TABLE DES MATIERES

INTRODUCTION GÉNÉRALE

Les besoins des utilisateurs ont connu une croissance remarquable dans le secteur de télécommunication en général et les réseaux sans fil en particulier. Pour pouvoir répondre à leurs exigences, des chercheurs ont travaillé pour améliorer les techniques utilisées et de nouvelles technologies ont été proposées.

Parmi les technologies qui consistent à satisfaire les utilisateurs sans fil : la radio cognitive. Cette dernière est un nouveau concept qui permet un fonctionnement dynamique et autonome des réseaux sans fil. Son objectif est d'améliorer la gestion du spectre à travers une gestion dynamique de ce dernier.

Une notion est souvent utilisée dans les réseaux de radio cognitive est la notion de clustering. Un de ses objectifs est de réduire la surcharge de communication dans ce type de réseau.

Dans le cadre de ce PFE, nous avons utilisé une méthode d'apprentissage supervisé qui est la méthode k-ppv (k-plus proches voisins) afin de réaliser une affectation rapide des nœuds RC dans un réseau de radio cognitive. Notre objectif est de proposer une version parallèle de cette méthode, et de la comparer avec la version séquentielle en faisant une modélisation basée sur les systèmes multi-agents.

Le reste de ce mémoire est structuré comme suit :

Le chapitre I est consacré aux réseaux de radio cognitive. Nous allons définir la notion de radio cognitive ensuite, expliquer leur architecture, leur cycle de cognition et leur fonctionnement, à la fin nous allons citer les différents domaines d'application de la radio cognitive.

Le chapitre II présente les systèmes multi-agents. Nous allons voir la notion d'agent et d'environnement, nous allons montrer les différentes manières d'interactions et de communications entre agents, enfin nous allons mentionner quelques plateformes de systèmes multi-agents.

Le chapitre III est destiné à la présentation de notre contribution dans le cadre de ce PFE à savoir la proposition d'une version parallèle de la méthode k-ppv. Nous présentons en détail notre application ainsi que les résultats obtenus.

CHAPITRE I :
Réseau de radio cognitive

I.1 Introduction

Le domaine de télécommunication a connu une évolution technologique très rapide, ce qui a obligé les chercheurs à trouver des techniques qui permettent une meilleure exploitation des équipements matériels et logiciels.

Parmi les technologies proposées, la radio cognitive, un concept qui était conçu après plusieurs technologies comme la radio logicielle. L'idée d'utiliser la radio cognitive était l'accès au spectre d'une manière opportuniste.

Dans ce chapitre, nous allons présenter les réseaux de radio cognitive avec leur principe, leur architecture, leur cycle de cognition, leur fonctionnement et leurs domaines d'applications.

I.2 Réseau de radio cognitive

I.2.1 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 [1].

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. Son travail consiste à décrire les radios intelligentes qui peuvent prendre des décisions de manière autonome.

I.2.2 Définition de la radio cognitive

Le terme radio cognitive est utilisé pour décrire un système ayant la capacité de détecter et de reconnaître son cadre d'utilisation, ceci afin de lui permettre d'ajuster ses paramètres de fonctionnement radio de façon dynamique et autonome et d'apprendre des résultats de ses actions et de son cadre environnemental d'exploitation [2].

La radio cognitive est un paradigme pour la communication sans fil. Pour une communication efficace, la radio cognitive permet de modifier ses paramètres de transmission ou de réception de manière automatique. Cette auto-configuration et auto-adaptation des paramètres sont basées sur une surveillance active de plusieurs facteurs dans l'environnement

interne ou externe à la radio tels que les besoins de l'utilisateur, l'environnement radio et l'état du réseau.

I.2.3 Principe

La radio cognitive permet l'utilisation de spectre temporellement inutilisé, qui est mentionné comme trou de spectre ou espace blanc. Si cette bande est encore utilisée par un utilisateur autorisé, la RC offre des mouvements à un autre espace blanc, en changeant son niveau de puissance de transmission ou arrangement de modulation pour éviter l'interférence. De part cette vision, deux éléments sont omniprésents dans chaque communication radio cognitive [3], utilisateurs primaires et utilisateurs secondaires.

I.2.3.1 Utilisateurs primaires (PU)

Appelés aussi « utilisateurs licenciés » sont des utilisateurs qui disposent d'une licence qui leur permet d'opérer sur des bandes spectrales qui leurs sont réservées, ainsi ils ont le droit de communiquer en toute liberté à tout instant sur leurs bande de fréquence.

I.2.3.2 Utilisateurs secondaires (SU)

Ces utilisateurs n'ont pas de licence mais peuvent accéder à des bandes de fréquence non utilisées par les utilisateurs primaires à n'importe quel moment et les libère une fois le service terminé ou une fois qu'un utilisateur primaire en aura besoin. Les SU accèdent au spectre de façon opportuniste sans déranger les utilisateurs primaires [4].

I.2.4 Architecture de la radio cognitive

La structure de la radio cognitive serait cette homogénéité d'une multitude de lois de conception grâce à laquelle un ensemble distinct de composants produit une succession de fonctions, de produits et de services. Telle est l'explication fournie par Mitola [5].

La figure I.1 présente l'architecture de la radio cognitive (RC).

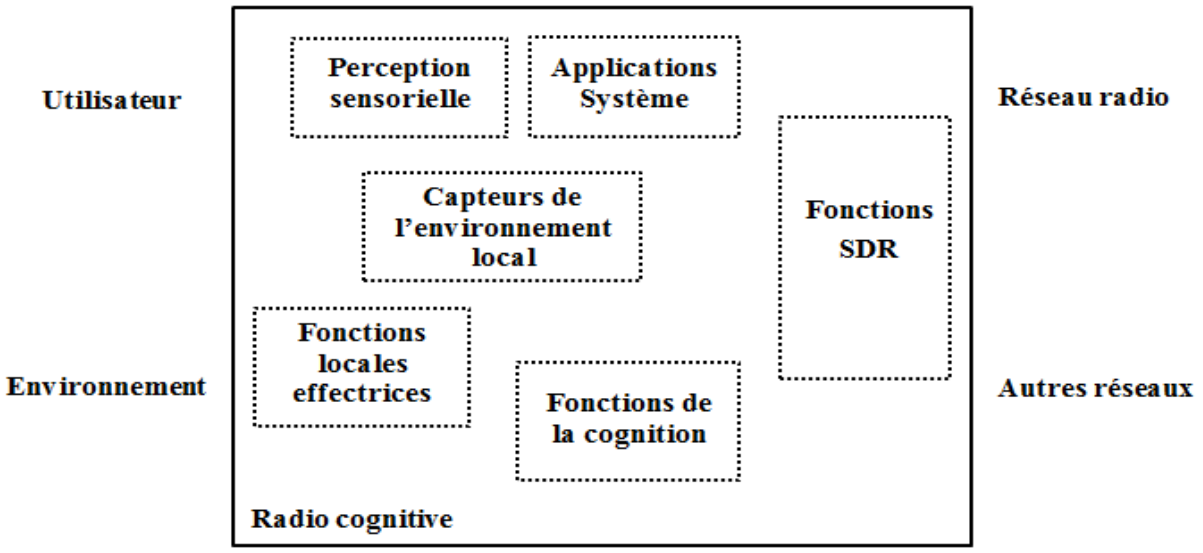


Figure I.1: Architecture de la radio cognitive [6].

Un nœud AACR comporte une suite minimaliste de 6 composantes fonctionnelles. Pour un élément fonctionnel, qui est une boîte noire à laquelle des rôles ont été concédés, la mise en œuvre des composants n'est pas déterminée. De cette façon, les détails des éléments de type logiciel ne sont pas précisés même s'ils seraient susceptibles de l'être [7].

Les six composantes fonctionnelles sont:

- La perception sensorielle de l'utilisateur intègre l'interface haptique (du toucher), acoustique, la vidéo et les fonctions de détection et de la perception.
- Les capteurs de l'environnement local (emplacement, la température, l'accéléromètre compas, etc.).
- Les applications systèmes (les services médias indépendants comme un jeu en réseau).
- Les fonctions SDR (qui comprennent la détection RF et les applications radio de la SDR).
- Les fonctions de la cognition (pour les systèmes de contrôle, de planification, de l'apprentissage).
- Les fonctions locales effectrices (synthèse de la parole, du texte, des graphiques et des affiches multimédias) [8].

I.2.5 Cycle de cognition

Le cycle de la cognition [9] est un schéma global décrivant l'ensemble des interactions entre les différents modules du système y compris l'environnement externe. L'analyse ou l'observation de son environnement, la planification de ses actions et la décision constituent le fondement de ses activités. La figure I.2 présente ce cycle de cognition.

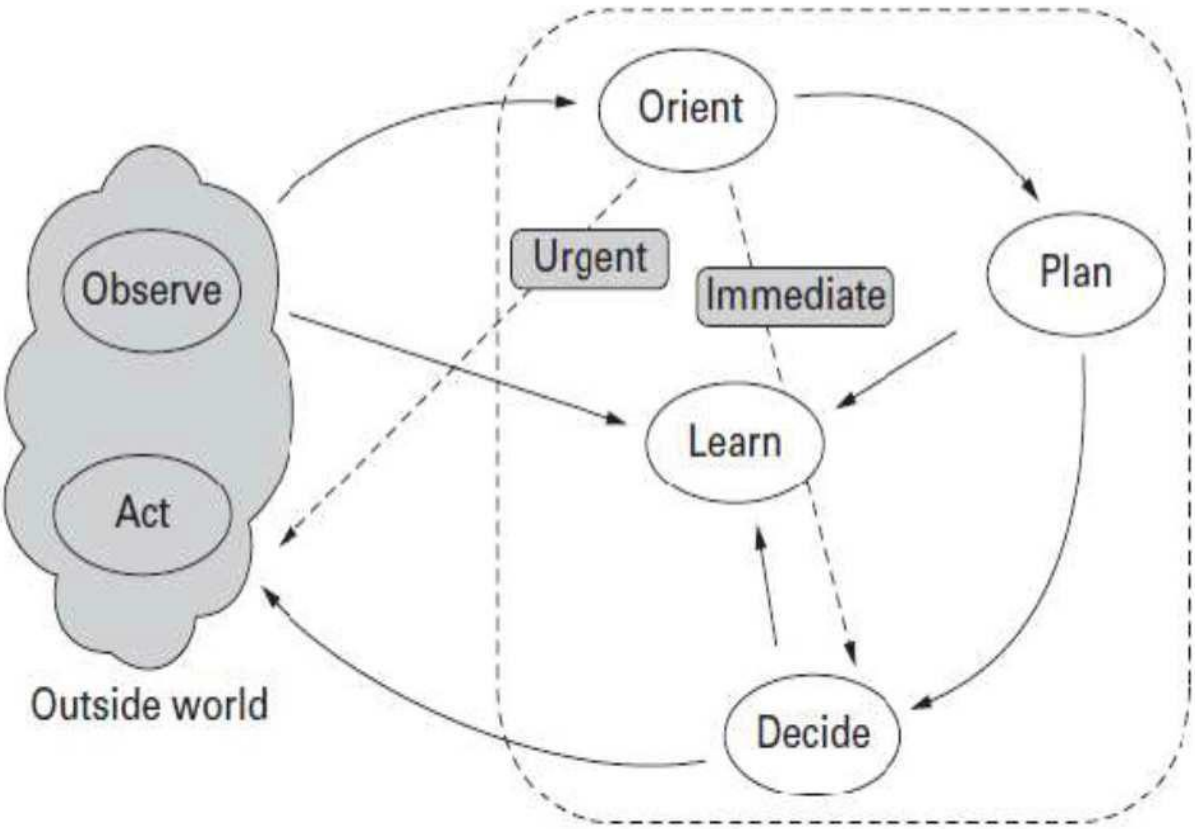


Figure I.2: Cycle de cognition de Mitola [10].

I.2.5.1 Phase d'observation (détecter et percevoir)

L'environnement de la radio cognitive est observé par l'analyse du flux de stimuli entrant. Dans cette phase, la RC associe l'emplacement, la température, le niveau de lumière des capteurs, et ainsi de suite pour en déduire le contexte de communication.

I.2.5.2 Phase d'orientation

C'est une étape importante dans l'établissement des priorités, la classification par affinité ou en fonction de la demande et du besoin ainsi que la planification des actions envisageables [11].

I.2.5.3 Phase de planification

Dans cette phase, il faut générer un plan pour traiter un message entrant dans le réseau. Le message entrant serait 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.

I.2.5.4 Phase de décision

C'est la dernière étape du cycle de cognition d'une radio cognitive. 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 [3].

1.2.5.5 Phase d'action

Permet de lancer les processus sélectionnés qui utilisent les effecteurs sélectionnés qui accèdent au monde extérieur ou aux états internes de la radio cognitive.

I.2.5.6 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 [3].

L’apprentissage peut se produire quand un nouveau modèle est créé en réponse à une action. Par exemple, les états internes antérieurs et courants peuvent être comparés avec les attentes pour en apprendre davantage sur l'efficacité d'un mode de communication [7].

