République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieure et de la recherche Scientifique
Université d'Adrar
Faculté des Sciences et de la Technologie
Département des Mathématiques et Informatique

Mémoire

En vue de l'obtention du Diplôme de Master en Informatique

Option: Réseaux et Systèmes Intelligents

Sous le Thème

Approche multi-agents pour résoudre le problème d'emploi du temps

Préparé par:

Melle. Meriem SADKI et Melle. Selma BENALLA

Sous la Direction de Mr. Mohamed KOHILI

Année Universitaire 2014 / 2015.

Remerciement

Ce mémoire est le résultat d'un travail de recherche de prés d'une Année, en préambule, nous tiendrons à adresser tous nos remerciements aux personnes avec lesquelles on a pu échanger ainsi que toute personne qui nous a aidés à la réalisation de ce projet.

En commençant par remercier tout d'abord Monsieur ***KOHILI*** *l'encadreur et le directeur de ce mémoire tout pour l'orientation, la patience et pour le temps qu'il nous a consacré.*

Nous adressons nos plus sincères remerciements aux membres des jurys pour l'intérêt qu'ils ont porté à notre recherche.

Enfin nous tiendrons à témoigner toute notre gratitude à « Melle. KEBIRE » pour ses efforts.

Dédicace

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie mon travail à mes très chers, respectueux et magnifiques parents qui m'ont soutenu tout au long de ma vie

- *A ma grande mère et à la mémoire de mon grand père*
- *A mes sœurs que j'ai tant aimé qu'elles assistent à ma soutenance*
- *A mes chers amis sans exception*
- *A toute les personne qui m'ont encouragé ou aidé au long de mes études*

Melle. Meriem SADKI

Je dédie ce projet à

- ***Mes parents***
- *Mes grands parents*
- *Mon mari.*
- *Mes belles-sœurs et beaux-frères*
- *toute la famille :* ***BENALLA&&AIMECHE&&TOUTAOU.***
- *Ma très chère collègue ~~Meriem~~ et tous mes amies.*
- *Ma promotion de l'année 2015 ;*

Sans oublier tous ceux que je n'ai pas cités.

~~BENALLA SELMA~~

Table des matières

Table de la matière

Liste des Figures

Liste des Figures

Résumé

Ce mémoire met en lumière dans le contexte générale la problématique de planification horaire dans les établissements scolaire et plus précisément le problème de gestion du temps du cours universitaire, ce dernier est définit comme une réservation des ressources humaines et matérielles dans un intervalle du temps avec pris-en compte certains conditions appelé « contraintes », ce problème est bien connu comme NP-difficile, autrement dit aucun des algorithmes existants n'est capable de résoudre à lui seul toutes les instances des problèmes dans un temps polynomial. En effet, la génération d'emploi du temps préoccupe toute société ou établissement actif ce qui a incité les chercheurs à proposer des méthodes et des approches afin d'obtenir un meilleur résultat. Le but de ce projet est de générer un système d'emploi du temps concerne les cours universitaire en utilisant l'approche multi-agents.

Mots clés : Système multi-agents, contraintes, problème emploi du temps.

Abstract

The aim of this project is to explain a problem of university courses timetabling, this can be described as the allocation of resources (time slots, rooms,) to events (lesson, TD and TP) in condition to respect two types of constraints (Hard and Soft). The academic timetabling generation has been classified as NP-hard problem this means that it is unlikely that it will be possible to find fast Algorithm to solve this problems.

So over the last three decades many approaches and models have been proposed like operational research, Algorithm genetic …etc to solve this problem, in this project we will try to generate a program of university courses timetable with multi-agents system.

Keys Words: multi-agents system, constraints, problem of university courses timetabling.

الملخـــــــص

في هذه المذكرة قمـنا بعرض مشكلة الجدولــة الزمـنية عـامة و جــدولة المحــاضرات الجامعية خاصة بحيث تكمن صعوبـة هذه المشكلة كصعوبة أكثر المشــاكل تعقيدا و المتمثلــة في NP-hard بمعنى أنَ العملـيات اللازمة لحلّ المشكلة تزداد بشـكل أســي طرديا مع حجم المشكلة خاصة إذا كــانت الهيئة تتميز بكثرة الأحداث و تنوعها مثل الجامعة.و نتيجة لهذه الصعوبات قدَم الكثير من الباحثين العديد من الحلول و الخوارزميات لحل هذه المشكلة مثل خوارزمية التطورية ....الخ.

تتركز هذه المذكرة على معالجة الآلية لجدولة الزمنية الخاصة بالمحاضرات الجــامعية و التي تعرف العمــلية بتخصيص مــوارد مـادية و بشرية في فتـرات محددة من الزمـن مع مـراعاة بـعض القيــود و الشــروط سواء كانت صارمة أو ليـنة و ذلك باستخـدام خوارزمية الأنظمة متعددة العملاء.

الكلمــات المفتاحية: الجدولة الزمنية، نظــام متعدد العملاء، القيـــود.

Introduction générale

1. Problématique et l'objectif de mémoire

Chaque jour, La machine est s'installe peu à peu dans notre vie quotidienne, l'être humain est toujours cherche de faciliter les taches et simplifier les conditions de la vie, en revanche ces travaux sont très difficiles et besoin du temps et des capacités physiques et intellectuelles comme : effectuer des calcules complexes, et cela sont causés beaucoup des problèmes. Alors que la communauté scientifique essayent de trouver les meilleurs outils d'automatisation pour minimisé intervention humain.

Et parmi les problèmes qui se trouve dans de nombreux domaines de la vie professionnelle on a le problème de gestion de temps ou planification horaire. Dans les hôpitaux : un certain nombre de personne constitués d'infirmiers et de médecins doivent être attribués aux postes de travail de manière à obéir à certaines règles de gestion des hôpitaux. Dans les usines : Les usines ont besoin d'optimiser le rendement de leurs chaines de production, alors chaque machine de production effectue une tâche particulière à une période fixée a priori. Donc cela nécessite souvent une élaboration périodique et stratégique de planning de travail personnel.

Le problème d'emploi du temps est une instance des problèmes de planification d'horaire, il est connu comme un NP-complet. Autrement dit, aucun des algorithmes existants n'est capable de résoudre à lui seul toutes les instances du problème dans un polynomial. Ce problème est situé dans plusieurs types d'organisation tel que : les établissements éducatifs notamment dans les universités qui consomment de nombreuses ressources et donc financières. Burke et ses collègues subdivisent PET universitaire en deux catégories principales : les cours et les examens.

L'emploi du temps du cours universitaire est une affectation des personnes et matérielles dans un certain nombre de créneau horaire en respectant les contraintes. Cette affectation a pour but de bien organiser, d'éviter les conflits et construire un environnement de travail adéquat. Cependant la réalisation d'un emploi du temps manuellement est très compliquée et très couteuse. Cette difficulté explique pourquoi aujourd'hui on est besoin d'un ordinateur pour automatiser ces tâches et pour

générer des emplois du temps efficaces dans un temps acceptable en adoptant des outils basés sur des algorithmes d'optimisation robustes.

L'automatisation de PET du cours universitaire consiste de réalisé un système robuste capable de traiter un emploi du temps d'une façon automatique a fin d'obtenir un bon résultat, ce problème joue un rôle très important c'est pour cela beaucoup des chercheurs sont intéressé par ce type de problème et proposent des différents solutions mais chacun d'eux a ses propres méthodes.

La problématique de ce mémoire est mettre en scène la planification horaire et plus particulièrement le problème d'emploi du temps consacre les cours universitaire, ce dernier est défini comme un recherche d'allocation d'un ensemble des ressources physiques (cours, Salle,…) et un nombre limité des enseignants dans un intervalle du temps. De façon à satisfaire certaines conditions qui s'appelle « contraintes ». Le raison qu'on a choisi cette version de problème parce qu'on est dans le monde universitaire, on comprend bien la difficulté de ce problème et on espère bien de finir ce travail par implémenter une application qui va facilite la tache aux administrateurs.

L'objectif de ce mémoire est d'obtenir un système qui peut générer un emploi du temps hebdomadaire du cours universitaire en utilisant l'approche multi-agents qui se base par un groupe des agents coopératifs et qui permettre de traiter d'une façon décentralisé. Le but à atteindre est :

- Etablir un système qui réaliser un calendrier concernant université.
- Pris-en compte les privilèges des enseignants dont le but de rendre le système dynamique par rapport au environnement.
- Utilisant approche multi-agent pour améliorer le résultat et dépasser certain problème tel que « problème du temps ».

2. Plan de travail

A par l'introduction générale et la conclusion générale, ce mémoire est organisé en trois chapitre :

Chapitre I : IAD et système multi-agents.

Dans ce chapitre on a introduire le domaine de l'intelligence artificielle distribué et citer quelque description concernant la notion d'agent, ainsi que son caractéristique et architecture puis on a détaillé les différents domaines et plateformes.

Chapitre II : Le problème d'emploi du temps universitaire.

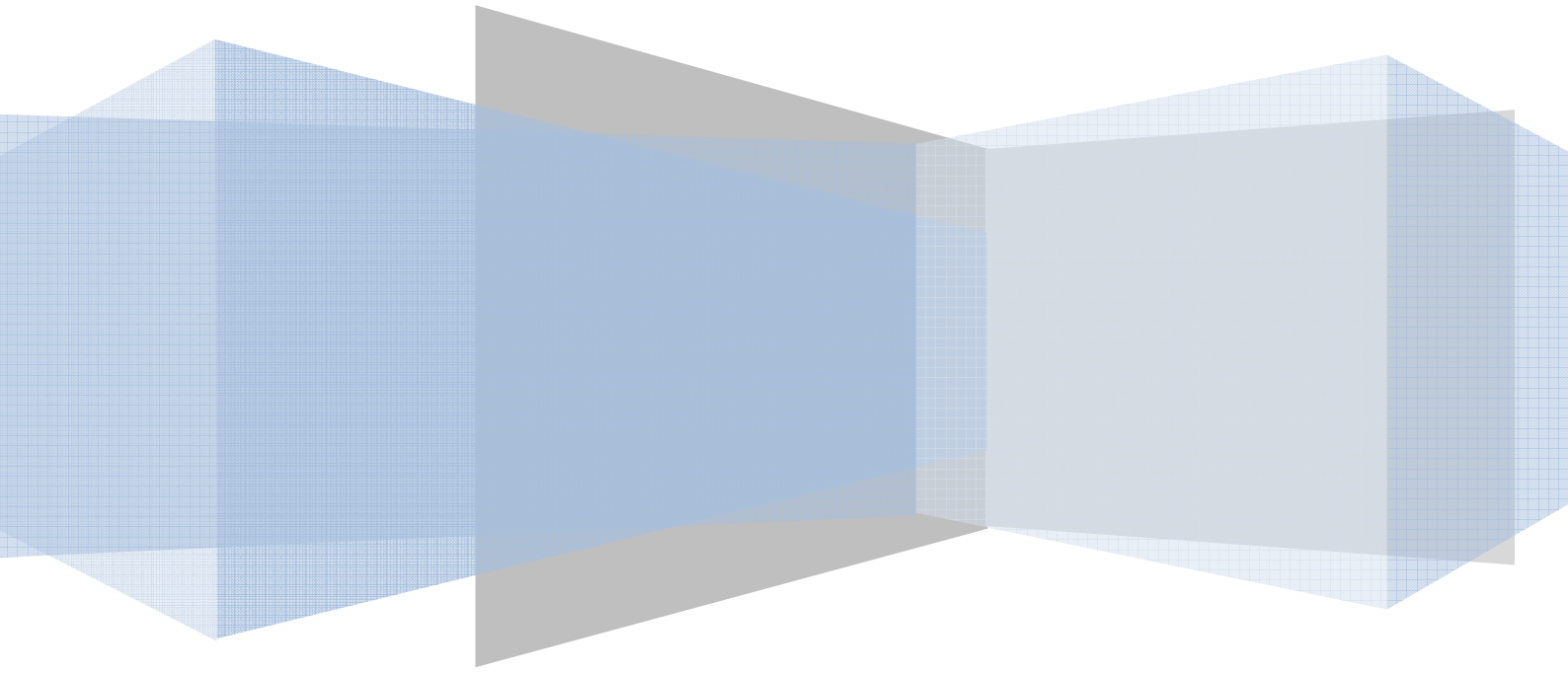
Le deuxième chapitre explique le problème d'emploi du temps qui est parmi les problèmes de satisfaction des contraintes, alors on va essayer de donner une définition formelle de CSP puis on va présente la problématique de gestion du temps avec ces caractéristiques …etc. Ainsi que les méthodes qui sont proposé pour résoudre ces genres des problèmes.

Chapitre III : La conception et implémentation.

Le troisième chapitre est basé sur la plus importante phase de conception pour bâtir notre système, c'est la phase de conception et de modélisation dont lesquelles on va essayer de donné des détailles sur l'architecture générale et la méthode que nous avons choisi. Puis on va passer à la phase de l'implémentation dont laquelle on a va présenter la réalisation de notre système et les petits tests afin d'évaluer les performances.

1ère Chapitre

IAD et Système Multi-Agents



1. Introduction

La révolution de l'IA[1] pendant les dernières années a été bouleversée le monde d'une manière assez profonde. Elle est clairement devenue incontournable. L'intelligence Artificielle remplace de plus en plus l 'Homme dans presque tous les domaines. Elle devient ainsi de plus en plus diffuser et distribuer dans des multiples objets et fonctionnalités qui sont amenés à coopérer. Et parce que les applications informatiques ont plusieurs problèmes tels que la taille, la complexité et la centralisation des systèmes informatiques, des efforts particuliers sont portés sur l'Intelligence Artificiel Distribué[2] et le Système multi Agents pour aborder ces problèmes. La décomposition du système en plusieurs agents permet d'avoir une meilleure réactivité et une meilleure adaptabilité à un environnement changeant. Par ailleurs, les interactions entre agents peuvent faire émerger des structures organisées (structures sociales), qui en retour contraignent et coordonnent le comportement des agents. Les systèmes multi - Agents s'appuient sur la métaphore de l'organisation collective, par opposition à l'Intelligence Artificielle qui s'appuie sur la métaphore du penseur isolé **[GUE, XX]**.

Dans ce chapitre, nous commençons par introduire Intelligence Artificielle Distribué passant sur la Définition, L'objectif et les branches, ensuite nous mettons l'accent sur le système multi-agent (SMA) à travers les points suivants :

- Notion d'agent.
- Définition de SMA.
- Les thèmes de recherche.
- La communication et les plateformes.

2. Intelligence Artificielle Distribué

L'évolution des domaines d'application de l'intelligence artificielle pour recouvrir des domaines complexes et hétérogènes tels que l'aide à la décision, la reconnaissance et la compréhension des formes, la conduite des processus industriels…etc., a montré les limites de l'approche classique de l'IA qui s'appuie sur centralisation de l'expertise au sein d'un système unique. Les travaux menés au début des années 70 sur la concurrence et la distribution ont contribué à la naissance d'une nouvelle discipline : Intelligence Artificielle Distribué (IAD) **[LAB, 93]**.

2.1. L'objectif de l'IAD

L'IAD a pour de remédier aux insuffisances de l'approche classique de l'IA en proposant la distribution de l'expertise sur un groupe d'agent devant être capables de travailler et d'agir dans un environnement commun et résoudre les conflits éventuels **[HOU et al, 14]**.

[1] Intelligence Artificielle
[2] Intelligence Artificielle Distribué

2.2. La définition de l'IAD

L'IAD est reconnue comme étant une discipline informatique qui s'intéresse à la modélisation et simulation des comportements humains dits intelligents tels que la perception, la prise de décision, la compréhension, l'apprentissage…etc.**[HUH, 87]**, cette tache est effectué par un ensemble d'entités distribuées qui représentent une sorte d'intelligence essayant d'atteindre des buts, autrement dit, permettant une société d'agents autonomes afin d'accomplir une tâche commune qui est en fait la résolution d'un problème particulier **[HOU et al, 14]**.

2.3. Les branches de l'IAD

L'IAD recouvre en réalité trois axes de recherche **[FRE et al, XX] [DUR et al, 90] [DAV, 80] [FEH, 83]** :

- **La résolution distribuée du problème:** elle cherche à déterminer comment repartir la résolution d'un problème entre plusieurs modules n'ayant que des connaissances partielles de celui-ci. Autrement dit, elle s'intéresse à la manière de diviser un problème particulier sur un ensemble d'entités distribués et coopérants. Elle s'intéresse aussi à la manière de partager la connaissance du problème et d'en obtenir la solution.
- **Les systèmes multi-agents :** cet axe concerne la coordination du comportement d'un ensemble d'agents autonomes et intelligents pour effectuer une action afin de résoudre un problème donnée.
- **L'Intelligence Artificielle parallèles :** Elle concerne le développement de langages et d'algorithmes parallèles pour l'IAD. L'IAP vise l'amélioration des performances des systèmes d'intelligence artificielle sans, toutefois, s'intéresser à la nature du raisonnement ou au comportement intelligent d'un groupe d'agents. ou chaque agent ou acteur peut être implanté sur un nœud différent d'un réseau de processeurs.

