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: *Modèle Intelligent et Décision (M.I.D)*

Thème

Aide à la décision dans les réseaux de radio cognitive en appliquant l'algorithme de Branch and Bound

Réalisé par :

- **M BENTALHA Kamal**
- **M ABDERRAHIM Zoheir**

Présenté le 04 Juillet 2017 devant le jury composé de MM.

- *M BENAZZOUZ Mourtada* *(Président)*
- *Mme BELHABI Amel* *(Encadreur)*
- *Mme AMRAOUI Asma* *(Examinateur)*

Année universitaire : 2016-2017

Remerciements

Avant tout, nous rendons grâce à DIEU tout puissant de nous avoir accordé la volonté et le courage pour réaliser ce mémoire.

Au terme de ce travail nous tenons tout d'abord à exprimer notre profonde gratitude à notre encadreur Mme. Amel BELHABI.

Et plus particulièrement M. Ghambaza pour l'aide qu'il nous a apporté durant la réalisation de ce travail.

Nos remerciements s'adressent à tous les membres du jury pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail.

Dédicace

Je dédie ce mémoire,
à tous ceux et toutes celles
qui m'ont accompagné et soutenu
durant cette année de formation

M. ABDERRAHIM Z.

Dédicace

Je dédie ce mémoire

À mes très chers parents que Dieu les garde

A ma chère femme qui m'a soutenu durant

ma formation

À mes chères petits enfants

À ma famille et mes amis.

M. BENTALHA K.

Table des matières

Liste des abréviations i

Liste des Illustrations ii

Introduction Générale 1

Chapitre I : La Radio Cognitive 3

I.1 Introduction 3

I.2 Radio logicielle (software radio) 3

I.3 La radio cognitive 4

I.3.1 Historique : 4

I.3.2 Définitions 4

I.3.3 Architecture de la RC 5

I.3.4 Cycle de cognition 6

I.3.5 Fonctions de la radio cognitive 8

I.3.6 Composantes de la radio cognitive 9

I.4 Les domaines d'application de la radio cognitive 10

I.5 Conclusion 11

Chapitre II : Aide à la décision et méthode de Branch and Bound 13

II.1 Introduction 13

II.2 Définition 13

II.3 Problématique d'aide à la décision 14

II.4 Le problème du sac à dos 14

II.5 L'optimisation combinatoire 16

II.6 La théorie de la complexité 17

II.7 Les différentes classes de complexité 17

II.7.1 La classe P 17

II.7.2 La classe NP 18

II.7.3 La classe NP-complets 18

II.7.4 La classe NP-difficiles 18
II.8 Les méthodes exactes 19
II.8.1 La méthode Branch and Bound (B&B) 20
II.9 Méthode heuristique 22
II.9.1 Définitions 22
II.9.2 Les algorithmes gloutons 22
II.10 CONCLUSION 24
Chapitre III : Implémentation et comparaison des résultats 26
III.1 Introduction 26
III.2 Un système multi-agents 26
III.3 Propriétés clés des Agents 27
III.4 Domaines d'application des agents 28
III.5 Topologie du réseau utilisé 28
III.6 Présentation du scenario 29
III.7 Présentation des algorithmes utilisés 30
III.8 Outils utilisés pour l'implémentation de l'application 31
III.8.1 JADE 31
III.8.2 JFreechart 32
III.9 Présentation de l'application 33
III.10 Résultats après simulation 37
III.10.1 interactions entre les SUs et les PUs 37
III.10.2 Résultats 01 après simulation 39
III.10.3 Résultats 02 après simulation 39
III.11 Conclusion : 40
Conclusion Générale et perspectives 42
Reference Bibliographique I

Liste des abréviations

RC : la radio cognitive

SDR : software defined radio

QoI : quality of information

EMI : interférence électromagnétique

B&B : branch and bound

PSE : Par séparation et évaluation

SMA : système multi-agents

KP : knapsack problem

MDKP : multidimensional knapsack problem

PU : Primary User

SU : Secondary User

JADE : Java Agent Development Framework

AC : Autonomic Computing

ACC : Agent Communication Channel

AMS : Agent Management System

DA : Dummy Agent

DF : Directory Facilitator

SA : Sniffer Agent

FIPA : Fountion for Intelligent Physical Agents

Liste des Illustrations

Figure I.1 : Architecture de la RC. ... P.5
Figure I.2 : Cycle de cognition de Mittola. P.6
Figure I.3 : Fonctions de gestion du spectre. P.8
Figure I.4 : Composantes de la radio cognitive..... P.9
Figure II.1 : La formule du sac à dos KP. P.14
Figure II.2 : La formule de MDKP. P.15
Figure II.3 : La complexité le l'algorithme. P.17
Figure II.4 : Classification des méthodes d'optimisation..... p.20
Figure II.5 : Arborescence de B&B p.21
Figure II.6: les étapes de l'algorithme glouton. P.23
Figure III.1 : Topologie du réseau (mode ad hoc). P.29
Figure III.2: Scénario proposé. P.30
Figure III.3: Algorithme de Branch and Bound. P.30
Figure III.4: Interface graphique (jade GUI). P.32
Figure III.5: Organigramme de l'application. P.34
Figure III.6: Interface d'accueil P.34
Figure III.7: Onglet de démarrage test. P.35
Figure III.8: Onglet de Gestion des SUs. P.35
Figure III.9: Onglet de Gestion des PUs. P.36
Figure III.10: Onglet des Graphes. P.36
Figure III.11: Lancement du test de simulation. P.37
Figure III.12: Agent sniffer. P.38
Figure III.13: Comparaison entre l'algorithme de B&B et l'algorithme Glouton
En termes de valeur optimale. P.39
Figure III.14: Comparaison entre l'algorithme de B&B et l'algorithme Glouton
En termes de Temps d'exécution P.39

Introduction Générale

Introduction Générale

e concept de radio cognitive est en réalité une interaction entre la technologie sans fil et l'intelligence artificielle. En effet, la capacité de radio cognitive intégrée à un terminal lui Loffre la possibilité d'interagir avec son environnement radio afin de s'y adapter, de détecter les fréquences libres et de les exploiter. Le terminal aura suffisamment de capacités lui permettant de gérer efficacement l'ensemble des ressources radio. Néanmoins, la mise en œuvre de cette technologie nécessite des études approfondies sur la détection des ressources libres, le partage du spectre, la gestion de la mobilité spectrale, la gestion de la mobilité des utilisateurs, etc.

Notre problématique est d'apporter de l'aide à la décision dans la gestion dynamique du spectre entre les utilisateurs primaires et secondaires dans le contexte de la radio cognitive. Pour cela nous avons modélisé notre problématique en problème combinatoire comme un problème de sac a dos Multidimensionnel, pour le résoudre nous avons appliqué deux points cruciaux:

- Le premier est l'algorithme de Séparation et évaluation (branch and Bound) en calculant la borne supérieur (upper bound) pour palier au problème d'explosion combinatoire.
- Le deuxième est l'utilisation des systèmes multi- agents.

Aussi nous avons ajouté un algorithme glouton comme une deuxième solution pour faire une comparaison des résultats.

Notre mémoire est composé de trois chapitres : le premier chapitre est consacré pour définir tous les concepts de base de la radio cognitive, le deuxième chapitre est conçu pour présenter les bases de l'aide à la décision dans le cadre de la radio cognitive ainsi l'utilité des méthodes exactes (Branch and Bound), les méthodes heuristiques (Glouton) dans la résolution des problèmes combinatoires et les problèmes des sacs à dos Multidimensionnels. En fin le troisième chapitre est basé sur l'implémentation de la solution proposée et les simulations des deux méthodes exacte et heuristique en utilisant la plate forme Jade.

CHAPITRE I

LA RADIO COGNITIVE

Chapitre I : La Radio Cognitive

I.1 Introduction

Les ressources spectrales sont devenues de plus en plus précieuses avec la prolifération rapide de standards et de services de radiocommunication.

Il est aujourd'hui largement reconnu que les systèmes sans fil de communications numériques n'exploitent pas l'intégralité de la bande de fréquence disponible, on appelle ces ressource spectrales non exploitées des espaces libres ou des opportunités. le problème donc est comment exploiter ces espaces libres ?

De cela née le besoin d'introduire la notion de cognition dans le réseau pour objectif de:

- Management flexible des ressources spectrales
- Qualité du service
- Sécurité
- Contrôle d'accès

Nous allons aborder dans ce chapitre la notion de cognition dans le réseau de communication radio et définir les principales caractéristiques de la radio cognitive ainsi que ses principales fonctions.

I.2 Radio logicielle (software radio)

La radio logicielle permet, idéalement, à des équipements de communiquer avec n'importe quel standard de radiocommunications par la seule modification du logiciel embarqué, et donc sans modification d'un quelconque élément matériel. Cependant, le caractère statique des protocoles actuels de communication pose les questions de l'optimisation de l'efficacité spectrale et de la flexibilité du domaine radio. De cette réflexion, concernant directement la pérennité des télécommunications modernes, est né le domaine de la radio intelligente ou cognitive radio. Cette évolution, aujourd'hui incontournable dans le monde des radiocommunications, donne la possibilité aux appareils de communication, devenus plus autonomes, de choisir les meilleures conditions de communication [2].

- **La radio logicielle restreinte (software defined radio)**
  La radio logicielle restreinte est un système de communication radio qui peut s'adapter à n'importe quelle bande de fréquence et recevoir n'importe quelle modulation en utilisant le même matériel.
  Les opportunités qu'offre le SDR lui permettent de résoudre des problèmes de la gestion dynamique du spectre. Les équipements SDR peuvent fonctionner dans des réseaux sans fil hétérogènes c'est-à-dire qu'un SDR idéal peut s'adapter automatiquement aux nouvelles fréquences et aux nouvelles modulations.

I.3 La radio cognitive

I.3.1 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.

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. D'après lui : ***« Une radio cognitive peut connaître, percevoir et apprendre de son environnement puis agir pour simplifier la vie de l'utilisateur.*** » [3]

I.3.2 Définitions

Le principe de la radio cognitive, repris dans la norme IEEE 802.22, nécessite une gestion alternative du spectre qui est la suivante : un mobile dit secondaire pourra à tout moment accéder à des bandes de fréquence qu'il juge libre, c'est-à-dire, non occupées par l'utilisateur dit primaire possédant une licence sur cette bande.

L'utilisateur secondaire devra les céder une fois le service terminé ou une fois qu'un utilisateur primaire aura montré des velléités de connexion.

L'une des principales caractéristiques de la **radio cognitive** est la capacité d'adaptation où les paramètres de la radio (**fréquence porteuse**, **puissance**, **modulation**, **bande passante**) peuvent être modifiés en fonction de : l'environnement radio, la situation, les besoins de l'utilisateur, la coopération des utilisateurs, l'état du réseau, la géolocalisation,...etc.

I.3.3 Architecture de la RC

Mitolla a défini l'architecture d'une radio cognitive par un ensemble cohérent de règles de conception par lequel un ensemble spécifique de composants réalise une série de fonctions de produits et de services.

Les six composantes fonctionnelles de l'architecture d'une radio cognitive sont:

- ✓ La perception sensorielle (Sensory Perception : SP) de l'utilisateur qui inclut l'interface haptique (du toucher), acoustique, la vidéo et les fonctions de détection et de la perception.
- ✓ Les capteurs de l'environnement local (emplacement, température, accéléromètre, etc.).
- ✓ Les applications système (les services médias indépendants comme un jeu en réseau).
- ✓ Les fonctions SDR (qui incluent la détection RF et les applications radio de la SDR).
- ✓ Les fonctions de la cognition (pour les systèmes de contrôle, de planification, d'apprentissage).
- ✓ Les fonctions locales effectrices (synthèse de la parole, du texte, des graphiques et des affiches multimédias).

