Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : *Réseaux et Systèmes Distribué (R.S.D)*

Thème

Amélioration des performances d'un RRC en utilisant la parallélisation de la programmation dynamique

Réalisé par :

- **Youssouf BRAHIMI**
- **Karim Ilyes MAHMOUDI**

Présenté le 4 Juillet 2017 devant le jury composé de MM.

- *LEHSAINI Mohamed* *(Président)*
- *AMRAOUI Asma* *(Examinateur)*
- *BENMAMMAR Badr* *(Encadrant)*

Année universitaire : 2016-2017

Remerciements

Nos remerciements s’adressent à Dieu tout puissant qui nous a donné le courage, la force et les moyens pour réussir et donner le meilleur de nous-mêmes.

Nos remerciements s'adressent à notre encadrant Mr Badr BENMAMMAR pour avoir accepté de diriger ce travail, on est reconnaissant pour son estimable soutien, sa clairvoyance et ses compétences qui nous ont beaucoup impressionnés.

Nous adressons toute notre gratitude à l’ensemble du jury, dont Mr LEHSAINI, Mr BENMAMMAR, Mme AMRAOUI en leur témoignant notre profond respect et immense intérêt, pour leurs estimables efforts ainsi qu’à leur importante contribution à notre évaluation.

Ce travail n’aurait pas été possible sans le soutien de nos familles, c’est pour ça que nous tenons à remercier nos parents ainsi que nos proches pour tous les encouragements.
Nos remerciements les plus chaleureux vont à tous ceux qui nous ont apportés de l’aide de loin ou de prêt.
Et enfin nous tenons à remercier nos amis ainsi que nos collègues qui ont contribué à l’accomplissement de ce travail.

Table des matières

Table des matières 1

Liste des figures 4

Liste des tableaux 4

GLOSSAIRE 5

Introduction Générale 6

Chapitre 1 : Réseaux de radio cognitive 8

1.1 Introduction 8

1.2 Historique 9

1.3 Définition et principe 9

1.4 Radio logiciel 10

1.5 Relation entre radio cognitive et radio logicielle restreinte 10

1.6 Architecture de la radio cognitive 12

1.7 Cycle de cognition 13

1.8 Réseaux de radio cognitive 14

1.8.1 Architecture des réseaux de radio cognitive 15

1.8.1.1 Réseau primaire 15

1.8.1.2 Réseau secondaire 15

1.8.2 Fonctions des réseaux de radio cognitive 16

1.8.2.1 Détection du spectre (Spectrum sensing) 16

1.8.2.2 Partage du spectre (Spectrum sharing) 16

1.8.2.3 Gestion du spectre (Spectrum management) 16

1.8.2.4 Mobilité du spectre (Spectrum mobility) 17

1.8.3 Domaines d'application des réseaux de radio cognitive 18

1.9 Conclusion 18

Chapitre 2 : Méthodes de résolution 19

2.1 Introduction 19
2.2 Les méthodes exactes 19
2.2.1 La programmation dynamique 19
2.3 Les méthodes approchées 20
2.3.1 Les heuristiques 20
2.3.2 Les métaheuristiques 21
2.3.2.1 Cuckoo search 22
2.3.2.2 L'algorithme Firefly 24
2.4 Conclusion 26
Chapitre 3 : Implémentation de l'application et évaluation des résultats 28
3.1 Introduction 28
3.2 La fonction objectif (fitness function) 28
3.3 Réalisation des algorithmes et résultats 29
3.3.1 La programmation dynamique 29
3.3.2 L'algorithme cuckoo search 32
3.3.2.1 Initialisation de la population 32
3.3.2.2 Génération de nouvelles solutions 32
3.3.2.3 Critère d'arrêt 32
3.3.3 L'algorithme Firefly 32
3.3.3.1 Génération de la population initiale 32
3.3.3.2 Classement 33
3.3.3.3 Déplacement et mise à jour 33
3.3.3.4 Critère d'arrêt 33
3.3.3.5 Paramétrage de l'algorithme firefly 34
3.4 Outils utilisés pour l'implémentation de l'application 34
3.4.1 NetBeans 34
3.4.2 Adobe Photoshop 35

3.4.3 JFreeChart.....35
3.5 Présentation de l'application.....36
3.6 Etude Comparative.....41
3.6.1 Comparaison en termes de temps de traitement côté PU.....41
3.6.2 Comparaison en termes de gain coté PU.....43
3.6.3 Comparaison en termes de nombre de SU satisfaits.....44
3.7 Conclusion.....45
Conclusion générale.....46
Références Bibliographiques.....47
Résumé.....50

Liste des figures

Figure 1.1 : architecture de la radio cognitive [8] 12
Figure 1.2 : cycle de cognition [9] 13
Figure 1.3 : les fonctions de RRC [10] 17
Figure 3.1 : matrice de la programmation dynamique 29
Figure 3.2 : temps d'exécutions par rapport au nombre de threads 31
Figure 3.4 : interface d'accueil de l'application 36
Figure 3.5 : interface choix de l'application 37
Figure 3.6 : interface comparaison de l'application 38
Figure 3.7 : interface programmation dynamique 39
Figure 3.8 : interface choix méthodes approchées 40
Figure 3.9 : interface configuration algorithme CuckooSearch 40
Figure 3.10 : comparaison en termes de temps de traitement 41
Figure 3.11 : comparaison en termes de gain(coût) 43
Figure 3.12 : comparaison en termes du nombre de SU satisfaits 44

Liste des tableaux

Tableau 3.1 : temps d'exécution en fonction du nombre de threads 30
Tableau 3.2 : coût et nombre de SU satisfaits obtenus par la programmation dynamique 31
Tableau 3.3 : résultats de l'algorithme Cuckoo search.....32
Tableau 3.4 : résultats de l'algorithme Firefly 34
Tableau 3.5 : résultats de la comparaison en termes de temps de traitement 42

GLOSSAIRE

Acronyme	Signification
RRC	**R**éseaux de **R**adio **C**ognitive
RC	**R**adio **C**ognitive
RDL	**R**adio **D**éfini par **L**ogiciel
SRC	**S**ystème de **R**adio **C**ognitive
FCC	**F**ederal **C**ommunications **C**ommission
Ofcom	**Of**fice of **Com**munications
WRC	**W**orld **R**adiocommunication **C**onference
KTH	**K**ungliga **T**ekniska **H**ögskolan
SDR	**S**oftware **D**efined **R**adio
RF	**R**adio **F**réquence
PU	**P**rimary **U**ser
SU	**S**econdary **U**ser
URC	**U**tilisateur de la **R**adio **C**ognitive
CPU	**C**entral **P**rocessing **U**nit
IA	**I**ntelligence **A**rtificielle
FA	**F**irefly **A**lgorithme
CS	**C**ucko**o** **S**earch
API	**A**pplication **P**rogramming **I**nterface
CDDL	**C**ommon **D**evelopment and **D**istribution **L**icense
GPL	**G**eneral **P**ublic **L**icense
HTML	**H**yper**T**ext **M**arkup **L**anguage
PHP	**H**ypertext **P**reprocessor
XML	E**x**tensible **M**arkup **L**anguage
LGPL	**L**esser **G**eneral **P**ublic **L**icense

Introduction Générale

L'évolution des systèmes de transmission augmente les risques d'interférences et de saturation du spectre électromagnétique. En effet, nous assistons actuellement à la multiplication des normes et des standards de télécommunication vu les progrès récents dans ce domaine. Le nombre croissant de standards normalisés permet d'agrandir l'éventail des offres et des services disponibles pour chaque utilisateur, d'ailleurs, la plupart des radiofréquences disponibles ont déjà été allouées [1].

Une étude réalisée par la FCC a montré que certaines bandes de fréquences sont partiellement occupées dans des emplacements particuliers et à des moments particuliers. Et c'est pour toutes ces raisons que la Radio Cognitive (RC) est apparue. L'idée de la RC est de partager le spectre entre un utilisateur dit primaire, et un utilisateur dit secondaire. L'objectif principal de cette gestion du spectre consiste à obtenir un taux maximum de l'exploitation du spectre radio, pour que cela fonctionne, l'utilisateur secondaire doit être capable de détecter d'une manière dynamique l'espace blanc, de se configurer pour transmettre, de détecter le retour de l'utilisateur primaire et ensuite cesser de transmettre et chercher un autre espace blanc. Le standard IEEE 802.22, qui est basé sur ce concept, est actuellement en cours de développement [1].

Dans ce PFE, nous tenterons d'apporter une solution à la gestion du spectre en utilisant une technique basée sur la négociation entre un PU et plusieurs SUs, pour cela nous avons étudiés deux approches de résolution, la première est dite exacte et la seconde est dite approchée. Nous avons choisi la programmation dynamique comme méthode exacte ainsi que deux métaheuristiques qui sont Cuckoo Search (CS) et Firefly (FA) comme méthodes approchées afin d'élaborer notre étude comparative selon différents critères que nous avons fixés (temps de traitement, coût obtenu et nombre de SU satisfaits). Cette étude comparative nous a permis d'aboutir à des résultats démontants laquelle de ces méthodes citées offre le meilleur rendement pour la gestion du spectre. A noter ici que nous avons opté pour la parallélisation de la programmation dynamique car la version sans threads est déjà traitée dans la littérature. [1]

Le présent rapport est organisé comme suit :

Dans le premier chapitre : nous donnerons une présentation claire et détaillée de la RC ainsi que ses composantes et ses fonctionnalités et aussi sa frontière avec la radio logicielle puis de son architecture, des réseaux de RC de leurs fonctionnalités et de leurs domaines d'application.

Dans le deuxième chapitre : nous présentons les différentes méthodes utilisées lors de la réalisation de ce PFE. Nous présentons les méthodes exactes et les méthodes approchées en se focalisant sur les heuristiques et les métaheuristiques ainsi que leurs caractéristiques. Nous donnons également plus de détails sur les trois algorithmes : programmation dynamique, Cuckoo Search et Firefly ainsi que leurs fonctionnements.

Dans le dernier chapitre : nous présentons notre travail réalisé en détaillant les différents paramètres utilisés lors de l'implémentation des trois algorithmes pour ensuite présenter notre application et enfin nous expliquerons notre étude comparative en discutant des résultats obtenus.

Chapitre 1 :

Réseaux de radio cognitive

1.1 Introduction

Le concept de Radio Cognitive (RC) est apparu comme un nouveau paradigme en 1999 en tant que prolongement de la Radio Définie par Logiciel (RDL). Il décrit la situation dans laquelle les dispositifs radio intelligents et les entités de réseau associées se communiquent de telle manière qu'ils sont en mesure d'ajuster leurs paramètres de fonctionnement en fonction des besoins de l'utilisateur / du réseau et d'apprendre de l'expérience en même temps. Depuis, il y a eu beaucoup d'efforts dans le milieu de la recherche sur les sujets liés à la RC. Les activités de normalisation sur les systèmes de Radio Cognitive (SRC) (y compris TV WhiteSpaces-TVWS) ont également été lancées et ont progressé dans de nombreux organismes de normalisation. Presque tous les organismes de réglementation des États-Unis, d'Europe et de l'Asie et du Pacifique ont reconnu l'importance de SRC pour façonner la répartition du spectre. Les régulateurs comme FCC aux États-Unis et Ofcom au Royaume-Uni ont ouvert la voie à un accès secondaire à des appareils non autorisés sur des chaînes de télévision. Enfin, la conférence mondiale de radiocommunication (WRC 2012) [2] témoigne des discussions sur les modifications réglementaires requises pour permettre l'introduction du SRC. Malgré tous ces progrès, la radio cognitive est principalement un sujet de recherche aujourd'hui, et le marché sans fil n'a pas encore vu un déploiement / exploitation commerciale massive de la technologie RC. La RC présentée par Mitola reste un concept futuriste où une connaissance omniprésente est intégrée dans tous les types de dispositifs / équipements / applications qui sont conscients de nos besoins / souhaits, les exécutant et rendant notre vie quotidienne plus facile [3].

La radio cognitive est une technologie clé qui pourrait révolutionner le monde des réseaux sans fil. Ces derniers jouissent continuellement d’une demande qui croît et qui engendre un important encombrement du spectre car l’efficacité de celui-ci est désormais mise en péril, la RC est le concept qui permet de répondre à ce défi ; mieux utiliser le spectre c’est aussi augmenter les débits et rendre plus fiable la couche physique.

1.2 Historique

L'idée de la radio cognitive a été présentée officiellement par Joseph Mitola III à un séminaire à KTH, l'Institut royal de technologie, en 1998, publié plus tard dans un article de Mitola et Gerald Q. Maguire, Jr en 1999 [4]. Connu comme le « Père de la radio logicielle ». Dr. Mitola est l'un des auteurs les plus cités dans le domaine. Mitola combine son expérience de la radio logicielle ainsi que sa passion pour l'apprentissage automatique et l'intelligence artificielle pour mettre en place la technologie de la radio cognitive. Et donc d'après lui :

« Une radio cognitive peut connaître, percevoir et apprendre de son environnement puis agir pour simplifier la vie de l'utilisateur ».

1.3 Définition et principe

La RC est une forme de communication sans fil dans laquelle un émetteur/récepteur peut détecter intelligemment les canaux de communication qui sont en cours d'utilisation et ceux qui ne le sont pas, et peut se déplacer dans les canaux inutilisés. Ceci permet d'optimiser l'utilisation des fréquences radio disponibles du spectre tout en minimisant les interférences avec d'autres utilisateurs [1].

La radio cognitive est une nouvelle technologie qui permet, à l'aide d'une radio logicielle, de définir ou de modifier les paramètres de fonctionnement de la fréquence radio d'un nœud réseau (téléphone sans fil ou un point d'accès sans fil), comme par exemple, la gamme de fréquences, le type de modulation ou la puissance de sortie [5].

