République algérienne démocratique et populaire
Ministre de l'enseignement supérieur et de la recherche scientifique
UNIVERSITÉ SAAD DAHLAB BLIDA 1
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

MEMOIRE DE MASTER
En Informatique

Option : Systèmes Informatiques et Réseaux (SIR)

THÈME :

Les Métaheuristique pour l'Optimisation d'Energie et du Délai d'Attente dans les Réseaux de Capteur sans Fil.

Réalisé par
AMIRAT Lina
LABIDI Mourad

Encadré par
Mme BOUTOUMI Bachira
Mme AROUSSI Sana

Présenté devant
M. OULD KHAOUA Mohamed
Mme TOUBALINE Nesrine

Setenu le 02/07/2024

Remerciements

Après avoir rendu grâce au tout puissant pour la force et le courage qu'il nous a permis d'avoir pour terminer ce travail, nous tenons à adresser nos remerciements les plus sincères à toutes les personnes qui nous ont aidées, orientées et encouragées.

Nous exprimons nos profonds remerciements à nos promotrices, *Mme B. Boutoumi* et *Mme S. Aroussi*, pour la confiance qu'elles nous ont accordé en nous confiant ce projet, ainsi que pour l'aide précieuse qu'elles nous ont apporté. Leurs conseils avisés, leur patience, leurs encouragements et leur regard critique nous ont été d'une grande aide pour améliorer la qualité des différents chapitres de notre mémoire.

Nous remercions également *Mr M.Ould Khaoua* et *Mme N.Toubaline* pour nous avoir fait l'honneur d'accepter d'examiner notre travail.

Nous exprimons notre gratitude à *Mme F.Z.Zahra* pour ses encouragements et ses conseils précieux tout au long de notre projet.

Nous remercions également *Mlle A.Amirat* pour son suivi constant, son soutien indéfectible, et sa disponibilité. Son implication personnelle et ses retours constructifs ont été essentiels pour le bon déroulement de notre travail.

Un grand merci à *Mlle I.Isoleh* pour son aide inestimable et pour nous avoir généreusement prêté sa machine, ce qui nous a permis de mener à bien nos travaux.

Nous remercions chaleureusement tous les enseignants du département d'informatique de l'USDB1 que nous avons eu pendant notre cursus de Master en Systèmes Informatiques et Reseaux (SIR) pour les connaissances et le savoir qu'ils nous ont transmis pendant ces deux dernières années.

Enfin, nous remercions toutes les personnes qui ont contribué de prêt ou de loin à l'aboutissement de ce travail.

AMIRAT Lina.
LABIDI Mourad.

Dédicace

Je dédie ce travail

À l'homme de ma vie, mon exemple éternel, mon soutien moral, celui qui s'est toujours sacrifié pour me voir réussir, à toi papa, à qui je dis Merci.

À la source de mes efforts, ma vie et mon bonheur, maman que j'adore.

À mes chères et aimables sœurs Hiba, Sarra et Anfel, mon beau-frère Adel et mon neveu Racim.

À mon partenaire, Mourad, qui a tant donné pour que nous achevions ce travail et qui ne m'a jamais manqué de son soutien.

À tous mes amis, mes collègues d'études et toutes les personnes que je connais et j'apprécie et que je n'ai pas citées.

AMIRAT Lina

Dédicace

C'est avec une profonde gratitude et sincères mots que je dédie ce travail en premier lieu à Mes chères parents qui quel que soit l'objet qu'on essayera de leur offrir, il n'atteindra jamais ce que j'ai envie de leur dire et exprimer.

A ma chère MAMAN

La perle de mon existence, quelle brave dame que tu es, que de sacrifices consentis à mon égard afin que je progresse dans mes études. Je tombe en admiration devant la bonté de ton Cœur à nulle pareille.

Quels que soient mes caprices et mes écarts tu m'as toujours soutenue, trouvant les mots justes pour me ramener sur le bon chemin.

L'évènement que nous célébrons aujourd'hui t'est entièrement dédié. Que dieu te préserve santé et longue vie

A ma très chère sœur et son fils.

A toute ma famille, mes grands-parents, mes oncles et tantes maternelles, cousins et cousines.

A mes meilleurs amis.

A ceux qui m'ont supportée, encouragée.

A tous les membres du CSC Club.

Et spécialement à ma collègue Lina merci pour tes conseils et tes encouragements, mais aussi pour les bons moments qui ont contribué à rendre ces années inoubliables.

LABIDI Mourad.

Résumé

Les réseaux de capteurs sans fil (RCSFs) se composent de nombreux capteurs miniatures autonomes capables de collecter et de transmettre des données sur leur environnement. Ces capteurs, bien que performants, sont généralement alimentés par des batteries, ce qui entraîne un défi significatif en termes de consommation d'énergie et de durée de vie des dispositifs.

Dans ce mémoire, nous nous sommes penchés sur le problème crucial de la consommation d'énergie dans les RCSFs tout en cherchant à minimiser le délai d'attente des communications. Pour ce faire, nous avons implémenté une politique de vacances de travail à deux seuils (VTDS), une méthode de gestion de l'énergie innovante qui permet aux capteurs de se mettre en mode veille selon des seuils définis pour économiser de l'énergie. Parallèlement, nous avons exploité plusieurs métaheuristiques, telles que le recuit simulé, la recherche Tabou et l'optimisation par essaims particulaires, pour affiner et optimiser les paramètres de cette politique.

Les résultats d'évaluation de notre solution montrent une réduction significative de la consommation d'énergie, tout en maintenant des délais d'attente acceptables pour la traitement des paquets. De plus, l'optimisation par essaims de particulaires (OEP) s'est démarquée comme étant la méthode la plus efficace parmi les métaheuristiques testées. Ces résultats démontrent l'efficacité et la pertinence de notre approche pour améliorer la gestion énergétique des réseaux de capteurs sans fil.

Mots clés : Réseaux de capteurs sans fil (RCSFs), Consommation d'énergie, Politique de vacances de travail à deux seuils (VTDS), Métaheuristiques.

Abstract

Wireless sensor networks (WSNs) consist of numerous miniature autonomous sensors capable of collecting and transmitting environmental data. Despite their performance, these sensors are generally battery-powered, presenting a significant challenge in terms of energy consumption and device lifespan.

In this thesis, we focused on addressing the critical issue of energy consumption in WSNs while seeking to minimize communication latency. To achieve this, we implemented a Two-Threshold Working Vacation Policy (VTDS), an innovative energy management method that allows sensors to enter sleep mode based on defined thresholds to conserve energy. Additionally, we utilized several metaheuristics, such as simulated annealing, Tabu search, and particle swarm optimization (PSO), to refine and optimize the parameters of this policy.

The evaluation results of our solution demonstrate a significant reduction in energy consumption while maintaining acceptable packet processing delays. Notably, particle swarm optimization (PSO) emerged as the most effective method among the tested metaheuristics. These results highlight the efficiency and relevance of our approach for improving energy management in wireless sensor networks.

Keywords : Wireless Sensor Networks (WSNs), Energy Consumption, Two-Threshold Working Vacation Policy, Metaheuristics.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Les progrès récents de la microélectronique et des technologies de communication sans fil ont conduit à l'émergence de dispositifs miniaturisés et autonomes appelés nœuds capteurs, capables de surveiller et de collecter des données environnementales et physiques [1]. L'ensemble de ces nœuds, forment un Réseau de Capteurs Sans Fil (RCSF) ou Wireless Sensor Network (WSN), qui surveille une région ou un phénomène d'intérêt, et fournit des informations utiles par la combinaison des mesures prises par les différents capteurs et de les communiquer ensuite via le support sans fil.

De nombreux domaines d'application sont alors envisagés tels que la détection et la surveillance des désastres, le contrôle de l'environnement, la médecine et la santé, la domotique ... etc[2]. Ce qui caractérise les RCSFs par un déploiement dense et à grande échelle dans des environnements limités en termes de ressources. Les limites imposées sont la limitation des capacités de traitement, de stockage et surtout d'énergie car ils sont généralement alimentés par des piles où le changement ou le rechargement des batteries est difficile est parfois impossible en raison de l'emplacement des noeuds [1]. Il est donc crucial de mettre en place une stratégie efficace de gestion de l'énergie du réseau afin de prolonger sa durée de vie tout en minimisant la perte d'énergie.

Pour adresser le problème de la consommation d'énergie, diverses techniques ont été développées [3, 4, 5]. Parmi celles-ci, la politique de vacances de travail à deux seuils VTDS (Two Thresholds Working Vacation Policy) est particulièrement notable. Cette politique permet au capteur de basculer entre les états actif et inactif en fonction de deux seuils prédéfinis, N1 et N2, réduisant ainsi significativement sa consommation d'énergie [4]. Cependant, une augmentation des seuils N1 et N2 peut entraîner un accroissement du délai d'attente des paquets [4]. Il est donc impératif de trouver un équilibre optimal entre la réduction de la consommation d'énergie et le maintien d'un délai d'attente raisonnable. Ce problème est particulièrement difficile à résoudre car il relève de la catégorie des problèmes NP-difficiles, nécessitant une analyse approfondie et des méthodes d'optimisation pour identifier les valeurs optimales des paramètres.

Dans ce même contexte, l'objectif de ce projet de fin d'études de master est de proposer des solutions pour résoudre ce problème d'optimisation des paramètres de la politique de VTDS. L'application d'une méthode exacte étant coûteuse en termes de temps et de ressources, nous avons recours à des métaheuristiques, notamment le Recuit Simulé (RS) [6], la Recherche Tabou (RT) [7] et l'Optimisation par Essaim de Particules (OEP) [8]. En utilisant ces techniques avancées, nous visons à minimiser la consommation d'énergie tout en réduisant les délais d'attente, assurant ainsi une performance optimale des réseaux de

capteurs sans fil.

Après cette introduction, notre mémoire se présente sous forme de trois chapitres comme suit :

- Le premier chapitre définit les concepts fondamentaux des réseaux de capteurs sans fil et leurs défis, l'optimisation combinatoire, ainsi que les techniques de modélisation des systèmes.
- Le deuxième chapitre présente en détail notre contribution, couvrant le principe de la politique de vacance de travail à deux seuils (VTDS), sa modélisation en utilisant les chaînes de Markov, ainsi que l'adaptation des métaheuristiques à notre problème d'optimisation.
- Le troisième chapitre décrit en détail l'implémentation des différents systèmes développés, y compris la modélisation des capteurs par les chaînes de Markov et l'application des métaheuristiques. Il présente également les résultats des expérimentations menées pour évaluer l'efficacité de notre contribution.

Enfin, notre mémoire s'achève par une conclusion générale incluant quelques perspectives.

Chapitre 1

Etat de L'art

1.1 Introduction

Ce chapitre résume les principales terminologies, concepts et notions clés liées au projet de fin d'études. Il se divise en trois sections majeures visant à fournir une perspective approfondie sur les sujets abordés.

La première section établit les bases conceptuelles en présentant d'abord les généralités sur les réseaux de capteurs sans fil (RCSFs). Elle explore leurs nombreux domaines d'application variés avant d'identifier les problématiques inhérentes à ces réseaux, en mettant particulièrement l'accent sur la consommation énergétique qui représente un défi majeur. Cette section passe ensuite en revue les différentes solutions innovantes proposées dans la littérature scientifique pour relever les défis liés aux RCSFs, notamment en terme d'économies d'énergie.

La deuxième section est consacrée aux problèmes d'optimisation et à leurs méthodes de résolution. Elle aborde les concepts fondamentaux de l'optimisation, sa modélisation et sa classification. Un focus particulier est mis sur les métaheuristiques en tant qu'approches de résolution approchées pour les problèmes d'optimisation complexes et de grande taille.

La troisième section se concentrera sur la modélisation du système. Elle présentera les Réseaux de Pétri Stochastiques Généralisés (RdPSGs) et les Chaînes de Markov à Temps Continu (CMTCs), deux outils puissants pour représenter et analyser les dynamiques complexes des systèmes. Ces méthodes permettront de comprendre et de modéliser les comportements des réseaux de capteurs sans fil afin d'optimiser leurs performances.

1.2 Généralités Sur les Réseaux de Capteurs Sans Fil

1.2.1 Capteur 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 exécute les trois fonctions fondamentales suivantes : acquisition de données, traitement de ces données et transmission des résultats aux stations de base [1].

1.2.2 Architecture Matérielle d'un Nœud Capteur

Un nœud capteur est constitué de divers éléments ou modules, chacun dédié à une fonction spécifique. Il intègre également une source d'énergie. Comme représenté dans la figure 1.1, un capteur est formé de quatre unités fondamentales [1] :

- **Unité d'acquisition :** C'est une unité de capture ou d'acquisition des grandeurs physique tels que la température et la luminosité. Elle contient un convertisseur appelé Convertisseur Analogique Numérique (CAN) qui transforme les données analogiques en grandeurs numériques compréhensibles par l'unité de traitement.
- **Unité de traitement :** Elle supervise les procédures permettant au nœud de collaborer avec d'autres nœuds pour accomplir les tâches d'acquisition, tout en stockant les données collectées. Cette unité se compose de deux interfaces : la première liée à l'unité d'acquisition et la deuxième connectée au module de transmission. Généralement, elle est associée à des unités de stockage de type RAM (Random Access Memory), ROM (Read Only Memory) ou flash, en fonction de la plateforme utilisée et du type d'application du nœud capteur.
- **Unité de communication (Unité Radio) :** Cette unité est responsable de toutes les communications 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.
- **Unité d'énergie :** C'est une batterie qui fournit l'alimentation aux unités précédemment mentionnées. Elle est généralement ni rechargeable ni remplaçable. La limitation de la capacité énergétique au niveau des capteurs constitue la contrainte principale lors de la conception de protocoles pour les réseaux de capteurs.

On peut trouver d'autres composants additionnels qui peuvent être implantés dans un capteur, on cite parmi eux [1] :

- **Système de localisation :** Avec ce système, le nœud peut se localiser de manière autonome au sein de son réseau.
- **Système de régénération énergétique :** Il sert à reconstituer l'énergie consommée par un réapprovisionnement grâce à une source externe comme les cellules solaires, la vibration, etc.
- **Système de déplacement du 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.

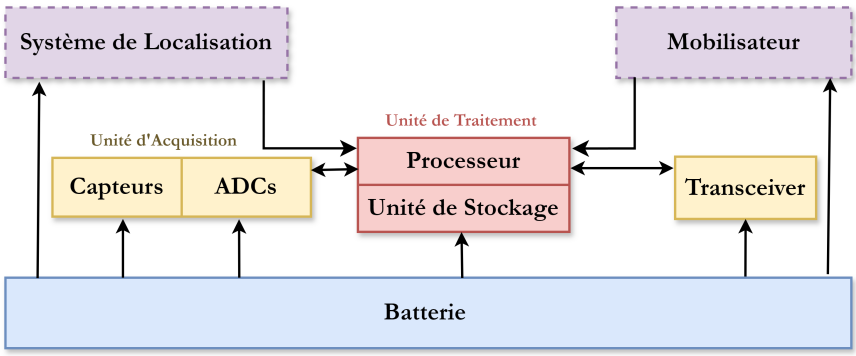

FIGURE 1.1 – Architecture physique d'un nœud capteur [1].

1.2.3 Réseau de Capteurs Sans Fil (RCSF)

Un réseau de capteurs sans fil (RCSF) dit *"Wireless Sensor Network : WSN"* est un type spécial de réseau comportant un grand nombre de nœuds (capteurs). La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils peuvent être aléatoirement dispersés dans une zone géographique, appelée champ de captage, zone de couverture ou zone d'intérêt [9]. Comme représentés dans la figure 1.2, un capteur surveillant son environnement capte et propage les données récoltées des capteurs appartenant à sa zone de couverture. De plus, ces nœuds sont reliés à une ou à plusieurs stations de base (SB) qui permettent l'interconnexion avec d'autres réseaux et la récupération des données. L'utilisateur du RCSF peut ainsi adresser des requêtes aux SB en précisant le type de données requises et recueillir puis analyser les événements recensés par les capteurs sur la zone de couverture [2].

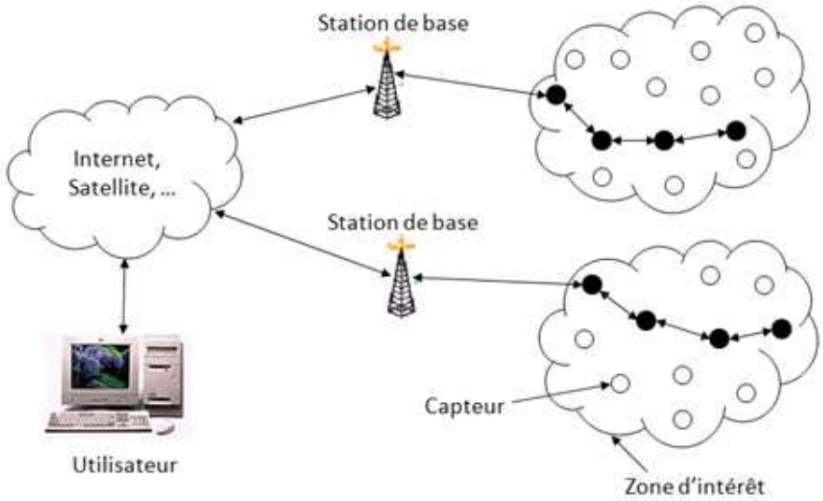

FIGURE 1.2 – Schéma général d'un réseau capteur sans fil [2].

1.2.3.1 Domaines d'applications des réseaux de capteurs

La miniaturisation, l'adaptabilité, le coût réduit et la communication sans fil ont permis aux réseaux de capteurs de pénétrer de nombreux domaines d'application et d'élargir les applications existantes [2].

- **Applications militaires :** Plusieurs projets ont été lancés pour aider les unités militaires dans un champ de bataille et protéger les villes contre des attaques, telles que les menaces terroristes [2, 10].
- **Applications à la surveillance :** L'application des réseaux de capteurs dans le domaine de la sécurité peut diminuer considérablement les dépenses financières consacrées à la sécurisation des lieux et des êtres humains. Le déploiement d'un réseau de capteurs de mouvement peut constituer un système d'alarme qui servira à détecter les intrusions dans une zone de surveillance.
- **Applications environnementales :** Le contrôle des paramètres environnementaux par les réseaux de capteurs peut donner naissance à plusieurs applications. Par exemple, le déploiement des thermo-capteurs dans une forêt peut aider à détecter un éventuel début de feu et par la suite faciliter la lutte contre les feux de forêt avant leur propagation.
- **Applications médicales :** Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, de rythme cardiaque, de l'attention,...etc.).
- **Domotique :** Les capteurs peuvent être embarqués dans des appareils, tels que les aspirateurs, les fours à micro-ondes, les réfrigérateurs, les magnétoscopes. Ces capteurs embarqués peuvent interagir entre eux et avec un réseau externe via Internet pour permettre à un utilisateur de contrôler les appareils domestiques localement ou à distance.

1.2.4 Les Défis Rencontrés par les Réseaux de Capteurs Sans Fil

Lors de la conception et du déploiement des Réseaux de Capteurs Sans Fil (RCSFs), de nombreux défis émergent (la figure 1.3), nécessitant une attention particulière. Parmi eux [11] :

- **Couverture :** La couverture se réfère à la portée de détection, c'est-à-dire qu'aucune région ne devrait être laissée non couverte par les nœuds capteurs. De plus, la distance entre deux chefs de clusters devrait être aussi grande que possible afin d'éviter les chevauchements de zones de couverture.
- **Durée de Vie :** L'alimentation énergétique de chaque nœud se vide régulièrement, rendant dans certaines situations la recharge ou le remplacement de la batterie du capteur lorsque nécessaire plus difficile. Par conséquent, la durée de vie du réseau (voir 1.2.5) dépend fortement de la gestion efficace de l'énergie consommée par les capteurs.
- **Latence :** Dans une configuration typique des RCSFs, une bande passante fixe est allouée pour le transit des données entre les nœuds. À mesure que le nombre de

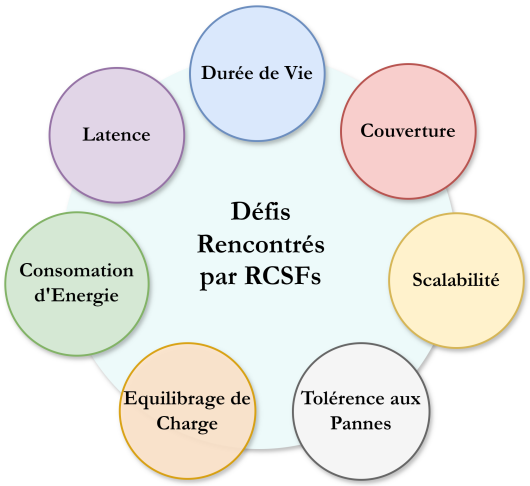

FIGURE 1.3 – Défis Rencontrés par les RCSFs [11].

nœuds augmente, la compétition pour le canal sans fil s'intensifie, entraînant des retards de routage et de mise en mémoire tampon (buffer).

- **Consommation d'Énergie :** La consommation d'énergie représente un défi majeur dans les RCSFs, car elle affecte directement la durée de vie et la performance du réseau d'où la transmission de données est la tâche la plus énergivore pour un capteur.
- **Équilibrage de Charge :**Tous les nœuds dans un réseau de capteurs doivent épuiser leur énergie au même rythme. Si un nœud consomme de l'énergie à un rythme plus élevé que les autres, ce nœud particulier s'épuisera rapidement, entraînant une durée de vie plus courte.
- **Tolérance aux Pannes / Fiabilité :** Étant donné que le remplacement des nœuds vieillissants peut être difficile, le réseau doit être tolérant aux pannes afin de réduire le nombre de défaillances individuelles, ce qui pourrait raccourcir la durée de vie du réseau.
- **Scalabilité :** La performance du réseau ne doit pas être dégradée en augmentant la taille du réseau. Par conséquent, le réseau doit maintenir une efficacité opérationnelle même lorsqu'un grand nombre de nœuds est ajouté.

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

1.2.5 Consommation d'Énergie dans les RCSFs

La durée de vie d'un réseau de capteurs sans fil (RCSF) est généralement déterminée par la période pendant laquelle le réseau maintient une connectivité adéquate, assure une couverture effective de la zone de collecte et maintient un taux de perte de nœuds inférieur à un seuil prédéfini. Voici quelques définitions proposées dans la littérature [12] :

- Durée jusqu'à ce que le premier nœud épuise toutes ses énergies.
- Durée jusqu'à ce qu'un certain pourcentage de capteurs épuisent leurs énergies.
- Durée jusqu'à ce que tous les capteurs épuisent leurs énergies.

1.2.5.1 Consommation d'Énergie du Capteur

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 niveaux d'énergie consommés par un nœud capteur, résultant principalement des opérations de capture, de traitement et de communication des données [13].