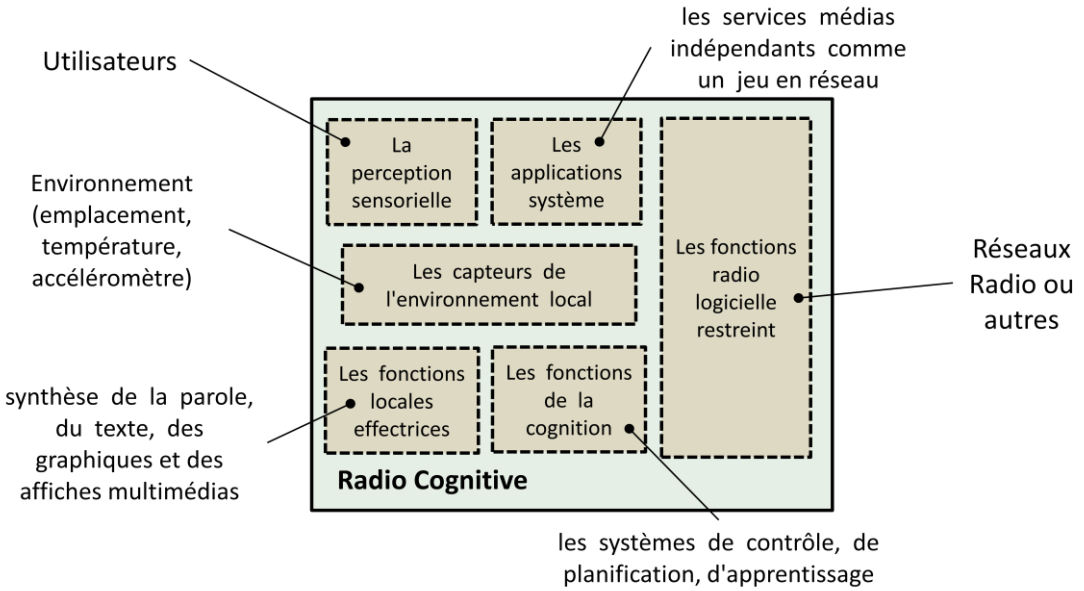

Figure I.1 : Architecture de la RC [1]

I.3.4 Cycle de cognition

La composante cognitive de l'architecture de la radio cognitive comprend une organisation temporelle, des flux d'inférences et des états de contrôle. Ce cycle synthétise cette composante de manière évidente. Les stimuli entrent dans la radio cognitive comme des interruptions sensorielles envoyées sur le cycle de la cognition pour une réponse. Une telle radio cognitive observe l'environnement, s'oriente, crée des plans, décide, et puis agit.

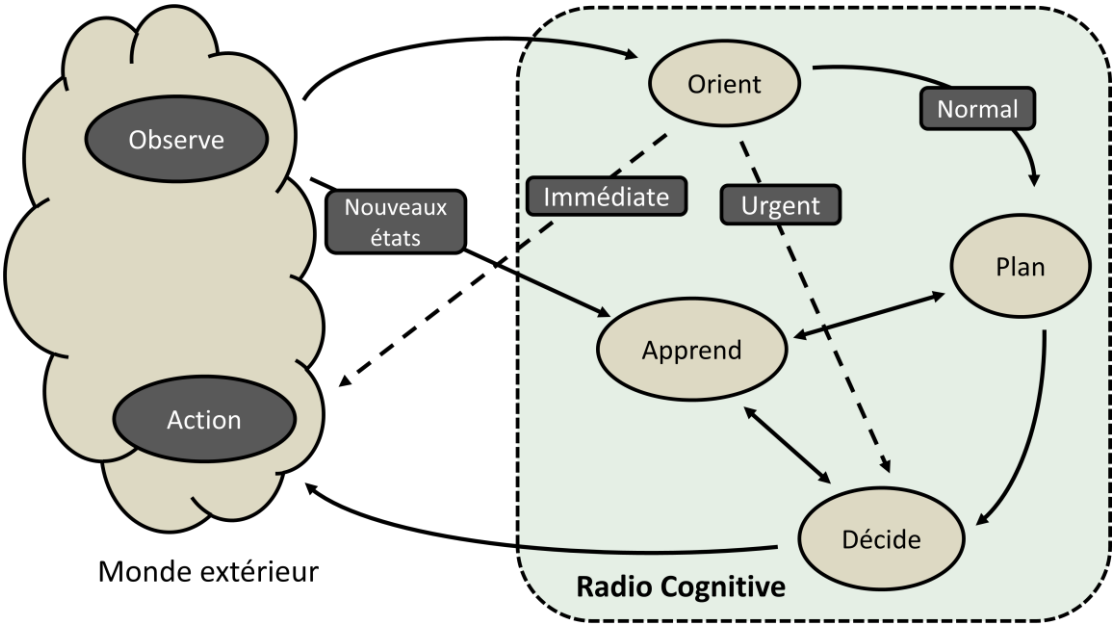

Figure I.2 : Cycle de cognition de Mittola [1]

a) Phase d'observation (détecter et percevoir)

La RC observe son environnement par l'analyse du flux de stimuli entrant. Dans la phase d'observation, la RC associe l'emplacement, la température, le niveau de lumière des capteurs, et ainsi de suite pour en déduire le contexte de communication. Cette phase lie ces stimuli à des expériences antérieures pour discerner les modèles au fil du temps. La radio cognitive rassemble les expériences en se souvenant de tout.

b) Phase d'orientation

La phase d'orientation détermine l'importance d'une observation en liant à celle-ci une série connue de stimuli. Cette phase fonctionne à l'intérieur des structures de données qui sont analogues à la mémoire à court terme (STM), que les gens emploient pour s'engager dans un dialogue sans forcément se souvenir de tout à la même mesure que dans la mémoire à long terme (LTM). Le milieu naturel fournit la redondance nécessaire pour lancer le transfert de la STM à la LTM. La correspondance entre les stimuli courants et les expériences stockées se fait par reconnaissance des stimuli ou par reliure.

c) Phase de planification

La plupart des stimuli sont traités avec délibérative plutôt qu'avec réactivité. Un message entrant du réseau serait normalement traité par la génération d'un plan (dans la phase de plan, la voie normale). Le plan devrait également inclure la phase de raisonnement dans le temps. Généralement, les réponses réactives sont préprogrammées ou apprises en étant dit, tandis que d'autres réactions de délibération sont prévues.

d) Phase de décision

La phase de décision sélectionne un plan parmi les plans candidats. La radio peut alerter l'utilisateur d'un message entrant ou reporter l'interruption à plus tard en fonction des niveaux de QoI (Quality of Information) statués dans cette phase.

e) Phase d'action

Cette phase lance les processus sélectionnés qui utilise les effecteurs sélectionnés qui accèdent au monde extérieur ou aux états internes de la radio cognitive. L'accès au monde extérieur consiste principalement à composer des messages qui doivent être envoyés dans l'environnement en audio ou exprimés dans différents langages appropriés.

f) Phase d'apprentissage

L'apprentissage dépend de la perception, des observations, des décisions et des actions. L'apprentissage initial est réalisé à travers la phase d'observation dans laquelle toutes les perceptions sensorielles sont continuellement comparées à l'ensemble de l'expérience antérieure pour continuellement compter les événements et se souvenir du temps écoulé depuis le dernier événement.

I.3.5 Fonctions de la radio cognitive

Les principales fonctions de la radio cognitive sont les suivantes:

a. Détection du spectre (Spectrum sensing)

Pour détecter le spectre non utilisé et le partager sans interférence avec d'autres utilisateurs.

b. **Gestion du spectre (Spectrum management)**

Capter les meilleures fréquences disponibles pour répondre aux besoins de communication des utilisateurs. L'analyse des spectres et la prenne de décision, qui dépend de l'environnement qui peut être :

- **non-coopératif**
- **coopératif**

c. Partage du spectre :

Fournir la méthode de planification du spectre équitable entre tous les utilisateurs de la radio cognitive existant.

d. Mobilité du spectre (Spectrum mobility)

C'est le processus qui permet à l'utilisateur de la radio cognitive de changer sa fréquence de fonctionnement.

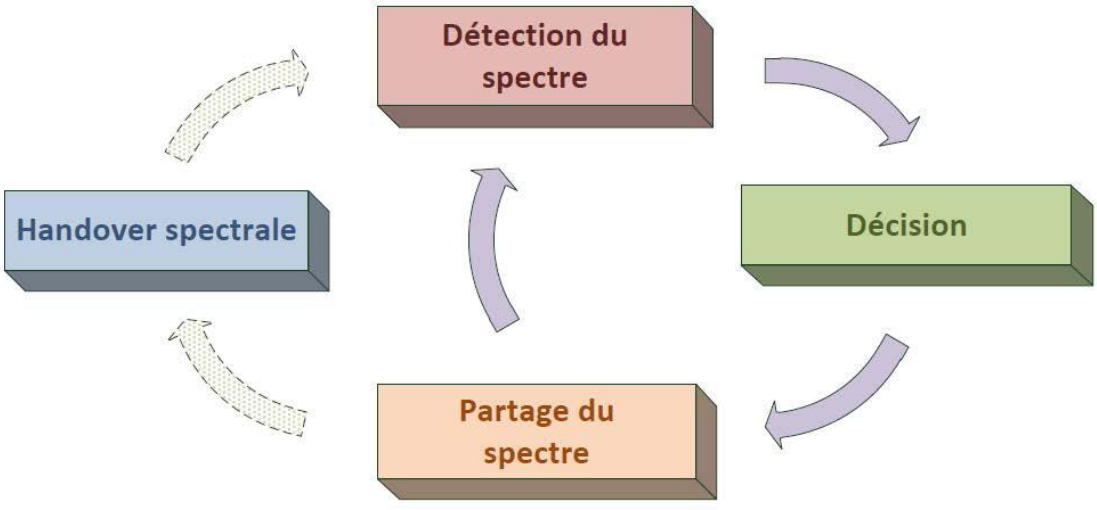

Figure I.3 Fonctions de gestion du spectre [1]

I.3.6 Composantes de la radio cognitive

Les différentes composantes d'un émetteur/récepteur radio cognitive qui mettent en œuvre ces fonctionnalités sont présentées dans la figure ci-dessous.

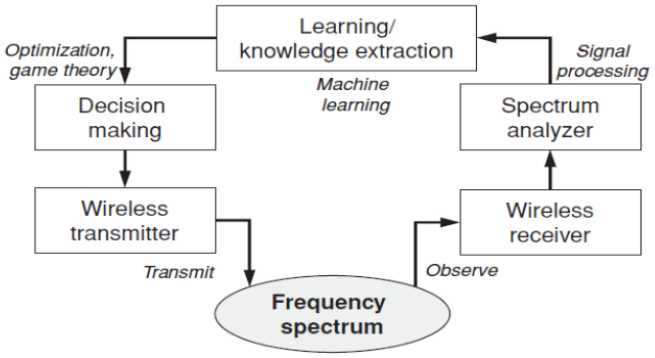

Figure I.4 Composantes de la radio cognitive [1]

- **Emetteur / Récepteur:** un émetteur/récepteur SDR sans fil est le composant majeur avec les fonctions du signal de transmission de données et de réception. En outre, un récepteur sans fil est également utilisé pour observer l'activité sur le spectre de fréquence (spectre de détection).
  - Les paramètres émetteur/récepteur dans le nœud de la radio cognitive peuvent être modifiés dynamiquement comme dicté par les protocoles de couche supérieure.
- **Analyseur de spectre (Spectrum analyser):**L'analyseur de spectre utilise les signaux mesurés pour analyser l'utilisation du spectre (par exemple pour détecter la signature d'un signal provenant d'un utilisateur primaire et trouver les espaces blancs du spectre pour les utilisateurs secondaires).
  - L'analyseur de spectre doit s'assurer que la transmission d'un utilisateur primaire n'est pas perturbée si un utilisateur secondaire décide d'accéder au spectre. Dans ce cas, diverses techniques de traitement du signal peuvent être utilisées pour obtenir des informations sur l'utilisation du spectre.
- **Extraction de connaissances et apprentissage (Knowledge extraction/learning):** L'apprentissage et l'extraction de connaissances utilisent les informations sur l'utilisation du spectre pour comprendre l'environnement ambiant RF (par exemple le comportement des utilisateurs sous licence). Une base de connaissances de l'environnement d'accès au spectre est construite et entretenue, qui est ensuite utilisée pour optimiser et adapter les paramètres de

transmission pour atteindre l'objectif désiré sous diverses contraintes. Les algorithmes d'apprentissage peuvent être appliqués pour l'apprentissage et l'extraction de connaissances.

- **Prise de décision (Decision making):** Après que la connaissance de l'utilisation du spectre soit disponible, la décision sur l'accès au spectre doit être faite. La décision optimale dépend du milieu ambiant, elle dépend du comportement coopératif ou compétitif des utilisateurs secondaires. Différentes techniques peuvent être utilisées pour obtenir une solution optimale.

I.4 Les domaines d'application de la radio cognitive

- **Coexistence de différentes technologies sans fil :**

Les nouvelles technologies sans fil (IEEE 802.22) sont en cours d'élaboration pour la réutilisation des fréquences radio allouées à d'autres services sans fil (service TV), La radio cognitive est une solution qui fournit la coexistence de ces différentes technologies et services sans fil [3].

- **Services de cyber santé (eHealth services):**

La plupart des dispositifs médicaux de soins utilisés sont sans fil et sont sensibles aux EMI (interférences électromagnétiques).

En outre, différents dispositifs biomédicaux (équipement et appareils chirurgicaux, de diagnostic et de suivi) utilisent la transmission RF.

Dans ce cas la radio cognitive peuvent être appliqués pour l'utilisation du spectre de ces dispositifs mais doit éviter toute interférence avec l'autre.

- **Réseaux d'urgence :**

les réseaux de sécurité publique et d'urgence peuvent profiter des concepts de la radio cognitive pour fournir la fiabilité et la flexibilité de communication sans fil.

- **Réseaux militaires :**

Avec la radio cognitive, les paramètres de la communication sans fil peuvent être adaptés de manière dynamique en fonction du temps et de l'emplacement ainsi que de la mission des soldats.