La figure présentée ci-dessous résume le cycle de cognition en seulement trois phases :

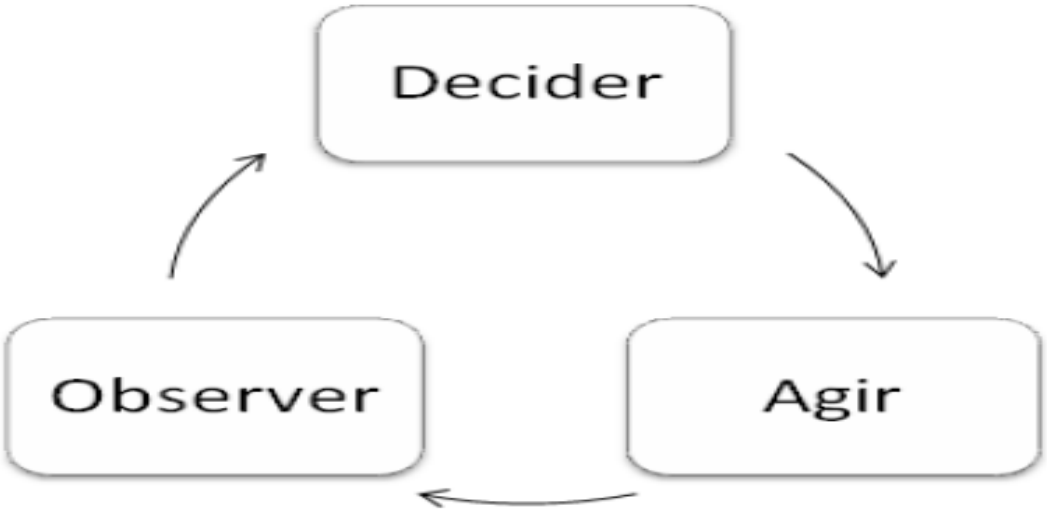


Figure I.3: Cycle de cognition simplifié [10].

I.2.6 Composante de la radio cognitive

La figure I.4 illustre les différentes composantes d'un émetteur/récepteur radio cognitive qui mettent en œuvre ces fonctionnalités :

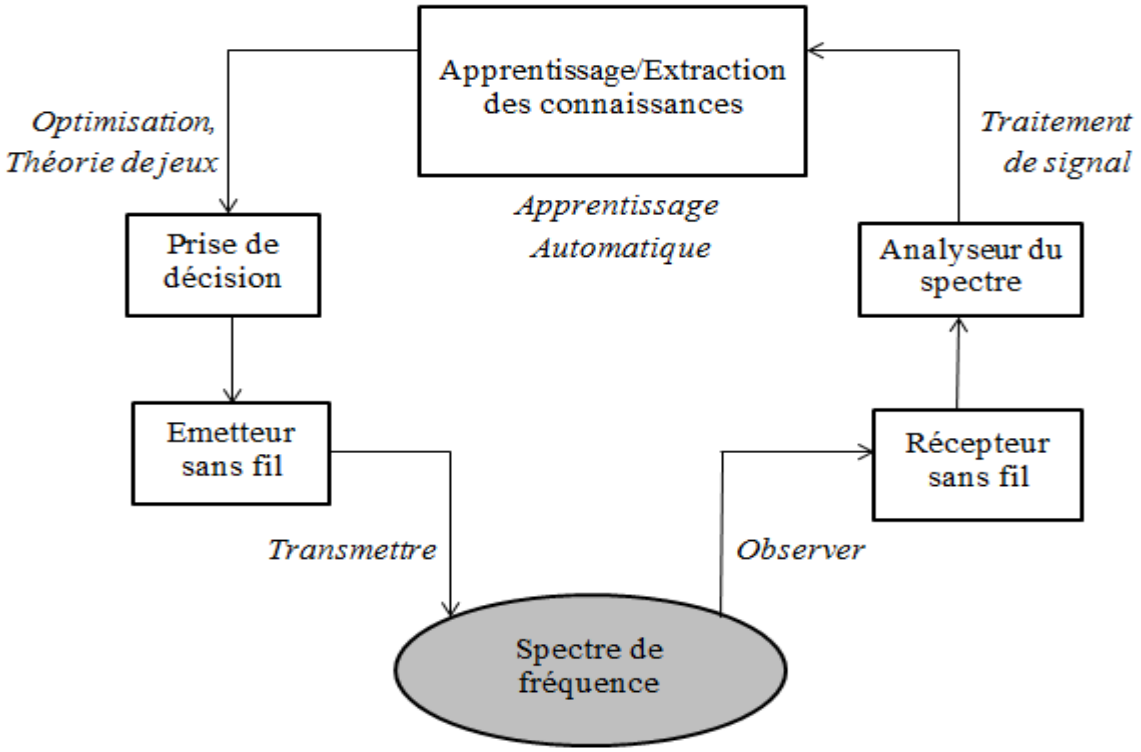


Figure I.4: Composantes de la radio cognitive [3].

I.2.6.1 Emetteur/Récepteur

Transmettant et recevant des données, l'émetteur / récepteur sans fils est l'un des composants cardinales 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.

I.2.6.2 Analyseur de spectre

L'analyseur de spectre doit s'assurer que la transmission d'un utilisateur primaire (PU) n'est pas perturbée si un utilisateur secondaire (SU) 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.

I.2.6.3 Extraction/apprentissage des connaissances

Utilisant une base de connaissances de l'environnement construite et entretenue par des algorithmes d'apprentissage et d'extraction de connaissances, une radio cognitive optimise et adapte les paramètres de transmission pour atteindre un objectif donné sous divers circonstances [12].

I.2.6.4 Prise de décision

Après que la connaissance de l'utilisation du spectre soit disponible, la décision sur l'accès au spectre doit être faite. Cette décision optimale dépend du milieu ambiant, du comportement (coopératif / compétitif) des seconds utilisateurs.

I.2.7 Fonctionnement de la radio cognitive

Les fonctions suivantes sont principales pour le bon fonctionnement de la radio cognitive :

I.2.7.1 Détection du spectre

C'est la fonctionnalité de base, elle consiste à détecter le spectre non utilisé et le partager sans interférence avec d'autres utilisateurs. L'un des objectifs de la détection du spectre, en particulier pour la détection des interférences, est d'obtenir le statut du spectre (libre /occupé), de sorte que les spectres peuvent être consulté par un SU en vertu de la contrainte d'interférence. Le défi réside dans le fait de mesurer l'interférence au niveau du récepteur primaire causée par les transmissions d'utilisateurs secondaires [13].

I.2.7.2 Gestion de spectre

Cette fonction permet de Capter les meilleures fréquences disponibles pour répondre aux besoins de communication des utilisateurs.

Les radios cognitives devraient décider de la meilleure bande de spectre pour répondre aux exigences de qualité de service sur toutes les bandes de fréquences disponibles, donc les fonctions de gestion du spectre sont nécessaires pour les radios cognitives. Ces fonctions de gestion peuvent être classées comme suit [3]:

- **Analyse du spectre**

Les résultats obtenus de la détection du spectre sont analysés pour estimer la qualité du spectre.

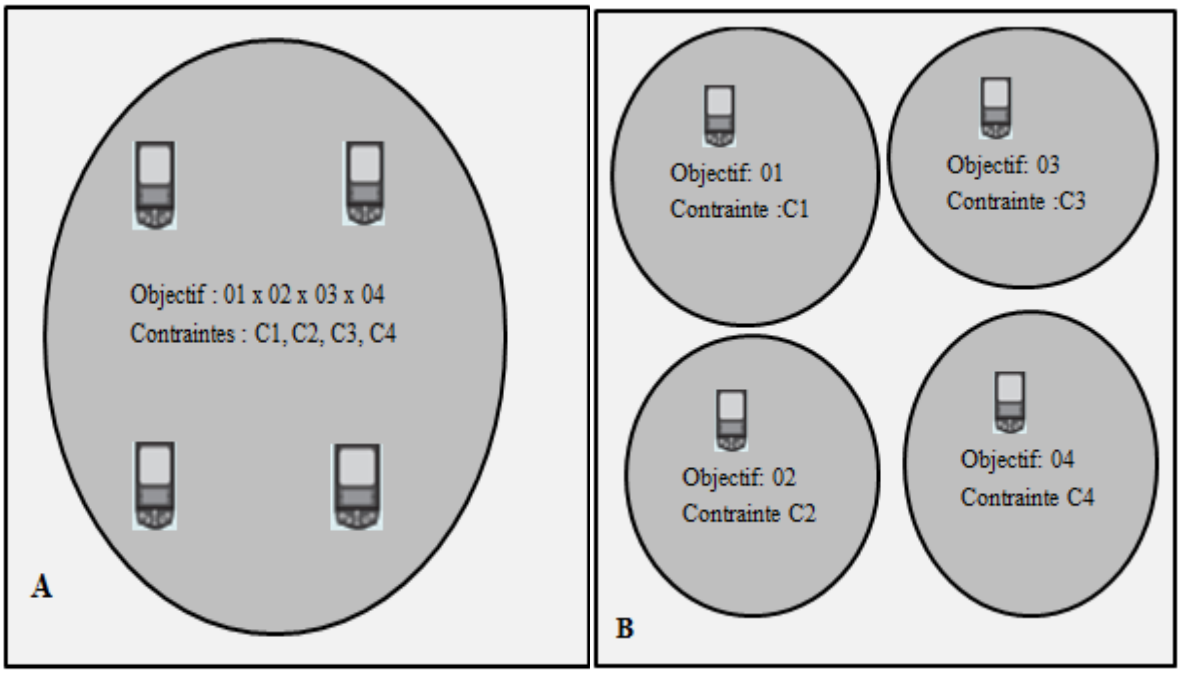
Des algorithmes d'apprentissage de l'intelligence artificielle sont des techniques qui peuvent être employées par les utilisateurs de la RC pour l'analyse du spectre.

- **Décision du spectre**

Après l'analyse du spectre vient la prise de décision pour l'accès au spectre. L'utilisateur secondaire doit décider quel spectre doit-il choisir en fonction d'u objectif, cet objectif peut être multiple [14].

Dans un système RC coopératif/non coopératif, il existe deux utilisateurs (PU, SU) qui peuvent être influés sur l'accès au spectre [15].

La figure I.5 représente les deux différents environnements. Dans un environnement non coopératif, chaque utilisateur a son propre objectif, par contre dans un environnement coopératif tous les utilisateurs peuvent collaborer les uns avec les autres pour atteindre un seul objectif.



Environnement coopératif. Environnement non coopératif.

Figure I.5: Accès coopératif et non coopératif au spectre [16].

I.2.7.3 Mobilité du spectre

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

Les réseaux RC utilisent le spectre de manière dynamique en permettant à de terminaux radio de fonctionner dans la meilleure bande de fréquence disponible, de maintenir les exigences de communications transparentes au cours de la transition à une meilleure fréquence [6].

I.2.8 Domaines d'application de radio cognitive

La radio cognitive peut être appliquée à différents scénarios de communications sans fil, nous présentons quelques-uns :

- **Les réseaux sans fil de prochaine génération**

La radio cognitive (RC) devrait être une technologie clé pour la prochaine génération de réseaux sans fil hétérogènes.

Pour l'utilisateur, un dispositif mobile avec des interfaces d'air multiples (WiFi, WiMAX, cellulaires) peut observer l'état des réseaux d'accès sans fil et prendre une décision sur la sélection de l'accès au réseau pour communiquer avec.

Pour le fournisseur, les ressources radio de plusieurs réseaux peuvent être optimisées pour l'ensemble des utilisateurs de mobiles et de leurs exigences de QoS.

- **La coexistence de différentes technologies sans fil**

La radio cognitive est une solution qui fournit la coexistence de différentes nouvelles technologies sans fil qui sont en cours d'élaboration pour la réutilisation des fréquences radio allouées à d'autres services sans fil (service TV).

- **Les services de cyber santé (eHealth services)**

Différents types de technologies sans fil sont adaptées dans les services de santé pour améliorer l'efficacité de la gestion des soins de santé et la gestion des patients.

Cependant, la plupart des dispositifs médicaux de soin utilisés sont sans fil et sont sensibles aux EMI (interférence électromagnétique). En outre, différents dispositifs biomédicaux (équipement et appareils chirurgicaux, de diagnostic et de suivi) utilisent la transmission RF.

Dans ce cas, la radio cognitive peut être appliquée pour l'utilisation du spectre de ces dispositifs mais doit éviter toutes les interférences avec l'autre.

- **Les réseaux d'urgence**

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

- **Les 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 [3].

I.3 Conclusion

Nous avons présenté dans ce chapitre une technologie principale pour les futures communications sans fil qui est la radio cognitive (RC). La RC offre une solution équilibrée au problème de l'encombrement du spectre. Son objectif est d'améliorer les performances des réseaux.

CHAPITRE II :
Les systèmes multi-agents

II.1 Introduction

Il est devenu nécessaire de développer des applications informatiques coopératives capables de prendre en compte les caractéristiques de tels systèmes. Le domaine de "systèmes multi-agents" qui est actuellement un champ de recherche très actif, s'intéresse aux comportements collectifs produisent par les interactions de plusieurs agents. Les SMA proposent des modèles et des techniques bien adaptés pour modéliser et développer de telles applications.

Dans ce chapitre nous présenterons les notions de systèmes multi-agents (SMA) avec leurs caractéristiques, leurs utilisations, leurs domaines d'applications. Nous allons ensuite déterminer l'environnement, l'interaction et la communication entre agents. A la fin, nous donnerons quelques plateformes des systèmes multi-agents.

II.2 Agent

II.2.1 Définition d'un agent

Il existe plusieurs domaines qui utilisent la notion d'agent, parmi ces domaines, nous trouvons : la biologie, la psychologie cognitive, la sociologie et l'informatique. Ce qui fait que peu de compromis existent lorsqu'il s'agit de définir le terme « agent », en général, la définition diffère selon le type d'application et le comportement de l'agent.

Parmi les définitions les plus connues est celle de [17] :

> *« Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.»*

II.2.2 Caractéristiques des agents

D'après Wooldrige et Jennings [18], nous pouvons citer les caractéristiques suivantes pour la notion d'agent :

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

- **Proactif :** un agent capable d'avoir un comportement opportuniste, dirigé par ses buts ou sa fonction d'utilité, et prendre des initiatives au bon moment.

- **Situé :** l'agent est capable de recevoir des entrées sensorielles provenant de son environnement et ainsi effectuer des actions qui peuvent faire des changements dans cet environnement.

- **Réactive :** un agent perçoit son environnement et répond aux changements qui s'y produisent en un temps raisonnable.

- **Social :** un agent peut interagir avec d'autres agents (logiciels ou humains).