- **Energie de Capture :** Elle est dépensée durant l'échantillonnage, le traitement de signal, la conversion analogique/numérique et l'activation de la sonde de capture effectuées par l'unité d'acquisition. Dans de nombreux cas, la consommation d'énergie de cette unité est négligeable par rapport à l'énergie consommée par les modules de traitement et communication .
- **Energie du Traitement :** C'est l'énergie consommée durant le traitement des données. Elle peut être divisée en deux parties : l'énergie de 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 programme). 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.
- **Energie de Communication :** Elle est divisée en deux parties : la puissance de réception et l'énergie d'émission. Cette énergie est déterminée par la quantité de données transmises et la distance de transmission, ainsi que par les caractéristiques physiques de l'unité radio. Elle représente la plus grande proportion de l'énergie totale consommée par un capteur.

L'histogramme présenté par la figure 1.4 représente les portions d'énergie consommée par les différentes unités.

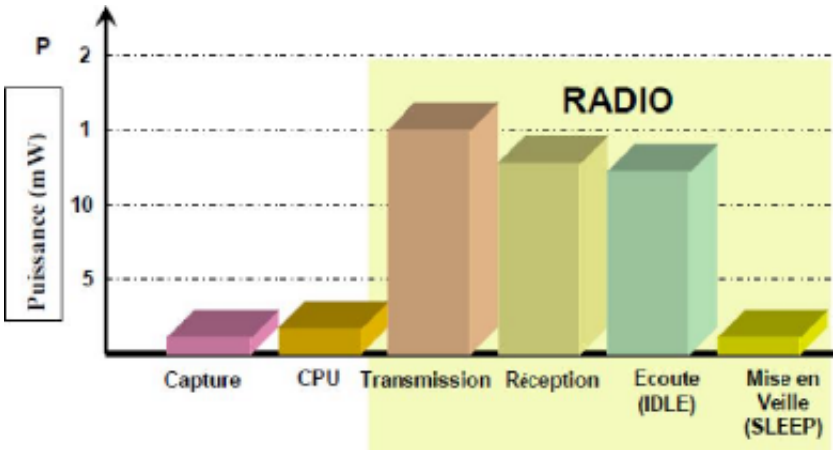

Figure 1.4 – Consommation d'énergie en acquisition, traitement et communication [14].

1.2.5.2 Facteurs Contribuant à la Surconsommation d'Énergie

La surconsommation d'énergie se produit lorsque la quantité d'énergie utilisée dépasse les niveaux normaux ou les besoins prévus. Cela peut être causé par divers facteurs qui

génèrent un gaspillage d'énergie. Voici quelques-uns [15] :

État du Module Radio : L'unité radio est responsable de la surconsommation d'une grande partie de l'énergie des capteurs. La radio fonctionne en quatre modes :

- **Mode Actif :** Le nœud ne reçoit ni ne transmet. Cette condition entraîne une perte de puissance due à une écoute inutile du canal de transmission.
- **Mode d'Envoi :** La radio transmet le paquet.
- **Mode de Réception :** La radio reçoit un paquet.
- **Mode Veille :** La radio est éteinte.

Ainsi, le passage fréquent du mode actif au mode sommeil peut avoir comme effet une surconsommation 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.

Les Collisions : Les nœuds capteurs sont composés essentiellement par une seule antenne radio. Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées nécessitant une retransmission, d'où cette retransmission consomme plus d'énergie.

L'écoute Passive : L'écoute passive se produit lorsqu'un nœud écoute le canal pour une réception possible. C'est une source de gaspillage importante d'énergie dans les RCSFs.

La sur-écoute (Overhearing) : Cette source de dissipation d'énergie se produit lorsqu'un nœud capteur reçoit des paquets qui ne lui sont pas destinés (à une destination différente). Elle conduit à une perte d'énergie additionnelle à cause de l'implication des autres capteurs dans la réception des données.

La surcharge (Overhead) : Les paquets de données dans les RCSFs sont généralement petits, par conséquent, les en-têtes et les autres types de paquets (tels que les messages de contrôle, comme ACK) impliquent un niveau élevé de gaspillage d'énergie.

La surémission (Overemitting) : La surémission est causée lorsque la livraison du message échoue en raison de l'inactivité du nœud de destination. En effet, les messages envoyés sont considérés inutiles et consomment une énergie additionnelle.

La taille des paquets : 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 la surcharge. Dans le cas contraire, une grande puissance de transmission est nécessaire pour des paquets de grande taille.

1.2.5.3 Techniques d'économies d'énergies dans les RCSFs

Plusieurs classifications des techniques de conservation d'énergie ont été proposées dans la littérature. Rault et Bouabdallah ont proposé dans [16] l'une des principales classifications de ces techniques :

- **Optimisation radio :** Afin de réduire la dissipation d'énergie liée aux communications sans fil, les chercheurs ont tenté d'optimiser les paramètres radio tels que la puissance de transmission, la distance de transmission, les intervalles de veille et d'activité… etc.
- **Réduction de données :** Réduire la quantité de données à transmettre vers la station de base.
- **Schémas de veille/réveil (Sleep/wakeup) :** Ils visent à adapter l'activité des nœuds pour économiser l'énergie en mettant la radio en mode veille.
- **Routage écoénergétique :** Il consiste à déterminer un acheminement optimal des paquets à travers le réseau.
- **Recharge de batterie :** Plusieurs études de recherche récentes abordent les techniques de récupération d'énergie et de recharge sans fil visant à recharger les batteries des capteurs sans intervention humaine.

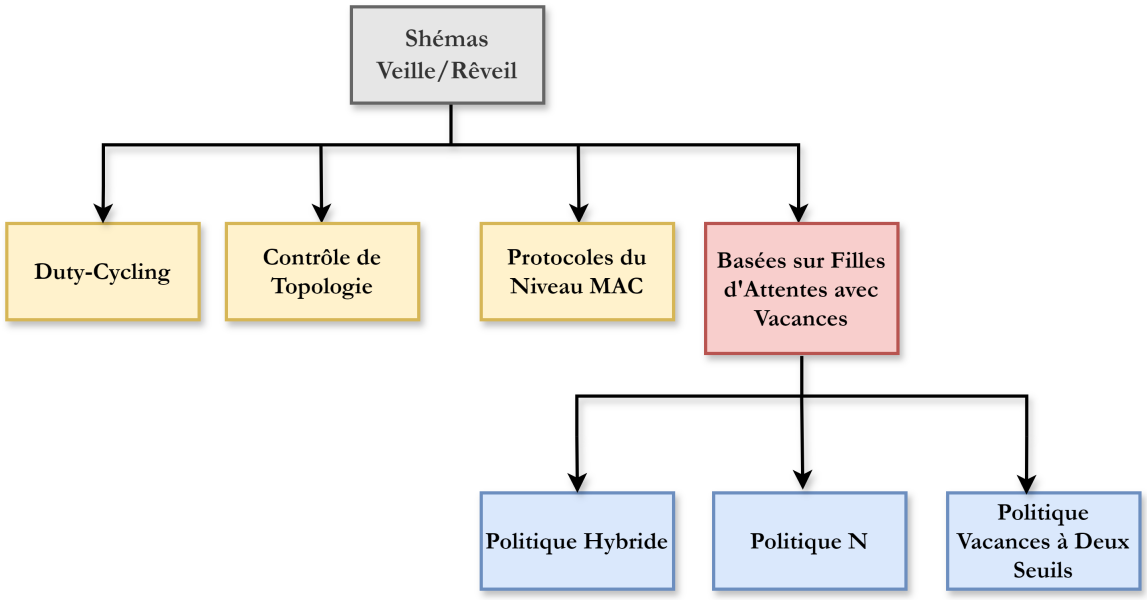

Figure 1.5 – Classifications des techniques de Veille/Réveil.

L'énergie de communication représente la plus grande proportion de l'énergie totale consommée par un capteur [13]. Ainsi, les états d'inactivité sont des sources majeures de consommation d'énergie au niveau du composant radio [16]. Par conséquent, nous nous concentrons sur les techniques de veille/réveil qui sont résumées dans la figure 1.5 .

Techniques de Duty-cycling : Elle consiste à planifier l'état de la radio des nœuds en fonction de l'activité du réseau [16], i.e mettre la radio de l'émetteur en mode veille (low-power) à chaque fois que la communication n'est pas nécessaire afin de minimiser l'écoute

passive et de favoriser le mode veille. 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.

Ces techniques sont généralement divisées en trois catégories [17].

- **À la demande (On-demand) :** Réveiller un nœud uniquement lorsqu'un autre souhaite communiquer avec lui,
- **Asynchrone (Asynchronous) :** Chaque nœud se réveille de manière indépendante, mais sa période active doit coïncider avec celle de ses voisins.
- **Rendez-vous :** Les nœuds se réveillent simultanément avec leurs voisins selon un calendrier de réveil. Ensuite, ils passent en mode veille jusqu'à leur prochain rendez-vous.

Les protocoles basés sur duty-cycling sont indéniablement parmi les plus économes en énergie. Cependant, ils présentent une latence lors de la mise en veille. Un nœud doit attendre que son récepteur soit actif pour qu'il puisse lui envoyer un paquet. De plus, dans certains cas, il devient impossible pour un nœud de diffuser des informations à tous ses voisins car ils ne sont pas actifs simultanément. Ainsi, le défi majeur associé aux régimes à la demande réside dans la manière d'informer un nœud en sommeil qu'un autre nœud est prêt à communiquer avec lui [17].

Techniques de contrôle de topologie : Misra et al. proposent dans [18] une solution qui maintient la couverture du réseau tout en minimisant la consommation d'énergie en activant seulement un sous-ensemble de nœuds avec une zone de chevauchement minimale. L'objectif est de réduire le nombre de nœuds actifs tout en maintenant une connectivité et une couverture adéquates. Avec des capteurs déployés de manière redondante, certains nœuds peuvent être désactivés sans compromettre les opérations et la connectivité du réseau, prolongeant ainsi sa durée de vie. Comme illustré dans la figure 1.6, un capteur doit rester activé dans chaque zone carrée, tandis que les autres nœuds sont désactivés.

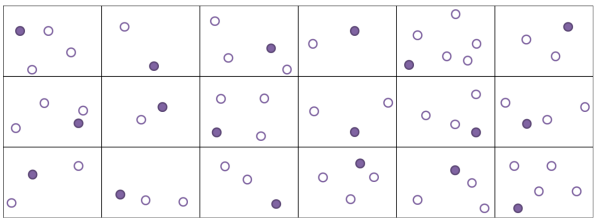

FIGURE 1.6 – Exemple d'une méthode de contrôle de la topologie appliquée à un réseau [16].

Techniques des protocoles du niveau MAC : De nombreux protocoles MAC pour les réseaux de capteurs sans fil ont été proposés, mettant principalement l'accent sur la gestion de l'énergie [19]. Les protocoles MAC basés sur TDMA (Time Division Multiple Access) divisent le temps en trames composées de créneaux temporels. Chaque nœud se voit attribuer un ou plusieurs créneaux par trame pour émettre ou recevoir des données, selon un algorithme d'ordonnancement spécifique. Cette approche permet aux nœuds d'activer leur

radio uniquement pendant leurs créneaux attribués et de la désactiver le reste du temps, économisant ainsi de l'énergie. Ces protocoles organisent souvent les réseaux de capteurs sans fil en clusters avec un nœud coordinateur (cluster head) qui gère l'attribution des créneaux aux nœuds de son cluster. Les principaux avantages de cette méthode sont l'efficacité énergétique et l'absence de collisions grâce à l'attribution déterministe des créneaux. Cependant, ils présentent plusieurs inconvénients [20] :

- Les difficultés à s'adapter aux changements topologiques fréquents des réseaux de capteurs (variations du canal, défaillances de nœuds, etc.).
- La nécessité d'une approche centralisée avec un coordinateur pour l'attribution des slots, ce qui introduit de la complexité et des points uniques de défaillance.
- L'allocation des slots peut être problématique dans des réseaux très dynamiques.

Malgré ces limites, les protocoles TDMA restent une approche intéressante pour la gestion de l'énergie dans les réseaux de capteurs en adaptant les mécanismes d'attribution des slots aux contraintes spécifiques du réseau déployé.

Techniques basées sur les modèles des files d'attentes avec vacances : Dans la littérature, les modèles de file d'attente avec vacances sont largement utilisés pour modéliser et analyser le comportement dynamique des nœuds capteurs avec économie d'énergie. Initialement, la politique N-vacance (N-policy) présentée par Yadin et Naor dans [21], a été utilisée pour contrôler le passage entre l'état de vacance et l'état de service d'une file d'attente avec vacance. Cette politique a été utilisée comme politique pour contrôler le passage entre l'état inactif et l'état occupé du nœud capteur dans un RCSF pour économiser l'énergie. Elle est conçue comme un schéma efficace de mise en veille par file d'attente, caractérisé par le fait que les clients sont accumulés et que le serveur inactif est activé lorsqu'il y a N ($N \geq 1$) clients ou plus dans le système, et que le serveur est désactivé lorsque le système devient vide.

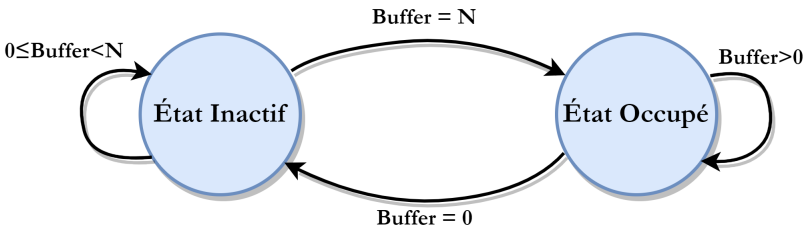

Figure 1.7 – Diagramme de transition d'états d'un nœud capteur avec la politique N-vacance [3].

Il a été prouvé dans la littérature que la politique de N-vacances est une méthode efficace pour améliorer la durée de vie des réseaux de capteurs sans fil. Cependant, elle peut avoir un impact négatif significatif sur le délai de latence des paquets de données [3]. À cet effet, plusieurs travaux ont été proposés dans le but d'améliorer l'efficacité énergétique et de minimiser la latence :

Politique hybride : En 2017, l'étude [3] a présenté une nouvelle politique de vacances appelée la politique hybride (Hybrid-policy). Celle-ci combine la N-policy avec des vacances

aléatoires. Ainsi, un nœud peut passer à l'état occupé et transmettre ses paquets si son buffer atteint le seuil N, ou après une période de vacance aléatoire, même si le nombre de paquets est inférieur à N. L'objectif était de modéliser le comportement des capteurs avec buffer limité et source de trafic finie, en considérant ces différentes politiques via les réseaux de Petri stochastiques généralisés (RdPSGs).

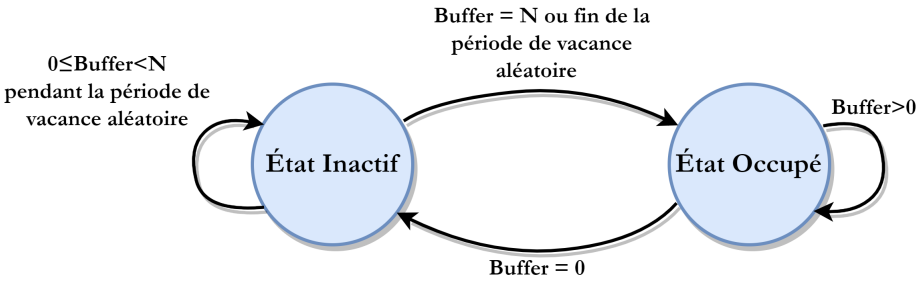

FIGURE 1.8 – Diagramme de transition d'états d'un nœud capteur avec la politique hybride.

Politique de vacances à deux seuils : En 2018, la politique de vacances à deux seuils (Two Thresholds Vacation Policy) a été introduite pour les réseaux de capteurs sans fil dans [4]. Cette politique, qui s'appuie sur la N-policy avec deux seuils N_1 et N_2 (avec $N_1 > N_2$), a été développée pour résoudre le problème du délai d'attente en basculant entre trois états :

- **Inactif (Idle) :** le nœud est en état de veille et consomme très peu d'énergie. Il attend de recevoir des paquets.
- **Semi-occupé (Semi-busy) :** le nœud est dans un état intermédiaire où il commence à accumuler des paquets mais ne passe pas encore à une activité complète.
- **Occupé (Busy) :** le nœud est en plein fonctionnement, traitant et transmettant activement les paquets.

En utilisant deux seuils, cette approche permet une gestion plus fine des périodes d'activité et d'inactivité des nœuds, optimisant ainsi l'utilisation de l'énergie et améliorant l'efficacité globale du réseau.

Modèle de file d'attente avec priorité selon la politique N : Plus récemment, en 2023, l'étude [5] a proposé un modèle de file d'attente à priorités avec la N-policy et un buffer de capacité finie K. Chaque nœud capteur fournit deux types de trafic de données, le trafic d'origine et le trafic de transmission. Ce trafic de données est classé en haute priorité ou basse priorité en fonction de son taux moyen d'arrivée. En effet, le trafic avec un taux d'arrivée moyen élevé est considéré comme de haute priorité, tandis que le trafic avec un taux d'arrivée moyen faible est considéré comme de basse priorité. Les nœuds restent en veille jusqu'à ce que le nombre total de paquets atteint N. Lorsqu'il est occupé, il transmet d'abord les paquets de haute priorité avant d'envoyer les paquets de basse priorité.

Ces différents travaux proposent des variantes et améliorations de la N-policy vacation initiale, en introduisant de nouveaux concepts comme les vacances aléatoires, les multiples seuils ou la prise en compte de buffers de capacité finie, dans le but d'optimiser les compromis énergie-latence pour les réseaux de capteurs.

1.2.6 Synthèse

Le choix du seuil N dans les politiques de gestion d'énergie est critique, car il dépend de nombreux paramètres et n'admet pas de formule mathématique exacte pour déterminer sa valeur optimale. De plus, on remarque d'après les résultats des tests présentés dans la figure 1.9 de la politique "two thresholds working vacation" que la consommation énergétique moyenne en fonction du seuil $N1$ commence par diminuer lorsque $N1$ augmente, mais au-delà d'une certaine valeur, elle se remet à croître. Cette observation illustre la difficulté de choisir la valeur optimale du seuil $N1$ permettant de minimiser la consommation tout en préservant les performances. Une approche par recherche exhaustive permettrait certes d'identifier cette valeur optimale, mais prendrait un temps considérable, rendant cette approche impraticable pour des systèmes de grande taille aux dimensions réalistes.

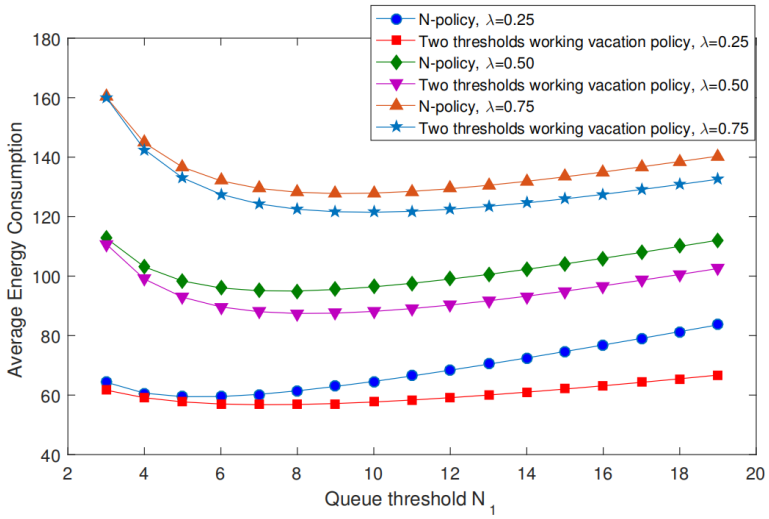

Figure 1.9 – Consommation énergétique moyenne vs. seuil de file d'attente N_1, λ : 0,25 $\sim$ 0,75 [4].

Face à cette complexité, l'optimisation combinatoire apparaît comme une approche prometteuse pour résoudre ce type de problèmes. En effet, les techniques d'optimisation combinatoire permettent de modéliser rigoureusement les différentes contraintes et objectifs, puis d'explorer efficacement l'espace de solutions possibles afin d'identifier les configurations optimales selon les critères définis. C'est dans cette optique que la seconde partie de ce chapitre sur l'état de l'art se concentre sur les concepts et méthodes d'optimisation combinatoire. Nous aborderons les principes fondamentaux de cette discipline, sa modélisation, ainsi que les différentes méta heuristiques développées pour résoudre de manière approchée les problèmes d'optimisation de grande taille et de haute complexité. L'objectif est de poser les bases théoriques nécessaires à l'utilisation judicieuse de ces techniques pour l'optimisation des paramètres clés dans les réseaux de capteurs sans fil, tels que les seuils de politique N, en vue d'atteindre les meilleurs compromis énergie-performance.

Après avoir présenté les différentes techniques d'économie d'énergie dans les réseaux de capteurs sans fil, il est important d'aborder l'aspect d'optimisation. En effet, l'optimisation joue un rôle clé pour tirer la meilleure partie de ces techniques et maximiser leurs

bénéfices en termes d'efficacité énergétique et de performances globales du réseau. La section suivante introduira le concept d'optimisation appliqué aux RCSFs, en définissant ses principes, ses classifications et les différentes méthodes de résolution associées.

1.3 Problèmes d'Optimisation

L'optimisation combinatoire occupe un rôle prépondérant dans plusieurs domaines tels que la recherche opérationnelle, les mathématiques et l'informatique. Son importance provient d'une part de la complexité inhérente aux problèmes d'optimisation, et d'autre part, du vaste éventail d'applications pratiques pouvant être modélisées sous la forme de problèmes d'optimisation combinatoire[22]. En effet, la majorité des problèmes d'optimisation combinatoire appartiennent à la classe des problèmes NP-difficiles, ce qui signifie qu'aucun algorithme efficace permettant de les résoudre de manière exacte n'a encore été découvert à ce jour.

1.3.1 Définitions

L'optimisation est une branche des mathématiques et de l'informatique qui a pour objectif de modéliser, analyser et résoudre des problèmes pratiques. Son principe fondamental consiste à déterminer les solutions qui satisfont au mieux des objectifs quantitatifs formulés, tout en respectant un ensemble éventuel de contraintes. Cela revient donc a résoudre un problème d'optimisation[23].

Un problème d'optimisation est défini par un ensemble discret de solutions possibles S. Parmi ces solutions, un sous-ensemble SR représente les solutions admissibles (réalisables) qui respectent les contraintes imposées. Une fonction objectif, à maximiser ou à minimiser, est associée à chaque solution $s \in SR$ en lui attribuant une valeur numérique $f(s)$. Résoudre un tel problème consiste à trouver une solution optimale $s^* \in SR$ qui optimise (maximise ou minimise) la valeur de la fonction objectif f [23].

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

- Un ensemble des variables de décision X,
- Un ensemble de solutions S, où chaque solution est composée d'une ou plusieurs variables de décision,
- Un ensemble de contraintes C (éventuellement vide) que les variables doivent satisfaire, définissant l'espace des solutions réalisables,
- Un sous-ensemble SR de S représentant les solutions admissibles (réalisables) qui satisfont les contraintes C,
- Une fonction objectif f qui associe à chaque solution $s \in SR$ une valeur numérique $f(s)$.

L'objectif est de trouver une solution optimale $s^* \in SR$ appelée optimum global(la figure 1.10) qui :

- Dans le cas de minimisation : $\forall s \in SR, f(s^*) \leq f(s)$.