I.5 Conclusion

Corréler la gestion dynamique du spectre constitue une problématique majeure dans les réseaux à radio cognitive. Pour remédier à ce problème, nous sommes basés sur la technique du sac a dos multidimensionnelle avec la méthode du PSE (B&B) en utilisant ainsi les systèmes multi-agents pour obtenir une solution de gestion d'allocation dans les réseaux mobiles à radio cognitive. Dans ce premier chapitre, nous avons introduit des généralités sur la radio cognitive.

Afin que le lecteur comprenne davantage notre approche, nous consacrons le chapitre suivant à la présentation de l'aide à la décision pour les difficultés combinatoires, et l'utilité de la méthode par évaluation et séparation pour trouver les meilleures solutions à notre problématique

CHAPITRE II

Aide a la décision
et
Méthode Branch & Bound

Chapitre II : Aide à la décision et méthode de Branch and Bound

II.1 Introduction

Nous allons présenter dans ce chapitre les bases de l'aide à la décision dans le cadre de la radio cognitive ainsi l'utilité des méthodes exactes et les méthodes heuristiques dans la résolution des problèmes combinatoires comme les problèmes des sacs à dos Multidimensionnels.

II.2 Définition

L'aide à la décision est appliquée plus généralement pour tout problème présentant à la fois un enjeu et une difficulté.

→ 3 types de difficultés : Combinatoire, Incertain, Multicritère

- Les difficultés Combinatoires :
  - ✓ explosion combinatoire
- Les difficultés liées à l'Incertain
  - ✓ mauvaise intuition humaine
  - ✓ limites des approches probabilistes
- Les difficultés Multicritères

  les critères sont :
  - ✓ souvent conflictuels
  - ✓ souvent incommensurables
  - ✓ parfois qualitatifs (avis d'expert,...)

L'*aide à la décision* est une approche scientifique des problèmes de décision qui se posent dans tout contexte socio-économique.

- L'aide à la décision prend appui sur des modèles pour aider un acteur intervenant dans un processus de décision à obtenir des éléments de réponses aux questions qu'il se pose.[12]
- L'aide à la décision conduit cet acteur à accroître la cohérence entre son système de valeurs et ses décisions.

- L'aide à la décision peut aboutir à une "prescription" dont le but n'est pas de se substituer à la décision,

II.3 Problématique d'aide à la décision

- Choisir, dans un ensemble d'actions; la meilleur (ou un petit sous-ensemble des meilleurs actions)
- Classer les actions de la meilleure à la plus mauvaise (le classement peut être complet ou non)
- Affecter les actions à des catégories prédéfinies

II.4 Le problème du sac à dos

Le problème du sac à dos est un des problèmes d'optimisation académiques les plus importants. Il est noté par KP (De l'anglais, Knapsack Problem), son importance est liée en particulier au fait qu'il peut être utilisé en tant que sous problème dans de nombreux problèmes d'optimisation.

Dans sa version la plus simple, dénommée « sac à dos unidimensionnel binaire», le KP peut être défini comme suit: Supposant que nous avons un sac à dos qui a une capacité maximale C. d'autre part, nous avons un ensemble de N objets. Chaque objet i a un poids w_i et un profit p_i. Le problème consiste à sélectionner un sous ensemble d'objets de manière à maximiser le profit total du sac à dos sans dépasser sa capacité maximale. Le problème peut être formulé comme suit:

Maximiser $\sum_{i=1}^{n} pixi$
Avec $\sum_{i=1}^{n} wixi \leq C$, j=1,…….,N

$x_i = \begin{cases} 1 & Si\ l'objet\ i\ est\ sélectionné \\ 0 & Sinon \end{cases}$ i=1,…….,N

Figure II.1 la formule du sac à dos KP

Il est à noter que le problème du sac à dos unidimensionnel (appelé tout court: problème du sac à dos) est un problème d'optimisation combinatoire [4] et qu'il appartient à la classe des problèmes NP-Difficiles.

- **Le problème du sac à dos multidimensionnel**

La variante multidimensionnelle du problème du sac à dos est une issue de la classe des problèmes du sac à dos. Elle est notée par MKP (de l'anglais, Multidimensional Knapsack Problem). Le MKP est un problème d'optimisation combinatoire qui appartient à la classe des problèmes NP-Difficiles [5]. Dans ce problème, on suppose qu'on a un sac à dos avec M dimensions. Chaque dimension a une capacité maximale Cj (j=1,…, M). D'autre part, on a un ensemble d'objets. Chaque objet a une valeur (profit) pi (i=1,…, N) et un poids wji dans la dimension j du sac à dos. Le problème qu'on cherche à résoudre consiste à trouver un sous ensemble d'objets de manière à maximiser la valeur totale du sac à dos sans dépasser les capacités de toutes ses dimensions. Le MKP peut être formulé comme suit:

Maximiser $\sum_{i=1}^{n} pixi$

Avec $\sum_{i=1}^{n} wjixi \leq C_j$, j=1,…….,M

$x_i = \begin{cases} 1 & Si\ l'objet\ i\ est\ sélectionné \\ 0 & Sinon \end{cases}$ i=1,…….,N

Figure II.2 la formule de MKP

Le problème du Sac à Dos Multidimensionnel peut être utilisé pour formuler une panoplie de problèmes industriels tels que les problèmes de budgets, d'allocation des ressources physiques et logiques dans un système informatique distribué, de la découpe du stocke, de la gestion des projets et du stockage des conteneurs [6]. Vu son importance et son appartenance à la classe des problèmes NP-Difficiles, le problème du sac à dos multidimensionnel a sollicité l'attention de plusieurs communautés de chercheurs. En effet, sa résolution à fait l'objectif principal de plusieurs travaux proposés par plusieurs chercheurs.

Comme notre problème consiste à gérer l'allocation du spectre dans le contexte de la radio cognitive entre les utilisateurs primaires et secondaires nous avons introduit cette notion de Sac à dos Multidimensionnel, en considérant que les Pus représentent le

Sac avec une capacité qui est le paramètre suivant (nombre de canaux libres) à remplir par des objets qui sont les Sus ayant un poids et un profit (nombre de canaux demandés, Prix unitaire).

II.5 L'optimisation combinatoire

L'optimisation combinatoire, également appelée l'optimisation discrète, est une discipline qui recouvre l'optimisation en mathématiques appliquées et l'informatique. Dans un problème d'optimisation combinatoire, nous avons en général [7] :

– Un ensemble S de solutions s .

– Une fonction objectif f qui permet d'évaluer chaque solution s ∈ S , f : s → R;

– Un ensemble R, sous ensemble de S , dont les éléments sont appelés les solutions réalisables.

Un problème d'optimisation combinatoire consiste donc à déterminer un ensemble O, $O \subseteq R \subseteq S$ tel que :

$$\vee s \in O, f(s) = opt_{\acute{s} \in S}\{f(\acute{s}) | \acute{s} \in R\}$$

Où opt est soit sous forme de maximisation max, soit sous forme de minimisation min. Les solutions dans O sont appelées les solutions optimales. En général, il n'est pas nécessaire de trouver toutes les solutions dans O. Une seule solution de O suffira la plupart du temps. En tant que discipline, l'optimisation combinatoire est relativement récente, bien que certains problèmes d'optimisation combinatoire aient été étudiés séparément depuis longtemps. A partir des 1950, les outils mathématiques de programmation linéaire et de programmation linéaire en nombres entiers deviennent disponibles. C'est grâce à cette avancée que les problèmes d'optimisation combinatoire et leurs interrelations ont commencé à être étudiés dans un même contexte.

Parmi les exemples classiques des problèmes d'optimisation combinatoire, nous pouvons citer : le problème d'affectation de tâches à des ressources, le problème de transport, le problème du plus court chemin, le problème du voyageur de commerce et l'exemple de notre étude le problème du sac à dos .

II.6 La théorie de la complexité

La théorie de la complexité recouvre la complexité des algorithmes et la complexité des problèmes. Un algorithme est une méthode de calcul itérative conçue pour résoudre un problème précis.

La complexité des algorithmes a pour objectif d'évaluer le temps de calcul d'un algorithme en fonction de la taille du problème qu'il doit résoudre. Le temps de calcul d'un algorithme peut être mesuré par le nombre d'opérations élémentaires nécessaires (additions, divisions, tests, etc.) pour exécuter cet algorithme. En fonction des objectifs visés, nous pouvons chercher à établir la complexité dans le meilleur des cas, la complexité dans le pire des cas ou la complexité en moyenne.

Le tableau de la Figue II.3 résume la complexité de l'algorithme en fonction du comportement asymptotique de f (n).

$f(n)$	Complexité de l'algorithme
$O(1)$	complexité constante
$O(log(n))$	complexité logarithmique
$O(n)$	complexité linéaire
$O(nlog(n))$	complexité quasi-linéaire
$O(n^2)$	complexité quadratique
$O(n^3)$	complexité cubique
$O(n^p)$	complexité polynomiale
$O(n^p log(n))$	complexité quasi-polynomiale
$O(2^n)$	complexité exponentielle
$O(n!)$	complexité factorielle

Figure II.3 La complexité le l'algorithme

II.7 Les différentes classes de complexité

II.7.1 La classe P

Un algorithme déterministe est une séquence d'opérations à exécuter dans un ordre déterminé. En utilisant les mêmes données d'entrée, un algorithme déterministe fournit toujours le même résultat en sortie. Un problème de décision appartient à la classe P si et seulement si il est possible de le résoudre à l'aide d'un algorithme déterministe en temps polynomial.

II.7.2 La classe NP

La classe NP (Nondeterministic Polynomial) est celle des problèmes de décision (i.e à réponse dans oui, non) pour lesquels il existe un algorithme déterministe polynomial qui permet de vérifier la validité d'une solution proposée du problème supposée à réponse oui. Un tel algorithme déterministe polynomial est appelé un vérifieur.

La communauté scientifique admet généralement que P ≠ NP (sauf preuve du contraire).

II.7.3 La classe NP-complets

Parmi l'ensemble des problèmes appartenant à NP, il en existe un sous ensemble qui contient les problèmes les plus difficiles : on les appelle les problèmes NP complets.

Un problème NP-complets possède la priorité que tout problème dans NP peut être transformé (réduit) en celui-ci en temps polynomial. C'est à dire qu'un problème est NP-complets quand tous les problèmes appartenant à NP lui sont réductibles. Si on trouve un algorithme polynomial pour un problème NP- complets, on trouve alors automatiquement une résolution polynomiale de tous les problèmes de la classe NP.

II.7.4 La classe NP-difficiles

Un problème est NP-difficile s'il est plus difficile qu'un problème NP-complet, c'est à dire s'il existe un problème NP-complet se réduisant à ce problème par la réduction de Turing.

Ceci explique pourquoi, lors de l'étude d'un nouveau problème, on commence par chercher à classer ce problème. Si l'on parvient à montrer qu'il est polynomial, le problème sera résolu. Si par contre, on parvient à montrer qu'il est NP-complet, la recherche d'un algorithme exact pour résoudre un tel problème ne sera pas de première priorité, et il sera approprié de se concentrer sur des méthodes heuristiques que la plupart des spécialistes de l'optimisation combinatoire ont orienté leurs recherche pour les développer. Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible.

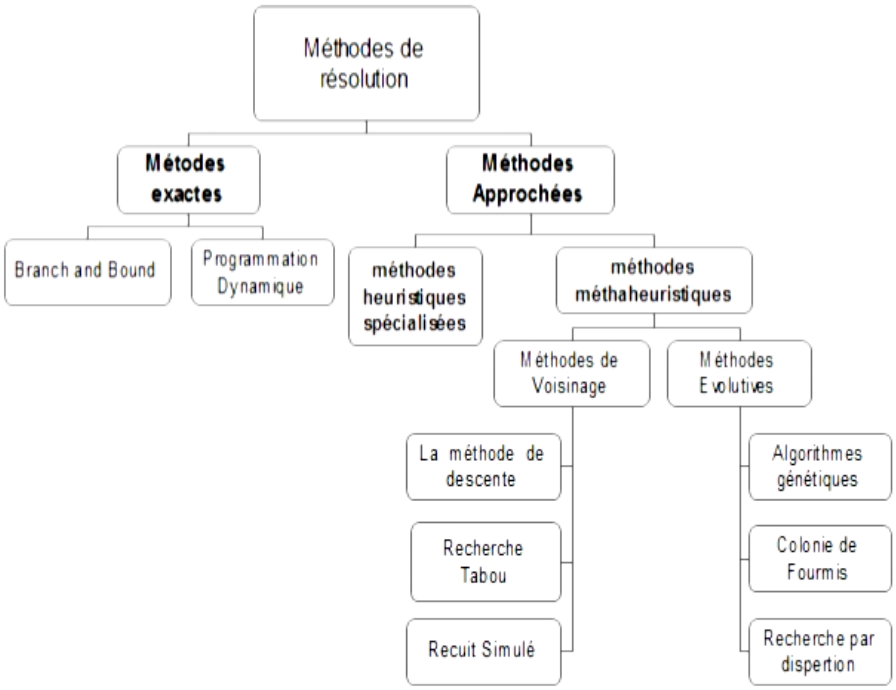

Figure II.4 Classification des méthodes d'optimisation

II.8 Les méthodes exactes

L'intérêt des méthodes exactes réside dans le fait quelles assurent l'obtention de la solution optimale du problème traité. En fait, elles permettent de parcourir la totalité de l'ensemble de l'espace de recherche de manière à assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche. Cependant, les méthodes exactes sont très connues par le fait qu'elles nécessitent un coût de recherche souvent prohibitif en termes de ressources requises. En effet, le temps de recherche et/ou l'espace mémoire nécessaire pour l'obtention de la solution optimale par une méthode exacte sont souvent trop grands, notamment avec des problèmes de grandes tailles. De ce fait, la complexité de ce type d'algorithme croit exponentiellement avec la taille de l'instance à traiter, elle devient très importante face à des problèmes comprenant plusieurs variables, fonctions objectifs et/ou critères.