II.2.3 Typologies des agents

Il existe deux différents types d'agent dans les SMA, nous trouvons les agents cognitifs (agent intelligent) et les agents réactifs (agent sans intelligence) comme il existe un autre type d'agents appelé agent hybride apportant une réponse aux imperfections des deux types précédents.

II.2.3.1 Agents réactifs

Les agents réactifs ont un comportement du type « stimulus – réponse ». L'agent réactif ne possède pas une représentation complète de son environnement et n'est pas capable de tenir compte de ses actions passées. De ce fait, un agent réactif est extrêmement simple. Il dispose d'un processus de raisonnement procédural, d'un protocole et d'un langage de communication réduit [19].

Un problème complexe peut être résolu par ce type d'agents. Les systèmes multi-agents constitués uniquement d'agents réactifs possèdent un grand nombre d'agents.

II.2.3.2 Agents cognitifs

Les agents cognitifs disposent d'une base de connaissances comprenant les diverses informations liées à leurs domaines d'expertise et à la gestion des interactions avec les autres agents et leur environnement. Les agents sont généralement "intentionnels" c'est à dire qu'ils possèdent des buts et des plans explicites leur permettant d'accomplir leurs buts [20].

Lorsqu'un problème est très complexe, l'agent cognitif peut le résoudre. Les systèmes multi-agents constitués uniquement d'agents cognitifs sont constitués d'un nombre d'agents faible. Le tableau II.1 ci-dessous montre la différence entre les agents cognitifs et les agents réactifs.

Caractéristiques	Agent cognitif	Agent réactif
Capacité de raisonnement	Elevé	Faible
Représentation explicite de l'environnement	Oui	Non
Nombre d'agents	Grand	Petit
Structure	Complexe	Stimulus/Action
Temps de réponse	Lent	Rapide

Tableau II.1: Comparatif entre agent cognitif et réactif [3].

II.2.3.3 Agents hybrides

Les agents purement réactifs ont un comportement assez simpliste alors que les agents cognitifs utilisent des mécanismes de raisonnement qui sont un peu difficile à manipuler et qui ne sont pas suffisamment réactifs. Afin d'apporter une réponse à ces imperfections, des architectures hybrides en couches ont été proposées [21].

II.2.4 Domaines d'application des agents

Les agents sont utilisés dans les domaines suivants :

- **L'énergie :** par exemple l'achat de la puissance dans une manière intelligente, la gestion des réseaux.
- **L'industrie :** par exemple l'automatisation des processus et de la production, la maison intelligente.

- **La communication :** par exemple la gestion de réseaux, le commerce électronique, la maison intelligente, les services du réseau personnel.

- **La santé :** par exemple la supervision des malades, les systèmes du support.

II.3 Système Multi-Agents

II.3.1 Définition

Un Système Multi-agents (ou SMA) est un système distribué composé d'un ensemble d'agents qui interagissent entre eux dans un environnement commun. Les agents dans les SMA peuvent être des humains, des robots, des machines où chacun possédant une ou plusieurs compétences élémentaires. Le but est de faire travailler ensemble les agents pour résoudre un problème complexe ou effectuer une tâche spécifique.

Ferber [20] a défini un SMA de la manière suivante :

« *Un système Multi-agents est un système composé des éléments suivants :*

- *Un environnement (E) est un espace disposant généralement d'une métrique.*

- *Un ensemble d'objets (O) situés dans l'espace, ils sont passifs, ils peuvent être perçus, détruits, créés et modifiés par les agents.*

- *Un ensemble d'agents (A) qui sont les entités actives du système.*

- *Un ensemble de relations (R) qui unissent les objets entre eux.*

- *Un ensemble d'opérations (Op) permettant aux agents de percevoir, de détruire, de créer, de transformer et de manipuler les objets.*

- *Un ensemble d'opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification (les lois de l'univers).»*

La figure II.1 représente l'interaction d'un agent avec son environnement et avec les autres agents.

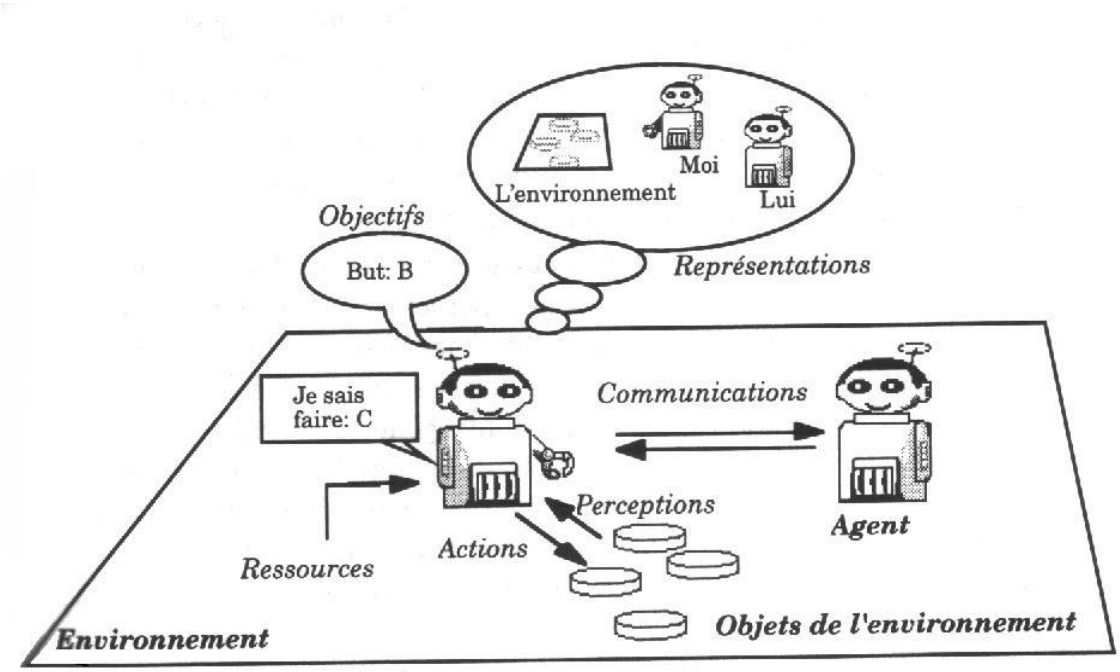


Figure II.1: Un système Multi-Agents (SMA) Feber [20].

II.3.2 Caractéristiques des SMA

Les SMA sont caractérisés par plusieurs éléments qui les distinguent des autres systèmes informatiques. Nous en fournissons une liste issue de la littérature, proposée dans [17]. En effet, un SMA possède les caractéristiques suivantes :

- **Distribution :** le système est modulaire, l'élément de base étant l'agent.
- **Autonomie :** un agent est en activité permanente et prend ses propres décisions fonction de ses objectifs et de ses connaissances.
- **Décentralisation :** les agents sont indépendants, il n'y a pas de décisions centrales.
- **Echange de connaissances :** les agents sont capables de communiquer entre eux, selon des langages plus ou moins élaborés.
- **Interaction :** les agents ont une influence locale sur le comportement des autres agents, généralement il n'a pas d'ordre, seulement des requêtes.
- **Organisation :** les interactions créent des relations entre les agents et le réseau de ces relations forme une organisation qui peut évoluer au cours du temps.

- **Intelligibilité :** les SMA proposent une manière naturelle de modéliser d'autres systèmes ou de mettre en œuvre des applications, ce qui les rend simples à apprendre pour un utilisateur extérieur.

II.3.3 Utilisation d'un SMA

L'utilisation d'un SMA est faite lorsque la complexité d'un problème est très grande par rapport à sa résolution avec un seul système. Cette complexité est causée par quelques restrictions qui peuvent être logicielles, matérielles ou par l'hétérogénéité de plusieurs composantes entretenant de multiples relations entre elles, travaillant sur des échelles de temps différentes. Pour assurer un contrôle autonome dans un système distribué et dont les caractéristiques sont très dynamiques, les SMA représentent un excellent outil.

II.3.4 Application des SMA

Les SMA peuvent être vus comme la rencontre de divers domaines informatique tels que l'intelligence artificielle (IA), l'intelligence artificielle distribuée(IAD), les systèmes distribués, le génie logiciel . Comme on peut les trouver dans la simulation de phénomènes complexes (sociologie, la physique des particules, la chimie et la robotique …), Télécoms, Jeux vidéo, E-commerce, E-learning. Ils s'inspirent également d'études issues d'autres disciplines connexes notamment la médecine, la psychologie sociale, les sciences cognitives et bien d'autres.

II.4 Environnement

On appelle un environnement d'un SMA, le milieu qui permet aux agents d'évoluer. Il fournit un support commun aux actions des agents, il permet ainsi l'interaction dans le système, et il constitue une source d'information à laquelle les agents peuvent accéder au travers de leur perception.

II.4.1 Propriétés de l'environnement

Russel et Norvig [22] ont proposé différentes propriétés de l'environnement d'un SMA, ces propriétés sont présentées comme le suivant :

- **Accessible ou inaccessible :** un environnement est dit accessible si l'agent peut obtenir une information complète et exacte. Sinon, l'environnement est inaccessible.

- **Déterministe ou indéterministe :** un environnement est dit déterministe si le prochain état de l'environnement est déterminé d'une manière unique par l'état courant et par l'action de l'agent. Si le résultat est incertain, alors on est dans le cas d'un environnement non déterministe.

- **Episodique ou non épisodique :** un environnement est dit épisodique si les prochaines évolutions ne dépendent pas des actions précédentes (déjà réalisées). Les environnements épisodiques sont plus simples du point de vue du développeur d'agents parce que l'agent peut décider quelle action appliquer en se basant seulement sur l'épisode courant.

- **Statique ou dynamique :** un environnement statique est un environnement qui ne change pas d'état que lors de l'application d'actions par l'agent. Par contre un environnement est dynamique si son état peut se modifier sans l'action de l'agent dans un intervalle de temps entre deux perceptions de l'agent.

- **Discret ou continu :** un environnement est dit discret s'il comporte un nombre fixe et limité d'actions et de perceptions possibles. Si tout passage d'un état de l'environnement à un autre nécessite le passage par une séquence d'états intermédiaires, alors on a un environnement continu.

II.5 Interactions entre agents

La relation dynamique de deux ou plusieurs agents à travers un ensemble d'actions réciproques est nommée interaction. L'un des propriétés principales de l'agent dans un SMA est d'interagir avec les autres agents.

Ces interactions sont généralement définies comme toute forme d'actions exécutées au sein du système d'agents. Cela a pour effet de changer le comportement d'un autre agent. Il existe trois types d'interactions :

II.5.1 Coopération entre agents

Des agents en coopération cherchent la réussite du groupe. Ils travaillent ensemble pour résoudre un problème commun.

II.5.2 Coordination entre agents

Un groupe d'agents est en coordination, lorsqu'il existe un agent centralisateur qui crée des plans d'action et assigne les tâches aux autres agents du groupe.

II.5.3 Négociation entre agents

Dans le cas des SMA, la négociation est une composante de base de l'interaction et un processus de communication d'un groupe d'agents permettant d'atteindre un accord mutuellement accepté.

Il existe deux types de négociation :

- **Négociation compétitive :** les agents d'intérêts différentst entent un choix de groupe sur des alternatives bien définies. Ils ont des buts indépendants et interagissent entre eux.

- **Négociation coopérative :** les agents ont un but commun aussi. Ils sont collaboratifs, ils coopèrent.

II.6 Communication entre agents

L'interaction d'un agent avec d'autres agents demande un langage commun de communication.

La communication est généralement basée sur trois éléments:

II.6.1 Protocole d'interaction entre agents

Le protocole d'interaction est un ensemble de règles qui sont partagées par les agents; un ensemble de messages convenus est exigé pour l'interaction entre agents, de règles pour des actions basées sur la réception de divers messages, et d'acceptations des voies de transmission [21]. Ces contraintes, règles et modèles peuvent être soustraits et formalisés comme AIP (Agent Interaction Protocol). L'AIP sont des modèles représentant les messages de communication et les contraintes correspondantes sur le contenu de tels messages.

II.6.2 Transport de messages

La communication des agents est essentielle aussi bien pour échanger les données entre les agents que les connaissances. Cette communication exige des moyens par exemple : les sockets de niveau le plus bas qui permettent aux différents agents codés en Java de communiquer entre eux. Comme on peut citer les deux méthodes distantes (RMI pour Remote Method Invocation) et la technologie CORBA (Common Request Broker Architecture).

II.6.3 Langages de communication entre agents

Pour échanger les informations et les connaissances, les agents utilisent les ACL (Agent Communication Language) qui a été créé pour assurer l'interopérabilité entre des agents autonomes et distribués.

Parmi les langages d'ACL les plus connues KQML et FIPA-ACL.

- **KQML** (KnowledgeQuery and Manipulation Language) est issu d'un projet de la DARPA [23]. Ila été conçu comme étant à la fois un format de message et un protocole de transfert de messages.

- **FIPA-ACL** a été créé par l'organisme international FIPA (Foundation for Physical Intelligent Agents) afin de créer un langage de communication agent standard. FIPA-ACL se base sur la théorie des actes du langage. La syntaxe similaire à celle de KQML**.**

II.7 Plateformes des SMA

Les plates-formes sont des outils permettant de faciliter la construction et l'exploitation des systèmes multi-agents.

Il existe plusieurs plateformes qui permettent la prise en charge des fonctions de base d'un simulateur multi-agents comme la communication, le cycle de vie des agents, la perception et l'environnement.

Parmi les plateformes les plus connues, nous trouvons Madkit [24], Mason [25], NetLogo [26] et Gama [27]. Cependant, ces plateformes n'intègrent pas de mécanismes de

parallélisme, il est nécessaire de développer une surcouche à la main, pour distribuer ou paralléliser une simulation.

Le deuxième type de plateformes est les plateformes multi-agents qui prennent en compte nativement le parallélisme comme RepastHPC [28], D-Mason [29], Pandora [30], FLAME [31], JADE (Java Agent DEvelopment) [32].

