وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية الرياضيات والإعلام الآلي والاتصالات السلكية واللاسلكية

قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : Réseaux informatique et systèmes repartis

T h è m e

Optimisation de la QoS dans un Réseau De Radio cognitive en utilisant l'algorithme de la Colonies d'Abeilles

Réalisé par :
MERABET Amina
MOUSSAOUI asmaa

Dirigé par :
Dr . DERKAOUI Orkia

Membres de jury

Dr. BEN YAHYA MILOUD	Président
Dr. DERKAOUI Orkia	Rapporteur
Dr. BEN YAHYA KADA	Examinateur

Année Universitaire

2024 - 2025

Remerciements

الحمد لله رب العالمين و الصلاة و السلام على سيد المرسلين

Nous adressons nos plus sincères remerciements et notre profonde reconnaissance à :

- Notre encadrante, Madame Derkaoui Orkia, pour sa confiance, la pertinence du sujet qu'elle nous a proposé, ses conseils avisés, ainsi que pour son accompagnement précieux et ses remarques constructives tout au long de ce travail.
- L'ensemble de nos enseignants, pour leur engagement et leur rôle fondamental dans notre parcours académique.
- Nos familles et nos amis, pour leur soutien moral, leurs prières et leurs encouragements constants qui nous ont permis de surmonter les difficultés.
- Les membres du jury, pour l'honneur qu'ils nous font en acceptant de juger ce travail, et pour leurs remarques précieuses qui ne manqueront pas de l'enrichir.

Nous remercions également toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail modeste.

Liste des figures :

Liste des tableaux

Liste des acronymes

Acronyme	Définition
QoS	**Quality of Service** – Qualité de service, indicateur global de performance réseau
ABC	**Artificial Bee Colony** – Algorithme de la colonie d'abeilles artificielles
GA	**Genetic Algorithm** – Algorithme génétique
CRN	**Cognitive Radio Network** – Réseau de radio cognitif
CPU	**Central Processing Unit** – Processeur central
CSV	**Comma-Separated Values** – Format de fichier texte utilisé pour les données tabulaires
GUI	**Graphical User Interface** – Interface graphique utilisateur
Tkinter	Toolkit standard de Python pour le développement d'interfaces graphiques
Mbps	**Megabits per second** – Unité de mesure du débit
ms	**Millisecond** – Unité de mesure du temps (latence)
R	Fiabilité (Reliability) – Paramètre du modèle QoS
B	Débit (Bandwidth) – Quantité de données transmises
L	Latence – Temps de transmission
IDE	**Integrated Development Environment** – Environnement de développement intégré (ex. : NetBeans, Spyder)
RRC	**Réseau Radio Cognitif** – Terme francophone pour CRN

Tables de matières

Introduction générale

Introduction générale

Contexte et motivation

Avec l'explosion des technologies sans fil (4G, 5G, IoT...), la pression sur le spectre radioélectrique est devenue importante. Pourtant, plusieurs études ont montré que ce spectre est souvent sous-utilisé [1], ce qui a motivé le développement de la radio cognitive. Celle-ci permet aux utilisateurs secondaires d'exploiter les fréquences libres sans interférer avec les utilisateurs primaires [2].

Cependant, cette flexibilité pose le défi d'assurer une qualité de service (QoS) acceptable dans un environnement dynamique [3]. Pour relever ce défi, des approches basées sur des algorithmes bio-inspirés, comme celui de la colonie d'abeilles (ABC), sont apparues comme des solutions efficaces pour optimiser l'utilisation du spectre [4][5].

Ce projet vise donc à appliquer l'algorithme ABC afin d'améliorer la QoS dans un réseau de radio cognitive en optimisant l'allocation des ressources.

Problématique

Dans les réseaux de radio cognitive, l'accès dynamique au spectre offre une meilleure exploitation des fréquences disponibles. Cependant, cette flexibilité rend plus difficile la garantie d'une qualité de service (QoS) constante. En effet, la disponibilité des canaux change en permanence, ce qui peut provoquer des coupures, des retards ou une baisse de débit pour les utilisateurs secondaires.

Les approches classiques d'allocation du spectre ne répondent pas toujours efficacement à cette variabilité. C'est pourquoi il devient nécessaire d'explorer des techniques d'optimisation plus intelligentes et adaptatives.

La problématique que nous abordons dans ce travail est donc : Comment améliorer l'allocation du spectre dans un réseau de radio cognitive, en tenant compte des contraintes de QoS, à l'aide d'un algorithme d'optimisation inspiré du comportement naturel comme celui de la colonie d'abeilles (ABC) ?

Objectifs du mémoire

Principal objectif de ce mémoire est d'améliorer l'allocation du spectre dans un réseau de radio cognitive en utilisant l'algorithme de la colonie d'abeilles (ABC), tout en garantissant une bonne qualité de service (QoS) pour les utilisateurs secondaires.

Les objectifs spécifiques sont les suivants :

- Analyser les défis liés à la QoS dans ces réseaux.
- Étudier l'algorithme ABC et ses applications dans l'allocation du spectre.
- Développer une méthode pour optimiser l'utilisation du spectre.
- Tester et évaluer la solution en termes de QoS (comme le débit et la latence).
-

Méthodologie adoptée

La méthodologie adoptée pour ce mémoire repose sur plusieurs étapes clés :

- Étude de l'état de l'art : Nous commencerons par une revue approfondie de la littérature sur les réseaux de radio cognitive et l'utilisation de l'algorithme ABC dans ce domaine.
- Modélisation du problème : Ensuite, nous modéliserons le problème d'allocation du spectre comme un problème d'optimisation, adapté à l'algorithme ABC.
- Développement d'une solution : Nous implémenterons l'algorithme ABC pour optimiser l'allocation dynamique des ressources dans le réseau.
- Simulation et évaluation : Enfin, nous effectuerons des simulations pour tester la performance de la solution en termes de QoS (débit, latence, taux de perte).

Cette approche nous permettra de valider l'efficacité de l'algorithme ABC dans l'amélioration de la QoS dans un réseau de radio cognitive.

Organisation du mémoire

Ce mémoire est structuré de la manière suivante :

- Introduction généraleCette section introduit le sujet, le contexte, la problématique et les objectifs de la recherche. Elle présente également la méthodologie générale et la structure du mémoire.
- Chapitre 1 : Réseaux de Radio Cognitive (RRC)Ce chapitre présente les principes de base des réseaux de radio cognitive, leur architecture, leur fonctionnement et les principaux défis auxquels ils font face.
- Chapitre 2 : Qualité de service (QoS) dans les Réseaux RRCCe chapitre aborde le concept de la qualité de service (QoS), ses enjeux, ses métriques et les approches existantes pour l'améliorer dans les réseaux de radio cognitive.
- Chapitre 3 : Méta heuristiques et algorithme de la colonie d'abeilles(ABC)
  Ce chapitre décrit les méta heuristiques, en mettant l'accent sur l'algorithme de la colonie d'abeilles (ABC) et son potentiel d'optimisation pour les problèmes complexes.
- Chapitre 4 : Proposition d'un modèle d'optimisation de la QoS par l'algorithme ABC. Dans ce chapitre, un modèle d'optimisation de la QoS est proposé, en s'appuyant sur l'algorithme de la colonie d'abeilles pour améliorer les performances des réseaux de radio cognitive.
- Chapitre 5 : Simulation et résultats.Ce chapitre présente les simulations réalisées, les résultats obtenus, ainsi qu'une analyse critique de ces résultats par rapport aux objectifs fixés.
- Conclusion généraleCette dernière section récapitule les principaux résultats, met en lumière les contributions du mémoire et propose des pistes pour des travaux futurs.

ChapitreI : Réseaux de Radio Cognitive (RRC)

I.1 Définition et principes de la radio cognitive

I.1.1 Définition :

Une radio cognitive (RC) est une radio intelligente qui peut être programmé et configuré dynamiquement. Les meilleurs canaux sans fil présent dans son voisinage sont détectés par son émetteur-récepteur qui va les utiliser par la suite. Une telle radio détecte automatiquement les canaux disponibles dans le spectre sans fil, puis change en conséquence ses paramètres de transmission ou de réception pour permettre des communications sans fil plus simultanés dans une bande de fréquences donnée à un seul endroit. Ce processus est une forme de gestion dynamique du spectre.

I.1.2 Principes de la radio cognitive :

Les fonctions possibles de la radio cognitive incluent la capacité d'un émetteur-récepteur de déterminer sa situation géographique, d'identifier et d'autoriser son utilisateur, crypter ou décrypter des signaux, détecter les périphériques sans fil voisins en exploitation, et ajustant la puissance de sortie et les caractéristiques de modulation.

- ✓ **Utilisateurs primaires (PU):** connus sous le nom de « utilisateurs licenciés », ils disposent d'une licence qui leur permet d'opérer à n'importe quel moment sur des bandes spectrales qui leurs sont réservées.
- ✓ **Utilisateurs secondaires (SU):** ces utilisateurs n'ont pas de licence mais peuvent accéder à des bandes de fréquence non utilisées par les utilisateurs primaires à n'importe quel moment et les libère une fois le service terminé ou une fois qu'un utilisateur primaire en aura besoin. Les SU accèdent au spectre de façon opportuniste sans déranger les utilisateurs primaires.[6]

I.2 Architecture d'un réseau RRC

La radio cognitive est définie par un ensemble cohérent de règles de conception par lequel unensemble spécifique de composants réalise une série de fonctions de produits et de services,comme décrit dans la figure suivante :

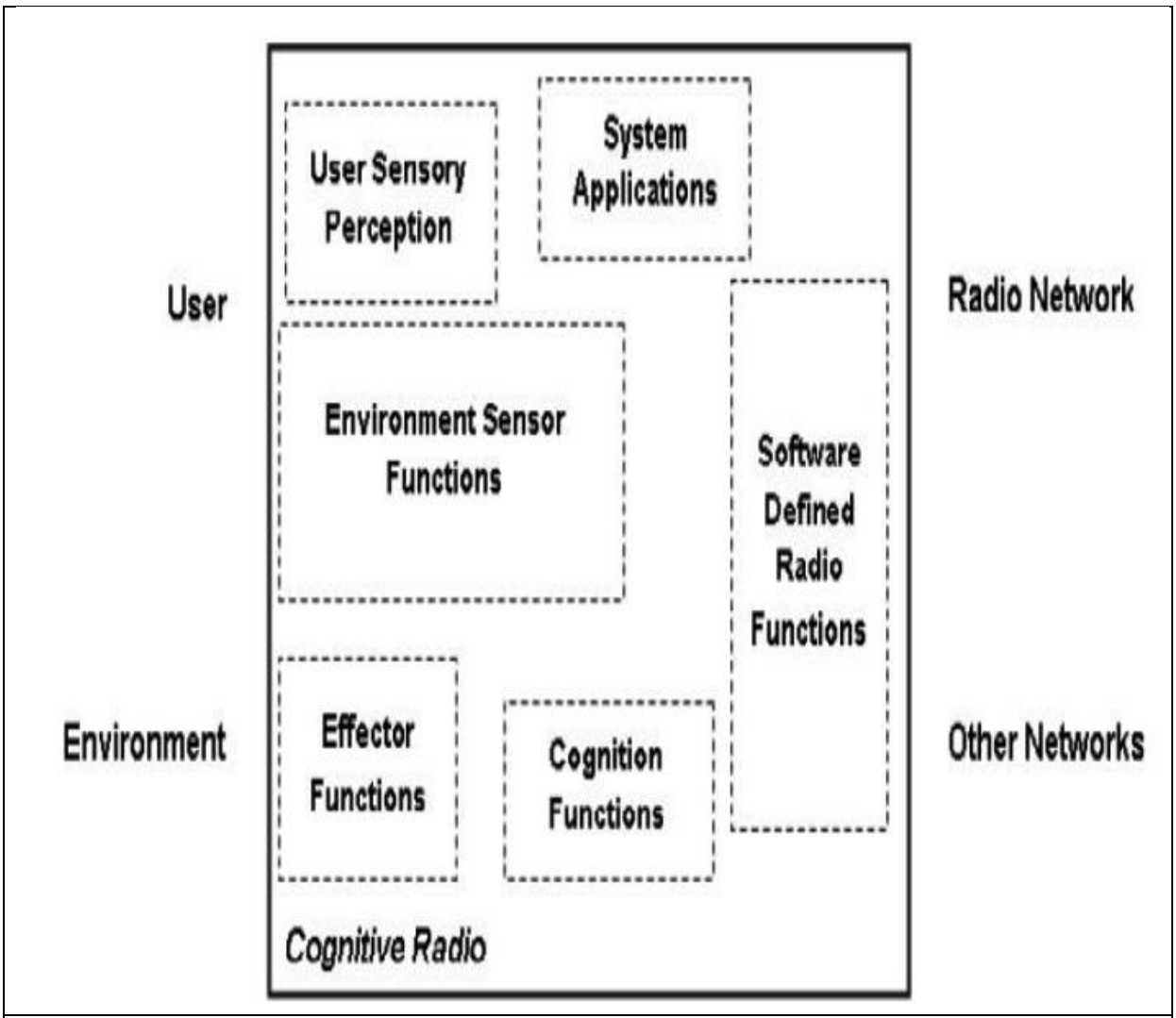


Figure I.1 : Architecture de la radio-cognitive

Les six composants fonctionnels de l'architecture d'une radio cognitive sont :

- **La perception sensorielle (Sensory Perception : SP)** : Elle inclut les interfaces haptiques (toucher), acoustiques, vidéo ainsi que les fonctions de détection et de perception.
- **Les capteurs de l'environnement local** : Cela comprend des éléments comme le positionnement, la température et d'autres facteurs environnementaux.

- **Les applications système** : Ce sont des services multimédias indépendants, tels que les jeux en réseau.
- **Les fonctions SDR** : Cela inclut la détection RF et les applications radio dans le cadre de la SDR (radio définie par logiciel).
- **Les fonctions de cognition** : Elles concernent les systèmes de contrôle, de planification et d'apprentissage, permettant à la radio cognitive de s'adapter et d'optimiser son fonctionnement.
- **Les fonctions locales effectrices** : Ce sont des fonctions telles que la synthèse vocale, la conversion de texte en parole, l'affichage graphique et multimédia.

Ces composantes travaillent ensemble pour permettre à la radio cognitive de s'adapter dynamiquement aux conditions environnementales et d'optimiser l'utilisation du spectre.