Cette capacité permet d'adapter chaque appareil aux conditions spectrales du moment et offre donc aux utilisateurs un accès plus souple, efficace et complet à cette ressource. Cette approche peut améliorer considérablement le débit des données et la portée des liaisons sans augmenter la bande passante ni la puissance de transmissions. La RC offre également une solution équilibrée au problème de l'encombrement du spectre en accordant d'abord l'usage prioritaire au propriétaire du spectre, puis en permettant à d'autres de se servir des portions inutilisées du spectre.

Un réseau cognitif coordonne les transmissions suivant différentes bandes de fréquences et différentes technologies en exploitant les bandes disponibles à un instant donné et à un endroit donné. Il a besoin d'une station de base capable de travailler sur une large gamme de

fréquences afin de reconnaître différents signaux présents dans le réseau et se reconfigurer intelligemment.

1.4 Radio logiciel

La radio logicielle est un ensemble de technologies pour définir les paramètres et les fonctions des émetteurs-récepteurs radio, y compris la fréquence des opérateurs, la bande passante de modulation, le codage des canaux et l'agilité fréquence / espace / temps / code. Dans les générations précédentes de systèmes de télécommunications, ces paramètres et fonctions étaient parfois sélectionnables, mais généralement définis par le matériel, avec une programmation programmée limitée au traitement du signal numérique en bande de base [6].

Après plusieurs années de développement un nouveau concept est apparu qui est la radio logicielle restreinte (software defined radio ou SDR), elle définit une collection de technologies matérielles et logicielles où certaines ou toutes les fonctions d'exploitation de la radio (également appelées traitement de couche physique) sont implémentées via un logiciel ou un microprogramme modifiable fonctionnant sur des technologies de traitement programmables. Ces dispositifs comprennent des tableaux de portes programmables sur site, des processeurs de signaux numériques, des processeurs à usage général ou d'autres processeurs programmables spécifiques à l'application. L'utilisation de ces technologies permet d'ajouter de nouvelles fonctions et capacités sans fil aux systèmes radio existants sans nécessiter de nouveau matériel.

1.5 Relation entre radio cognitive et radio logicielle restreinte

On s'attend à ce que les technologies de la radio logicielle restreinte et de la radio cognitive offrent une flexibilité supplémentaire et offrent une efficacité améliorée à l'utilisation globale du spectre.

Ces technologies peuvent être combinées ou déployées de manière indépendante, et peuvent être mises en œuvre dans des systèmes de tout service de radiocommunication, n’importe quel système utilisant des technologies de radio logicielle restreinte ou de radio cognitive doit fonctionner conformément aux dispositions du règlement des radiocommunications.

Les systèmes de radio cognitive sont un domaine d'activité de recherche, et les applications sont en cours d'étude et d'essai. Les systèmes qui utilisent certaines fonctionnalités cognitives ont déjà été déployés, et certaines administrations autorisent ces systèmes (par exemple, la sélection dynamique de la fréquence). Ces administrations ont des processus nationaux d'approbation des équipements pour protéger les services existants contre les interférences nuisibles. Un système radio mettant en œuvre des technologies de radio cognitive peut cependant avoir un impact sur les pays voisins et une coordination peut être nécessaire. Lorsqu'il existe des applications dans lesquelles la technologie des systèmes radio cognitifs est implémentée sur une base de non-interférence et de non-protection, l'administration concernée devrait s'assurer que les interférences ne seront pas générées.

La technologie radio logicielle restreinte fonctionne maintenant dans certains systèmes et réseaux dans les services mobiles, de radiodiffusion et de radiodiffusion par satellite, fixe et mobile par satellite. Elle offre une flexibilité dans la conception du système radio et peut aider à la compatibilité directe.

La mise en œuvre complète du concept de systèmes de radio logicielle restreinte et de systèmes de radio cognitive est susceptible d'être réalisée progressivement pour plusieurs raisons, y compris l'état actuel de la technologie. L'utilisation de ces technologies dans certains groupes peut poser des défis spécifiques et uniques de nature technique ou opérationnelle, qui doivent faire l'objet d'une évaluation attentive et complète de l'union international de télécommunication.

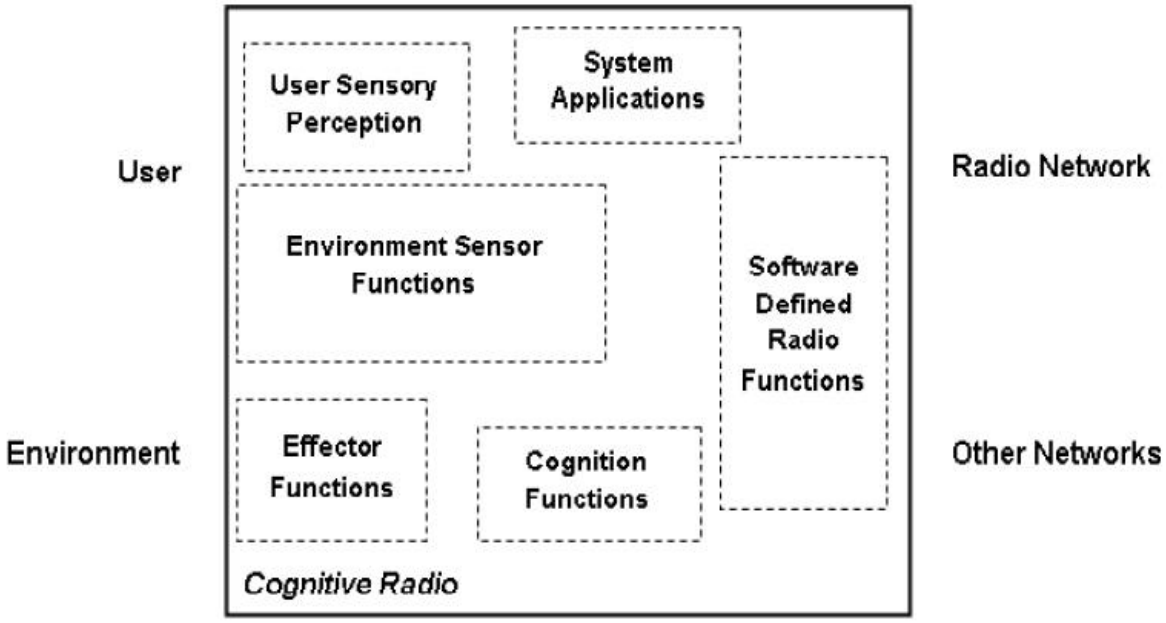


Figure 1.1 : architecture de la radio cognitive **[7]**

1.6 Architecture de la radio cognitive

Les six composantes fonctionnelles de l'architecture de la radio cognitive sont [7]:

1. La perception sensorielle de l'utilisateur qui inclut l'interface haptique (du toucher), acoustique, la vidéo et les fonctions de détection et de la perception. Les fonctions SP de l'utilisateur peuvent inclure un matériel optimisé, par exemple, pour calculer les vecteurs de flux vidéo en temps réel pour aider la perception d'une scène.

2. Les capteurs de l'environnement local (emplacement, la température, l'accéléromètre, compas, etc.)

3. Les applications système (les services médias indépendants comme un jeu en réseau).

4. Les fonctions SDR (qui incluent la détection RF et les applications radio de la SDR).

5. Les fonctions de la cognition (pour les systèmes de contrôle, de planification, de l'apprentissage).

6. Les fonctions locales effectrices (synthèse de la parole, du texte, des graphiques et des affiches multimédias).

1.7 Cycle de cognition

La figure ci-dessous schématise les différentes activités de la cognition qui sont : l'observation, l'orientation, la planification, la décision, l'action et l'apprentissage.

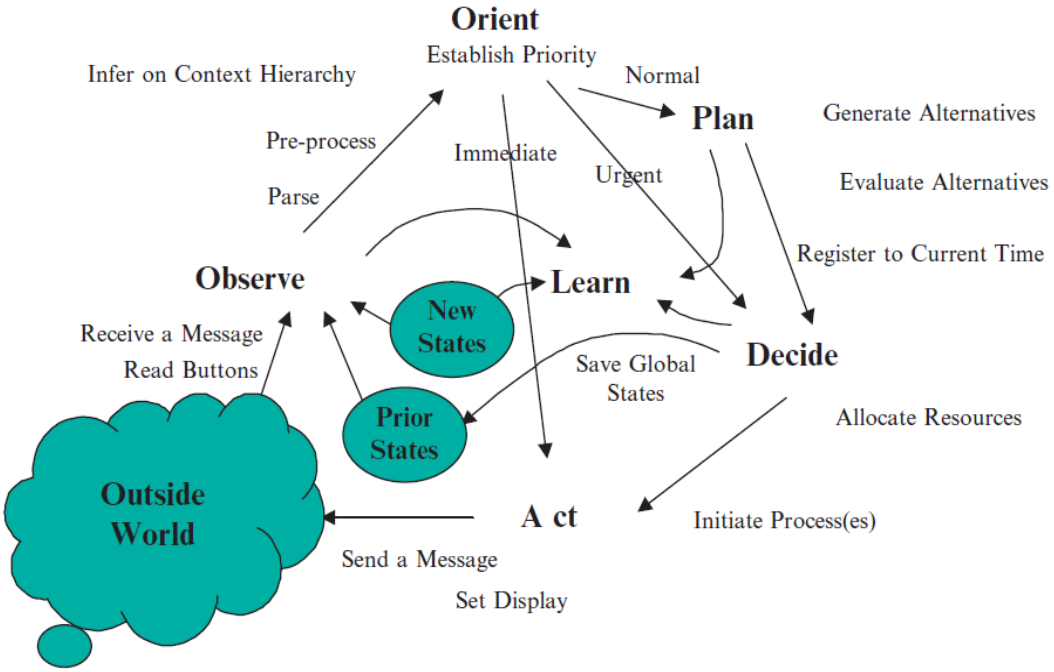


Figure 1.2 : cycle de cognition **[8]**

- **Phase d'observation (Observe)**

Dans la phase d'observation, la radio cognitive observe l'environnement et analyse le flux entrant afin de détecter les nouveautés.

- **Phase d'orientation (Orient)**

La phase d'orientation détermine l'importance de l'observation et de l'intensité de la réaction, parfois elle provoque une action qui sera lancé immédiatement comme un comportement réactif.

- **Phase de planification (Plan)**

Dans la phase de planification Un message entrant du réseau serait normalement traité par la génération d'un plan. Dans la recherche de qualité industrielle des radios cognitives, les

modèles formels de la causalité seraient incorporés dans les outils de planification. Le plan devrait également inclure la phase de raisonnement dans le temps.

- **Phase de décision (Decide)**

L'objectif de cette phase est de choisir le meilleur plan parmi les plans de candidature.

- **Phase d'action (Act)**

Cette phase lance les processus sélectionnés qui s'orientent vers le monde extérieur ou aux états internes de la radio cognitive.

L'accès au monde extérieur consiste d'abord à composer des messages qui doivent être envoyés dans l'environnement en audio ou exprimés dans différents langages. Les actions sur les états internes comprennent le contrôle de ressources comme les canaux radio.

Une action radio cognitive peut également mettre à jour les modèles internes, par exemple, l'ajout de nouveaux modèles aux modèles internes qui existent.

- **Phase d'apprentissage (Learn)**

Cette phase dépend de la perception, des observations, des décisions et des actions. Elle peut se produire lorsqu'un nouveau modèle est créé en réponse à une action afin de mieux apprendre sur les différentes fonctionnalités de la radio cognitive.

1.8 Réseaux de radio cognitive

Les réseaux de radio cognitive sont une approche novatrice de l'ingénierie sans fil dans laquelle les radios sont conçues avec un niveau d'intelligence et d'agilité sans précédent. Cette technologie avancée permet aux appareils radio d'utiliser le spectre (c'est-à-dire les fréquences radio) de manière entièrement nouvelle et sophistiquée. Les radios cognitives ont la capacité de surveiller, de détecter les conditions de leur environnement d'exploitation et de reconfigurer dynamiquement leurs propres caractéristiques pour mieux correspondre à ces conditions.

À l'aide de calculs complexes, les réseaux de radio cognitive (RRC) peuvent identifier des altérations potentielles pour la qualité de la communication, comme les interférences, la perte de chemin, l'ombre et la décoloration multi-voies. Ils peuvent ensuite ajuster leurs paramètres

de transmission, tels que la puissance, la fréquence et la modulation pour assurer une expérience de communication optimisée pour les utilisateurs.

1.8.1 Architecture des réseaux de radio cognitive

L'architecture des réseaux de radio cognitive se divise en deux : réseau primaire et réseau secondaire.

1.8.1.1 Réseau primaire

Le réseau primaire comporte une licence, cette dernière est attribuée par des organisations gouvernementales. Les réseaux mobiles sont un modèle de réseaux primaires dans lesquels les bandes spectrales sont les possesseurs.

Le réseau primaire détient deux éléments :

- **Utilisateur primaire (PU) :** L'utilisateur primaire possède une licence qui lui simplifie l'accès à certaines bandes spectrales. Elle lui offre également la capacité d'opérer à tout moment sur ces stations de base et de ne pas être perturbé par des utilisateurs inconnus. L'accès est inspecté uniquement par leurs stations de base, là où aucune interférence extérieure ne doit intervenir.
- **Station de base des PU :** La station de base des PU est une architecture immobile du réseau primaire qui lui est doté d'une licence lui permettant d'agir sur la bande spectrale telles que les stations de base des systèmes cellulaires.