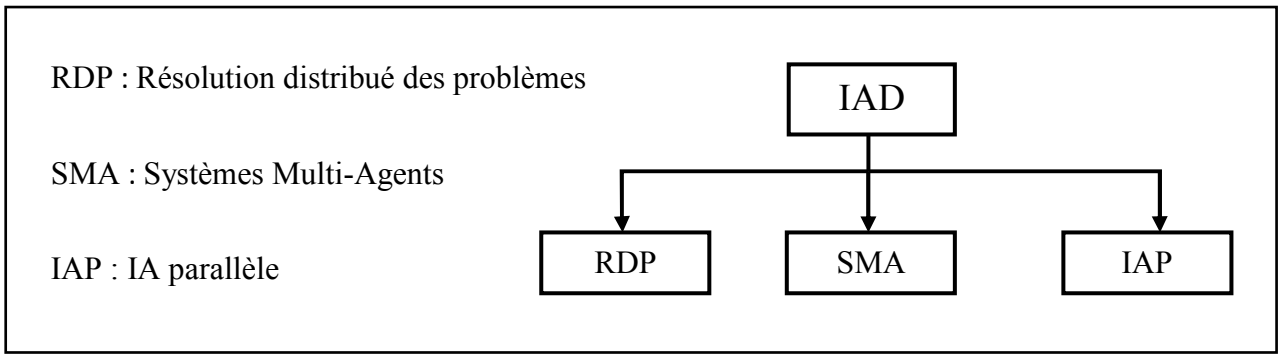


Figure I.1 : Les domaines de l'IAD

3. Notion d'agent

3.1. Qu'est ce qu'un Agent ?

Il ya énormément des définitions d'agents, elles se focalisent sur le même terme, mais différent le type d'application pour laquelle est conçu l'agent :

1. **Définition1***:* « Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu » **Wooldridge&Jennings** **[WEI, 99] [WOO, XX].**

2. **Définition2***:* « On appelle agent une entité autonome physique ou virtuelle, qui est capable d'agir sur un elle-même et sur son environnement, qui, dans un univers multi-Agents, pour communiquer avec d'autre agents, et dont le comportement et la conséquence de ses observations et des interactions avec des autres agents » **Ferber** **[FER, 95]**.

3. **Définition3***:*« Les agents intelligents sont des entités logiciels qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela, ils utilisent une sorte de connaissance ou de représentation des buts ou des désires de l'utilisateur» **IBM** **[GIL, 97] [Web1]**

4. **Définition4***:*« Les Agents intelligents exécutent sans interruption trois fonctions : perception des conditions dynamiques dans l'environnement ; action pour affecter des conditions dans l'environnement ; et le raisonnement pour interpréter des perceptions, résoudre des problèmes, mener des inférences et déterminer des actions.» **Hayes-Roth** **[FRA, 96] [HOU, 14]**

5. **Définition5***:*« Une entité devient un agent aussitôt qu'elle est capable d'exercer un contrôle local sur ses processus de perception, de communication, d'acquisition de connaissances, de raisonnement, de prise de décision ou d'exécution. » **Drogoul** **[DOG, 93]**

3.2. Les caractéristiques d'un Agent intelligent

Les chercheurs en intelligence artificielle s'accordent sur la nécessité de l'existence de quelques caractéristiques pour qu'on puisse parler d'agents. L'agent n'a pas forcement contient toutes les caractéristiques, à l'heure actuelle, aucun produit ne rassemble toutes ces caractéristiques **[GIL, 97] [CHA, 01]** :

- **Situé :** L'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.
- **Autonome :** L'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ces propres actions en fonction de son état interne et de son environnement.
- **Proactif :** L'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment, c'est-à-dire que l'agent fait sa propre activité et son propre but, et

ce dernière n'arrête pas en réponse aux messages reçus des autres agents, mais son activité est dirigé par son but.

- **Capable de répondre à temps :** L'agent doit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis.
- **Social :** certain agents sont interactifs avec des autres agents (logiciels ou humains) à fin d'accomplir des taches ou aider ces agents à accomplir les leurs, cette communication pourrait être par des moyens propriétaires ou par certain systèmes comme (KQML).
- **mobile:** L'agent doit pouvoir être multi-plate-forme et multi-architecture. Il doit pouvoir se déplacer sur le réseau où ils accomplissent des tâches sans que l'utilisateur ait le moindre contrôle sur celles-ci.
- **Adaptable :** L'agent a la capacité de modifier les propriétés de ses différentes taches afin de satisfaire les demandes internes et externes.

3.3. Les types d'agents

La notion d'agent peut on général classés en plusieurs types, les principaux types sont : agent cognitif dits « intelligent » et agent réactive dits « moins intelligent ». A ces deux types, on s'ajoute l'architecture hybride et l'architecture BDI conçue pour résoudre les problèmes.

3.3.1. Agent réactive

L'école réactive considère qu'il n'est pas nécessaire que les agents soient intelligents Individuellement pour que le système ait un comportement global intelligent, elle suppose que les agents sont très simples(ne possèdent pas une représentation de leur environnement ni de mémoire et leur comportement est de type « stimuli _ réponse ») et que l'intelligence émerge de l'interaction entre ces agents, ils se caractérisent par des agents qui ont la capacité de réagir rapidement à des problèmes simples, ils ne nécessitant pas un haut niveau de raisonnement. Le meilleur exemple de ce genre de systèmes constitués d'entité de faible intelligence mais dont la coopération conduit à des systèmes très intelligents, sont les sociétés d'insectes (les fourmis). La figure suivante présente l'architecture réactive.

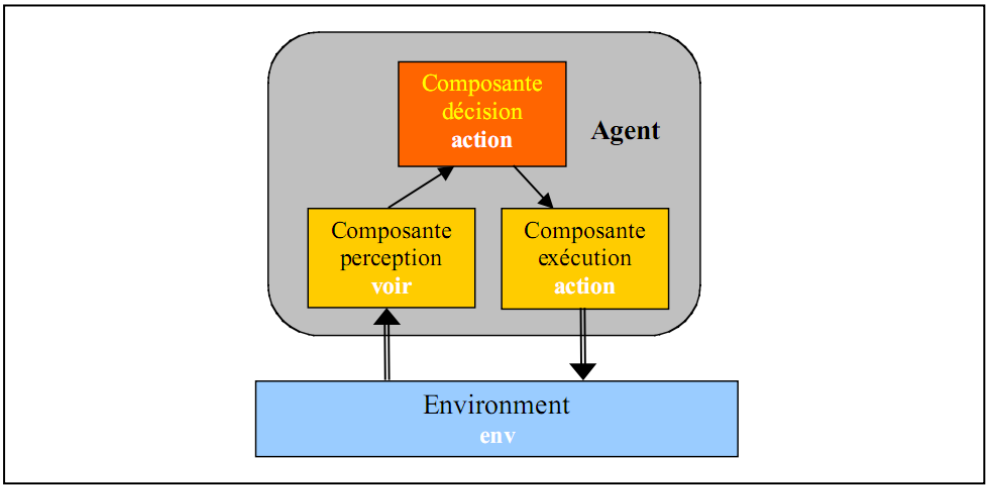


Figure I.2 L'architecture réactive [Web1]

3.3.2. Agent cognitifs

Les agents cognitifs sont fondés sur la coopération d'agents capables, à eux seuls, d'effectuer des opérations complexes**.** Chaque agent dispose d'une capacité de raisonnement, d'une aptitude à traiter des informations diverses liées au domaine d'application, et d'informations relatives à la gestion des interactions avec d'autres Agents et l'environnement. **[ZAH et al, XX]** Autrement dit, la structure d'architecture cognitive est munie d'une base de connaissance comprenant l'ensemble des informations et de savoir-faire nécessaire à la réalisation de sa tache ainsi qu'a la gestion des interactions.

3.3.3. Agent hybride

Une architecture hybride d'un agent intelligent est une architecture composée d'un ensemble de modules organisés dans une hiérarchie, chaque module étant soit une composante cognitive avec représentation symbolique des connaissances et capacités de raisonnement, soit une composante réactive. De cette manière, on combine le comportement proactif de l'agent, dirigé par les buts, avec un comportement réactif aux changements de l'environnement. En plus, on espère obtenir simultanément les avantages des architectures cognitives et réactives, tout en éliminant leurs limitations. **[web1]**

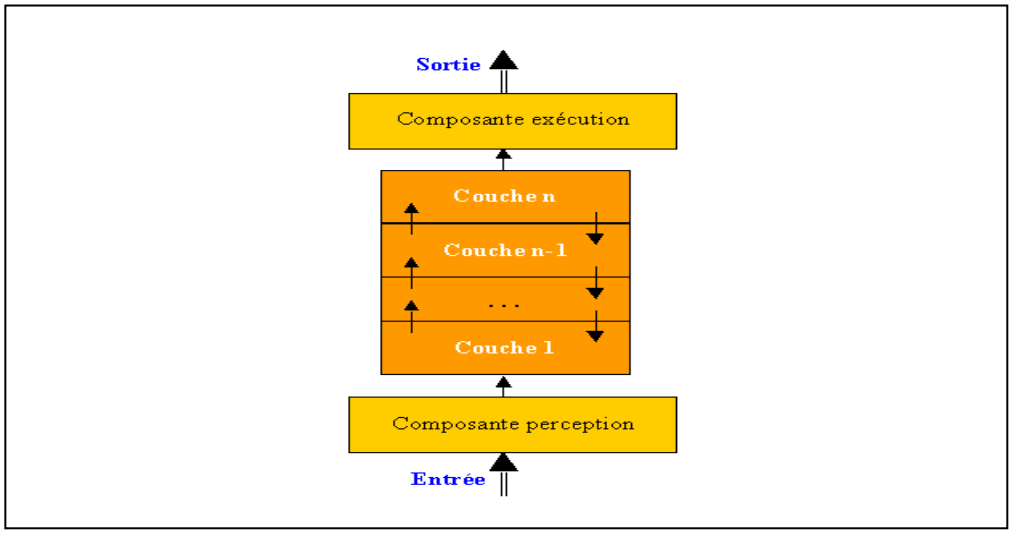


Figure I.3: L'architecture hybride **[Web1]**

3.3.4. Agent BDI

Pour aborder le problème classique de définition et d'implémentation d'agent de nombreuse architecture multi-agents ont été propres. Les chercheurs ont développé l'architecture BDI (*Belief, Desire, Itention*) (Croyance, Désire, Intention), cet agent est généralement représentées par les états suivants **[TAI, 12]** :

1. Désirs: les désirs de l'agent. Les désirs sont formalisés sous la forme d'un ensemble de critères qui seront utilisés pour évaluer les plans.

2. Croyances: les croyances de l'agent concernant le fonctionnement du système. Les croyances sont utilisées pour calculer les valeurs des désirs (critères).

3. Intention: Quand un agent acquiert de nouvelles informations (par ses propres moyens de perception ou par des messages envoyés par d'autres agents), il met automatiquement à jour sa base de croyances. Lorsque l'agent n'a pas de plan à exécuter (base d'intention vide), il commence par évaluer chacun des plans.

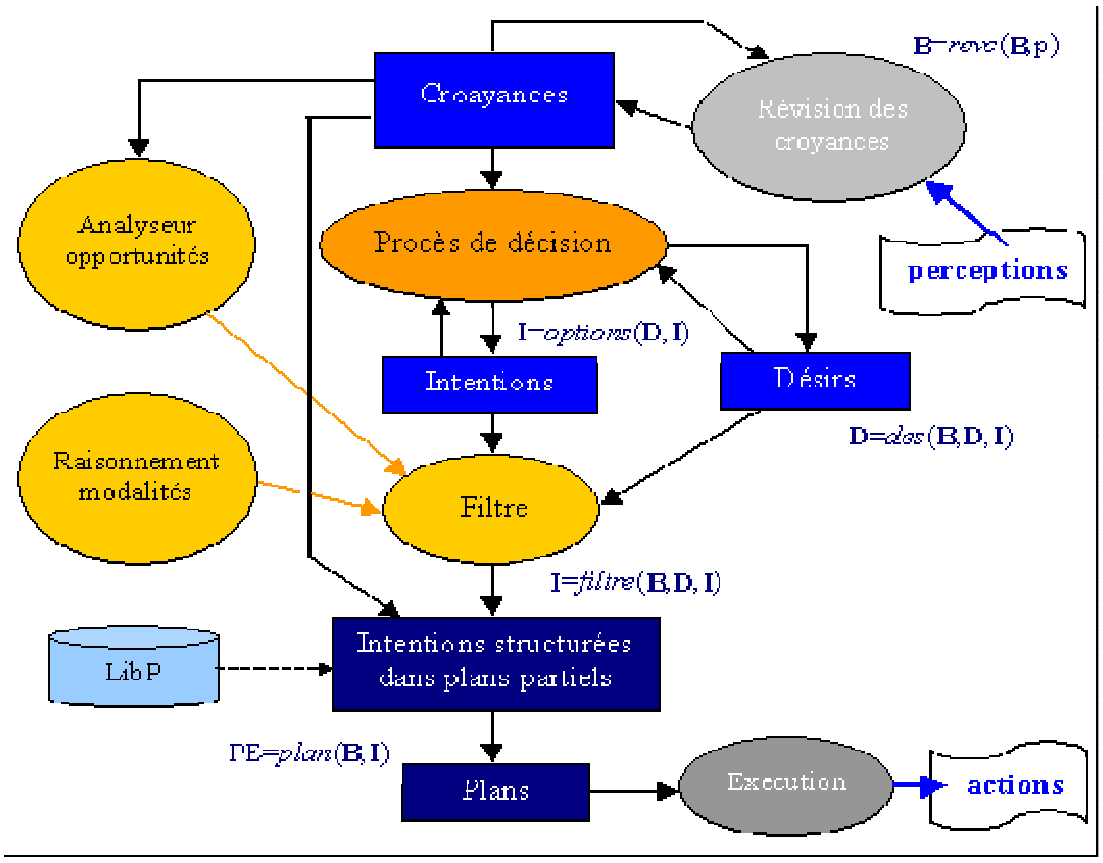


Figure I.4 : L'architecture BDI **[Web1]**

3.4. L'environnement

L'environnement est considérer comme une entité fondamentale dans les systèmes multi-agents **[GUE, XX]**. Un agent ne peut exister sans environnement. L'environnement est une structure dans laquelle l'agent évolue. Un agent va agir sur son environnement et l'environnement va agir sur l'agent. Amedzro a mentionné que la notion d'environnement à plusieurs dimensions **[AME, XX]**:

- **L'environnement Medium d'interaction :** Dans ce cas, les interactions entre les agents sont premiers (il faut donc les spécifier avant l'environnement) et l'environnement apparait comme le choix de conception de ces interactions.
- **L'environnement comme un stabilisateur** : Si on veut que la dynamique du système multi-agent se stabilise dans son comportement ou sa structure, il peut être nécessaire d'introduire des normes régulatrices qui doivent être issues d'un environnement.

S.Russel&P.Norving sont expliqués comment un agent intelligent communiqué avec son environnement **[WEY, 04]**: "Un agent est tout ce qui peut être considérer comme un apercevant de son environnement grâce à des capteurs et agissant sur son environnement " **[RUS, 95]**, la figure suivante présente la relation entre l'agent et l'environnement.

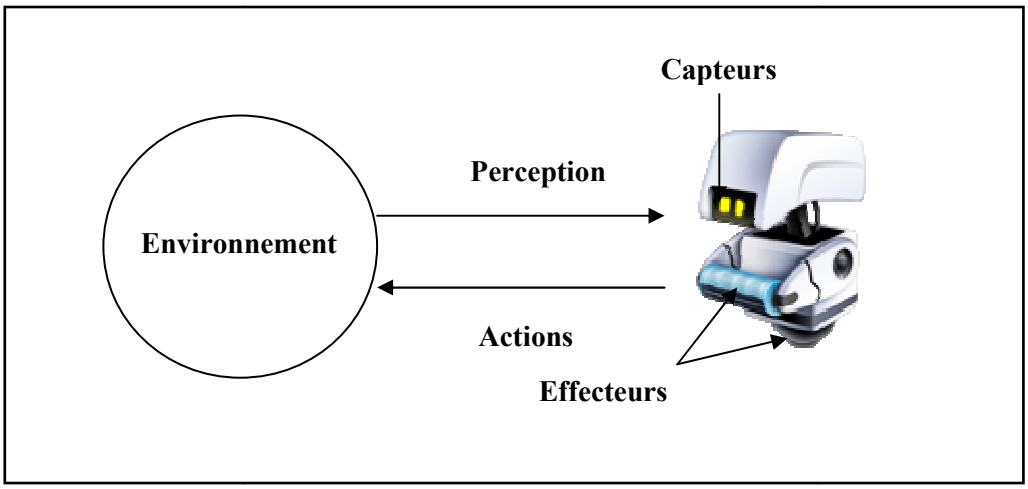


Figure I.5 : la relation entre l'agent son environnement

3.4.1. Propriétés

➢ **Accessible ou inaccessible :** si un agent peut, à l'aide des primitives de perception déterminer l'état de l'environnement et ainsi procéder, par exemple, à une action.

Un environnement accessible est un environnement dans lequel un agent peut obtenir une information complète, exacte et mise à jour de son état (l'environnement). La majorité des environnements modérément complexes (incluant, par exemple, le monde réel et Internet) sont inaccessibles.

➢ **Déterministe ou non-Déterministe :** selon que l'état futur de l'environnement ne soit, ou non, fixé que par son état courant et les actions de l'agent.