JADE est la plateforme utilisée pour la réalisation de notre application. Elle sera présentée dans le chapitre suivant.

II.8 Conclusion

Nous avons présenté dans ce chapitre les systèmes multi-agents qui présentent un domaine très ouvert pour la recherche. Nous avons décrit les éléments constituant un SMA tels que l'agent, l'environnement et les interactions.

Le chapitre suivant sera consacré à la réalisation de notre application qui est l'affectation des nœuds dans des clusters en utilisant l'approche K-plus proches voisins (K-PPV) sous la plateforme JADE.

CHAPITRE III :
Implémentation de l'application et évaluation des résultats

III.1 Introduction

Un traitement séquentiel consiste à exécuter un code étape par étape, où chaque opération se déclenche que lorsque l'opération précédente est terminée, ce qui donne un temps d'exécution élevé. Pour améliorer ce temps d'exécution, un mode de traitement parallèle est conseillé où plusieurs opérations peuvent être réalisées simultanément en utilisant les threads. Dans ce chapitre, nous proposons une version parallèle de l'algorithme des k-plus proches voisins (k-ppv). L'objectif est de faire une affectation rapide des nœuds dans un réseau de radio cognitive qui est déjà organisé sous forme de clusters. Une comparaison est faite avec la version de base (séquentielle) à la fin de ce chapitre afin de montrer l'intérêt de paralléliser cette méthode.

III.2 Présentation de la méthode utilisée (K-ppv)

L'algorithme des k-plus proches voisins (k-ppv) ou k-nearest neighbors en anglais (K-NN) est une méthode d'apprentissage supervisé dédiée à la classification, elle a été considérée parmi les plus simples algorithmes d'apprentissage artificiel.

Cette méthode est paramétrée par « k », le nombre de voisins à prendre en considération lors du classement d'un nouveau point. Le choix de ce paramètre est très important à la classification; si la valeur de **k** est petite (exemple, k=1) on risque de tomber dans des variations "aléatoires" ou "bruit" des clusters et si le contraire (la valeur de k est grande), le calcul va être lourd.

Dans ce PFE, nous avons choisi k=nombre de cluster+1(choix souvent utilisé dans la littérature).

L'opérateur de distance le plus souvent utilisé dans cette méthode est la distance Euclidienne; qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points :

$$dist(x,y) = \sqrt{\sum_{i=1}^{n} (\mathrm{xi} - yi)^2}$$

III.2.1 Principe de fonctionnement

Le principe de cet algorithme de classification est associé à une fonction de distance et à une fonction de choix de la classe majoritaire en fonction des classes des voisins les plus proches, qui constitue le modèle. La figure suivante montre ce principe qui a pour but de trouver la valeur de la classe où l'inconnu x va être affecté.

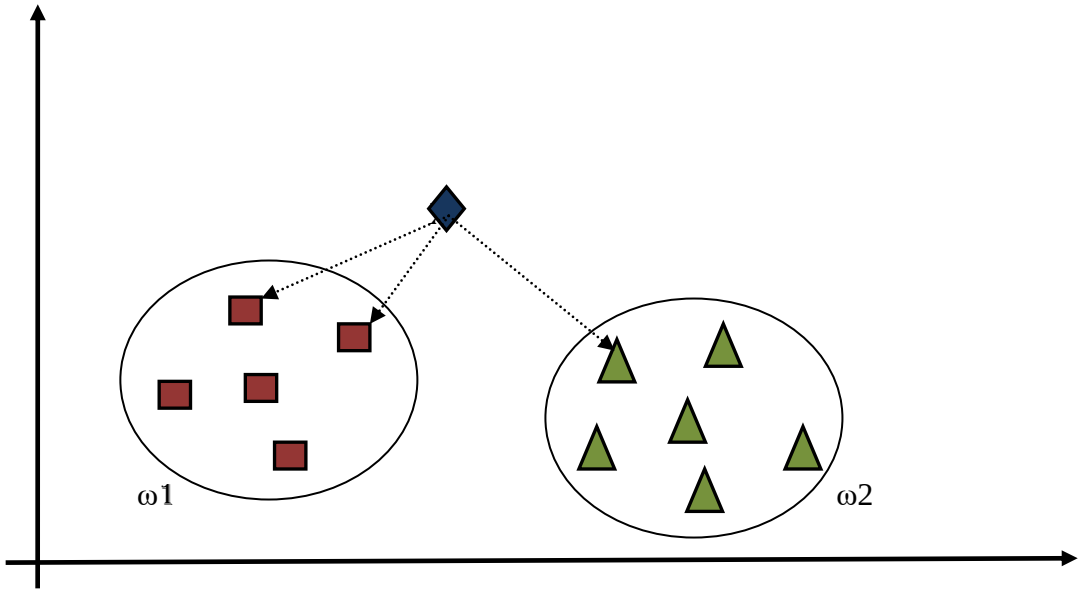


Figure III.1: Principe de fonctionnement de l'algorithme K-ppv.

Nous avons donc deux classes, et nous avons pris k=3.

Parmi les 3 voisins, nous avons 2 qui appartiennent à ω1 et 1 qui appartient à ω2 donc x sera affecté à ω1, la classe majoritaire.

III.3 Algorithme proposé

Le clustering est un mécanisme de gestion de topologie qui organise les nœuds en groupes logiques afin d'améliorer les performances du réseau. Chaque cluster comprend deux types de nœuds, à savoir le CH (cluster head) et les CM (cluster members). CH et ses CM communiquent régulièrement entre eux. C'est la communication intra-cluster.

Le clustering a été adopté dans les RRC à cause de ses avantages, il permet en particulier de :

- Réaliser le passage à l'échelle (scalabilité) car il réduit considérablement le coût de la communication entre les nœuds. En effet, les CMs échangent les informations uniquement avec leurs CHs respectifs.

- Garantir la stabilité du réseau puisque tout changement sur le réseau, provoque des mises à jour locales entre les nœuds membres et leurs CH respectifs.
- Aussi, avec le clustering, les CHs et les CMs peuvent coopérer plus efficacement afin de détecter les activités des utilisateurs primaires.

Donc, dans ce qui suit, nous allons considérer un réseau de radio cognitive (RRC) organisé sous la forme de M clusters.

Notre objectif est de faire une affectation rapide des nouveaux nœuds qui apparaissent dans ce type de réseau.

Le code suivant représente le comportement de base (séquentiel) de l'algorithme k-ppv.

Variables i, N, K, M : entier
Début
N ← nombre de nœuds dans le réseau
M ← nombre de clusters
K ← M + 1
Pour i allant de 1 à N faire
Calculer la distance avec le nœud Node
Trier selon la distance
Fin pour
Affecter le nœud dans son cluster (k plus proches voisins + classe majoritaire)
N ←N + 1 *// à la fin de l'affectation, le nœud incrémentera le nombre N et donc le prochain nœud travaillera sur N + 1 nœuds*
Fin

La version séquentielle n'est pas adaptée dans le cadre d'un réseau qui est organisé sous forme de clusters. En effet, un nœud doit attendre la fin du traitement relatif au précédent nœud (celui qui est apparu avant lui dans le réseau), afin de réaliser son traitement.

Ce comportement basique de l'algorithme augmente considérablement le temps d'exécution surtout dans le cas d'apparition d'un nombre très important de nœuds dans le réseau.

Afin d'améliorer le temps d'exécution de cet algorithme, nous proposons une version parallèle de ce dernier dans le cadre d'un réseau de radio cognitive.

L'objectif est de réaliser un calcul parallèle des distances relatives aux nœuds déjà existants dans le réseau. Un mécanisme de synchronisation entre threads est par la suite utilisé afin d'affecter le nœud dans son cluster.

Chaque nouveau nœud qui vient d'apparaître sur le réseau sera pris en charge par un thread différent.

Le code que nous proposons pour la version parallèle est comme suit :

Variables globales Y, N : entier // variables partagées entre les threads
Structures P : Point // afin de représenter un nœud RC
Di : Distance // pour représenter les distances
liste_Dist : Liste_des_distances // une liste chainée de type Distance
liste_Points : Liste_des_points // une liste chainée de type Point
Variable var : entier
Début
1.N ←Calculer_taille (liste_Points)
2.Y ← N
3.liste_Dist← P. Calculer_distances (P, liste_Points)
4.liste_Dist←P.Trier_Distances (liste_Dist)
5.Début bloc synchronisé
6.var ←Calculer_taille (liste_Points) -1
7.Pour i allant de Y+1 à N faire
8.Di ←P.Calculer_distance_nouveau_point (P, liste_Points(var))
9.liste_Dist← P. Inserer_nouvelle_distance (Di, liste_Dist)
10.var ← var-1
11. Fin pour
12.P.setGroupe (P.Classe_majoritaire (liste_Dist))
13. Ajouter (P, liste_Points)
14.Fin bloc synchronisé
Fin

Les lignes 1, 2, 3 et 4 sont exécutées en parallèles avec tous les threads. Il s'agit ici de paralléliser le calcul de la distance qui sépare les nouveaux nœuds qui viennent d'apparaître dans le réseau et tous les nœuds déjà existants dans le même réseau.

Les lignes de 5 à 14 représentent le bloc synchronisé. C'est une partie qui doit être exécutée en exclusion mutuelle (un seul thread à la fois). Le premier thread qui exécute ce

bloc synchronisé affecte le premier nœud dans son cluster, par la suite, le deuxième thread affecte le deuxième nœud, le troisième thread affecte le troisième nœud et ainsi de suite.

III.4 Outils utilisés

Afin de valider notre contribution dans le cadre de ce PFE, nous avons opté par une approche agent. Donc, chaque nouveau nœud qui vient d'apparaître dans le réseau RC sera modélisé par un agent.

La construction d'un SMA nécessite une plateforme multi-agents qui permet la construction et la mise en service d'agents au sein d'un environnement particulier.

Pour la réalisation de notre application, nous avons opté pour la plateforme JADE [33] en raison de sa simplicité d'utilisation.

III.4.1 JADE

JADE (Java Agent Development) est une plate-forme multi-agents développée en Java par le groupe de recherche : GruppoTelecom, Italie afin de construire des systèmes multi-agents et de réaliser des éventuelles applications.

JADE respecte la norme FIPA [34], elle possède trois modules principaux qui sont activés à chaque démarrage de la plate-forme :

- **Système de Gestion d'Agents (AMS) :** est l'agent central qui supervise les accès à la plate-forme et les autres agents.
- **Canal de Communication entre Agents (ACC) :** est l'agent qui fournit la route pour les interactions entre les agents dans et hors de la plate-forme.
- **Facilitateur d'Annuaire (DF) :** est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents.

Ainsi que d'autres modules :

- **Un runtime Environment :** l'environnement où les agents peuvent exécuter.
- **Une librairie de classes :** les classes que les développeurs utilisent pour écrire leurs agents.
- **Une suite d'outils graphiques :** un ensemble d'outils qui facilitent le débogage, la gestion et la supervision de la plate-forme des agents.

III.4.2 JFreeChart

JFreeChart [35] est une API open source pour le langage de programmation java. Elle permet de créer des graphes et des diagrammes pour afficher des données sous plusieurs formats tels que des barres, des lignes, ou des nuages de points.

III.5 Présentation de l'application

Notre application a été réalisée par une machine dotée d'un processeur Intel® Core™ i3-2348M possédant 4 CORES. C'est pour cette raison que nous sommes limités à l'affectation de seulement 4 nœuds en parallèle. Par contre, nous pouvons traiter un nombre très important de nœuds, ce nombre peut atteindre les 400000 nœuds déjà existant dans le réseau et organisé sous forme de clusters. Nous avons traité deux scénarios. Dans le premier cas, nous avons fixé le nombre des nœuds par cluster et nous avons varié le nombre de clusters. Dans le deuxième cas, nous avons fixé le nombre de clusters et nous avons varié le nombre de nœuds par cluster.

Afin d'avoir plus de crédibilité dans la comparaison des résultats, nous avons pris la moyenne de dix simulations.

La figure III.2 représente l'interface d'accueil de notre application.

Figure III.2: Interface d'accueil.

Dans notre interface d'accueil, nous avons deux boutons de type JButton «**Démarrer**» et «**Quitter**»

- «**Démarrer**» pour lancer l'application.
- «**Quitter**» pour sortir de l'application.

En cliquant sur Démarrer, l'interface de simulation s'affiche, nous distinguons deux onglets :

➢ Onglet du 1er cas : Nombre de nœuds fixe.