1.8.1.2 Réseau secondaire

Nommé réseau non-licencié car il ne possède pas de licence pour fonctionner sur la bande spectrale. D'où le besoin de plusieurs fonctionnalités supplémentaire afin de classer les bandes spectrales licenciées de façon opportuniste, le réseau secondaire se compose de quatre éléments :

- **Utilisateur à radio cognitive** : L'utilisateur à radio cognitive URC, ou bien utilisateur secondaire SU appelé aussi utilisateur non-licencié, ne dispose pas de licence pour pouvoir diffuser sur la bande spectrale. Cependant, ils seront aptes à partager la bande spectrale avec les utilisateurs primaires à l'aide des fonctionnalités supplémentaires dont ils sont équipés.

- **Station de base des URC :** nommée aussi station de base non licenciée, est une architecture immobile dotées de capacités cognitives. L'URC se connecte à la station de base pour accéder à d'autre réseaux ou services.
- **Serveur spectral :** Le serveur spectral fait partie du RRC, il est utilisé pour partager les ressources spectrales entre plusieurs URC dans le même réseau. Ce serveur est relié à chaque réseau secondaire et opère comme un gestionnaire d'information spectrale.

1.8.2 Fonctions des réseaux de radio cognitive

Les principales fonctions des réseaux de radio cognitive sont :

1.8.2.1 Détection du spectre (Spectrum sensing)

La détection du spectre permet au réseau de radio cognitive de réunir les informations sur l'état d'utilisation des bandes spectrales. Le RRC doit donc détecter les bandes utilisées par les PU et les SU ainsi que les trous du spectre avant de faire l'attribution des bandes du spectre aux SU. En effet, le RRC doit repérer les informations concernant la transmission des PU et des SU pour mesurer l'interférence qu'ils peuvent accepter et s'ils peuvent coexister avec les autres utilisateurs secondaires, puisque le RRC assure un accès dynamique au spectre pour les SU, il doit aussi détecter les trous du spectre ou les bandes du spectre non utilisées afin de les utiliser dans la prochaine allocation. Les trous du spectre peuvent provenir des bandes sous licence appartenant aux utilisateurs primaires qui lâchent dans certain cas leurs bandes et donc le RRC pourra les affecter aux utilisateurs secondaires. Ils peuvent aussi provenir des bandes non soumises à des licences qui était utilisée par les utilisateurs secondaires qui les ont lâchés pour utiliser des bandes plus favorables en termes de qualité de service.

1.8.2.2 Partage du spectre (Spectrum sharing)

Cette fonction existe pour coordonner les transmissions entre les utilisateurs secondaires pour accéder au spectre.

1.8.2.3 Gestion du spectre (Spectrum management)

Les fonctions de gestion du spectre sont indispensables pour les radios cognitives afin de prendre la meilleure bande de spectre pour répondre aux exigences de qualité de service sur toutes les bandes de fréquences libres.

Ces fonctions de gestion peuvent être classées comme suit :

- **Analyse du spectre :** c'est le moyen d'analyser l'activité sur le spectre de fréquences, pour assurer que la transmission d'un utilisateur licencié PU n'est pas brouillée si le secondaire SU décide d'accéder pour pouvoir estimer la qualité du spectre, plusieurs techniques sont employées pour analyser le spectre, comme les algorithmes d'apprentissages de l'intelligence artificielle.
- **Décision sur le spectre :** Cette fonction est indispensable pour l'accès au spectre, elle dépend des résultats retenus par la phase d'analyse du spectre, plusieurs règles décisionnelles sont appliquées afin de déterminer la ou les bandes les plus adaptées à la transmission en cours.

1.8.2.4 Mobilité du spectre (Spectrum mobility)

C'est le processus qui permet à l'utilisateur de la radio cognitive de modifier sa fréquence de fonctionnement. L'utilisation du spectre d'une façon dynamique permet à des terminaux radio de fonctionner dans la meilleure bande de fréquence disponible.

La figure ci-dessous résume les différentes fonctions du RRC :

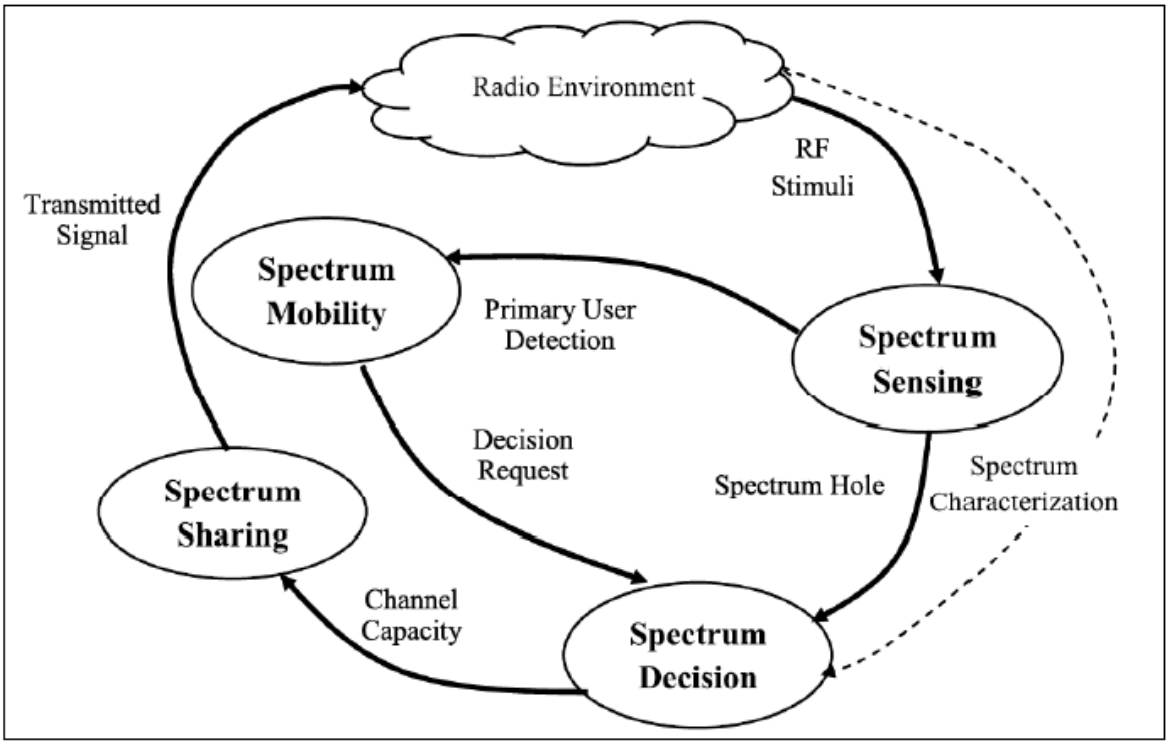


Figure 1.3 : les fonctions de RRC **[9]**

1.8.3 Domaines d'application des réseaux de radio cognitive

La radio cognitive a touché plusieurs domaines, parmi ces derniers on peut citer :

- **Réseaux d'urgence :** ils ont tiré profit des concepts de la radio cognitive pour garantir la fiabilité et la flexibilité de communication sans fil.
- **Réseaux sans fil de la prochaine génération :** La radio cognitive est la technologie clé qui pourrait résoudre le problème d'encombrement du spectre électromagnétique.
- **Services de cyber santé :** le concept de la radio cognitive peut s'employé dans certains services médicaux comme la gestion des soins de santé et la prise en charge des patients, ou bien les appareils.
- **Coexistence de différentes technologies sans fil :** la radio cognitive assure la coexistence de plusieurs technologies sans fil.

1.9 Conclusion

A travers ce chapitre, nous avons pu introduire une présentation précise et détaillée de la radio cognitive dans ses composantes et ses fonctionnalités, ainsi que sa frontière avec la radio logicielle. Les possibilités et éventualités assurés par cette technologie ainsi les champs d'action à exploiter et faire valoir pour la réalisation du concept de réseaux de radio cognitive, ont été démontrés.

Le chapitre suivant sera consacré à la présentation des différentes méthodes de résolution qui seront utilisées dans le cadre de notre projet.

Chapitre 2 :

Méthodes de résolution

2.1 Introduction

En ingénierie, les problèmes d'optimisation combinatoire consistent à sélectionner la meilleure méthode de résolution, il existe deux classes de méthodes pour résoudre ces problèmes.

Les méthodes exactes : ce sont les méthodes qui garantissent l'optimalité de la solution obtenue. Si la taille des données est importante, les méthodes exactes ne permettant pas d'avoir un temps de calcule raisonnable.

Les méthodes approchées : ne garantissent pas que la solution qu'elles fournissent soit optimale, elles sont fréquemment utilisées pour résoudre de problèmes qui sont souvent trop difficiles pour être résolus avec des méthodes exactes.

Dans ce chapitre, nous présentons en particulier la programmation dynamique pour les méthodes exactes et Cuckoo search et Firefly pour les méthodes approchées.

2.2 Les méthodes exactes

Les méthodes exactes sont également connues sous le nom de méthodes complètes, explorent l'espace de recherche de manière exhaustive et permettent de trouver la solution exacte d'un problème d'optimisation combinatoire, ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité [10]. Parmi ces méthodes, on trouve le branch-and-cut (B&C) [11], le branch-and-bound (B&B) [12], le branch-and-price (B&P) [13], branch-and-cut-and-price (B&C&P) [14] et la programmation dynamique [15].

Dans le cadre de notre projet nous avons utilisé la programmation dynamique comme méthode de résolution exacte.

2.2.1 La programmation dynamique

La programmation dynamique a été initiée par Bellman en 1957 [15], elle s'appuie sur un principe simple, appelé le principe d'optimalité de Bellman : « Une politique optimale est formée de sous-politiques optimales ». Le mot politique désigne une séquence de décisions définissant une solution d'un problème d'optimisation. Concrètement, la programmation dynamique est une méthode utilisée pour résoudre des problèmes où une séquence de

décisions optimale doit être trouvée. L'idée de base est que l'on peut déduire une ou la solution optimale d'un problème en combinant des solutions optimales d'une série de sous-problèmes consistant à choisir des séquences plus courtes de décisions. Les solutions des problèmes sont calculées de manière ascendante, c'est-à-dire qu'on débute par les solutions des sous-problèmes les plus petits pour ensuite déduire progressivement les solutions de l'ensemble [16].

La programmation dynamique est une approche d'optimisation qui transforme un problème complexe en une séquence de problèmes plus simples, sa caractéristique essentielle est la nature à plusieurs stades de la procédure d'optimisation. En plus des techniques d'optimisation décrites précédemment, la programmation dynamique fournit un cadre général pour analyser de nombreux types de problèmes. Dans ce cadre, une variété de techniques d'optimisation peut être utilisée pour résoudre des aspects particuliers d'une formulation plus générale. Habituellement, la créativité est nécessaire avant de pouvoir reconnaître qu'un problème particulier peut être efficacement lancé en tant que programme dynamique, et souvent des idées subtiles sont nécessaires pour restructurer la formulation afin qu'elle puisse être résolue efficacement.

La programmation dynamique est connue pour résoudre les problèmes les plus répandus comme le problème de sac à dos, la tour de Hanoi, le chemin le plus court par Dijkstra.

La programmation dynamique peut être utilisée à la fois de haut en bas et de bas en haut. Et bien sûr, la plupart du temps, en se référant à la sortie de la solution précédente, il est moins coûteux que de recalculer en termes de cycles CPU.

Dans le cadre de notre projet nous avons parallélisé un traitement basé sur la programmation dynamique en utilisant les threads.

2.3 Les méthodes approchées

2.3.1 Les heuristiques

Le terme heuristique est utilisé pour des algorithmes qui trouvent des solutions parmi celles réalisables, mais ils ne garantissent pas que le meilleur sera trouvé, donc ils peuvent être considérés comme des algorithmes approximatifs et non précis. Ces algorithmes trouvent généralement une solution proche du meilleur rapidement et facilement. Parfois, ces algorithmes peuvent être précis, c'est-à-dire qu'ils trouvent la meilleure solution.

La méthode utilisée à partir d'un algorithme heuristique est l'une des méthodes connues, mais pour être facile et rapide, l'algorithme ignore ou même supprime certaines des demandes du problème.

2.3.2 Les métaheuristiques

Les métaheuristiques sont des techniques d'optimisation qui conviennent parfaitement pour résoudre les problèmes logistiques. Il existe un grand nombre de métaheuristiques différentes, parmi elles les méthodes individuelles et les méthodes basées sur la population.

Les techniques individuelles visent à se déplacer dans l'espace de recherche en s'appuyant sur un système de quartier qui permet aux gens de construire un ensemble de solutions voisines à partir d'une solution actuelle. Il existe plusieurs méthodes d'optimisation individuelles comme la recherche locale [17], recuit simulé [18], algorithme kangourou [19], recherche local itérée et recherche Tabou [20].

L'objectif des méthodes basées sur la population est d'initier une interaction entre les individus qui forment une population afin de générer de nouvelles solutions. Il existe plusieurs méthodes basées sur la population comme les algorithmes évolutifs [21], l'algorithme de la colonisation des fourmis [22] et l'optimisation des essaims de particules [23].

- **Les algorithmes à base d'intelligence par essaim**

En intelligence artificielle (IA), l'intelligence en essaim fait référence à cette intelligence collective des sociétés d'animaux. Elle a fait l'objet de nombreux travaux de recherche, ce type d'algorithme se repose entre autres sur un modèle permettant de simuler le déplacement d'un groupe d'individus (oiseaux, insectes…). Cette méthode d'optimisation s'appuie sur la collaboration des individus entre eux. Cette idée veut qu'un groupe d'individu peu intelligent puisse posséder une organisation globale complexe. De même, grâce à des règles de déplacement très simples dans l'espace des solutions, les parties peuvent converger progressivement vers un minimum local. Cette métaheuristique donne l'aspect de mieux fonctionner pour des espaces en variables continues.