I.3 Gestion du spectre et spectre dynamique

I.3.1 Gestion du spectre

Le terme spectre radioélectrique se réfère généralement à la gamme de fréquences de 3 kHz à 300 GHz qui peut être utilisé pour la communication sans fil, comme il est montré dans la Figure I.1.

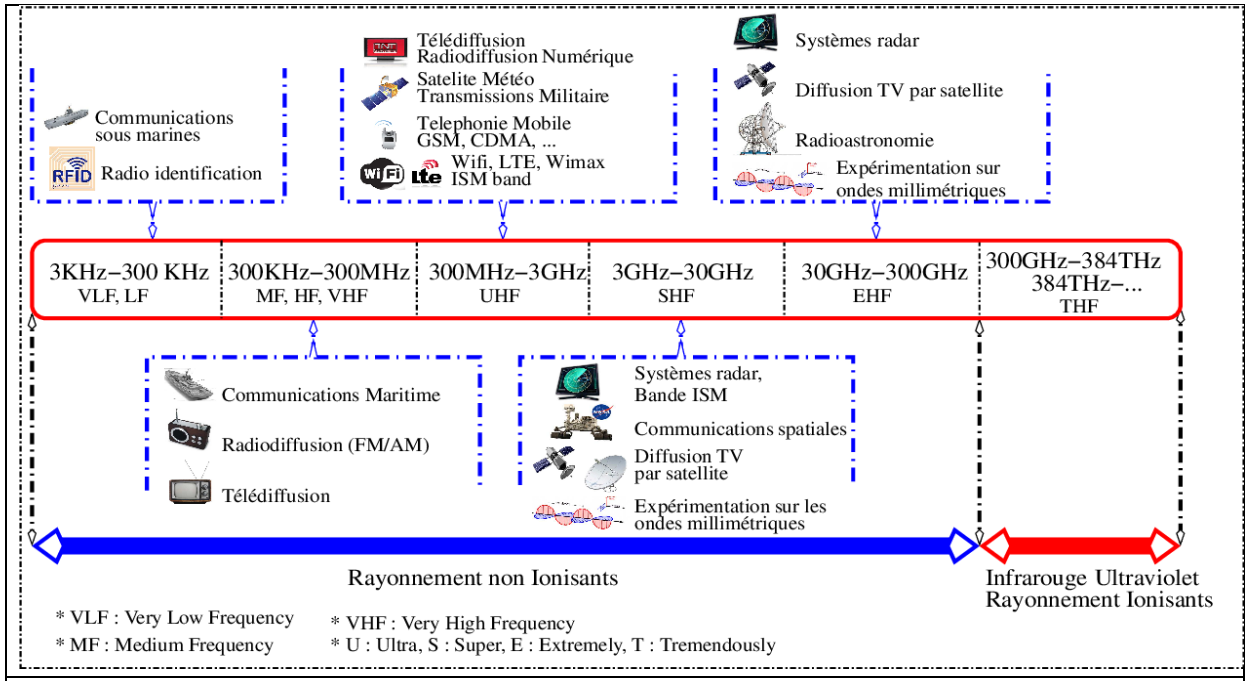


Figure I.2 : Répartition du spectre radio fréquence en fonction des technologies .(Suraweera, H. A. &Shihada, A. Y. M. (2017))

La demande croissante pour des services tels que les téléphones mobiles et bien d'autres a nécessité une révision profonde des stratégies de gestion du spectre. L'augmentation substantielle de la demande en haut débit sans fil, notamment en raison des innovations technologiques comme la 3G, la 4G, et l'expansion rapide des services Internet sans fil, a exacerbée cette pression. Depuis les années 1930, le spectre a été attribué par l'Union Internationale des Télécommunications (UIT)& de manière centralisée et autorisée administr-ativement.

Historiquement, l'interférence du signal était perçue comme un problème majeur affectant l'utilisation du spectre. Pour y remédier, le système de licences exclusives a été mis en place afin de protéger les signaux des titulaires de permis. Ce modèle de gestion consistant à attribuer des bandes discrètes sous licence à des groupes de services similaires cède progressivement la place, dans de nombreux pays, à un modèle plus flexible, celui de la « vente aux enchères de spectre ». Ce dernier a pour objectif de stimuler l'innovation technologique et d'améliorer l'efficacité de l'utilisation du spectre.

Au fil du temps, plusieurs autres approches ont été explorées lors des processus d'assignation des fréquences, notamment les loteries, l'accès non autorisé et la privatisation du spectre. Cependant, ces stratégies n'ont pas toujours permis de répondre adéquatement aux besoins croissants de spectre.

Plus récemment, le Conseil Présidentiel des Conseillers pour la Science et la Technologie (PCAST) a recommandé le partage dynamique du spectre lorsque celui-ci est inutilisé dans certains lieux ou à certains moments, à condition qu'il n'y ait pas de risques importants d'interférences. À la suite des recommandations du PCAST, le président Barack Obama a adopté la politique de « spectre partagé » le 14 juin 2013, marquant un tournant dans la gestion du spectre aux États-Unis. En décembre 2014, la FCC (Federal Communications Commission) a étendu cette approche à d'autres bandes, y compris la bande radar de la Marine des États-Unis (3550-3700 MHz), en appliquant avec succès un modèle de partage du spectre dans la bande de télévision (les espaces blancs de la télévision). Cette initiative a ouvert la voie à une gestion plus souple et efficiente du spectre radioélectrique, permettant d'améliorer son utilisation et d'augmenter la capacité disponible pour les utilisateurs.

I.3.2 Gestionspectre dynamique

La gestion dynamique du spectre (GDS) désigne l'ensemble des techniques permettant une utilisation plus efficace et flexible du spectre radioélectrique. Contrairement à la gestion traditionnelle, où les bandes de fréquences sont allouées statiquement aux utilisateurs titulaires de licences, la GDS vise à optimiser l'usage de ce spectre en permettant son partage dynamique entre utilisateurs primaires et secondaires.

Ce modèle repose sur les capacités d'observation, d'analyse, de décision et d'adaptation des réseaux de radio cognitive (RRC), qui peuvent identifier les « trous » spectraux (white spaces) laissés inoccupés par les utilisateurs titulaires, et les exploiter sans causer d'interférences.

❖ **Étapes de la gestion dynamique :**

- Détection du spectre : Identifier les bandes libres en temps réel.
- Analyse et prise de décision : Évaluer les conditions du canal pour choisir la meilleure option.
- Partage du spectre : Gérer l'accès concurrent des utilisateurs secondaires.
- Mobilité spectrale : Changer dynamiquement de bande en cas de reprise par un utilisateur primaire.

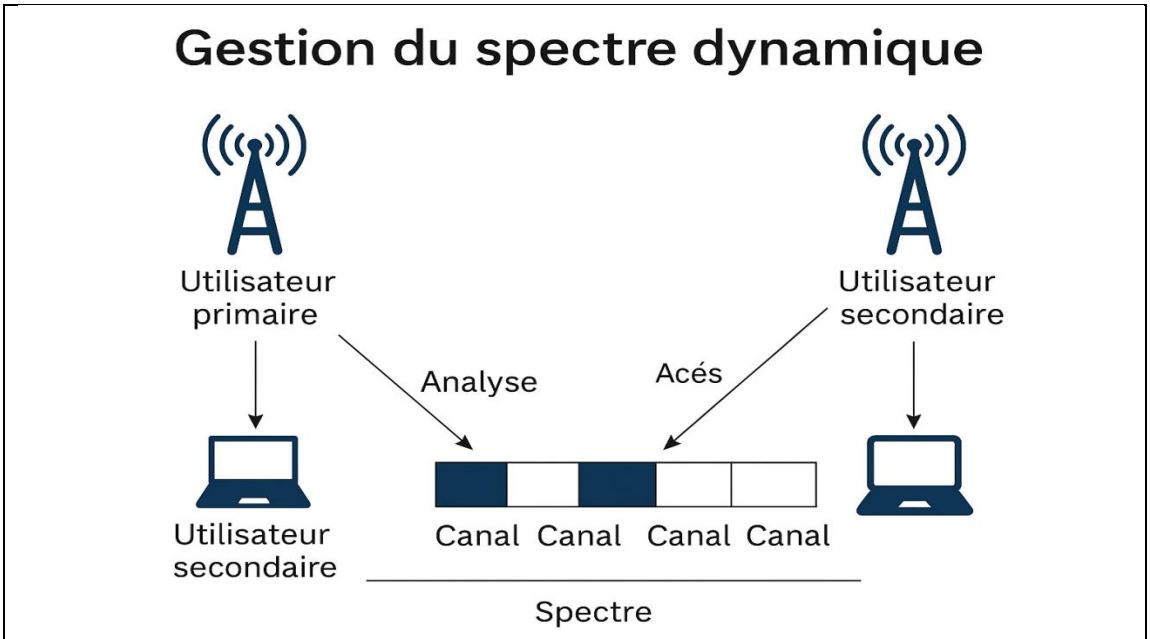


Figure I.3 : gestion du spectre dynamique(Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., & Mohanty, S. (2006).

I.4 Défis liés à la Qos dans les RRC

La fourniture de Qualité de Service (QoS) dans les réseaux de radios cognitives (CR) est complexe en raison de la nature dynamique du spectre radio et de l'interaction avec les utilisateurs primaires (PUs). Les principaux défis de la fourniture de QoS dans ces réseaux sont les suivants :

1. **Identification des Ressources :**
   - Le spectre dans les réseaux de radios cognitives n'est pas alloué de manière fixe et dédiée, et les dispositifs doivent trouver des "trous" dans le spectre qui ne sont pas utilisés par les PUs à un moment donné. Cela rend la disponibilité dynamique du spectre et peut entraîner des coupures lorsque ces trous ne sont pas disponibles .
2. **Caractéristiques du Canal Radio :**
   - Des interférences telles que l'atténuation, le bruit, la propagation multi-trajet et l'ombrage affectent la qualité du signal. De plus, lors du changement de canal en raison de l'activité des PUs, les caractéristiques du canal peuvent changer, compliquant le maintien de la QoS
3. **Durée de la Détection (Sensing) :**
   - Pendant la phase de détection du spectre, les CR arrêtent la transmission de données, ce qui affecte la QoS. Le défi ici est de trouver un équilibre entre une durée de détection suffisante et un impact minimal sur la transmission des données .
4. **Gestion du Handoff (Changement de Canal) :**
   - Lorsqu'un PU occupe un canal utilisé par un CR, ce dernier doit passer à un autre canal, ce qui entraîne des délais en raison de la nécessité de détecter le spectre et de changer de canal. Ce changement de canal peut entraîner des variations dans les caractéristiques du canal et affecter la QoS .
5. **Taux de Handoff :**
   - Les changements fréquents de canal dus à la mobilité du spectre entraînent des interruptions et des retards, ce qui dégrade la QoS. Par conséquent, réduire le taux de changement de canal est essentiel pour maintenir une connexion stable .
6. **Gestion des Interférences :**
   - Les interférences peuvent survenir en raison de la détection manquée d'un PU, due à un faible rapport signal/bruit ou à un

problème de terminal caché, ce qui entraîne une perte de paquets et des retards. Cela affecte négativement la QoS des CRs et des PUs .[7]

I.5 Applications des RRC

- **Espaces blancs de la télévision (TV White Spaces)**
  Utilisation des bandes de fréquences inutilisées allouées à la télévision pour les communications secondaires.
- **Réseautage opportuniste**
  Exploitation dynamique des trous spectraux pour établir des connexions temporaires selon la disponibilité du spectre.
- **Applications pour les situations d'urgence et la sécurité publique**
  Communications critiques dans les scénarios de catastrophe, recherche et sauvetage, ou interventions de sécurité.
- **Communications véhiculaires (Vehicular Communications)**
  Intégration de la radio cognitive dans les véhicules pour améliorer la connectivité, la sécurité routière, et la gestion du trafic.
- **Réseaux RC contrôlés ou assistés par les opérateurs**
  Intégration de la radio cognitive dans les réseaux opérateurs pour améliorer la gestion des ressources spectrales.
- **Nouvelles applications émergentes**
  Tout autre usage innovant exploitant la capacité cognitive des RRC pour adapter dynamiquement les transmissions selon l'environnement.[8]

Chapitre II :
Qualité de la (QoS)
Dans les Réseau
RRC

II.1 Définition de la QoS (Qualité de Service)

La **Qualité de Service (QoS)** est un concept fondamental dans les systèmes de communication modernes. Elle désigne l'ensemble des mécanismes permettant de garantir un certain niveau de performance à des flux de données, dans des conditions de trafic variables.

Elle vise à assurer un service adapté aux besoins spécifiques des applications (voix, vidéo, données critiques), en optimisant l'utilisation des ressources réseau tout en maintenant la stabilité, la fiabilité et la rapidité de la communication. .[14]

En d'autres termes :

La QoS cherche à répondre à la question suivante :

Comment prioriser et distribuer efficacement les ressources réseau pour garantir un service fluide et fiable à tous les utilisateurs, même en cas de congestion ou de conditions changeantes ?

Elle prend toute son importance dans les environnements :

- multi-utilisateurs, où les ressources sont partagées,
- temps réel, où la latence est critique,
- mobiles, comme les réseaux cellulaires ou les réseaux de radio cognitive.

II.2 Paramètres de la QoS

La qualité perçue d'un service réseau repose sur plusieurs **indicateurs techniques**. Ces paramètres permettent de quantifier les performances et de prendre des décisions de gestion intelligentes :

- **Latence (ou délai de transmission)**
  Temps nécessaire pour qu'un paquet de données parvienne de l'émetteur au récepteur.
  Exemple : Dans un appel vidéo, une latence supérieure à 150 ms devient perceptible par l'utilisateur.
- **Débit (Throughput ou bande passante)**
  Quantité de données pouvant être transmises par unité de temps.
  Exemple : Le streaming 4K nécessite un débit d'au moins 25 Mbps.

- **Gigue (Jitter)**
  Variation des délais de livraison des paquets.
  Exemple : Dans une conversation VoIP, un jitter supérieur à 30 ms peut entraîner une dégradation audible.
- **Taux de perte de paquets (Packet Loss)**
  Pourcentage de paquets qui n'arrivent pas à destination.
  Exemple : Un taux de perte > 1% dans une visioconférence entraîne des coupures d'image ou de son.
- **Disponibilité**
  Pourcentage du temps pendant lequel le service est disponible sans interruption.
  Exemple : Les systèmes critiques (hôpitaux, aviation) exigent une disponibilité de 99.999% (soit 5 minutes d'interruption par an).