— Dans le cas de maximisation : $\forall s \in SR, f(s^*) \geq f(s)$

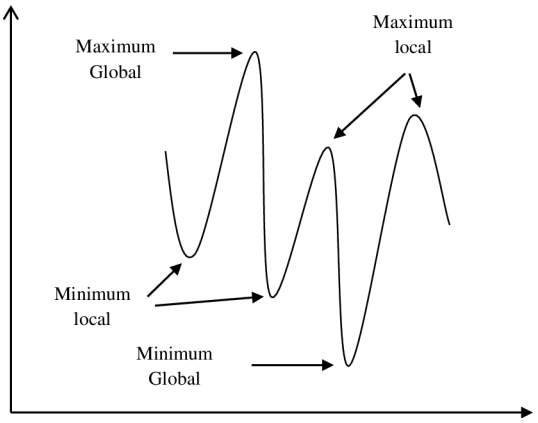

Figure 1.10 – Courbe représentant les Optimums locaux et globaux [24].

Face à un problème d'optimisation combinatoire donné, la question à laquelle on doit répondre est la résolution du problème dont il est nécessaire d'aborder les aspects suivants [25] :

— Définir l'ensemble des variables de décision X ainsi que l'ensemble des solutions possibles S,
— Exprimer l'ensemble des contraintes du problème C afin de définir l'ensemble des solutions réalisables SR,
— Formuler la fonction objectif f à optimiser (minimiser ou maximiser) en fonction des variables de décision,
— Choisir une méthode de résolution appropriée.

1.3.2 Types de Classification des Problèmes d'Optimisation

La classification des problèmes d'optimisation est un aspect crucial pour leur résolution efficace. En effet, les algorithmes développés sont généralement conçus pour résoudre un type particulier de problèmes d'optimisation. Leur performance se dégrade considérablement lorsqu'ils sont appliqués à des types de problèmes différents. Par conséquent, il est important de classer correctement les problèmes d'optimisation selon différents critères (voir la figure 1.11) avant de choisir la méthode de résolution la plus appropriée [24].

1.3.2.1 Classification selon le type des variables de décision

Les problèmes d'optimisation peuvent être classés en fonction de la nature des espaces dans lesquels les variables de décision prennent leurs valeurs, nous distinguons deux classes, à savoir : les problèmes d'optimisation continus et problèmes d'optimisation combinatoires (discrets). Un problème d'optimisation est dit mixte si ces variables peuvent être à la fois discrètes et continues.

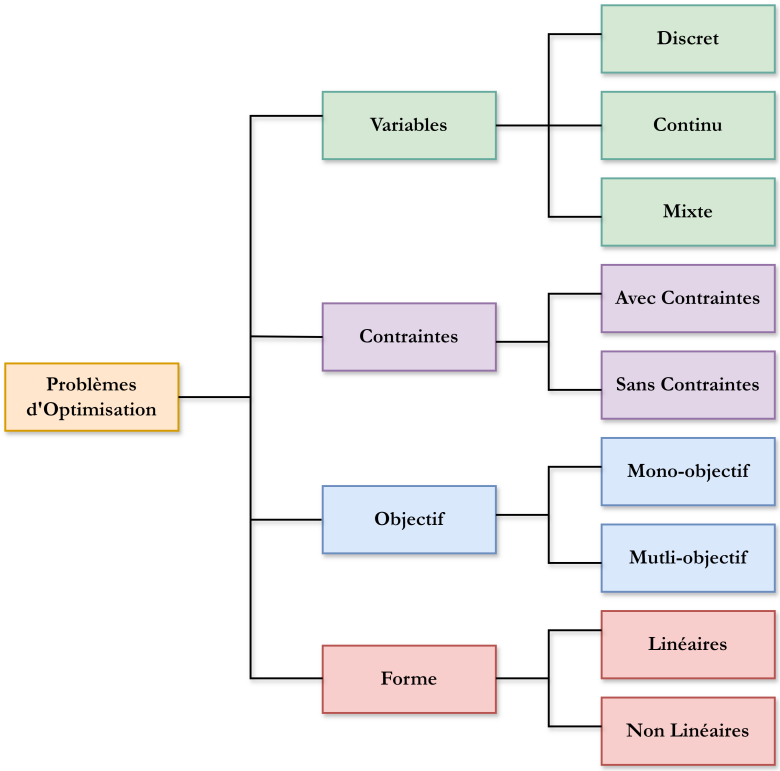

FIGURE 1.11 – Classification des problèmes d'optimisation[24].

1.3.2.2 Classification selon le nombre de contraintes

Un autre critère de classification des problèmes d'optimisation est l'utilisation ou non des contraintes sur l'espace d'état que les variables doivent satisfaire, nous trouvons deux types de problèmes dans cette classification, à savoir les problèmes d'optimisation sans contraintes et les problèmes d'optimisation avec contraintes.

1.3.2.3 Classification 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 objective. 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 objective sous forme d'une combinaison des différents objectifs.

1.3.2.4 Classification 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.

1.3.3 Méthodes d'Optimisation

Selon le type de la recherche, nous trouvons deux grandes catégories de méthodes d'optimisation (voir la figure 1.12) : les méthodes exactes (complètes) qui garantissent la complétude de la solution et les méthodes approchées (incomplètes) qui perdent la complétude afin de gagner en temps d'exécution [24].

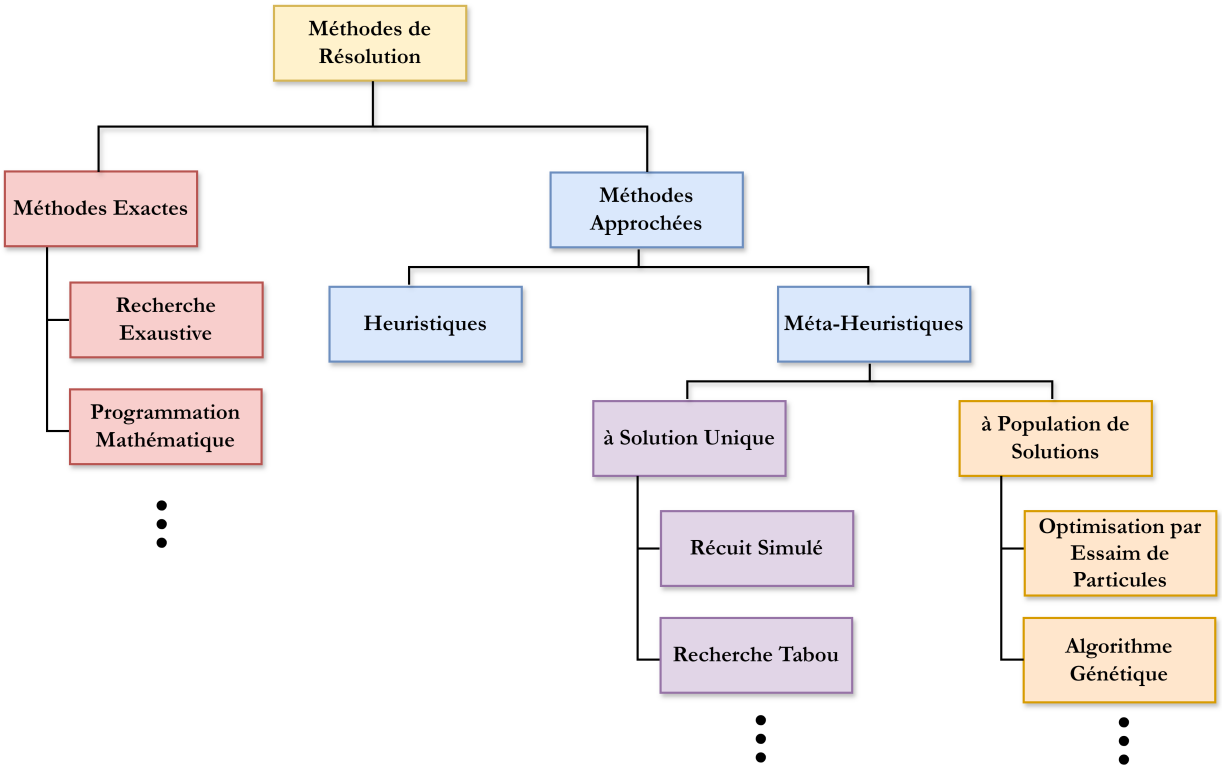

Figure 1.12 – Méthodes d'optimisation [24].

1.3.3.1 Méthodes exactes (complètes)

Les méthodes exactes (appelées aussi complètes) se reposent généralement sur la recherche arborescente de l'espace de solutions en produisant au moins une solution optimale du problème. Il y a deux types [24] :

Méthodes de recherche exhaustive. Elle énumèrent toutes les solutions possibles pour trouver la meilleure solution.

Méthodes de programmation mathématique. Ces méthodes utilisent des modèles mathématiques pour formuler le problème d'optimisation sous forme d'un programme linéaire ou d'autres formes spécifiques.

L'inconvénient majeur de ces méthodes est l'explosion combinatoire : Le nombre de combinaisons augmente avec l'augmentation de la dimension du problème. L'efficacité de ces algorithmes n'est prometteuse que pour les instances de problèmes de petites tailles.

1.3.3.2 Méthodes approchées (incomplètes)

Contrairement aux méthodes exactes, les méthodes approchées ne fournissent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) dans un temps raisonnable [24].

1.3.3.3 Les heuristiques

Une heuristique est un algorithme qui permet de trouver dans un temps polynomial une solution réalisable, tenant en compte d'une fonction objective, pas nécessairement optimale (approchée) ou exacte pour un problème d'optimisation difficile. Elle est spécifique au problème et ne peut pas être généralisée. Ce type de méthodes traduit une stratégie (une manière de penser) en s'appuyant sur la connaissance du problème[24].

1.3.3.4 Métaheuristiques

Les métaheuristiques sont des méthodes généralement inspirées de la nature. Contrairement aux heuristiques, elles ne sont pas propres à un problème précis, mais adaptables et applicables 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 [24]. Pour adapter une métaheuristique à un problème d'optimisation particulier, il est nécessaire de modéliser adéquatement un certain nombre d'ingrédients :

- **L'ensemble S des solutions** : définit le domaine dans lequel la recherche d'un optimum va s'effectuer.
- **La fonction objectif f à minimiser (ou à maximiser)** : décrit la qualité d'une solution au problème. 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 :** 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 :** est utilisé dans le cadre des méthodes évolutives, cet opérateur permet de générer de nouvelles solutions à partir des solutions présentes dans la population courante.

Dans la prochaine sous-section nous détaillerons les catégories existantes des métaheuristiques.

1.3.4 Classes des Métaheuristiques

Les métaheuristiques regroupent des méthodes qui peuvent se diviser en deux classes : métaheuristiques à solution unique et métaheuristiques à population de solutions [26].

1.3.4.1 Métaheuristiques à Solution Unique

Ces méthodes traitent une seule solution à la fois en la modifiant progressivement, afin de trouver la solution optimale, nous citons :

Recuit Simulé (RS). Recuit Simulé (Simulated Annealing SA) est une technique d'optimisation initialement proposée par Kirkpatrick et al [6], utilisée pour trouver une solution approchée à un problème d'optimisation global dans un grand espace de recherche. Elle est inspirée du processus de recuit en métallurgie, où un matériel est chauffé à une température élevée, permettant aux atomes de se déplacer librement. Ensuite, le matériel est refroidi très lentement. Pendant ce refroidissement, les atomes se réarrangent progressivement pour atteindre un état de plus basse énergie, conduisant à une structure cristalline plus stable et moins de défauts. Cette technique suit l'organigramme presenté dans la figure 1.13.

- **Initialisation :** On commence par une solution initiale (souvent générée de manière aléatoire).
- **Température :** Une température initiale élevée est définie. Cette température contrôle la probabilité d'accepter des solutions moins bonnes.
- **Génération de solutions voisines :** À chaque étape, une nouvelle solution est générée à partir de la solution actuelle par une petite modification (mutation).
- **Critère d'acceptation :** La nouvelle solution est acceptée selon un critère basé sur la différence de qualité entre la nouvelle solution et la solution actuelle, ainsi que sur la température. Si la nouvelle solution est meilleure, elle est toujours acceptée. Si elle est moins bonne, elle peut être acceptée avec une certaine probabilité pour éviter de rester bloqué dans des optima locaux.
- **Refroidissement :** La température est diminuée selon un schéma de refroidissement prédéfini (par exemple, linéaire ou exponentiel).
- **Arrêt :** Le processus continue jusqu'à ce que la température soit suffisamment basse ou qu'un critère d'arrêt soit atteint (comme un nombre maximum d'itérations).

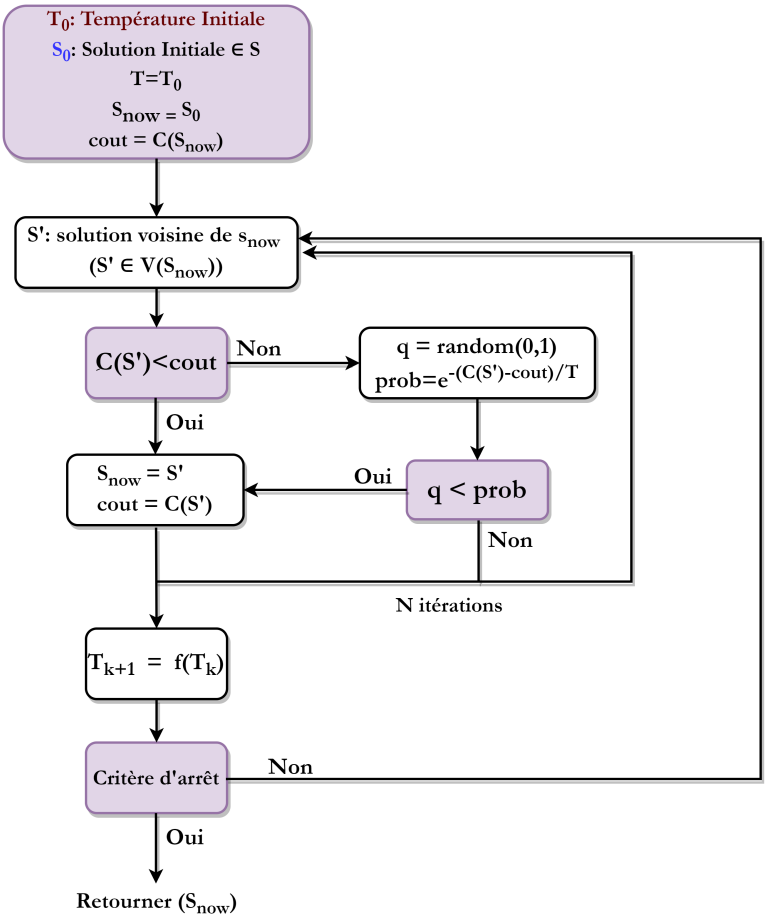

FIGURE 1.13 – Organigramme de la technique Recuit Simulé [27].

Recherche Tabou (RT). La recherche Tabou (Tabu Search TS) a été proposée par Fred Glover en 1986 [7]. Elle utilise la notion de mémoire pour éviter un optimum local. Le principe de la recherche tabou repose sur une méthode de déplacement sur l'espace des solutions, tout en cherchant constamment à améliorer la meilleure solution courante et en conservant en mémoire la liste des précédents déplacements et ainsi guider la recherche en dehors de zones précédemment parcourues. En général on ne va pas garder tous les déplacements (trop coûteux en mémoire), mais on va seulement empêcher l'accès à certaines solutions pendant un certain nombre d'itération(voir figure 19). Cette recherche suit la principe suivant :

- **Initialisation :**
  - Choisir une solution initiale s.
  - Définir la liste tabou (souvent vide au départ).
  - Fixer les paramètres de l'algorithme (taille de la liste tabou, nombre d'itérations, etc.).
- **Recherche Locale :**
  - Générer les voisins de la solution actuelle.
  - Évaluer les voisins pour trouver la meilleure solution candidate qui n'est pas interdite par la liste tabou.

- **Mise à Jour de la Liste Tabou :**
  - Ajouter les mouvements ou les solutions récemment explorés à la liste tabou.
  - Retirer les anciens éléments de la liste tabou selon la politique de gestion de la mémoire (par exemple, après un certain nombre d'itérations).
- **Acceptation de la Nouvelle Solution :**
  - Remplacer la solution actuelle par la meilleure solution candidate (même si elle est moins bonne que la solution actuelle, pour éviter les optima locaux).
- **Critères d'Aspiration :**
  - Si une solution candidate est exceptionnellement bonne (par exemple, meilleure que toutes les solutions trouvées jusqu'à présent), elle peut être acceptée même si elle est dans la liste tabou.
- **Itération :**
  - Répéter les étapes de recherche locale et de mise à jour de la liste tabou jusqu'à atteindre un critère d'arrêt (par exemple, un nombre maximal d'itérations ou une amélioration négligeable sur plusieurs itérations).

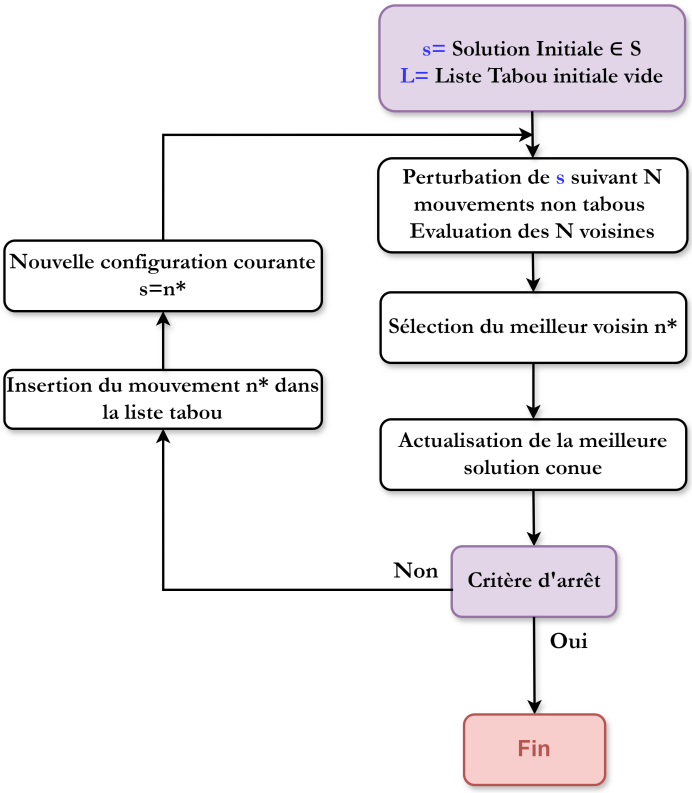

Figure 1.14 – Organigramme de la technique Recherche Tabou [28].

1.3.4.2 Métaheuristiques à Population de Solutions

Ces méthodes manipulent une population de solutions simultanément à chaque itération, jusqu'à l'obtention de la solution globale. Nous détaillons une des méthodes les plus utilisées.

Optimisation par Essaims de Particules (OEP). L'optimisation par essaims de particules (Particle Swarm Optimization PSO), inventée par James Kennedy et Russell Eberhart en 1995 [8], est une méthode d'optimisation inspirée du comportement social des animaux évoluant en essaim. Étant donné que la version initiale de l'algorithme d'OEP présentait des limitations en termes d'efficacité pour résoudre des problèmes d'optimisation complexes, Shi et Eberhart ont proposé en 1998 une version modifiée de cet algorithme dans le travail [29].

Principe de base : Cette métaheuristique est utilisée pour résoudre des problèmes d'optimisation en se basant sur des interactions simples entre des particules dans un espace de recherche. 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 que 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 comme connaissance que la position et la vitesse de ses plus proches voisins, d'où il utilise non seulement sa propre mémoire, mais aussi l'information locale de ses voisins pour décider de son propre déplacement. Le déplacement d'une particule est influencé par les trois composantes suivantes (Figure 20).

- **Composante inertielle (current velocity) :** la particule tend à suivre sa direction et sa vitesse de déplacement courantes.
- **Composante cognitive (best local position) :** la particule est attirée vers la meilleure position qu'elle a atteinte jusqu'à présent, exploitant sa propre expérience (mémoire de la meilleure position de la particule dans l'espace de recherche).
- **Composante sociale (global best position) :** la particule est attirée vers la meilleure position atteinte par ses voisins, exploitant l'expérience collective de l'essaim (meilleure position de l'ensemble de l'essaim, c'est-à-dire la position du leader).

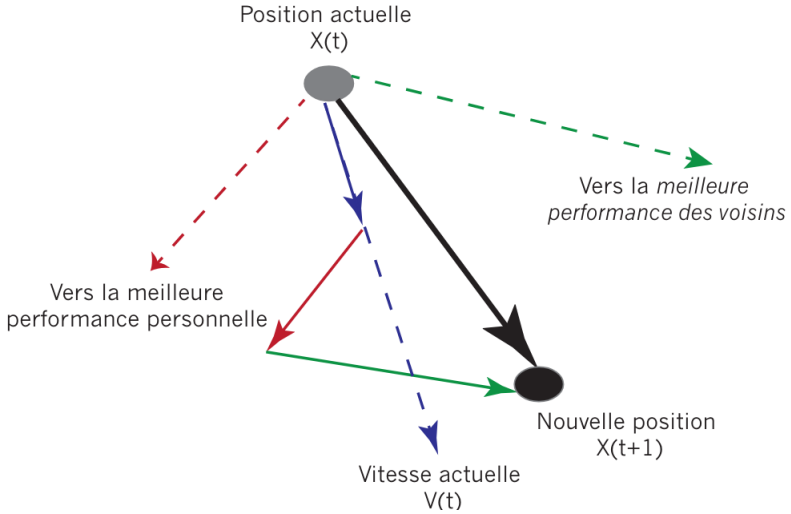

FIGURE 1.15 – Déplacement d'une particule [30].

Le voisinage peut être défini spatialement en prenant par exemple la distance euclidienne entre les positions de deux particules, en prenant la position de l'individu dans l'essaim [31]. Chaque particule i de l'essaim est caractérisée à la fois par sa position $\vec{X}_i$ et par un vecteur de changement de position (appelé vélocité ou vitesse) $\vec{V}_i$. Chaque particule dispose d'une

mémoire lui permettant de se souvenir de sa meilleure solution, découverte par le passé, que l'on note $\vec{P}_i^t$ (personal best) ainsi que de la meilleure position connue de son voisinage, noté $\vec{G}^t$ (global best). À chaque itération, chaque particule se déplace dans l'espace de recherche en suivant un vecteur, calculé comme somme pondérée des vecteurs représentant sa vitesse courante ($\vec{V}_i$), ainsi que sa $\vec{P}_i^t$. et sa $\vec{G}^t$. Sa nouvelle vitesse $\vec{V}_i^{t+1}$ est déterminée en utilisant équation 1.1 [30]. Équation 1.2 [30] calcule le nouvel emplacement de la particule.

$$\vec{V}_i^{t+1} = w\vec{V}_i^t + c_1 r_1(\vec{P}_i^t - \vec{X}_i^t) + c_2 r_2(\vec{G}^t - \vec{X}_i^t). \tag{1.1}$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1} \tag{1.2}$$

Avec :

- w est le facteur d'inertie ;
- c_1 et c_2 sont les coefficients d'accélération ;
- r_1 et r_2 sont des variables aléatoires uniformément distribuées entre 0 et 1.