Au départ de l'algorithme, chaque partie est donc positionnée aléatoirement ou non dans l'espace de recherche du problème. Chaque itération fait bouger les parties en fonction de trois composantes : sa meilleure solution, sa vitesse de déplacement ainsi que la meilleure solution obtenue dans son voisinage.

Dans le cadre de notre projet nous avons étudié deux métaheuristique à base d'intelligence par essaim, le cuckoo search (CS) et l'algorithme des lucioles (FA).

2.3.2.1 Cuckoo search

- **Inspiration**

Cuckoo sont des oiseaux éblouissant, non seulement à cause des beaux sons qu'ils peuvent faire, mais aussi à cause de leur stratégie de reproduction agressive. Certaines espèces comme les ani et Guira cuckoos pondent leurs œufs dans des nids communautaires, mais ils peuvent retirer les œufs des autres pour augmenter la probabilité d'éclosion de leurs œufs. Un certain nombre d'espèces engage le parasitisme obligatoire en posant leurs œufs dans les nids d'autres oiseaux d'accueil (souvent d'autres espèces). Certains oiseaux hôtes peuvent engager un conflit direct avec les cuckoos intrus. Si un oiseau hôte découvre que les œufs ne sont pas sa propriété, il va soit jeter ces œufs exotiques loin ou tout simplement abandonner son nid et construit un nouveau nid ailleurs.

Certaines espèces des cuckoo telles que le couvain-parasitaire Tapera ont évolué de telle manière que les cukoos parasites femelles sont souvent très spécialisés dans la mimique en couleur et motif des œufs de quelques espèces hôtes choisies. Ceci réduit la probabilité de leurs œufs étant abandonné et augmente ainsi leur reproductivité. De plus, le moment de la ponte de certaines espèces est également étonnant. Parasite cuckoos choisissent souvent un nid où l'oiseau hôte viens juste de poser ses propres œufs. En général, les œufs de cuckoo éclosent un peu plus tôt que les œufs des hôtes. Une fois que le premier poussin cuckoo est hachurée, la première action de l'instinct est qu'il va expulser les œufs hôtes aveuglément en les propulsant hors du nid, ce qui augmente la part du poussin cuckoo de nourriture fournie par son oiseau hôte. Les études montrent également qu'un poussin cuckoo peut aussi imiter l'appel de poussins d'accueil pour avoir accès à plus de possibilités d'alimentation. [24]

- **Lévy flight**

Dans la nature, la recherche des animaux pour la nourriture est d'une manière aléatoire ou quasi-aléatoire. En général, le chemin de recherche de nourriture d'un animal est véritablement une marche aléatoire car le prochain mouvement est basé sur la localisation (état actuel), et la probabilité de transition au prochain emplacement. Quelle direction il choisit dépend implicitement sur une probabilité qui peut être modélisée mathématiquement.

Par exemple, diverses études ont montré que le comportement vol de nombreux animaux et insectes a démontré les caractéristiques typiques des vols LEVY. [25]

Le terme « Lévy Flight » a été inventé par Benoît Mandelbrot [26], qui a utilisé cela pour une définition précise de la distribution des tailles de pas.

Le Lévy flight, est une marche aléatoire dans laquelle les étapes sont définies en fonction des longueurs des pas, qui ont une certaine distribution de probabilité, avec les directions des étapes étant isotrope et aléatoire.

Au cours de leurs trajectoires en air mort, les mouches des fruits (Drosophila melanogaster) explorent leur paysage à l'aide d'une série de voies de vol droites ponctuées par des saccades rapides à 90 ° [27]. Certaines saccades sont déclenchées par une expansion visuelle associée à l'évitement des collisions. Pourtant, de nombreuses saccades ne sont pas déclenchées par des repères visuels, mais plutôt apparaissent spontanément conduisant au style irrégulier du vol LEVY. [28]

- **Description de l'algorithme**

Pour plus de clarté dans la description de l'algorithme cuckoo search, trois règles idéalisées sont utilisées :

- Chaque cuckoo pond un œuf à la fois, et les déverse dans un nid choisi au hasard.
- Les meilleurs nids de haute qualité des œufs (solutions) seront reportés aux prochaines générations.
- Le nombre de nids disponibles d'accueil est fixe, et un hôte peut découvrir un œuf étrangé avec une probabilité pa $\in$ [0, 1]. Dans ce cas, l'oiseau hôte peut soit lancer l'œuf loin ou abandonner le nid afin de construire un nouveau nid dans un nouvel emplacement. Par souci de simplicité, cette dernière hypothèse peut être approchée par une fraction pa des n nids étant remplacés par de nouveaux nids (avec de nouvelles solutions aléatoires à de nouveaux emplacements), dans notre projet nous avons utilisé pa=10% de la taille de la population.

Pour un problème de maximisation, la qualité ou la fonction fitness d'une solution peut tout simplement être proportionnel à la fonction objectif.

Sur la base de ces trois règles, les étapes de base de la recherche cuckoo peuvent être résumées comme indiqué dans le pseudo-code ci-dessous [24]:

Tant que (t <NbrMaxGéneration) ou (critère d'arrêt) **faire**
Créer un cuckoo (nommé i) aléatoirement par Levy Flight
Evaluer sa qualité/fitness Fi
Choisir un nid parmi n (nommé j) aléatoirement
Si (Fi > Fj) **alors**
Remplacer j par la nouvelle solution
Fin Si
Abandonner une fraction (pa) des pires nids et construire des nouveaux à de nouveaux emplacements à travers Levy Flight
Garder les meilleures solutions (ou nids avec des solutions de qualité)
Classer les solutions et trouver le meilleur
Fin tant que

Lors de la génération de nouvelles solutions $\boldsymbol{x_i^{t+1}}$ pour un cuckoo appelé i (solution), un vol LEVY est effectué comme suit :

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

Ou $\boldsymbol{\alpha}$ est la taille de l'étape.

Le vol Lévy fournit une marche aléatoire tel que ses étapes aléatoires sont tirés de la distribution Levy avec grand pas. $\boldsymbol{Levy \sim u = t^{-\lambda}}$, avec $1< \lambda <3$.

2.3.2.2 L'algorithme Firefly

- **Inspiration**

Les lucioles (en anglais FireFly) sont de petits coléoptères ailés capables de produire une lumière clignotante. Il existe environ deux mille espèces de lucioles, et la plupart des lucioles produisent des flashs courts et rythmés. Le motif des flashes est souvent unique pour une espèce particulière. La lumière clignotante est produite par un processus de bioluminescence, et les vraies fonctions de ces systèmes de signalisation continuent de débattre.

Ces insectes sont capables de produire de la lumière à l'intérieur de leur corps grâce à des organes spéciaux situés très près de la surface de la peau. Cette production de lumière est due à un type de réaction chimique appelée bioluminescence. [29]

Les femelles peuvent imiter les signaux lumineux des autres espèces afin d'attirer des mâles afin de les capturer et les dévorer. Les lucioles ont un mécanisme de type condensateur, qui se décharge lentement jusqu'à ce qu'un certain seuil soit atteint, ils libèrent l'énergie sous forme de lumière. Le phénomène se répète de façon cyclique.

L'Algorithme des lucioles développé par Yang [30] est inspiré par l'atténuation de la lumière sur la distance et l'attraction mutuelle mais il considère toutes les lucioles comme unisexes.

- **Principe de fonctionnement**

L'algorithme Firefly (FA) développé par Xin-She Yang à l'Université de Cambridge en 2007 est basé sur l'émission de la lumière et l'absorption de la lumière ainsi que le comportement attractif entre les lucioles, il est développé pour résoudre les problèmes d'optimisation.

L'algorithme tire son inspiration du comportement clignotant des lucioles. Les trois règles sont idéalisées en décrivant le FA [30] [31] :

- Toutes les lucioles sont unisexes de telles sortes que seront attirées vers d'autres indépendamment de leurs sexes.
- L'attractivité des lucioles est directement proportionnelle à la luminosité entre eux, et est diminué une fois la distance entre deux lucioles augmente. Ainsi, pour tous les deux lucioles clignotantes quelconques, le moins lumineux sera attiré et se déplace donc vers le plus lumineux.
- La luminosité de la lumière clignotante peut être considérée comme une fonction objectif qui devra être optimisée.

Le principe de base de la plupart des méthodes d'optimisation des métaheuristiques est la génération aléatoire de la population initiale de solutions candidates réalisables. Pour cela le processus de l'algorithme de luciole commence avec l'initialisation de la population des lucioles et donc chaque luciole dans une population représente une solution candidate, La taille de la population détermine le nombre de solutions ou la taille de l'espace de recherche dont le but est d'orienter la recherche à la meilleure localisation.

Dans l'étape suivante, chaque luciole est évaluée en fonction de leurs conditions physiques (intensité lumineuse). À chaque nouvelle étape itérative, la luminosité et l'attirance de chaque luciole est calculée, la distance entre toutes lucioles peut être définie comme une distance cartésienne, la fonction de la distance développée est utilisée pour trouver la distance entre deux lucioles.

La fonction de l'attractivité est définie en utilisant l'intensité lumineuse, la distance entre lucioles, et un coefficient d'absorption, le mouvement de luciole est défini par une fonction de mouvement, en utilisant la position actuelle, l'attractivité et une marche aléatoire, après avoir comparé la luminosité de chaque luciole avec toutes les autres lucioles, les positions des

lucioles sont mises à jour en se basant sur les règles de connaissances sur les lucioles et sur leurs voisins. Après le déplacement, la nouvelle luciole est évaluée et l'intensité de sa lumière est mise à jour. Pendant la boucle de comparaison en paire, la meilleure solution actuelle est la mise à jour d'une manière itérative. Le Processus de comparaison par pair est répété jusqu'à la satisfaction des critères de terminaison.

L'algorithme des lucioles est donné ci-dessous :

Fonction objectif f(x), x = $(x_1, \dots, x_d)^T$

Générer la population initiale de lucioles x_i (i =1,2, …, n)

L'intensité lumineuse I_i de x_i est déterminée par f(x_i)

Définir le coefficient d'absorption de la lumière γ

Tant que (t < MaxGeneration)

Pour i = 1 jusqu'à n (n tt les lucioles)

Pour j = 1 jusqu'à n (n tt les lucioles) (boucle interne)

Si ($I_i < I_j$), deplacer la luciole i vers j ; **finsi**

Varier l'attractivité par la distance r via **exp[-γr]**

Evaluer les nouvelles solutions et mettre à jour l'intensité de la lumière

Fin pour

Fin pour

Classer les lucioles et trouver la meilleure solution actuelle

Fin tantque

Extraction de la solution.

2.4 Conclusion

A travers ce chapitre, nous avons présenté les méthodes de résolution que nous allons utiliser durant la réalisation de notre PFE, nous nous sommes focalisés sur une méthode de résolution exacte qui est la programmation dynamique en donnant son principe, ensuite nous avons expliqué les méthodes de résolution approchée précisément les métaheuristiques en donnant une définition claire et précise des deux algorithmes qui seront utilisés par la suite (le Firefly et le Cuckoo search).

Le chapitre suivant sera consacré à la présentation de notre application ainsi que les résultats obtenus lors de notre étude comparative entre la parallélisation de la programmation dynamique et l'utilisation du Firefly et du Cuckoo search.

Chapitre 3 :

Implémentation de l’application et évaluation des résultats

3.1 Introduction

Dans le domaine de la radio cognitive, l'utilisateur primaire (PU) qui est en possession d'une licence est libre d'accéder à n'importe quel moment au spectre via ses bandes de fréquence, contrairement à l'utilisateur secondaire (SU) qui pourra accéder seulement a des bandes de fréquence non utilisées par le (PU), selon le critère de coopération entre les deux sans produire des interférences.

Un moteur cognitif utilise généralement les métaheuristiques pour son fonctionnement, ainsi, nous avons réalisé dans le cadre de ce PFE une étude comparative entre deux métaheuristiques qui sont Cuckoo search et Firefly et une méthode exacte intitulée la programmation dynamique parallèle. L'objectif visé est de montrer l'intérêt de paralléliser une méthode exacte afin d'améliorer la gestion du spectre dans le cadre d'un réseau de radio cognitive.

Nous allons donc, dans ce qui suit présenter notre application avec ces différentes interfaces et ces simulations graphiques pour pouvoir détailler notre étude comparative et donner les résultats obtenus lors de la réalisation de ce travail.

3.2 La fonction objectif (fitness function)

La fonction objectif est une fonction qui sert à déterminer la meilleure solution a un problème d'optimisation combinatoire qui est soit de type maximiser ou de type minimiser jusqu'à l'obtention de la solution la plus favorable. Dans notre cas l'utilité de cette fonction est de maximiser le coût « bénéfice pour l'utilisateur primaire » et aussi le nombre de (SU) satisfait et de minimiser le temps de traitement « temps de réponse aux utilisateurs secondaire ».

Soit les paramètres suivants :

- n : le nombre de (SU).
- m : le nombre de canaux libres que possède le (PU).
- W : un tableau de taille n. W[i] est le nombre de canaux demandés par SUi.
- C : un tableau de taille n. C[i] est le prix proposé pour les W[i] par SUi.
- La fonction à optimiser est : Max $\sum_{i=0}^{n-1} C[i]$.
- La contrainte à respecter est : $\sum_{i=0}^{n-1} W[i] \leq$ m.

3.3 Réalisation des algorithmes et résultats

3.3.1 La programmation dynamique

Notre étude s'est basée sur le travail proposé par les auteurs dans [32], travail réalisé avec la programmation dynamique mais sans les threads.

