.

[24] Feigenbaum, E. A. (s.d.). (Edirors). Computers and Though. McGraw-HillINC. 6, New York, 1963.

[25] Fred G. Future paths for integer programming and links to artificial intelligence. Computer Operations Research, 13, 553-549, 1986.

[26] Osman, Ibrahim & Laporte, Gilbert. (1996). Metaheuristics: A Bibliography. Annals of Operational Research. 63. 513-628.

[27] Blum, C. and Roli, A. (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys (CSUR), 35, 268-308.

[28] Holland, J. H., 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press.

[29] AMIRA Gherboudj, « Méthodes de résolution de problèmes difficiles académiques » Thèse de Doctorat LMD Spécialité : Informatique, 2013, p 55, 56, 57, 58, 59, 67, 68.

[30] Kennedy, J. a. (1995). Particle swarm optimisation. Proceedings of the IEEE International Conference Neural Networks, 4, pp. 1942-1995.

[31] KIMOUCHE Mohammed Amine et BENZID Moussa, « Les méthodes métaheuristiques pour l'optimisation en génie électrique ». Mémoire de Master en Electromécanique, université de Jijel, 2019, p10, 41, 42.

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

[33] Kevin Polisano. Cours de Statistiques niveau L1-L2. Licence. France. 2018.

[34] Shi, Y. and Eberhart, R. (1998) A Modified Particle Swarm Optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, 4-9 May 1998, 69-73.