Algorithme : L'organigramme illustré dans la figure 1.16 présente les étapes principales de l'algorithme d'optimisation par essaim particulaire (PSO). Chaque étape est clairement définie pour montrer le flux de travail depuis l'initialisation de la population jusqu'à la détermination de la meilleure solution.

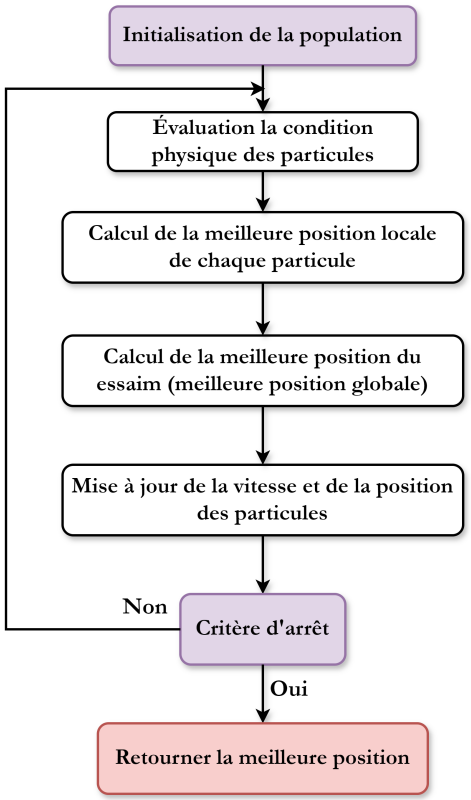

Figure 1.16 – Organigramme d'optimisation par essaims de particules [30].

- **Initialisation de la population :** Générer aléatoirement une population initiale de particules avec des positions et des vitesses dans l'espace de recherche.

- **Évaluation de la condition physique des particules :** Calculer la valeur de la fonction objectif (ou fitness) pour chaque particule, ce qui permet d'évaluer la qualité de chaque solution candidate.
- **Calcul de la meilleure position locale de chaque particule :** Comparer la position actuelle de chaque particule avec sa meilleure position passée $\vec{P}_i^t$ et mettre à jour si la position actuelle est meilleure.
- **Calcul de la meilleure position d'essaim (meilleure position globale) :** Comparer la meilleure position locale de chaque particule avec la meilleure position globale $\vec{G}^t$ et mettre à jour si une meilleure position est trouvée parmi toutes les particules.
- **Mise à jour de la vitesse et de la position des particules :** -Utiliser les équations de mise à jour pour ajuster la vitesse et la position de chaque particule en fonction de sa vitesse actuelle, de sa meilleure position locale, et de la meilleure position globale.
- **Critère d'arrêt :** Vérifier si le critère d'arrêt est atteint, tel qu'un nombre maximal d'itérations ou un seuil de performance. Si le critère d'arrêt est atteint, l'algorithme s'arrête.
- **Retourner la meilleure position :** Une fois que le critère d'arrêt est atteint, retourner la meilleure position trouvée par l'essaim comme solution optimale.

Forts de cette compréhension approfondie des métaheuristiques, nous nous tournons maintenant vers les méthodes de modélisation et d'évalutation de performances des systèmes, une étape cruciale pour l'analyse et l'optimisation des systèmes complexes tels que les réseaux de capteurs sans fil.

1.4 Modélisation du Système

Pour mener à bien l'analyse et l'optimisation d'un système complexe tel que les réseaux de capteurs sans fil, il est essentiel de disposer d'outils de modélisation mathématique adaptés. La modélisation permet de représenter fidèlement le comportement dynamique du système étudié, en tenant compte de ses différentes composantes, interactions et contraintes.

1.4.1 Réseaux de Pétri Stochastiques Généralisés

Les réseaux de Petri sont un outil de modélisation mathématique et graphique permettant de représenter et d'analyser des systèmes dynamiques à événements discrets. Ils constituent un formalisme puissant pour la description et l'étude des systèmes parallèles, concurrents, asynchrones et distribués.

Un réseau de Petri est composé de quatre éléments principaux :

- **Places :** représentées graphiquement par des cercles, elles modélisent les conditions ou les états du système.
- **Transitions :** représentées par des rectangles ou des traits, elles modélisent les événements qui font évoluer le système d'un état à un autre.

— **Arcs :** ils relient les places aux transitions et vice-versa, et définissent les conditions de franchissement des transitions.
— **Jetons :** représentés par des marquages (points noirs) dans les places, ils correspondent aux ressources ou aux données du système.

Le fonctionnement d'un réseau de Petri est régi par des règles de franchissement des transitions, basées sur la présence ou l'absence de jetons dans les places d'entrée. Lorsqu'une transition est franchissable, elle consomme des jetons des places d'entrée et en produit dans les places de sortie, modifiant ainsi le marquage global du réseau.

Les Réseaux de Pétri permettent de modéliser et d'analyser divers aspects des systèmes tels que les conflits, la synchronisation, le parallélisme, les partages de ressources et les interblocages. Grâce à leur structure graphique et formelle, ils offrent une représentation claire et précise des interactions complexes au sein des systèmes distribués. En plus de leur capacité à capturer les dynamiques du système, les Réseaux de Pétri offrent la possibilité d'effectuer des analyses qualitatives. Ces analyses permettent de vérifier des propriétés structurelles importantes telles que :

— **Vivacité :** Capacité du système à continuer de fonctionner sans bloquer, assurant que toutes les parties du système peuvent progresser.
— **Sûreté :** Absence de conditions dangereuses ou indésirables, garantissant que le système ne dépasse jamais certains seuils critiques.
— **Absence de Deadlocks :** Vérification que le système ne se bloque jamais définitivement, ce qui est crucial pour la fiabilité opérationnelle.

Les Réseaux de Pétri Stochastiques Généralisés (RdPSGs) sont une extension sophistiquée des réseaux de Pétri classiques, intégrant des aspects temporels et probabilistes pour mieux représenter les dynamiques complexes des systèmes. Cette combinaison permet de modéliser non seulement les événements discrets mais aussi les délais stochastiques associés aux transitions, offrant ainsi une représentation plus réaliste des systèmes dynamiques et stochastiques. Les RdPSGs sont particulièrement adaptés à la modélisation des systèmes où les interactions concurrentes et les phénomènes aléatoires jouent un rôle crucial, tels que les réseaux de capteurs sans fil.

1.4.2 Chaînes de Markov

Pour modéliser l'évolution au cours du temps (la dynamique) de systèmes, on choisit souvent des modèles aléatoires. Un de ces modèles aléatoires est les chaînes de Markov. Une chaîne de Markov est une suite de variables aléatoires $(X_n, n \in N)$ qui permet de modéliser l'évolution dynamique d'un système aléatoire : X_n représente l'état du système à l'instant n. La propriété fondamentale des chaînes de Markov, dite propriété de Markov, est que son évolution future dépend uniquement de l'état actuel, pas de l'historique. Autrement dit, conditionnellement à X_n, $(X_0, ..., X_n)$ et $(X_{n+k}, k \in N)$ sont indépendants.

Une chaîne de Markov est caractérisée par les éléments suivants :

— **États :** Un ensemble fini d'états possibles dans lesquels le système peut se trouver.
— **Générateur Infinitésimal :** Une matrice qui spécifie les probabilités de passer d'un état à un autre. Cette matrice est souvent appelée le générateur infinitésimal de la

chaîne de Markov.

Les chaînes de Markov peuvent être classées en plusieurs catégories en fonction de leurs caractéristiques, telles que :

- **Chaîne de Markov à temps discret :** Dans ce type de chaîne, les transitions d'état se produisent à des instants de temps discrets, comme jours, semaines, etc.
- **Chaîne de Markov à temps continu :** Dans ce type de chaîne, les transitions d'état peuvent se produire à tout moment dans un intervalle continu.
- **Chaîne homogène :** Les probabilités de transition entre les états restent constantes avec le temps.
- **Chaîne non homogène :** Les probabilités de transition entre les états peuvent varier avec le temps.

Les chaînes de Markov sont utilisées pour modéliser une grande variété de processus stochastiques, tels que la météo, les processus de file d'attente, les systèmes de communication, etc. Elles permettent de prédire les comportements futurs du système en fonction de l'état actuel, ce qui les rend très utiles dans de nombreux domaines d'application, y compris les réseaux, la génétique des populations, les algorithmes stochastiques d'optimisation, et la simulation. Les Chaînes de Markov à Temps Continu (CTMC) améliorent cette modélisation en ajoutant une dimension temporelle précise, permettant une analyse quantitative détaillée des dynamiques de ces systèmes. Les CTMC sont particulièrement efficaces pour évaluer les performances des systèmes, calculer les probabilités d'état stationnaire, les temps moyens de séjour dans les états, et les taux de transition entre les états. Cette capacité à effectuer des analyses quantitatives précises en fait un outil précieux pour modéliser et optimiser des systèmes complexes tels que les réseaux de capteurs sans fil, les systèmes de production, et les réseaux de communication.

1.5 Conclusion

En conclusion, ce chapitre a posé les bases théoriques et conceptuelles essentielles pour l'étude des problématiques liées aux réseaux de capteurs sans fil (RCSFs) et leur optimisation. Nous avons d'abord mis en lumière les défis majeurs de ces réseaux, notamment la gestion efficace de la consommation énergétique, en présentant diverses solutions innovantes issues de la littérature. Ensuite, nous avons introduit les concepts fondamentaux de l'optimisation combinatoire et des méthodes de résolution des problèmes d'optimisation, avec un focus particulier sur les métaheuristiques, qui sont particulièrement adaptées aux problèmes complexes et de grande taille rencontrés dans les RCSFs. Enfin, nous avons abordé la modélisation du système en présentant les Réseaux de Pétri Stochastiques Généralisés (RdPSGs) et les Chaînes de Markov à Temps Continu (CTMCs), deux outils puissants pour représenter et analyser les dynamiques complexes des systèmes.Une fois, l'étude bibliographique effectuée, nous présenterons dans le prochain chapitre, les éléments méthodologiques pour la conception de notre système.

Chapitre 2

Conception

2.1 Introduction

L'étude effectuée dans le chapitre précédent nous a permis d'avoir une compréhension approfondie sur les problématiques liées aux réseaux de capteurs sans fil et leur optimisation. Dans ce chapitre, nous abordons la conception de la solution proposée au problème lié à la conservation d'énergie dans les RCSFs et les méthodes d'optimisation de ses paramètres. Après avoir défini les principes de la Politique de Vacances de Travail à Deux Seuils VTDS [4], nous présentons la modélisation du système en utilisant les Chaînes de Markov à Temps Continu (CMTCs). Ensuite, nous détaillons le calcul de l'énergie consommée et du délai d'attente, qui sont des métriques cruciales pour l'évaluation des performances du système. Enfin, nous formulons le problème d'optimisation et discutons des méthodes de résolution basées sur des métaheuristiques, y compris le Recuit Simulé (RS), la Recherche Tabou (TS), et l'Optimisation par Essaim de Particules (OEP). Chaque section de ce chapitre vise à fournir une compréhension approfondie et à développer les outils nécessaires pour optimiser l'efficacité énergétique et la performance des RCSFs.

2.2 CMTCs Pour la Modélisation de la Politique VTDS

Dans notre démarche, nous nous appuyons sur la politique de vacances VTDS [4], discutée auparavant dans de l'état de l'art. Ce choix découle des résultats présentés dans la figure 1.9 , où il est clairement observé que l'approche de cette politique entraîne une réduction de la consommation d'énergie par rapport à l'utilisation de la N-policy. Nous commencerons par décrire en détail la politique de conservation d'énergie dans les capteurs sans fil dans la sous-section suivante.

2.2.1 Modèle Mathématique pour un Capteur Sans Fil

La politique VTDS correspond à la N-policy avec deux seuils N_1 et N_2 ($N_1 > N_2$) pour basculer entre trois états Inactif (Idle), Semi-occupé (Semi-busy) et Occupé (Busy).

Paramètres du Système : Les paramètres du système qui influent sur son fonctionnement et ses performances globales, sont essentiels pour comprendre la dynamique du système et ils sont utilisés dans la modélisation et l'analyse ultérieures. Ils sont définis comme suit :

- λ : taux moyen d'arrivé des paquets par unité de temps.
- K : capacité maximale du buffer du nœud.
- $N2$: seuil inférieur du buffer pour passer à l'état semi-occupé.
- $\mu 2$: taux de traitement des paquets durant l'état semi-occupé.
- $N1$: seuil supérieur du buffer pour passer à l'état occupé.
- $\mu 1$: taux de traitement des paquets durant l'état occupé.

Fonctionnement : Le noeud fonctionne selon son état actuel.

- **État Inactif :**
  - Le nœud est en veille et consomme un minimum d'énergie. Lorsqu'un paquet arrive, il est placé dans le buffer.
  - Si le nombre de paquets dans le buffer atteint $N2$, le nœud passe à l'état semi-occupé.
- **État Semi-occupé :**
  - Pendant cet état, le nœud commence à traiter les paquets à un taux de service réduit $\mu 2$.
  - Si le nombre de paquets atteint $N1$, le nœud passe à l'état occupé.
  - Si le buffer devient vide avant d'atteindre $N1$, le nœud retourne à l'état inactif.
- **État Occupé :**
  - Le nœud traite les paquets à un taux de service normal $\mu 1$ pour gérer la haute charge de travail.
  - Le nœud reste dans cet état jusqu'à ce que le buffer soit vide, puis il retourne à l'état inactif.

La représentation graphique du diagramme de transition associé à notre approche est présentée dans la Figure 22. Ce diagramme illustre les différents états du système ainsi que les transitions entre ces états.

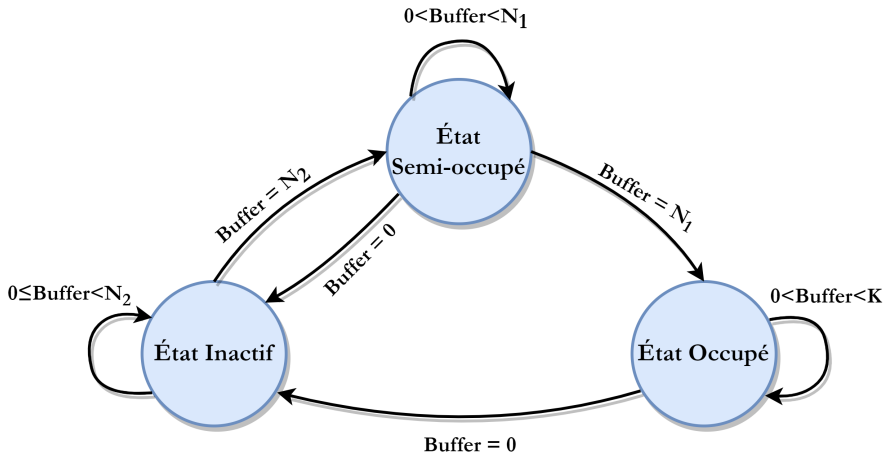

FIGURE 2.1 – Diagramme de Transition d'États d'un Nœud Capteur avec la Politique VTDS.

2.2.2 Modélisation du Noeud Capteur en Utilisant les CMTCs

Dans [4], le système a été modélisé par un Réseau de Pétri Stochastique Généralisé (RdPSG) en utilisant l'outil GreatSPN, permettant ainsi de réaliser une évaluation qualitative et quantitative. Cette évaluation a fourni des informations précieuses sur les performances du système. Cependant, GreatSPN ne permet pas de spécifier des plages de valeurs pour les paramètres du système, ce qui nécessite de multiples exécutions manuelles pour tester différentes configurations. Chaque exécution du modèle peut prendre un temps considérable et le besoin de multiples exécutions rend l'évaluation très coûteuse en termes de temps et de ressources. De plus, cet outil n'offre pas la possibilité d'automatiser les expérimentations du modèle en RdPSG en utilisant une application dédiée.

Pour ces raisons, nous adopterons une approche algorithmique basée sur les CMTCs pour mener les expérimentations nécessaires à l'optimisation des paramètres du système par les métaheuristiques abordées dans la suite du projet. Le système en question est modélisé par une file d'attente avec vacance de travail et buffer fini. La politique VTDS [4] est utilisée pour contrôler le passage entres les états de la file d'attentes. Le modèle du noeud capteur peut être représenté par une CMTC à deux dimensions. Chaque état est décrit par deux variables aléatoires $X(t)$ et $Y(t)$ avec $t \geq 0$ (voir figure 2.2), tels que $X(t)$ représente le nombre de paquet au niveau de buffer et $Y(t)$ représente l'état de système où :

- $Y(t) = 0$: le capteur est à l'état inactif,
- $Y(t) = 1$: le capteur est à l'état semi-occupé,
- $Y(t) = 2$: le capteur est à l'état occupé.

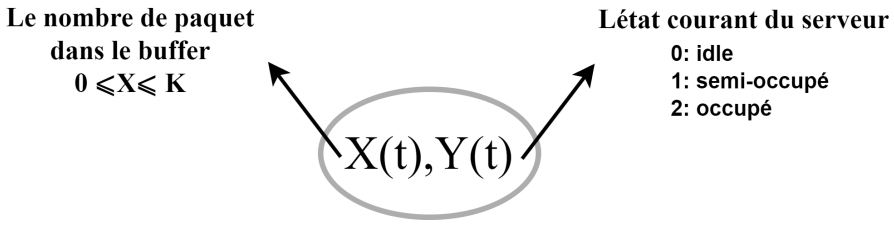

FIGURE 2.2 – Un État du Modèle du Noeud Capteur.

Pour obtenir une vue complète de la manière dont notre système évolue dans son ensemble, nous devons également prendre en compte les transitions entre tous les états possibles. La Figure 25 présente le modèle complet de la CMTC, offrant une vision des différentes trajectoires que notre système peut suivre.

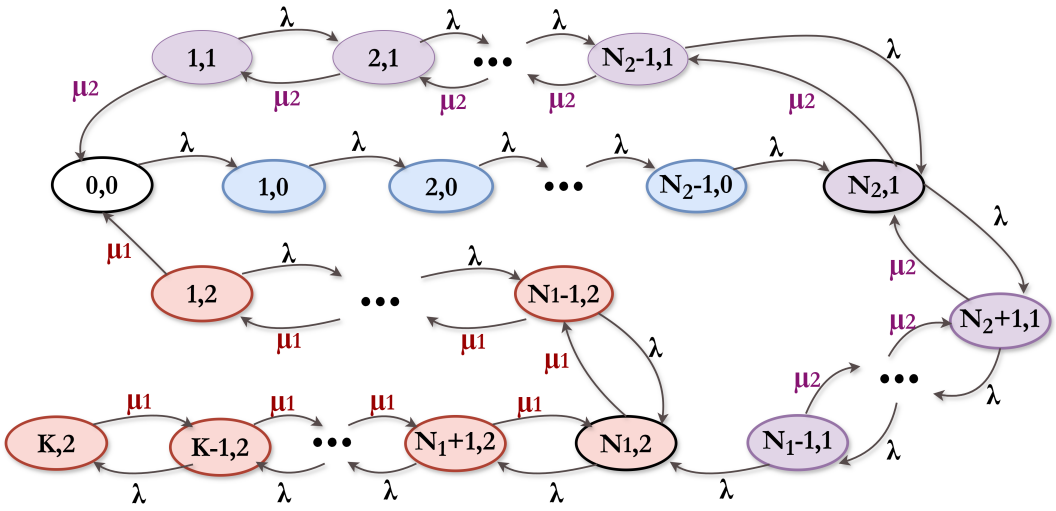

FIGURE 2.3 – Modèle CMTC avec la politique VTDS.

Cette CMTC peut être transformée en un générateur infinitésimal contenant $N_1 + N_2 + K - 1$ états à l'aide de l'algorithme 1 suivant.

Algorithme 1 : Algorithme de construction du générateur infinitésimal

Data : K ; N_1 ; N_2 ; μ_1 ; μ_2 ; λ.

Result : Q.

Initialiser Q par zéros;

Algorithme 1 : Algorithme de construction du générateur infinitésimal (suite)

```
foreach i ← 0 to K − 1 do
    Q[i][i + 1] ← λ;
    if i > N1 then
        Q[i][i − 1] ← μ1;
    else
        if i > N2 then
            Q[i][i − 1] ← μ2;
        end
    end
end
Q[K][K − 1] ← μ1;
if N2 = 1 then
    Q[N2][0] ← μ2;
else
    Q[N2][K + 1] ← μ2;
    Q[K + 1][N2] ← λ;
end
foreach i ← K + 1 to K + N2 − 2 do
    Q[i][i + 1] ← μ2;
    if i > K + 1 then
        Q[i][i − 1] ← λ;
    end
end
if N2 ≥ 2 then
    Q[K + N2 − 1][0] ← μ2;
    if N2 > 2 then
        Q[K + N2 − 1][K + N2 − 2] ← λ;
    end
end
Q[N1][K + N2] = μ1;
Q[K + N2][N1] = λ;
foreach i ← K + N2 to K + N1 + N2 − 3 do
    Q[i][i + 1] ← μ1;
    if i > K + N2 then
        Q[i][i − 1] ← λ;
    end
end
```

Algorithme 1 : Algorithme de construction du générateur infinitésimal (suite)

if $N_1 > 2$ **then**
  $Q[K + N_1 + N_2 - 2][K + N_1 + N_2 - 3] \leftarrow \lambda$;
end
$Q[K + N_1 + N_2 - 2][0] \leftarrow \mu_1$;
foreach $i \leftarrow 0$ *to* $N_1 + N_2 + K - 2$ **do**
  $Q[i, i] \leftarrow -\text{somme\_ligne}(i)$;
end
Retourner Q

2.2.3 Les Indices de Performance

Pour comprendre la dynamique de notre système, il est essentiel de noter qu'il est irréductible, car il comporte une seule composante fortement connexe, comme illustré dans la figure 2.3. Cela signifie qu'il est possible de passer d'un état à n'importe quel autre état au sein du système et qu'aucun état n'est isolé.

Pour évaluer efficacement la performance d'un capteur sans fil, il est crucial de calculer l'énergie consommée et le délai d'attente. En utilisant une CMTC, nous suivons plusieurs étapes, schématisées dans la figure 2.4. Ces étapes couvrent la construction de la CMTC à partir des paramètres du système jusqu'à l'obtention des métriques de performance souhaitées.