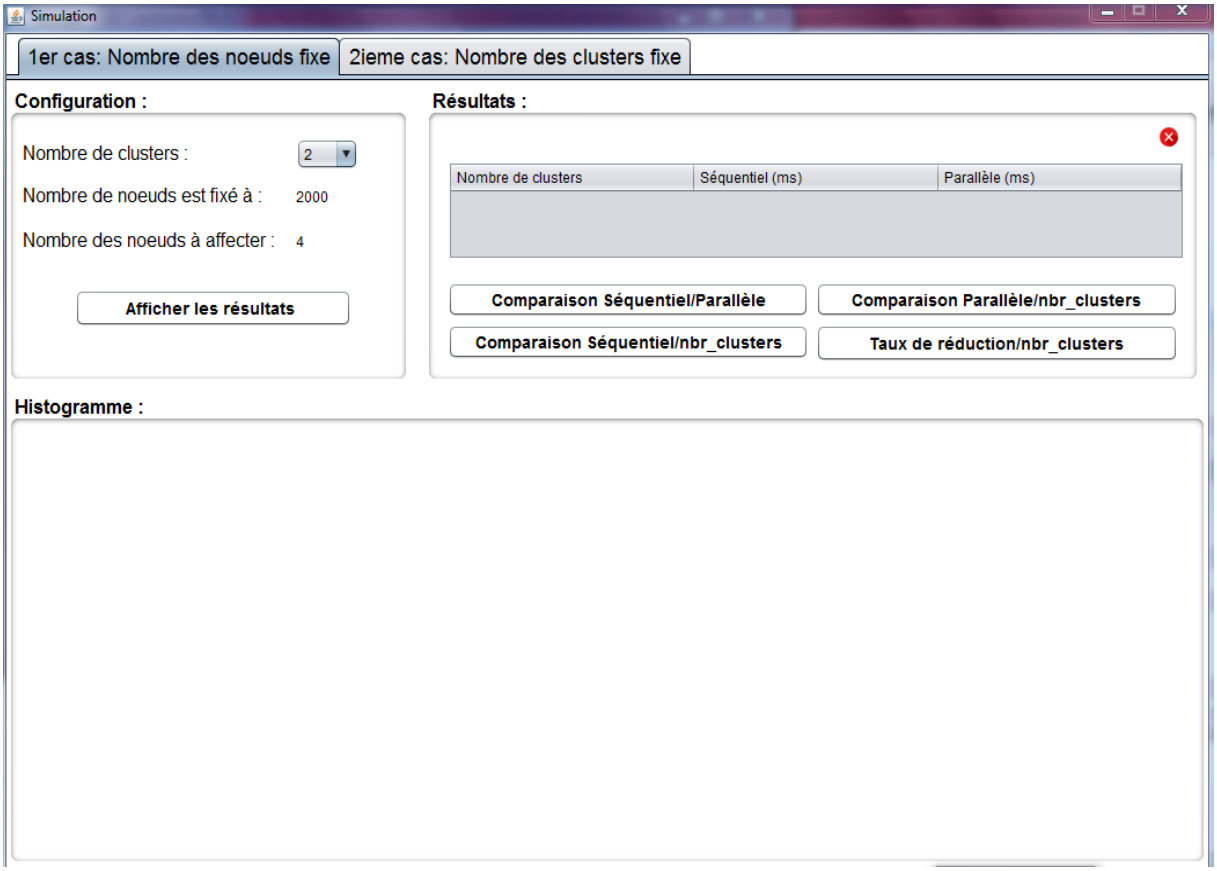


Figure III.3: Onglet 'Nombre des nœuds fixe'.

Cet onglet est composé d'un JComboBox pour le choix du nombre de clusters (2, 22, 42, 62, 82, 102, 202), un JLabel pour le choix du nombre de nœuds à affecter qui est fixé à 4 nœuds et un JLabel pour le choix du nombre de nœuds qui est fixé à 2000 dans ce cas.

La validation se fait en cliquant sur le bouton « Afficher les résultats ». La vue suivante est apparue :

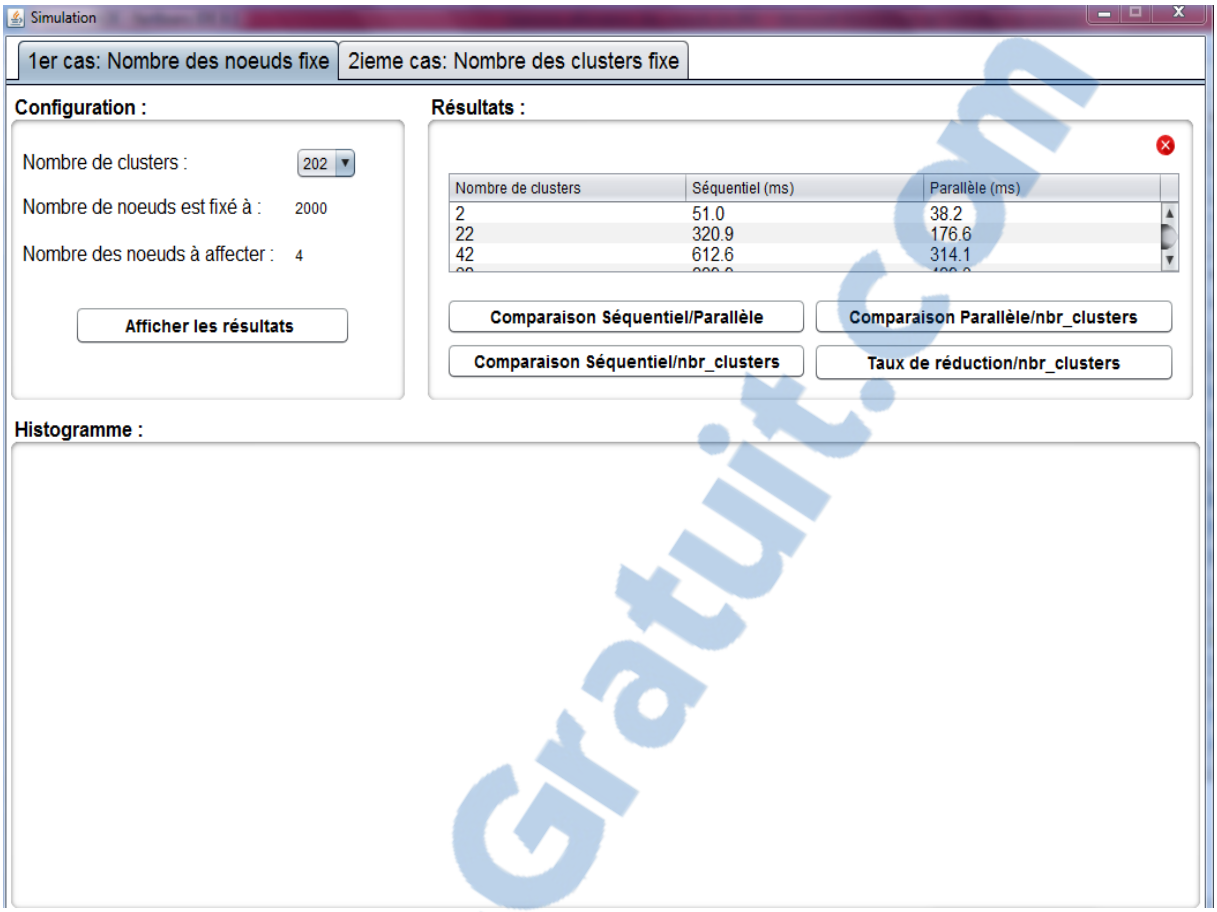


Figure III.4: Résultat de la simulation du premier cas: nombre de nœuds fixe.

Cette vue montre les résultats de la simulation concernant le premier cas (nombre de nœuds fixe par cluster), les valeurs de la moyenne de dix exécutions sont affichées dans le tableau.

➢ Onglet du 2ième cas : Nombre de clusters fixe.

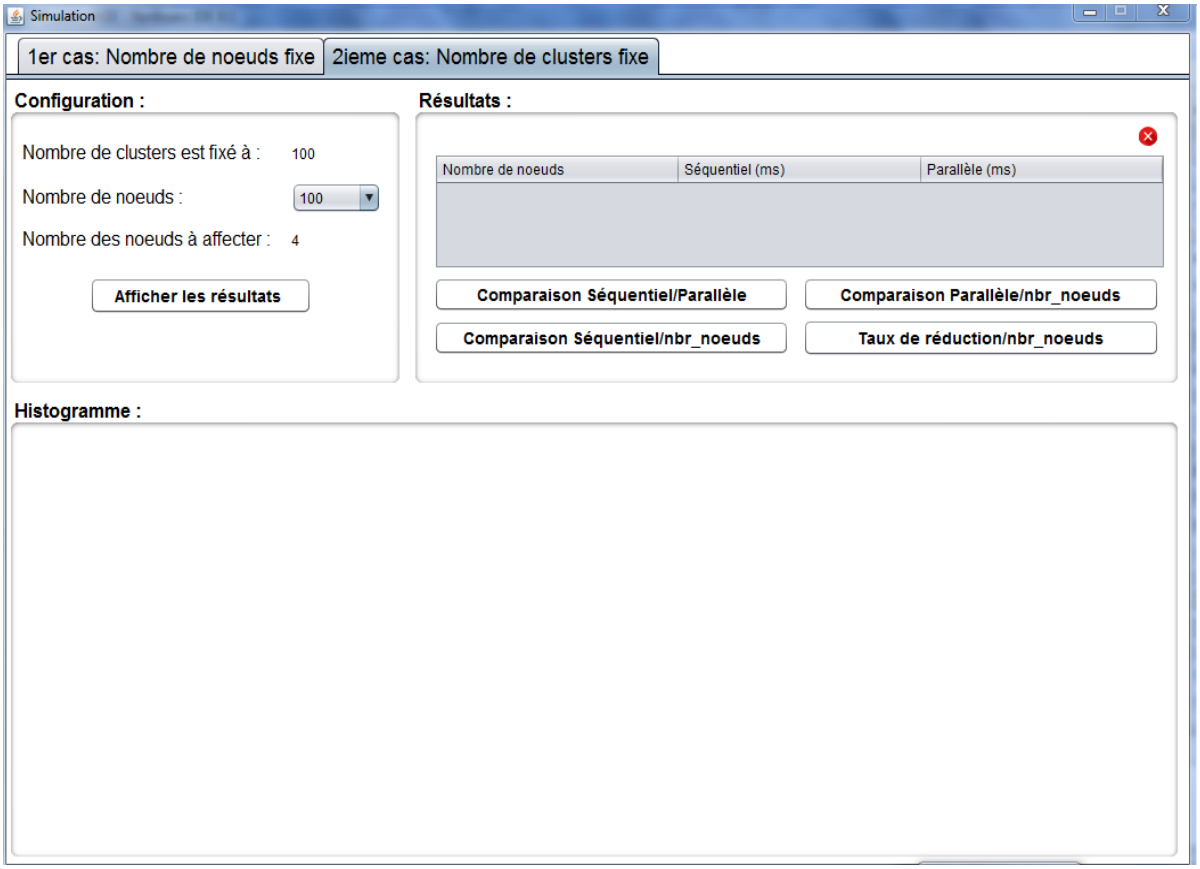


Figure III.5: Onglet 'Nombre de clusters fixe'.

Cet onglet est composé d'un JComboBox pour le choix du nombre de nœuds (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000), un JLabel pour le choix du nombre de nœuds à affecter qui est fixé à 4 nœuds et un JLabel pour le choix du nombre de clusters qui est fixé à 100.

La validation se fait en cliquant sur le bouton « Afficher les résultats ». La vue suivante est apparue :

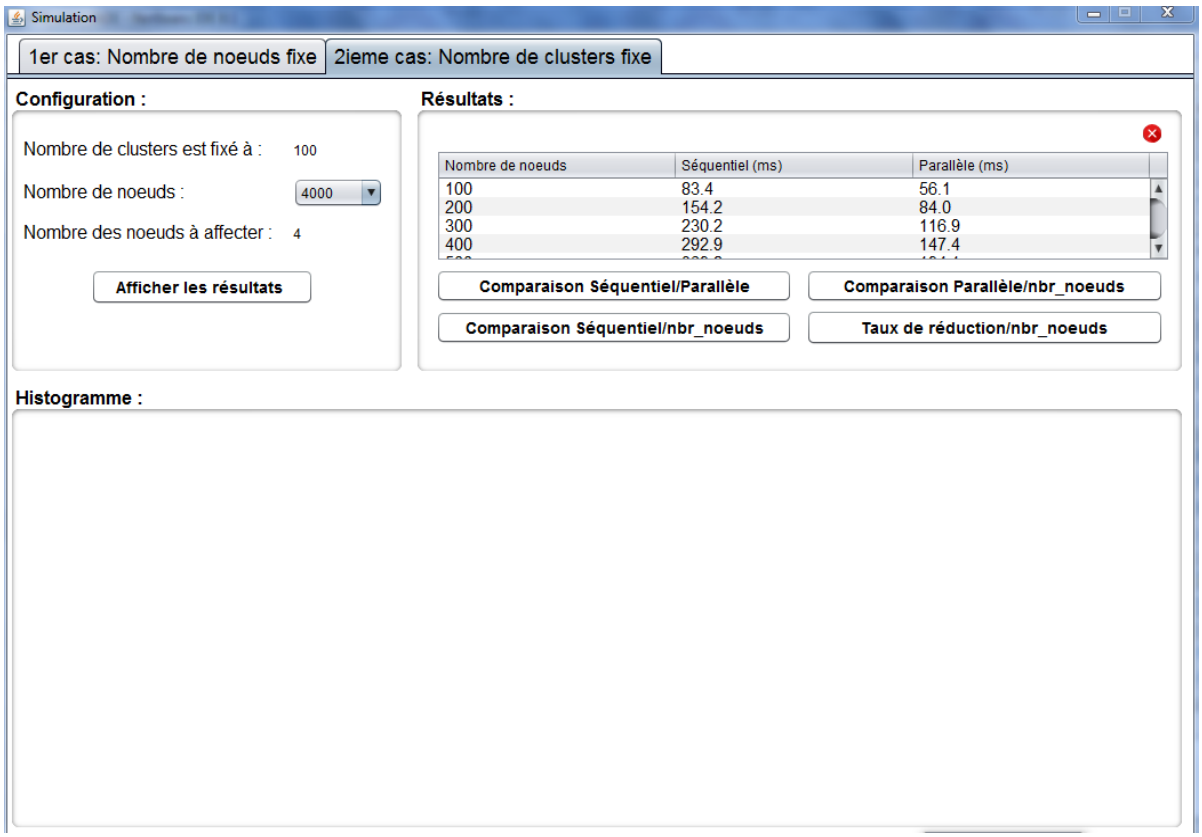


Figure III.6: Résultat de la simulation du deuxième cas: nombre de clusters fixe.

La figure précédente montre les résultats obtenus de la simulation concernant le deuxième cas (nombre de clusters fixe). Les valeurs de la moyenne de dix exécutions sont affichées dans le tableau.

III.6 Résultats obtenus

III.6.1 Nombre de nœuds fixe par cluster

Nous avons fixé le nombre de nœuds à 2000 nœuds par cluster, et nous avons varié le nombre de clusters de 2 à 202 avec un pas égal à 20.