➢ **Statique ou Dynamique :** c'est-a- dire que l'environnement peut changer tant que un agent délabrer ou non.

➢ **Discrète ou continue :** c'est-a- dire que le nombre des perceptions et les actions sont limitées ou non.

3.5. Agent vs Objet

La différence essentielle qui existe entre les objets et les agents ce que l'objet est défini par l'ensemble des services qu'il offre (ses méthodes) et qu'il ne peut refuser d'exécuter si un autre objet le lui demande. Par contre les agents disposent d'objectifs qui leur donnent une autonomie de décision vis à vis des messages qu'ils reçoivent. D'autre part, ils établissent des interactions complexes qui font intervenir des communications de haut niveau **[FER, 97]**.

Par résultat, On peut déduire qu'un agent peut être considéré comme un objet ayant des capacités supplémentaires:

1. Recherches de satisfaction (intentions, pulsions).

2. communications à base de langages plus évolués (actes de langages pour les agents cognitifs, propagation de stimuli pour des agents réactifs).

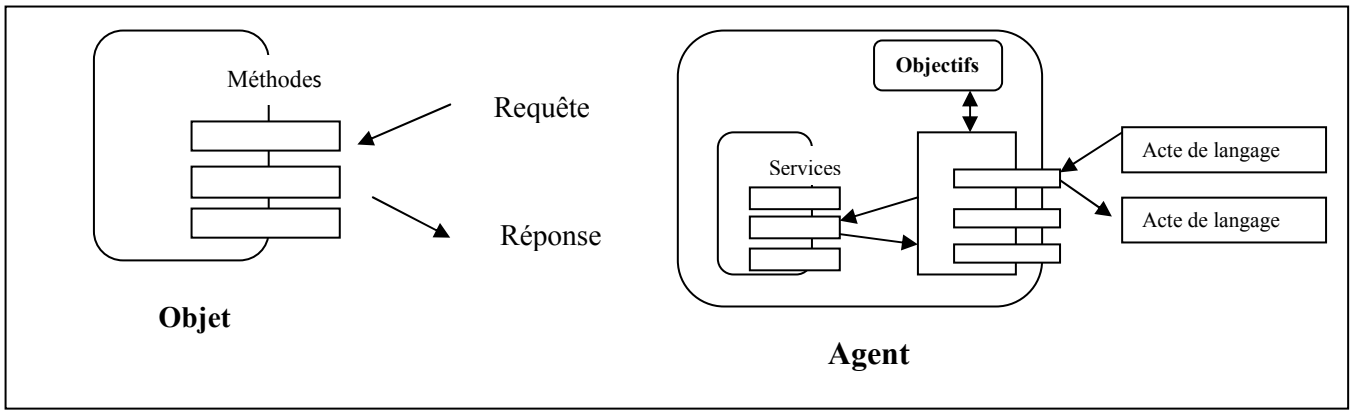


Figure I.6 : la différence entre l'objet et l'agent **[FER, 97]**.

4. Système multi- agents

Il est plutôt rare que les concepteurs d'agents n'aient besoin que d'un seul agent dans l'environnement qu'ils construisent. Lorsque plusieurs agents se retrouvent dans un même environnement et que ces agents ont besoin d'interagir entre eux, on parle alors de système multi-agents.

4.1. Définition : Un système multi agent est un ensemble organisé d'agent **[BRI, 01]**, qui travaillent selon les modes complexes d'interaction, pour réaliser leurs propres buts et par-là même atteindre l'objectif global désiré. Les agents peuvent interagir en communiquant directement entre eux ou par l'intermédiaire d'un autre agent ou en agissant sur leur environnement. Ferber est défini Le SMA est généralement caractérisé par :

1. Chaque agent a des informations ou des capacités de résolutions des problèmes limités, ainsi chaque agent a un point de vue partiel.

2. Il n'ya aucun contrôle global du système multi-agents.

3. Les données sont décentralisées.

4. Le calcul est asynchrone.

4.2. Les thèmes de recherche : Les thèmes de recherche de SMA sont particulièrement riches. Ferber **[FER, 97]** a cité certain domaines :

4.2.1. La résolution du problème

La résolution de problèmes au sens large concerne toutes les situations dans lesquelles les agents logiciels accomplissent des tâches utiles aux êtres humains. La résolution d'un problème se fait soit d'une manière centralisée ou décentralisée. On distingue trois types de résolution de problème :

- **La résolution distribuée de problèmes :** toutes les applications relevant de la résolution des problèmes supposent qu'ils sont possibles d'effectuer des tâches complexes en faisant appel à un ensemble de spécialistes ayant des compétences complémentaires. Donc la résolution d'un problème consiste à composer le problème en sous problème où chaque agent s'occupe d'un sous problème.
- **La résolution distribuée de problèmes distribués :** il s'agit essentiellement d'application telle que l'analyse, l'identification, le diagnostic et la commande de système physiquement réparti pour lesquelles il est difficile d'avoir une vision totalement centralisée.
- **Résolution par coordination :** les agents peuvent aussi servir d'une manière élémentaire à résoudre des problèmes au sens classique du terme où l'énoncé est bien posé et dont l'ensemble des informations est entièrement disponible par exemple trouve une affectation de tâche pour une machine-outil, définir un emploi de temps pour un collège. Dans le cas ou le domaine n'est pas distribué et l'expertise ne l'est pas non plus l'approche multi-agent peut apporter un mode de raisonnement nouveau en décomposant le problème de manière totalement différente.

4.2.2. Simulation multi agent

La simulation consiste à analyser les propriétés de modèle théorique du monde environnant. Le système multi-agent apporte une solution radicalement nouvelle au concept de simulation dans les sciences de l'environnement en offrant la possibilité de représenter directement les individus, leur comportement. La simulation multi-agent est fondée sur l'idée qu'il est possible de représenter sous forme informatique le comportement des entités qui agissent dans le monde et qu'il est ainsi possible de représenter un phénomène comme le fruit des interactions d'un ensemble d'agents disposant de leur propre autonomie opératoire.

4.3. L'interaction entre les agents

L'interaction entre les agents est une notion fondamentale dans le domaine de SMA, Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global. Cette participation permet au système d'évoluer vers un de ses objectifs et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication **[CHA, 96]**, de plus ils ont l'avantage de faire intervenir des schémas d'interaction sophistiqués. Les types courants d'interaction incluent **[CHA, 01]** :

- la coopération (travailler ensemble à la résolution d'un but commun).

➢ la coordination (organiser la résolution d'un problème de telle sorte que les interactions nuisibles soient évitées ou que les interactions bénéfiques soient exploitées).

➢ la négociation (parvenir à un accord acceptable pour toutes les parties concernées).

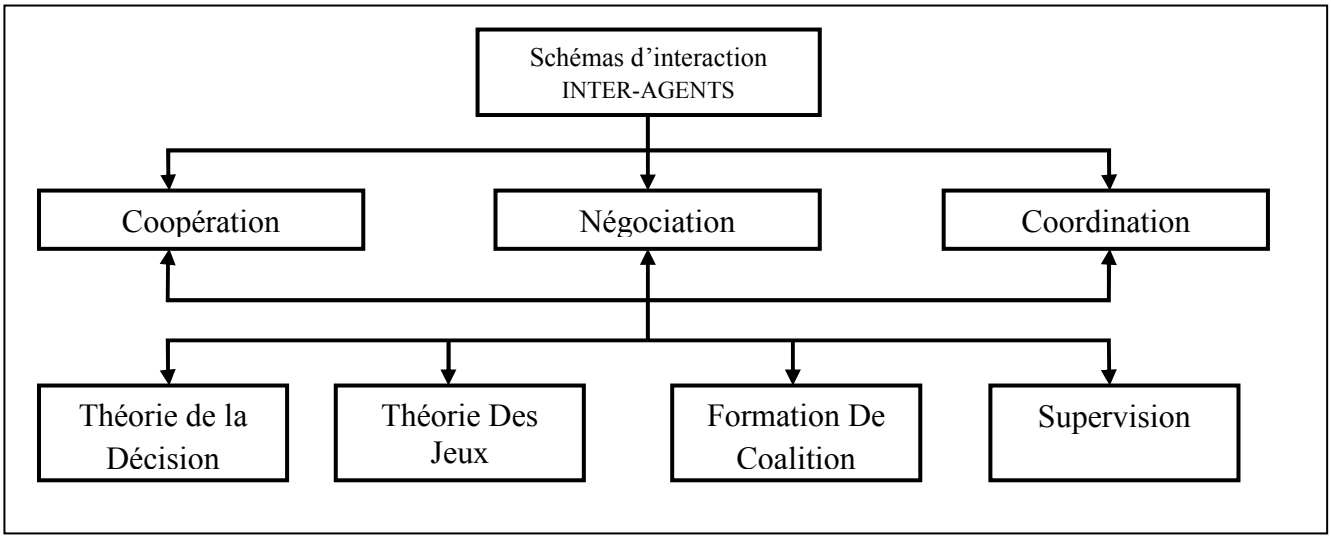


Figure I.7 : interaction inter-agents **[GLE, 05]**.

4.3.1. Coopération

La coopération est nécessaire quand un agent ne peut pas atteindre ses buts sans l'aide des autres agents, **J.Ferber** propose la définition suivante **[FER, 97]**: On dira que plusieurs agents coopèrent, ou encore qu'ils sont dans une situation de coopération, si l'une des deux conditions est vérifiée:

1. L'ajout d'un nouvel agent permet d'accroître différentiellement les performances du groupe.
2. L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

4.3.2. Négociation : La négociation joue un rôle fondamental dans les activités de coopération en permettant aux personnes de résoudre des conflits qui pourrait mettre en péril des comportements coopératifs, les chercheurs en IAD utilisent la négociation comme un mécanisme pour coordonner un groupe d'agents, La négociation est caractérisée par **[BER, 09]** :

- Un nombre faible d'agents impliqués dans le processus.
- Un protocole minimal d'actions : proposer, évaluer, modifier et accepter ou refuser une solution.

4.3.3. Coordination : Il y a essentiellement deux approches de coordination dans les SMA comportant des agents **[ALE, 06]** La première approche, appelée planification multi-agents centralisée, cette dernière suppose qu'il n'existe qu'un seul planificateur (confie à un agent spécialisé le soin de décider pour régler les conflits à partir de la vue globale qu'il possède des agents), c'est-a-dire qu'un seul agent capable de planifier et d'organiser les actions pour l'ensemble des agents **[FER, 95]**. La figure suivante présente le mécanisme de l'approche centralisé.

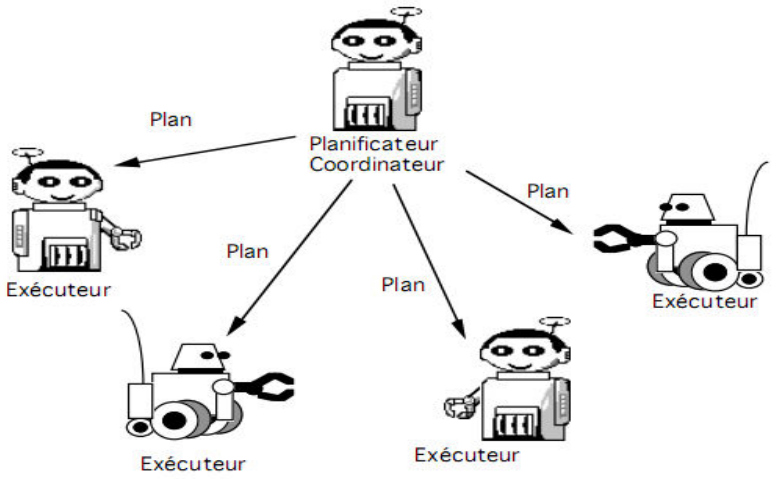


Figure I.8 : Approche planification multi-agents centralisée.

La seconde approche est appelé planification multi-agents distribuée. Aucun agent n'a de vue globale des activités du groupe. Dans ce cas, la détection des conflits est beaucoup plus difficile et il importe que chaque agent dispose des capacités de planification prenant en compte les plans des autres agents **[ALE, 06]**.

4.4. La communication

Les communications, dans les systèmes multi-agents comme chez les humains, sont à la base des interactions et de l'organisation sociale. Sans communication, l'agent n'est qu'un individu isolé, sourd et muet aux autres agents **[FER, 95]**.et cela signifier que la communication est un moyen ou une méthode de coopération qui existe en plusieurs façons **[HOU et al, 14]**:

- **Communication par environnement :** Les agents laissent des traces de leurs présences qui peuvent être perçus par d autres agents. La communication par environnement est très courte dans les systèmes réactifs. On la qualifie de non intentionnel et les deux autres modes sont qualifiés de communication intentionnelle.
- **Communication par l'envoi des messages (mode directe) :** ou les agents communiquent entre eux par l'envoi et la réception des messages.

➢ **Mode point à point :** l'agent émetteur du message connaît et précise l'adresse de ou des agent(s) destinataire(s). Ce type de communication est généralement le plus employé par les agents cognitif.

➢ **Mode par diffusion :** le message est envoyé à tous les agents du système. Ce type de transmission est très utilisé dans les systèmes dynamiques ainsi que les systèmes d'agent réactif.

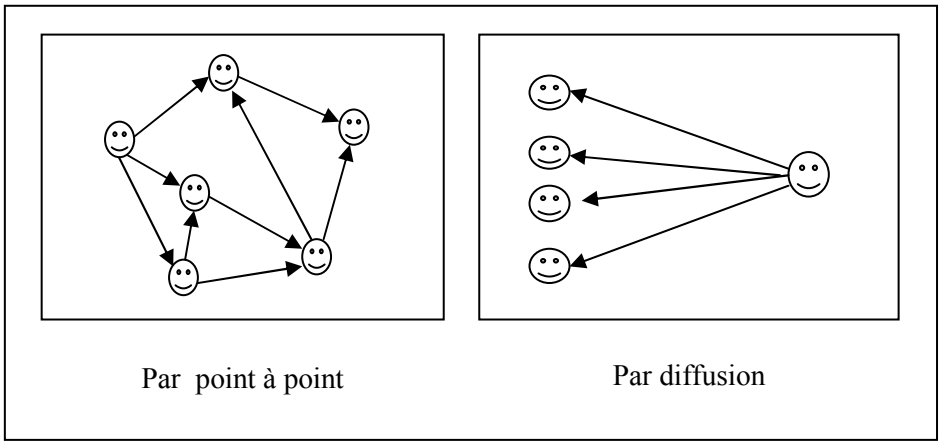


Figure I.9 : la communication par l'envoi des messages.

- **Communication par partage des informations (tableau noir) (mode indirecte) :** cette communication est basée sur l'utilisation d'une mémoire partagé.

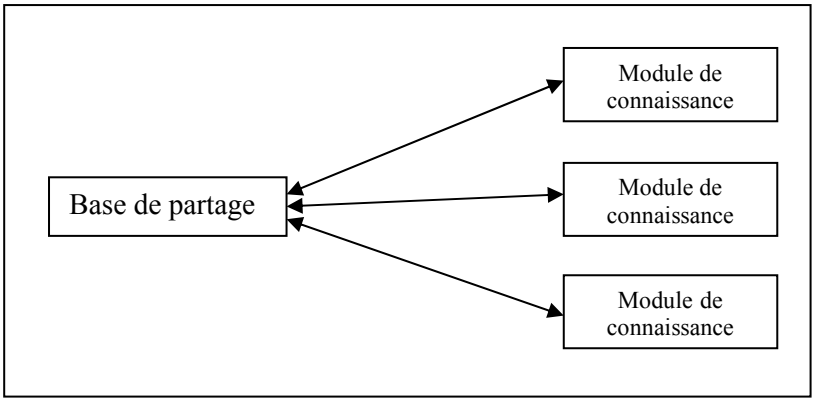


Figure I.10 : la communication par partage des informations.

4.4.1. Les langages de communication

Grâce à la coordination un système multi-agents peut réaliser ses tâches avec plus d'efficacité qu'un seul agent. Mais pour coordonner l'activité d'un ensemble hétérogène d'agents autonomes, il faut que les agents communiquent dans un langage compréhensible par tous les autres. On observe que dans un système ouvert un tel langage peut constituer une interface entre les agents.

- **KQML(Knowledge Query and Manipulation Language) :** pendant les années 1990 , une agence américaine pour la recherche militaire a développé KQML sous projet Knowledge Sharing Effort. Les travaux étaient dirigés par (Tim Finin) et (Jay Weber) et ses spécifications furent fournies en 1996. Ce dernier est fondé sur la théorie des actes de langage dans le but de permettre aux agents cognitifs de coopérer. **[web1] [FIN et al, 94]KQML** ou le **Knowledge Query and Manipulation Language** est un langage de haut niveau de communication entre agents. Ce langage est indépendant

de la syntaxe et de l'ontologie des messages, du mécanisme de transport et du langage de codage des messages. La figure suivante illustre les trois couches qui consistent un message KQML : contenu, communication et message.

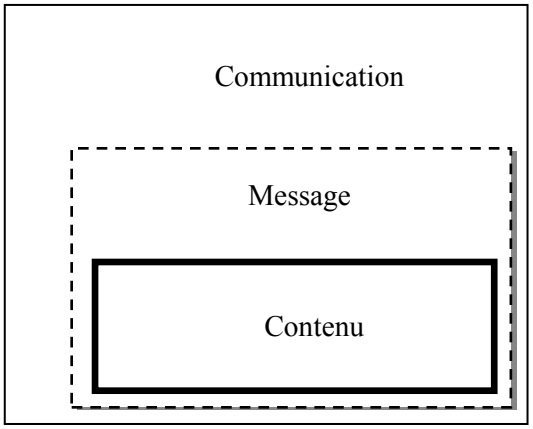


Figure I.11 : Les trois couches KQML.