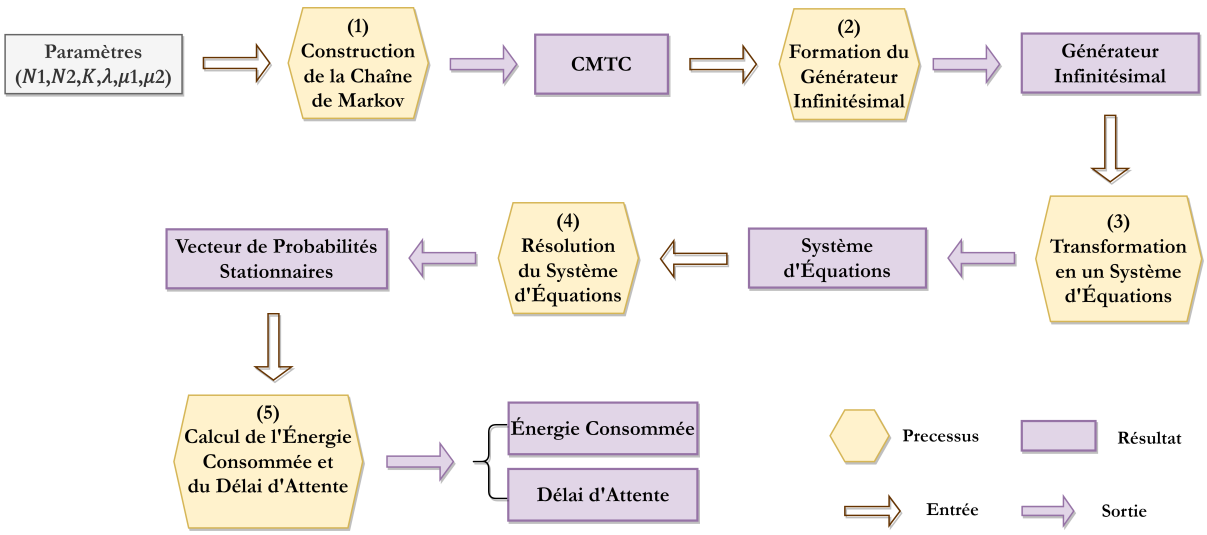

FIGURE 2.4 – Schéma des Étapes pour l'Évaluation de la Performance du Capteur Sans Fil avec une CMTC.

Construction de la CTMC : construire la chaîne de Markov appropriée en déterminant les transitions possibles entre les états. Les taux de transition sont dérivés des paramètres du système, tels que les taux de service et les taux d'arrivée.

Formation du Générateur Infinitésimal : construire une matrice Q où chaque élément q_{ij} représente le taux de transition de l'état i à l'état j.

Résolution du Système d'Équations : résoudre le système d'équations $\pi Q = 0$ avec la condition de normalisation $\sum \pi_i = 1$ pour trouver les probabilités d'état stationnaire π_i. Ces probabilités représentent la proportion de temps que le système passe dans chaque état à long terme.

Calcul de l'Énergie Consommée et du Délai d'Attente : pour le calcul de l'énergie consommée et du délai d'attente, nous avons adapté les équations issues de [4] pour les chaînes de Markov. Elles deviennent :

— Calcul de la probabilité d'être Inactif

$$P_I = \sum_{i=0}^{N_2-1} \pi_{i,0} \tag{2.1}$$

— Calcul de la probabilité d'être en Semi-Occupé

$$P_{SB} = \sum_{i=1}^{N_1-1} \pi_{i,1} \tag{2.2}$$

— Calcul le débit moyen d'arrivé

$$\bar{\lambda} = \lambda \cdot (\sum_{i=1}^{N_2-1} \pi_{i,0} + \sum_{i=1}^{N_1-1} \pi_{i,1} + \sum_{i=1}^{K-1} \pi_{i,2}) \tag{2.3}$$

— Calcul du nombre moyen de paquets dans la file d'attente

$$\bar{Q} = \sum_{i=1}^{N_2-1} i \cdot \pi_{i,0} + \sum_{i=1}^{N_1-1} i \cdot \pi_{i,1} + \sum_{i=1}^{K} i \cdot \pi_{i,2} \tag{2.4}$$

— Calcul du nombre moyen de paquets dans la file d'attente pendant l'état Semi Occupé

$$\bar{Q}_{SB} = \sum_{i=1}^{N_1-1} i \cdot \pi_{i,1} \tag{2.5}$$

— Calcul du nombre moyen de paquets dans la file d'attente pendant l'état Occupé

$$\bar{Q}_B = \sum_{i=1}^{K} i \cdot \pi_{i,2} \tag{2.6}$$

— Calcul de la longueur moyenne d'une période d'attente (Inactif)

$$\bar{I} = \frac{N_2}{\lambda} \tag{2.7}$$

— Calcul de la longueur moyenne d'une période Semi-occupée

$$\bar{SB} = min\{\frac{N_1 - N_2}{\lambda}, \frac{\bar{Q}_{SB}}{\mu_2}\} \tag{2.8}$$

— Calcul de la longueur moyenne d'une période Occupée

$$\bar{B} = \frac{\bar{Q}_B}{\bar{\mu}_1} \tag{2.9}$$

— Calcul de la durée moyenne d'un cycle

$$\bar{C} = \bar{B} + \bar{SB} + \bar{I} \tag{2.10}$$

— Calcul du délai moyen en utilisant la loi de *"Little"*

$$\bar{W} = \frac{\bar{Q}}{\bar{\lambda}} \tag{2.11}$$

— Calcul de l'énergie consommée

$$EC = EC_I \cdot P_I + EC_{SB} \cdot P_{SB} + EC_B \cdot (1 - P_I - P_{SB}) + \frac{EC_s}{\bar{C}} + \bar{Q} \cdot EC_{Tx} \tag{2.12}$$

Où :

- EC_I, EC_{SB}, EC_B : La consommation d'énergie de l'unité de communication (radio) à l'état Inactif, Semi-occupé et Occupé respectivement ;
- EC_s : La consommation d'énergie du changement de l'état du capteur ;
- EC_{Tx} : La consommation d'énergie pour maintenir chaque paquet présent dans le capteur

2.3 Problème d'Optimisation à Résoudre

Avec la politique VTDS, la conservation d'énergie dans les capteurs sans fil est grandement améliorée en ajustant les périodes d'activité et d'inactivité des capteurs [4]. En se concentrant sur la réduction de la consommation d'énergie, cette approche peut entraîner une augmentation du délai d'attente pour la transmission des paquets car le délai d'attente augmente avec l'augmentation des seuils [4]. Ce compromis entre la conservation d'énergie et l'augmentation des délais d'attente doit être soigneusement géré pour maintenir un équilibre optimal entre efficacité énergétique et performance du réseau.

L'objectif principal de notre projet de fin d'étude est de minimiser l'énergie consommée tout en maintenant un délai d'attente acceptable dans les capteurs sans fil, et donc d'améliorer la performance globale des RCSFs. Pour ce faire, nous concevons une solution basée sur les métaheuristiques pour trouver les valeurs optimales des paramètres de configuration de la politique VTDS, modélisée par les CMTCs. Les paramètres à optimiser incluent les seuils (N_1, N_2), la taille du buffer (K), le taux moyen d'arrivée (λ) et les taux moyens

de service (μ_1, μ_2) des paquets. Les valeurs optimales $(N_1^*, N_2^*, K^*, \lambda^*, \mu_1^*, \mu_2^*)$ permettent de minimiser la consommation d'énergie ainsi que le délai dans les RCSFs. Pour cela, nous devons optimiser (minimiser) une fonction objectif à deux critères (multi-objectifs).

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

- Un ensemble des variables de décision $X = \{N_1, N_2, K, \lambda, \mu_1, \mu_2\}$ où N_1, N_2 et K sont des variables naturelles et λ, μ_1 et μ_2 sont des variables réelles,
- Un ensemble de solutions $S = (N_1, N_2, K, \lambda, \mu_1, \mu_2)$,
- Un ensemble de contraintes C avec :
$$\begin{cases} N_2 < N_1 < K \\ \mu_2 < \mu_1. \end{cases}$$
- 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 $f(s)$, en prenant en compte deux critères : l'énergie (EC) et le délai $(\bar{W})$. Cette fonction multiobjectif est transformée en une fonction mono-objectif en calculant une somme pondérée entre l'énergie standardisée (E') et le délai standardisé (D') d'où :
  - La standardisation permet d'équilibrer les échelles des deux objectifs et d'éviter qu'un objectif aux valeurs naturellement plus grandes ne domine l'autre dans la somme pondérée, ce qui pourrait fausser le résultat de l'optimisation. Pour ce faire, nous devons d'abord calculer l'énergie maximale et le délai maximal en utilisant les équations respectives 2.13 et 2.14, puis diviser $E(s)$ et $D(s)$ par leurs valeurs maximales ;
  - $f(s) = a1 * E'(s) + a2 * D'(s)$;
- Il s'agit de trouver une solution optimale (ou optimum global) $s^* \in SR$ qui minimise la fonction objectif : $\forall s \in SR, f(s^*) \leq f(s)$

En résumé, notre problème d'optimisation est un problème :

- Discret car les variables de décision $X = \{N_1, N_2, K, \lambda, \mu_1\mu_2\}$ sont discrètes.
- Avec contrainte : des bornes sur les variables de décision.
- Multi-objectifs que nous avons reformulé en un problème mono-objectif.
- Linéaire car la fonction objectif est une somme pondérée de fonctions linéaires.

2.3.1 Calcul de l'Énergie Maximale

Pour calculer l'énergie maximale dans un capteur sans fil, nous utilisons l'équation 2.13, dérivée de l'équation 2.12, en appliquant les conditions suivantes :

$$EC_{max} = EC_I \cdot P_I + EC_B \cdot (1 - P_I) + EC_s + K \cdot EC_{TX} \tag{2.13}$$

- $P_{SB} = 0$, pour éliminer l'état semi-occupé,
- $P_I \approx 0$, pour maximiser l'état occupé $(1 - P_I)$,
- $\bar{C} = 1$, pour avoir un seul cycle par unité de temps,
- $\bar{Q} = K$, pour maximiser la consommation d'énergie due à la gestion des paquets présents dans le nœud capteur.

2.3.2 Calcul du Délai Maximal

En utilisant l'équation 2.14, élaborée à partir de l'équation 2.11, en tenant compte des conditions suivantes :

$$W_{max} = \frac{K}{\lambda} \tag{2.14}$$

- $\bar{Q} = K$, pour maximiser le nombre moyen de paquets dans le buffer,
- $\bar{\lambda} = \lambda$, en prenant la valeur minimale de λ.

2.4 Méthodes de Résolution

Pour résoudre notre problème d'optimisation des paramètres de la politique VTDS dans les capteurs sans fil, plusieurs méthodes approchées peuvent être utilisées. Nous utilisons trois méthodes de résolution basées sur des métaheuristiques : deux à solution unique, le Recuit Simulé (RS) et la Recherche Tabou (RS), et une à population de solutions, l'Optimisation par Essaim Particulaire (OEP), afin de pouvoir faire une comparaison entre elles. Les algorithmes principaux des métaheuristiques sont généralement conçus pour être applicables à une large gamme de problèmes d'optimisation. Cependant, ils nécessitent souvent des adaptations spécifiques pour répondre aux particularités et aux contraintes des problèmes spécifiques, notamenet pour notre problème d'optimisation.

2.4.1 Adaptation de Recuit Simulé pour le Problème d'Optimisation

Au premier chapitre, nous avons exposé l'organigramme du RS (figue 1.13) pour la résolution d'un problème complexe quelconque. Dans cette section, nous nous intéressons à son application au problème d'optimisation des paramètres de configuration de la politique VTDS. Chaque composant spécifique à l'algorithme comme la solution et la fonction objectif, sera adapté à ce problème.

Solution : c'est un vecteur des paramètres de 6-uplet $(N_1, N_2, K, \lambda, \mu_1, \mu_2)$

Fonction Objectif : la qualité de la solution est déterminée par la valeur de la fonction objectif f,

$$f(s) = 0.5 * E'(s) + 0.5 * D'(s); \tag{2.15}$$

où $E'(s)$ représente l'énergie normalisée (standardisée) et $D'(s)$ représente le délai normalisé (standardisé).

Initialisation : Pour chaque variable de décision (paramètre) N_1, N_2, K, λ, μ_1 et μ_2 nous calculons une valeur médiane de son espace de recherche. Ainsi, une solution initiale est générée en attribuant les médianes aux variables de décision, tout en vérifiant les contraintes

C. Ensuite, nous calculons un écart-type pour chaque paramètre qui va préciser l'intervalle de voisinage.

Génération d'une Solution Voisine :

— Pour chaque variable de décision (paramètre), nous calculons une moyenne pondérée entre la valeur actuelle et la médiane de l'espace de recherche, en utilisant des poids égaux (0,5).

$$Moyenne = 0,5 \cdot Valeur\_actuelle + 0,5 \cdot Mediane \tag{2.16}$$

— Un nouveau paramètre est généré en utilisant une distribution gaussienne centrée sur la moyenne calculée, avec un écart-type spécifique au paramètre.
— Nous vérifions si le nouveau paramètre respecte les limites définies de l'espace de recherche, ainsi que les contraintes C.
— Nous ajustons le nouveau paramètre pour qu'il corresponde au pas spécifié le plus proche.

2.4.2 Adaptation de la Recherche Tabou (RT)

L'organigramme de la RT (figure 1.15) fut présenté de manière générale au niveau du chapitre 1, nous allons maintenant nous intéresser à son adaptation au problème d'optimisation des paramètres de configuration de la politique VTDS. Les aspects phares de cette approche seront redéfinies comme suit :

Solution : tout comme RS une solution pour RT est un vecteur des paramètres de 6-uplet $(N_1, N_2, K, \lambda, \mu_1, \mu_2)$.

Fonction Objectif : la fonction objectif est la même que celle de RS.

Liste Tabou : une liste tabou enregistre les mouvements récents (changements de paramètres) pour éviter de revisiter les mêmes solutions.

Initialisation : Comme pour le RS, la solution initiale pour la RT est générée en attribuant les médianes aux variables de décision. Ensuite, les écarts-types pour chaque paramètre sont calculés pour déterminer l'intervalle de voisinage.

Génération de Voisinage : En utilisant le même principe que la génération de voisinage dans le RS, nous générons plusieurs solutions voisines à partir d'une solution courante. Cela permet d'explorer de manière exhaustive l'espace de solutions autour de la solution actuelle et d'augmenter la probabilité de trouver une meilleure solution.

Algorithme 2 : Algorithme de Recuit Simulé

Data : $MaxItration$; $Temp\_ini$; $Temp\_fin$; $TauxRef$; $espaces\_recherche$; $parametres\_pas$.
Result : $best\_sol$; $best\_cout$.
Initialiser les paramètres de la solution initiale Sol_ini par la médiane ;
$cout\_ini = f(Sol\_ini)$;
Calculer les écarts-types;
$sol\_courante \leftarrow Sol\_ini$;
$cout\_courant \leftarrow cout\_ini$;
$best\_sol \leftarrow Sol\_ini$;
$best\_cout \leftarrow cout\_ini$;
$temperature \leftarrow Temp\_ini$;
while $temperature > Temp\_fin$ **do**
  while *MaxItération n'est pas atteint* **do**
    Générer une solution voisine $sol\_voisine$;
    Évaluer la solution voisine $cout\_voisine$;
    if $cout\_voisine < cout\_courant$ **then**
      $sol\_courante \leftarrow sol\_voisine$;
      $cout\_courant \leftarrow cout\_voisine$;
      if $cout\_voisine < best\_cout$ **then**
        $best\_sol \leftarrow sol\_voisine$;
        $best\_cout \leftarrow cout\_voisine$;
      end
    else
      $probabilite\_acceptation = e^{-(cout\_voisine - cout\_courant)/temperature}$;
      if *une valeur aléatoire rand* $[0, 1] < probabilite\_acceptation$ **then**
        $sol\_courante \leftarrow sol\_voisine$;
        $cout\_courant \leftarrow cout\_voisine$;
      end
    end
  end
  $temperature \leftarrow temperature \times tauxRef$;
end
Retourner ($best\_sol, best\_cout$);

Algorithme 3 : Algorithme de la Recherche Tabou

Data : $MaxItration$; $NbrV$; $durTabIni$; $espaces\_recherche$; $parametres\_pas$.
Result : $best\_sol$; $best\_cout$.
Initialiser les paramètres de la solution initiale Sol_ini par la médiane ;
$cout\_ini = f(Sol\_ini)$;
Calculer les écarts-types;
$sol\_courante \leftarrow Sol\_ini$;
$cout\_courant \leftarrow cout\_ini$;
$best\_sol \leftarrow Sol\_ini$;
$best\_cout \leftarrow cout\_ini$;
$liste\_tabou \leftarrow []$;
while *MaxItération n'est pas atteint* **do**
    Générer de $NbrV$ solutions voisines;
    Évaluer les $NbrV$ solutions voisines (calculer les $f(s)$);
    Identifier la meilleure solution voisine $best\_sol\_voisine$ et son coût $best\_cout\_voisine$;
    if $best\_sol\_voisine \notin liste\_tabou$ **then**
        $sol\_courante \leftarrow best\_sol\_voisine$;
        $cout\_courant \leftarrow best\_cout\_voisine$;
        Ajouter $best\_sol\_voisine$ à $liste\_tabou$ avec sa $dureeTab \leftarrow durTabIni$;
        if $best\_cout\_voisine < best\_cout$ **then**
            $best\_sol \leftarrow best\_sol\_voisine$;
            $best\_cout \leftarrow best\_cout\_voisine$;
        end
        Réduire $dureeTab$ de chaque solution dans $liste\_tabou$;
        Retirer celles dont la durée est écoulée;
    end
end
Retourner $(best\_sol, best\_cout)$;

2.4.3 Adaptation de l'optimisation par essaim de particules (OEP)

Pour adapter cette méthode à notre problème d'optimisation des paramètres de la politique de vacances à deux seuils, les aspects fondamentaux de cette approche seront décrits de la façon suivante :

Particule : Elle représente une solution dans un espace de recherche de dimension D ($D = 6$). Une particule (une solution) i de la population est modélisée par son vecteur de position courante $x_i = (x_{i1}, x_{i2}, x_{i3}, xi4, x_{i5}, x_{i6}) = (N_1, N_2, K, \lambda, \mu_1, \mu_2)$ et son vecteur vitesse de déplacement $v_i = (v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5}, v_{i6})$.

Fonction objectif : Elle détermine la qualité d'une position X_i par la valeur $f(X_i) = 0.5 \cdot E'(X_i) + 0.5 \cdot D'(X_i)$, où $E'(X_i)$ représente l'énergie normalisée (standardisée) et $D'(Xi)$ représente le délai normalisé (standardisé).

Meilleures positions : Chaque particule i mémorise la meilleure position par laquelle elle est déjà passée (Personal Best), que l'on note $P_{best_i} = (p_{best_{i1}}, p_{best_{i2}}, p_{best_{i3}}, p_{best_{i4}}, p_{best_{i5}}, p_{best_{i6}})$. Ainsi que la meilleure position atteinte par ses particules voisines l'essaim (Global Best), notée $G_{best} = (g_{best_1}, g_{best_2}, g_{best_3}, g_{best_4}, g_{best_5}, g_{best_6})$.

Initialisation : Les positions initiales des particules et leurs vitesses sont générées aléatoirement dans l'espace de recherche, en respectant les bornes définies pour chaque paramètre. Cette initialisation aléatoire permet une exploration diversifiée de l'espace de solutions possibles.

Déplacement : Chaque particule i va se déplacer entre les deux itérations t et $t + 1$, en fonction de sa vitesse, de la meilleure position personnelle et de la meilleure position globale suivant les deux équations 1.1 et 1.2 presentées dans la sous-section 1.3.4.2 :

Algorithme 4 : Algorithme OEP

Data : $MaxItration$; $NbrP$; $espaces\_recherche$; $parametres\_pas$; W ; C_1 ;C_2 ;V_{max}.
Result : G_{best} ; $G_{best}\_fitness$.
Initialiser aléatoirement les x_i des $NbrP$ particules;
Initialiser les v_i par V_{max};
$fitness \leftarrow []$;
Calculer les $f(x_i)$ et les mettre dans $fitness$;
Initialiser les $p_{best}[i]$ par les x_i;
Initialiser les $P_{best}\_fitness[i]$ par les $fitness[i]$;

Algorithme 5 : Algorithme OEP (suite)

$G_{best} \leftarrow meilleure\_position$;
$G_{best}\_fitness \leftarrow Cout(meilleure\_position)$;
while *MaxItération n'est pas atteint* **do**
  foreach *particule i* **do**
    Déplacer la position $x_i(t)$ en utilisant (W, C_1, C_2) ;
    $fitness[i] \leftarrow f(x_i(t+1))$;
    if $fitness[i] < P_{best}\_fitness[i]$ **then**
      $p_{best}[i] \leftarrow x_i$;
      $P_{best}\_fitness[i] \leftarrow fitness[i]$;
    end
  end
  $idmin \leftarrow$ l'indice de la plus petit valeur dans $fitness$;
  if $fitness[idmin] < G_{best}\_fitness$ **then**
    $G_{best} \leftarrow x_{idmin}$;
    $G_{best}\_fitness \leftarrow fitness[idmin]$;
  end
end
Retourner ($G_{best}, G_{best}\_fitness$);

2.5 Conclusion

Dans ce chapitre, nous avons commencé par définir la politique de conservation d'énergie adoptée. Ensuite, nous avons présenté la modélisation du système en utilisant les files d'attentes avec vacance de travail, permettant une représentation précise de la dynamique du système. Ainsi, nous avons détaillé le calcul de l'énergie consommée et du délai d'attente, deux métriques essentielles pour évaluer les performances des capteurs sans fil. Cette évaluation nous a permis de comprendre les points critiques influençant la performance globale des RCSFs. Enfin, nous avons formulé le problème d'optimisation en soulignant l'influence cruciale de la politique de conservation d'énergie sur les performances du système et discuté des méthodes de résolution basées sur des métaheuristiques, notamment le Recuit Simulé (RS), la Recherche Tabou (RT) et l'Optimisation par Essaim de Particules (OEP). Chaque méthode a été adaptée spécifiquement à notre problème d'optimisation pour minimiser l'énergie consommée tout en réduisant le délai d'attente dans les capteurs sans fil.

Chapitre 3

Implémentation, Expérimentations et Analyse des Résultats

3.1 Introduction

Les expérimentations sont essentielles pour la validation de nos algorithmes et approches, cette étape nécessite de déterminer les caractéristiques des machines utilisées ainsi que la description de la partie logiciel dont nous avons eu besoin, mais surtout une bonne organisation des expérimentations. C'est justement l'objet de ce chapitre. Nous commencerons par détailler l'environnement de développement, en décrivant les caractéristiques matérielles des machines utilisées et les logiciels employés. Ensuite, nous nous concentrerons sur les implémentations, fournissant des détails sur les algorithmes développés et la méthodologie suivie. Nous aborderons ensuite les expérimentations, en décrivant les différents espaces de recherche utilisés pour la validation et l'évaluation des méthodes de résolution réalisées. Enfin, nous présenterons les résultats obtenus, accompagnés d'analyses et de comparaisons approfondies, afin de démontrer l'efficacité des approches proposées.

3.2 Environnement de Développement

Comme pour toute expérimentation, l'environnement de développement est crucial pour la réalisation des implémentations et des tests, ainsi que pour l'interprétation précise des résultats obtenus.

3.2.1 Matériel

Nous avons utilisé une machine (Laptop) possédant les caractéristiques et capacités décrites avec la table 3.2 suivante :

Table 3.1 – Caractéristiques de la Machine Utilisée.

Marque	DELL
Mémoire RAM	8 Go
Processeur	Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 1.38 GHz
Type de l'OS	Windows 11 64 bits

3.2.2 Logiciel

Le développement de nos algorithmes a été exclusivement fait sous système d'exploitation **Windows**, celui-ci muni des logiciels suivants :

Visual Studio Code : est un éditeur de code épuré avec un support pour les opérations de développement telles que le débogage, l'exécution de tâches et le contrôle de version.