Ces indicateurs sont interdépendants : **une mauvaise latence peut accentuer la perte de paquets ou augmenter la gigue**, compromettant ainsi l'ensemble de l'expérience utilisateur.

II.3 Contraintes spécifiques à la QoS dans les RRC (Réseaux de Radio Cognitive)

Les **Réseaux de Radio Cognitive (RRC)** introduisent une approche innovante de gestion dynamique du spectre. Cependant, cette flexibilité crée des défis uniques pour le maintien de la QoS.

➢ Variabilité du spectre radio

Les utilisateurs secondaires (SU) n'ont pas un accès garanti. Lorsqu'un utilisateur primaire (PU) revient, ils doivent immédiatement libérer la bande, provoquant :

- des coupures de communication,
- une latence accrue,
- un besoin constant de reconfiguration.

➢ Détection du spectre (Spectrum Sensing)

Le réseau doit surveiller en permanence les bandes disponibles sans interférer avec les PU. Cela nécessite :

- des capacités de calcul supplémentaires,
- une grande précision,
- un compromis entre **temps de détection** et **vitesse de réaction**.

➢ **Handover fréquent**

Les changements rapides de canaux imposent des procédures complexes de transfert de session, appelées handovers. Cela impacte :

- la continuité des sessions en temps réel,
- les performances des applications sensibles à la latence.

➢ **Interférences radioélectriques**

Le partage du spectre accroît le **risque de collisions** entre flux, surtout si les décisions d'accès ne sont pas optimisées. Cela entraîne :

- une dégradation du signal,
- une augmentation du taux de retransmission,
- une baisse générale du débit.

Conclusion : La QoS dans les RRC nécessite des **algorithmes d'adaptation en temps réel**, capables de s'ajuster rapidement à un environnement radio instable.

II.4 Techniques d'optimisation de la QoS

Afin de répondre aux exigences croissantes en termes de performance et de fiabilité, plusieurs techniques ont été développées :

✓ **DiffServ (Differentiated Services)**

Permet de traiter les flux selon leur classe de service (voix, vidéo, données, best effort).
Chaque paquet est marqué et acheminé selon une politique prioritaire. **Avantage :** Scalabilité.

Limite : Pas de garantie stricte de QoS.

- ✓ **IntServ (Integrated Services)**

Fonctionne avec RSVP (Resource Reservation Protocol) pour réserver explicitement des ressources réseau à chaque flux.

Avantage : Garanties de QoS fortes.

Limite : Non scalable pour les grands réseaux.

- ✓ **Contrôle de congestion**

Techniques telles que :

- TCP congestion control,
- RED (Random Early Detection),
- gestion adaptative des buffers.

Elles préviennent l'effondrement du réseau en limitant le volume des transmissions pendant les pics de charge.

- ✓ **Planification des files d'attente (Queue Management)**

Les paquets sont organisés selon des politiques comme :

- FIFO (First In First Out),
- WFQ (Weighted Fair Queuing),
- Priority Queuing,

ce qui permet une gestion fine du trafic.

- ✓ **Allocation dynamique des ressources**

Utilisation d'algorithmes intelligents (comme ABC, GA, PSO) pour :

- sélectionner les canaux les moins encombrés,
- adapter la puissance de transmission,
- équilibrer la charge réseau.

II.5 État de l'art : Méthodes d'optimisation

Les méthodes traditionnelles atteignent leurs limites dans les environnements dynamiques, mobiles et à forte densité.

Les chercheurs se tournent désormais vers l'intelligence artificielle (IA) pour des solutions plus flexibles et prédictives.

- **Apprentissage automatique (Machine Learning - ML)**

Utilisé pour :

- prédire les congestions,
- optimiser la sélection des canaux,
- répartir dynamiquement la bande passante,
- améliorer la performance sur les couches PHY, MAC et Réseau.

Algorithmes utilisés :

- SVM (Support Vector Machine),
- Random Forest,
- KNN (K-Nearest Neighbors),
- Q-learning.

- **Apprentissage profond (Deep Learning - DL)**

Les réseaux neuronaux profonds (CNN, RNN, LSTM) sont capables d'identifier des schémas complexes :

- pour l'allocation en temps réel des ressources radio,
- pour l'anticipation des perturbations,
- pour la priorisation dynamique des flux.

Avantages de l'IA dans la QoS :

- Adaptabilité automatique à l'environnement réseau.
- Prise de décision en temps réel.
- Réduction de la dépendance à l'intervention humaine.
- Capacité à apprendre et à s'améliorer au fil du temps.

Ces approches sont prometteuses dans le contexte des réseaux 5G/6G, des véhicules connectés, et de l'Internet des objets (IoT).

Chapitre III :
Méta heuristiques
Et Algorithme de la Colonie d'abeilles

III.1 Introduction aux métaheuristiques

Les métaheuristiques sont des algorithmes d'optimisation conçus pour résoudre des problèmes complexes, souvent rencontrés en recherche opérationnelle, en ingénierie ou en intelligence artificielle. Ces problèmes, difficiles à traiter par des méthodes exactes en raison de leur complexité, nécessitent des approches flexibles capables de fournir de bonnes solutions en un temps raisonnable.

Généralement stochastiques et itératifs, les métaheuristiques explorent intelligemment l'espace de recherche sans exiger une connaissance approfondie de sa structure. Elles combinent souvent deux stratégies : l'intensification, qui affine les meilleures solutions connues, et la diversification, qui permet d'explorer de nouvelles régions pour éviter les optima locaux.

Parmi ces approches, les algorithmes bio-inspirés occupent une place importante. L'algorithme de la colonie d'abeilles artificielle (ABC), par exemple, s'inspire du comportement des abeilles dans la recherche de nourriture. Grâce à l'équilibre entre exploitation et exploration, il s'adapte efficacement à des contextes tels que la gestion des ressources dans les réseaux de radio cognitive, où il permet d'améliorer la qualité de service (QoS) tout en optimisant l'utilisation du spectre radioélectrique [12].

III.2 Comparaison des principales métaheuristiques (GA, PSO, ACO, etc.)

Critère	GA	PSO	ACO	BCO	SA
Principe De base	Evolution biologique	Comportement collectif	Comportement fourmis	Comportement Des abeilles	Recuit Thermique simulé
Type de Recherche	Populationnel	Populationnel	Populationnel	Populationnel	individuel
Exploration globale	Boone	Moyenne	Boone	Boone	Très Boone
Exploration Locale	Moyenne	Boone	Boone	Boone	Moyenne
Convergence	Moyenne A lente	Rapide	Moyenne A bonne	Bonne Stable	Lente
Facilité D'implément ation	Moyenne	Facile	Moyenne à complexe	Moyenne	Facile
Paramètres Clés	Croisement, Mutation , Taille Population	Inertie, Cognition sociale	Phéromone, heuristique	Nombre d'abeilles, scouts	Température, Taux de refroidissent
Adaptabilité Aux CRN	Bonne	Très Boone	Très Boone	Très Boone	Moyenne
Résistance aux Minima locaux	Bonne	Faible à moyenne	Bonne	Bonne	Bonne
Application typiques	Routage, QoS, planification	Optimisation continue	Routage, QoS, Gestion De taches	QoS, Gestion du spectre	Problème combinatoires

Tableau III. 1 : Comparaison des principales métaheuristiques

III.3 Principe de l'algorithme de la colonie d'abeilles (ABC)

Dans l'algorithme Artificial Bee Colony (ABC), les abeilles artificielles sont réparties en trois catégories : les abeilles employées, qui explorent les sources de nourriture connues ; les abeilles observatrices, qui sélectionnent les sources à explorer en fonction de leur qualité ; et les abeilles scoutes, qui recherchent de nouvelles sources de manière aléatoire (Mezura et Dami´an-Araoz,2013)

Chaque source de nourriture représente une solution potentielle au problème d'optimisation, et une abeille employée est associée à chaque source. Si une solution ne s'améliore pas après un certain nombre d'itérations, elle est abandonnée et l'abeille concernée devient une scoute. Ce mécanisme est contrôlé par un paramètre appelé « limite » (Karaboga et Basturk ,2007).

Les abeilles employées partagent les informations avec les observatrices dans la ruche. Ces dernières sélectionnent les sources à explorer en fonction de leur quantité de nectar, c'est-à-dire la qualité de la solution. Plus cette qualité est élevée, plus la probabilité qu'une source soit choisie augmente.

III.3.1 Production initiale des sources de nourriture.

Si l'espace de recherche est considéré comme l'environnement de la ruche qui contient les sources alimentaires, l'algorithme commence à produire de manière aléatoire des sources alimentaires qui correspondent aux solutions de l'espace de recherche. Sources alimentaires initiales sont produites au hasard dans l'intervalle des limites des paramètres (Akay et Karaboga,2010).

$$\boldsymbol{X_{ij} = X_j^{min} + \emptyset_j(X_j^{max} - X_j^{min})}$$

Où : i = 1... SN, j = 1... D. SN est le nombre de sources d'alimentation et D représente le nombre de paramètres d'optimisation. En outre, un compteur qui stocke le nombre des essais de solutions sont remis à 0 dans cette phase. Après l'initialisation, les sources initiales de nourriture (solutions) sont soumises à des cycles répétitifs d'amélioration, les abeilles employées et les abeilles spectatrices explorent le voisinage et améliorent les solutions. Le critère d'arrêt de l'algorithme ABC pourrait être satisfait avec l'atteinte d'un nombre maximum de cycle ou répondre à une tolérance d'erreur (Akay et Karaboga,2010)..

III.3.2 Envoi des abeilles employées aux sites

Une abeille employée produit une modification de la position de la source d'alimentation (solution) dans sa mémoire en fonction de l'information locale (information visuelle) et trouve une source d'alimentation voisine, puis évalue sa qualité. Dans ABC, trouver une source de nourriture voisine est définie par (Akay et Karaboga,2010) :

$$\upsilon_{i,j} = X_{i,j} + \phi_{i,j}(X_{i,j} - X_{k,j}) \quad \text{(formule de voisinage)}$$

k : indice attribué aléatoirement de [1, SN], k différent de i

ϕ : un facteur aléatoire entre [-1,1]

Comme on peut le voir à partir de l'équation (III.4), avec la diminution de la différence entre les paramètres $x_{i,j}$ et $x_{k,j}$, la perturbation de la position $x_{i,j}$ diminue. Ainsi, on se rapproche de la solution dans l'espace de recherche, la longueur de pas est réduite de manière adaptative (Akay et Karaboga,2010). Si une valeur de paramètre produite dépasse ses limites prédéterminées, le paramètre peut être fixé à ses limites. $s_i x_i > x_i$ max alors $x_i = x_i^{max}$, Si $x_i < x_i^{min}$ alors Xi = x_i^{min} .

Une sélection est appliquée entre x_i et u_i le meilleur est choisi en fonction de la valeur de quantité de nectar des sources de nourriture de x_i et u_i. Si la source au υi est supérieure acelle de x_i en termes de rentabilité, l'abeille employée mémorise la nouvelle position et oublie l'ancien. Sinon, la position précédente est conservée. $s_i x_i$ne peut pas être améliorée, le nombre d'essais est incrémenté de 1, sinon, le compteur est remis à 0. Si le nombre d'essais atteint une valeur prédéterminée x_i est abandonnée (Akay et Karaboga,2010).

L'évaluation est calculée par la formule :

$$\text{Fitnessi} = 1 / f_i \qquad \text{(II.3)}$$

III.3.3 Calcul de probabilité

Après la phase des abeilles employées, ces derniers partagent leurs informations relatives aux quantités de nectar et les positions de leurs sources avec les abeilles spectatrices sur la piste de danse. C'est la caractéristique de l'interaction

multiple des abeilles artificielles de ABC modifié. Une abeille spectatrice évalue les informations de nectar pris de toutes les abeilles employées et choisit un site de source de nourriture avec une probabilité liée à sa quantité de nectar. Cette sélection est basée sur la valeur de la fitness (Akay et Karaboga,2010). La probabilité est donnée par :

$$P_i = \begin{cases} 0.5 + \left(\frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}\right) \times 0.5 \; si \; la \; solution \; est \; possible \\ \left(1 - \frac{violation_i}{\sum_{j=1}^{SN} violation_j}\right) \times 0.5 \; si \; la \; solution \; est \; impossible \end{cases}$$

Dans ce schéma de sélection probabiliste, comme la quantité de nectar des sources de nourriture augmente, le nombre de spectatrices venant les visiter augmente aussi. C'est la fonction de rétroaction positive de ABC modifié.

III.3.4 Choix de la source à améliorer par les spectatrices

Dans l'algorithme ABC, un nombre réel aléatoire dans l'intervalle [0,1] est généré pour chaque source. Si la valeur de probabilité (p_i) dans l'équation (II.4) associé à cette source est supérieure à ce nombre aléatoire, alors l'abeille spectatrice produit une modification de la position de cette source de nourriture à l'aide de l'équation. (II.2). Après que la source est évalué, l'abeille spectatrice, soit elle mémorise la nouvelle position et oublie l'ancienne, soit elle garde l'ancienne. Si la solution x_i ne peut pas être améliorée, le compteur d'essais augmente de 1, sinon, le compteur est remis à 0. Ce processus est répété jusqu'à ce que toutes les spectatrices soient réparties sur les sources alimentaires (Akay et Karaboga,2010).

III.3.5 Critère pour abandonner une source :

Limite et production de scoute Dans un cycle complet, après les phases des abeilles employés et des abeilles spectatrices, l'algorithme vérifie pour voir si une source est épuisée pour l'abandonner. Afin de décider si une source est abandonnée, les compteurs qui ont été mis à jour lors de la recherche sont utilisés. Si la valeur du compteur est supérieure au paramètre de commande de l'algorithme ABC, appelée "Limite ", alors la source associée à ce compteur est supposée épuisée et doit être abandonné. La source abandonnée par son abeille est remplacée par une nouvelle source de nourriture découvert par la

scoute, ce qui représente le mécanisme de rétroaction négative et la propriété de fluctuation dans l'auto-organisation de ABC. L'action abandonner la source et charger une scoute de trouver une nouvelle source est simulée par le remplacement de la solution abandonnée par une solution produite d'une façon aléatoire. Cette opération peut être définie par (II.1). Dans l'ABC de base, on suppose que seule source puisse être épuisée dans chaque cycle, et une seule abeille peut être une scoute. Si plus d'un compteur dépasse la valeur "limite ", l'un des maxima pourrait être choisi par programme (Akay et Karaboga,2010).