La figure ci-dessous montre le temps d'exécution de la version séquentielle par rapport au nombre de clusters.

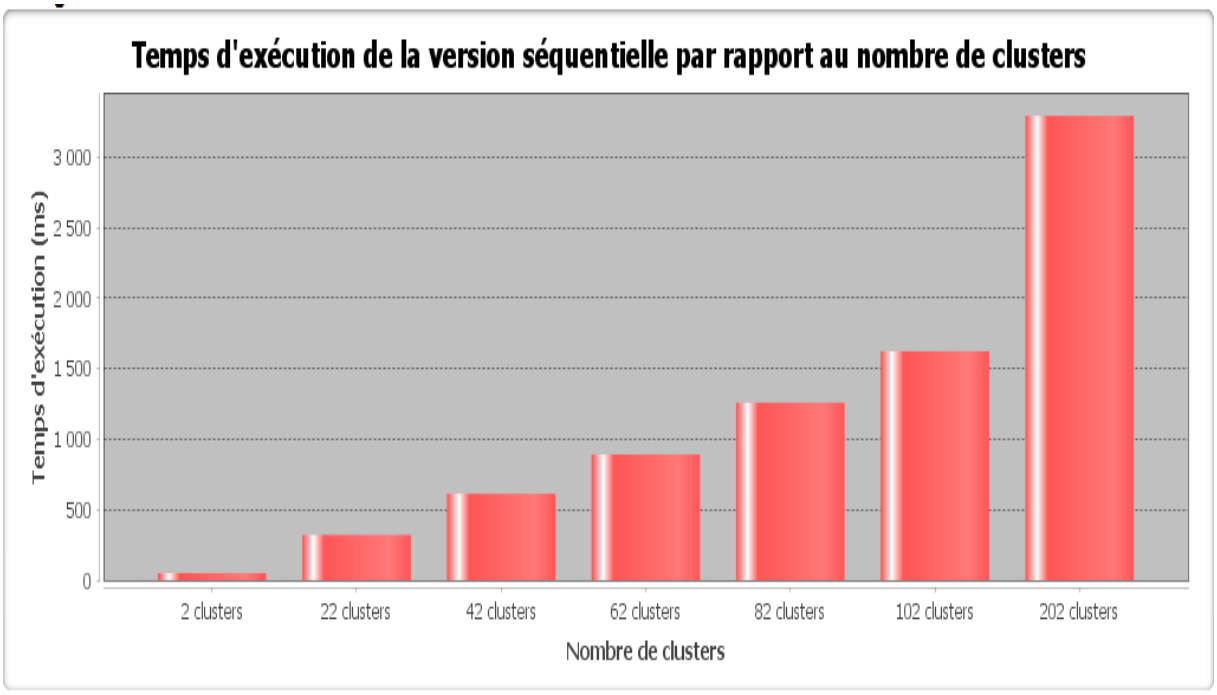


Figure III.7: Temps d'exécution de la version séquentielle par rapport au nombre de clusters.

La figure III.7 montre que plus le nombre de clusters est important plus le temps d'exécution est grand, avec 2 clusters le temps n'a pas dépassé 100 ms mais avec 202 clusters, il a dépassé les 3000 ms.

La figure III.8 montre les résultats obtenus de la version parallèle.

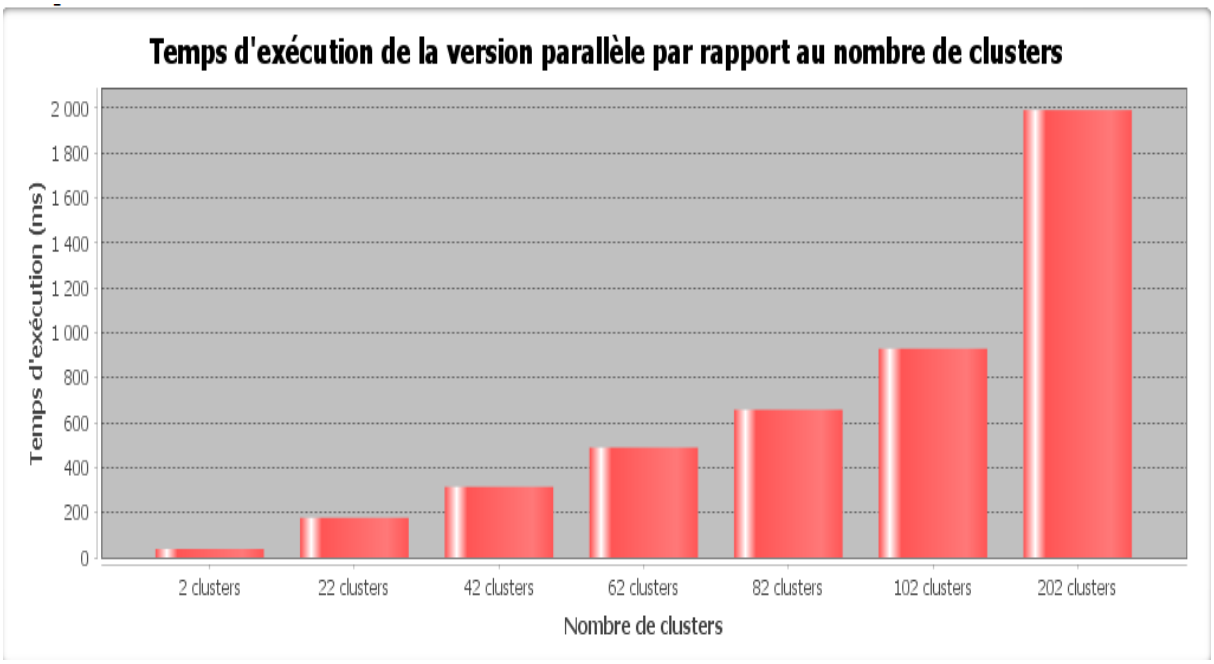


Figure III.8: Temps d'exécution de la version parallèle par rapport au nombre de clusters.

La figure III.8 montre qu'avec 2 clusters le temps n'a pas dépassé 100 ms mais avec 202 clusters le temps est dans les alentours des 2000 ms. On remarque que plus le nombre de clusters est important plus le temps d'exécution est grand.

La figure suivante montre la comparaison entre les deux versions par rapport au temps d'exécution.

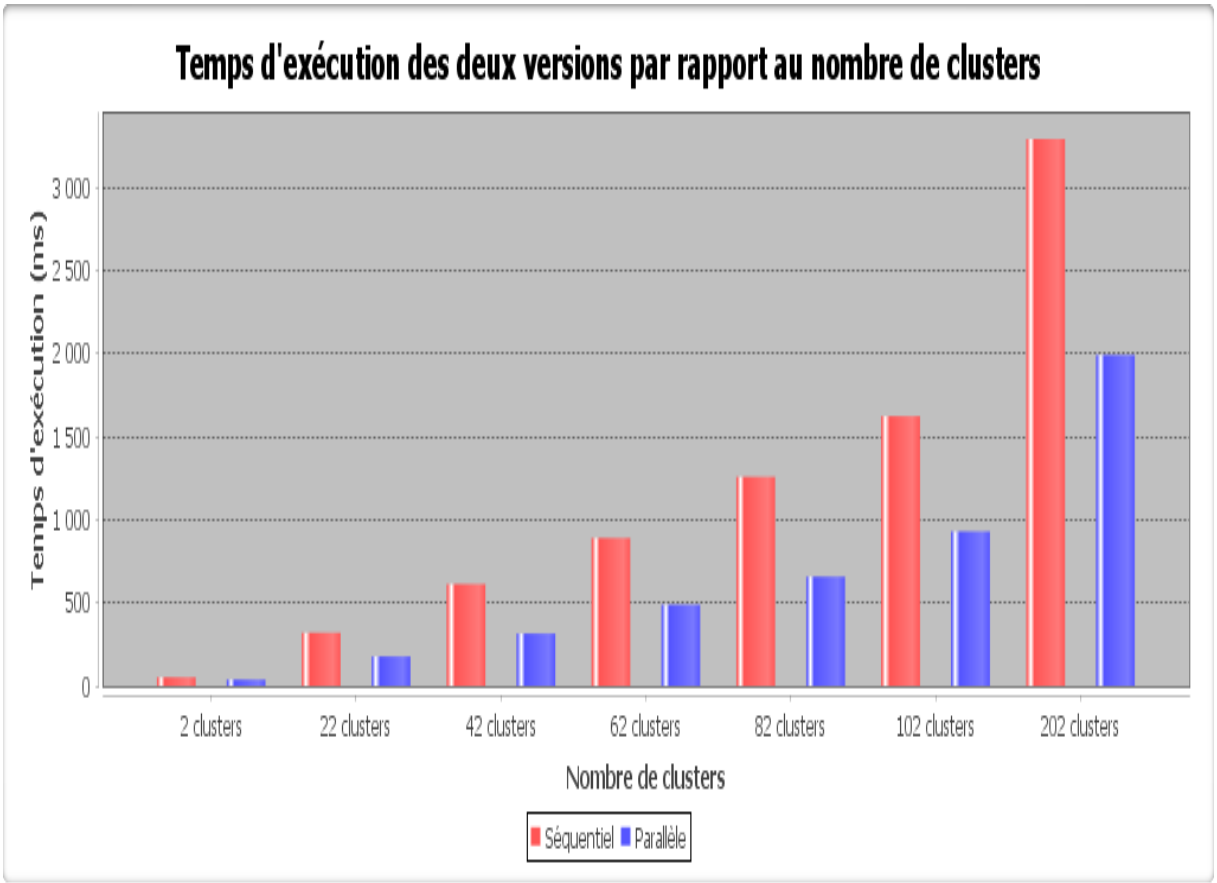


Figure III.9: Temps d'exécution des deux versions par rapport au nombre de clusters.

La figure III.9 montre que la version parallèle a amélioré nettement le temps d'exécution de la version séquentielle.

Le tableau suivant présente le taux de réduction du temps d'exécution de la version parallèle par rapport à la version séquentielle.

Nombre de clusters	Temps d'exécution en mode séquentiel (ms)	Temps d'exécution en mode parallèle (ms)	Taux de réduction (%)
2	51	38.2	25.1
22	320.9	176.6	44.97
42	612.6	314.1	48.73
62	889.9	489.3	45.02
82	1158.3	657	43.28
102	1622.4	928.5	42.77
202	3290.7	1990.1	39.52

Tableau III.1: Taux de réduction du temps d'exécution de la version parallèle par rapport à la version séquentielle.

Le tableau III.1 montre que la version parallèle a donné de meilleur résultat par rapport à la version séquentielle. Dans le cas de 42 clusters, le taux de réduction a pu dépasser le 48% ce qui est considéré comme un résultat satisfaisant.

La figure III.10 représente ces résultats sous forme d'histogrammes :

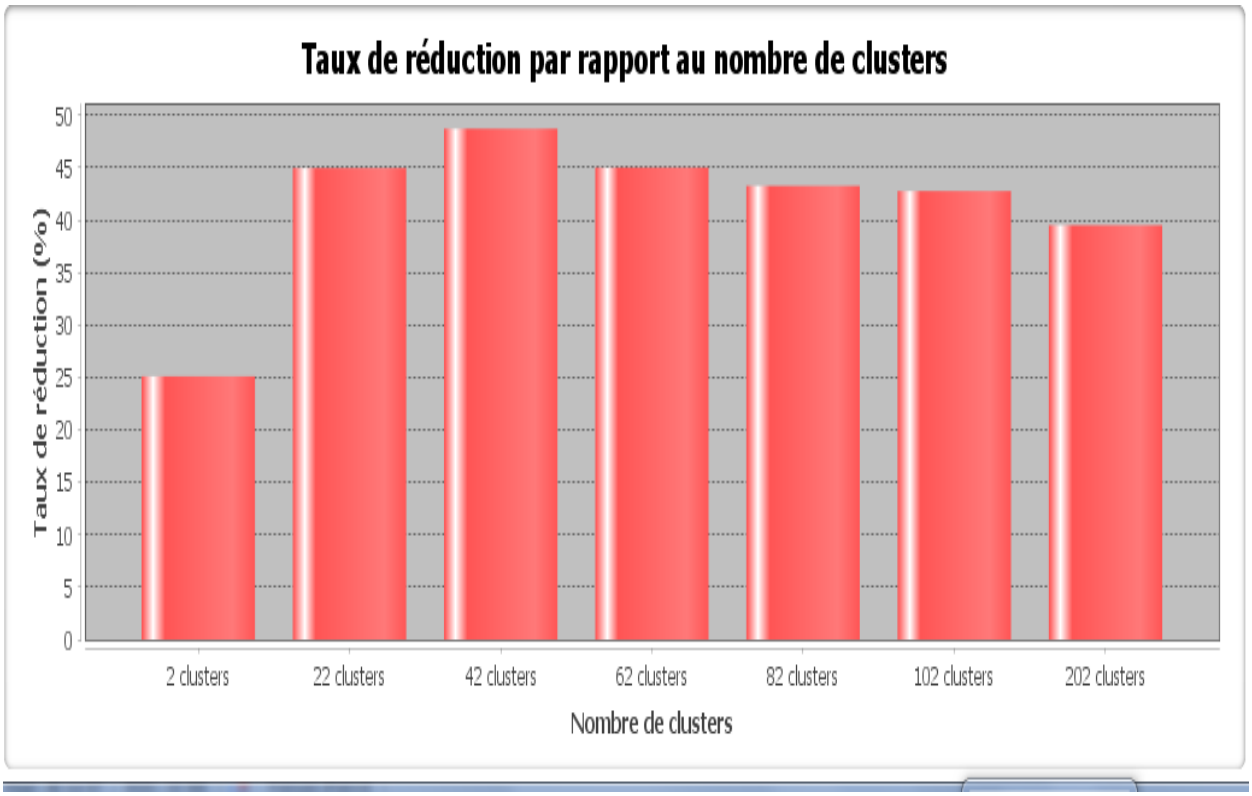


Figure III.10: Taux de réduction par rapport au nombre de clusters.