Python : est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. Il a plus ou moins récemment fait surface et qui est de plus en plus utilisé depuis surtout en vu de sa simplicité et polyvalence.

Matlab : est un environnement de programmation et de calcul numérique développé par *MathWorks*. Il est couramment utilisé dans les domaines de l'ingénierie, des sciences, de la finance, et de la recherche académique.

3.3 Implémentations

Les algorithmes des métaheuristiques testés dans ce chapitre ont été développés "from scratch" en utilisant le langage de programmation $Python$. Pour le calcul de l'énergie consommée et du délai d'attente selon l'approche VTDS, le programme s'appuie sur $Matlab$ pour résoudre le système d'équations détaillé dans la sous-section 2.2.3 et obtenir le vecteur des probabilités stationnaires. Cet appel à $Matlab$ a entraîné un impact notable sur le temps d'exécution, ajoutant une certaine latence au processus global.

Concernant les interfaces graphiques, nous avons opté pour utiliser $customtkinter$ qui est une bibliothèque basée sur $Python$.

3.3.1 Présentation des Interfaces Graphiques

Pour notre travail effectué dans le contexte de l'optimisation des paramètres de la politique de vacances à deux seuils (VTDS) pour conserver la consommation d'énergie tout en minimisant le délai d'attente, nous avons développé une interface principale. Cette fenêtre

permet d'évaluer une combinaison de paramètres en calculant le délai d'attente et l'énergie consommée par un capteur avec cette configuration. De plus, elle permet à l'utilisateur d'optimiser les paramètres de la VTDS en utilisant diverses méthodes (figure 3.2) de résolution telles que la recherche exhaustive, le recuit simulé, la recherche Tabou et l'optimisation par essaim de particules.

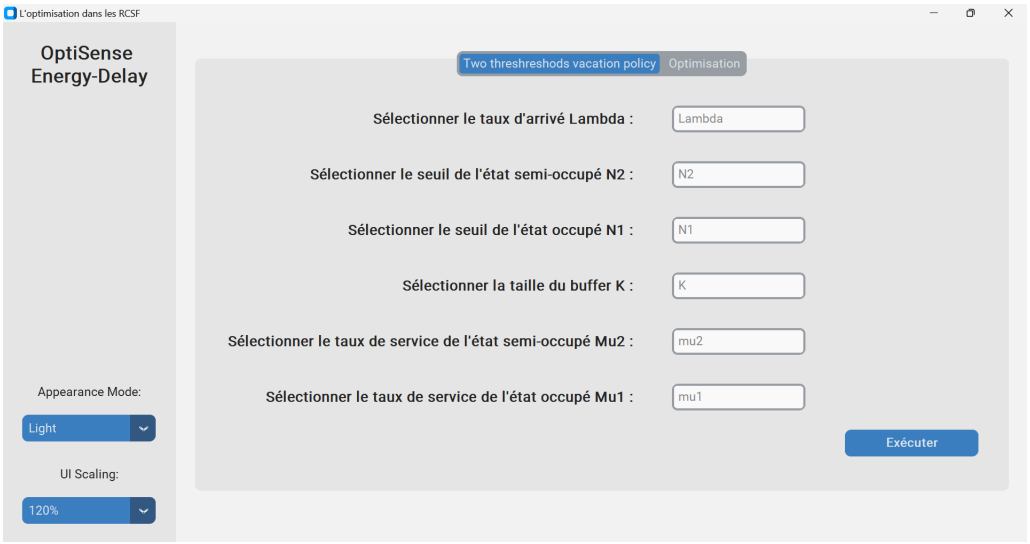

FIGURE 3.1 – Fenêtre d'Évaluation d'une Combinaison de Paramètres de la VTDS.

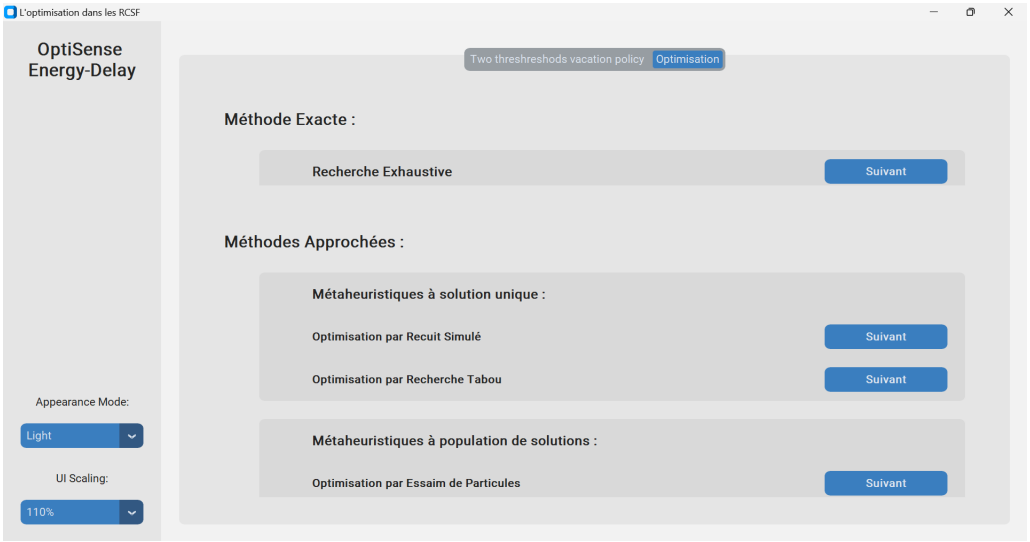

FIGURE 3.2 – Fenêtre des Méthodes d'Optimisation.

Pour les fenêtres d'optimisation, l'utilisateur peut sélectionner l'espace de recherche souhaité ainsi que préciser les valeurs des paramètres de la métaheuristique choisie. Par exemple, les interfaces dédiées à l'optimisation par recuit simulé permettent de choisir un espace de recherche, comme illustré dans la figure 3.3, et de définir les paramètres de la métaheuristique ainsi que les poids de la fonction objectif, comme montré dans la figure

3.4. Ces options offrent une flexibilité à l'utilisateur pour adapter l'optimisation en fonction des besoins spécifiques du système.

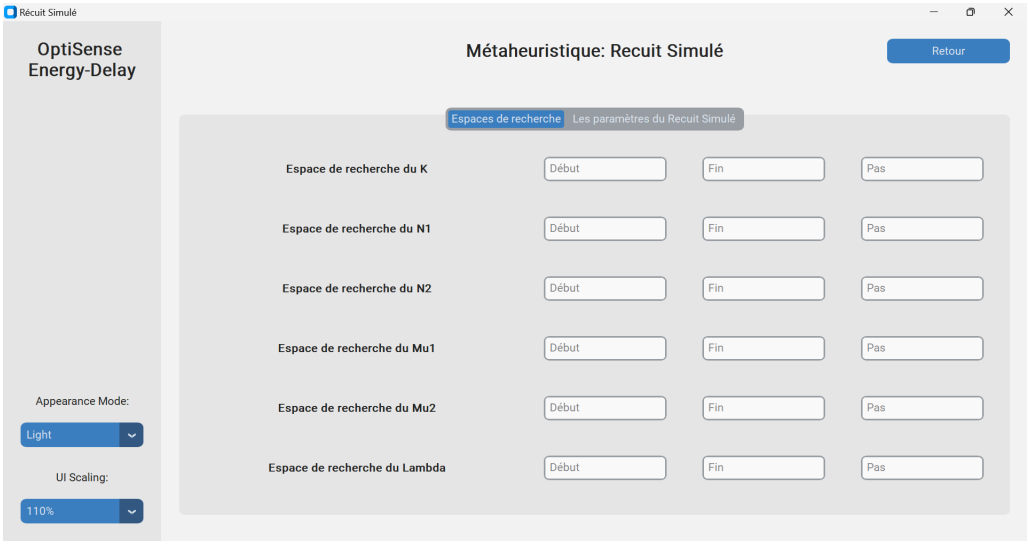

Figure 3.3 – Fenêtre d'Optimisation en Utilisant le RS (Espace de Recherche).

Figure 3.4 – Fenêtre d'Optimisation en Utilisant le RS (Paramètres).

L'exécution d'une méthode d'optimisation aboutit à l'affichage d'une fenêtre de résultats (figure 3.5). Celle-ci dévoile la solution optimale obtenue, assortie de la valeur de la fonction objectif pour l'énergie et le délai, ainsi que de deux graphiques illustrant l'évolution de ces derniers en fonction du temps.

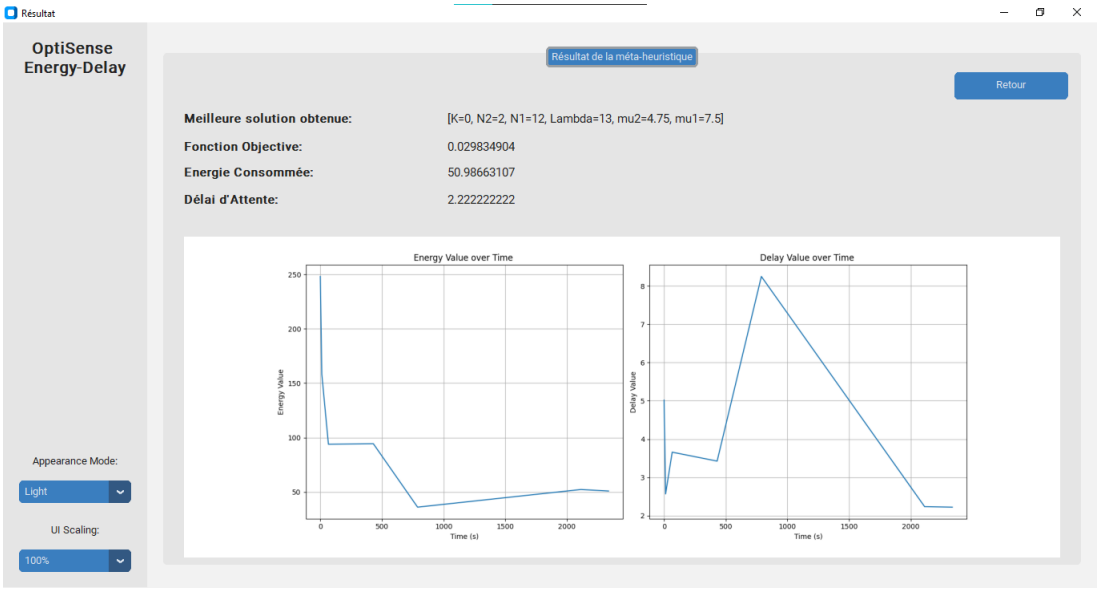

FIGURE 3.5 – Fenêtre du Résultat d'Optimisation.

3.4 Expérimentations

Pour valider l'efficacité des métaheuristiques développées et leur capacité à optimiser les paramètres de la VTDS pour les capteurs sans fil, nous avons mené une série d'expérimentations. Ces tests ont été conçus pour comparer les performances des algorithmes en termes de minimisation de la consommation d'énergie et du délai d'attente. Afin de déterminer la meilleure solution possible, chaque métaheuristique a été testée sous différentes configurations, avec des paramètres variés, pour identifier les combinaisons optimales.

Dans tout ce qui suit, nous fixerons les valeurs des énergies dans l'équation 2.12 comme suit [4] :

- $EC_I = 10$,
- $EC_{SB} = 25$,
- $EC_B = 500$,
- $EC_{Tx} = 5$,
- $EC_s = 300$.

3.4.1 Espaces de Recherche

Dans notre étude, nous avons travaillé avec deux espaces de recherche distincts, comme détaillé dans la table 3.2. Le premier espace de recherche, restreint, a été conçu pour permettre une évaluation de toutes les solution possibles. Cette approche nous a permis d'obtenir une référence fiable avec laquelle nous avons pu comparer nos algorithmes afin de vérifier s'ils atteignent le minimum attendu, validant ainsi leur capacité et leur efficacité. Cette validation était cruciale étant donné que les algorithmes ont été implémentés from

scratch. Une fois cette base de référence établie, nous avons pu procéder à des tests plus ambitieux dans un espace de recherche large, permettant une évaluation plus complète de la performance des algorithmes.

Le second espace de recherche, large, a été employé pour mener des expérimentations objectives à plus grande échelle. Cet espace étendu a permis de tester les métaheuristiques dans des conditions plus variées et complexes, fournissant ainsi une évaluation approfondie de leurs performances en termes de qualité de solution et de temps d'exécution. L'objectif ultime était d'identifier la solution optimale des paramètres de la VTDS.

Table 3.2 – Espaces de Recherche Utilisés

Espace de recherche I	
λ	[0,25 ; 0.75] pas = 0.25
μ_2	[0,25 ; 0.75] pas = 0.25
μ_1	$[\mu_2 + 0,25 ; 5]$ pas = 0.25
N_2	[1 ; 6] pas = 1
N_1	$[N_2 + 2 ; 16]$ pas = 1
K	$[N_1 + 4 ; 20]$ pas = 1

Espace de recherche II	
λ	[0,25 ; 5] pas = 0.25
μ_2	[0,25 ; 5] pas = 0.25
μ_1	[5 ; 10] pas = 0.25
N_2	[1 ; 48] pas = 1
N_1	[2 ; 49] pas = 1
K	[3 ; 50] pas = 1

3.4.2 Espace de Recherche I

Nous allons présenter en détail les tests effectués sur l'espace de recherche I ainsi que leurs résultats. Ces tests ont pour but d'évaluer l'efficacité et la stabilité des différentes métaheuristiques.

3.4.2.1 Méthode de Référence

Cette méthode de référence implique de calculer l'énergie, le délai et la fonction objectif pour toutes les combinaisons de l'$Espace\ de\ recherche\ I$ avec la politique VTDS. Les résultats obtenus, dont les cinq meilleurs sont présentés dans la table 3.3 ci-dessous, servent comme une référence pour évaluer les performances des métaheuristiques. La durée d'exécution de cette méthode est de 5 jours, 2 heures, 30 minutes et 7 secondes.

Table 3.3 – Meilleurs Résultats de la Méthode de Référence

$\boldsymbol{\lambda}$	$\boldsymbol{K}$	$\boldsymbol{N_1}$	$\boldsymbol{N_2}$	$\boldsymbol{\mu_1}$	$\boldsymbol{\mu_2}$	$\boldsymbol{EC}$	$\boldsymbol{\bar{W}}$	**Fct objectif**
0,25	7	3	1	5	0,75	87,637202	1,393	0,092605052
0,25	[8 ;20]	3	1	5	0,75	87,637203	1,393	0,092605053
0,25	7	3	1	4,75	0,75	87,871150	1,397	0,092847822
0,25	[8 ;20]	3	1	4,75	0,75	87,871151	1,397	0,092847823
0,25	[7 ;20]	3	1	4,5	0,75	88,131995	1,400	0,093118771

Nous passons maintenant aux résultats des trois métaheuristiques implémentées sur ce premier espace de recherche.

3.4.2.2 Recuit Simulé

Pour optimiser notre problème à l'aide de la métaheuristique RS, nous avons fixé la température minimale à 0,1 et le taux de refroidissement à 0,99, puis nous avons varié la température initiale avec des valeurs de 25, 50, 100 et 200. Pour chaque valeur de température initiale, nous avons effectué 10 tests distincts afin d'évaluer les performances et la stabilité de l'algorithme. Les résultats obtenus pour chaque série de tests sont présentés dans les tables 3.4, 3.5, 3.6 et 3.7 respectivement. Ces tables fournissent une vue d'ensemble des performances de RS sous différentes conditions de température initiale, mettant en lumière les variations de consommation d'énergie, de délai, de fonction objectif et du temps d'exécution.

Table 3.4 – Performances du RS avec une Température Initiale de 25.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	8	3	1	4,25	0,75	88,42	1,404	0,0934	52m :7s
Test 2	0,25	8	4	1	4	0,75	84,89	1,694	0,0944	54m :40s
Test 3	0,25	8	3	1	4,75	0,75	87,87	**1,396**	**0,0928**	56m :41s
Test 4	0,25	8	4	1	5	0,75	84,42	1,686	0,0939	51m :56s
Test 5	0,25	8	4	1	3,5	0,75	85,23	1,700	0,0948	51m :44s
Test 6	0,25	8	3	1	4,5	0,75	88,13	1,400	0,0931	50m :45s
Test 7	0,25	13	9	1	4,25	0,75	**81,82**	1,994	0,0960	50m :42s
Test 8	0,25	10	6	1	3,25	0,75	82,44	1,934	0,0957	50m :23s
Test 9	0,25	8	4	1	5	0,75	84,42	1,686	0,0939	**50m :18s**
Test 10	0,25	8	3	1	3,5	0,75	89,56	1,421	0,0946	50m :28s

Table 3.5 – Performances du RS avec une Température Initiale de 50.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	8	3	1	4	0,75	88,75	1,409	0,0937	57m :02s
Test 2	0,25	8	4	2	5	0,75	**56,87**	3,479	0,0958	57m :16s
Test 3	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	58m :41s
Test 4	0,25	8	4	1	5	0,75	84,42	1,686	0,0939	57m :22s
Test 5	0,25	8	3	1	3,75	0,75	89,13	1,415	0,0941	58m :39s
Test 6	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	57m24s
Test 7	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	58m :11s
Test 8	0,25	9	5	1	2,5	0,75	83,72	1,865	0,09588	59m :47s
Test 9	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	57m :19s
Test 10	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	**56m :52s**

Table 3.6 – Performances du RS avec une Température Initiale de 100.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :3m :05s
Test 2	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	**1h :2m :56s**
Test 3	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :2m :59s
Test 4	0,25	9	5	1	4	0,75	83,13	1,851	0,0952	1h :3m :24s
Test 5	0,25	10	6	1	4,5	0,75	**82,31**	1,930	0,0956	1h :3m :26s
Test 6	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :3m :39s
Test 7	0,25	8	4	1	4,5	0,75	84,63	1,686	0,0942	1h :4m :21s
Test 8	0,25	8	3	1	4	0,75	88,75	1,409	0,0937	1h :4m :37s
Test 9	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :4m :04s
Test 10	0,25	9	5	1	4,5	0,75	83,03	1,849	0,0950	1h :3m :55s

Table 3.7 – Performances du RS avec une Température Initiale de 200.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :13m :30s
Test 2	0,25	9	5	1	3	0,75	83,46	1,859	0,0955	1h :13m :05s
Test 3	0,25	11	7	1	3,25	0,75	**82,05**	1,970	0,0959	1h :13m :27s
Test 4	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :12m :45s
Test 5	0,25	8	3	1	3	0,75	90,65	1,438	0,0957	1h :13m :13s
Test 6	0,25	10	6	1	5	0,75	82,28	1,929	0,0955	1h :13m :25s
Test 7	0,25	8	4	1	5	0,75	84,42	1,686	0,0939	1h :11m :45s
Test 8	0,25	8	3	1	3,25	0,75	90,06	1,429	0,0951	1h :11m :36s
Test 9	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	1h :13m :14s
Test 10	0,25	9	5	1	4	0,75	83,13	1,851	0,0952	**1h :09m :21s**

Afin de valider la stabilité de l'algorithme de RS, nous avons calculé la variance de la fonction objectif pour chaque température initiale. Les résultats sont presentés dans la figure 3.6

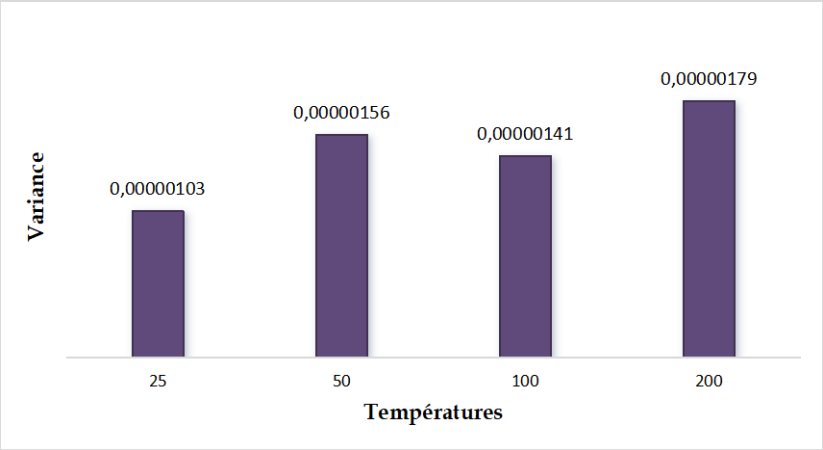

FIGURE 3.6 – Histogramme de la Variance de la Fonction Objectif en Fonction de la Température Initiale pour le RS.

Analyse

D'après les tables, la métaheuristique RS a atteint la valeur minimale de la fonction objectif déterminée par la méthode de référence, soit 0,0926. En outre, l'analyse de l'histogramme de la variance révèle que la variance maximale observée est de 0,00000179, démontrant ainsi une stabilité significative de l'algorithme.

3.4.2.3 Recherche Tabou

En utilisant la métaheuristique la RT, nous avons fixé le nombre de voisins à et la durée de la liste tabou à 5. Ensuite, nous avons expérimenté avec différentes valeurs du nombre d'itérations : 5, 6, 7 et 8. Pour chaque valeur d'itération, nous avons effectué 10 tests distincts afin d'analyser les performances et la stabilité de l'algorithme. Les résultats de ces tests sont présentés dans les tables 3.8, 3.9, 3.10 et 3.11 respectivement.