Il existe de nombreuses algorithmes exacts y compris l'algorithme du simplexe, la programmation dynamique, l'algorithme A*, les algorithmes de séparation et évaluation (Branch and Bound, Branch and Cut, Branch and Price et Branch and Cut and Price), les algorithmes de retour arrière (Backtracking),

II.8.1 La méthode Branch and Bound (B&B)

La méthode par séparation et évaluation (nommée Branch and Bound en anglais) est notée B&B. C'est une des méthodes qui permettent la résolution exacte de problèmes d'optimisation, notamment les problèmes d'optimisation combinatoires dont on cherche à minimiser le coût de la recherche. B&B propose un mécanisme de recherche très intelligent, grâce à lequel elle permet une bonne exploitation de l'espace de recherche et l'aboutissement à la solution optimale plus rapidement que d'autres méthodes exactes en combinant deux principes primordiaux: la séparation et l'évaluation.

Le principe de base de la méthode B&B se base sur la technique «Diviser pour régner ». Elle consiste à dissocier le problème en sous problèmes de manière à représenter le problème sous forme d'une arborescence, où chaque nœud correspond à une solution partielle. Les solutions partielles se forment de manière incrémentale en s'enfonçant dans l'arbre. Chacune des solutions partielles potentielles possède une borne supérieure et une autre inférieure. Ces dernières sont utilisées pour couper quelques branches de l'arbre et ainsi éviter d'explorer tout l'arbre. En fait, si l'évaluation partielle d'un nœud x_i a montré que sa qualité est supérieure à la borne supérieure, le sous arbre en question sera élagué; sinon, le nœud sera divisé en sous nœuds. Ce processus se répète tant qu'il reste des branches non parcourues et la recherche continue jusqu'à trouver la solution optimale si elle existe.

L'utilisation de la méthode B&B nécessite:

- Une solution initiale permettant d'entamer la recherche.
- Une stratégie permettant la division du problème P en sous problèmes P_i.
- Une fonction permettant le calcul des différentes bornes.
- Une stratégie de parcours de l'arbre: parcourir en profondeur, en largeur…etc.

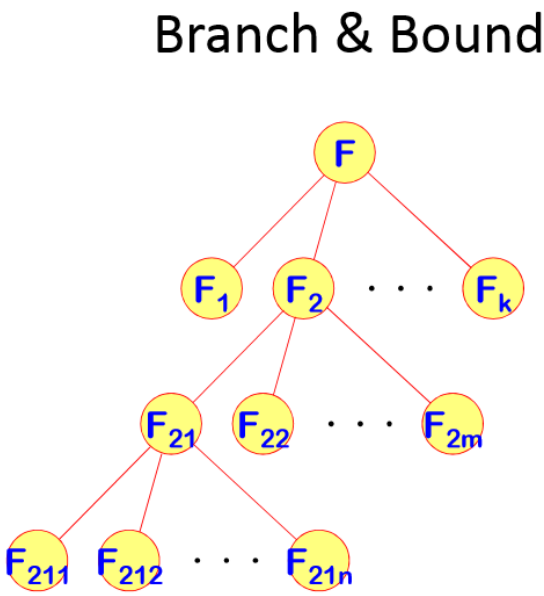

Figure II.5 Arborescence de B&B

Le point fort de cette méthode réside dans le fait qu'elle ne parcourt pas les sous branches dont on peut savoir à priori qu'elles ne permettent pas d'améliorer la solution rencontrée ce qui est établi grâce aux bornes des nœuds, cela permet de trouver de bonnes solution en un temps de recherche raisonnable.

L'efficacité de la méthode B&B a attiré l'attention de nombreux chercheurs. Par conséquent, plusieurs améliorations de l'algorithme B&B ont été proposées, y compris les algorithmes: Branch and Cut (noté B&C) [8], Branch and Price (noté B&P) [9], Branch and Cut and Price (B&C&P).

II.9 Méthode heuristique

II.9.1 Définitions

- étymologie : du grec ancien eurisko, trouver.
- Une heuristique est une technique qui améliore l'efficacité d'un processus de recherche, en sacrifiant éventuellement l'exactitude ou l'optimalité de la solution [3].
- Pour des problèmes d'optimisation (NPcomplets) où la recherche d'une solution exacte (optimale) est difficile (coût exponentiel), on peut se contenter d'une solution satisfaisante donnée par une heuristique avec un coût plus faible.
- méthode approchée conçue pour un problème d'optimisation particulier permettant de trouver des solutions avec un temps de calcul raisonnable.

II.9.2 Les algorithmes gloutons

- C'est une méthode approchée simple et d'usage général. Son principe est qu'à chaque étape durant le processus de recherche, l'option localement optimale est choisie jusqu'à trouver une solution (exacte ou non). Les choix faits durant le processus de recherche ne sont jamais remis en cause (pas de retour arrière).
- Exemple: (voir la figure II.6)

On veut totaliser une somme d'argent S en utilisant un nombre minimal de pièces appartenant à un ensemble donné.

A chaque étape on choisit la plus grande pièce <= S tout en mettant à jour la nouvelle somme (S – la pièce choisie):

Pour S=257 DA et pièces={100DA, 50DA, 20DA, 10DA, 5DA, 2DA, 1DA}

- à l'étape 1 :

S1 = 257 , on choisit 1 pièce de 100DA

- à l'étape 2 :

S2 = 157 , on choisit 1 pièce de 100DA

- à l'étape 3 :

S3 = 57 , on choisit 1 pièce de 50DA

- à l'étape 4 :

S4 = 7 , on choisit 1 pièce de 5DA

- à l'étape 5 :

S5 = 2 , on choisit 1 pièce de 2DA

- La solution trouvée est donc:

2x100 DA , 1x50 DA , 1x5 DA et 1x2 DA

Figure II.6 les étapes de l'algorithme glouton

II.10 CONCLUSION

Dans ce chapitre, nous avons abordé quelques notions sur l'aide à la décision et nous avons aussi introduits les problèmes du sac à dos et ses variantes.

Ensuite nous avons abordé les méthodes exactes et les méthodes heuristiques .

Dans le chapitre suivant, nous allons présenter l'implémentation du logiciel pour les deux algorithmes Branch and Bound et Glouton avec des tests et comparaisons.

CHAPITRE III

Implémentation
et
comparaison des résultats

Chapitre III : Implémentation et comparaison des résultats

III.1 Introduction

Notre allons présentés dans ce chapitre l'implémentation des deux solutions pour résoudre le problème des MDknapsacs dans le cadre de l'optimisation combinatoire qui sont les algorithmes Branch and Bound et Glouton et les résultats obtenues après la simulation des deux méthodes.

Nous avons opté pour la plate-forme JADE (Java Agent Développement Framework) afin de simuler l'ensemble des interactions dans le système.

III.2 Un système multi-agents

« On appelle système multi-agent (ou SMA), un système composé des éléments suivants :

- Un environnement E, c'est à dire un espace disposant généralement d'une métrique.
- Un ensemble d'objets O. Ces objets sont situés, c'est à dire que, pour tout objet, il est possible à un moment donné, d'associer une position dans E.
- Ces objets sont passifs, c'est à dire qu'ils peuvent être perçus, crées, détruits et modifiés par les agents.
- Un ensemble A d'agents, qui représentent les entités actives du système.
- Un ensemble de relations R qui unissent les objets et les agents entre eux.
- Un ensemble d'opérations Op permettant aux agents A de percevoir, produire, consommer, transformer et manipuler les objets de O.
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers. » [10].

« Un Système Multi-Agents (SMA) comporte plusieurs agents qui interagissent entre eux dans un environnement commun. Certains de ces agents peuvent être des personnes ou leurs représentants (avatars), ou même des machines mécaniques. S'il

y a moins de trois agents, on parle plutôt d'interaction homme/machine, ou machine/machine que de systèmes multi-agents.» [11]

III.3 Propriétés clés des Agents

Autonomie: un agent peut agir sans l'intervention directe d'un tiers (agent ou humain) et contrôler ses actions ainsi que son état interne.

Réactivité: un agent perçoit son environnement et répond aux changements qui s'y produisent en un temps raisonnable. L'environnement peut être le monde physique, un utilisateur via une interface graphique, d'autres agents, un système d'information (internet), ...

Communication: un agent peut communiquer avec d'autres agents ainsi qu'avec des utilisateurs humains.

Aptitude sociale: un agent peut interagir avec d'autres agents de façon coopérative ou compétitive pour atteindre ses objectifs.

Pro-activité: l'agent est capable, sur sa propre initiative, de se fixer des buts pour atteindre ses objectifs (opportuniste).

Agent Intelligent: un agent est « intelligent » s'il est capable de réaliser des actions flexibles et autonomes pour atteindre les objectifs qui lui ont été fixés. La flexibilité correspond aux propriétés de Réactivité, Pro-activité, Aptitudes sociales.

Propriétés additionnelles :

Apprentissage: un agent peut mémoriser ses expériences et adapter son comportement en conséquence.

Mobilité: un agent peut se déplacer d'une machine à une autre et éventuellement se dupliquer.

III.4 Domaines d'application des agents

Systèmes ouverts (Exemple Internet).

- Exigences : négociation, coopération, entremetteur, annuaire, appels d'offres.

❖ **Systèmes distribués par nature:** (géographique, fonctionnelle, humaine).

❖ **Systèmes Complexes:** société d'entités coopérantes, robotique collective.

- Les agents apportent modularité, abstraction, possibilité de simulation.

❖ **Assistant expert** (délégation). Exigences :

✓ **autonomie:** à partir d'une spécification vague et imprécise, il doit déterminer comment le problème peut être résolu et ensuite le résoudre sans une intervention constante de l'utilisateur.

✓ **Pro-activité** : il ne doit pas attendre la fin de chaque action utilisateur pour agir mais faire des suggestions à l'utilisateur.

✓ **Adaptabilité**: il doit connaître les préférences de l'utilisateur et leur évolution et les intégrer dans son comportement.

❖ **Système légataire** (agentification).

Nous avons opté pour la plate-forme JADE (Java Agent Development Framework) afin de simuler l'ensemble des interactions dans le système.

III.5 Topologie du réseau utilisé

Nous avons opté pour un type spécifique de réseau, celui des réseaux ad-hoc pour son architecture sans infrastructure et sa facilité d'adaptation.

Nous supposons que les Pus et les Sus sont répartis sur des zones géographiques bien précises et qu'ils ne changent pas de zone pendant la négociation. Le CPU sera l'intermédiaire entre les SUs et les PUs.

La figure II.1 illustre la topologie du réseau dont nous nous sommes servis, composée de plusieurs PUs, un CPU, et plusieurs SUs.

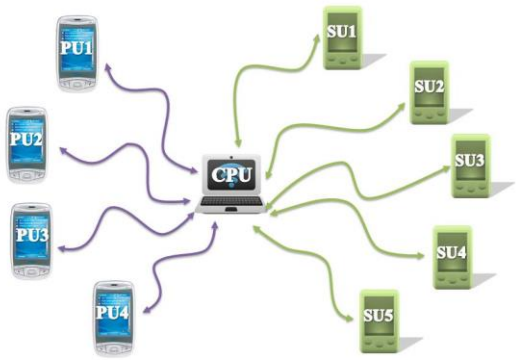

Figure III.1: Topologie du réseau (mode ad hoc).

III.6 Présentation du scenario

Pour la réalisation et l'implémentation de notre application, nous nous sommes centrés sur un modèle typique de négociation : celui de « plusieurs à plusieurs ».

Etant donné qu'un nombre important de PUs chaque PU est caractérisé par un nombre de canaux disponible rend les négociations fastidieuses vis-à-vis des SUs ou chaque SU est caractérisé par un poids et un profit , nous avons décidé d'implémenter un CPU sur la zone géographique cible. Le CPU détient, en permanence et en temps réel, l'ensemble des informations relatives à l'ensemble des PUs et des Sus présents à l'intérieur de sa sphère.