Un message KQML est ainsi composé d'une expression encapsulée dans une enveloppe de message, elle-même encapsulée dans elle-même encapsulée dans une enveloppe de communication. La couche de contenu est généralement constituée d'une expression dans un langage de représentation de connaissances **[YAN, 03].**

- **FIPA-ACL (**Foundation For Intelligent Agents) : La fondation pour les agents intelligents physiques est une organisation destinée à établir des standards afin de favoriser l'interopérabilité des applications, des services et des équipements informatiques. L'organisation fut dissoute en 2005 et remplacée par un comité des standards de l'IEEE **[web 2]**. Le plus utilisé des standards de la FIPA est un standard de langage de communication ACL (Agent Communication Language). Tout comme KQML qui base sur la théorie des actes de langage : les messages sont vus comme des actes communicatifs car ils sont prévus pour effectuer une certaine action en vertu de l'envoi. Les spécifications de FIPA-ACL se composent d'un ensemble de types de message et de leur sémantique (les effets de chaque message sur les états mentaux des interlocuteurs) **[ALE, 06]**. Généralement, FIPA-ACL distingue deux couches qui contiennent le message, À la couche interne, le contenu des messages peut être exprimé dans une langue quelconque logique. La couche externe décrit les locutions que les agents peuvent utiliser dans leur communication. Le contenu des messages est enveloppé dans ces locutions. FIPA-ACL possède 22 actes communicatifs. On explique certains d'entre eux **[Web 3]**:

➢ **Confirmation:** L'expéditeur informe le récepteur qu'une proposition est vraie, où le récepteur est connu pour être certain de la proposition.

➢ **Informer**: ou L'expéditeur informe le récepteur que la proposition est vraie.

➢ **Request**: L'expéditeur demande le récepteur pour effectuer une action ou d'un acte de communication.

4.5. Les plateformes de développement

Plusieurs plates formes existent permettant de développer et d exécuter des systèmes multi-agents conformément aux normes précédemment évoquées. Les plates formes offrent des classes d'abstractions pour les agents, ainsi que des classes de communication permettant des interactions entre agents tout en respectant les standards (Fipa-ACL, KQML, etc.). Voici quelques unes des plates formes multi-agents les plus connues **[HOU et al, 14]:**

4.5.1. JADE (Java Agent Developement Framework) : est une plate-forme multi-agent développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie) qui a comme but la construction des systèmes multi-agent et la réalisation d'applications conformes à la norme FIPA. JADE comprend deux composantes de base : une plate-forme agents compatible FIPA et un paquet logiciel pour le développement des agents Java **[Web3] [FAB et al, 07]**.

4.5.2. ZEUS: est une plate-forme multi-agent conçue et réalisée par British Telecom (Agent Research Programme of BT Intelligent Research Laboratory) pour développer des applications collaboratives **[NAD, XX]**. Cette Plateforme, développé en Java, génère automatiquement le code Java des agents spécifiés graphiquement. Les caractéristiques des domaines d'applications de ZEUS ont été définies par les concepteurs. Parmi ces caractéristiques, on peut mentionner **[Web3]** :

- chaque agent crée un plan qui nécessite un raisonnement explicite pour atteindre son but.
- la résolution de problèmes nécessite une coopération entre agents.
- le rôle de chaque agent consiste à contrôler un système externe qui réalise une tâche du domaine d'application, la résolution de problème est ainsi effectuée par ce système externe et contrôlée par les agents.

Un agent dans Zeus est constitué en trois couches : la couche de définition, qui contient les capacités de raisonnement et des algorithmes d'apprentissage, la couche organisationnelle, qui contient la base de connaissances des accointances de l'agent, et la couche de coordination, qui définit les interactions avec les autres agents **[YAN, 03]**.

4.5.3. MadKit : est une plate-forme développée par le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. MADKIT est libre pour l'utilisation dans l'éducation. Ce dernier est écrit en Java et est fondé sur le modèle organisationnel ALAADIN **[Web3]**. MadKit est basé sur trois principes **[FER, 00]**:

- Architecture à micro-noyau.
- Agentification systématique des services.

- Découplage applicatif entre noyau, agents et application d'accueil.

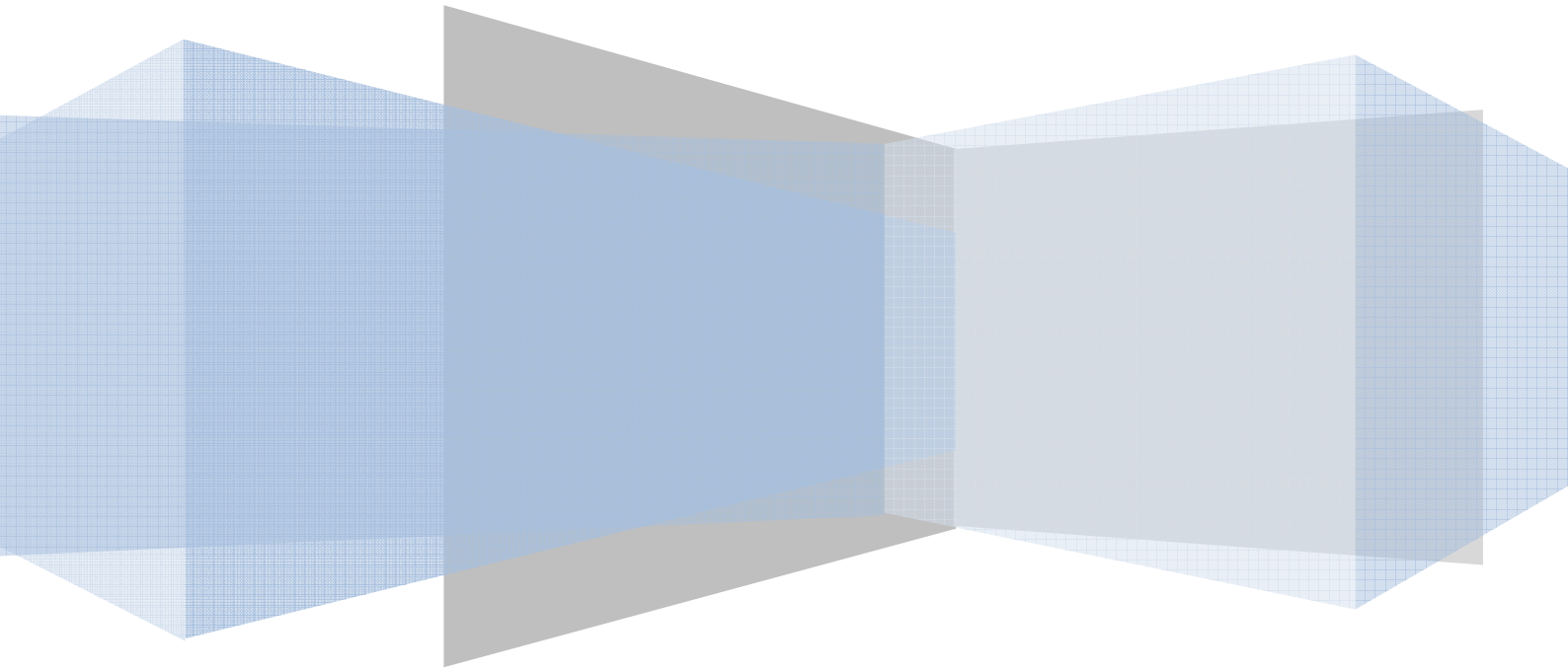
5. Conclusion

Dans ce chapitre, on a fait un tour d'horizon sur le domaine d'Intelligence Artificielle Distribuée passant sur l'objectif, la définition et les branches. Puis on a présenté les principales notions de systèmes multi-agents, ce dernier est considéré comme l'un des branches fondamentales de l'IAD qui sont largement répondus et développées durant de quarante années passées, ce branche a pour but de représenter des problèmes possédant de multiples méthode de résolution et perspectifs et touche à des domaines différents comme la science cognitive, la philosophie, la biologie, …etc. Alors de notre coté on a intéressé a bien détaillé les concepts d'agents et les systèmes multi-agents. Raisons pour lesquelles on les a choisis comme méthode permettant de mieux représenter le problème d'emploi de temps de façon décentralisée afin d'obtenir une solution améliorée.

ème

2 Chapitre

Problème d'emploi du temps universitaire



1. Introduction

La génération d'une application qui résoudre le problème d'emploi du temps est parmi les problèmes très complexes, il est classifié dans les cas généraux comme (NP-hard) problèmes, en autre terme, les fonctions nécessaires pour résoudre ce problème doivent augmenter d'une façon exponentielle, ce qui rend difficile a résoudre. Depuis prés de quarante ans, ce type de problème a attiré l'attention de la communauté scientifique de plusieurs disciplines, dont la recherche opérationnelle et l'IA. Alors que plusieurs approches et modèles ont été proposés comme algorithme génétique **[SAM et al, XX]**, Système expert, SMA **[MOH et al, 11]** … etc.

2. Le problème de satisfaction des contraintes

De nombreux problèmes peuvent être exprimés en termes de contraintes comme le problème d'emploi du temps, gestion d'agenda, de gestion de trafic ainsi que certain problème de planification et d'optimisation comme le problème de routage de réseaux de télécommunication.

2.1. Définition d'un CSP

Un CSP est un problème modélise par des variables, un domaine, et des contraintes. Les contraintes sont posées sur des variables, chacune des variables prend des valeurs dans un domaine **[FER et al, 05].**

Formellement la définition d'un problème par contrainte se fera de façon suivante :

Soit un triplet (X, D, C) tel que :

1. $X= \{X_1, X_2, \ldots, X_n\}$ est l'ensemble des variables du problème.
2. D est le domaine de définition de chacun des variables, soit l'ensemble des valeurs que peut prendre X_i.
3. $C= \{C_1, C_2, \ldots, C_n\}$ est l'ensemble contraintes variables de X, qui limite les valeurs que peuvent prendre simultanément ces variables.

2.2. Méthode de résolution de CSP

Plusieurs approches sont utilisées pour résoudre les CSP, on distingue les méthodes exactes (ou complètes) et les méthodes approchées (ou incomplètes). Les méthodes exactes font une recherche arborescente en instanciant les variables une par une et effectuent des réparations en cas d'échec. Les méthodes approchées font une réparation d'une configuration en parcourant d'une manière aléatoire l'espace de recherche**[TRO, 06]**.

2.3. CSP et PET

Le problème d'emploi du temps est un problème défini en termes de contraintes (de temps, d'espaces ou plus généralement de ressources comme d'ailleurs d'autres problèmes tels que :

_ Les problèmes de planification et d'ordonnancement : planifier une production, gérer un trafic ferroviaire.

_ les problèmes d'affectation du personnel à des tâches, des entrepôts à des marchandises,… etc.

Ces différents problèmes fortement contraint ayant la particularité commune d'être caractérisés par une très forte combinatoire, peuvent être considérés comme des problèmes de satisfaction de contraintes (CSP : Constraint Satisfaction Problems) car ces problèmes se formulent aisément en CSP quand ils ne nécessitent que des contraintes fortes**[TRO, 06]**.

3. Présentation du problème d'emploi du temps

3.1. Définition

L'emploi du temps est un plan représentatif définissant l'affectation d'un ensemble de taches et activités à un ensemble de ressources en se basant sur des périodes de temps bien précises, soumettant à un ensemble de contraintes à respecter **[HOU et al, 14]**.

3.2. Problématique de planification horaire (emploi du temps)

Comme on l'indique, la planification horaire et un processus très complexes qui vise a organisé des activités humaines dans un intervalle du temps (généralement une semaine). La planification horaire est considérée comme un outil à des programmes permettant d'organiser et d'ordonnancer le travail des ressources humaines et affecter pour chaque période du temps sur un horizon donné, de telle manière que les besoins par intervalle soient couverts et que les différentes contraintes soient satisfaites**[TRO, 06]**.

Dans le monde pratique, il existe un grand nombre de variantes du problème d'emploi du temps qui diffère les uns des autres selon le type d'établissement impliqué (usine, école, société, … etc.) et selon le type de contraintes. Dans ce cas on va concentrer sur PET d'enseignement et puis on va cerner PET universitaire.

3.2.1. Les caractéristiques de PET

Le PET consiste à définir un ensemble des ressources importants pour la réalisation un nombre de tache dans une unité du temps, ces ressources est subdivisé en deux types : Le premier type c'est les moyens techniques ou physiques comme (Salles, matérielles, climatiseurs, ordinateurs…etc.), Le deuxième type c'est les moyens humaines (ingénieurs, enseignants, …etc.). L'affectation de ces ressources est faite en respectant des certains contraintes et préférences. Ces derniers (contraintes) peuvent différer d'un problème à un autre suivant la spécificité ainsi que les caractéristiques attendues de l'emploi du temps. Aussi ces contraintes sont souvent classées en deux catégories, la première regroupe les contraintes dures dans lesquelles l'emploi du temps est infaisable ou inacceptable

lorsque ne satisfait pas ce genre des contraintes, La seconde catégorie regroupe des contraintes appelées souvent des contraintes préférences dont la satisfaction a différents degrés d'importance mais dont le non respect n'empêche pas une application plus au moins acceptable de l'emploi du temps trouvé **[HOU et al, 14]**.

3.2.2. Le PET universitaire

Le problème d'emploi du temps universitaire est une instance des problèmes d'ordonnancement cyclique les plus connues dans la littérature, il s'agit d'ordonnancer les tâches (qui possède un caractère cyclique) d'un ensemble d'enseignants en leur allouant un ensemble de salles et en leur fixant leurs dates de début et de fin **[HOU et al, 14]**.

Dans la vie quotidienne, les universités ont besoin un calendrier de la planification, mais l'élaboration d'un emploi du temps universitaire satisfaisant est considéré comme une taches très difficiles que l'emploi du temps scolaire, car le système des salles est changé périodiquement (Le groupe des étudiants n'a pas eu une salle spécifié) et cela augmente la complexité du ce problème**[]**.et avec le progrès réalisé dans les technologies matérielles et logicielles, la communauté scientifique continue à travailler sur le problème.

Burke et ses collègues **[BUR et al, 02]** notent à cet égard que ce type de problème se divise en deux catégories principales : les cours et les examens. Différents aspects séparent ces deux catégories **[MOY et al, XX]**. Par exemple :

- **Emploi du temps des cours :** établir le programme hebdomadaire pour tous les cours d'un ensemble d'université réduisant au minimum que possible les chevauchements des cours ayant les groupes des étudiants communs.
- **Emploi du temps des examens :** L'emploi du temps d'un examen est un peu différent par rapport l'emploi du temps des cours car il impose certains contraintes qui ne sont pas prise en compte dans celui des cours. L'emploi du temps des examens possèdent des caractéristiques particulières suivantes :

a) Il ya seulement un examen pour chaque module.

b) Le nombre des périodes peut changer contrairement au cas du cours.

c) plusieurs examens peuvent se tenir en même temps dans la même salle ou un même examen peut être dispatché dans plusieurs salles.

3.2.3. Le PET des cours universitaires

Dans cette problématique, on a choisi de traiter le PET consacrés au cours. Ce genre de problème est défini comme un ensemble des cours qui se déroule dans des périodes spécifiés durant la semaine, ces cours à besoin un nombre des salles et des professeurs suffisant pour bien contenir le nombre important des groupes des étudiants.

L'établissement d'un emploi du temps concernant les cours des universités consiste a pris en compte des contraintes qui doivent respecter, ces contraintes sont subdivisé on deux catégories :

- **Les contraintes dures (hard) [HOU et al, 14]:** Se sont des contraintes qui doivent être satisfaite car la violation d'un de ces contraintes peut générer un emploi du temps inacceptable.

_ Un enseignant peut enseigner un module dans une salle pendant une séance.

_ Un groupe occupe une salle pendant une séance.

_ Chaque enseignant ne peut enseigner que le module de ces compétences.

_ Le module doit avoir un nombre limite de séance (TD, cours, TP).

- **Les contraintes préférence (Soft) [HOU et al, 14]:** La violation de ces contraintes n'a aucun effet sur la génération d'une solution satisfiable

_ Il faut que l'affectation des salles et des périodes de temps permette de satisfaire au mieux les préférences des enseignants.

_ Il faut que l'affectation des salles aux différentes séances permette de satisfaire aux mieux certaines préférences.

3.3. Génération automatique d'un emploi du temps

La génération automatique d'un emploi du temps est une activité de création, de gestion et de maintenance d'un emploi du temps avec un ordinateur et avec l'intervention minimale de l'être humain en satisfaisant au mieux les ressources humaines, matérielles et temporelles .