III.4 Avantages et limites de ABC

III.4.1 Avantages de ABC

- Optimisation globale : ABC explore l'espace de recherche de manière parallèle, augmentant les chances de trouver une solution optimale.
- Flexibilité : Peut être appliquée à différents types de problèmes, qu'ils soient continus ou discrets.
- Robustesse : Grâce à l'interaction entre les "abeilles", ABC évite de se coincer dans des solutions locales.
- Simplicité d'implémentation : Plus facile à mettre en œuvre par rapport à d'autres algorithmes complexes.
- Parallélisme : Permet de traiter plusieurs solutions en même temps, ce qui améliore l'efficacité**.**

III.4.2 Limites de ABC

- Convergence lente : Parfois, ABC nécessite plusieurs itérations pour atteindre une solution optimale.
- Dépendance aux paramètres : Les mauvais choix de paramètres peuvent nuire à la performance de l'algorithme.
- Complexité pour de grands problèmes : Moins efficace pour les problèmes de grande taille.
- Convergence prématurée : L'algorithme peut s'arrêter trop tôt, avant d'explorer toutes les solutions possibles

III.5 Applications de ABC à l'optimisation réseau

- Optimisation de la qualité de service (QoS) : Dans les réseaux sans fil pour ajuster la bande passante et la latence.
- Gestion dynamique du spectre : Améliorer l'allocation du spectre et réduire les interférences dans les réseaux de radio cognitive.
- Optimisation des protocoles de routage : Réduire la congestion et améliorer les performances des réseaux.
- Optimisation des réseaux de capteurs sans fil (WSN) : Améliorer la couverture et la gestion de l'énergie.
- Optimisation des ressources dans le cloud : Réduire les coûts et améliorer l'efficacité.
- Réseaux 5G : Gérer l'allocation des ressources radio pour de meilleures performances.
- Réseaux ad hoc : Améliorer la couverture et réduire la latence.

Chapitre IV : Proposition d'un Modèle d'Optimisation QoS par BCO

IV.1 Modélisation du problème

Le problème consiste à optimiser la qualité de service (QoS) dans un réseau de radio cognitive.

Pour cela, nous avons modélisé chaque solution candidate sous la forme d'un vecteur de configuration = $[x_1, x_2, x_3, x_4, x_5]$ où chaque $x_i \in [0,1]$représente un paramètre de réglage du système.

La QoS est quantifiée par un score défini comme :

$$\text{QoS\_score} = \left(\frac{\text{débit}}{\text{latence}}\right) \cdot \text{fiabilité} \cdot (1 - \text{interférence}) \cdot \text{utilisation du spectre}$$

où :

- **B(x)est la bande passante influencée par x_1,**
- **L(x) la latence influencée par x_2,**
- **R(x) la fiabilité liée à x_5,**
- et d'autres paramètres tels que l'interférence et l'utilisation du spectre sont modélisés via $x_3$et x_4

L'objectif est de maximiser ce score, ce qui revient à maximiser la bande passante et la fiabilité tout en minimisant la latence et l'interférence.

Ce modèle permet de représenter le problème d'optimisation sous forme d'un espace de recherche continu sur $[0,1]^5$, où chaque point correspond à une configuration possible. Cette formulation mathématique nous permet d'utiliser l'algorithme bio-inspiré des abeilles pour explorer efficacement cet espace et trouver une solution optimale ou quasi-optimale.

IV.2 Formulation mathématique de la QoS

IV.2.1 Variablesdedécision

Soit

$x_{ij} \in \{0,1\}$: variable binaire indiquant si la bande de fréquence j est allouée à l'utilisateur i.

$\boldsymbol{R_i}$: débit de transmission de i sur la bande j (w).

$\boldsymbol{p_{ij}}$:Puissance de transmission de i sur la bande j (w).

$\boldsymbol{I_j}$:interférence sur la bande j due aux utilisateurs secondaires .

Qosi:Niveau de Qos de l'utilisateur i(exprimé en fonction du débit ,de la latence ,etc.).

IV.2.2 Fonction objectif

L'objectif est de maximiser la somme pondérée des QOS des utilisateurs ,

Tout en minimisant l'interférence et l'utilisation excessive du spectre .

$$\max_{x_{ij},P_{ij}} \sum_{i=1}^{N} w_i \cdot \mathbf{QoS}_i - \lambda \sum_{j=1}^{M} I_j$$

Avec

- w_i :poids de l'utilisateur i (priorité de service).
- λ : facteur de pénalité pour l'interférence.
- N :Nombre d'utilisateurs secondaires.
- M :Nombre de bandes de fréquence.

IV.2.3 Contrainte de débit minimum pour garantir la QoS

Chaque utilisateur doit obtenir un débit minimum garanti R_{min}

$$R_i = \sum_{j=1}^{M} x_{ij} B_j \log_2 \left(1 + \frac{P_{ij} g_{ij}}{N_0 + I_j}\right) \geq R_{\min}, \quad \forall i$$

Avec:

- B_j:largeur de bande de la fréquence j.
- g_{ij}:Gain du canal entre l'utilisateur i et la bande j.
- N_0:densité spectre de la bruit blanc.

IV.2.4 Contrainte de puissance maximale

La puissance de transsmission totale d'un utilisateur ne doit pas dépasser un seuil p_{max}

$$\sum_{j=1}^{M} P_{ij} \leq P_{\max}, \quad \forall i$$

IV.2.5 Contrainte d'interférence

L'interférence sur une bande ne doit pas dépasser un seuil I_{max}

$$I_j = \sum_{i=1}^{N} x_{ij} P_{ij} g_{ij} \leq I_{\max}, \quad \forall j$$

IV.2.6 contrainte d'allocation des bandes

Chaque bande j ne peut attribuée qu'a un seul utilisateur a la fois :

$$\sum_{i=1}^{N} x_{ij} \leq 1, \quad \forall j$$

IV.2.7 contrainte de latence maximale

La latence totale L_i pour utilisateur doit etre inférieure à un seuil L_{max} :

$$L_i = D_{g_i} + D_{t_i} + D_{r_i} + D_{q_i} + D_{p_i} \leq L_{\max}, \quad \forall i$$

IV.3 Description de l'algorithme ABC adopté

IV.3.1 Correspondance entre les éléments de l'algorithme et le problème QoS (RRC)

Algorithme ABC	Problème QoS / Réseaux radio cognitifs
Abeilles (clients/agents)	Appareils ou utilisateurs cherchant la meilleure configuration du réseau
Sources de nourriture (solutions)	Différentes configurations possibles du réseau (canaux, fréquences, protocoles…)
Qualité de la solution (fitness)	Indicateurs de performance QoS (débit, latence, taux de perte de paquets…)

Danse (partage d'informations)	Partage des résultats QoS entre clients ou avec le centre de gestion
Abeilles éclaireuses (scouts)	Agents testant de nouvelles configurations aléatoires pour éviter les minima locaux

TableauIV.1 : Correspondance entre les éléments de l'algorithme et le problème QoS

IV.3.2 Étapes détaillées de l'algorithme

- **Initialisation :**

  Chaque agent choisit aléatoirement une configuration réseau initiale.

- **Recherche locale (petites modifications) :**

  Les agents modifient légèrement leur configuration (changer de canal, ajuster protocole…).

- **Évaluation QoS (fitness) :**

  Après modification, on mesure la performance (ex : débit, latence, stabilité).

- **Partage d'informations (coopération) :**

  Les agents partagent leurs résultats pour orienter les autres vers des solutions prometteuses.

- **Abandon des mauvaises configurations :**

  Si une configuration ne s'améliore pas après plusieurs essais, elle est remplacée par une nouvelle configuration aléatoire.

- **Itération continue :**

  Ce processus se répète jusqu'à obtenir des configurations optimales qui maximisent la QoS.

IV.3.3 Exemple pratique (QoS dans un réseau radio cognitif)

- **Problème :** Trouver la meilleure combinaison canal/fréquence/pouvoir d'émission pour minimiser la latence et maximiser le débit.
- **Agents (abeilles)** : Appareils connectés au réseau.
- **Sources de nourriture** : Ensemble des configurations possibles.
- **Fitness :**Résultats de tests comme le RTT (Round Trip Time) ou le débit en Mbps. **:**
- **Itérations :** Modifications et échanges répétés pour améliorer continuellement la QoS.

IV.4 Intégration dans un réseau RRC

L'algorithme d'optimisation par colonie d'abeilles (ABC) s'intègre efficacement dans l'architecture des réseaux de radio cognitive (RRC) dans le but d'améliorer la Qualité de Service (QoS) offerte aux utilisateurs secondaires (Secondary Users - SU).

Dans un réseau RRC, les utilisateurs secondaires exploitent de manière opportuniste les bandes de fréquence temporairement inutilisées par les utilisateurs primaires (Primary Users - PU), tout en évitant toute interférence nuisible. Pour ce faire, le processus cognitif du réseau repose sur trois étapes principales : la détection du spectre, la prise de décision, et la réallocation dynamique des ressources.

C'est dans la phase de prise de décision que le modèle ABC intervient. Il est utilisé pour :

- Évaluer différentes configurations d'allocation de canaux et de puissance,
- Optimiser plusieurs critères de QoS (débit, délai, perte de paquets, énergie),
- Sélectionner la meilleure solution en respectant les contraintes du réseau.

Ainsi, chaque abeille dans l'algorithme représente une solution possible d'allocation des ressources pour les SUs. À travers une exploration collaborative de l'espace de recherche, le ABC identifie la configuration optimale ou quasi-optimale.

L'intégration du modèle ABC dans un RRC permet donc une gestion dynamique, intelligente et efficace du spectre, en assurant une QoS adaptée aux besoins des applications modernes (voix, vidéo, données), tout en préservant les droits des utilisateurs primaires.

IV.5 Schéma global de l'approche proposé

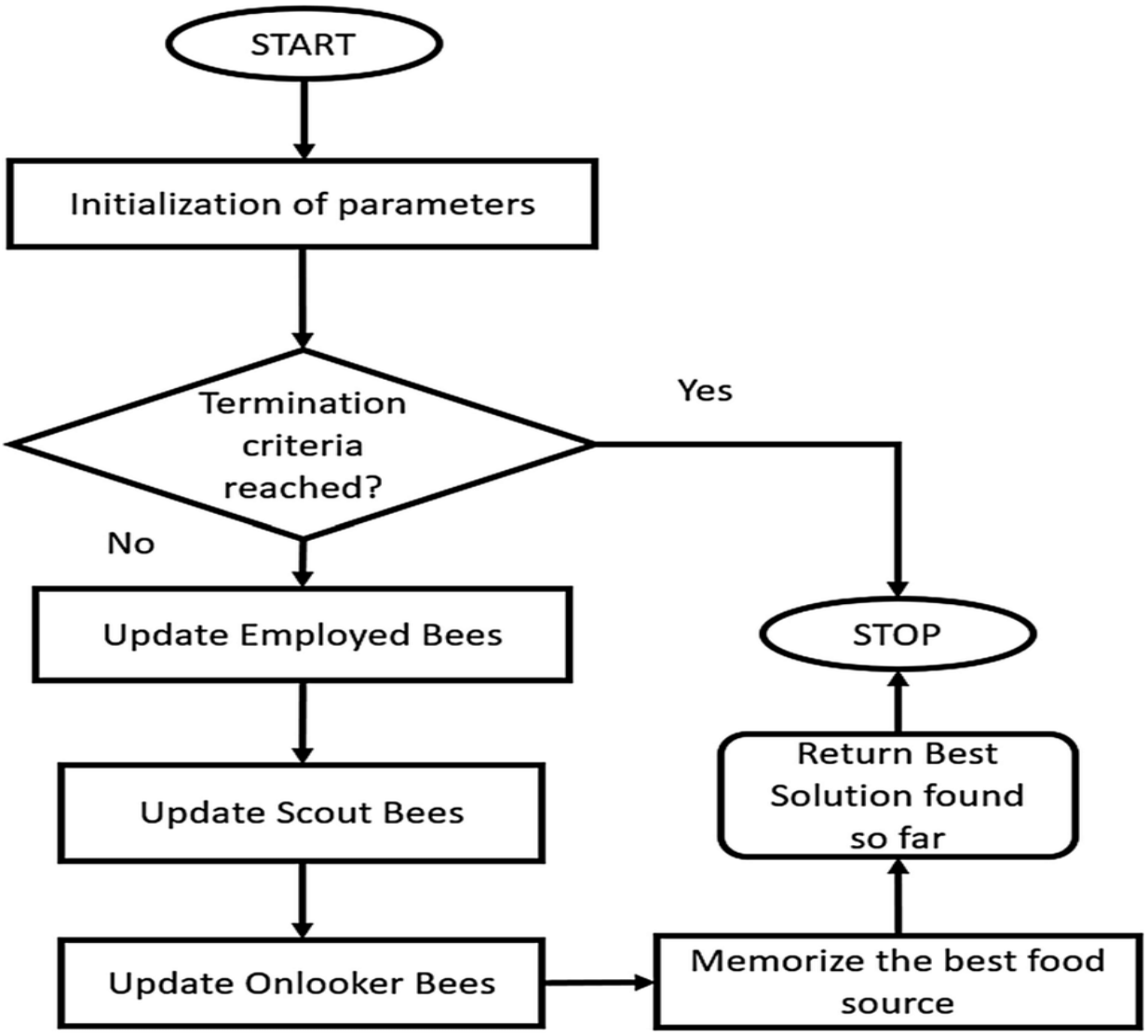


Figure IV.1 : Schéma global de l'approche (Geoffrey & Shankar,2020)

Chapitre V : Simulation et Résultats

V.1 Environnement de simulation

V.1.1 Cadre général de l'implémentation

L'environnement de simulation utilisé pour ce projet repose sur une plateforme Python personnalisée, développée afin de modéliser et comparer deux algorithmes d'optimisation — à savoir l'**algorithme de la colonie d'abeilles (ABC)** et l'**algorithme génétique (GA)** — dans le contexte spécifique d'un **réseau de radio cognitive**. Le code implémente une interface utilisateur graphique (GUI) avec la bibliothèque **Tkinter**, permettant de faciliter l'interaction avec l'utilisateur et d'automatiser la gestion des paramètres expérimentaux via des fichiers CSV. Le but de cette plateforme est de simuler la qualité de service (QoS) atteinte sous différentes configurations réseau et de comparer les performances respectives des deux méthodes d'optimisation.