La figure III.10 montre que le taux de réduction est très satisfaisant (entre 25% et 48%).

III.6.2 Nombre de clusters fixe

Dans le deuxième cas, nous avons fixé le nombre de clusters à 100, et nous avons varié le nombre de nœuds de 100 à 4000.

La figure ci-dessous montre le temps d'exécution de la version séquentielle par rapport au nombre de nœuds.

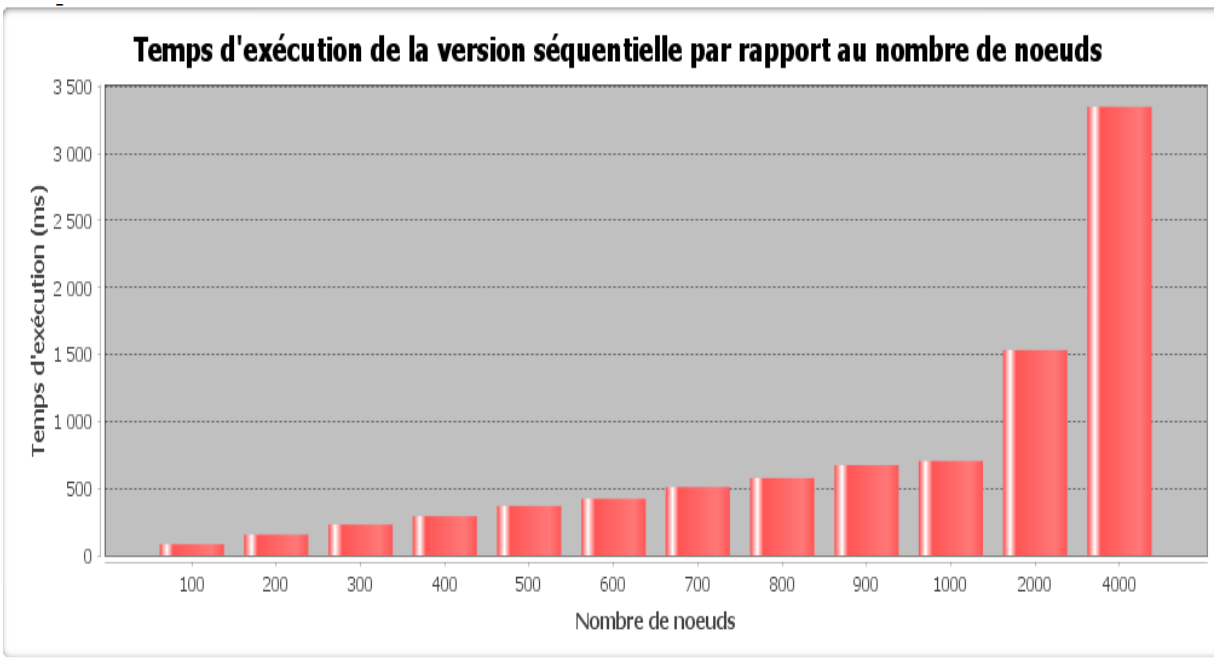


Figure III.11: Temps d'exécution de la version séquentielle par rapport au nombre de nœuds.

La figure III.11 montre que plus le nombre de nœuds est important, plus le temps d'exécution est grand; de 100 à 700 nœuds le temps n'a pas dépassé 500 ms, mais avec 2000 nœuds le temps a dépassé les 1500 ms et avec 4000, le temps a dépassé les 3000 ms.

La figure III.12 montre les résultats obtenus de la version parallèle.

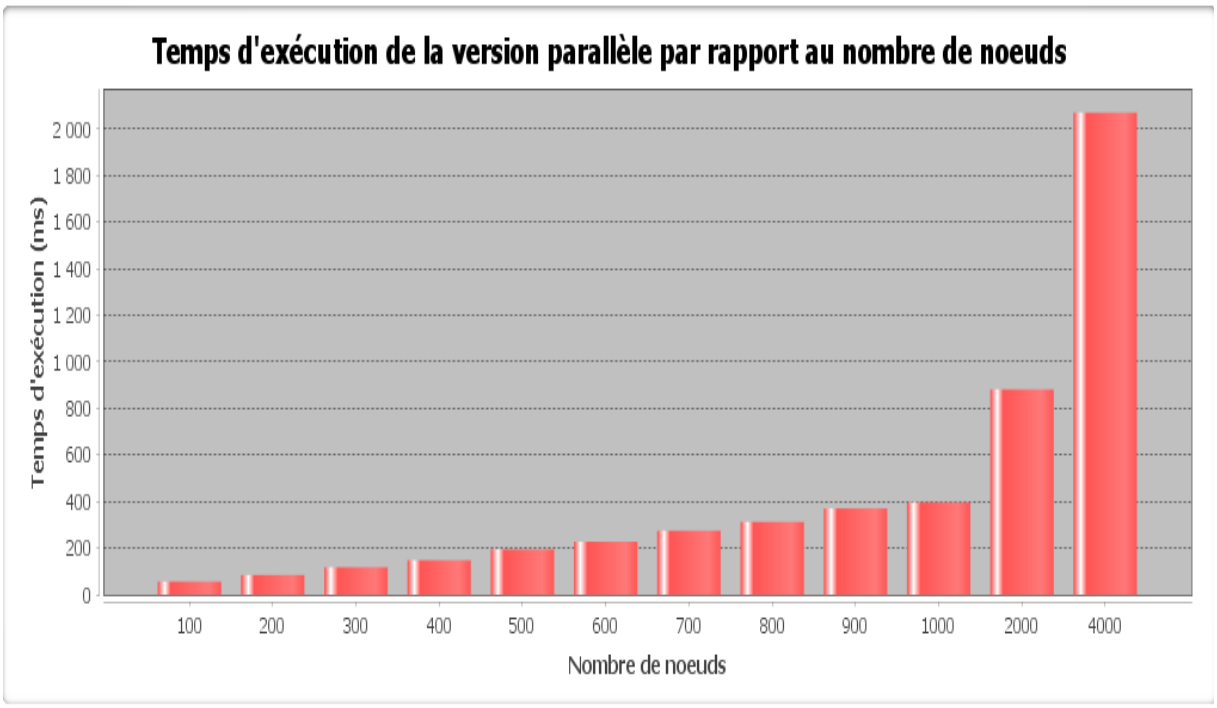


Figure III.12: Temps d'exécution de la version parallèle par rapport au nombre de nœuds.

La figure III.12 montre que plus le nombre de nœuds est important, plus le temps d'exécution est grand; de 100 à 1000 nœuds le temps n'a pas dépassé 500 ms, mais avec 2000 nœuds le temps a dépassé les 1000 ms et avec 4000 nœuds, le temps a dépassé 2000 ms.

La figure suivante montre les résultats obtenus par les deux versions.

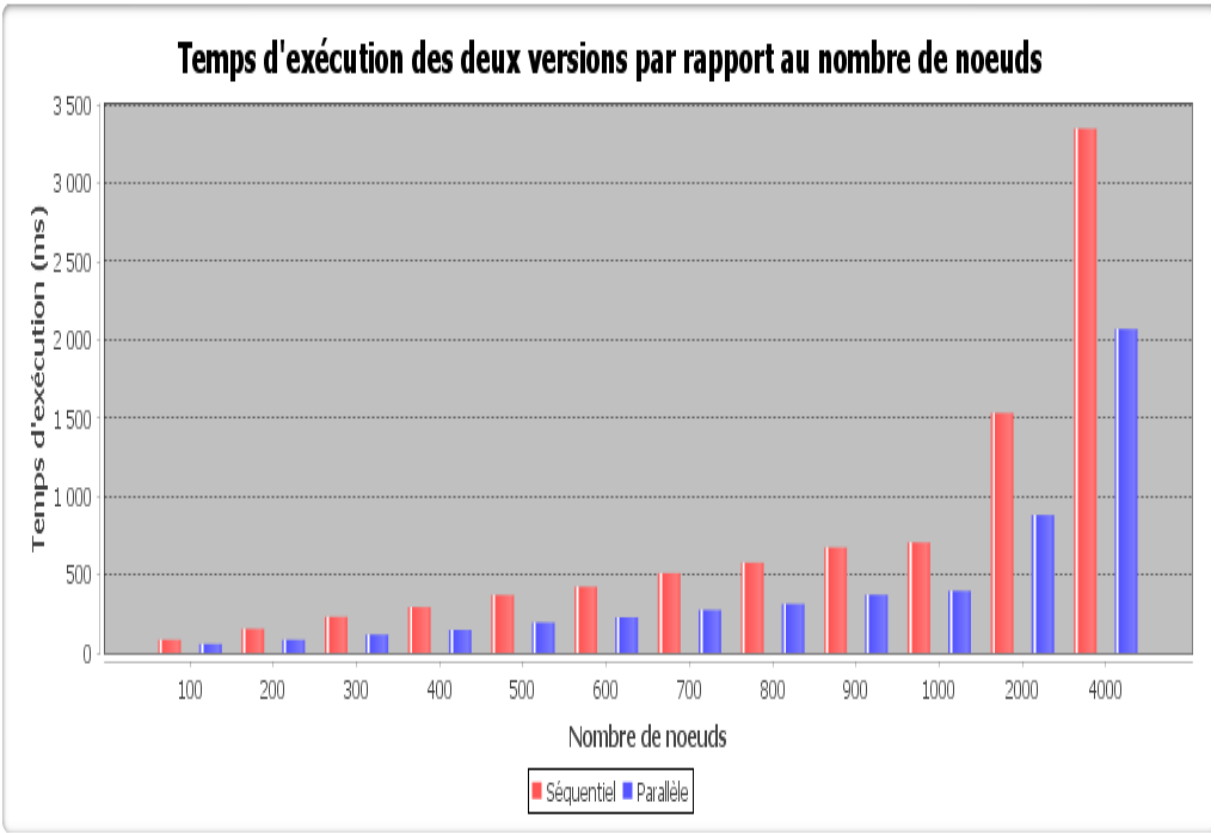


Figure III.13: Temps d'exécution des deux versions par rapport au nombre de nœuds.

La figure III.13 montre que la version parallèle a amélioré nettement le temps d'exécution de la version séquentielle.

Le tableau suivant présente le taux de réduction du temps d'exécution de la version parallèle par rapport à la version séquentielle.

Nombre de nœuds	Temps d'exécution en mode séquentiel (ms)	Temps d'exécution en mode parallèle (ms)	Taux de réduction (%)
100	83.4	56.1	32.73
200	154.2	84	45.53
300	230.2	116.9	49.22
400	292.9	147.4	49.68
500	369.2	194.1	47.43
600	422.5	227.2	46.22
700	510.2	274.7	46.16
800	576.7	311.9	45.92
900	673.9	370.3	45.05
1000	704.8	396	43.81
2000	1532.3	880.6	42.53
4000	3348.6	2069	38.21

Tableau III.2: Taux de réduction du temps d'exécution de la version parallèle par rapport à la version séquentielle.

Le tableau III.2 montre que la version parallèle a donné un résultat meilleur par rapport à la version séquentielle. Dans le cas de 400 nœuds, le taux de réduction a pu dépasser les 49% une chose qui est considérée comme un résultat très satisfaisant. La figure III.14représente ces résultats sous forme d'histogrammes :

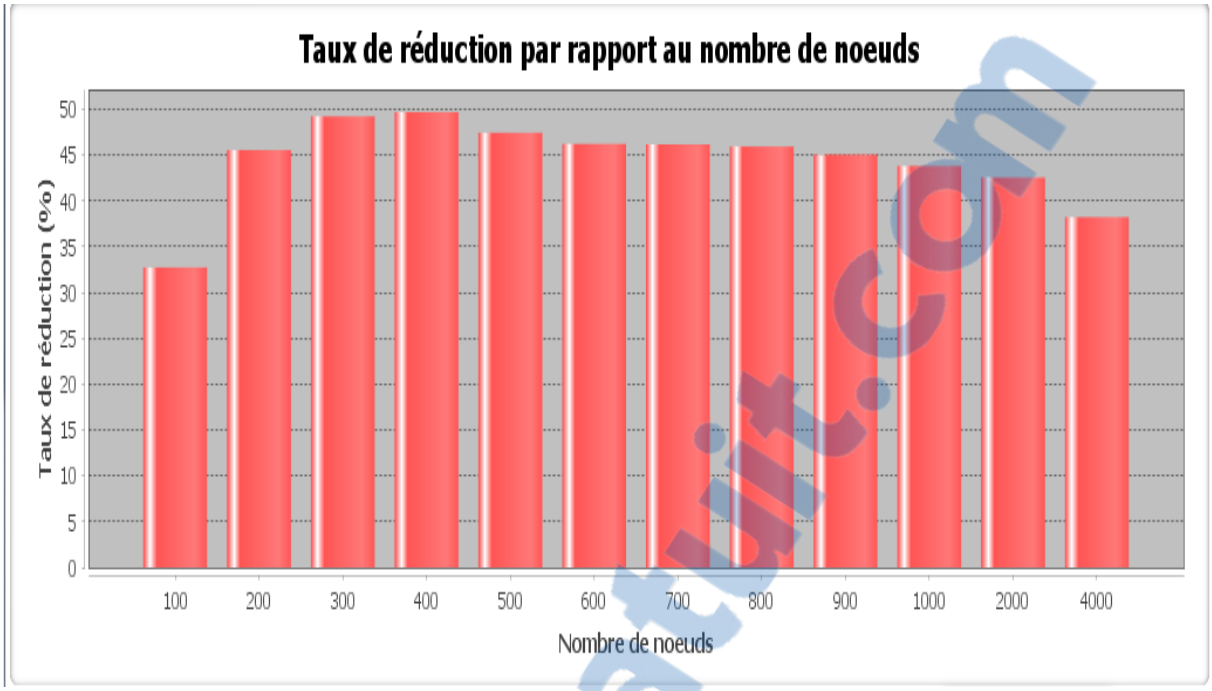


Figure III.14: Taux de réduction par rapport au nombre de nœuds.