Dans notre cas, les SUs et les PUs négocient leur accord par l'intermédiaire d'un CPU sur une base de multiples critères tels que le profit, le nombre de canaux, et le nombre de canaux disponible. Cette multitude de paramètres nous a conduits à introduire une méthode exacte qui est basée sur la technique branch and bound.

La figure suivante nous montre que les SUs va initier des négociations avec les PUs a l'intermédiaire d'un CPU. Le CPU applique la technique branch and bound pour choisir l'offre la mieux adaptée au besoin des SUs et des PUs, puis envoie aux SUs et Pus les offres désignées.

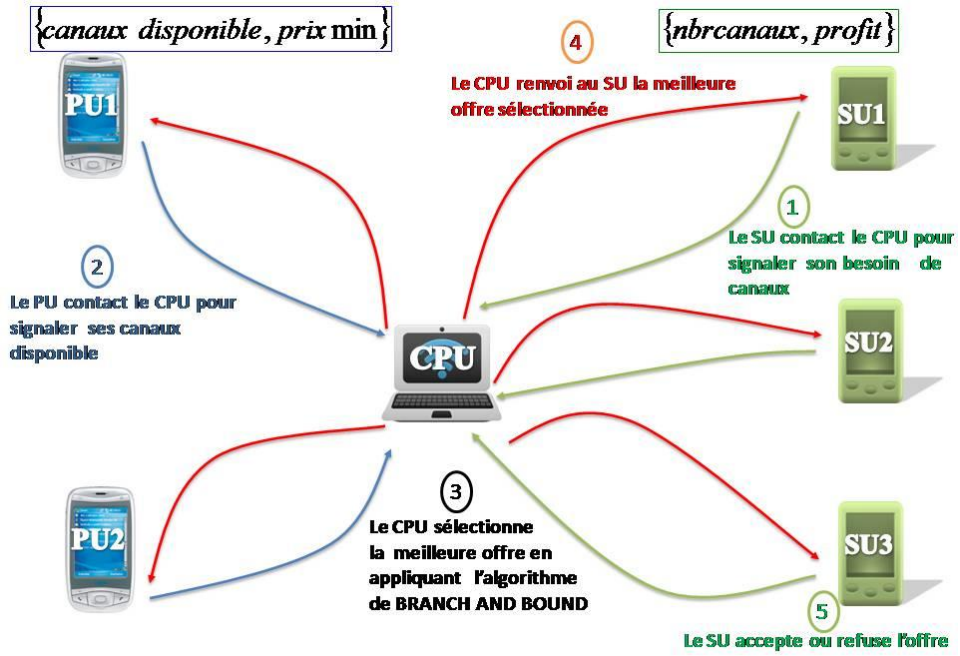

Figure III.2: Scénario proposé.

III.7 Présentation des algorithmes utilisés

Notre application s'établit au tour de 2 critères {nombre de canaux libres, profit d'un canal.

```
Entrée: Une instance d'un problème du knapsack ;
Sortie: Une solution optimale x

   1. j:= 1; Meilleure Valeur:= 0;
   2. Calcul de la borne supérieure Upper Bound que l'on peut atteindre depuis j ;
   3. Si (MeilleureValeur) alors ;
      //   Développer une branche de l'arborescence en profondeur d'abord pour
obtenir une solution réalisable x0 ;
         j:= j + 1 ;
         Si (Z(x0) _ MeilleureValeur) alors
          x = x0 ;
         MeilleureValeur:= Z(x0) ;
          FinSi
       FinSi
    4. j:= maxfl: l < j et x0
     l = 1g
    Tant que j existe faire
      x0
        j:= 0 ;
    Recalculer la borne supérieure ;
      Fin Tant que
    5. x0:= x.
```

Figure III.3: Algorithme de Branch and Bound.

III.8 Outils utilisés pour l'implémentation de l'application

Pour l'implémentation d'un système multi-agent, nous devons utiliser une plate-forme multiagent spécifique qui nous permet la construction et la mise en service d'agents au sein d'un environnement particulier. Dans le cadre de notre application, nous avons opté, parmi plusieurs plates-formes pour « JADE » (Java Agent DEvelopment framework) en raison de sa simplicité d'utilisation.

De plus, notre application se constitue de plusieurs agents : un agent par chaque SU, un agent par CPU et un agent par chaque PU.

III.8.1 JADE

JADE est une plate-forme multi-agent créée par le laboratoire TILAB et décrite par Bellifemine et al. Dans [10]. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA3 (Foundation for Intelligent Physical Agents). Elle est implémentée en JAVA et fournit des classes qui implantent « JESS » pour la définition du comportement des agents. JADE possède trois modules principaux (nécessaire aux norms FIPA).

- **DF** : Director Facilitor » pourvoit un service de « pages jaunes» à la plate-forme.
- **ACC** : Agent Communication Channel » gère la communication entre les agents.
- **AMS** « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.
- Ainsi, que d'autres modules :
- **Un runtime Environment**: l'environnement où les agents peuvent vivre. Il doit être activé pour pouvoir lancer les agents.
- **Une librairie de classes** : que les développeurs utilisent pour écrire leurs agents.
- **Une suite d'outils graphiques** : qui facilitent le débogage, la gestion et la supervision de la plate-forme des agents.

La FIPA est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes.

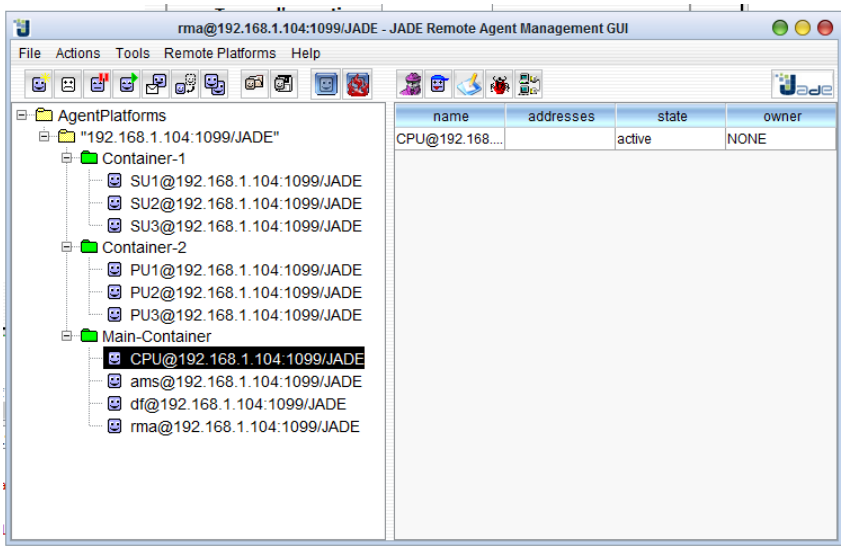

Figure III.4: Interface graphique (jade GUI).

- **Dummy Agent (DA)**: cet outil offre la liberté aux utilisateurs d'interagir avec les agents JADE d'une façon particulière. L'interface permet la composition et l'envoi de messages ACL, et maintient une liste de messages ACL envoyés et reçus
- **Sniffer Agent (SA)** : quand un utilisateur décide d'épier un agent ou un groupe d'agents, il se sert d'un agent Sniffer. Chaque message partant ou allant vers ce groupe est capté et affiché sur l'interface du sniffer. L'utilisateur peut voir et enregistrer tous les messages, pour éventuellement les analyser plus tard. L'agent peut être lancé du menu du RMA ou de la ligne de commande comme suit : Java jade.Boot sniffer :
- **Introspector Agent** : Cet agent permet de gérer et de contrôler le cycle de vie d'un agent s'exécutant et de la file de ses messages envoyés et reçus.

III.8.2 JFreechart

JFreeChart est une API Java open source sous licence LGPL (Lesser General Public Licence) qui permet d'afficher des données statistiques sous la forme de graphiques et de diagrammes de très bonne qualité. Elle possède plusieurs formats dont le camembert, les barres ou les lignes et propose de nombreuses options de configuration pour personnaliser le rendu des graphiques. Elle peut s'utiliser dans des applications

standalones ou des applications web et permet également d'exporter le graphique sous la forme d'une image.

III.9 Présentation de l'application

Dans cette section, nous allons décrire le fonctionnement de notre application, les configurations faites ainsi que les résultats obtenus. Nous serons amenés à parler aussi au sujet des comparaisons effectuées avec les deux methodes excate(B&B) et heuristique(Gloutonne).

La réalisation de cette application a été faite sous Netbeans version 8.1 en se basant sur JADE version 4.4.0 avec l'utilisation de l'outil JFreeChart version 1.0.14 pour créer des graphiques et des diagrammes de très bonne qualité.

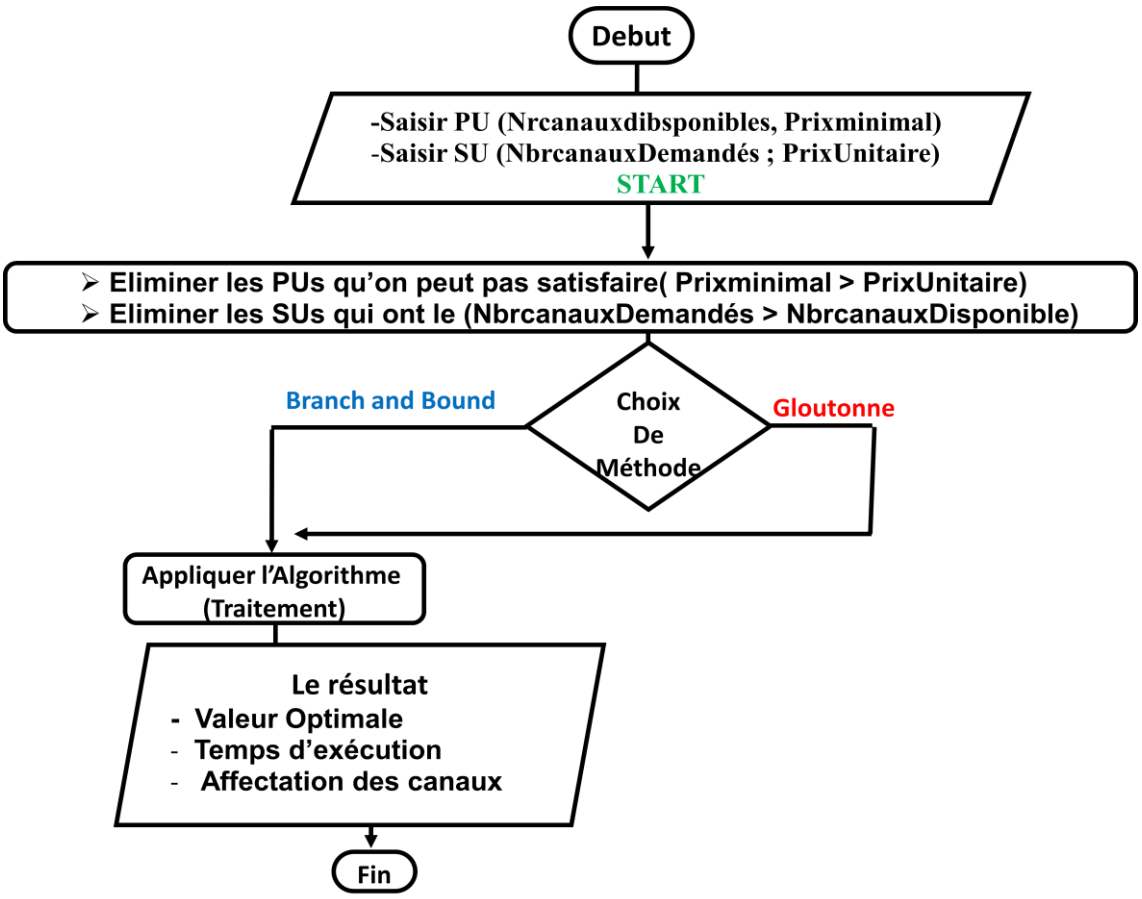

Figure III.5: Organigramme de l'application

Notre application se compose de plusieurs SUs, plusieurs Pus et un CPU contiendra les informations sur les paramètres relatifs à l'ensemble des PUs et des SUs se trouvant dans sa zone géographique. Chaque PU détiendra deux paramètres (nombre de canaux disponibles, prix) et chaque SU détiendra également deux paramétres (nombre de canaux demandés, prix unitaire).

Au lancement du test chaque PU et chaque SU envoient au CPU les informations sur ses paramètres (critères). Quant au CPU, il chargera les informations reçus dans des tableaux dynamiques dont les clés sont les noms des PUs et des SUs.

L'arraylist est mis à jour à chaque changement de paramètre chez un PU ou bien SU.