Nous avons donc visé en premier lieu l'amélioration de la programmation dynamique en se basant sur le parallélisme. Notre traitement s'effectue sur une matrice de taille n*m d'entier, le code s'exécute ligne par ligne et le résultat de la ligne suivante dépend de celui de la ligne précédente.

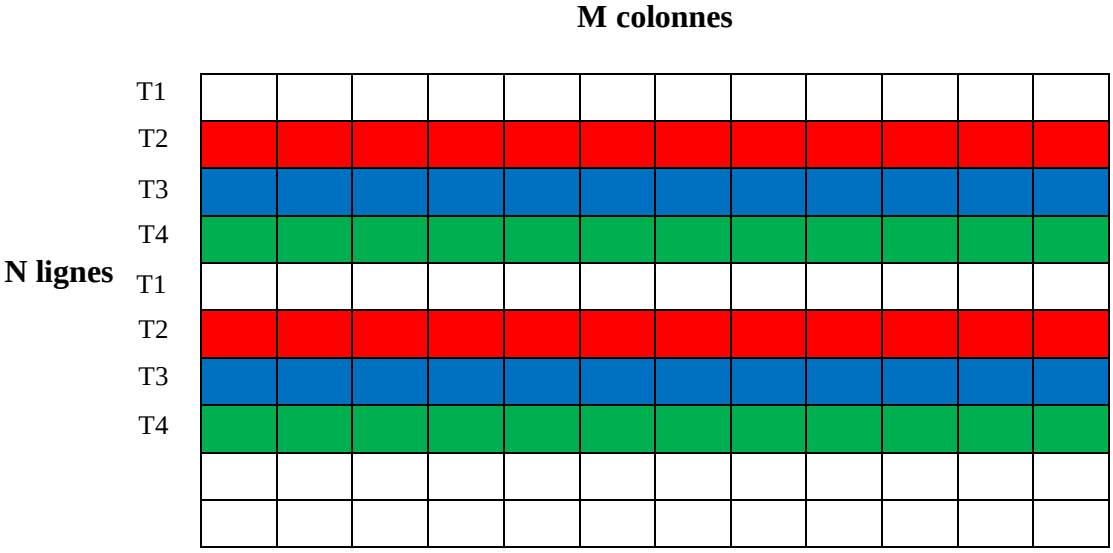


Figure 3.1 : matrice de la programmation dynamique

Comme nous l'avons déjà expliqué, la programmation dynamique est une approche d'optimisation qui transforme un problème complexe en une séquence de problèmes plus simples. Sa caractéristique essentielle est la nature à plusieurs stades de la procédure d'optimisation. Dans le cas de 4 threads par exemple (voir figure 3.1), notre idée est d'exécuter le premier thread sur la première ligne de la matrice, le deuxième thread sur la deuxième ligne, le troisième thread sur la troisième ligne, etc. Mais comme le traitement du second thread dépend du résultat du premier thread et le traitement du troisième thread dépend du résultat du second thread et ainsi de suite. Le deuxième thread sera lancé sur la deuxième ligne lorsque le premier thread a traité les cellules m / 4 de la première ligne de la matrice (4 ici est le nombre de threads). Le troisième thread sera lancé sur la troisième ligne lorsque le premier thread a traité 2 * m / 4 cellules de la première ligne. Le quatrième thread sera lancé

sur la quatrième ligne lorsque le premier thread a traité 3 * m / 4 cellules de la première ligne. Ensuite, le premier thread continuera à fonctionner sur la 5ème ligne. Le deuxième thread continuera à fonctionner sur la 6ème ligne, le troisième thread sur la 7ème ligne et le 4ème thread sur la 8ème ligne et ainsi de suite jusqu'à ce que toute la matrice soit traitée.

A noter ici que la machine utilisée pour réaliser ces simulations possède les caractéristiques suivantes : 2 cpu xeon avec 16 cœurs et 32 threads et une mémoire vive de 112 GB ddr4.

Les paramètres utilisés sont comme suit :

n= 12000.

m=15000.

Les formules des tableaux W et C sont :

W[i]= i / 10 + 10 pour i allant de 0 à n.

C[i] = (i * 2) + 5 pour i allant de 0 à n.

Ces paramètres seront toujours utilisés dans les autres algorithmes pour effectuer correctement l'étude comparative.

Le tableau suivant représente les résultats obtenus en termes de temps d'exécution avec 2, 4, 8, 16 et 32 threads respectivement. Une moyenne de 10 exécutions a été considérée systématiquement pour chaque nombre de thread.

Nombre de threads	1	2	4	8	16	32
Temps d'exécution (en ms)	3645	2862	1564	1100	700	700

Tableau 3.1 : temps d'exécution en fonction du nombre de threads

Le graphe suivant montre les résultats obtenus.

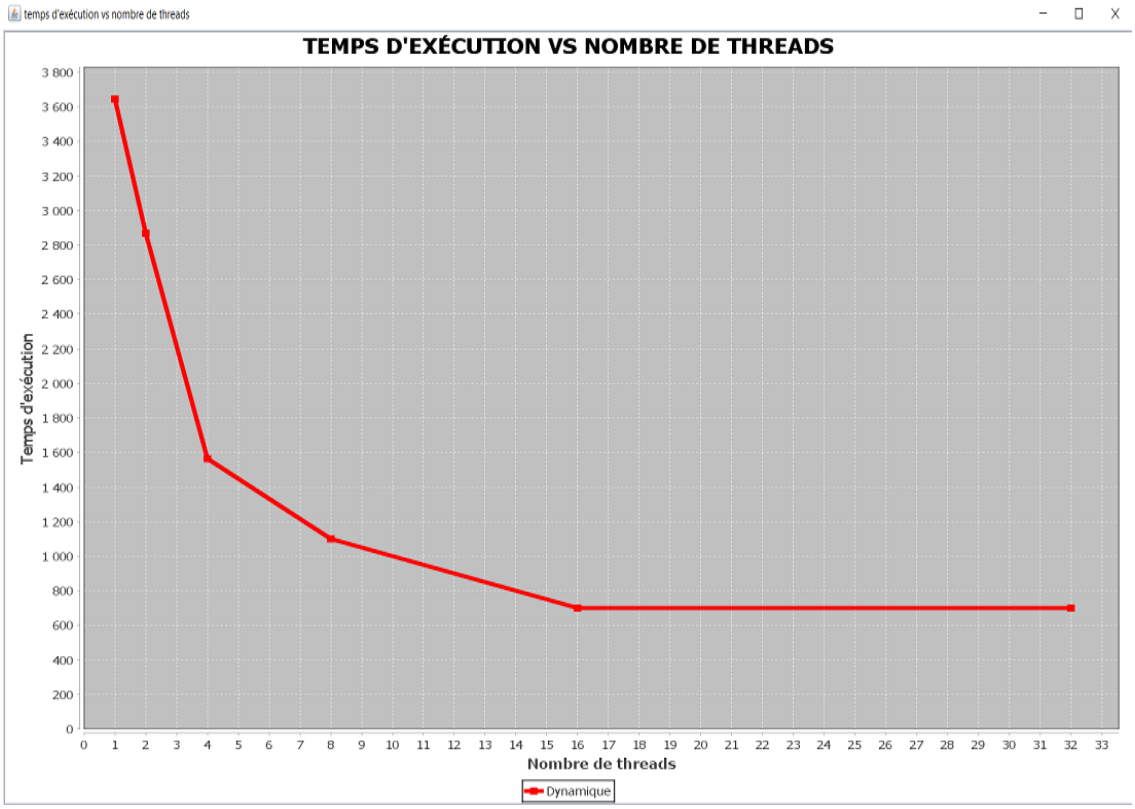


Figure 3.2 : temps d'exécutions par rapport au nombre de threads

Nous remarquons qu'en augmentant le nombre de threads le temps de traitements sera réduit et ça jusqu'à l'utilisation de 16 threads. Une fois le nombre 16 est atteint le temps va plus au moins se stabiliser jusqu'à l'utilisation de 32 threads. Mais à partir de 32 threads le temps de traitement va augmenter à cause de la communication entre les threads.

A noter ici que le coût obtenu par la programmation dynamique et le nombre de SU satisfaits sont fixes peu importe le nombre de threads (voir le tableau 3.2).

Coût	Nombre de SU satisfaits
297699	13

Tableau 3.2 : coût et nombre de SU satisfaits obtenus par la programmation dynamique

3.3.2 L'algorithme cuckoo search

3.3.2.1 Initialisation de la population

Après avoir fait plusieurs tests afin d'avoir de meilleures performances, nous avons opté pour une population initiale de 100 individus. Chaque individu de notre population représente une ligne de la matrice, cette dernière est une combinaison de 0 et 1.

Le 1 veut dire que le SU fait partie de la solution.

3.3.2.2 Génération de nouvelles solutions

Après avoir générer la population initiale, l'algorithme calcule pour chaque ligne de la matrice sa fonction objectif, le résultat obtenu est sauvegardé dans un tableau trié par ordre croissant, le meilleur individu de cette population est celui qui a le plus de gain. Ensuite, un nouvel individu sera créé aléatoirement par Levy flight, puis il sera comparé à un autre individu créé aussi aléatoirement. Si la fonction objectif de l'individu créé par Levy flight est supérieure à celle de l'individu créé aléatoirement on remplace ce dernier, sinon il n y'aura pas de changement. L'algorithme néglige une fraction de pires individus, cette fraction représente 10% de la population, ces pires individus seront remplacés par Levy flight.

3.3.2.3 Critère d'arrêt

Nous avons choisi de fixer un certain nombre de génération comme critère d'arrêt de notre algorithme.

Le tableau suivant représente les résultats obtenus par l'algorithme CS :

Coût	Temps d'exécution en ms	Nombre de SU satisfaits
295637	1696	22

Tableau 3.3 : résultats de l'algorithme Cuckoo search

3.3.3 L'algorithme Firefly

3.3.3.1 Génération de la population initiale

L'algorithme des lucioles « Firefly algorithm » génère une population initiale qui est une combinaison de solutions possibles, cette population est initialisée aléatoirement.

Dans une situation où l'on ne connait rien sur le problème à résoudre, il est important que la population initiale soit répartie sur tout le domaine de recherche.

Par contre le choix d'une population avec une taille trop grande peut augmenter considérablement le temps de calcul sachant que le temps est un critère primordial dans notre projet, et si la taille de la population est trop petite, il y aura une convergence rapide car l'algorithme n'a pas un grand aperçu sur l'espace de recherche, dans notre projet la taille de la population qu'on a pu choisir afin de parvenir à de meilleurs résultats est 100.

3.3.3.2 Classement

Le classement se fait par rapport à la fonction objectif, de ce fait dans notre algorithme le classement se fait selon l'intensité de la lumière de chaque luciole. Ce classement sert à déterminer le meilleur individu ou le pire individu, dans notre cas la fonction objectif cherche à maximiser les critères donc le classement se fait par ordre croissant ce qui donne que le meilleur est le maximum du classement.

3.3.3.3 Déplacement et mise à jour

À chaque nouvelle étape itérative de l'algorithme, la luminosité et l'attirance (atrractiveness) de chaque luciole (firefly) est calculée via la fonction objectif, ensuite l'algorithme compare chaque luciole avec tout le reste des lucioles, comparaison deux à deux selon la fonction objectif. Si la fonction objectif de la i-ème luciole est inférieure à la fonction objectif de la j-ème luciole I(i) < I(j) ce qui est exprimé par l'attirance, la première se déplacera vers la deuxième.

Les positions des lucioles sont mises à jour après avoir comparé la luminosité de chaque luciole avec toutes les autres lucioles et aussi en prenant compte des règles de connaissance de la luciole et de leurs voisins, ces règles sont la distance entre deux lucioles comparées, un mouvement aléatoire et la position initiale.

Par la suite, la nouvelle luciole est évaluée, sa position et son intensité de lumière sont mises à jour.

3.3.3.4 Critère d'arrêt

Pareil comme pour le CS, nous avons choisi de fixer un certain nombre de génération comme critère d'arrêt de notre algorithme.

3.3.3.5 Paramétrage de l'algorithme firefly

La mise en place de l'algorithme des lucioles n'est pas si compliqué contrairement à son paramétrage car le choix des valeurs est une étape assez difficile, tout d'abord car ces paramètres dépendent du type de problème à résoudre, la plupart du temps les valeurs de ces paramètres sont configurées en fonction des résultats obtenus lors d'une séquence d'exécution de l'algorithme.

Dans notre algorithme Firefly nous avons pris en compte différents paramètres comme la taille de la population initiale ainsi que l'attraction de la luciole qui est en relation avec l'intensité de la lumière, la distance aussi car c'est un paramètre très important et enfin le coefficient d'absorption $\gamma \in [0,1]$, car il contrôle la variation de l'attractivité en fonction de la distance entre deux lucioles.

Après avoir fait une dizaine d'exécutions de notre algorithme nous avons pu obtenir nos résultats en termes de temps d'exécution, coût et nombre de SU satisfaits. Le tableau suivant représente les différents résultats obtenus :

Coût	Temps d'exécution en ms	Nombre de SU satisfaits
291154	1925	30

Tableau 3.4 : résultats de l'algorithme Firefly

3.4 Outils utilisés pour l'implémentation de l'application

3.4.1 NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels que le C, le C++, le JavaScript, le XML, le Groovy, le PHP et HTML, il a une base d'utilisateur très large et une communauté en croissance constante [33]. La création de notre application a été faite sous la version 8.1 qui s'enrichit de plusieurs supports comme l'amélioration de plusieurs Frameworks.