TABLE 3.8 – Performances de la RT avec 5 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	12	8	1	4	0,75	81,88	1,987	0,0960	2m :54s
Test 2	0,25	8	3	1	3	0,75	90,65	1,438	0,0957	2m :52s
Test 3	0,75	11	7	5	4,75	0,75	107,04	4,135	0,1468	2m :53s
Test 4	0,25	12	8	1	1,5	0,75	81,98	1,990	0,0961	2m :51s
Test 5	0,25	8	3	1	3,75	0,75	89,13	**1,415**	**0,0941**	2m :51s
Test 6	0,25	12	8	3	3,75	0,75	**45,57**	5,956	0,1207	**2m :45s**
Test 7	0,25	9	5	1	2,75	0,75	83,58	1,861	0,0957	2m :54s
Test 8	0,25	9	5	1	2,75	0,75	83,58	1,861	0,0957	2m :54s
Test 9	0,25	9	5	1	1,25	0,75	85,48	1,909	0,0979	2m :52s
Test 10	0,25	8	4	1	3	0,75	85,69	1,709	0,0953	2m :52s

Table 3.9 – Performances de la RT avec 6 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	14	10	1	1,75	0,75	**81,81**	1,998	0,0961	3m :26s
Test 2	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	3m :30s
Test 3	0,25	8	3	1	3,75	0,75	89,13	1,415	0,0941	3m :34s
Test 4	0,25	8	3	1	4	0,75	88,75	1,409	0,0937	3m :32s
Test 5	0,25	8	3	1	5	0,75	87,63	**1,39**	**0,0926**	3m :20s
Test 6	0,25	10	6	1	4,5	0,75	82,31	1,930	0,0956	3m :26s
Test 7	0,25	14	10	1	1,25	0,75	**81,81**	1,998	0,0961	**3m :19s**
Test 8	0,25	11	7	1	4,5	0,75	82,00	1,969	0,0958	3m :24s
Test 9	0,25	8	3	1	5	0,75	87,63	**1,393**	**0,0926**	3m :20s
Test 10	0,25	14	10	1	1,5	0,75	**81,81**	1,998	0,0961	3m :25s

Table 3.10 – Performances de la RT avec 7 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	9	5	1	3,5	0,75	83,275	1,854	0,09536	**3m :57s**
Test 2	0,25	13	9	1	3,25	0,75	81,828	1,994	0,09609	**3m :57s**
Test 3	0,25	8	3	1	5	0,75	87,637	**1,393**	**0,09260**	3m :58s
Test 4	0,25	15	11	1	3,25	0,75	81,792	1,999	0,09612	**3m :57s**
Test 5	0,25	11	7	1	2,25	0,75	82,139	1,973	0,09605	3m :58s
Test 6	0,25	13	9	1	3	0,5	**73,370**	3,835	0,11455	3m :59s
Test 7	0,25	12	8	1	2,75	0,75	81,906	1,988	0,09606	4m :01s
Test 8	0,25	9	5	1	5	0,75	82,946	1,847	0,09498	3m :58s
Test 9	0,25	15	11	1	2,25	0,75	81,793	1,999	0,09612	3m :58s
Test 10	0,25	14	10	1	0,5	0,75	81,870	2,002	0,09624	3m :59s

Table 3.11 – Performances de la RT avec 8 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	9	5	1	2,75	0,75	83,583	1,861	0,09571	4m :32s
Test 2	0,25	8	3	1	4	0,75	88,755	1,409	0,09376	4m :26s
Test 3	0,25	14	10	1	4,75	0,75	**81,799**	1,997	0,09611	4m :31s
Test 4	0,25	13	9	1	4,75	0,75	81,820	1,994	0,09608	4m :31s
Test 5	0,25	13	9	1	3	0,75	81,830	1,994	0,09609	**4m :25s**
Test 6	0,25	8	3	1	5	0,75	87,637	**1,393**	**0,09260**	4m :33s
Test 7	0,25	8	4	1	2,5	0,75	86,353	1,722	0,09609	4m :32s
Test 8	0,25	9	5	1	3,25	0,75	83,361	1,856	0,09546	4m :37s
Test 9	0,25	8	3	1	3,75	0,75	89,131	1,415	0,09415	4m :29s
Test 10	0,25	12	8	1	2,75	0,75	81,906	1,988	0,09606	4m :31s

Pour chaque valeur d'itération, nous avons calculé la variance de la fonction objectif sur les 10 tests effectués. Ces variances ont ensuite été illustrées sous forme d' un histogrammes dans la figure 3.7.

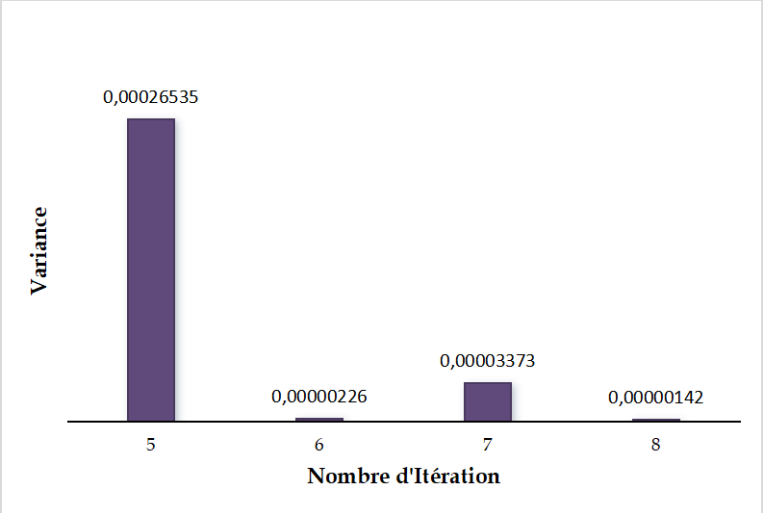

FIGURE 3.7 – Histogramme de la Variance de la Fonction Objectif en Fonction du Nombre d'Itération pour la RT.

Analyse

Le fait que la Recherche Tabou (RT) ait atteint la valeur minimale déterminée par la méthode de référence, particulièrement pour les nombres d'itérations de 6, 7 et 8, démontre l'efficacité de l'algorithme. La variance maximale de 0,00026535 observée pour 5 itérations atteste également de la stabilité du fonctionnement de cette métaheuristique.

3.4.2.4 Optimisation par Essaim de Particules

Pour optimiser notre problème en utilisant l'OEP, nous avons fixé le nombre de particules à 5 , le facteur d'inertie w et la vitesse maximale à 1, ainsi les coefficients d'accélération (c_1, c_2) à 1.5. Ensuite, nou avons varié le nombre d'itérations avec des valeurs de 5, 6, 8 et 10 . Pour chaque configuration, nous avons effectué 10 tests distincts afin d'évaluer l'algorithme. Les résultats de ces tests sont présentés dans les tables 3.12, 3.13, 3.14 et 3.15 respectivement.

Table 3.12 – Performances de l'OEP avec 5 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	19	15	1	1,5	0,75	81,78	1,999	0,09613	2m :51s
Test 2	0,75	10	6	1	5	0,25	126,08	3,296	0,15115	2m :55s
Test 3	0,25	9	5	1	4,5	0,75	83,03	**1,849**	**0,09508**	2m :57s
Test 4	0,25	11	7	1	2,5	0,75	82,11	1,972	0,09602	2m :55s
Test 5	0,5	12	8	5	4,5	0,75	**42,32**	7,993	0,14166	2m :55s
Test 6	0,25	10	6	1	5	0,75	82,28	1,929	0,09557	2m :56s
Test 7	0,25	11	7	2	5	0,75	53,94	3,945	0,09983	2m :57s
Test 8	0,25	14	10	1	1,25	0,75	81,81	1,998	0,09614	2m :56s
Test 9	0,25	14	10	1	5	0,75	81,79	1,997	0,09611	2m :55s
Test 10	0,25	16	12	2	4,25	0,75	53,75	3,999	0,10042	**2m :49s**

Table 3.13 – Performances de l'OEP avec 6 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	**3m :27s**
Test 2	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	3m :31s
Test 3	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	3m :28s
Test 4	0,25	16	12	2	1,5	0,75	**53,75**	3,999	0,10043	3m :32s
Test 5	0,25	14	10	1	5	0,75	81,79	1,997	0,09611	3m :28s
Test 6	0,25	12	8	1	3,25	0,75	81,89	1,987	0,09604	3m :51s
Test 7	0,25	12	7	1	5	0,75	81,99	1,968	0,09587	3m :32s
Test 8	0,25	17	9	1	5	0,75	81,81	1,994	0,09608	3m :30s
Test 9	0,25	10	6	1	2,75	0,75	82,53	1,936	0,09587	3m :34s
Test 10	0,25	17	13	1	1,5	0,75	81,78	1,999	0,09613	3m :31s

Table 3.14 – Performances de l'OEP avec 8 Itérations

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	13	3	1	5	0,75	87,63	**1,393**	**0,09260**	4m :14s
Test 2	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	4m :24s
Test 3	0,25	13	8	1	2,75	0,75	81,90	1,988	0,09607	4m :21s
Test 4	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	**4m :12s**
Test 5	0,25	10	4	1	5	0,75	84,42	1,686	0,09399	4m :22s
Test 6	0,25	14	9	1	3	0,75	**81,83**	1,995	0,09610	4m :22s
Test 7	0,25	9	5	1	5	0,75	82,94	1,848	0,09499	4m :22s
Test 8	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	4m :21s
Test 9	0,25	11	7	1	5	0,75	81,99	1,969	0,09587	4m :21s
Test 10	0,25	11	5	1	4,5	0,75	83,03	1,849	0,09508	4m :21s

TABLE 3.15 – Performances de l'OEP avec 10 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	14	7	1	4	0,75	82,02	1,970	0,09591	**5m :09s**
Test 2	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	5m :22s
Test 3	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	5m :19s
Test 4	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	5m :20s
Test 5	0,25	17	5	1	5	0,75	82,94	1,848	0,09498	5m :22s
Test 6	0,25	7	3	1	5	0,75	87,63	**1,393**	**0,09260**	5m :21s
Test 7	0,25	8	4	1	5	0,75	84,42	1,686	0,09399	5m :22s
Test 8	0,25	13	7	1	2,75	0,75	82,08	1,972	0,09600	5m :21s
Test 9	0,75	10	6	3	5	0,75	109,89	3,063	0,13438	5m :20s
Test 10	0,25	11	7	1	5	0,75	**81,99**	1,969	0,09587	5m :24s

Pour évaluer la stabilité de l'algorithme OEP, nous avons calculé la variance de la fonction objectif pour chaque nombre d'itérations et illustré ces résultats dans la figure 3.8.

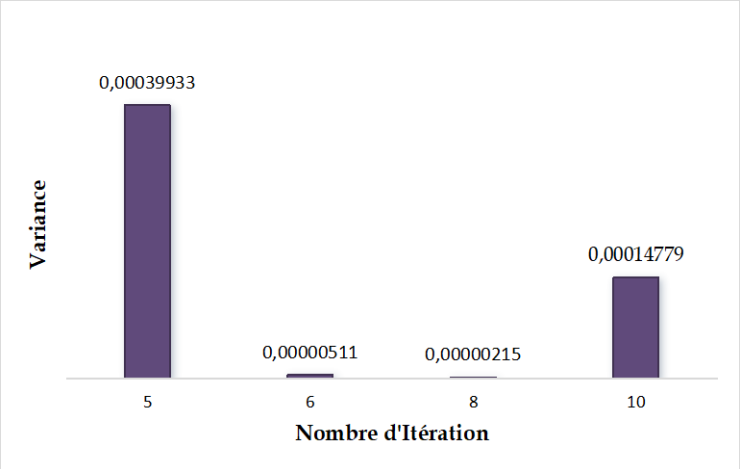

FIGURE 3.8 – Histogramme de la Variance de la Fonction Objectif en Fonction du Nombre d'Itération pour l'OEP.

Analyse

D'après les résultats des tests, nous remarquons que l'algorithme OEP a réussi à minimiser la fonction objectif à la valeur minimale trouvée par la méthode de référence, soit 0,09260. La variance des résultats atteint un maximum de 0,00039933, ce qui démontre la stabilité et l'efficacité de l'algorithme

3.4.2.5 Récapitulatif

En analysant les résultats obtenus par les trois métaheuristiques sur *l'Espace de Recherche I*, nous pouvons confirmer leur efficacité et leur stabilité dans la recherche de so-

lutions optimales. Le Recuit Simulé (RS), la Recherche Tabou (RT) et l'Optimisation par Essaim de Particules (OEP) ont tous démontré leur capacité à atteindre la valeur minimale de la fonction objectif identifiée par la méthode de référence, tout en maintenant une faible variance, signe de leur stabilité. Ces résultats valident l'utilisation de ces métaheuristiques pour optimiser les paramètres de configuration de la VTDS.

3.4.3 Espace de Recherche II

Avant de détailler les expérimentations des métaheuristiques sur *l'Espace de Recherche II*, il est important de rappeler que cet espace "large" a été spécialement conçu pour tester les algorithmes dans des conditions plus variées et complexes. Ces expérimentations permettront de comparer la qualité des solutions obtenues et les temps d'exécution et d'identifier la solution optimale des paramètres de la VTDS.

3.4.3.1 Recuit Simulé

Pour aborder notre problème d'optimisation avec la métaheuristique RS, nous avons fixé certains paramètres clés : une température minimale de 0,1 et un taux de refroidissement de 0,99. Ensuite, nous avons exploré diverses températures initiales : 50, 100 et 200. Pour chaque température, nous avons effectué 10 essais afin d'identifier le meilleur paramétrage qui donne les valeurs optimales de la fonction objectif et du temps d'exécution. Les résultats obtenus sont présentés dans les tables 3.16, 3.17 et 3.18 respectivement. Ces tableaux offrent un aperçu des performances du RS sous différentes températures initiales.

Table 3.16 – Performances du RS avec une Température Initiale de 50.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	9	3	2	5,75	3,25	53,62	2,308	0,03131	**55m :28s**
Test 2	0,25	14	11	1	6,5	4	85,95	**0,267**	0,04160	55m :36s
Test 3	0,25	10	4	3	10	4,25	41,98	4,233	0,03057	55m :47s
Test 4	0,25	19	13	3	6,25	1,25	43,52	5,000	0,03322	55m :57s
Test 5	0,75	23	17	7	8,5	4,5	60,11	4,267	0,03929	56m :4s
Test 6	0,5	23	17	7	9,25	3,5	49,14	6,333	0,03924	56m :13s
Test 7	0,25	13	12	2	7,5	4,75	50,98	2,222	**0,02983**	56m :24s
Test 8	0,25	18	13	3	8,75	3,25	**41,46**	4,333	0,03058	56m :36s
Test 9	0,5	18	12	4	7,75	4	57,20	3,286	0,03546	56m :45s
Test 10	0,25	13	7	3	7	3,25	**41,46**	4,333	0,03058	56m :54s

Table 3.17 – Performances du RS avec une Température Initiale de 100.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	18	12	2	8	4	51,15	**2,267**	0,03003	**57m :00s**
Test 2	0,75	11	10	6	7,25	3,5	63,66	3,695	0,03956	57m :08s
Test 3	0,25	14	13	3	7,75	3	41,58	4,364	0,03071	57m :15s
Test 4	0,25	19	13	3	8	4,5	41,07	4,235	0,03015	57m :25s
Test 5	0,25	14	13	3	6	1,75	42,61	4,667	0,03196	57m :33s
Test 6	0,25	15	9	2	7,25	3,75	51,22	2,286	**0,03011**	57m :44s
Test 7	0,5	18	17	7	7,5	4,25	48,68	6,267	0,03885	57m :50s
Test 8	0,25	12	6	4	8,5	1,75	39,28	6,638	0,03530	57m :59s
Test 9	0,25	16	10	6	6,25	3,5	**36,41**	10,308	0,04311	58m :06s
Test 10	0,75	13	12	7	8,75	4,5	60,12	4,267	0,03930	58m :12s

Table 3.18 – Performances du RS avec une Température Initiale de 200.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	20	14	4	5	3,25	37,74	6,333	0,03381	**58m :21s**
Test 2	0,25	16	10	5	7,25	3,25	**36,51**	8,333	0,03822	1h :14m :10s
Test 3	0,25	17	11	3	7	2	42,31	4,571	0,03158	1h :14m :18s
Test 4	0,25	11	5	4	5	2,75	40,51	6,369	0,03522	1h :14m :28s
Test 5	0,5	15	9	3	8	3,25	67,37	**2,364**	0,03799	1h :14m :34s
Test 6	0,25	13	7	3	8,5	3	41,58	4,364	0,03071	1h :14m :41s
Test 7	0,25	12	6	2	7,25	3	51,50	**2,364**	0,03043	1h :14m :48s
Test 8	0,25	11	5	3	8,25	4	41,28	4,265	**0,03032**	1h :14m :59s
Test 9	0,25	13	7	3	8,75	0,75	45,50	5,894	0,03640	1h :14m :49s
Test 10	0,25	14	13	3	10	2,75	41,71	4,400	0,03087	1h :14m :59s

Les moyennes de la fonction objectif et du temps d'exécution pour chaque série de dix tests à une température donnée sont illustrées dans les histogrammes de la figure 3.9.

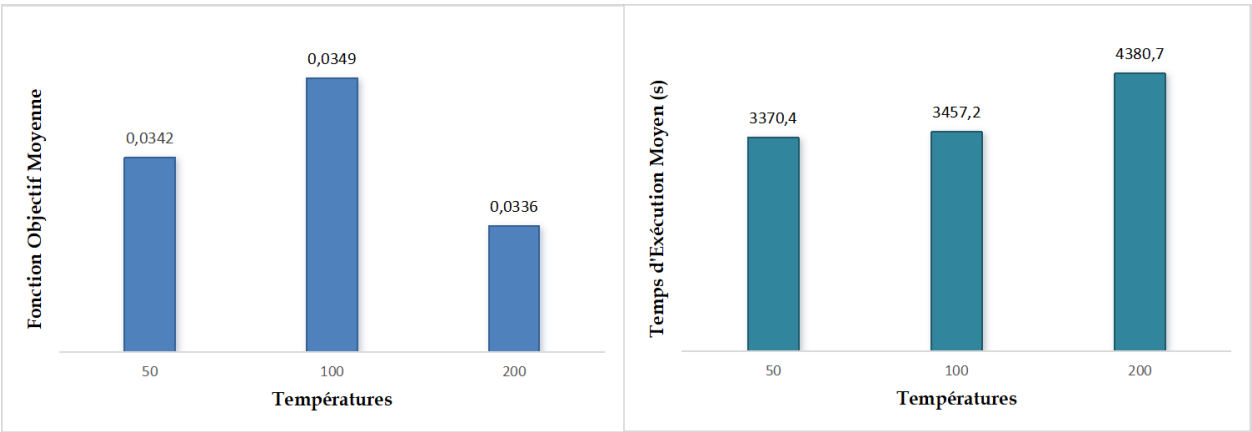

Figure 3.9 – Résultats du Recuit Simulé pour Différentes Températures.

Analyse

Ces résultats montrent que bien que la température initiale de 200 ait permis d'obtenir la meilleure fonction objectif moyenne, elle a également conduit à un temps d'exécution nettement plus élevé. En comparaison, une température initiale de 50 a offert un bon compromis entre la qualité de la solution et le temps d'exécution. Ainsi, la meilleure valeur de la fonction objectif obtenue était de 0.02983, trouvée avec une température initiale de 50. Cette solution optimale correspondait aux paramètres suivants : $(N_1^*, N_2^*, K^*, \lambda^*, \mu_1^*, \mu_2^*)$ = (12, 2, 13, 0.25, 7.5, 4.75)

3.4.3.2 Recherche Tabou

Avec la métaheuristique de la Recherche Tabou (RT), nous avons d'abord fixé le nombre de voisins et la durée tabou à 10. Ensuite, nous avons varié le nombre d'itérations en testant les valeurs 25, 50 et 100 itérations. Pour chaque configuration, 10 essais ont été effectués et représentés dans les table 3.19, 3.20 et 3.21 respectivement.

Table 3.19 – Performances de la RT avec 25 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	17	13	3	8	3	41,58	4,364	0,03071	28m :25s
Test 2	0,25	15	9	2	7,5	4,5	51,03	**2,235**	**0,02989**	27m :15s
Test 3	0,25	11	5	4	5,75	1,25	43,77	6,737	0,04106	26m :59s
Test 4	0,25	11	5	4	7,5	3,25	39,12	6,307	0,03440	27m :20s
Test 5	0,75	18	17	7	5	2	63,05	4,800	0,04203	27m :29s
Test 6	0,25	8	7	4	9,25	4,25	**37,40**	6,250	0,03656	**26m :50s**
Test 7	0,25	9	3	2	8,75	2,75	53,20	2,346	0,03120	27m :26s
Test 8	0,25	11	10	2	8,75	3	51,49	2,364	0,03043	28m :24s
Test 9	0,5	20	14	4	5,25	4,25	57,09	3,267	0,03535	28m :49s
Test 10	0,25	10	4	3	8,5	5	41,85	4,202	0,03044	28m :34s

Table 3.20 – Performances de la RT avec 50 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	13	12	2	8,75	5	50,94	**2,211**	**0,02978**	1h :7m :17s
Test 2	0,25	13	7	3	7	2,25	42,07	4,500	0,03129	**53m :54s**
Test 3	0,25	19	13	3	7	3,75	41,28	4,286	0,03037	55m :41s
Test 4	0,25	14	13	3	6,5	1,5	43,00	4,800	0,03248	56m :16s
Test 5	0,5	20	14	4	6,75	2,25	58,51	3,571	0,03679	56m :36s
Test 6	0,25	6	5	4	7	2,5	**40,01**	6,393	0,03504	56m :53s
Test 7	0,25	14	13	3	7	3,25	41,46	4,333	0,03058	57m :0s
Test 8	0,25	19	13	3	9	3,5	41,36	4,308	0,03047	54m :32s
Test 9	0,25	14	13	3	9	5	40,97	4,211	0,03004	54m :33s
Test 10	0,25	12	6	2	6,25	4,5	51,03	2,235	0,02989	54m :33s

Table 3.21 – Performances de la RT avec 100 Itérations.

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	12	6	2	8,75	4	51,15	2,267	0,03003	**1h :46m :08s**
Test 2	0,25	10	4	3	10	4,25	41,98	4,233	0,03057	1h :46m :52s
Test 3	0,25	19	13	3	5,5	4,5	41,07	4,235	0,03015	1h :46m :34s
Test 4	0,25	5	4	3	7	4,75	42,16	4,215	0,03062	1h :46m :14s
Test 5	0,25	14	13	3	8,25	4,5	41,07	4,235	0,03015	1h :46m :35s
Test 6	0,25	13	12	2	7	4	51,15	2,267	0,03003	1h :46m :50s
Test 7	0,25	15	14	4	7	2,25	**38,39**	6,500	0,03453	1h :47m :6s
Test 8	0,25	10	9	3	5,75	3,25	41,46	4,333	0,03058	1h :47m :50s
Test 9	0,25	9	8	2	7,25	5	50,94	**2,211**	**0,02978**	1h :48m :48s
Test 10	0,25	9	3	2	9	3,25	52,73	2,298	0,03086	1h :47m :30s

Afin de pouvoir déterminer les meilleurs paramètres qui optimisent la fonction objectif et minimisent le temps d'exécution, nous avons calculer la moyenne de la fonction objectif et du temps d'exécution pour chaque paramétrage. Les resultats sont affichés dans la figure 3.10.

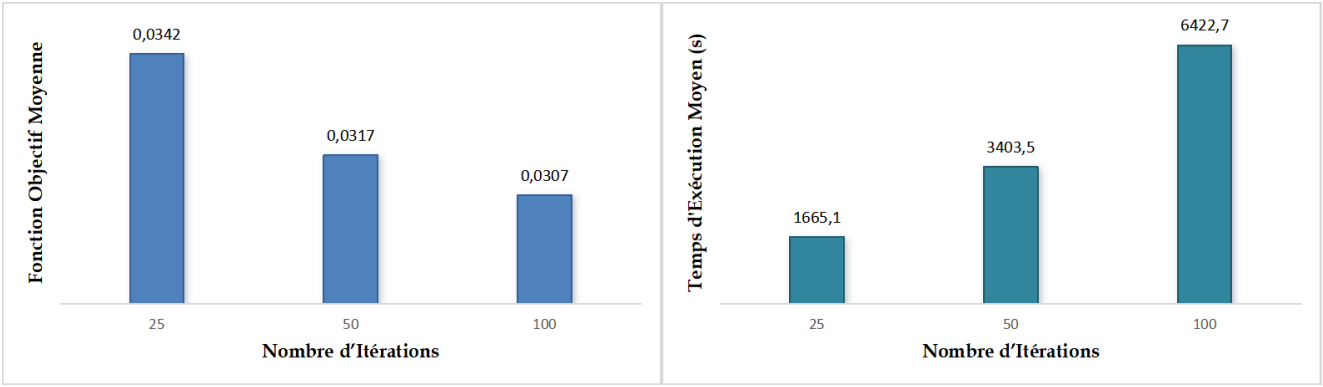

Figure 3.10 – Résultats de la Recherche Tabou pour Différentes Itérations.