Voici à quoi ressemble l'interface d'accueil de notre application :

Figure III.6: interface d'accueil

En cliquant sur le Bouton START, l'interface de simulation s'affiche, nous distinguons 5 onglets :

- Onglet de démarrage du test:

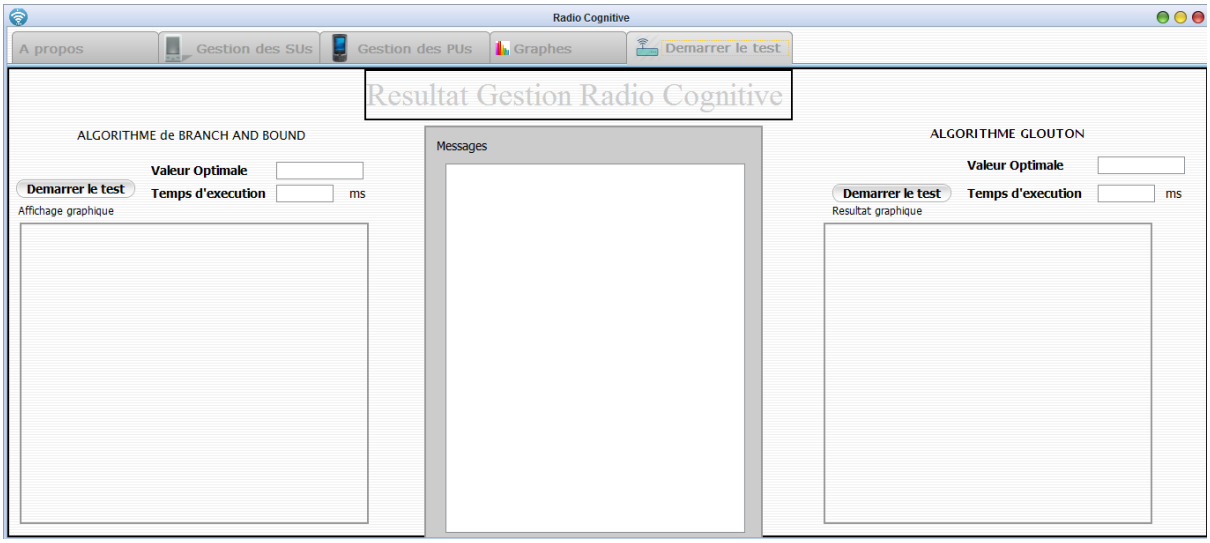

Figure III.7: Onglet de démarrage test.

- Onglet de gestion des SUs

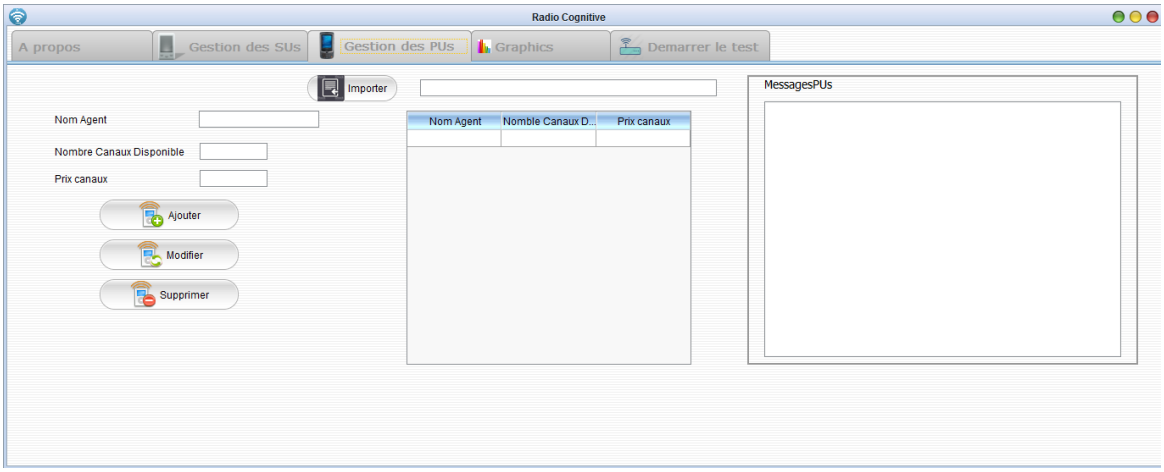

Figure III.8: Onglet de Gestion des SUs.

- Onglet de gestion des PUs

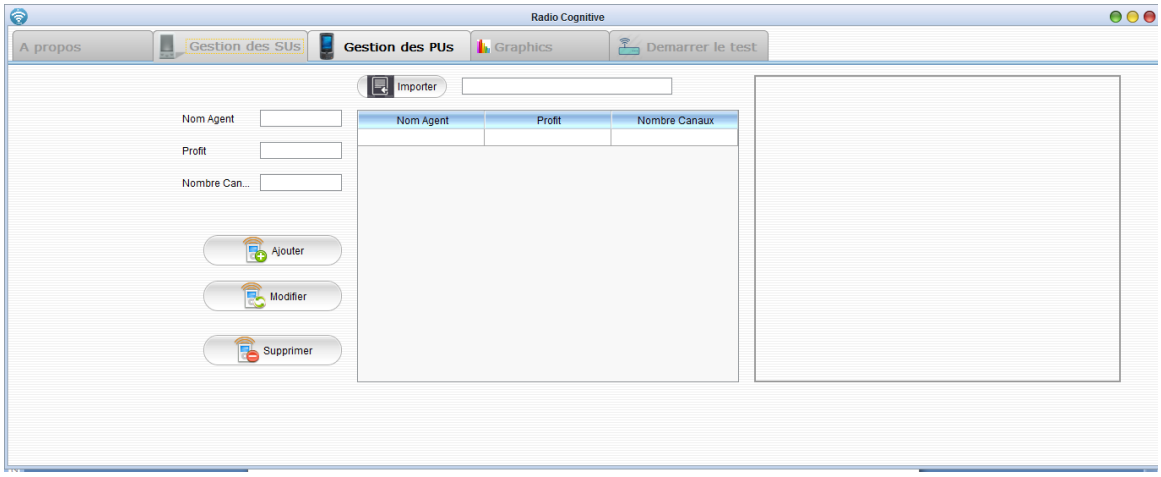

Figure III.9: Onglet de Gestion des PUs.

- Onglet pour afficher les Graphes :

Figure III.10: Onglet des Graphes.

III.10 Résultats après simulation

III.10.1 interactions entre les SUs et les PUs

Dans le scénario de simulation proposer on utilise 04 SUs et 02 PUs. Dans cette simulation

Les agent SUs	Les agent PUs
SU1 demande(5 canaux , PrixUnitaire 25 DA) SU2 demande(20 canaux , PrixUnitaire 20 DA) SU3 demande(35 canaux , PrixUnitaire 35 DA) SU4 demande(5 canaux , PrixUnitaire 60 DA)	PU1 possède (40 canaux,Prixmin 15 DA) PU2 possede (5 canaux,prixmin 10 DA)

La figure III.11 donne une idée sur notre simulation.

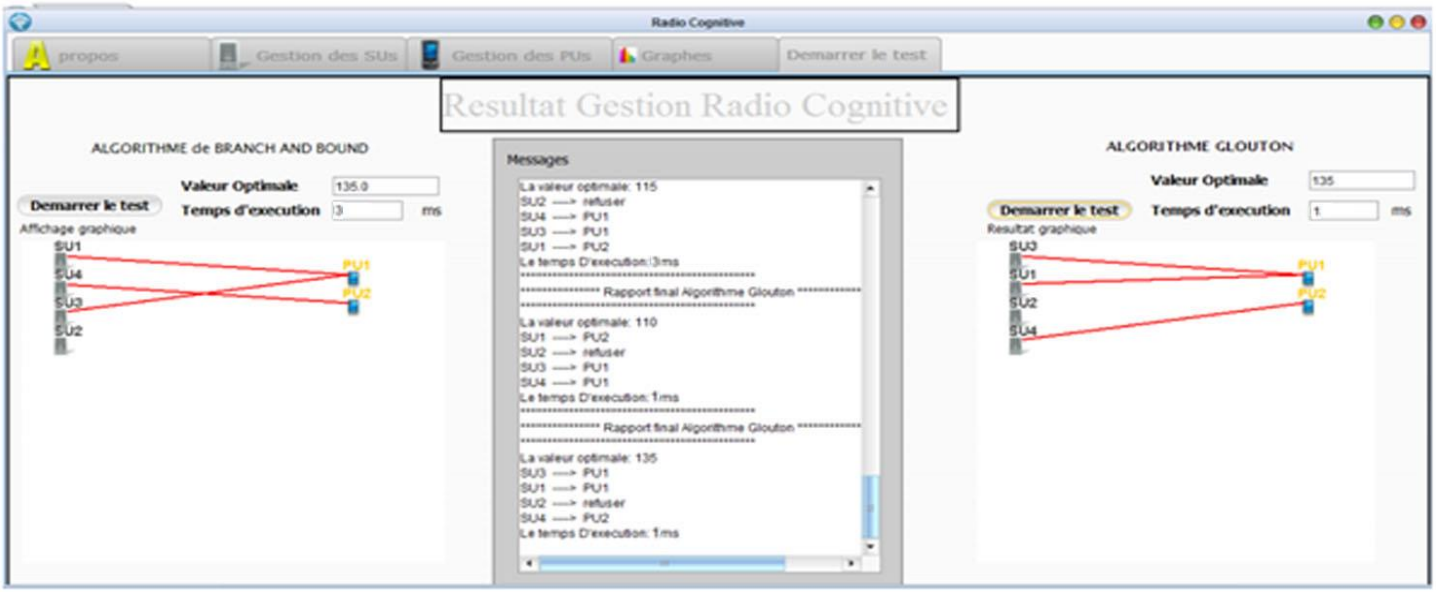

Figure III.11: lancement du test de simulation.

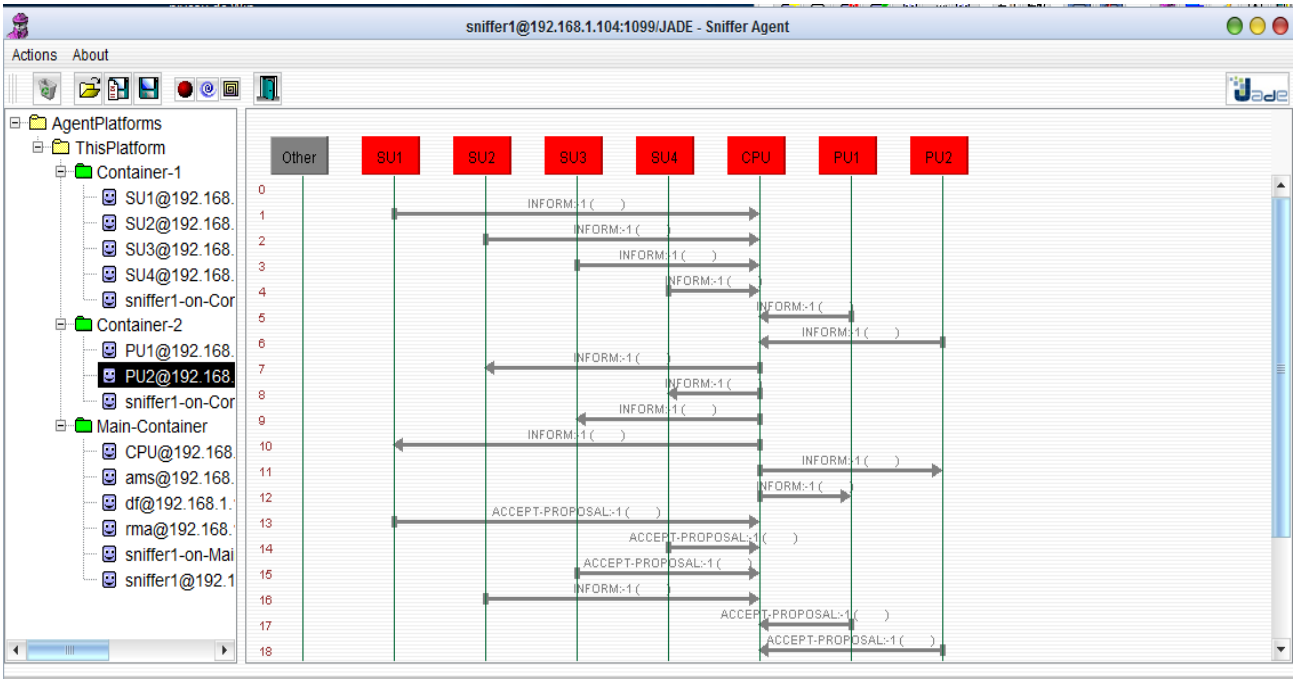

Figure III.12: Agent sniffer.

D'après la Figure III.12, les SUs informent le CPU pour signaler son besoin de canaux, et les PUs informent le CPU pour signaler ses canaux disponible, le CPU sélectionne la meilleure offre en appliquant l'algorithme approprier, ensuite renvoi aux SUs et Les PUs la meilleur offre sélectionnée, après les SUs acceptent ou refusent l'offre.