3.4.2 Adobe Photoshop

Photoshop est un logiciel de retouche, de traitement d'image développé et fabriqué par Adobe Systems Inc, lancé en 1990 sur MacOs puis en 1992 sur Windows. Photoshop est considéré comme l'un des leaders dans le logiciel de retouche photo. Le logiciel permet aux utilisateurs de manipuler, de recadrer, de redimensionner et de corriger la couleur sur les photos numériques. Le logiciel est particulièrement populaire chez les photographes professionnels et infographistes [34], il est principalement utilisé pour le traitement de photographies numériques, il travaille sur les images matricielles car les images sont constituées d'une grille de points appelée pixels. L'intérêt de ces images est d'améliorer l'aspect visuel de notre application.

3.4.3 JFreeChart

JFreeChart est une API Java permettant de créer des graphiques et des diagrammes de très bonne qualité. Elle possède plusieurs formats dont les barres ou les lignes, le camembert et propose de nombreuses options de configuration pour personnaliser le rendu des graphes. Cette API est open source et sous licence LGPL (Lesser General Public Licence), en Revanche la documentation est payante [35]. On a utilisé cette API pour pouvoir présenter les résultats obtenus lors de la réalisation de notre étude comparative.

3.5 Présentation de l'application

Au lancement de notre application, l'interface d'accueil apparait comme suit :



Figure 3.4 : interface d'accueil de l'application

En cliquant sur le bouton « Lancer », l'interface de choix des méthodes de résolutions s'affiche (figure 3.5).

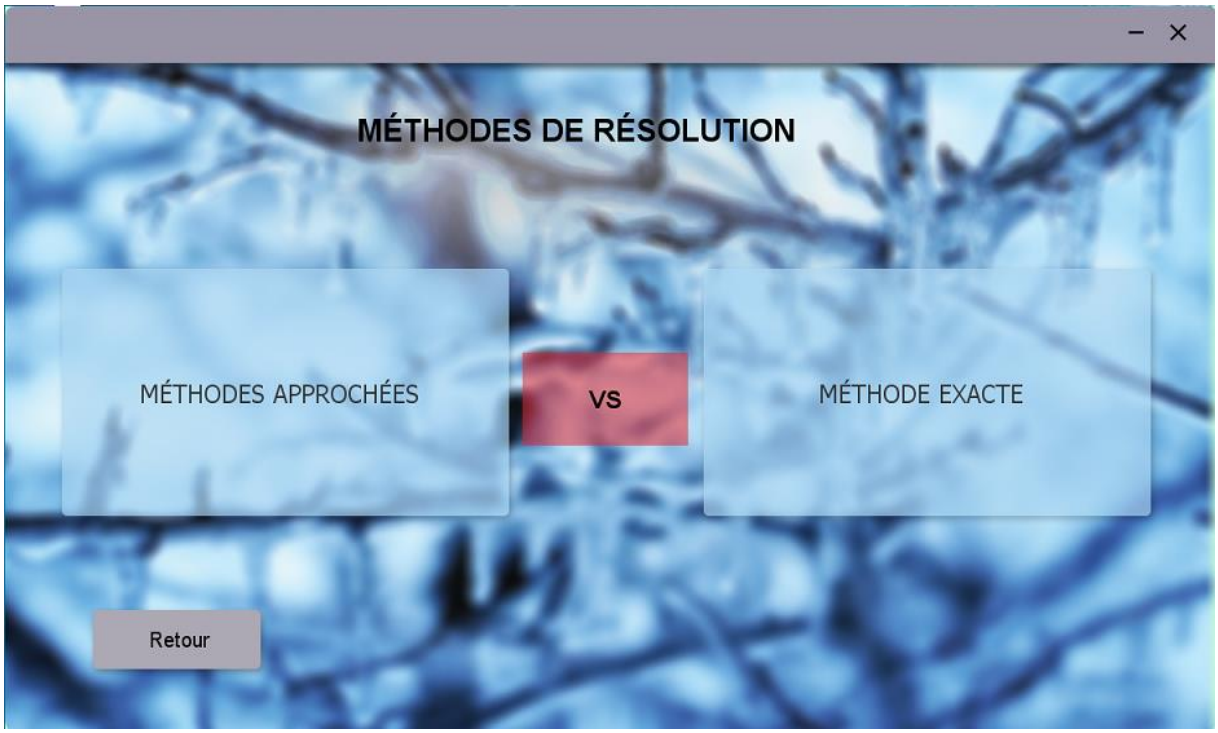


Figure 3.5 : interface choix de l'application

Dans cette interface, nous avons les trois boutons suivants :

- Méthodes approchées : ce bouton permet d'accéder aux algorithmes métaheuristiques.
- Méthode exacte : ce bouton permet d'aboutir à la programmation dynamique.
- VS : le bouton « versus », son rôle est de faire une comparaison entre la méthode exacte et les méthodes approchées en termes de plusieurs critères.

Au pied de chaque interface nous retrouvons le bouton « retour » qui permet à l'utilisateur de retourner à la page précédente.

Après avoir cliqué sur le bouton « VS », on voit s'afficher l'interface suivante :



Figure 3.6 : interface comparaison de l'application

Dans cette interface, l'utilisateur retrouve trois boutons radio, permettant chacune la comparaison entre la méthode exacte et les deux méthodes approchées ;

- Temps d'exécution : le rôle de ce bouton choix est d'afficher un graphe « en lignes », qui élucide cette comparaison en termes de temps d'exécution obtenu.
- SU Satisfaits : en cochant sur ce bouton choix, un graphe « en barre » apparait, qui montre la comparaison en termes de nombre des utilisateurs secondaires satisfaits.
- Coût : ce choix permet d'afficher un graphe « en barre », qui indique la comparaison en termes du coût obtenu par l'utilisateur primaire.

Après avoir sélectionné l'un des boutons radio, un clic sur le bouton « Simuler » fait apparaitre un graphe.

L'interface « programmation dynamique » (figure 3.7), obtenu en cliquant sur les boutons « méthodes exacte » puis « programmation dynamique », le carré « configuration » regroupe deux choix :

- Saisir les valeurs : l'utilisateur peut entrer manuellement les valeurs dans les trois champs à savoir « Nombre de threads » ; « nombre de canaux » ; « nombre de SU »

puis valider en cliquant sur le bouton « Simuler », les résultats sont ainsi affichés sous forme de graphes « en barre ».

- Valeurs par défaut : cette configuration permet de réaliser une simulation avec des valeurs déjà fixés par défaut qui sont « Nombre de threads = 4 » ; « nombre de canaux = 15000 » ; « nombre de SU = 12000 », en donnant un graphe « en barre »

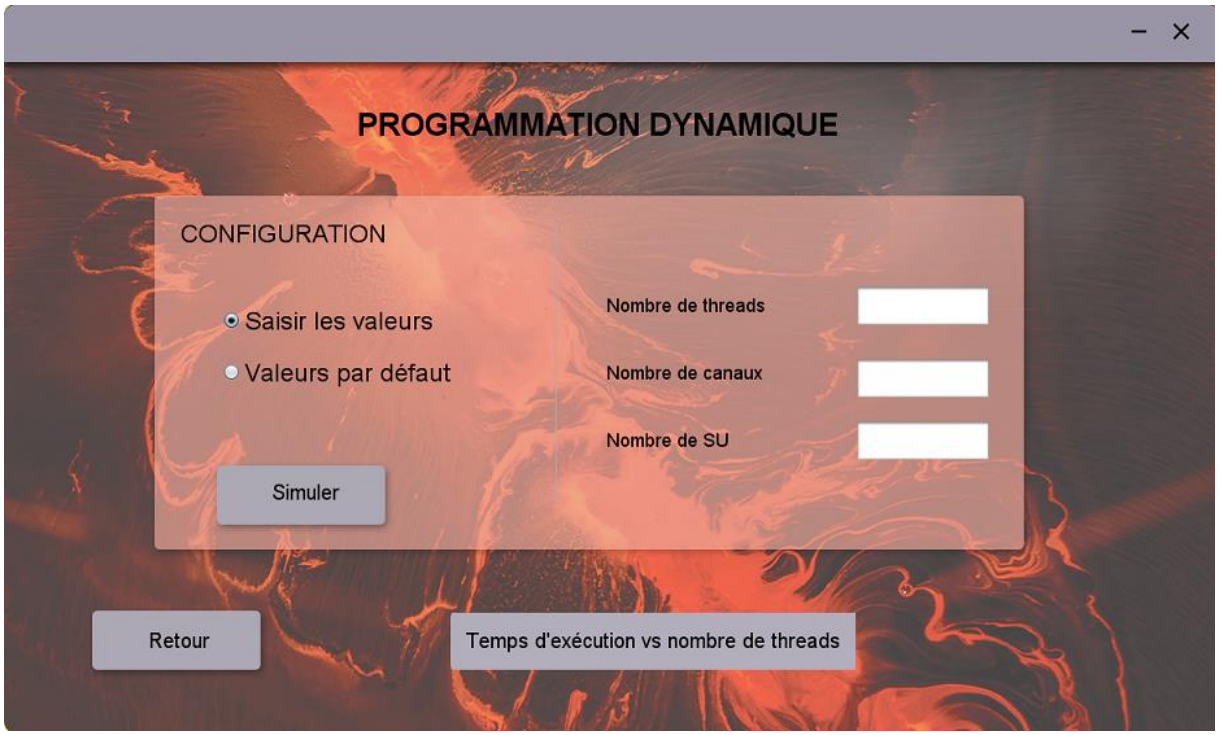


Figure 3.7 : interface programmation dynamique

Par ailleurs, apparait dans l'interface « programmation dynamique », un autre bouton « temps d'exécution vs nombre de threads », qui envisage la relation entre le nombre de threads utilisés et le temps d'exécution obtenu, sous forme d'un graphe « en ligne ».

L'utilisateur voit s'afficher, en cliquant sur le bouton « méthodes approchées », les deux algorithmes métaheuristiques (figure 3.8) :

- Cuckoo search
- Firefly

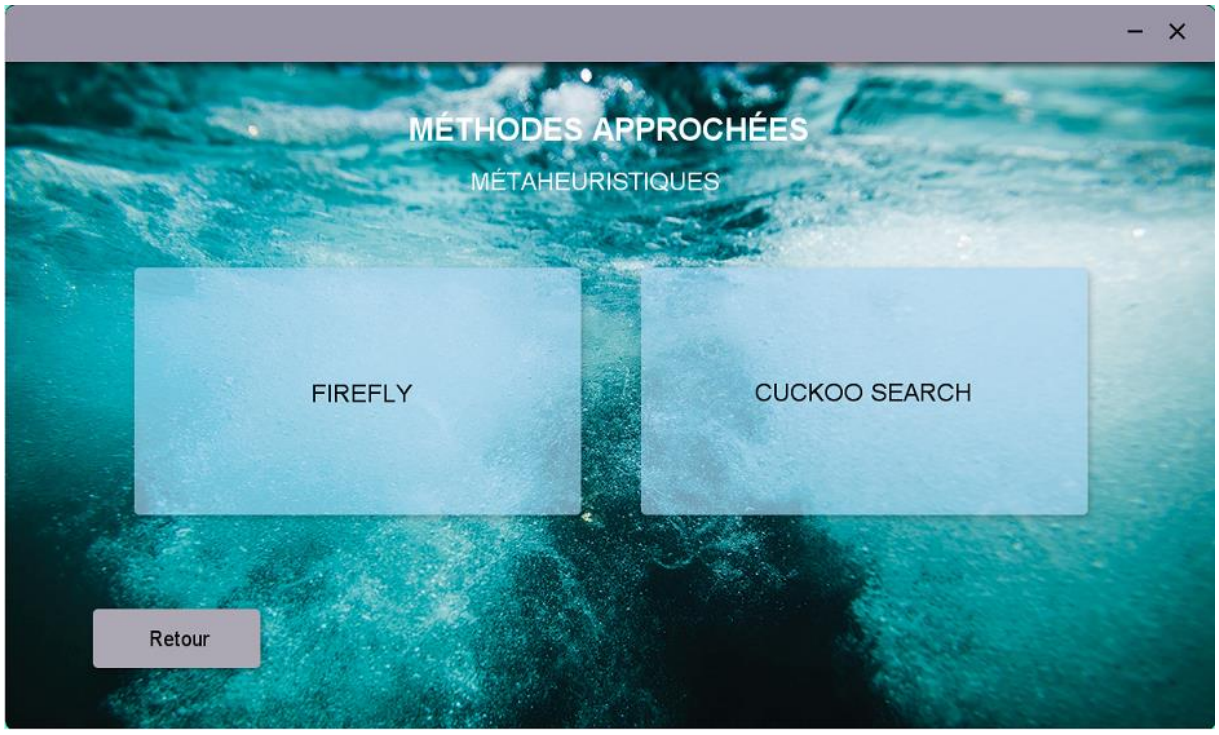


Figure 3.8 : interface choix méthodes approchées

Le principe d'appliquer l'un de ces algorithmes est identique à celui de l'interface de configuration de la programmation dynamique, néanmoins, le champ « nombre de threads » se trouve remplacé par le champs « Taille de la population » (figure 3.9).