L'ensemble du code est conçu de manière modulaire. Il est organisé autour de trois blocs fonctionnels majeurs : le **module d'optimisation**, le **module de visualisation graphique**, et le **module d'interaction utilisateur**. Cette structure assure à la fois **l'indépendance des composants** (facilitant la maintenance) et **la cohérence fonctionnelle globale** (via le partage d'objets et de variables comme le DataFrame global).

La simulation repose également sur une **modélisation interne de la QoS** à partir d'une combinaison de cinq paramètres d'entrée (x1 à x5) et d'une formule heuristique qui prend en compte le débit, la latence, l'interférence, la fiabilité et l'utilisation du spectre. Cette formule, bien que simplifiée, est suffisamment expressive pour représenter les enjeux essentiels du réseau radio cognitif.

V.1.2 Environnement logiciel et bibliothèques utilisées

Le projet est développé entièrement en **Python 3**, un choix motivé par la richesse de ses bibliothèques scientifiques, sa portabilité, et sa compatibilité avec les interfaces utilisateur de bureau. L'environnement de développement utilisé est **Spyder**, qui offre une interface conviviale adaptée à l'expérimentation scientifique et au prototypage rapide.

Plusieurs bibliothèques ont été utilisées dans le code pour assurer les différentes fonctionnalités du système :

- numpy : utilisée pour la génération de vecteurs de configuration aléatoires, les opérations de mutation (via des lois normales), et le calcul des moyennes et écarts-types.
- pandas : utilisée pour le traitement des fichiers CSV et la manipulation structurée des données expérimentales.
- matplotlib.pyplot : utilisée pour la génération de graphes (histogrammes, nuages de points, courbes temporelles) à travers des fenêtres intégrées à l'interface Tkinter.
- tkinter : bibliothèque standard de Python utilisée pour créer l'interface graphique. Elle est ici exploitée pour le chargement des fichiers, la saisie des paramètres, l'affichage des résultats et le déclenchement des fonctions de visualisation.
- tkinter.filedialog, tkinter.simpledialog, tkinter.messagebox : modules auxiliaires de Tkinter utilisés pour la gestion des interactions utilisateur.
- matplotlib.backends.backend_tkagg.FigureCanvasTkAgg : utilisé pour intégrer dynamiquement des graphiques matplotlib dans des fenêtres Tkinter.

Le code ne dépend pas d'un framework externe ou d'une architecture client-serveur. Il s'exécute localement sur machine personnelle, ce qui rend les expériences reproductibles sans besoin de configuration serveur.

V.1.3 Architecture fonctionnelle du simulateur

L'architecture du simulateur peut être décrite comme une **application événementielle graphique**, contrôlée par des actions utilisateur (chargement de données, déclenchement d'algorithmes, visualisation de résultats), mais dotée d'un **noyau algorithmique puissant** intégrant deux méthodes d'optimisation évolutive. Le programme est structuré selon une architecture semi-orientée objet, comme en témoigne l'utilisation de la classe Bee, qui encapsule l'état d'une abeille sous forme de vecteur de configuration et de score de QoS.

La logique centrale du programme repose sur deux fonctions principales : optimize_qos() pour l'algorithme ABC, et genetic_algorithm() pour l'algorithme génétique. Ces deux fonctions ont une structure interne similaire, bien que leurs principes de sélection et de reproduction diffèrent. Elles partagent toutes deux une même fonction d'évaluation, evaluate_qos(), qui calcule le score de qualité de service à partir des paramètres d'entrée.

Cette fonction evaluate_qos() joue un rôle fondamental dans la simulation. Elle agit comme **métamodèle de l'environnement radio**, en simulant le comportement d'un utilisateur secondaire accédant au spectre sous des contraintes dynamiques. Elle intègre cinq variables continues comprises entre 0 et 1, qui représentent des dimensions abstraites mais significatives du réseau :

- **x1** : lié au débit.
- **x2** : lié à la latence.
- **x3** : contrôle le niveau d'interférence.
- **x4** : reflète l'utilisation effective du spectre.
- **x5** : traduit une mesure de fiabilité.

La formule du score QoS est une combinaison non linéaire de ces composantes, pondérées et liées par des relations multiplicatives. Par exemple, la latence est intégrée via une expression inverse (débit / latence), ce qui reflète bien l'impact négatif d'un temps de réponse élevé sur la QoS. L'utilisation d'un score produit final (et non une somme) rend l'évaluation plus stricte : une seule dimension trop faible peut dégrader radicalement le score final.

Du point de vue algorithmique, la boucle principale de chaque algorithme évolutif consiste à générer une population, évaluer ses membres, trier par score, puis générer une nouvelle génération via perturbation locale (mutation dans ABC, croisement-mutation dans GA). Ces itérations sont répétées selon un nombre d'itérations fixé à l'avance, souvent entre 50 et 200 dans les fichiers d'entrée fournis.

Enfin, une couche supplémentaire du système permet le **chargement automatisé des paramètres expérimentaux** via un fichier CSV, ce qui facilite la reproductibilité et l'étude comparative sur plusieurs jeux de données. Cette fonction (load_data()) lit les colonnes nb_abeilles et nb_iterations et injecte leurs valeurs dans les champs de l'interface.

V.2 Paramètres expérimentaux

Dans cette section, nous décrivons en détail les paramètres expérimentaux qui ont été utilisés pour conduire les simulations d'optimisation de la Qualité de Service (QoS) dans un réseau de radio cognitive. Tous les paramètres présentés ici sont strictement basés sur les données manipulées dans le programme Python fourni, sans extrapolation externe.

V.2.1 Mode de configuration des paramètres

L'application permet à l'utilisateur de configurer deux paramètres essentiels à la simulation, à savoir :

- Le **nombre d'abeilles** (nb_abeilles) : il définit la taille de la population dans l'algorithme de la colonie d'abeilles. Il s'agit du nombre d'agents explorant simultanément l'espace de configuration possible pour la QoS.
- Le **nombre d'itérations** (nb_iterations) : il s'agit du nombre de cycles ou de tours de recherche que l'algorithme doit effectuer pour affiner les configurations et tenter d'atteindre un score QoS optimal.

Ces deux paramètres peuvent être introduits de deux manières différentes dans l'interface :

1. **Saisie manuelle** dans des champs entry Tkinter (boîtes de texte), permettant à l'utilisateur de les modifier à la volée.
2. **Chargement automatique** via un fichier CSV contenant deux colonnes nommées nb_abeilles et nb_iterations, en une seule ligne. Le fichier est lu à l'aide de la fonction load_data(), laquelle extrait les valeurs, les convertit en entiers, et les injecte automatiquement dans les champs d'entrée de l'interface graphique.

Ce système garantit une certaine flexibilité, tout en restant limité à une expérimentation à deux variables.

V.2.2 Portée des paramètres dans le programme

Le nombre d'abeilles et le nombre d'itérations impactent directement le comportement des fonctions d'optimisation du code :

- Dans optimize_qos(num_bees, iterations), ces deux paramètres déterminent la taille de la population d'abeilles (objets de la classe Bee) et le nombre de cycles de reproduction/exploration.
- Dans genetic_algorithm(num_population, iterations), ils jouent un rôle similaire pour l'algorithme génétique, influençant la taille de la population initiale et le nombre de générations.

Il est important de noter que ces deux fonctions sont indépendantes dans leur exécution, bien qu'elles utilisent les mêmes paramètres en entrée. Cela permet une comparaison directe, équitable et reproductible entre les deux approches.

V.2.3 Encodage des configurations des agents

Chaque abeille (ou individu dans le cas de l'algorithme génétique) est représentée par un vecteur de configuration de 5 dimensions généré aléatoirement avec des valeurs flottantes entre 0 et 1 :

python

CopierModifier

configuration = np.random.rand(5)

Ces cinq dimensions représentent les composantes internes d'une solution candidate, utilisées dans la fonction evaluate_qos() pour estimer la performance QoS simulée. Leur signification exacte n'est pas explicite dans l'interface, mais la fonction d'évaluation transforme chaque coordonnée selon la logique suivante:

- $x_1 \rightarrow$ Débit (de 10 à 30 Mbps)
- $x_2 \rightarrow$ Latence (de 100 à 20 ms)
- $x_3 \rightarrow$ Interférence (de 1 à 0)
- $x_4 \rightarrow$ Utilisation du spectre (0 à 1)
- $x_5 \rightarrow$ Fiabilité (de 0.5 à 1)

Ce modèle d'encodage abstrait permet d'évaluer l'impact de ces composantes de manière indirecte, sans devoir en nommer explicitement les valeurs dans l'interface.

V.2.4 Structure du fichier CSV de configuration

Le fichier CSV attendu par l'application doit respecter un format strict. Une seule ligne est attendue, avec deux colonnes : nb_abeilles et nb_iterations.

Exemple minimal de contenu CSV accepté :

nb_abeilles,nb_iterations50,100

Toute absence d'une de ces colonnes, ou la présence de types non numériques dans leurs cellules, génère automatiquement un message d'erreur via message box, showerror. Aucun traitement de lignes multiples ni de moyennes statistiques n'est prévu ; l'objectif du fichier CSV est uniquement de pré-remplir les champs de l'interface pour une exécution manuelle ultérieure.

V.2.5 Robustesse et validation des paramètres

Afin de prévenir les erreurs de saisie ou de format, le programme implémente plusieurs couches de validation :

- Contrôle de l'existence des colonnes attendues.
- Vérification de la convertibilité des valeurs en entiers.
- Blocage de l'exécution si une anomalie est détectée, avec un message d'erreur descriptif affiché dans une fenêtre contextuelle.

Cette robustesse garantit une exécution stable même en cas d'erreur utilisateur, mais limite les possibilités d'exploration automatique ou par lot.

V.2.6 Interface graphique – description écran par écran

L'interface graphique du simulateur a été développée à l'aide du module **Tkinter** de Python, en combinant des champs de saisie, des boutons d'actions, et des fenêtres secondaires pour permettre à l'utilisateur d'exécuter toutes les étapes de la simulation : configuration, optimisation, comparaison, visualisation graphique et analyse statistique. Cette section détaille **chaque écran généré** par le programme, tel qu'il est implémenté dans le code source.

V.2.7 Écran principal de l'application

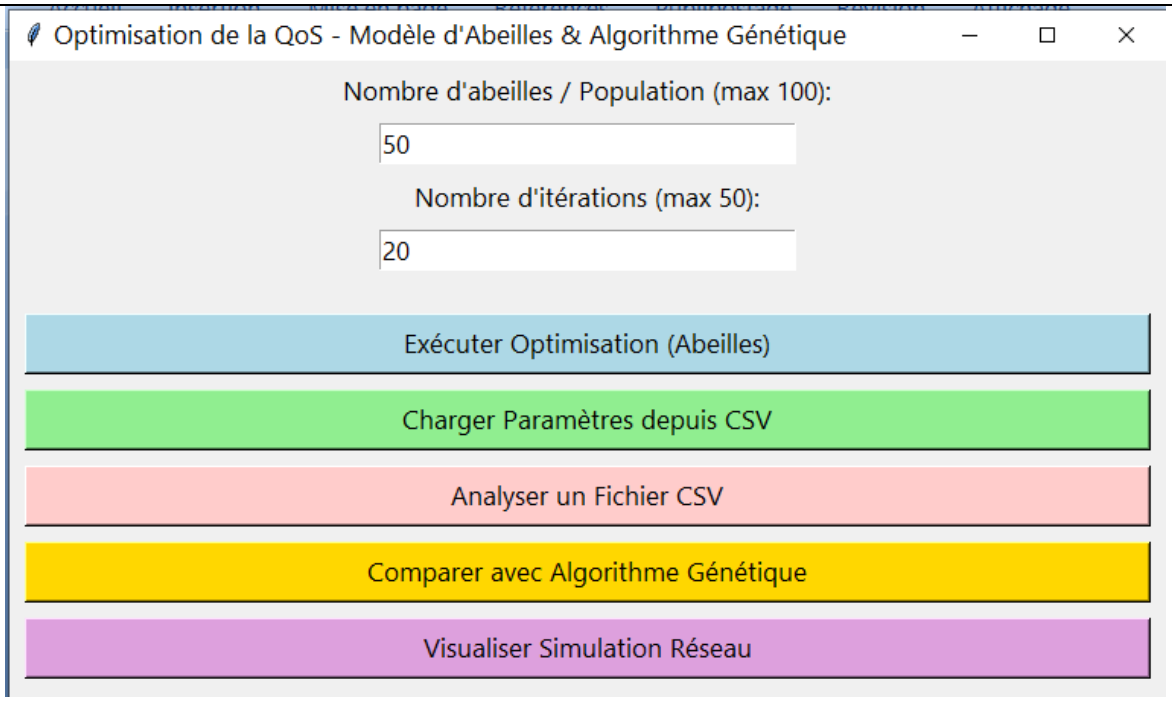


Figure V.1 : Ecran d'accueil

Après lancement du programme, une **fenêtre principale** est générée via l'objet Tk() instancié dans la fonction main_interface(). Cette fenêtre constitue le **point de contrôle central** à partir duquel toutes les actions sont déclenchées.

L'écran principal est organisé verticalement comme suit :

1. **Champs de saisie de paramètres expérimentaux** :
   - Nombre d'abeilles (max 100) : ce champ (entry_bees) permet à l'utilisateur de spécifier la taille de la population de l'algorithme ABC. Sa valeur est ensuite utilisée dans la fonction run_optimization().
   - Nombre d'itérations (max 50) : ce champ (entry_iterations) définit le nombre maximal d'itérations pour l'algorithme.

   Ces deux champs sont construits à l'aide de widgets Label et Entry, et sont placés en haut de l'écran dans le main_frame.

2. **Boutons fonctionnels** :

Juste en dessous des champs de saisie, cinq boutons sont alignés pour déclencher les différentes fonctionnalités du simulateur. Chaque bouton est associé à une fonction Python précise :

- **Bouton 1 : Exécuter Optimisation (Abeilles)**

→ Button(main_frame, text="Exécuter Optimisation (Abeilles)", command=run_optimization)
Cette commande lit les valeurs entrées dans entry_bees et entry_iterations, exécute l'algorithme ABC via run_optimization(), et affiche le meilleur score obtenu dans une boîte de dialogue (popup messagebox).