La génération automatique d'un emploi du temps nécessite la construction d'un système capable de gérer les ressources temporelles et matérielles, les contraintes imposées et de résoudre les conflits entre ces ressources. Mettre au point un tel système est une tache difficile et factieuse pour l'être humain. Outre la réalisation manuelle d'un emploi du temps est très compliquée, très couteuse **[ABB et al, 06].**

3.3.1. Prétraitement

Formellement on peut présenter le problème d'emploi du temps comme suit :

- Un ensemble de cours $E_c = \{C_1 \dots C_m\}$, Les C_i ayant pour valeur les numéros des cours suivis par un groupe des étudiants.
- Un ensemble de salles $E_s = \{S_1 \dots S_n\}$, Les S_i ayant pour valeur les numéros des Salles, Il y a plusieurs types de salle, salle de cours, salle de TD, salle de TP, salle de sport.
- Un ensemble d'enseignants $E_e = \{E_1 \dots E_k\}$.
- Un ensemble de groupes $E_g = \{G_1 \dots G_l\}$. Chaque groupe regroupe un ensemble d'étudiants qui étudient au même niveau dans une même filière.
- Un ensemble de périodes $E_p = \{P_1 \dots P_t\}$ pendant lesquelles différentes leçons peuvent être placées.

3.3.2. Formulation mathématique des contraintes

Une séance est acceptable s'il contient un professeur, un groupe et une salle, le problème se trouve dans le cas de conflit c'est pour cela le programme doivent respecter certains contraintes. On motionne ci-dessous les plus fréquentes :

- **Contrainte 1 :** un enseignant E_i ne peut donner au plus un cours pendant une période P_k :

$$\forall \boldsymbol{Ei} \in [\mathbf{1}..\boldsymbol{Ne}], \forall \boldsymbol{Pk} \in [\mathbf{1}..\boldsymbol{Np}] \sum_{\boldsymbol{Sj}\in[\mathbf{1}..\boldsymbol{Ns}]} \left(\sum_{\boldsymbol{Cl}\in[\mathbf{1}..\boldsymbol{Nc}]} \boldsymbol{MAT\ Ei}[\boldsymbol{Sj}][\boldsymbol{Pk}][\boldsymbol{Cl}] \right) \leq \mathbf{1}$$

- **Contrainte 2 :** une salle ne peut contenir au plus un cours C_l donnée par un unique enseignant E_i pendant une période P_k :

$$\forall \boldsymbol{Sj} \in [\mathbf{1}..\boldsymbol{Ns}] \sum_{\boldsymbol{Ei}\in[\mathbf{1}..\boldsymbol{Ne}]} \left(\sum_{\boldsymbol{Pk}\in[\mathbf{1}..\boldsymbol{Np}]} \left(\sum_{\boldsymbol{Cl}\in[\mathbf{1}..\boldsymbol{Nc}]} \boldsymbol{Mat\ Sj}[\boldsymbol{Ei}][\boldsymbol{Pk}][\boldsymbol{Cl}] \right) \right) \leq \mathbf{1}$$

- **Contrainte 3 :** un module doit avoir un nombre limite de séance (TD, cours, TP).

$$\forall \boldsymbol{Cl} \in [\mathbf{1}..\boldsymbol{Nc}], \sum_{\boldsymbol{Pk}\in[\mathbf{1}..\boldsymbol{Np}]} \left(\sum_{\boldsymbol{Ei}\in[\mathbf{1}..\boldsymbol{Ne}]} \left(\sum_{\boldsymbol{Sj}\in[\mathbf{1}..\boldsymbol{Ns}]} \boldsymbol{Mat\ Cl}[\boldsymbol{Pk}][\boldsymbol{Ei}][\boldsymbol{Sj}] \right) \right) \leq \boldsymbol{Nlimite}$$

3.4. Les approches de résolution

Malgré les difficultés d'implémentation, plusieurs approches proposées et étudiées par de nombreux chercheurs, ce qui offre des solutions réalisables mais qui ne sont pas toujours efficaces. On regroupe ces méthodes en deux classes : la première classe contenant les méthodes centralisées et la seconde classe contenant les méthodes centralisées.

3.4.1. Approches centralisées

Plusieurs approches centralisées ont été proposées pour la résolution de ces problèmes. Les premières tentatives de résolution étaient, les méthodes basées sur la théorie des graphes, la programmation linéaire et les techniques de satisfaction des contraintes. Mais ces méthodes n'ont pas donné de solutions traitant toutes les instances et les contraintes de ce problème **[HOU et al, 14]**.

C'est pour cela, ils ont cédé le pas à d'autres types de méthodes adaptées à ces types de problème, à savoir les méta-heuristiques telles que la recherche tabou, le recuit simulé, les colonies de fourmis, le système expert et l'algorithme génétique.

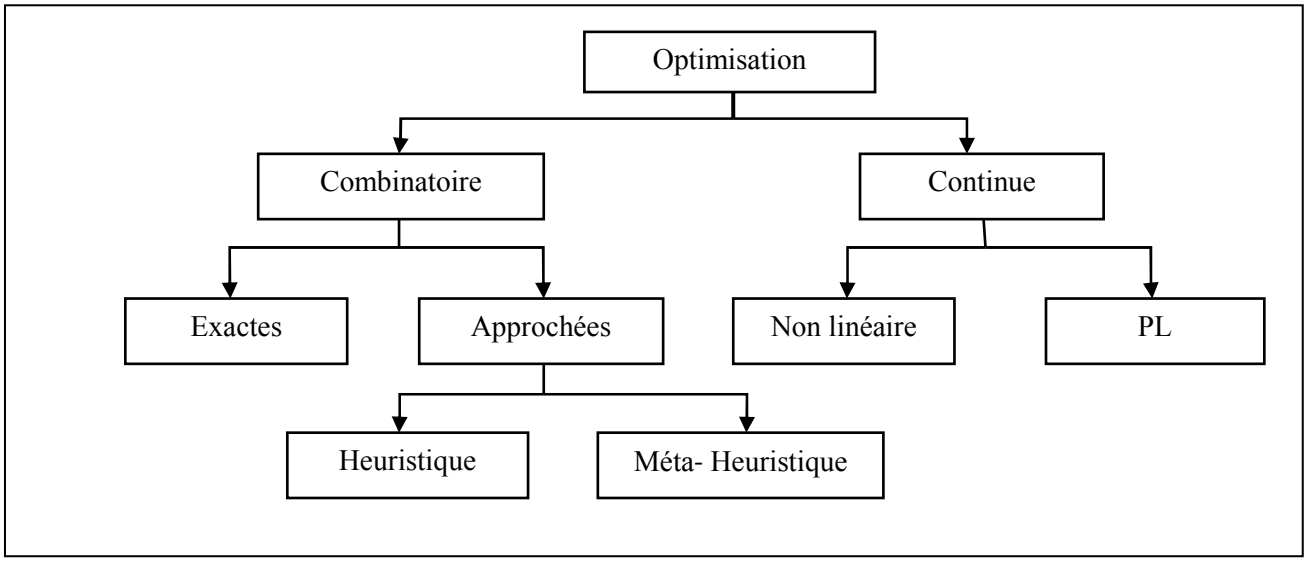


Figure II.1 : Classification des méthodes de classification.

3.4.1.1. Programmation linéaire

Un programme linéaire (PL) est un problème d'optimisation qui consiste à maximiser ou minimiser une fonction-objectif linéaire soumise à un ensemble des contraintes exprimées sous la forme d'équations ou d'inéquations linéaires. Ces contraintes définissent le polyèdre des solutions admissibles. La principale caractéristique d'un programme linéaire réside dans sa facilité de résolution grâce au célèbre algorithme de résolution du Simplexe, qui consiste à se déplacer d'un sommet du polyèdre des solutions admissibles à un sommet adjacent de coût inférieur (ou égal) **[OUM, 09]**.

Daskalaki et ses collègues **[DAS et al, 04]** ont utilisé la P.L.N.E pour présenter une formulation du PET universitaire. Ils ont ajouté plusieurs fonctionnalités telles que les variables multidimensionnelles (intégrant plus de détails du problème dans le modèle) et une fonction de cout (permettant l'introduction de certaines préférences concernant les jours, les salles et les périodes de temps) donnant plus flexibilité au système **[HOU et al, 14]**.

Dans le problème d'emploi du temps, on peut le modéliser avec la méthode de PL comme suit :

$$Min\ Z = \sum_{i=1}^{p} \sum_{j=1}^{c} \sum_{k=1}^{h} \text{Cijk Xijk , avec} \sum_i Xijk \leq 1, \sum_j Xijk \leq 1, \sum_k ijk \leq 1$$

$$Xijk = \begin{cases} 1 & \text{Si l'enseignant i enseigne à la salle j dans période k} \\ 0 & \text{Sinon} \end{cases}$$

E, C, P représentent respectivement les nombres d'enseignants, de classes et de périodes ; C_{ijk} : étendre le cout d'affectation faible on veut affecter l'enseignant i à classe j à séance k et très grand sinon.

L'inconvénient majeur de cette méthode réside au niveau de la taille des données manipulées et des contraintes car celle-ci est grande, cependant les données générées coutent chères en terme d mémoire à occuper et sont très difficiles à manipuler.

3.4.1.2. Recherche de Tabou

La méthode Tabou a été développée par Glover **[GLO, 89]**. Pour certains chercheurs cette méthode apparait plus satisfaisante sur le plan scientifique que le recuit simulé, car la partie « aléatoire » de la méthode a disparu **[TRO, 06]**. La méthode Tabou examine un échantillonnage de solution N(s) et retient la meilleure s' même si s'est plus mauvaise que s. L'algorithme donc ne s'arrête pas au premier optimum trouvé.

La méthode Tabou mémorisé les k dernières configurations visitées dans une mémoire à coure terme, cette dernière est appelée la liste Tabou (qui a donne le nom de la méthode), elle permet d'éviter toutes les cycles de longueur inférieure ou égale à k. L'algorithme de la recherche taboue est comme suite **[MUS, XX]**:

Recherche du Tabou {

Initialisation des paramètres ;

Prend une solution initiale (S_0) ;

Convergé = faux ;

Tant que non-convergé {

Prend un ensemble de solutions S dans le voisin de S_0 ($S \in N(S_0)$) ;

Déplacé=faux ;

Tant que non- Déplacé {

Si S pas Tabou {

Calculer la fonction d'objectif F, trouver $\sigma = f(S) - f(S_0)$;

Poussez S dans la liste Tabou ;

Déplacé= vrai ;

Si ($\sigma < 0$) Accepter solution ($S = S_0$) ;}

Suivant $S \in N(S_0)$; }

Convergé= test de convergence ; }

Retourné S_0 comme la meilleure solution ;

}

Figure II.2 : L'algorithme de recherche Tabou.

La résolution de problème d'emploi du temps à l'aide de méthode recherche passe par l'algorithme ci-dessus.

3.4.1.3. Système expert

Le système expert est développé par un groupe de recherche d'université Standford en 1965, L'idée fut d'introduire la connaissance des experts dans les ordinateurs en les rendant intelligents. Le concept de système expert inclut de plus l'idée selon laquelle cette connaissance des experts n'est pas noyée dans un algorithme, elle est explicite. L'expertise se traduit bien souvent par un ensemble de règles. Déductives qui reflètent par leur enchaînement le raisonnement des experts eux-mêmes. Le programme va lire ces règles et établir un raisonnement en tentant de les appliquer, ce que nous appellerons les chaîner **[Web4]**.

Le PET peut être adapté à l'approche proposée ci-dessus par Solotrevsky, compris le PET du cours. On considère les cours comme activités et les périodes comme ressources à assigner aux activités. Les coups secs et durs possèdent cinq types de règles: les règles d'affectation, les règles de contrainte, les règles locales de changement, les règles de contexte et les règles prioritaires.

- **Les règles d'affectation :** assignent des cours aux périodes une par une. Elles sont le noyau du système. Ces règles sont fournies par l'utilisateur mais l'heuristique utilisée n'est pas prédéfinie.
- **Les règles de contrainte :** Ces règles indiquent les contraintes que la solution doit satisfaire. Elles sont contrôlées chaque fois qu'un nouveau cours est assigné à une période. Les règles de contraintes sont regroupées en deux : les règles positives (règles dont les contraintes devant être satisfaites) et négatives (règles pour les contraintes pouvant être violées). Ces règles identifient des conflits dès qu'elles surgissent.
- **Les règles locales de changement :** le but de ces règles est de défaire une affectation précédente afin de créer l'occasion d'assigner l'affection en cours.
- **Les règles de contexte :** choisissent le contexte actif. Le système tient compte des contextes multiples : dans divers contextes les cours et périodes peuvent avoir la priorité différente. La priorité détermine quel objet parmi les autres est traité d'abord.
- **Les règles prioritaires :** déterminent les priorités des cours et des périodes dans chaque contexte. Les priorités sont calculées pour chaque nouveau contexte.

3.4.1.4. Algorithme génétique

Le AG sont des branches des Algorithme évolutionnaires la plus connu et la plus utilisée. La particularité des ces algorithme est qu'ils font évoluer des populations d'individus codés par des chaines de langueur fixe qui passe par le processus suivant :

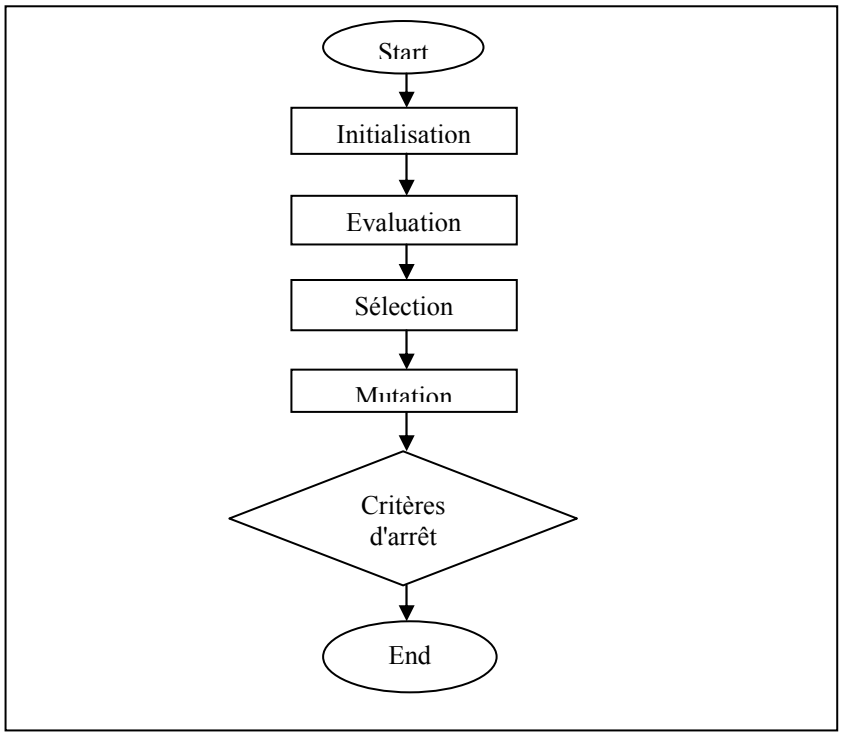


Figure II.3 : L'algorithme AG.

Les AG ont la capacité de produire des solutions de bonne qualité même si les dimensions du problème augmentent. Ils sont été appliqués à une large variété de problèmes comme le PET ou iles ont donné des résultats acceptables.

3.4.2. Approche décentralisé

Cette Approche est associé au système multi-agents ou Le problème est distribué sur un ensemble d'agents qui sont responsables soit d'un ensemble de variables, soit d'un ensemble de contraintes, et qui communiquent entre eux. Cette approche est basée sur problème de satisfaction distribué.

Yokoo et son collègue **[YOK, 00]** ont été définir DisCSP comme un réseau des contraintes distribuées de quintuplet (X, D, C, A, Ψ) ou :

_X, D et C sont définis comme précédemment dans un CSP.

_A= (A_1, A_2, …, A_n) est un ensemble de n agents.

_ Ψ : X →A est la fonction qui associe chaque variable de X à un agent de A.