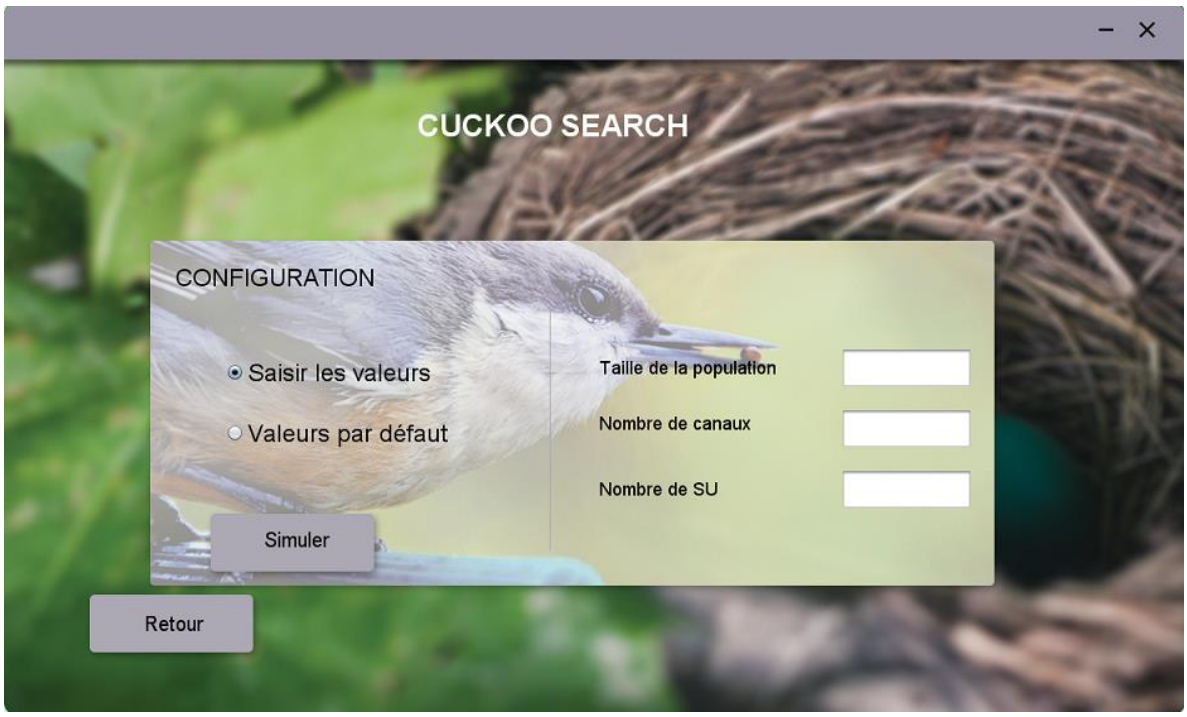


Figure 3.9 : interface configuration algorithme Cuckoo search

3.6 Etude Comparative

Dans cette partie, nous allons élaborer une comparaison entre les résultats obtenus avec l'implémentation de la programmation dynamique ainsi que les deux métaheuristiques Cuckoo search et Firefly. Cette comparaison a été conçue en se basant sur plusieurs critères, qui sont : le temps de traitement côté PU, le coût et le nombre de SU satisfaits.

Après avoir lancé une dizaine de simulation de nos algorithmes avec des paramètres similaires (nombre de SU=12000, nombre de canaux=15000), nous avons fait leur moyenne afin de présenter les résultats obtenus lors de notre étude.

3.6.1 Comparaison en termes de temps de traitement côté PU

Le graphe et le tableau suivent représente les résultats obtenus.

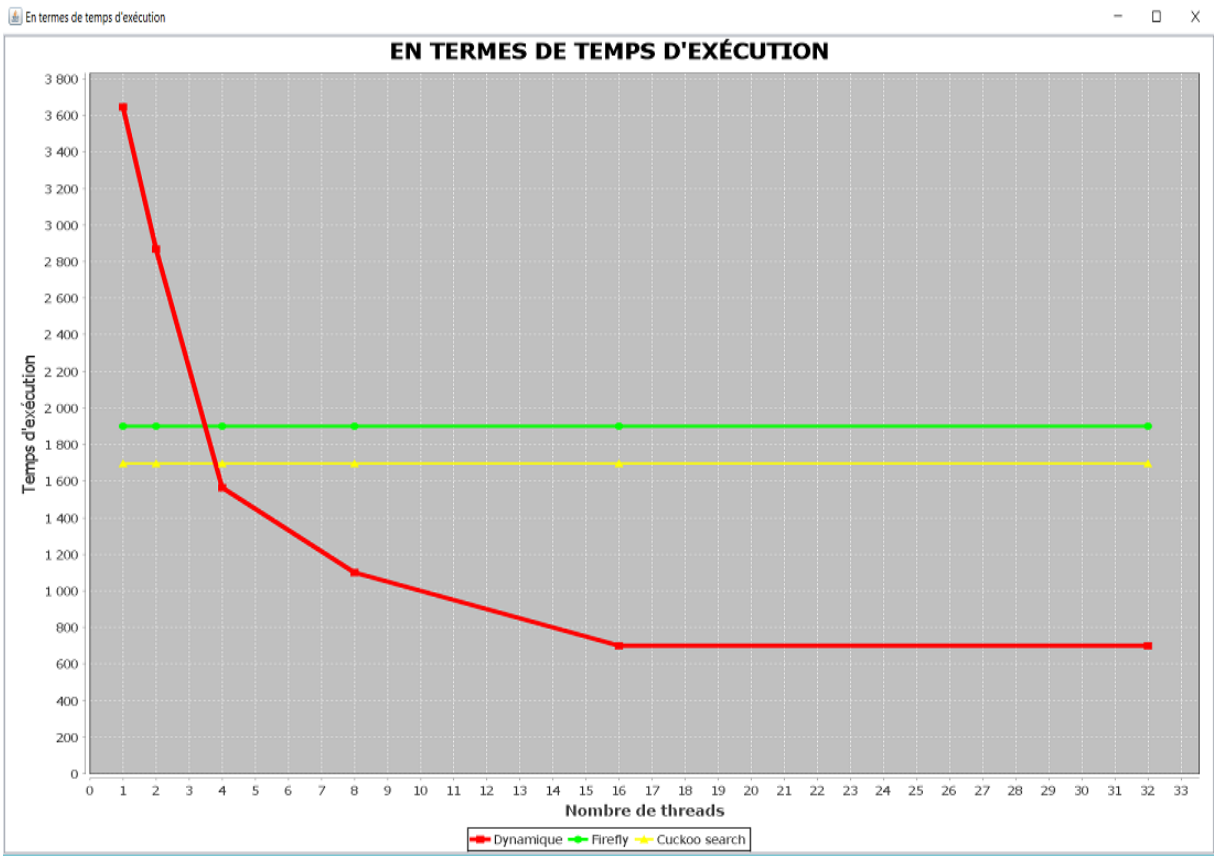


Figure 3.10 : comparaison en termes de temps de traitement

Nombre de threads / Temps D'exéc En ms Program Dyn	1	2	4	8	16	32
	3645	2862	1564	1100	700	700
Temps d'exéc Cuckoo Search En ms	1696					
Temps d'exéc Firefly En ms	1925					

Tableau 3.5 : résultats de la comparaison en termes de temps de traitement

Il est nécessaire pour l'utilisateur primaire de satisfaire les demandes des utilisateurs secondaires dans un laps de temps très court, car ce critère est très important à cause de l'environnement de la radio cognitive et aussi pour satisfaire la demande des utilisateurs secondaire.

Au début, la différence est remarquable entre les trois graphes de la figure 3.10 qui illustre le temps de traitement des trois algorithmes. On remarque que les résultats des deux métaheuristique cuckoo search et firefly sont fixes, leurs valeurs sont inchangées par rapport au graphe de la programmation dynamique qui dépend du nombre de threads. On voit clairement que le temps de traitement (d'exécution) diminue pour la programmation dynamique en utilisant plus de threads. A partir de 4 threads, on constate que la programmation dynamique se rapproche des deux métaheuristiques pour ensuite les dépasser et obtenir de meilleurs résultats.

On remarque aussi que le temps de traitement pour la programmation dynamique va diminuer entre 4 et 16 threads pour se stabiliser après avoir franchis la barre des 16 threads, notez bien

qu'une fois le nombre de 32 threads dépassé, le graphe va s'accroître à cause de la perte du temps lors de la communication entre les threads.

Donc, d'après notre étude, on peut dire que la programmation dynamique offre de meilleurs résultats lorsqu'on utilise plus de 4 threads, c'est-à-dire qu'en combinant la programmation dynamique (méthode exacte) et les threads, on obtient des résultats remarquablement supérieurs aux cuckoo search et firefly (métaheuristiques).

3.6.2 Comparaison en termes de gain coté PU

Le graphe suivant représente une comparaison entre les 3 algorithmes en termes de coût.

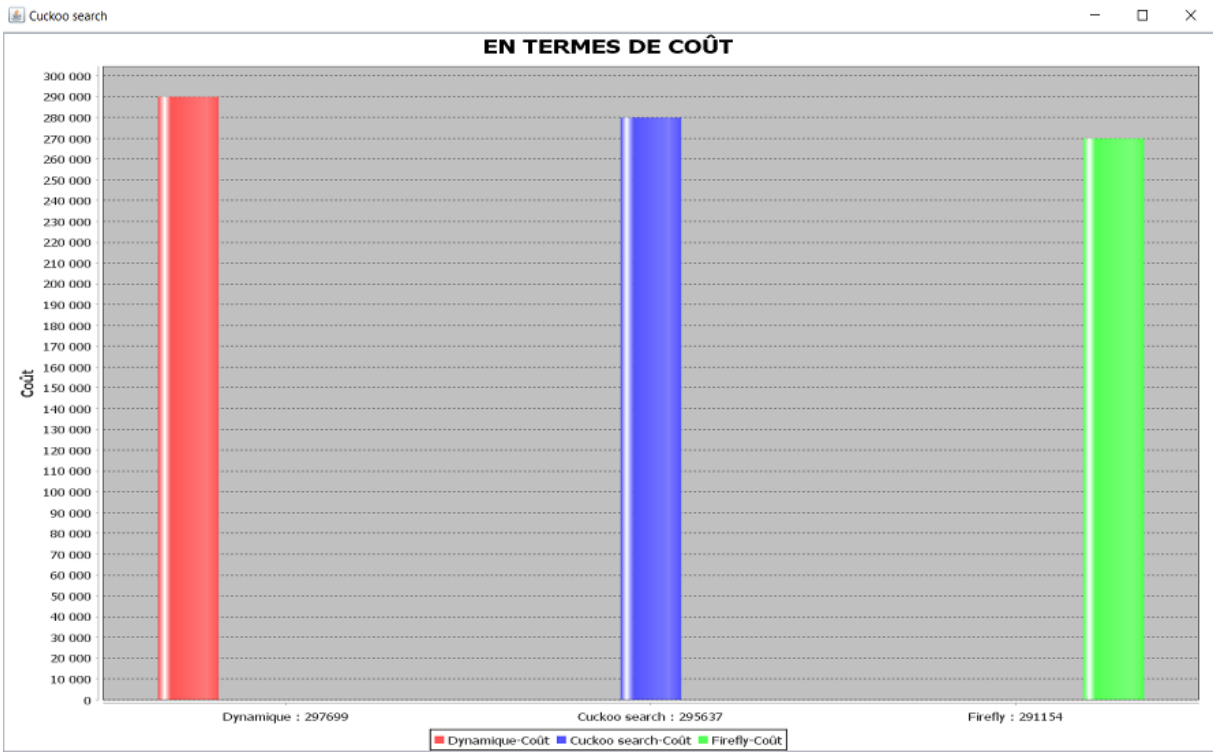


Figure 3.11 : comparaison en termes de gain (coût)

La figure 3.11 illustre clairement un histogramme de comparaison en termes de gain (coût) entre les trois algorithmes : la programmation dynamique, cuckoo search et firefly.

On remarque clairement que le gain le plus élevé est obtenu avec la programmation dynamique suivi par le cuckoo search et enfin le firefly.

Cette valeur de gain obtenu par la programmation dynamique est carrément indépendante du nombre de threads utilisés, car l'algorithme de la programmation dynamique prend toujours les individus dont les gains sont meilleurs, c'est pour cela, les résultats des deux algorithmes

métaheuristiques ne peuvent jamais donner des résultats proches à la valeur obtenue par l'algorithme de la programmation dynamique en termes de gain.

Donc l'utilisation de la programmation dynamique est très bénéfique pour l'utilisateur primaire car elle lui permet d'obtenir plus de gain durant la négociation avec les utilisateurs secondaires en se comparant aux cuckoo search et firefly .

3.6.3 Comparaison en termes de nombre de SU satisfaits

Le graphe suivant représente une comparaison entre les 3 algorithmes en termes de SU satisfait.