III.10.2 Résultats 01 après simulation

D'après la Figure III.13 nous remarquons que le taux de satisfaction dans le cas d'utilisation le l'algorithme de B&B est meilleure que celle de l'algorithme Glouton.

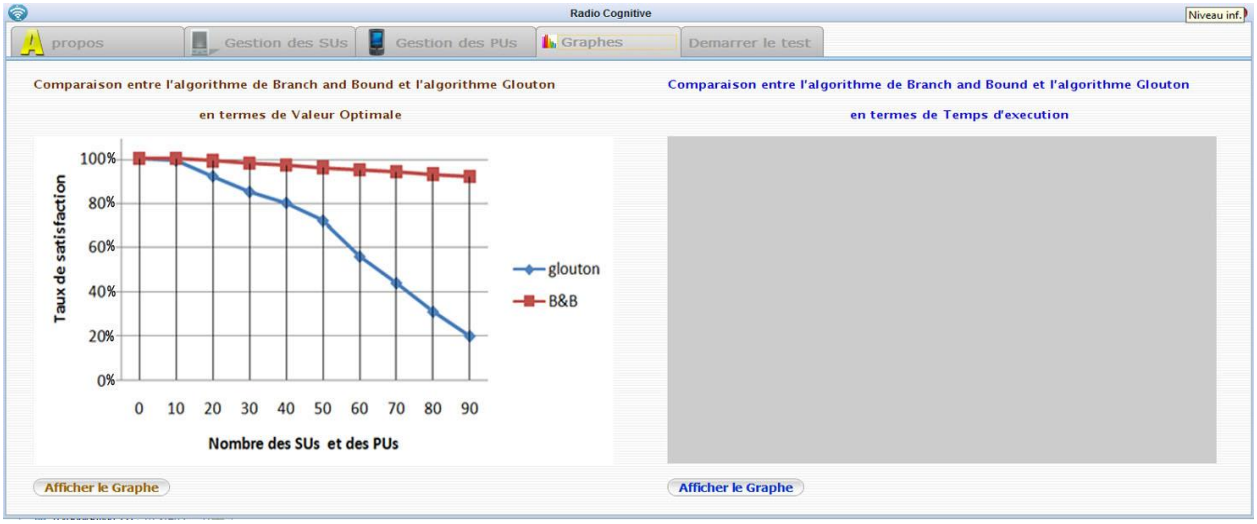

Figure III.13: Comparaison entre l'algorithme de B&B et l'algorithme Glouton

En termes de valeur optimale.

III.10.3 Résultats 02 après simulation

D'après la Figure III.14 nous remarquons que le temps d'exécution dans le cas d'utilisation le l'algorithme de B&B augmente plus rapide que celle de l'algorithme Glouton.

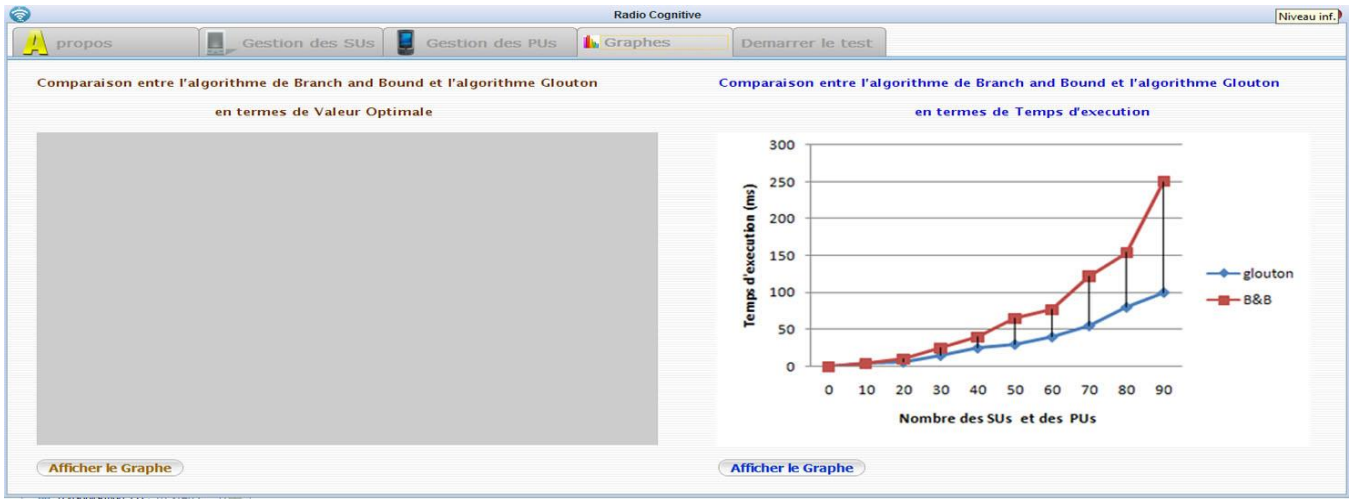

Figure III.14: Comparaison entre l'algorithme de B&B et l'algorithme Glouton

En termes de Temps d'exécution.

III.11 Conclusion :

Dans ce chapitre, nous avons étudié les deux méthodes exactes et heuristiques pour résoudre notre problématique. A travers les simulations que nous avons réalisées, nous avons pu constater l'intérêt des méthodes exactes (B&B) par apport aux méthodes heuristiques dans le contexte d'un réseau de radio cognitive vue les résultats obtenues (le profit maximum coté PUs, temps d'exécution meilleure).

Conclusion Générale

Conclusion Générale et perspectives

L'informatique autonome a pour but de diminuer la complexité en utilisant des technologies pour gérer un système en minimisant l'intervention d'opérateurs humains. Son objectif est de créer des environnements ayant la capacité de s'autogérer et de s'adapter aux changements dynamiques. L'autonomie dans les réseaux de radio cognitive est principalement focaliser sur la gestion du spectre qui permet l'amélioration du débit sans pour autant à dégrader les communications des autres. Un réseau autonome est caractérisé par quatre points bien distincts: l'auto-optimisation, l'auto-configuration, l'auto-protection et l'auto-guérison.

Dans notre PFE nous nous intéressons à l'auto-optimisation en utilisant la méthode de B&B pour résoudre le problème de type explosion combinatoire en calculant des bornes supérieures. Les résultats obtenus à travers la simulation avec JADE ont montré l'intérêt d'appliquer les principes de l'aide à la décision dans le cadre d'un réseau de radio cognitive. On suggère l'utilisation des autres méthodes comme celles des méta-heuristiques qui contiennent plusieurs algorithmes (PSO, Colonie d'abeilles, Colonie de Fourmies) à fin de voir l'utilité de celles-ci dans le contexte de la radio cognitive.

Reference Bibliographique

[1] J. Mitola, « Cognitive radio architecture : The Engineering Foundations of Radio XML »

[2] ET Docket 03-222 « Notice of proposed rule making and order » FCC, December 2003.

[3] « Badr Benmammar. Présentation de la radio cognitive » . 3rd cycle. 2012.

[4] J. Jorge. « Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires ». Thèse de doctorat. Université de Nante, France. 2010.

[5] M. Kong, P. Tian «Apply the Particle Swarm Optimization to the Multidimensional Knapsack Problem ». In proc. L. Rutkowski et al. (Eds.): ICAISC 2006, LNAI 4029, pp. 1140 -1149. Springer, 2006.

[6] P.C. Chu et J.E. Beasley. « A Genetic Algorithm for the Multidimensional Knapsack Problem. Journal of Heuristics ». Vol. 4, N° 1, pp. 63-86, 1998.

[7] J. Teghem. « Programmation Linéaire ». Ellipses, 1996.

[8] M. Padberg et G. Rinaldi. « A branch and cut algorithm for the resolution of large- scale symmetric traveling salesman problem ». SIAM Review. Vol. 33, N°1, pp. 60- 100, 1991.

[9] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. « Branch-and-price: column generation for solving huge integer programs. Operations Research ». Vol. 46, N° 3, pp 316-329, 1998.

[10] Jacques Ferber, « les systèmes multi-agents », InterEditions,1995.

[11] J.P. Briot et Y. Demazeau (Editeurs) « Principes et architecture des Systèmes multi-agents ». Hermes, 2001.

[12] A. H. Land et A. G. Doig. « An automatic method for solving discrete programming problems». Econometrica, vol. 28, pages 497–520, 1960. (Cité pages 9 et 26).

Résumé

Dans cet œuvre nous apportons de l'aide à la décision aux utilisateurs primaires d'un réseau radio cognitive. Nous modélisons notre problème comme un problème du sac à dos multidimensionnel. Pour le résoudre, l'algorithme par séparation et évaluation (PSE) est appliqué, c'est une méthode générique de résolution de problèmes d'optimisation combinatoire, qui fournit une solution exacte. Pour palier au problème de l'explosion combinatoire, notre stratégie est basée sur deux points cruciaux. Le premier étant l'utilisation d'une borne supérieure lors du parcours des solutions réalisables. Le second est d'adopter des agents, répartis sur la même zone géographique. Nous avons aussi introduit l'algorithme glouton qui représente une méthode dite approchée pour une évaluation et comparaison des résultats.

Mots de clés : Radio Cognitive. Sac à dos Multidimensionnel. Algorithme Par Séparation et Evaluation(PSE). Algorithme Glouton. Systèmes multi-agents. JADE.

الخلاصـــــة

قدمنا في هذا المشروع دعم اتخاذ القرار للمستخدمين الأوليين للشبكات راديو إدراكية. صممنا هذه المشكلة في شكل مشكلة حقيبة الظهر متعددة الإبعاد. من أجل الوصول إلى حل استخدمنا خوارزمية التفريع و التقييم، تستخدم هذه الخوارزمية لحل المشكلات المتداخلة، و تسمح بالحصول على نتائج دقيقة. لكن لتجنب التوسع التداخلي استعملنا إستراتيجية تعتمد على نقطتين أساسيتين. الأولى و هي استعمال حد أقصى عند تحديد الحلول الممكنة. الثانية تتمثل في استعمال عملاء منتشرين على نفس المنطقة الجغرافية.

لتدعيم هذه الدراسة أضفنا خوارزمية كلوتون التي تعطي نتائج تقريبية.

الكلمات المفتاحية: راديو إدراكية، اتخاذ القرار، نظام متعدد العملاء، محفظة الظهر متعددة الأبعاد، التفريع والتقييم، كلوطون.

Abstract

In this work we bring of the decision-making aid to the primary education users of a radio operator network cognitive. We model our problem like a problem of the multidimensional Knapsack. To solve it, the algorithm Branch and Bound (B&B) are applied, it is a generic method of resolution of problems of combinative optimization, which provides an exact solution. For stage with the problem of the combinative explosion, our strategy is based two point crucial. The first being the use of an upper limit (Upper Bound) at the time of the course of the realizable solutions. Second is to adopt agents, distributed on the same geographical zone.

We also introduced the greedy algorithm which represents a so-called approximate method for evaluation and comparison of the two solutions.

Key works: Cognitive Radio, Multi-Agents system, MDKnapsack, Branch and Bound, Greedy.

ANNEXES

********** Code de l'Agent SU **********

```
public class SUU extends Agent{
@Override
protected void setup(){
try{
//the file path
File fileSU = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\Visual
isationSU.txt");
//if the file not exist create one
                    if (!fileSU.exists()) {
                    fileSU.createNewFile();
                    }

                    BufferedWriter bw = null;
                    FileWriter fw = null;
try{
   fw = new FileWriter(fileSU, true);
                        //enregistrer le service SU dans le DF
                DFAgentDescription dfd = new DFAgentDescription();
                dfd.setName(getAID());
                ServiceDescription sd = new ServiceDescription();
                sd.setType("achat_canal");
                sd.setName("JADE-achat_canal");
                dfd.addServices(sd);
                try {
                   DFService.register(this, dfd);
                }
                catch (FIPAException fe) {
                }
                // l'agent su recuperer ces arguments
                //System.out.println("Je suis l'"+ getAID().getName());

                int  nbc=0;
                int profit=0;
                String nom_agent="";
                Object[] args = getArguments();
                if (args != null) {
                   nom_agent = (String) args[0];
                   profit = (int) args[1];
                   System.out.println("profit  proposer par l'agent SU= "+args[1]);
                   bw.write(" "+args[1]+ "profit de canal proposer par l'agent SU=  ");
                   bw.newLine();
                   nbc= (int) args[2];
                   System.out.println("nombre canaux demander par l'agent SU= "+args[2]);
                    }
                ACLMessage msg11 = new ACLMessage(ACLMessage.REQUEST);
                msg11.setContent("" + nom_agent + "," + profit + "," + nbc);
                 bw.write(nom_agent + "," + profit + "," + nbc );
                 bw.newLine();
                msg11.addReceiver(new AID("AgentCPU",AID.ISLOCALNAME));
                msg11.setConversationId("achat_request");

                send(msg11);
    ///////////////recevoir le resultat de la part de l'agent cpu //////////////////
```

```
                addBehaviour(new CyclicBehaviour(this){
                   @Override
                   public void action(){
                      //System.out.println("Moi l'agent SU j'attend la reponse ...");
                      MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
                      ACLMessage msg = receive(mt);
                     // System.err.println("le message recu: "+msg.getContent());
                      if (msg!= null){
                         try {
                            ACLMessage reply = msg.createReply();
                            reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
                            reply.setContent("l'offre est acceptée");
                            myAgent.send(reply);
                         } catch (Exception e) {
                          ACLMessage reply = msg.createReply();
                            reply.setPerformative(ACLMessage.FAILURE);
                            reply.setContent("l'offre n'est pas acceptée");
                            myAgent.send(reply);
                         }
                         String nom_Agent;
                          nom_Agent = msg.getSender().getLocalName();
                         if (nom_Agent.contains("AgentSU"))
                         {
                            try{
                               //the file path
                               File file = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\Visual
isationSU.txt");
 //if the file not exist create one
                if(file.exists()){
                   file.delete();
                   file.createNewFile();
                               }
  try (FileWriter fw = new  FileWriter(file, true); BufferedWriter bw = new BufferedWriter(fw))
{
                                  bw.newLine();
                                  bw.write(msg.getContent());
                                  //close FileWriter
                                  }
                            }catch(Exception ex){
                               ex.printStackTrace();
                            }   }    }
                      else{
                         block();
                      }
                   }  });
                         bw.close();
                         fw.close();

}catch(IOException iOException) {}
} catch (Exception e) {                        }     }
}
```