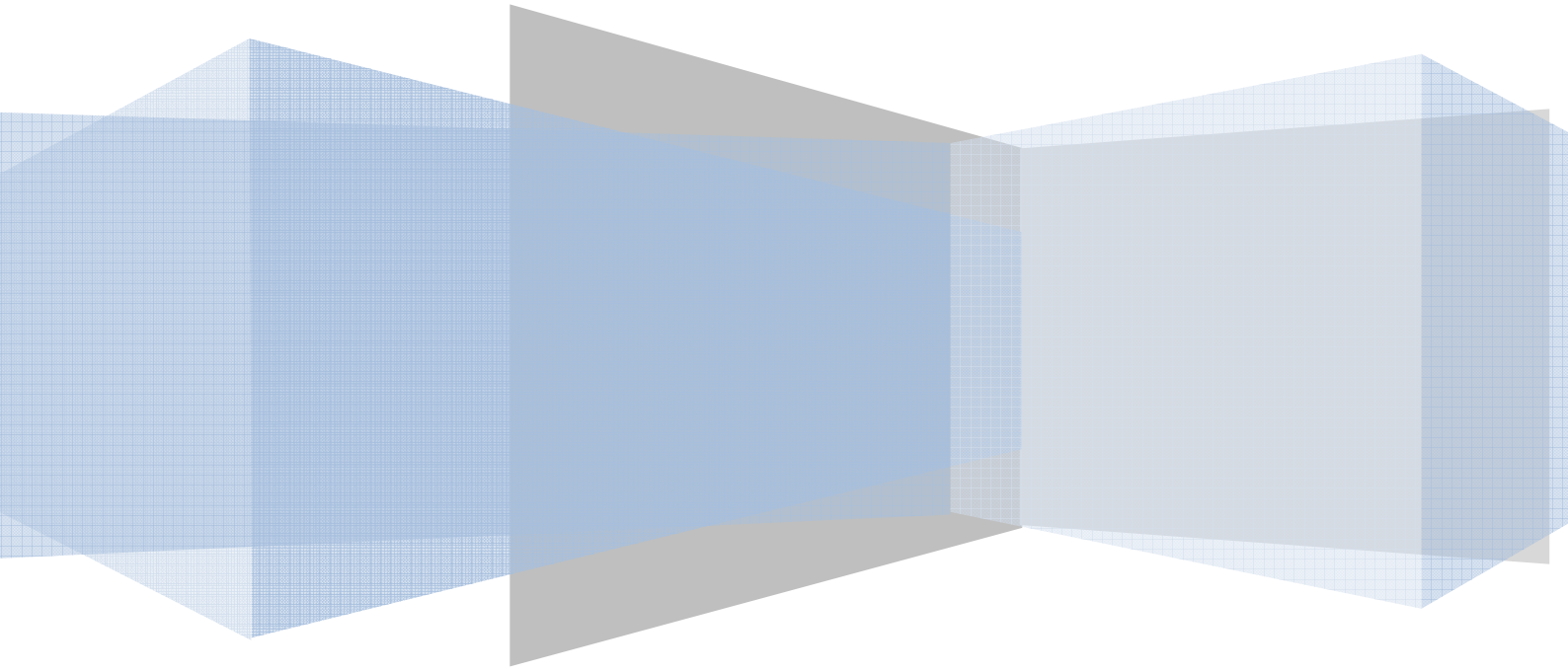
4. Conclusion

Dans la deuxième chapitre, on a met les points sur le problème d'emploi du temps puis on a cité les différents caractéristique de PET, ensuite on a fait une description générale de PET universitaire et la génération automatique de ce problème, et enfin on a détaillé les approches de résolution de PET que se soit approche centralisé et décentralisé.

ème

3 Chapitre

Conception et Implémentation



1. Introduction

Ces dernières décennies, le développement d'une application concernant la génération automatique d'emploi du temps a été focalisée par un grand nombre de la communauté scientifique. Cela donnant plusieurs approches et travaux de recherches qui ont été proposé la résolution de ce problème, parmi les (on a déjà expliqué le principe des certains méthodes) on trouve l'approche décentralisé qui ce base sur les SMA.

Dans ce chapitre, on va présenter les différentes étapes de la réalisation une application d'emploi du temps des cours universitaires.

2. Conception

La phase de conception est incontournable, elle est la plus importante étape dans le processus de la génération qui on ne peut pas éviter ou développer. Dans ce point, on va présenter l'architecture générale de l'application puis on a détaillé les différentes phases.

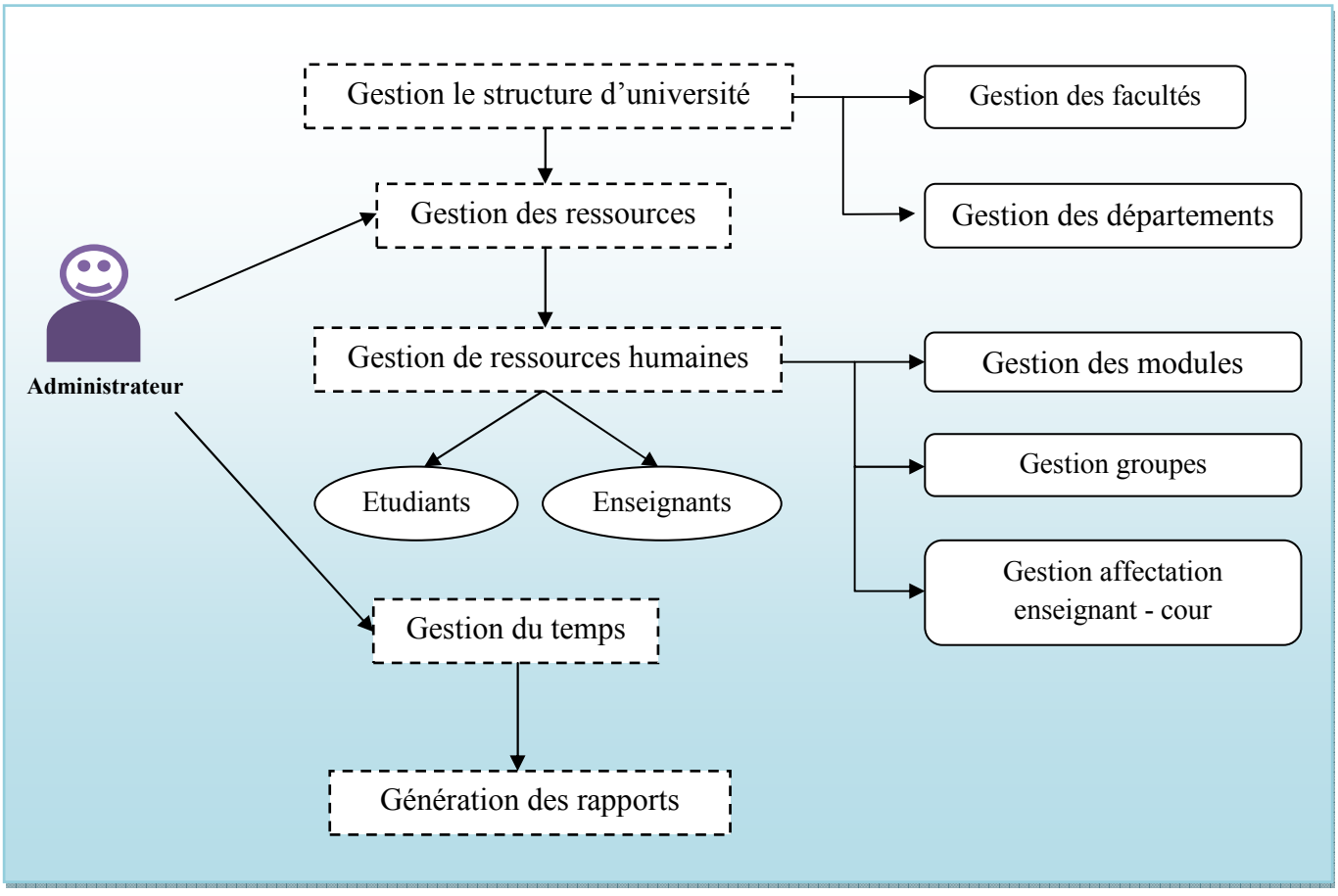


Figure III.1 : La conception générale de l'application.

2.1. Gestion de la structure d'université

La gestion des structures universitaires est réalisée par l'administrateur de l'administration pédagogique, dans cette phase l'administrateur doit se connecter au système de gestion qui lui demander le mot de passe, le système vérifier l'existence de l'administrateur et ouvre la session.

Après l'authentification, la session ouverte offre au l'administrateur deux services :

2.1.1. Gestion des facultés

Au niveau de ce service l'administrateur peut :

➢ **Ajouter une faculté :** l'administrateur peut ajouter une faculté à la liste des facultés avec ces caractéristiques et ces informations nécessaires.

➢ **Modifier une faculté :** permet de modifier les caractéristiques d'une faculté comme suite :

1. L'administrateur sélectionne une faculté dont laquelle il veut la modifier.
2. Le système affiche toutes les informations de la faculté choisi.
3. L'administrateur ajoute et confirme les modifications.

➢ **Supprimer une faculté :** Ce service permet d'éliminer une faculté à partir de la liste des facultés, ce processus passe par les étapes suivant :

1. L'administrateur du système demande la suppression d'une faculté.
2. L'administrateur entre le nom d'une faculté puis il confirme.

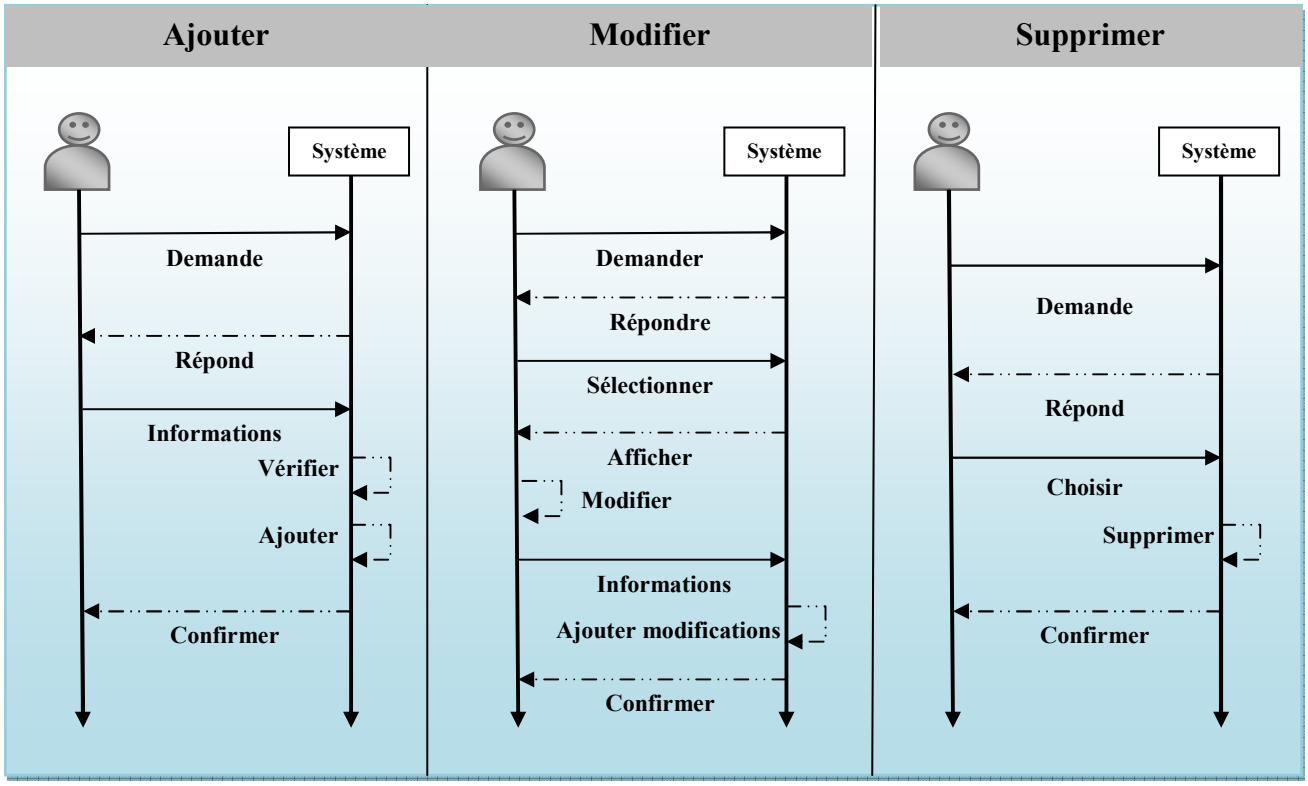


Figure III.2 : gestion de faculté.

2.1.2. Gestion des Départements

La gestion de département est comme celui de la faculté. Elle aussi offre trois services (ajouter, Modifier, Supprimer).

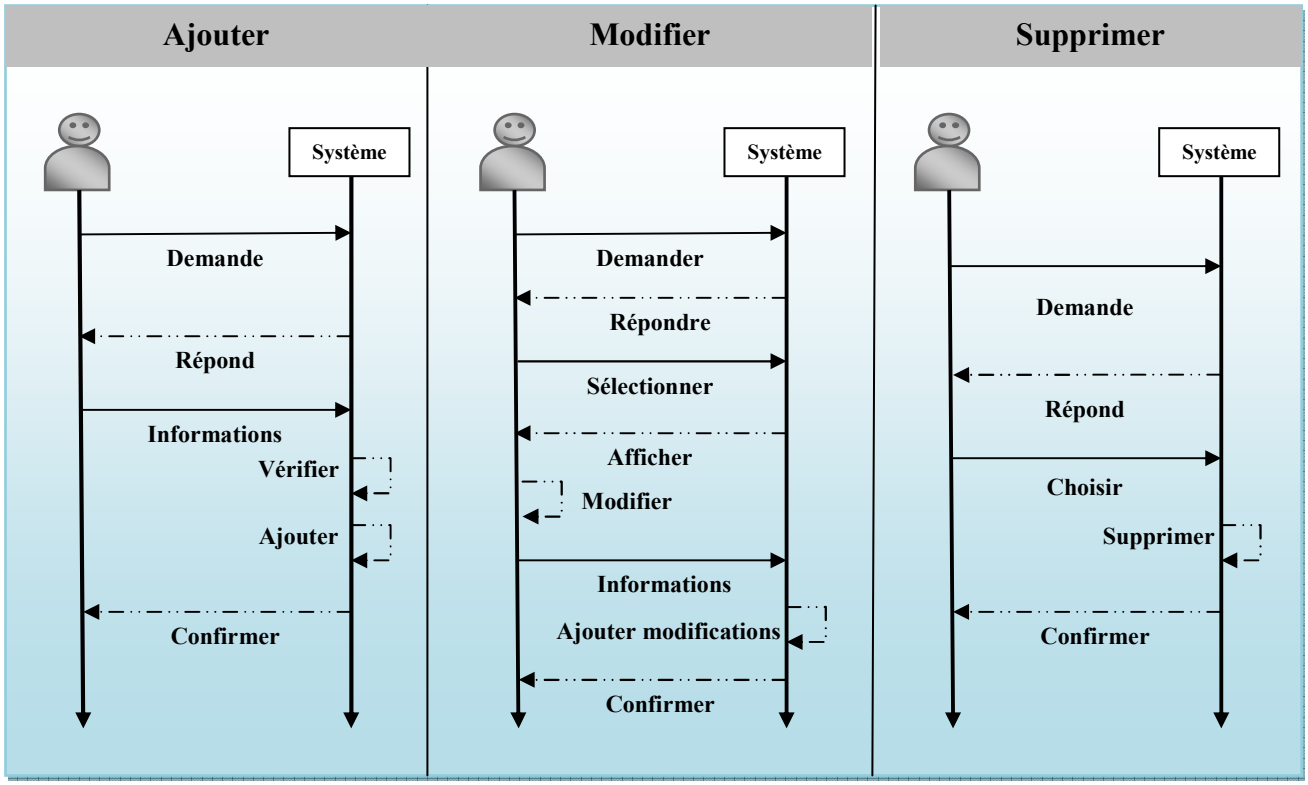


Figure III.3 : gestion de département.

2.1.3. Gestion des ressources

2.1.3.1 Gestion des modules et groupes

L'objectif de cette étape est gérer tous les modules et les groupes concernant les filières d'un département. Comme les étapes précédents, la gestion des modules ou groupes offre trois services de gestion (Ajouter, Modifier, Supprimer).

➢ **Ajouter un module /groupe :** Permet d'ajouter un (Module/groupe) à la liste des (Modules / groupes) selon l'enchaînement suivant :

1. L'administrateur du système demande d'ajouter un nouveau (Module/groupe).
2. Le système répondre au demande de l'administrateur et affiche la session.
3. L'administrateur saisit les informations et les caractéristiques d'un (Module/groupe) puis confirme la modification.

➢ **Modifier un module /groupe :** permet de modifier les informations d'un (Module/groupe). Le processus est comme suite :

1. L'administrateur demande la modification d'un module ou groupe qu'il a choisi.
2. Le système est toujours répondre à la demande de l'administrateur et afficher les informations du module ou groupe sélectionné.
3. L'administrateur ajouter les modifications et confirmer.

➢ **Supprimer un module /groupe :** La gestion de (Module/groupe) donne au l'administrateur l'avantage de suppression.

1. L'administrateur demande le service de suppression.
2. Le système répond à la demande de l'administrateur.
3. L'administrateur saisit le nom ou le matricule ….. et confirme.
4. Le système mettre-à-jour et sauvegarde la modification.