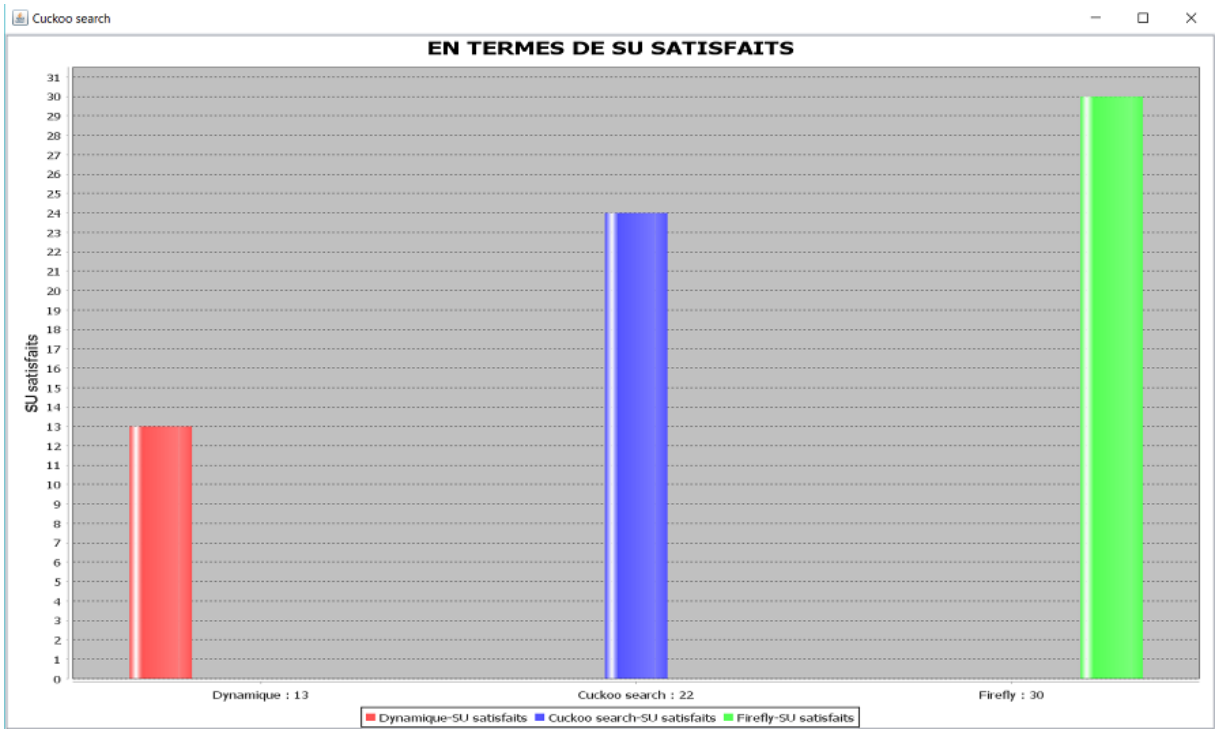


Figure 3.12 : comparaison en termes du nombre de SU satisfaits

La figure 3.12 illustre un histogramme de comparaison en termes de nombre de SU satisfaits entre les trois algorithmes.

On constate que l'algorithme firefly nous donne la meilleure valeur, suivie par l'algorithme cuckoo search et en dernière position par la programmation dynamique, cette dernière offre ses résultats en termes de nombre de SU satisfaits indépendamment du nombre de threads.

On peut déduire que la métaheuristique firefly offre le meilleur résultat en termes du nombre de SU satisfaits grâce à son mode de recherche qui est plus adapté à ce type de problème.

3.7 Conclusion

A travers ce chapitre, nous avons pu présenter les résultats obtenus durant notre étude. Ces résultats montrent que l'implémentation des threads pour la programmation dynamique permet un estimable bénéfice pour le temps de réponse coté PU, ayant déjà montré sa preuve avec le gain « coût », néanmoins, elle satisfait moins de SU par rapport aux autres métaheuristiques (CS et FA). Un moteur cognitif utilisant nativement les métaheuristiques peut également se baser sur les méthodes exactes mais à condition de trouver un moyen algorithmique pour paralléliser la méthode exacte.

Conclusion générale

Les différentes recherches qui ont été menés au sujet de la radio cognitive ont permis de répondre à des défis relatifs à sa réalisation, ainsi qu'à son application auprès de technologies innovantes.

La radio cognitive est la technologie clé qui permettra de résoudre les problèmes de saturation du spectre et les conflits d'accès à ce dernier.

Les échanges qui sont fait lors d'une négociation d'accès au spectre entre les utilisateurs primaires et secondaires, sont observés comme une opération délicate et aussi comme un problème d'optimisation combinatoire.

Ainsi dans notre étude, nous nous somme focalisés sur une solution qui pourrait néanmoins donner un meilleur rendement à cette négociation, notre projet est bâti sur deux types d'approches pour aborder ce type de problème. D'une part les méthodes qui garantissent l'optimalité de la solution obtenue, ces méthodes sont appelées méthodes exactes, d'autre part les métaheuristiques qui permettent d'obtenir de bons résultats rapidement mais qui ne garantissent pas que l'optimum ait été trouvé, ces méthodes sont appelées méthodes approchées.

Trois approches nous intéressent plus exactement, car elles sont à la base de tout le travail effectué dans ce projet. Il s'agit de la programmation dynamique, une méthode d'optimisation exacte et des deux algorithmes métaheuristiques cuckoo search et firefly qui représente les méthodes d'optimisation approchées.

Dans ce travail nous avons donc expliqué ces trois algorithmes d'une manière très détaillée et nous avons introduit tout le vocabulaire qui leur est associé et qui nous a été utile dans la comparaison élaborée entre eux afin d'obtenir un résultat optimal qui va pouvoir appuyer le choix d'une méthode parmi celles étudiées.

Ces résultats obtenus lors de l'étude comparative, valident un bénéfice pour la parallélisation de la méthode exacte en termes de temps de traitement et coût obtenu côté PU.

Finalement et comme perspective, nous pouvons élargir ce travail pour aborder la parallélisation pour d'autres méthodes exactes comme le Branch and Bound, le Branch and Price afin d'obtenir des résultats plus performants.

Références Bibliographiques

[1] B. Benmammar, and A. Amraoui. Radio resource allocation and dynamic spectrum access.,» John Wiley & Sons, 2013.

[2] International telecommunication union, «FINAL ACTS world radiocommunication conference,» Genève, 2012.

[3] B. Sayrac, Cognitive Radio and its Application for Next Generation Cellular and Wireless Networks (Introduction to the radio cognitive), Springer, 2012.

[4] G. M. J. Mitola, «Cognitive radio: making software radios more personal,» pp. 13-18, Aout 1999.

[5] S. Haykin, «Cognitive radio: Brain empowered wireless communications,» vol. 23, pp. 201-220, 7 février 2005.

[6] Joseph. Mitola. III, «Encyclopedia of Telecommunications,» 15 avril 2003.

[7] Ngom, Insa, and Louis Diouf «La radio cognitive,»Master professionnel télécommunications 2008.3(2007).

[8] P. Charalampidis, Écrivain, *cognitive radio architecture: the engineering foundations of radio xml*. [Performance]. wiley, 2006.

[9] «A survey of cognitive radio and TV white spaces In Malaysia, [En ligne] Available:https://www.researchgate.net/publication/260410679 _A_survey_of_cognitive_radio_and_TV_white_spaces_in_Malaysia. [dernier accès le 22 mai 2017].

[10] Puchinger, Jacob, and Günther R.Raidl «Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification» International Work-Conference on the Interplay Between Natural and Artificial Computation *Springer*,

Berlin Heidelberg, 2005.

[11] G. M. Padberg, «A branch-and-cut algorithm for the resolution of largescale symmetric traveling salesman problems,» *SIAM review*, pp. 60-100, 1991.

[12] Lawler, Eugene L., and David E. Wood. "Branch-and-bound methods: A survey." Operations

research 14.4 (1966): 699-719.

[13] Barnhart, Cynthia, et al. "Branch-and-price: Column generation for solving huge integer programs." Operations research 46.3 (1998): 316-329.

[14] Fukasawa, Ricardo, et al. "Robust branch-and-cut-and-price for the capacitated vehicle routing problem." Mathematical programming 106.3 (2006): 491-511.

[15] R. E. Bellman, «Dynamic Programming,» *Princeton University Press,* 1957.

[16] S. Jacquin, Hybridation des métaheuristiques et de la programmation dynamique pour les problémes d'optimisation mono et multi-objectif : Application à la production d'énergie, Lille, 2015.

[17] M.-E. Marmion, Recherche locale et optimisation combinatoire: de l'analyse structurelle d'un probléme à la conception d'algorithmes efficaces, Lille: université lille 1, 2011.

[18] P. e. G.Dreyfus, La méthode du recuit simulé, Paris: IDSET, 1988.

[19] J. M. Pollard, Kangaroos, Monopoly and Discrete Logarithms, England: journal of Cryptology, 2000.

[20] F. Glover, «Future paths for integer programming and links to artificial intelligence,» chez *Computers & operations research*, ELSEVIER, 1986, pp. 533-549.

[21] Eiben, Agoston E., and James E. Smith. Introduction to evolutionary computing. Vol. 53. Heidelberg: springer, 2003.

[22] Dréo, Johann, et al. Métaheuristiques pour l'optimisation difficile. Eyrolles, 2003.

[23] Eberhart, Russell C., Yuhui Shi, and James Kennedy. Swarm Intelligence (The Morgan Kaufmann Series in Evolutionary Computation). 2001.

[24] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. IEEE, 2009.

[25] Yang, Xin-She, and Suash deb. "Engineering optimisation by cuckoo search." International Journal of Mathematical Modelling and Numerical Optimisation 1.4 (2010):330-343.

[26] B. Mandelbrot, A life in many dimensions, world scientific publishing Co, 2015.

[27] Tammero, Lance F., and Michael H. Dickinson. «The influence of visual landscape on the free

flight behaviour of fruit fly Drosophila melanogaster,» *journal of experimental biology* 205.3 (2002):327-343.

[28] Reynolds, Andy M., and Mark A. Frye. "Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search." PloS one 2.4 (2007): e354.

[29] Stanger-Hall, Kathrin F., James E. Lloyd, and David M. Hillis. "Phylogeny of North American fireflies (Coleoptera: Lampyridae): implications for the evolution of light signals." Molecular phylogenetics and evolution 45.1 (2007): 33-49.

[30] X.-S. Yang, Nature-inspired Metaheuristic algorithms, united kingdom: luniver press, 2010.

[31] Łukasik, Szymon, and Sławomir Żak. "Firefly algorithm for continuous constrained optimization tasks." International Conference on Computational Collective Intelligence.

Springer Berlin Heidelberg, 2009.

[32] Amraoui, Asma, et al. "Auction-based Agent Negotiation in Cognitive Radio Ad Hoc Networks." International Conference on Ad Hoc Networks. Springer Berlin Heidelberg, 2012.

[33] «Bienvenue à NetBeans,» [En ligne]. Available: https://netbeans.org/index_fr.html. [dernier accès le 10 juin 2017].

[34] «What is Photoshop? definition and meaning – BusinessDictionary.com,» [En ligne]. Available: http://www.businessdictionary.com/definition/Photoshop.html. [dernier accès le 8 juin 2017].

[35] «JFreeChart - JFree.org,» [En ligne]. Available: www.jfree.org/jfreechart/. [dernier accès le 9 juin 2017].

[36] B. Benmammar, «Réseaux de radio cognitive Allocation des ressources radio et accès dynamique au spectre,» arXiv preprint arXiv:1407.2705, 2014.

MLA

Résumé

Au cours de cette dernière décennie, le spectre a connu beaucoup de problèmes tels que la limitation et l'interférence. Ainsi, la radio cognitive est considérée comme une nouvelle approche pour améliorer l'utilisation efficace du spectre sans fil existant, entre l'utilisateur dit « primaire » qui possède une licence sur des bandes de fréquence non-occupées et un autre dit « secondaire » qui pourra accéder à tous moments à ces bandes. Dans ce travail, nous avons étudié un scénario de gestion du spectre, en utilisant une technique basée sur la négociation entre un utilisateur primaire et plusieurs utilisateurs secondaires. Pour cela, nous avons choisi la programmation dynamique comme méthode exacte, et deux métaheuristiques qui sont Cuckoo search et Firefly comme méthodes approchées, afin d'établir une étude comparative selon différents critères que nous avons fixés, tels que le temps de traitement, le coût obtenu, ainsi que le nombre d'utilisateurs secondaires satisfaits. Cette étude comparative nous a permis de prouver que le parallélisme de la programmation dynamique abouti à de meilleurs résultats par rapport aux autres algorithmes métaheuristiques.

Mots clés : Radio cognitive, programmation dynamique, métaheuristique, Cuckoo search, Firefly.

Abstract

Over the past decade, the spectrum has faced many problems such as limitation and interference. Thus, cognitive radio is seen as a new approach to improve the efficient use of the existing wireless spectrum between the so-called "primary" user who has a license on unoccupied frequency bands and another called "secondary" Which will be able to access at all times to these bands. In our project, we studied a scenario of spectrum management, using a technique based on the negotiation between a primary user and several secondary users. For this, we chose dynamic programming as an exact method, and two metaheuristics which are Cuckoo Search and Firefly as approximate methods, in order to establish a comparative study according to different criteria we have fixed, such as the processing time, the cost obtained, as well as the number of secondary users satisfied. This comparative study enabled us to prove that the parallelism of dynamic programming yielded better results than other metaheuristic algorithms.

Keywords: Cognitive radio, Dynamic programming, Metaheuristic, Cuckoo search, Firefly.

ملخص

خلال العقد الماضي، واجه الطيف العديد من المشاكل مثل الحد والتداخل. وهكذا، ينظر إلى الراديو المعرفي كنهج جديد لتحسين كفاءة استخدام الطيف اللاسلكي الحالي بين ما يسمى المستخدم "الأساسي" الذي لديه ترخيص على نطاقات التردد غير مشغولة والمستخدم "الثانوي" والذي سوف تكون لديه القدرة على الوصول في جميع الأوقات إلى هذه النطاقات. في مشروعنا، درسنا سيناريو إدارة الطيف، وذلك باستخدام تقنية تقوم على التفاوض بين المستخدم الأساسي والعديد من المستخدمين الثانوي. لهذا قمنا باختيار البرمجة الديناميكية كطريقة دقيقة، واثنين من ميتاهوريستيكش منهم بحث الوقواق واليراعة كطرق تقريبية، من أجل إنشاء دراسة مقارنة وفقا لمعايير مختلفة قمنا بتثبيتها، مثل وقت المعالجة والتكلفة وكذلك عدد المستخدمين الثانوي اللذين تم ارضائهم. وقد مكنتنا هذه الدراسة المقارنة من إثبات أن التوازي في البرمجة الديناميكية حقق نتائج مثلى مقارنة بالخوارزميات الميتاهوريستيكش الأخرى.

الكلمات المفتاحية: الراديو المعرفي، البرمجة الديناميكية، ميتاهوريستيكش، بحث الوقواق، اليراعة.