La figure III.14 montre que le taux de réduction est très satisfaisant (entre 32% et 50%).

III.7 Conclusion

Dans ce chapitre, nous avons présenté notre application qui consiste à réaliser l'affectation de nœuds RC dans un réseau de radio cognitive qui est déjà structuré sous forme de clusters.

Une version parallèle est implémentée à l'aide de la méthode k-ppv pour accélérer l'affectation des nœuds. Cette version a été comparée avec la version naïve (séquentielle) et a donné un résultat meilleur en termes temps d'exécution par rapport à la version de base.

CONCLUSION GÉNÉRALE

La radio cognitive est une nouvelle technologie qui est apparue pour répondre aux besoins des utilisateurs dans le domaine de télécommunication. Ce concept a pour but d'améliorer les performances des réseaux sans fil en termes de gestion spectrale.

Dans ce projet de fin d'études, nous avons utilisé la méthode k-ppv afin de réaliser une affectation rapide des nœuds RC dans un réseau de radio cognitive qui est déjà organisé sous forme de clusters. Nous avons donc proposé une version parallèle de cette méthode et nous avons comparé les résultats obtenus par rapport à la version de base (séquentielle). D'un point de vue modélisation, nous avons opté pour une approche multi-agents et nous avons choisi JADE comme plate-forme SMA pour la réalisation des différentes simulations.

Les résultats obtenus montrent que la version parallèle de la méthode est plus performante que la version séquentielle. Le taux de réduction relatif au temps d'exécution de la version parallèle par rapport à la version séquentielle a dépassé les 49%, un résultat qu'on peut le considérer comme très satisfaisant.

Comme perspective à ce travail, nous pensons qu'il serait intéressant de traiter un nombre plus important de nœuds à affecter mais pour cela il faut disposer d'une machine dotée de plusieurs CORES (plus de 4 CORES). Il serait aussi intéressant de traiter le cas des pannes des nœuds et comment les clusters seront modifiés suite à ces pannes. Considérer aussi des clusters mobiles serait intéressant également comme perspective.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Mitola, Joseph, and Gerald Q. Maguire Jr. Cognitive radio: making software radios more personal. Personal Communications, IEEE 6.4 (1999): 13-18.

[2] Glisic S., «ADVANCED WIRELESSNETWORKS Cognitive, Cooperative and Opportunistic 4G Technology », Second Edition, University of Oulu, Finland.

[3] Amraoui Asma, « Vers une architecture multi-agents pour la radio cognitive opportuniste », thèse de doctorat MID, Université de Tlemcen. 2015.

[4] AmraouiIkram, BengherraWafaa, « Allocation de ressources dans un réseau de radio cognitive en se basant sur les méta-heuristiques:BatInspiredAlgorithm et Bee ColonyAlgorithm », PFE Master RSD. Université de Tlemcen. juin 2015.

[5] Mitola, J. (2009). Cognitive radio architecture evolution. Proceedings of the IEEE, 97(4), 626-641.

[6] A. Amraoui, W. Baghli, « Synthèse de profils applicatifs dans le cadre des réseaux radio cognitive », Mémoire de fin d'études pour l'obtention du diplôme de Master MID. Université de Tlemcen.Juillet 2011.

[7] Ngom, I., & Diouf, L. (2007). « La radio cognitive ». Master Professionnel Télécommunications, 2008.

[8] A.Metref, « Contribution à l'étude du problème de Synchronisation de porteuse dans le contexte de la Radio Intelligente », Novembre 2010.

[9] Palicot J. Cognitive radio: an enabling technology for the green radio communications concept. In Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly 2009 Jun 21 (pp. 489-494). ACM.

[10] Badr Benmammar. Présentation de la radio cognitive. 3rd cycle. 2012. disponible sur : http://hal.archives-ouvertes.fr/docs/00/68/23/44/PDF/RC-Pres-Benmammar.pdf.

[11] Bendella Med Saléh, « Gestion de spectre dans les réseaux de radio cognitive par la formation de coalitions », Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique, Université de Tlemcen, 2014.

[12] Taleb Mohammed Housseyn, « Implémentation du clustering dans un réseau de radio cognitive en utilisant la plate-forme JADE », Mémoire de fin d'études pour l'obtention du diplôme de Master RSD. Université de Tlemcen. Juin 2015.

[13] Amraoui A, Benmammar B, Bendimerad FT. Accès Dynamique au Spectre dans le Contexte de la Radio Cognitive. In2ième édition de la conférence nationale de l'informatique destinée aux étudiants de graduation et de post-graduation 2012 Apr 16.

[14] B. Benmammar, A. Amraoui, F. Krief, « A Survey on Dynamic Spectrum Access Techniques in Cognitive Radio Networks », International Journal of Communication Networks and Information Security (IJCNIS), vol. 5, No. 2, pp: 68-79. ISSN: 2076-0930(Print), ISSN: 2073-607X (Online), August 2013.

[15] Ibtissem Larbi, Badr Benmammar, « Négociation de spectre dans les réseaux de radio cognitive », Rapport de recherche, Laboratoire de télécommunications de Tlemcen (LTT), Université Abou Bekr Belkaid Tlemcen, 2013.

[16] Benmammar B, Amraoui A. Radio resource allocation and dynamic spectrum access. John Wiley & Sons; 2013 Feb 5.

[17] O. Boissier, S. Gitton, P. Glize, « Caractéristiques des Systèmes et des Applications », dans Systèmes Multi-Agents, vol. 29, pp. 25-54, Editions TEC & DOC, 2004.

[18] Wooldridge, Michael, and Nicholas R. Jennings. Intelligent agents: Theory and practice. The knowledge engineering review 10.02 (1995): 115-152.

[19] Moraïtis, Pavlos. Paradigme multi-agent et prise de décision distribuée. Diss. 1994.

[20] Ferber, Jacques, and Jean-François Perrot. Les systèmes multi-agents: vers une intelligence collective. InterEditions, 1995.

[21] Müller, Jean-Pierre. Des systèmes autonomes aux systèmes multi-agents: Interaction, émergence et systèmes complexes. Mémoire d'habilitation à diriger des recherches, Université de Montpellier II (2002).

[22] Russel, Stuart, and Peter Norvig.Artificial Intelligence: A Modern Approach, 2003. EUA: Prentice Hall.

[23] Finin, Tim, et al. KQML as an agent communication language. Proceedings of the third international conference on Information and knowledge management.ACM, 1994.

[24] O.GutknechtetJ. Ferber, Madkit: A generic multi-agent platform.In Proceedings of the fourth international conference on Autonomous agents, (2000)78–79.

[25] S. Luke, C. Cioffi-Revilla, L. Panaitet K. Sullivan, MASON: A New Multi-Agent Simulation Toolkit, 2004.

[26] S. Tisue et U. Wilensky,Netlogo : Design and implementation of a multi-agent modeling environment, In Proceedings of Agent 2004 (2004) 7–9.

[27] E.Amouroux, T.Q. Chu, A. Boucher et A. Drogoul,Gama : an environment for implementing and running spatially explicit multi-agent simulations. In: Agent computing and multi-agent systems, Springer, 2009.

[28] N. CollieretM. North, Repast HPC: A platform for large-scale agent-based modeling, Wiley, 2011.

[29] G. Cordasco, R. Chiara, A. Mancuso, D. Mazzeo, V. Scarano et C. Spagnuolo, A Framework for Distributing Agent-Based Simulations, Parallel Processing Workshops, Lecture Notes in Computer Science,7155 (2012) 460-470.

[30] E.S. Angelotti, E.E. ScalabrinetB.C. Avila, Pandora: a multi-agent system using paraconsistentlogic,In Computational Intelligence and Multimedia Applications, Proceedings. Fourth International Conference IEEE, 2001.

[31] S.Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth et C. Greenough,Exploitation of high performance computing in the flame agent-based simulation framework, In Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication, WashingtonUSA, 2012.

[32] F. Bellifemine, A. Poggi et G. Rimassa, Jade a FIPA compliant agent framework, London, 1999.

[33] http://jade.tilab.com/. Consulter en mars 2018.

[34] http://www.fipa.org/. Consulter en mars 2018.

[35] https://code.google.com/archive/p/mygoal/wikis/JFreeChartTutoriel20071202.wiki.

Consulter en mai 2018.

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS

Acronyme	Signification
AACR	Adaptative Aware Cognitive Radio
ACC	Agent Communication Channel
AMS	Agent Management System
ACL	Agent Communication Language
AIP	Agent Interaction Protocol
CH	Cluster Head
CM	Cluster Membres
CORBA	Common Request Broker Architecture
DF	Director Facilitor
EMI	Electro Magnetic Interference
FIPA	Foundation for Physical Intelligent Agents
IA	Intelligence Artificielle
IAD	Intelligence Artificielle Distribuée
JADE	Java Agent DEvelopment
K-ppv	K-Plus Proches Voisins
KQML	Knowledge Query and Manipulation Language
KTH	Institut Royal de Technologie
PFE	Projet de Fin d'Etude
QOI	Quality Of Information
QOS	Quality Of Sevice
RC	Radio Cognitive
RF	Radio Frequency
RMI	Remote Method Invocation
RRC	Réseau de Radio Cognitive
SDR	Software Defined Radio
SMA	Système Multi-Agents
SP	Sensory Perception

ANNEXE

Partie de code de la version parallèle

➢ Code de l'agent principal :

```
public class Main_agent extends Agent
{
ArrayList<Point>listePP;
ArrayList<Distance>listeD ;
staticint N, k; // deux variable partagées entre les agents
intnbr_points=4 ; // nombre de nœuds à affecter
staticint i=1;
        protected void setup()  {
Object []args =getArguments();
/* récupérer la liste des nœuds déjà organisé   dans  des clusters */
if(args.length==1)
listePP=(ArrayList<Point_Para>) args[0];
    N=listePP_.size();
    k=N;
/* Les x des points du 1er cluster  sont variés entre [1,401] et les y sont variés entre [20,80]
La distance séparant 2 clusters et égale à 10
Tous les clusters ont la même distance d'intervalle (400) */
intmaxX=811; // correspond à l'intervalle globale [1, maxX] ; 811 pour le cas de 2 clusters
//Lancer le timer pour calculer le temps d'exécution
 long temps_debut=System.currentTimeMillis();
/* la boucle sert à envoyer des points généré par random aux agents pour calculer les
distances entre le point et les autres points déjà existants  */
for(int m=1;m<nbr_points+1;m++)
 {
Point P= new Point ((int) ((Math.random()*(maxX-1+1))+1) ,(int) ((Math.random()*(80-
1+1))+1),0);
  AID id=new AID("agent_para"+m,AID.ISLOCALNAME);
envoyer(P,listePP,id);
  }
addBehaviour(new CyclicBehaviour(this)

                public void action()
                { ACLMessagemsg_rec= receive();
                        if (msg_rec!=null)
```

```
                                { try {

// Envoie_obj contient la liste des points et le point à affecter
Envoie_objenv=(Envoie_obj) msg_rec.getContentObject();
listeD= env.getListeD();
listePP=suite(listePP,env.getP());
i++;
if(i==nbr_points+1)
{long temps_fin=System.currentTimeMillis();
longresultat =temps_fin-temps_debut;
                                    i=1;
System.out.println("temps écoulé : version parallèle = "+resultat);
                                } // fin if
}catch (UnreadableException ex) {
                                } //fin catch
                                }//fin if
                               }// fin méthode action
                    });
   }

  /* Une méthode qui permet l'envoie d'un point à affecter, une liste des points à l'agent
identifié par id*/
        public voidenvoyer(Point p, ArrayList<Point > li, AID id)
{
ACLMessagemsg = new ACLMessage(ACLMessage.INFORM);
try{
li.add(0, p); // le point à affecter est ajouter à la liste des points au début (position 0)
     // l'objet à envoyer est la liste li
msg.setContentObject(li)

msg.addReceiver(id);
send(msg);
li.remove(0);
}catch(IOException o)
{ }
    }
```

```
   // La méthode « suite » permet de poursuivre le traitement d'affectation
   public ArrayList<Point >suite(ArrayList<Point >lp, Point P)

{
int var=lp.size()-1;
        Distance di ;
for(int i=k+1; i<=N;i++)
{
       di=P.distance_nouv_point(P, lp.get(var));
listeD = P.insertion_nouv_dist(di, listeD);
var--;
}

P.classe_majoritaire(listeD);
P.setGroupe(P.classe_majoritaire(listeD));
lp.add(P);
   N++;
returnlp;
    }
}
```

➢ Code d'un agent ordinaire:

```
public class Thread_Ag_Para extends Agent
{   ArrayList<Distance >listeDP ;
ArrayList<Point >listePP;
   Point P;
protected void setup()
{addBehaviour(new CyclicBehaviour(this)
                { public void action()
                        { ACLMessagemsg_rec= receive();
                                if (msg_rec!=null)
                                { try {
listePP=(ArrayList<Point>)msg_rec.getContentObject();
/* après la récupération de la liste, il faut séparer le point à affecter de la liste*/
P=listePP.get(0);
listePP.remove(0);
listeDP=traitement(P,listePP);
Envoie_objobj= new Envoie_obj(listeDP,P);
ACLMessagemsg = new ACLMessage(ACLMessage.INFORM);
msg.setContentObject(obj);
msg.addReceiver(msg_rec.getSender());
send(msg);
} catch (UnreadableException ex) {
}   catch (IOException ex) {}
}
if(msg_rec==null) { block();}
}});
 }
/* La méthode suivante permet de calculer la distance entre le point à affecter et tous les
points existants*/
 public ArrayList<Distance>traitement(Point p, ArrayList<Point>listepp)
{
ArrayList<Distance >ld= new ArrayList<Distance >();
ld=p.distance_point(p,listepp);
ld=p.trier_distance(ld);
returnld;
}}
```