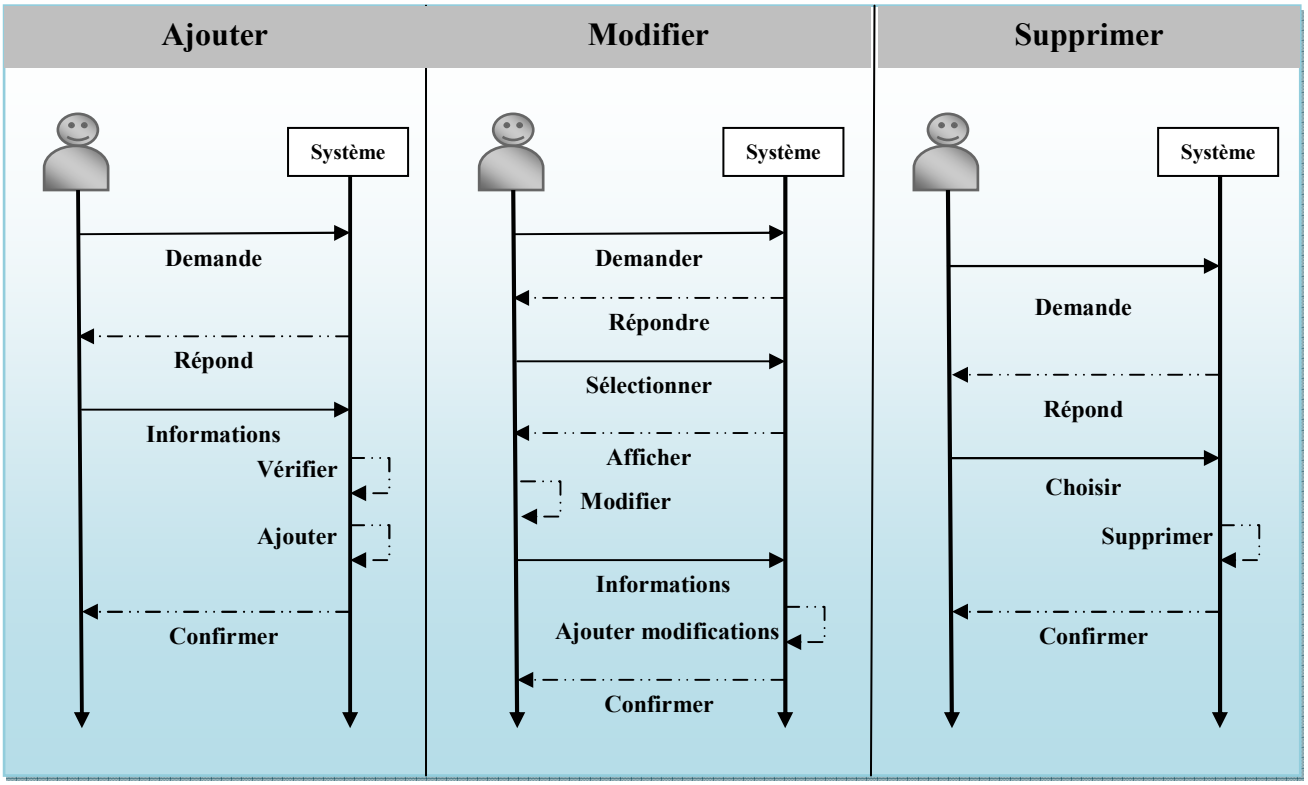


Figure III.4 : gestion de module/groupe.

2.1.3.2. Gestion de ressources humaines (enseignant/ étudiant)

La gestion des enseignants et des étudiants joue un rôle très fondamental dans la phase de gestion du temps.

➢ **Ajouter enseignant/ étudiant:** pour ajouter un étudiant ou un enseignant, l'administrateur doit sélectionner son choix.

1. Le système affiche le contenu à partir du choix sélectionné.

2. L'administrateur remplir les champs nécessaires et confirmer.

3. Enfin le système mettre-à-jour la liste et sauvegarde les modifications.

➢ **Modifier enseignant/ étudiant:** La modification des informations ou caractéristiques d'un étudiant /enseignant est faite comme suite :

1. L'administrateur demande la suppression.

2. Le système est toujours répondre.

3. L'administrateur choisit un étudiant /enseignant à partir de la liste.

4. Le système affiche les informations.
5. L'administrateur ajouter les modifications et confirmer.
6. Le système mettre-à-jour et sauvegarder la modification.

➢ **Supprimer enseignant/ étudiant:** la suppression est fait dans les étapes suivants :

1. L'administrateur demande la suppression.
2. Le système répondre au demande de l'administrateur.
3. L'administrateur tapez le nom ou le matricule de étudiant /enseignant qu'il veut éliminer et confirmer.
4. Le système cherche dans la liste de étudiant /enseignant et faite la suppression.

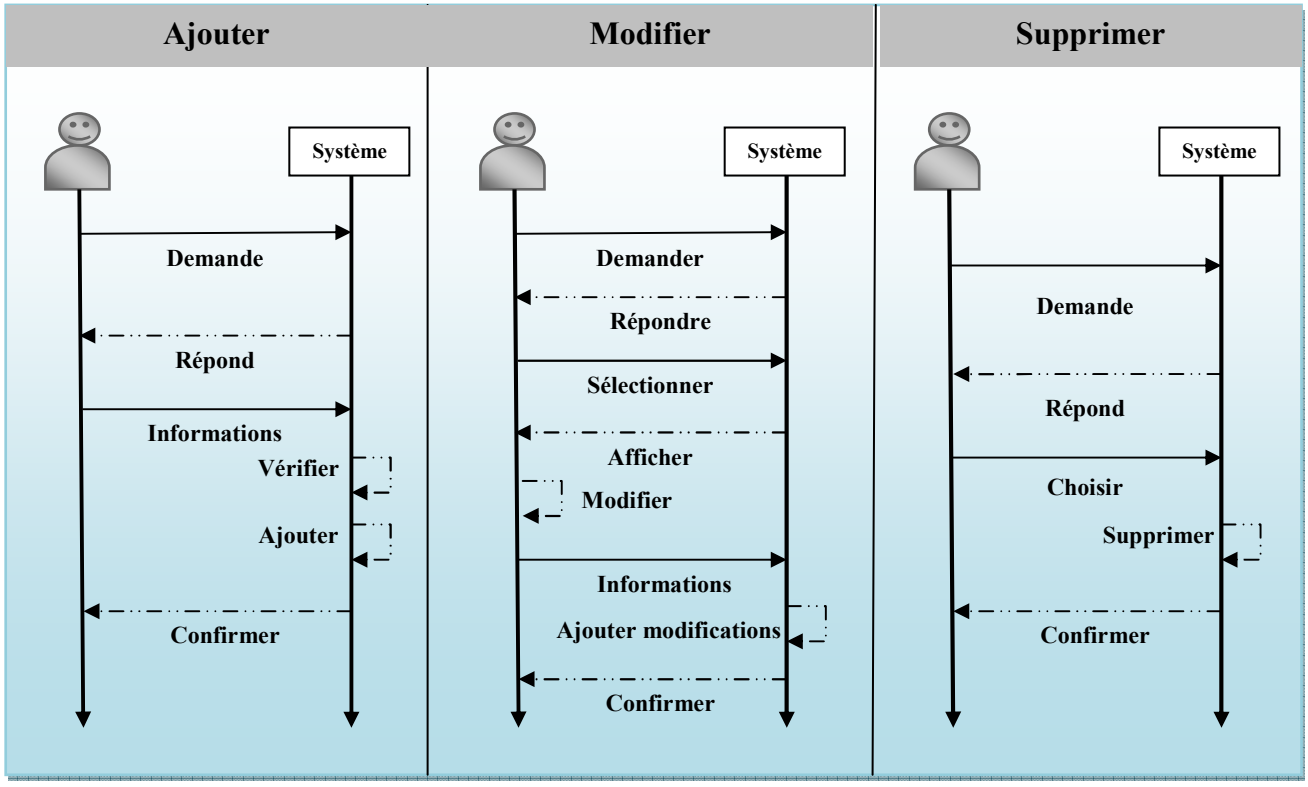


Figure III.5 : gestion de enseignant/ étudiant.

2.3.1.3. Gestion Affectation enseignant _ cours

L'affectation enseignant/ cour est une étape obligatoire car un enseignant ne peut pas enseigne tous les modules sauf les modules de ces compétences. L'opération d'affectation des matières aux enseignants se faite de façon manuel par l'administrateur.Cette étape a un seule service :

➢ **Affectation :** Ce service permet d'affecter un module à un enseignant.

1. L'administrateur du système demande l'affectation.
2. Le système répond à la demande de l'administrateur.
3. L'administrateur choisit un enseignant et le module qu'il doit enseigner en plus L'administrateur choisi aussi le type du module soit (cour, TD, TP).puis confirme.

4. Le système sauvegarde l'affectation dans la base de données.

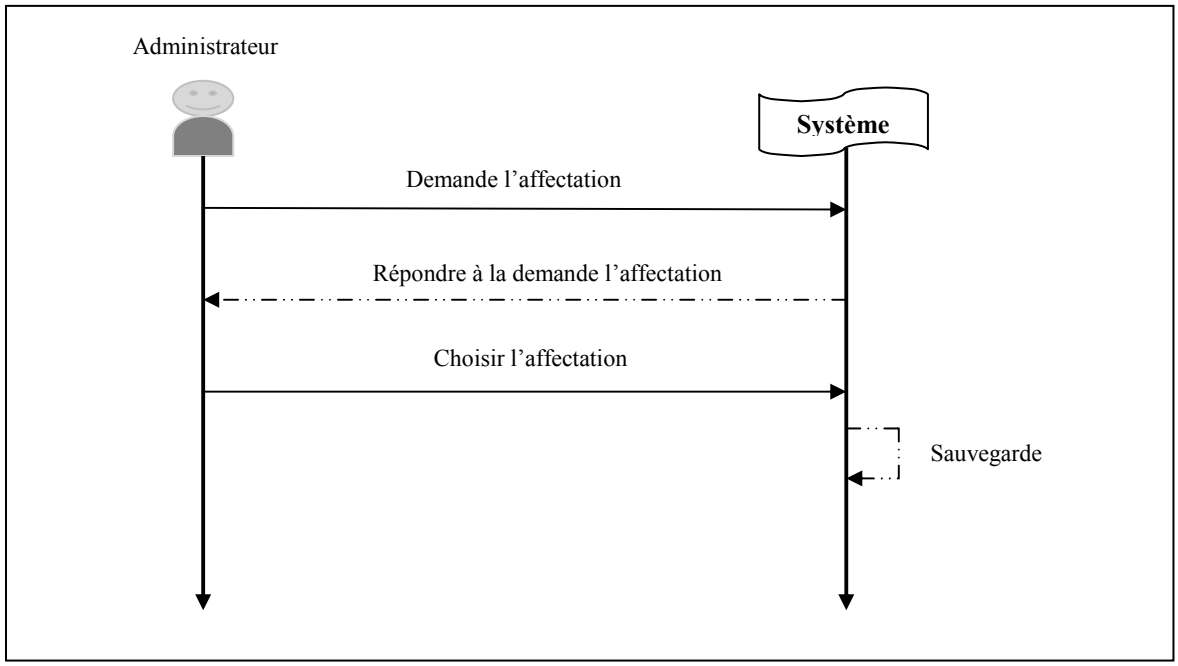


Figure III.6 : Affectation enseignant _ cour.

2.2. Gestion du temps

Cette étape présente le processus de notre architecture qui génère un emploi du temps universitaire, l'algorithme de ce processus est basé sur l'approche décentralisé ou le système multi-agent.

2.2.1. CSP Distribué

CSP distribué est un CSP dans lequel les variables et les contraintes sont distribuées entre un ensemble d'agents, chaque agent possède un sous-ensemble des variables. Les sous-ensembles des variables des agents forment une partition de X (l'ensemble des variables du problème). La répartition de l'ensemble des variables entre les agents conduit à distinguer deux types de contraintes : intra-contraintes et inter-contraintes **[BEL, XX]**.

- **intra-contraintes :** constituent le sous-ensemble de C des ces contraintes entre les variables locales d'un agent.
- **inter-contraintes :** est le sous-ensemble de contraintes C qui lient les variables de deux agents distincts.

2.2.2. Présentation de l'approche

La formalisation de problème de l'affectation d'un enseignant et une salle à un cour en respectant un créneau prédéfinis est caractérisé par :

a. Le duré journal de l'enseignement dans les systèmes universitaires est 450min (cinq séances par jour) et 2250 min par semaine (ça veut dire cinq jour).

b. Il ya un nombre maximum de cours à donner hebdomadaire.

c. Les enseignants ont des privilèges et ne sont pas toujours disponibles au cours de la semaine.

Le problème de génération est défini par les quintuplés suivants: {Cour, Enseignant, Salle, Contrainte, Horaire} :

- Cour= {C_1, C_2, …., C_C} l'ensemble des C cours.
- Enseignant= {E_1, E_2, E_3, …, E_e} l'ensemble des E enseignants.
- Salle= {S_1, S_2, S_3,…, S_S} l'ensemble des S salles.
- Horaire= {H1, H2, …, H_h} l'ensemble des h horaires dans une semaine.
- L'ensemble des contraintes.

2.2.2.1. L'Architecture multi-agents

On a trois catégories d'agent :

- **La première catégorie :** Cette catégorie présente « Agent enseignant » qui analyse et tester à chaque fois les demandes de réservation.
- **La seconde catégorie :** la deuxième catégorie présente « Agent cours », ce type des agents proposent au agent enseignant et agent salle pour réserver un enseignant et une salle au un cours dans une période du temps précis.
- **La troisième catégorie :** Cette catégorie présente « Agent salle », ce type d'agent est aussi faite d'analyse puis accepter ou refuser le demande de l'agent cour.

Dans le cas ou le problème traite de manière centralisé on définie une matrice géante notée ***Mat***[1..*e*][1..*s*][1..*c*][1..*h*] ou e_i C enseignants, s_i C salles, c_i C cours, h_i C horaires. Si le problème traite de façon décentralisée la matrice ***Mat*** est subdivisé aux sous-matrices et distribuer aux tous les agents du système en plus, les informations sont dispatchés entre les agents participant à la résolution, cette matrice est devient comme suite :

- L'agent enseignant e_i a une matrice ***Mat*** $_{ei}$ [1..s][1..c][1..h] tel que : s_i C salles, c_i C cours, h_i C horaires, e_i est fixé. Cet agent ne connait que le contenu de ***Mat*** $_{ei}$.
- L'agent salle s_i connait une matrice ***Mat*** $_{si}$ [1..e][1..c][1..h] tel que : e_i C enseignants, c_i C cours, h_i C horaires, s_i est fixé. L'agent salle aussi ne connait que le contenu sa matrice.
- L'agent cour C_i connait une matrice ***Mat*** $_{ci}$ [1..e][1..s][1..h] tel que : e_i C enseignants, s_i C salles, h_i C horaires.

2.2.2.2. Le processus de système

Le processus de la phase de gestion du temps concernant ce système est passe par trois phases :

- Phase d'initialisation.
- Phase de négociation et de communication.
- Phase de génération du résultat final.

➢ **La Phase d'initialisation :** au cour de cette phase, chaque agent de ce système initialise sa propre matrice que se soit Agent Cour, Agent Enseignant, Agent Salle. Cette initialisation se fait remplir toute la matrice par des zéros ça veut dire elle est vide et aucune position n'est réserver.

➢ **La phase de négociation :** Cette phase représente le noyau de l'architecture de système, elle est basée sur la communication directe par d'échange les messages, en d'autre terme les groupes des agents émettent et recèpent jusqu'à l'avoir la satisfaction et l'accord de tous les membres des groupes en pris en compte les conditions des contraintes dures concernant le problème.

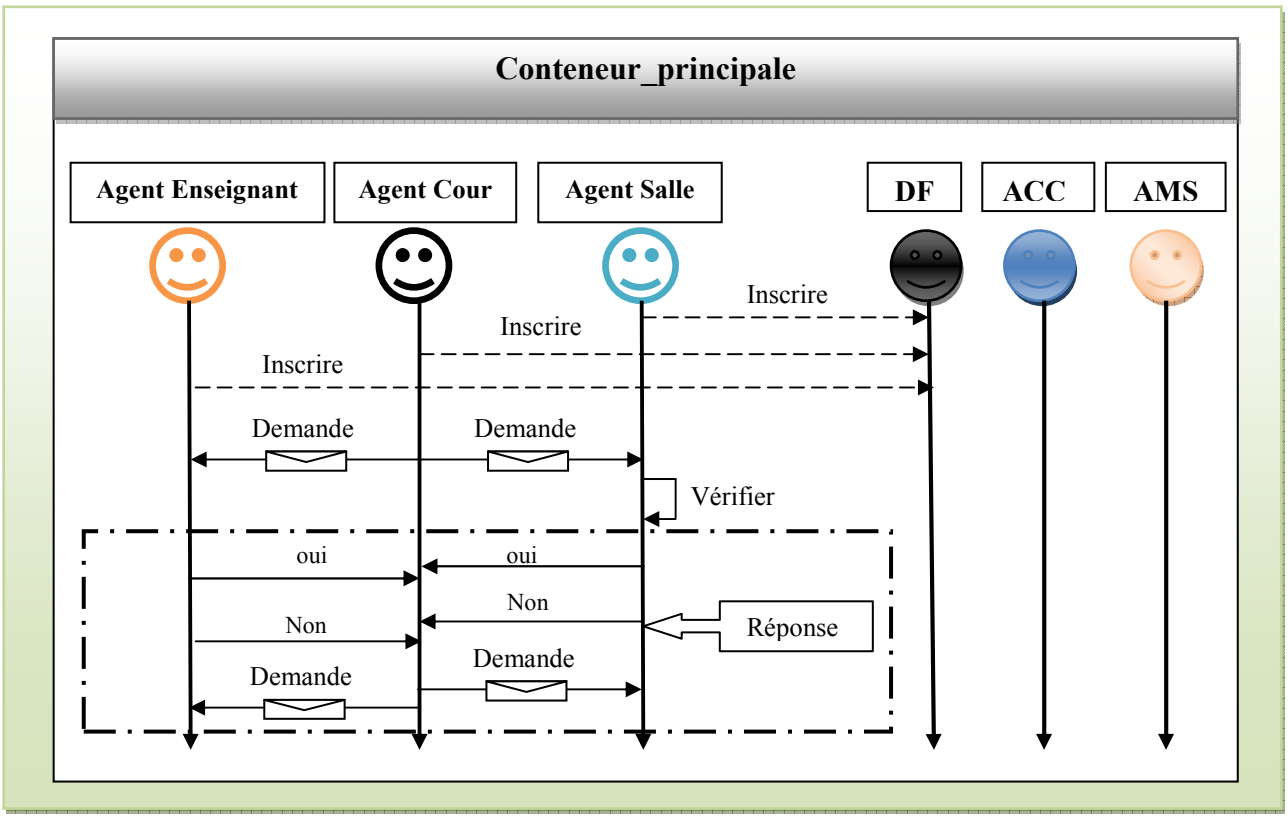


Figure III.7 : Le processus de négociation.