********** Code de l'Agent PU**********

```
public class PUU extends Agent{
@Override
protected void setup(){
   ////////////////////debut insertion dans le fichier text
try{
//the file path
                    File filePU = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\Visual
isationPU.txt");
//if the file not exist create one
                    if (!filePU.exists()) {
                        filePU.createNewFile();
                      }
                    BufferedWriter bw = null;
                    FileWriter fw = null;
try{
   fw = new FileWriter(filePU, true);
                        bw = new BufferedWriter(fw);
                        bw.newLine();
                DFAgentDescription dfd = new DFAgentDescription();
                dfd.setName(getAID());
                ServiceDescription sd = new ServiceDescription();
                sd.setType("achat_canal");
                sd.setName("JADE-achat_canal");
                dfd.addServices(sd);
                try {
                   DFService.register(this, dfd);
                }
                catch (FIPAException fe) {
                }
                // l'agent su recuperer ces arguments
          String nomagent="";
          int  nbrcanauxdispo=0;
          int prixpu=0;
          //double prixu=0;
          Object[] args = getArguments();
          if (args != null) {
             nomagent=(String) args[0];
             nbrcanauxdispo = (int) args[1];
Projet_Fin_etude.jTVisualisation_PU.append("nombre canaux disponible pour l'agent PU=
"+args[1]+"\n");
             prixpu= (int) args[2];
 Projet_Fin_etude.jTVisualisation_PU.append("prixu de canaux proposer par  agentsu=
"+args[2]+"\n");
          }
          ACLMessage msg11 = new ACLMessage(ACLMessage.REQUEST);
          msg11.setContent(""+nomagent+","+nbrcanauxdispo+","+prixpu);
          msg11.addReceiver(new AID("AgentCPU",AID.ISLOCALNAME));
          msg11.setConversationId("achat_request");
          send(msg11);
                DFAgentDescription templ = new DFAgentDescription();
                MessageTemplate mt1 =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
                ACLMessage msg = receive(mt1);
```

```
            addBehaviour(new CyclicBehaviour(this){
               @Override
               public void action(){
                  //System.out.println("Moi l'agent PU j'attend la reponse ...");
                  MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
                  ACLMessage msg = receive(mt);
                 // System.err.println("le message recu: "+msg.getContent());
                  if (msg!= null){
                     try {
                        ACLMessage reply = msg.createReply();
                        reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
                        reply.setContent("l'offre est acceptée");
                        myAgent.send(reply);
                     } catch (Exception e) {
                      ACLMessage reply = msg.createReply();
                        reply.setPerformative(ACLMessage.FAILURE);
                        reply.setContent("l'offre n'est pas acceptée");
                        myAgent.send(reply);
                     }
                      String nom_Agent;
                     nom_Agent = msg.getSender().getLocalName();
                     if (nom_Agent.contains("AgentPU"))
                     {
                         try{
                           //the file path
                           File file = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\PUF1.
txt");
                           //if the file not exist create one
                           if(!file.exists()){
                              file.createNewFile();
                           }
                           try (FileWriter fw = new  FileWriter(file, true); BufferedWriter bw =
new BufferedWriter(fw)) {
                              bw.newLine();
                              bw.write(msg.getContent());
                              //close FileWriter
                                                      }
                        }catch(Exception ex){
                           ex.printStackTrace();
                        }           }           }
                  else{
                     block();
                  }  }   });

            bw.close();
            fw.close();

}catch(IOException iOException) {}
} catch (Exception e) {                       }

}
}
```

********** Code de l’ Agent CPU **********

```
public class AgentCPU extends Agent {

    ArrayList<SUVSPU> toutpusu = new ArrayList();
    ArrayList<SUVSPU> toutpusu1 = new ArrayList();
    public String nom_Agent;
    private int toutagents;
    public int c=0;
    public static long  te;
   static  Algorithm alg;

    @Override
    protected void setup() {

            File file = null;
        try {
            sleep(4000);
            file = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\SUF.t
xt");
            //if the file not exist create one
            if (!file.exists()) {
                file.createNewFile();
            }
            FileWriter fw = new FileWriter(file);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write("#nom_agent,profit,nbrcanal");
            bw.close();
            //close FileWriter
            fw.close();
            File file2 = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\PUF.t
xt");
            //if the file not exist create one
            if (!file2.exists()) {
                file2.createNewFile();
            }

            FileWriter fw2 = new FileWriter(file2);
            BufferedWriter bw2 = new BufferedWriter(fw2);
            bw2.close();
            //close FileWriter
            fw2.close();
            DFAgentDescription template1 = new DFAgentDescription();
            ServiceDescription sd1 = new ServiceDescription();
            sd1.setType("achat_vente_canal");
            sd1.setName("JADE-achat_vente_canal");
            template1.addServices(sd1);
            DFService.register(this, template1);
            DFAgentDescription[] list1 = DFService.search(this, template1);
            //DFAgentDescription[] list2 = DFService.search(this, template2);
            AID[] agents1 = new AID[list1.length];
        } catch (InterruptedException interruptedException) {
        } catch (IOException iOException) {
        } catch (FIPAException fIPAException) {
```

```
    }

        addBehaviour(new CyclicBehaviour(this){
            @Override
            public void action(){
                // pour recevoire un message
                try {

                    MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.REQUEST);

                    ACLMessage msg = receive(mt);
                     if (msg != null) {

                       String nom_Agent;

                       nom_Agent = msg.getSender().getName();

                       if (nom_Agent.contains("SU")) {
                          File file = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\SUF.txt");
                          //if the file not exist create one
                          if (!file.exists()) {
                             try {
                                file.createNewFile();
                             } catch (IOException ex) {
            Logger.getLogger(AgentCPU.class.getName()).log(Level.SEVERE, null, ex);
                             }
                          }
                          try {
                             FileWriter fw = new FileWriter(file, true);
                             BufferedWriter bw = new BufferedWriter(fw);
                             bw.newLine();

                             System.out.println(msg.getContent());
                             bw.write(msg.getContent());
                             bw.close();
                             //close FileWriter
                             fw.close();
                          } catch (IOException iOException) {
                          }

                       } else {

                          //debut creation fichier
                          //the file path
                          File file = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\PUF.txt");
                          //if the file not exist create one
                          if (!file.exists()) {
                             try {
                                file.createNewFile();
 } catch (IOException ex) {
                    Logger.getLogger(AgentCPU.class.getName()).log(Level.SEVERE, null, ex);
                             }
```

```
                        }
                        try {
                            FileReader fr = new FileReader(file);
                            BufferedReader br = new BufferedReader(fr);
                            boolean debut = false;
                            String line;
                            if ((line = br.readLine()) == null) {
                                debut = true;
                            } else {
                                debut = false;
                            }

                            br.close();
                            fr.close();
                            FileWriter fw = new FileWriter(file, true);
                            BufferedWriter bw = new BufferedWriter(fw);
                            if (debut == false) {
                                bw.newLine();
                            };

                            bw.write(msg.getContent());
                            bw.close();
                            //close FileWriter
                            fw.close();

                        } catch (IOException iOException) {
                        }

                        //the file path
                        File file1 = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\PUFN
ame.txt");
                        //if the file not exist create one
                        try {
                            if (!file1.exists()) {
                                file1.createNewFile();
                            }
                            FileReader fr = new FileReader(file1);
                            BufferedReader br = new BufferedReader(fr);
                            boolean debut = false;
                            String line;
                            if ((line = br.readLine()) == null) {
                                debut = true;
                            } else {
                                debut = false;
                            }
                            br.close();
                            fr.close();
                            FileWriter fw = new FileWriter(file1, true);
                            BufferedWriter bw = new BufferedWriter(fw);
                            if (debut == false) {
                                bw.newLine();
                            };
                            bw.write(msg.getSender().getLocalName());
                            bw.close();
```

```
                    fw.close();
                } catch (IOException iOException) {
                }       }

            } else {
                block();
            }
            try {
                if (msg == null) {
                    ArrayList<PU> PUS = new ArrayList<PU>();
                    ArrayList<SU> SUS = new ArrayList<SU>();
                    PUS =
readFilePUS("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\PUF.txt")
SUS =
readFileSUS("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitiv
e\\SUF.txt");

                    for (int i=0; i<SUS.size();i++){
                        for (int j=0;j<PUS.size();j++){
                        if (SUS.get(i).nbrcanal > PUS.get(j).canaldisp){SUS.remove(i); break;}
                    }
                    }
                    int min = SUS.get(0).nbrcanal;

                    for (int i=0; i<SUS.size();i++){
                        if (min>SUS.get(i).nbrcanal){min = SUS.get(i).nbrcanal;}
                    }
                    for (int i=0; i<PUS.size();i++){
                        if (PUS.get(i).canaldisp < min){PUS.remove(i); break;
                        }       }
                     long start = System.currentTimeMillis();
                    Solver sol = new Bruteforce(PUS, SUS);
                    alg = sol.selectAlgorithm();
                    alg.run();
                    long end = System.currentTimeMillis();
                    File file2 = new
File("D:\\projet_fin_etude\\Sma_projet_Radio_Cognitiveaissi1_10\\src\\RadioCognitive\\Visual
isation.txt");
//if the file not exist create one
                    if (!file2.exists()) {
                        try {
                            file2.createNewFile();
                        } catch (IOException ex) {
            Logger.getLogger(AgentCPU.class.getName()).log(Level.SEVERE, null, ex);
                        }           }
                    BufferedWriter bw = null;
                    FileWriter fw = null;
                    try {

                        fw = new FileWriter(file2, true);
                        bw = new BufferedWriter(fw);
                        bw.newLine();
                        //Projet_Fin_etude.V.append("\n");
              ACLMessage msgSU = new ACLMessage(ACLMessage.INFORM);  //ajouter
              ACLMessage msgPU = new ACLMessage(ACLMessage.INFORM); //ajouter
```

```
                for(int i=0; i<alg.solution.length;i++){
                Projet_Fin_etude.V.append(alg.SUS.get(i).nom_agent.substring(5,8) + "  ---->  ");

                            if (alg.solution[i]==0){
                   bw.write("refuser ");
                   Projet_Fin_etude.V.append("refuser "+"\n");
                            //System.out.print("Refuser"+"  ");
                     msgSU.setContent(alg.SUS.get(i).nom_agent + " n'est pas affecter" + " ");
                     msgSU.addReceiver(new AID(alg.SUS.get(i).nom_agent,
AID.ISLOCALNAME));
                     send(msgSU);
                }else{
 msgSU.setContent(alg.SUS.get(i).nom_agent + "  est affecte a " + alg.PUS.get(alg.solution[i]-
1).nomagent + "  ");
                     msgPU.setContent(alg.PUS.get(alg.solution[i]-1).nomagent+" vendre a l'agent
: "+alg.SUS.get(i).nom_agent);
  msgSU.addReceiver(new AID(alg.SUS.get(i).nom_agent, AID.ISLOCALNAME));
msgPU.addReceiver(new AID(alg.PUS.get(alg.solution[i]-1).nomagent,
AID.ISLOCALNAME));
                     send(msgSU);
                     send(msgPU);
                }

               bw.newLine();
                        }
                            te=end-start;
                            bw.newLine();
                            bw.newLine();
                            bw.write("Le temps D'execution: " + (end - start) + "ms");
                            Projet_Fin_etude.jTTempsExecusion.setText(String.valueOf(te));
                            bw.close();

//close FileWriter
                            fw.close();

                          } catch (IOException iOException) {
                          }

                        }
                   } catch (Exception e) {
                   }
                } catch (Exception e) {
                }           }       });

   }    }
```