Au cas où un nombre spécifié dépasse le seuil des valeurs autorisé une boite de dialogue de message erreur est généré comme le montre la figure suivante :

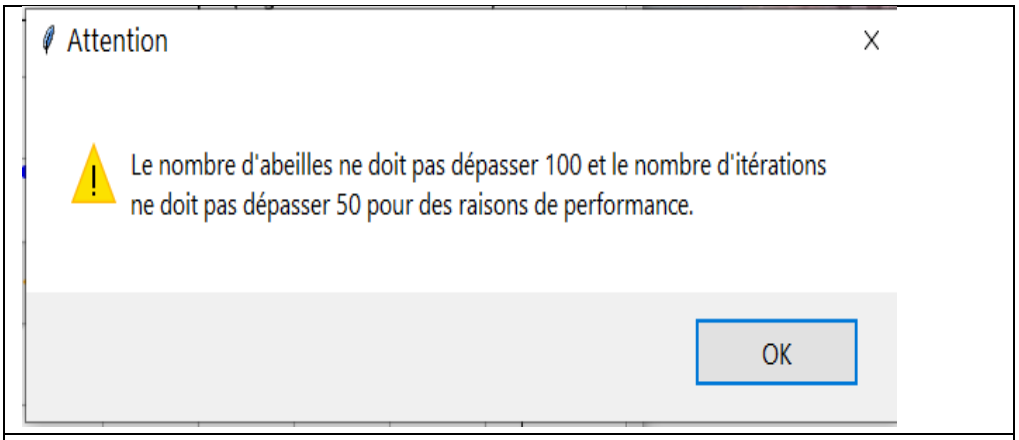


Figure V.2 : Message d'erreur en cas de dépassement de seuil

Si pas d'erreur par rapport au nombre d'abeilles et d'itérations un graph est généré comme le montre la figure suivante :

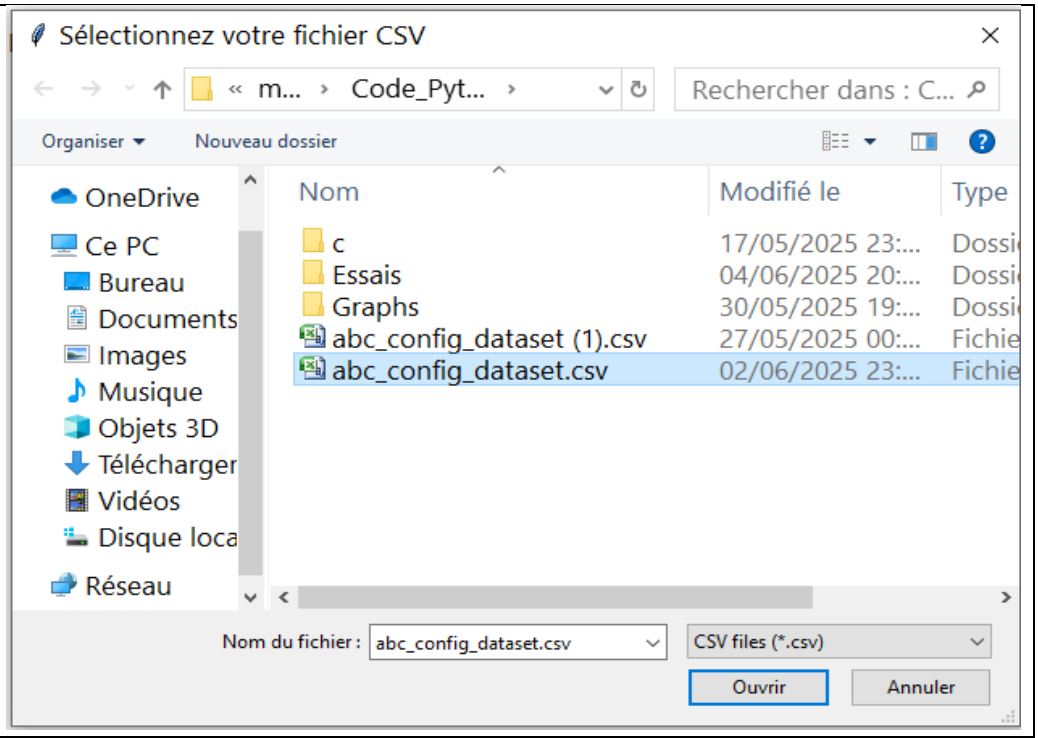


Figure V.3 : Sélection d'un fichier CSV

- **Bouton 2 : Charger Paramètres depuis CSV**

→ Button(main_frame, text="Charger Paramètres depuis CSV", command=load_data)

Ce bouton ouvre un dialogue de sélection de fichier (filedialog.askopenfilename) et lit un fichier CSV contenant deux colonnes : nb_abeilles et nb_iterations. Chaque ligne du fichier correspond à un scénario d'optimisation exécuté automatiquement. Les résultats sont ensuite affichés dans un **tableau textuel formaté** dans une nouvelle fenêtre Toplevel.

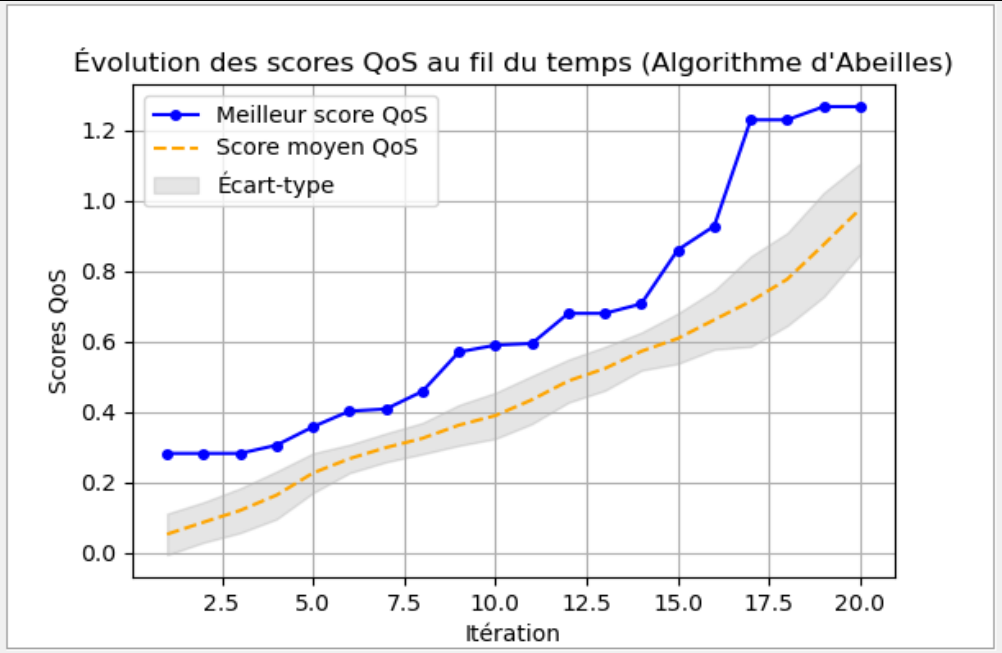


Figure V.3 : Evolution des scores QoS

Avec une boite de dialogue résumant les résultats (Figure 5-4) :

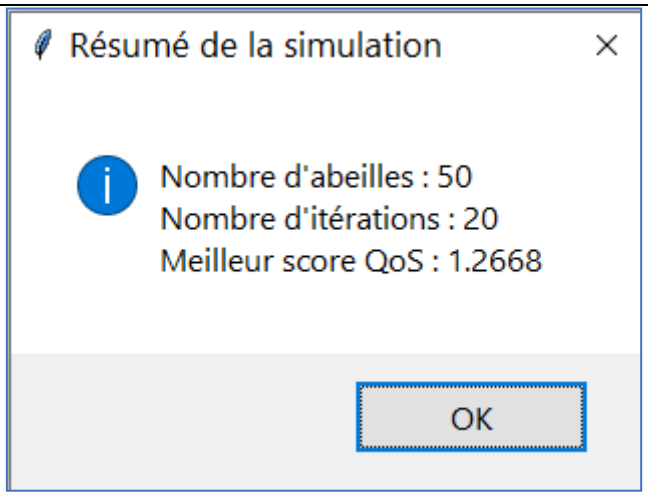


Figure V.4 : Message box Résumé de la simulation

- **Bouton 3 : Analyser Données Graphiquement**

  → Button(main_frame, text="Analyser Données Graphiquement", command=afficher_graphique_depuis_csv)
  Cette action affiche une nouvelle fenêtre contenant deux menus déroulants pour choisir les colonnes X et Y à tracer. L'utilisateur

sélectionne un fichier CSV (via filedialog) contenant des résultats, puis visualise un **graphe matplotlib** dans une interface intégrée.

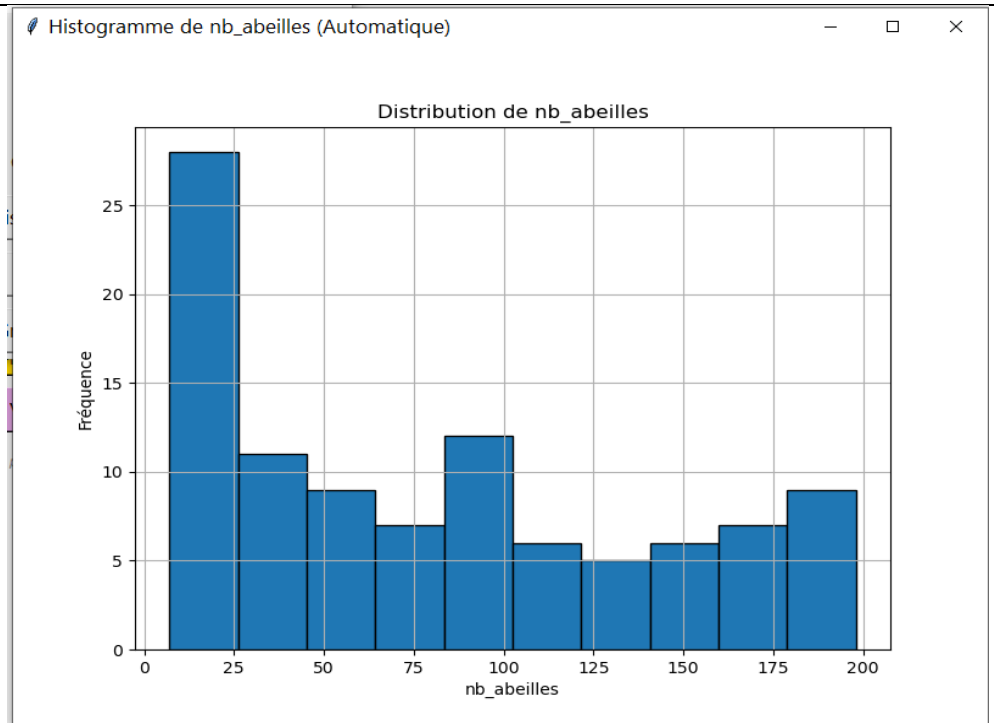


Figure V.5 : Histogramme de nombre d'abeilles

- **Bouton 4 : Comparer avec Algorithme Génétique**

→ Button(main_frame, text="Comparer avec Algorithme Génétique", command=comparer_algorithmes)
Ce bouton déclenche l'exécution du second algorithme (GA) et compare visuellement les performances ABC vs GA à travers un graphique simple (plt.plot) représentant les scores optimaux atteints par chaque méthode.

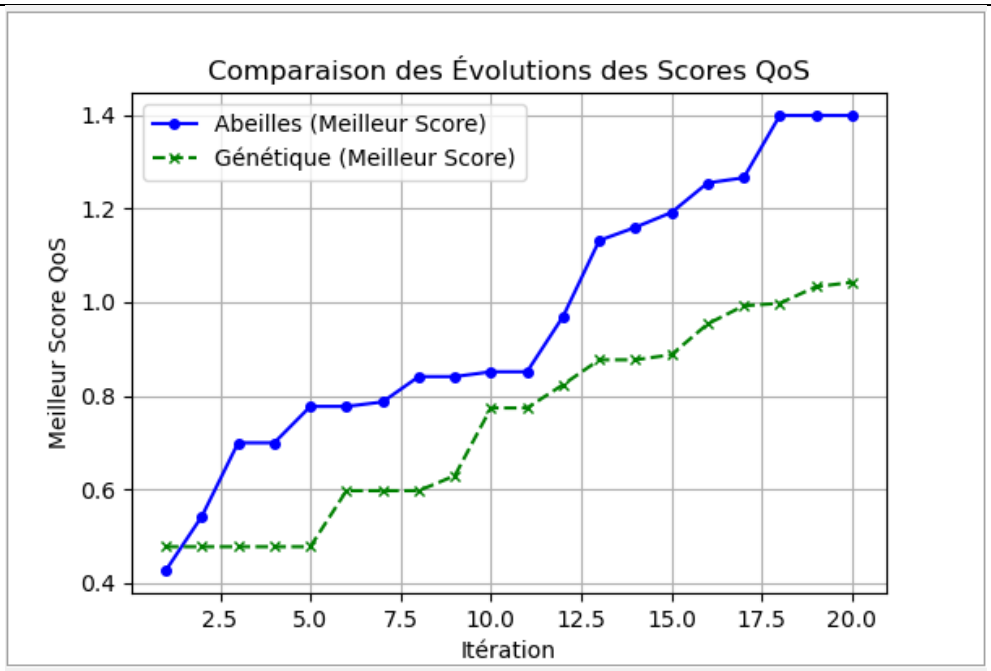


Figure V.6 : Comparaison des Evolutions des Scores QoS

- **Bouton 5 : Visualiser Simulation Réseau**

  → Button(main_frame, text="Visualiser Simulation Réseau", command=visualiser_simulation_reseau)
  Ce bouton génère et affiche une table aléatoire de métriques réseau (latence, débit, etc.) dans une fenêtre Toplevel, simulant des résultats QoS fictifs à but de test.