Analyse

Les resultats des expérimentations de la RT montrent que, bien que l'augmentation du nombre d'itérations améliore la minimisation de la fonction objectif, cela se traduit également par une augmentation significative du temps d'exécution. L'application de la métaheuristique de la Recherche Tabou (RT) a permis d'obtenir une valeur optimale de la fonction objectif de 0.02978. Les deux solutions correspondantes sont les suivantes :

$$(N_1^*, N_2^*, K^*, \lambda^*, \mu_1^*, \mu_2^*) = \begin{cases} (12, 2, 13, 0.25, 8.75, 5) \\ ou \\ (8, 2, 9, 0.25, 7.25, 5) \end{cases}$$

3.4.3.3 Optimisation par Essaim de Particules

En appliquant la métaheuristique de l'Optimisation par Essaim de Particules (OEP), nous avons configuré plusieurs paramètres clés : la vitesse maximale et le facteur d'inertie à 1, ainsi que les coefficients d'accélération à 2. Le nombre d'itérations a été établi à 20. Afin d'explorer l'impact de la taille de la population, nous avons varié le nombre de particules à 10, 30 et 50. Pour chaque configuration de particules, nous avons réalisé 10 tests, de manière similaire aux autres métaheuristiques. Les resultats sont présentés dans les tables 3.22, 3.23 et 3.24 suivants.

Table 3.22 – Performances de l'OEP avec 10 Particules

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,75	32	27	8	8	5	58,39	4,902	0,04006	20m :40s
Test 2	0,25	40	33	2	10	5	50,94	**2,211**	**0,02978**	20m :52s
Test 3	0,25	40	29	4	10	5	**37,23**	6,211	0,03326	20m :42s
Test 4	0,5	37	15	5	6,5	5	51,87	4,222	0,03526	20m :35s
Test 5	1	46	31	9	10	5	66,82	4,250	0,04245	**20m :21s**
Test 6	0,5	45	17	7	8,25	5	48,35	6,222	0,03858	22m :13s
Test 7	0,5	38	32	7	3,5	3,25	49,34	6,364	0,03941	20m :29s
Test 8	0,25	35	3	1	10	0,25	72,23	2,735	0,04124	20m :43s
Test 9	0,25	41	31	4	5,25	5	**37,23**	6,211	0,03326	20m :46s
Test 10	1	41	27	9	8,75	5	66,82	4,250	0,04245	20m :34s

Table 3.23 – Performances de l'OEP avec 30 Particules

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	33	23	2	9	5	**50,94**	**2,211**	**0,02978**	1h :2m :48s
Test 2	0,25	37	22	2	8,25	5	**50,94**	**2,211**	**0,02978**	**1h :2m :18s**
Test 3	0,25	34	15	2	5,25	5	**50,94**	**2,211**	**0,02978**	1h :2m :42s
Test 4	0,25	34	16	2	5,25	5	**50,94**	**2,211**	**0,02978**	1h :5m :30s
Test 5	0,25	43	37	2	6	5	**50,94**	**2,211**	**0,02978**	1h :5m :20s
Test 6	0,25	44	21	2	7,5	5	**50,94**	**2,211**	**0,02978**	1h :4m :37s
Test 7	0,25	41	22	2	10	5	**50,94**	**2,211**	**0,02978**	1h :3m :12s
Test 8	0,25	47	42	2	5,25	5	**50,94**	**2,211**	**0,02978**	1h :3m :11s
Test 9	0,25	37	31	2	10	5	**50,94**	**2,211**	**0,02978**	1h :2m :38s
Test 10	0,25	48	36	2	5,75	5	**50,94**	**2,211**	**0,02978**	1h :2m :44s

Table 3.24 – Performances de l'OEP avec 50 Particules

	λ	K	N_1	N_2	μ_1	μ_2	EC	$\bar{W}$	FO	TE
Test 1	0,25	42	35	2	5,25	5	**50,94**	**2,211**	**0,02978**	**1h :44m :06s**
Test 2	0,25	39	26	2	9,5	5	**50,94**	**2,211**	**0,02978**	1h :44m :13s
Test 3	0,25	42	17	2	6,5	5	**50,94**	**2,211**	**0,02978**	1h :44m :10s
Test 4	0,25	38	21	2	7,75	5	**50,94**	**2,211**	**0,02978**	1h :46m :01s
Test 5	0,25	45	40	2	5,25	5	**50,94**	**2,211**	**0,02978**	1h :50m :33s
Test 6	0,25	50	21	2	5,75	5	**50,94**	**2,211**	**0,02978**	1h :46m :34s
Test 7	0,25	37	23	2	7,25	5	**50,94**	**2,211**	**0,02978**	1h :48m :38s
Test 8	0,25	45	38	2	5,25	5	**50,94**	**2,211**	**0,02978**	1h :52m :50s
Test 9	0,25	41	18	2	7,75	5	**50,94**	**2,211**	**0,02978**	1h :45m :59s
Test 10	0,25	40	35	2	5,5	5	**50,94**	**2,211**	**0,02978**	1h :44m :33s

Pour chaque série de tests, nous avons calculé les moyennes des valeurs obtenues pour la fonction objectif et le temps d'exécution. Ces moyennes ont ensuite été représentées sous forme d'histogrammes dans la figure 3.11, ce qui nous a permis de visualiser et de comparer les performances de l'algorithme OEP sous différentes configurations de particules. Cette méthode nous offre une perspective claire sur les performances globales de l'algorithme et les compromis entre la qualité des solutions et le temps d'exécution.

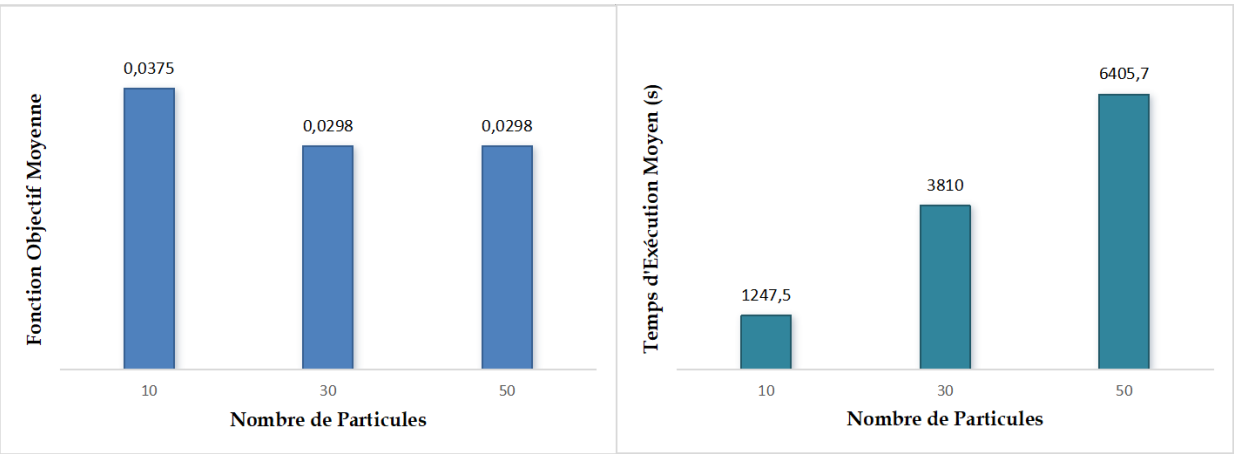

Figure 3.11 – Résultats de l'Algorithme OEP pour Différentes Itérations.

Analyse

Ces résultats montrent une amélioration notable de la fonction objectif moyenne lorsque le nombre de particules est augmenté de 10 à 30, tandis que le temps d'exécution augmente également de manière significative. L'augmentation du nombre de particules de 30 à 50 n'améliore pas la fonction objectif moyenne, mais continue d'augmenter considérablement le temps d'exécution, indiquant un point de rendement décroissant au-delà de 30 particules.

En examinant les résultats pour les populations de 30 et 50 particules, il est évident que touts les tests convergent vers la même valeur optimale de la fonction objectif (0.02978). Cependant, ces résultats sont obtenus avec des ensembles de paramètres (solutions) différents. Cela montre que, bien que la fonction objectif atteigne son minimum, les chemins pour y parvenir peuvent varier.

3.4.3.4 Comparaison entre RS, RT et OEP

Les trois métaheuristiques ont montré des résultats prometteurs sur l'espace de recherche large. Nous souhaitons désormais comparer les meilleurs résultats obtenus par chacune de ces métaheuristiques pour évaluer leur performance relative. Cette comparaison se concentrera sur deux aspects principaux : la minimisation de la fonction objectif et le temps d'exécution. Ces critères sont essentiels pour déterminer l'efficacité globale et la praticité des algorithmes dans des scénarios d'optimisation réels.

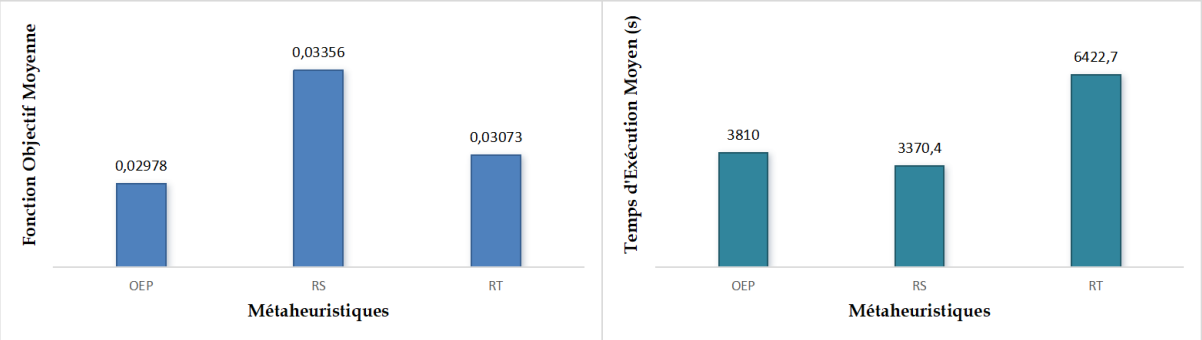

FIGURE 3.12 – Histogrammes des Meilleurs Résultats des Métaheuristiques pour l'Espace de Recherche II.

L'OEP se démarque comme la métaheuristique la plus performante sur l'espace de recherche large, offrant la meilleure optimisation de la fonction objectif (0.02978) avec un temps d'exécution relativement raisonnable (3810 s). La RT, bien qu'offrant une bonne qualité de solution (0.03073), est pénalisée par un temps d'exécution élevé (6422 s), tandis que le RS, avec un temps d'exécution plus court (3370.4 s), présente une performance légèrement inférieure en termes de qualité de solution (0.03356).

3.4.3.5 Comparaison des Performances entre les Politiques N-policy et VTDS Optimisées par OEP

Dans [32], la politique N-policy a été adoptée pour améliorer la conservation de l'énergie dans les capteurs sans fil. Afin de trouver le meilleur équilibre entre consommation d'énergie et délai d'attente, ils ont optimisé les paramètres de cette politique en utilisant des métaheuristiques, notamment l'OEP. Cette approche nous permet de comparer directement avec notre propre optimisation de la politique VTDS par l'OEP. Les résultats comparatifs des deux optimisations, en termes de consommation d'énergie, de délai d'attente et de temps d'exécution de la métaheuristique, sont présentés dans la table 3.25, offrant une perspective claire sur l'efficacité relative de chaque méthode.

TABLE 3.25 – N-policy vs VTDS avec l'OEP.

	EC	$\bar{W}$	TE
N-Policy	396.64	2.43	2h :33m :33s
VTDS	50,94	2,21	20m :52s

Les résultats montrent que la politique VTDS, optimisée par l'OEP, offre une réduction significative de la consommation d'énergie et du délai d'attente par rapport à la N-policy, avec un temps d'exécution de la métaheuristique nettement plus court. Cette comparaison souligne l'efficacité de notre approche dans l'optimisation des paramètres de configuration pour les RCSFs.

3.5 Conclusion

En conclusion, les analyses approfondies menées dans ce chapitre ont permis de mettre en lumière les performances remarquables des métaheuristiques RS, RT et OEP pour l'optimisation des paramètres de configuration de la VTDS.

L'OEP s'est distingué comme la métaheuristique la plus performante, offrant la meilleure solution en termes de minimisation de la fonction objectif (0.02978) avec un temps d'exécution raisonnable (3810 s).

La comparaison avec la politique N-policy existante a démontré la supériorité de la VTDS optimisée par OEP, en termes de réduction de la consommation d'énergie et du délai d'attente, tout en conservant un temps d'exécution de la métaheuristique nettement plus court.

Conclusion Générale

Le présent projet de fin d'études de master s'intègre dans le contexte de l'économie d'énergie dans les réseaux de capteurs sans fil. Nous nous sommes intéressés à la conservation de l'énergie consommée par les capteurs sans fil, ainsi qu'à l'optimisation de leurs performances en utilisant les politiques de vacance de travail. Le travail réalisé est de proposer une autre approche pour évaluer la politique VTDS proposée dans [4] pour économiser l'énergie et minimiser le délai. L'approche d'expérimentation de la politique VTDS utilisée dans [4] ne permet pas de trouver les valeurs des paramètres qui permettent d'avoir les valeurs optimales de l'énergie et du délai. S'ajoute a ça, l'outil utilisé pour l'évaluation des performance du modèle (*GreatSPN*) n'offre pas la possibilité d'automatiser les expérimentations.

Pour ce faire, nous avons commencé par présenter une revue de la littérature sur les concepts liés à notre travail, notamment les RCSFs et leurs défis, nous avons présenté quelques techniques d'économies d'énergies dans les RCSFs. Une approche essentielle pour optimiser la consommation d'énergie consiste à utiliser des techniques de modélisation formelles basées sur les files d'attente avec vacance qui offrent un potentiel considérable pour optimiser la consommation d'énergie. Toutefois, la difficulté réside dans la détermination des valeurs optimales des paramètres de configuration de ces techniques, afin d'équilibrer la consommation d'énergie et le délai d'attente.

Notre contribution s'est basée sur la politique de vacances de travail à deux seuils proposée par Boutoumi et al. dans [4]. Nous avons choisi cette politique en raison de son efficacité reconnue pour gérer l'énergie dans les réseaux de capteurs sans fil. Cependant, pour mieux comprendre et optimiser son comportement, nous avons représenté le modèle du noeud capteur en utilisant des chaînes de Markov à temps continu (CMTCs) afin de capturer le comportement dynamique et stochastique du réseau.Ensuite, nous avons adapté des métaheuristiques, telles que le Recuit Simulé (RS), la Recherche Tabou (RT) et l'Optimisation par Essaim de Particules (OEP) à notre problème d'optimisation pour déterminer les valeurs optimales des paramètres de configuration de la politique VTDS.

L'approche VTDS et les métaheuristiques ont été implémentées de manière indépendante, en utilisant le langage de programmation Python, ainsi que Matlab pour la résolution du système d'équations. Il convenait alors de varier les paramètres empiriques des algorithmes afin de trouver la solution optimale pour chacun d'entre eux.

Le fonctionnement et la stabilisation des métaheuristiques implémentées ont été validés à l'aide d'une méthode de référence proposée sur un espace de recherche restreint,

ce qui a permis de comparer leurs performances à une référence fiable. Par la suite, nous avons étendu notre expérimentation à un espace de recherche large pour évaluer leur performance dans des conditions plus variées et complexes. Les résultats obtenus ont démontré que l'OEP se distingue comme la métaheuristique la plus performante, offrant la meilleure optimisation de la fonction objectif avec un temps d'exécution raisonnable. La RT, tout en produisant des solutions de qualité louable, a souffert de délais d'exécution prolongés. Le RS, quant à lui, avec son exécution plus rapide, il a présenté une qualité de solution légèrement inférieure. Ces résultats mettent en lumière les points forts et les points faibles de chaque métaheuristique face à notre problème d'optimisation.

De plus, la comparaison avec la politique N-policy existante a mis en évidence la performance supérieure de la VTDS optimisée par OEP, qui se traduit par une réduction notable de la consommation d'énergie et du délai d'attente. De plus, la métaheuristique OEP se distingue par un temps d'exécution nettement plus court, soulignant son efficacité et sa praticité.

Enfin, en s'appuyant sur les résultats obtenus, notre étude ouvre la voie à plusieurs perspectives de recherche intéressantes pour l'optimisation des paramètres de configuration VTDS. Nous citons :

— Hybridation des métaheuristiques afin d'exploiter leurs forces respectives et d'obtenir des performances encore meilleures. Cette hybridation pourrait consister à utiliser une première métaheuristique pour explorer l'espace de recherche et une autre pour affiner la solution.
— Intégration de l'incertitude et de la dynamique du réseau, car les RCSFs sont soumis à des incertitudes (trafic variable) d'où, il serait intéressant d'adapter les métaheuristiques pour trouver des solutions adaptables aux changements du réseau.

Bibliographie

[1] Ian F Akyildiz et al. "A survey on sensor networks". In : *IEEE Communications magazine* 40.8 (2002), p. 102-114.

[2] Mohamed Lehsaini. "Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique". Thèse de doct. Besançon, 2009.

[3] Bachira Boutoumi et Nawel Gharbi. "An energy saving and latency delay efficiency scheme for wireless sensor networks based on GSPNs". In : *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE. 2017, p. 0645-0650.

[4] Bachira Boutoumi et Nawel Gharbi. "Two thresholds working vacation policy for improving energy consumption and latency in WSNs". In : *Queueing Theory and Network Applications : 13th International Conference, QTNA 2018, Tsukuba, Japan, July 25-27, 2018, Proceedings 13*. Springer. 2018, p. 168-181.

[5] Bachira Boutoumi et Nawel Gharbi. "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)*. IEEE. 2023, p. 1-6.

[6] Scott Kirkpatrick, C Daniel Gelatt Jr et Mario P Vecchi. "Optimization by simulated annealing". In : *science* 220.4598 (1983), p. 671-680.

[7] Fred Glover. "Future paths for integer programming and links to artificial intelligence". In : *Computers & operations research* 13.5 (1986), p. 533-549.

[8] James Kennedy et Russell Eberhart. "Particle swarm optimization". In : *Proceedings of ICNN'95-international conference on neural networks*. T. 4. ieee. 1995, p. 1942-1948.

[9] Touat Fatma et Touat Nawal. "Application de détection de présence par réseau de capteurs sans fil.* Simulation dans l'environnement TinyOS". Thèse de doct. Université Mouloud Mammeri, 2015.

[10] JM Hall et al. *Gamma-ray identification of nuclear weapon materials*. Rapp. tech. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 1997.

[11] Gunjan. "A Review on Multi-objective Optimization in Wireless Sensor Networks Using Nature Inspired Meta-heuristic Algorithms". In : *Neural Processing Letters* 55.3 (2023), p. 2587-2611.

[12] Marcos Augusto M Vieira et al. “Survey on wireless sensor network devices”. In : *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 03TH8696)*. T. 1. IEEE. 2003, p. 537-544.

[13] MOAD Sofiane. “Optimisation de la consommation d’énergie dans les réseaux de capteurs sans fil”. In : *Master recherche en 2éme année informatique Université : FSIC-Rennes* 1 (2008).

[14] Salheddine Kabou et A Belgourari. “Etat de l’art sur les réseaux de capteurs sans fil”. In : *Université de Bechar Algérie-Licence en informatique* (2010).

[15] Thien D Nguyen, Jamil Y Khan et Duy T Ngo. “Energy harvested roadside IEEE 802.15. 4 wireless sensor networks for IoT applications”. In : *Ad Hoc Networks* 56 (2017), p. 109-121.

[16] Tifenn Rault, Abdelmadjid Bouabdallah et Yacine Challal. “Energy efficiency in wireless sensor networks : A top-down survey”. In : *Computer networks* 67 (2014), p. 104-122.

[17] Ricardo C Carrano et al. “Survey and taxonomy of duty cycling mechanisms in wireless sensor networks”. In : *IEEE Communications Surveys & Tutorials* 16.1 (2013), p. 181-194.

[18] Sudip Misra, Manikonda Pavan Kumar et Mohammad S Obaidat. “Connectivity preserving localized coverage algorithm for area monitoring using wireless sensor networks”. In : *Computer communications* 34.12 (2011), p. 1484-1496.

[19] Rahim Kacimi. “Techniques de conservation d’énergie pour les réseaux de capteurs sans fil”. Thèse de doct. Institut National Polytechnique de Toulouse-INPT, 2009.

[20] Diery Ngom. “Optimisation de la durée de vie dans les réseaux de capteurs sans fil sous contraintes de couvertureet de connectivité réseau”. Thèse de doct. Université de Haute Alsace-Mulhouse ; Université Cheikh Anta Diop (Dakar), 2016.

[21] Micha Yadin et Pinhas Naor. “Queueing systems with a removable service station”. In : *Journal of the Operational Research Society* 14 (1963), p. 393-405.

[22] Christos H Papadimitriou et Kenneth Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 2013.

[23] Jin-Kao Hao, Philippe Galinier et Michel Habib. “Méthaheuristiques pour l’optimisation combinatoire et l’affectation sous contraintes”. In : *Revue d’intelligence artificielle* 13.2 (1999), p. 283-324.

[24] Yassine Meraihi. “Qualité de service dans les réseaux sans fil maillés/vanet”. Thèse de doct. Université M’Hamed Bougara : Faculté des hydrocarbures et de la chimie, 2016.

[25] Aroussi Sana. *Cours de Heuristiques et Méta-Heuristiques niveau M1*. 2023.

[26] KHERICI Nada. *Cours de Méthodes de Résolution en Optimisation Combinatoire niveau M1*. 2021.

[27] Xin-She Yang. “Chapter 5 - Simulated Annealing”. In : *Nature-Inspired Optimization Algorithms (Second Edition)*. Sous la dir. de Xin-She Yang. Second Edition. Academic Press, 2021, p. 83-90.

[28] Ahmed Salhi. “Contribution à l’optimisation de l’écoulement de puissance en utilisant la logique floue associée aux réseaux de neurones (Neuro-Flou)”. Thèse de doct. Thèse de doctorat, Université de Biskra, 2015.

[29] Yuhui Shi et Russell Eberhart. “A modified particle swarm optimizer”. In : *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE. 1998, p. 69-73.

[30] Ilhem Boussaid. “Perfectionnement de métaheuristiques pour l’optimisation continue”. Thèse de doct. Paris Est, 2013.

[31] James Kennedy et Rui Mendes. “Population structure and particle swarm performance”. In : *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600)*. T. 2. IEEE. 2002, p. 1671-1676.

[32] BOUKADER Younes et KERROUCHI Zakaria. *Méta-heuristiques pour l’Optimisation du seuil dans une Politique d’Economie d’Energie dans les Capteurs Sans-Fil*. 2023.