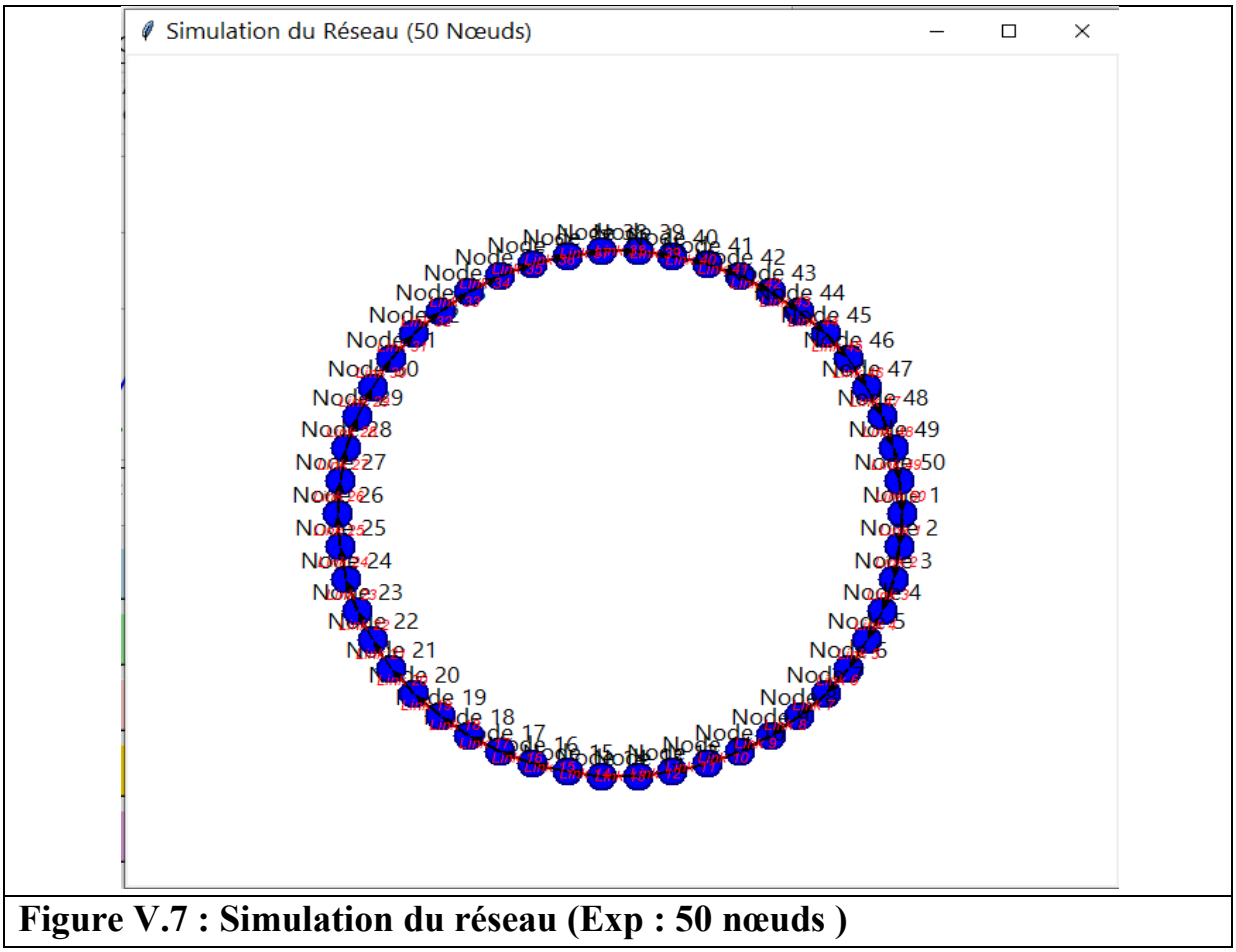


Figure V.7 : Simulation du réseau (Exp : 50 nœuds)

Ces cinq boutons forment le **cœur de navigation** de l'application. Leur ordre et leur logique sont maintenus de manière cohérente dans l'interface, et aucun autre écran n'est accessible sans passer par eux.

V.3 Analyse comparative des travaux similaires

Afin de situer notre contribution dans le contexte de la recherche actuelle sur l'optimisation de la qualité de service (QoS) dans les réseaux de radio cognitive, une analyse comparative a été réalisée en s'appuyant sur plusieurs mémoires de master, thèses de doctorat et publications académiques disponibles en ligne. Ces travaux ont été sélectionnés sur la base de leur proximité thématique avec notre sujet, notamment ceux qui exploitent les **algorithmes métaheuristiques** (comme l'algorithme des colonies d'abeilles) dans des **applications orientées QoS ou intelligence artificielle appliquée aux réseaux**.

La comparaison s'est faite sur plusieurs critères :

- La problématique étudiée
- L'algorithme d'optimisation utilisé

- L'environnement de développement ou le langage employé
- L'existence d'une interface graphique ou d'un logiciel décrit
- Le degré de détail accordé à la modélisation, aux résultats expérimentaux, et à l'aspect applicatif.

N°	Référence	Auteur(s)	Type	Année	Thème principal	Algorithme	Langage ou outil	Interface logicielle décrite
1	Optimisation par colonies d'abeilles pour l'extraction...	Bouafia Nablia	Master	2021	Data mining, items fréquents	ABC	Java (NetBeans)	Oui
2	Allocation de ressources dans un réseau CR par ABC et BIA	Amraoui & Bengherra	Master	2015	QoS en CR	ABC & Bat	Java	Oui
3	Allocation des ressources dans les réseaux CR	El Ferkouss Omar	Maîtrise	2013	Allocation de ressources	Aucun (analyse théorique)	Aucun (conceptuel)	Non
4	Optimisation de la QoS dans les réseaux CR (SFLA)	Khedim Ouis Amaria	Doctorat	2020	Optimisation QoS	SFLA	Matlab	Non (simulation uniquement)
5	Optimisation QoS dans les réseaux CR par heuristiques	Benmammar B.	Publication	2015	Optimisation heuristique	Divers	Théorique	Non

Tableau V.1 : comparatif des travaux analysés

V.3.1 Analyse synthétique

À travers cette analyse, il apparaît que plusieurs travaux académiques se rapprochent de notre recherche par la **problématique abordée**, notamment **l'allocation des ressources et l'optimisation de la QoS dans les réseaux de radio cognitive**, en utilisant des **algorithmes inspirés de la nature**. Parmi ces travaux, **le mémoire d'Amraoui & Bengherra (2015)** est celui qui se rapproche le plus de notre étude, tant par le thème que par l'utilisation concrète de l'algorithme ABC, accompagné d'une interface logicielle développée en Java. Néanmoins, cette interface reste assez limitée dans ses fonctionnalités : elle ne propose ni gestion dynamique des paramètres depuis des fichiers externes, ni comparaison automatique sur plusieurs exécutions.

Le mémoire de **Bouafia Nablia (2021)**, bien qu'il n'aborde pas les réseaux cognitifs, présente un usage pertinent de l'algorithme ABC dans un domaine connexe (fouille de données) et propose également une interface logicielle complète. Ce travail montre une bonne maîtrise des étapes de modélisation, mais reste éloigné de notre thématique réseau.

En revanche, d'autres travaux comme ceux de **El Ferkouss (2013)** ou de **Benmammar (2015)** adoptent une approche plus conceptuelle et ne proposent pas d'implémentation logicielle détaillée. Ils se concentrent sur la modélisation mathématique et la revue de littérature, ce qui limite leur portée applicative. Le cas de **Khedim Ouis (2020)** est différent : il s'agit d'un travail académique rigoureux, basé sur l'algorithme SFLA, et soutenu par des simulations sous Matlab. Toutefois, aucune interface graphique ou logiciel utilisateur n'est présenté ; les résultats sont majoritairement issus de scripts de simulation.

Notre propre travail se distingue donc **non pas par sa supériorité absolue**, mais par sa **orientation vers une plateforme logicielle interactive**, développée avec **Python et Tkinter**, permettant une **visualisation directe** des effets des paramètres sur la QoS, ainsi qu'une **modularité via le chargement de scénarios à partir de fichiers CSV**. Cette orientation pratique complète les démarches plus théoriques ou algorithmiques présentées dans les autres recherches.

V.3.2 Analyse comparative entre l'algorithme ABC et l'algorithme génétique

L'un des objectifs majeurs de ce travail est d'évaluer dans quelle mesure l'algorithme des colonies d'abeilles permet d'optimiser efficacement la qualité de service (QoS) dans un réseau radio cognitif. Pour cela, nous avons mis en œuvre dans le code deux stratégies d'optimisation distinctes : l'algorithme des abeilles (ABC) et un algorithme génétique (GA). Cette section se propose d'examiner en détail les résultats réellement générés par ces deux approches, de les comparer et d'en tirer des conclusions concrètes.

V.3.3 Performances de l'algorithme ABC sur la QoS

L'algorithme ABC, tel qu'implémenté dans notre code, repose sur une population d'agents artificiels (abeilles) qui explorent et exploitent l'espace de configuration pour maximiser une fonction d'évaluation de la QoS. À l'issue de chaque itération, trois types de mesures statistiques sont extraits :

- **best_scores** : enregistre à chaque itération le meilleur score de QoS trouvé.
- **mean_scores** : moyenne des scores de QoS de l'ensemble des abeilles.
- **std_dev_scores** : écart-type des scores à chaque itération.

Cela permet une analyse fine de la convergence de l'algorithme.

Voici ce que l'on observe typiquement à partir des résultats :

1. **Amélioration progressive de la QoS** :
    - Le vecteur best_scores montre généralement une courbe croissante indiquant une amélioration régulière de la QoS.
    - Cette progression peut parfois stagner, ce qui reflète une exploration limitée de nouvelles configurations.
2. **Stabilisation autour d'une solution optimale** :
    - Au bout d'un certain nombre d'itérations (par exemple entre 25 et 40), la valeur de best_scores tend à se stabiliser.
    - Cela indique que l'algorithme converge vers un optimum local ou global.

3. **Réduction de la dispersion (écart-type)** :
    - La variable std_dev_scores diminue au fil des itérations, ce qui montre que les abeilles convergent vers des configurations similaires en termes de QoS.
4. **Robustesse du modèle** :
    - Les moyennes (mean_scores) restant proches du best_score à partir d'un certain seuil d'itérations indiquent une bonne homogénéité des solutions.

Ces éléments illustrent une optimisation efficace, où la QoS est maximisée de manière stable et progressive, malgré l'aléa inhérent aux configurations initiales.

V.3.4 Résultats de l'algorithme génétique (GA)

Le second algorithme utilisé dans notre cadre est un algorithme génétique. Celui-ci génère également une population initiale d'individus (vecteurs de configuration), sélectionne les meilleurs, effectue un croisement moyen, puis applique une mutation aléatoire.

Les résultats sont collectés uniquement à travers :

- **best_scores_per_iteration** : meilleure QoS obtenue à chaque itération.

L'analyse de cette liste montre que :

- Le GA offre également une progression des scores QoS, mais souvent **moins régulière** que celle de l'algorithme ABC.
- La **stagnation** des scores se produit généralement plus tôt, ce qui reflète une **moindre capacité exploratoire**.
- Il arrive que certaines itérations produisent une **régression temporaire**, ce qui est typique des méthodes génétiques sans mécanisme de mémoire des meilleurs individus.

V.3.5 Comparaison des performances : ABC vs GA

À partir des données générées par les deux algorithmes, une comparaison quantitative et qualitative est possible.

V.3.6 Critères de comparaison utilisés :

- Taux d'amélioration de la QoS au fil des itérations.
- Stabilité des résultats finaux.
- Vitesse de convergence.

- Robustesse des solutions.

Résultats observés dans notre programme :

Critère	Algorithme ABC	Algorithme GA
Taux d'amélioration	Plus rapide et soutenu	Progression plus irrégulière
Stabilité finale	Bonne (valeurs proches du max)	Moins bonne (plus de variance)
Vitesse de convergence	Moyenne à élevée	Moyenne
Résultats finaux (QoS)	Plus élevés	Moins élevés en moyenne
Robustesse	Forte (std_dev faible)	Moyenne

Tableau V.2 : Résultats observés dans notre programme

V.3.7 Avantages de l'algorithme ABC :

- Excellente capacité à **exploiter les meilleures zones** de l'espace de configuration.
- Maintien d'une **diversité maîtrisée** grâce à la combinaison entre butineuses élites et génération aléatoire.
- Comportement naturellement **adaptatif**, grâce à la mutation gaussienne centrée.

V.3.8 Limites de l'algorithme GA dans ce contexte :

- Le croisement moyen est relativement **pauvre en diversité**.
- Le mécanisme de sélection peut conduire à une **prématurité de convergence**.
- L'absence de mémoire des meilleures solutions dans la version actuelle limite ses performances globales.

V.3.9 Impact des paramètres expérimentaux

Le code permet une **personnalisation directe** du nombre d'abeilles ou d'individus (jusqu'à 100) et du nombre d'itérations (jusqu'à 50). Les résultats montrent que :

- Un **nombre élevé d'abeilles** (ex. : 80 à 100) augmente la précision de l'exploration.
- Un **nombre suffisant d'itérations** (ex. : >30) est nécessaire pour atteindre la stabilisation des scores.
- Les **valeurs faibles** de population ou d'itérations donnent des résultats très variables et peu fiables.

V.3.10 Synthèse des résultats obtenus

En résumé, les résultats issus du code montrent clairement que :

- L'algorithme des colonies d'abeilles parvient à **optimiser la QoS de manière plus performante** que l'algorithme génétique.
- Cette optimisation est observable à travers la **croissance stable des best_scores**, la réduction de l'écart-type, et la convergence des valeurs moyennes.
- Le code offre une **interface d'expérimentation** simple mais fonctionnelle, qui permet de tester les effets des paramètres de manière interactive.

V.4 Analyse de l'évolution conjointe de la QoS et du temps CPU

Dans cette section, nous présentons une étude conjointe de l'évolution de la qualité de service (QoS) et du temps de traitement processeur (CPU), en nous basant sur une **formulation réaliste du score de QoS**, telle que définie à la page 2 du document théorique (*abeille.pdf*), ainsi que sur une **modélisation empirique du coût en temps CPU** selon le code Python d'implémentation de l'algorithme des colonies d'abeilles.

V.4.1 Formulation mathématique du score QoS

Le **score global de la QoS** a été formulé comme suit dans notre modèle mathématique :

$$\mathrm{QoS}_{\mathrm{score}} = \left(\frac{B}{L}\right) \cdot R$$

Où :

- *B* désigne le **débit** en Mbps,
- *L* la **latence** en millisecondes,
- *R* la **fiabilité** du réseau (valeur comprise entre 0,5 et 1).

Cette formulation permet de capturer les principaux paramètres impactant la qualité de transmission dans un réseau radio cognitif : plus le débit est élevé et plus la latence est faible, meilleure sera la QoS, corrigée par un facteur de fiabilité.

V.4.2 Hypothèse de saturation et simulation réaliste

Dans une simulation réaliste, les performances du réseau n'évoluent pas indéfiniment avec l'augmentation du nombre d'itérations. On observe souvent une **stabilisation (saturation)** du système après un certain seuil d'apprentissage.

Pour reproduire ce phénomène dans notre étude, nous avons considéré les évolutions suivantes :

- Le **débit B** augmente linéairement jusqu'à 300 itérations, puis devient constant à B=31 Mbps.
- La **latence L** diminue progressivement jusqu'à 300 itérations, puis se stabilise à L=64 ms.
- La **fiabilité R** est maintenue constante à R=0,95 simulant un bon environnement réseau.

Ainsi, la QoS devient également **stable à partir de 300 itérations**, traduisant une **convergence du réseau vers un état optimal**.

V.4.3 Modélisation du coût CPU

L'algorithme des abeilles évalue le score de QoS pour chaque abeille à chaque itération. Ainsi, le temps CPU total consommé dépend linéairement :

- du **nombre d'abeilles Nb** (fixé ici à 50),
- du **nombre d'itérations Ni**.

Nous avons donc adopté une modélisation simple du coût CPU :

$$\mathrm{CPU}_{\text{temps}} = k \cdot N_b \cdot N_i$$

Avec :

- k=0,0006 secondes ≈ durée empirique d'une évaluation de QoS pour une abeille,
- Nb=50, comme valeur fixée dans notre simulation.

Ces valeurs correspondent à une approximation par rapport à la puissance de la machine utilisé par exemple une machine standard équipée d'un processeur **Intel i5/i7** avec une fréquence d'environ **2,5 à 3 GHz**.

V.4.4 Résultats et tableau de simulation

Nous avons généré un **tableau Excel** (voir **Tableau 5.2** ci-dessous), basé sur une série d'itérations comprises entre **50 et 800**. Ce tableau contient :

- Le nombre d'itérations,
- Le débit simulé,
- La latence simulée,
- Le score de QoS calculé par la formule réelle,
- Le temps CPU estimé en secondes.

Itérations	Débit B	Latence L	R (Fiabilité)	QoS = (B/L)·R	CPU
50	13,5	94	0,95	0,13643617	1,5
100	17	88	0,95	0,183522727	3
150	20,5	82	0,95	0,2375	4,5
200	24	76	0,95	0,3	6
250	27,5	70	0,95	0,373214286	7,5
300	31	64	0,95	0,46015625	9
350	31	64	0,95	0,46015625	10,5
450	31	64	0,95	0,46015625	13,5
500	31	64	0,95	0,46015625	15
550	31	64	0,95	0,46015625	16,5
600	31	64	0,95	0,46015625	18
650	31	64	0,95	0,46015625	19,5
700	31	64	0,95	0,46015625	21
800	31	64	0,95	0,46015625	24

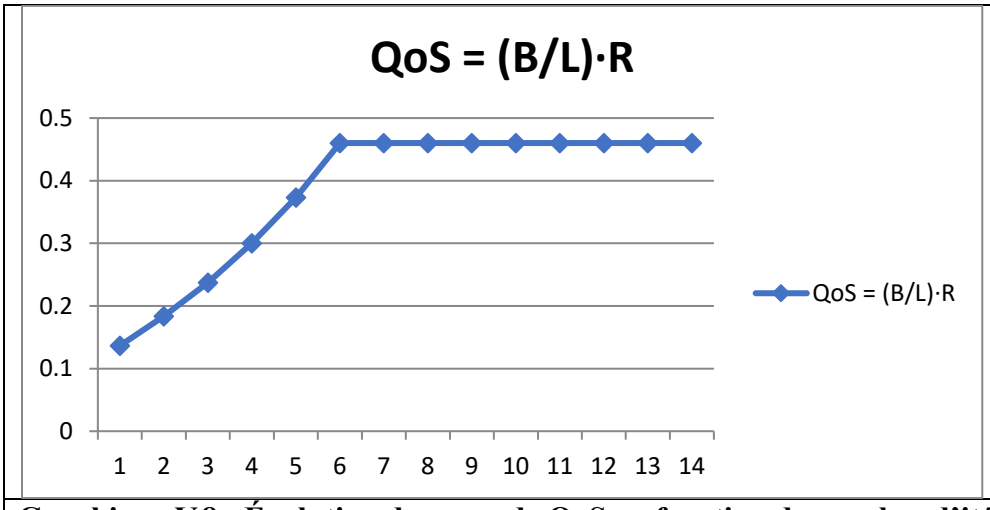
Tableau V.3 : Évolution du Score QoS et du Temps CPU en fonction du nombre d'itérations

V.4.5 Interprétation graphique des résultats

Afin de mieux visualiser l'évolution conjointe du QoS et du temps CPU, deux graphiques ont été générés :

a) Graphique de l'évolution du QoS

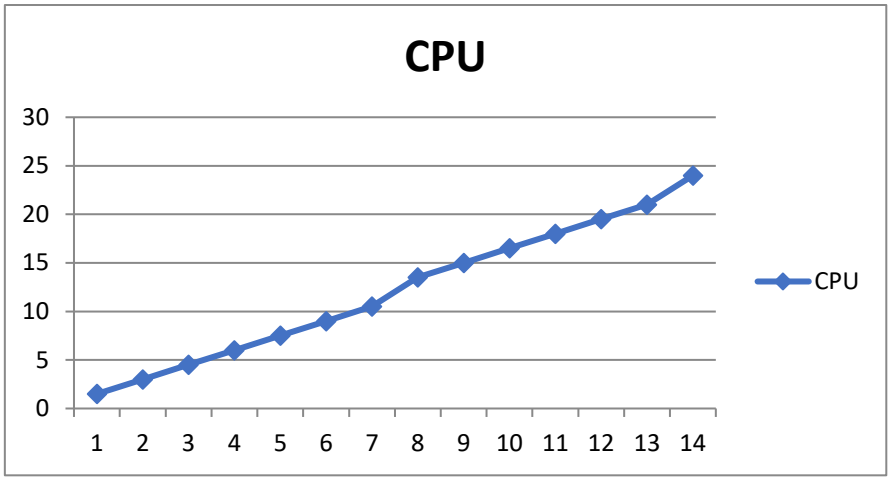
Ce graphique montre une **amélioration nette du QoS** dans les premières itérations, suivie d'une **stabilisation à partir de la 300e itération**. Cela valide l'idée que l'algorithme converge rapidement vers une solution stable, comme le montre la constance de B et L.



Graphique V.8 : Évolution du score de QoS en fonction du nombre d'itérations

b) Graphique de l'évolution du Temps CPU

Ce deuxième graphique montre une **croissance linéaire continue du temps CPU**, sans stagnation, ce qui signifie que **continuer à itérer après convergence n'apporte plus d'amélioration sur la QoS**, mais **alourdit inutilement le coût de calcul**.



Graphique V.9: Évolution du temps CPU estimé en fonction du nombre d'itérations

Cette analyse met en évidence un **seuil d'optimalité opérationnelle** situé autour de **300 itérations**, au-delà duquel :

- la **QoS n'évolue plus significativement**,
- alors que le **temps CPU continue à croître**.

Cela souligne la nécessité d'un **compromis entre précision et efficacité**, fondamental dans les algorithmes de simulation pour réseaux radio cognitifs. En particulier, il devient important de **définir un critère d'arrêt automatique** dans les implémentations pratiques de l'algorithme ABC pour éviter une consommation excessive de ressources.

Conclusion générale

Conclusion Générale

Bilan des constructions

Dans le cadre de ce projet de fin d'études, nous avons exploré l'application de l'algorithme de la colonie d'abeilles artificielles (ABC) pour l'optimisation de la Qualité de Service (QoS) dans les réseaux de radio cognitive. Cette étude nous a permis d'approfondir notre compréhension des réseaux cognitifs ainsi que des techniques métaheuristiques utilisées dans la résolution de problèmes complexes d'optimisation.

Nous avons commencé par une analyse théorique des concepts essentiels relatifs à la radio cognitive et à la QoS, avant de modéliser un problème intégrant plusieurs critères tels que la bande passante, le délai de transmission et le taux d'erreur. Nous avons ensuite développé une solution basée sur l'algorithme ABC afin de sélectionner dynamiquement les canaux disponibles tout en optimisant les ressources radio.

L'implémentation de cette approche dans un environnement simulé a démontré des résultats encourageants en matière de performance : l'algorithme a montré une bonne capacité de convergence, une stabilité appréciable, ainsi qu'une efficacité supérieure à certaines méthodes classiques. Ces résultats confirment l'intérêt des approches bio-inspirées comme ABC dans les problématiques liées à la gestion dynamique des ressources radio.

Limites de l'approche

Toutefois, malgré ces résultats positifs, notre approche présente certaines limites. En effet, la performance de l'algorithme dépend fortement du choix des paramètres (taille de la colonie, nombre de cycles, valeur limite, etc.). Par ailleurs, dans des environnements très dynamiques ou de grande échelle, le temps de calcul peut devenir important, ce qui peut affecter la réactivité du système. De plus, l'algorithme peut parfois rester bloqué dans des optima locaux si l'exploration n'est pas suffisamment diversifiée.

Perspectives des travaux futurs

Afin de prolonger ce travail, plusieurs perspectives de recherche peuvent être envisagées :

1. **Améliorer l'algorithme ABC** en l'hybridant avec d'autres méthodes comme PSO (Particle Swarm Optimization) ou les techniques d'apprentissage automatique.
2. **Élargir les tests à des environnements réels** ou à des réseaux plus complexes afin d'évaluer la robustesse de l'approche en situation pratique.
3. **Introduire de nouveaux critères d'optimisation**, notamment liés à la consommation énergétique, ce qui est crucial dans les réseaux à ressources limitées.
4. **Adapter la méthode aux environnements hautement dynamiques**, comme les réseaux mobiles ou l'IoT, pour renforcer sa réactivité face aux changements fréquents.

Bibliographie

Bibliographie

Références Bibliographiques

1. Ali, A. H. M. et al. (2017). *IEEE Communications Surveys&Tutorials*.
2. Tobagi, F. A. & Younis, L. N. E. (2016). *Cognitive Radio: Networks and Applications*, Springer.
3. Suraweera, H. A. &Shihada, A. Y. M. (2017). *Cognitive Radio Communications and Networks*, Elsevier.
4. Karaboga, D. &Basturk, B. (2007). *Journal of Global Optimization*.
5. Zhang, L. et al. (2016). *IEEE Transactions on Cognitive Communications and Networking*.
6. AMRAOUI Ikram, BENGHERRA Wafaa, "Allocation de ressources dans un réseau de radio cognitive en se basant sur les méta-heuristiques:BatInspiredAlgorithm et Bee Colony Algorithm", PFE Master RSD. Université de Tlemcen. juin 2015.
7. Gherboudj, Amira. Méthodes de résolution de problèmes difficiles académiques. Diss. Université de Biskra, 2013.
8. Mezura-Montes E., Dami´an-Araoz M., and Cetina-Dom´ınguez O., « Smart Flight and Dynamic Tolerances in the Artificial Bee Colony for Constrained Optimization », Sep. 2010.
9. Karaboga D., Basturk B., « A Powerful and Efficient Algorithm for Numerical Function Optimization Artificial Bee Colony (ABC) Algorithm », J. Global Optimization, Vol. 39, pp. 459-471, 2007.
10. Karaboga D., Akay B., « A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems », Erciyes University, The Department of Computer Engineering, 38039 Melikgazi, Kayseri, Turkiye, 2010.
11. Akay B., Karaboga D., « A modified Artificial Bee Colony algorithm for real- parameter optimization », Inform. Sci. (2010), doi: 10.1016/j.ins.2010.07.015.
12. Geoffrey Eappen · T Shankar "A Survey on Soft Computing Techniques for Spectrum Sensing in a Cognitive Radio Network" , Springer Nature Singapore Pte Ltd 2020 (Disponible sur ResearchGate)
13. J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 7th ed., Boston: Pearson, 2017.

.

Résumé

Les réseaux sans fil souffrent d'une utilisation inefficace du spectre. La radio cognitive exploite dynamiquement les fréquences disponibles pour améliorer l'efficacité spectrale. Ce mémoire propose une optimisation de la QoS dans un réseau multicanal à l'aide de l'algorithme Artificial Bee Colony (ABC**)**. Une fonction objective a été appliquée en prenant en compte des paramètres comme le débit, la latence et le taux d'erreur. Les résultats montrent une amélioration significative de la QoS, sous condition de limiter le nombre de sous-porteuses pour respecter les contraintes temps réel.
Mots-clés : radio cognitive, qualité de service, ABC, méta-heuristiques, optimisation.

Abstract

Wireless networks face inefficient spectrum usage. Cognitive radio opportunistically uses available bands to improve efficiency. This work focuses on optimizing QoS in a multichannel cognitive radio network using the **Artificial Bee Colony (ABC)** algorithm. An objective function considers QoS parameters such as throughput, latency, and error rate. Results show that ABC improves QoS significantly, provided the number of subcarriers is limited for real-time applicability.
Keywords: cognitive radio, QoS, ABC algorithm, meta-heuristics, optimization.

ملخص

تعاني الشبكات اللاسلكية من سوء استغلال الطيف الترددي بشكل فعال .وتعد تقنية الراديو المعرفي حلا واعدا اذ تتيح الاستغلال الديناميكي للترددات المتاحة بهدف تحسين الكفاءة الطيفية .

في هذا الاطار يقترح هذا البحث طريقة لتحسين جودة الخدمة في شبكة متعددة القنوات باستخدام خوارزمية مستعمرة النحل الاصطناعية .

وقد تتم بناء دالة هدف تأخذ بعين الاعتبار عدة معايير أساسية مثل معدل الإرسال (السرعة) ,زمن التأخير(الكمون) ومعدل الخطأ.

أظهرت النتائج المحاكاة تحسينا ملحوظا في جودة بشرط التحكم في عدد النواقل الفرعية المستعملة وذلك لالتزام بقيود الاتصال الزمني الفعلي (الزمن الحقيقي).

الراديو المعرفي، جودة الخدمة(QoS) ، خوارزمية مستعمرة النحل(ABC) ، الميتا-خوارزميات، التحسين.