La figure ci-dessus explique processus de négociation entre les agents, ce processus est commencé par l'enregistrement de tous agent de système dans une page qui s'appelle « page jaune » de l'agent DF, ce dernier fournit ce service pour lesquelles l'agent découvre les autres membres à partir d'un annuaire de page jaune et peut communique avec eux.

Après les inscriptions l'agent cour cherche dans sa matrice une affectation {enseignent, salle, horaire} vide puis il envoie un demande de réservation aux agents salle et enseignant élus et attendre la réponse.

Quand l'agent salle et enseignant reçoivent les propositions de l'agent cour ils commencent d'analyser les contraintes dures et préférences de système puis ils répondent au l'agent cour, Si la réponse est

« accepter » alors l'agent cour note que celle affectation est la sienne, enfin tous les agents {enseignant, salle, cour} mettre-à-jour ses matrices.

➢ **La Phase de génération du résultat final :** Après la négociation, le résultat sera enregistré dans une base de données qui se générer comme un emploi du temps générale.

2.2.2.3. La structure des agents

Chaque agent de ce système possède une structure interne qui lui donné la capacité de d'agir, interagir et de communiquer avec les autres agents. La structure est contient : un ensemble des comportements, une mémoire local.

➢ **La structure de l'agent cour :** L'agent cour est un agent réactif, il a comme des connaissances des informations partiels sur les agents salles et enseignants (leur noms, adresse). Ces comportements est résumé par l'envoi des demandes aux agents et attendre la réponse pour réaliser l'affectation de se propre emploi.

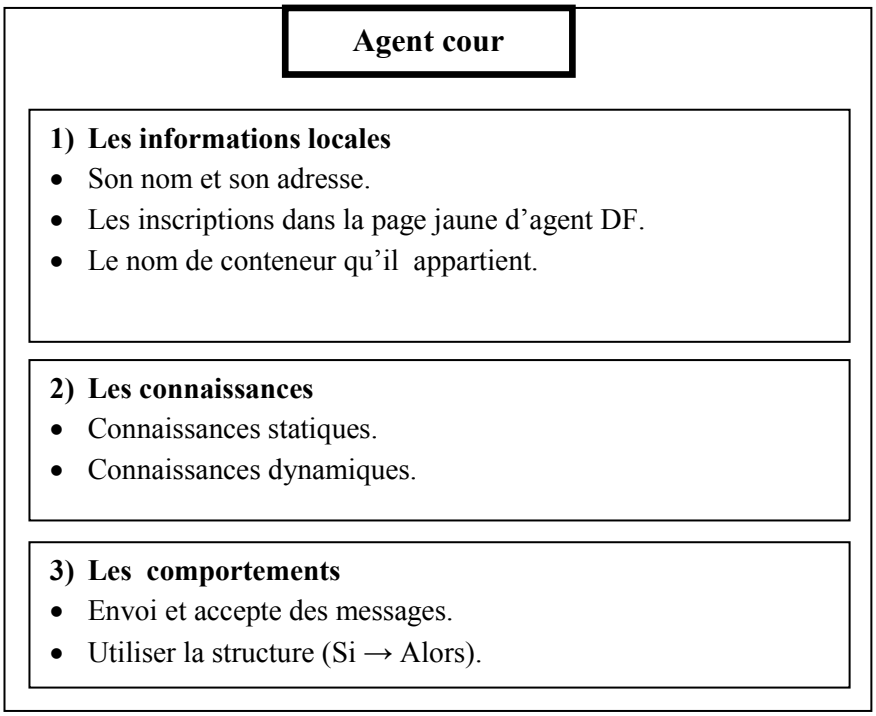


Figure III.8 : La structure de l'agent cour.

➢ **La structure de l'agent enseignant :** L'agent enseignant est aussi de type réactifs, sa structure est presque comme l'agent cour, sauf ce dernier a un comportement d'un planificateur ça veut dire, diriger la conversation …etc. Leurs connaissances statiques sont son nom et l'adresse et son conteneur, et Leurs connaissances dynamiques sont la matrice dans laquelle l'affectation désirable se fait.

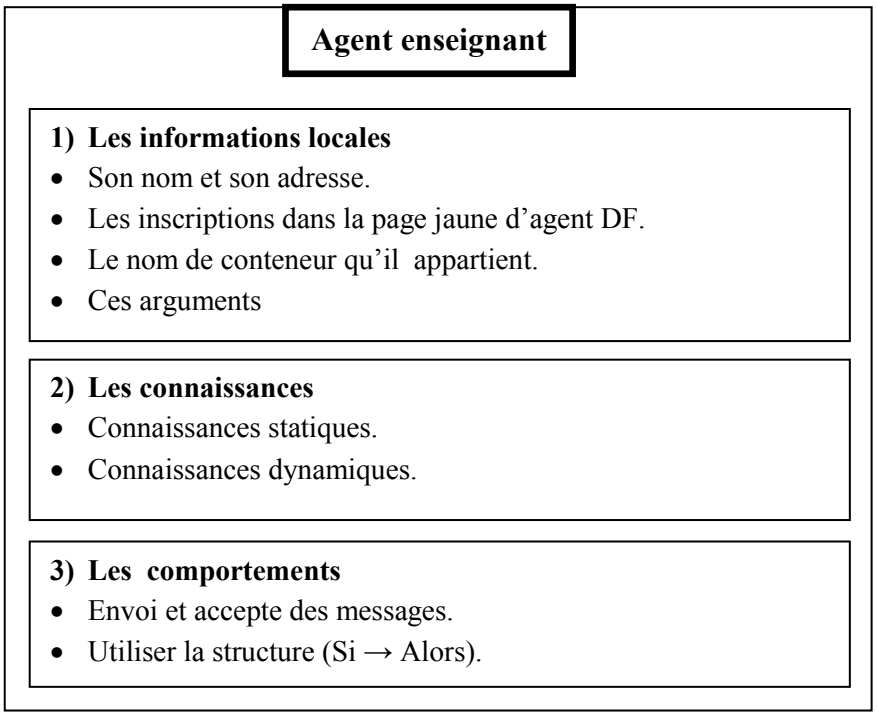


Figure III.9 : La structure de l'agent enseignant.

➢ **La structure de l'agent salle :** La class salle est de type réactifs, sa structure est presque comme l'agent cours, Leurs connaissances statiques sont son nom et adresse et son conteur.

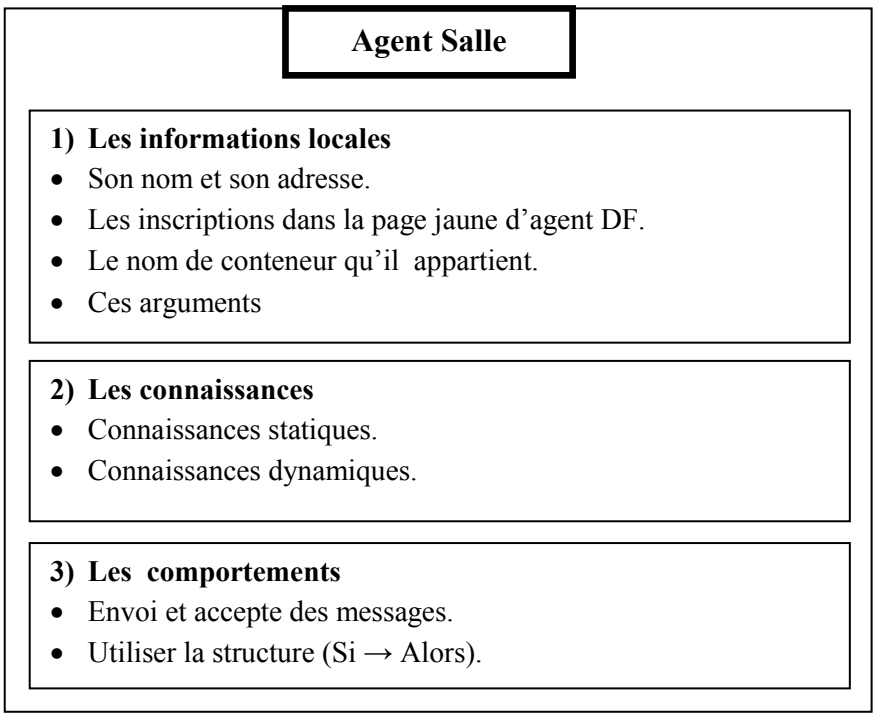


Figure III.10 : La structure de l'agent Salle.

2.2.3. Génération des rapports

Après la génération de l'emploi du temps, le résultat a besoin de rapporter ou les données est présenter de manière synthétique et lisible afin d'imprimer. Cette option est offre deux services :

- Imprimer le rapport.
- Envoyer le rapport par émail.

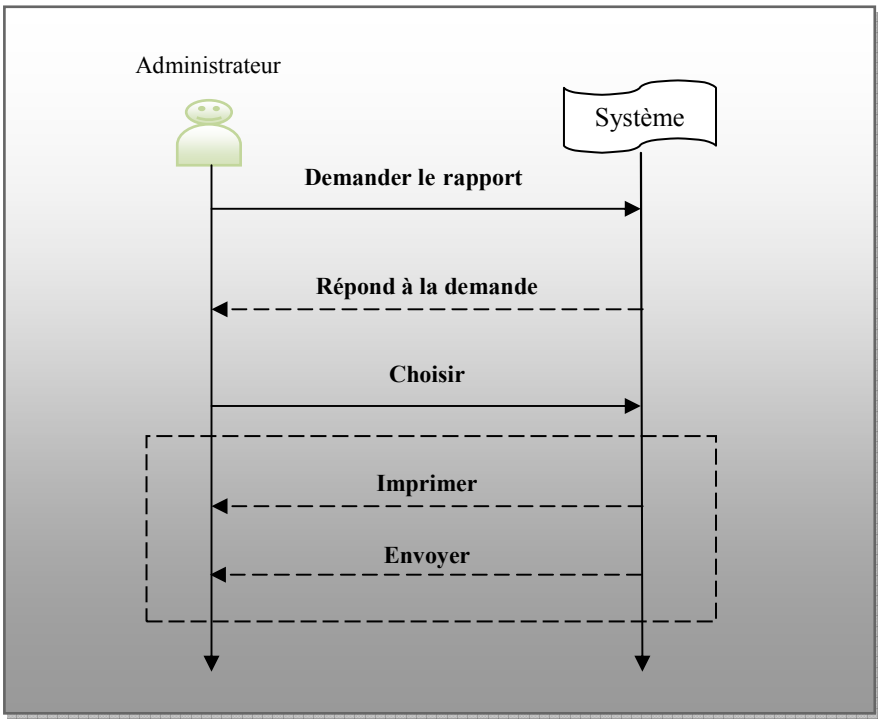


Figure III.11 : Gestion de Rapport.

3. Implémentation

Après avoir détaillé l'architecture et la conception du problème on passe maintenant à la phase de réalisation. L'objectif de cette phase est mettre les points sur l'application à travers l'environnement de développement, l'interface graphique et quelque résultat.

3.1. L'environnement de développement

L'environnement de travail est composé de deux coté : le coté physique c'est le matériel décrivant les différents caractéristiques de la machine utilisée, et le coté logiciel contenant des petites définitions du langage de programmation, la plateforme multi-agents et la base de donnée utilisée.

3.1.1. Langage de programmation « Java »

3.1.1.1. Présentation

Le langage java est un langage de programmation orienté objet crée par James Gosling et Patrick Naughton employés de l'entreprise Sun Microsystème qui est officiellement présenté en 1995. L'objectif central de java est que les logiciels écrit dans ce langage doivent être très facilement portables sur plusieurs systèmes exploitation tels que : UNIX, Windows, Mac OS.

Le langage java utilise les notions usuelles de la programmation orientée objet : la notion de classe, d'encapsulation, d'héritage, d'interface, de virtualité, de généricité, … etc. Il est accompagné d'un ensemble énorme de bibliothèques standard couvrant de très nombreux domaines, notamment des bibliothèques graphiques. C'est un langage qui présente d'excellentes propriétés de portabilité du code. Son gros point faible est une relative lenteur, surtout si on le compare à des langages comme le C++. Cependant, ce défaut a été résolu en grande partie par l'introduction de la technologie JIT (compilateur *Just-In-Time*, « juste à temps »), qui compile le code à la première exécution, permettant une exécution quasiment aussi rapide qu'en C/C++.

3.1.1.2. Domaines d'application

Il est possible d'utiliser Java pour créer des logiciels dans des environnements très diversifiés :

- Applications sur client lourd (JFC) ;
- Applications Web, côté serveur (servlets, JSP, Struts, JSF) ;
- Applications réparties (EJB) ;
- Applications embarquées ;
- Applications sur carte à puce ;
- Application Traitement automatique des langues ;
- Développement de jeux ;
- …etc.

3.1.2. Plateforme Jade

JADE (Java Agent Development Framework) est une plateforme multi agents répartie (multi-hôtes) développé par la société Telecom Italia Lab « Tilab» en 1999**.** JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA. Elle est implémentée en JAVA et fourni des classes qui implémentent « JESS » pour la définition du comportement des agents. JADE possède trois modules principaux (nécessaire aux normes FIPA) qui sont activés à chaque démarrage de la plate-forme :

- DF « Directory Facilitator » fournit 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.

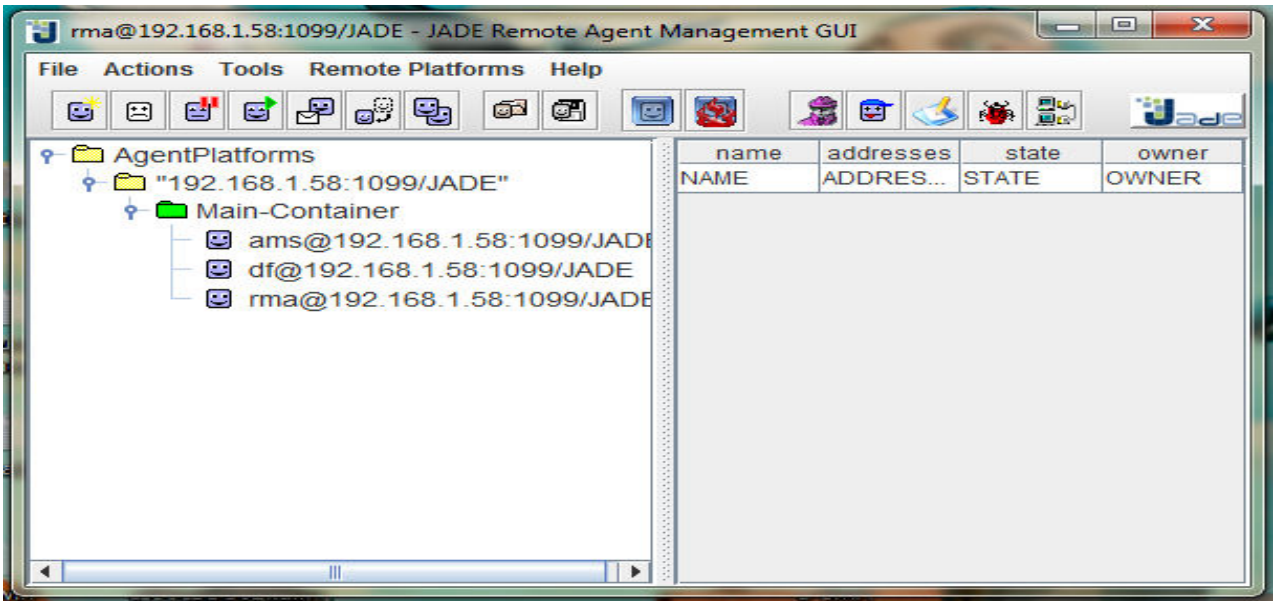


Figure III.12: L'interface de plateforme Jade

3.1.2.1.L'architecture de Jade

JADE reprend donc l'architecture de l'Agent Management Reference Model proposé par FIPA. Les services de base proposés sont le Directory Facilitator (DF) et l'Agent Management System (AMS). Il est possible de lui demander de tenir en plus le service de Message Transport Service (MTS) pour communiquer entre plusieurs plates-formes. Mais ce service sera chargé à la demande pour ne conserver par défaut que les fonctionnalités utiles à tout type d'utilisation. L'agent est l'acteur fondamental de la plate-forme, un Agent IDentifier (AID) identifie un agent de manière unique. Le DF est un composant qui fait office d'annuaire. C'est un service de « pages jaunes » qui permet de mettre en relation les agents avec leurs compétences. Un agent peut enregistrer ses compétences dans le DF ou interroger le DF pour connaître les compétences proposées par les autres agents. L'AMS est un autre composant important car il contrôle l'accès et l'utilisation de la plate-forme et maintient un répertoire contenant les adresses de transport des agents de la plate forme. Ce service est plus un service de type « pages blanches » qui effectuent la correspondance entre l'agent et l'AID.

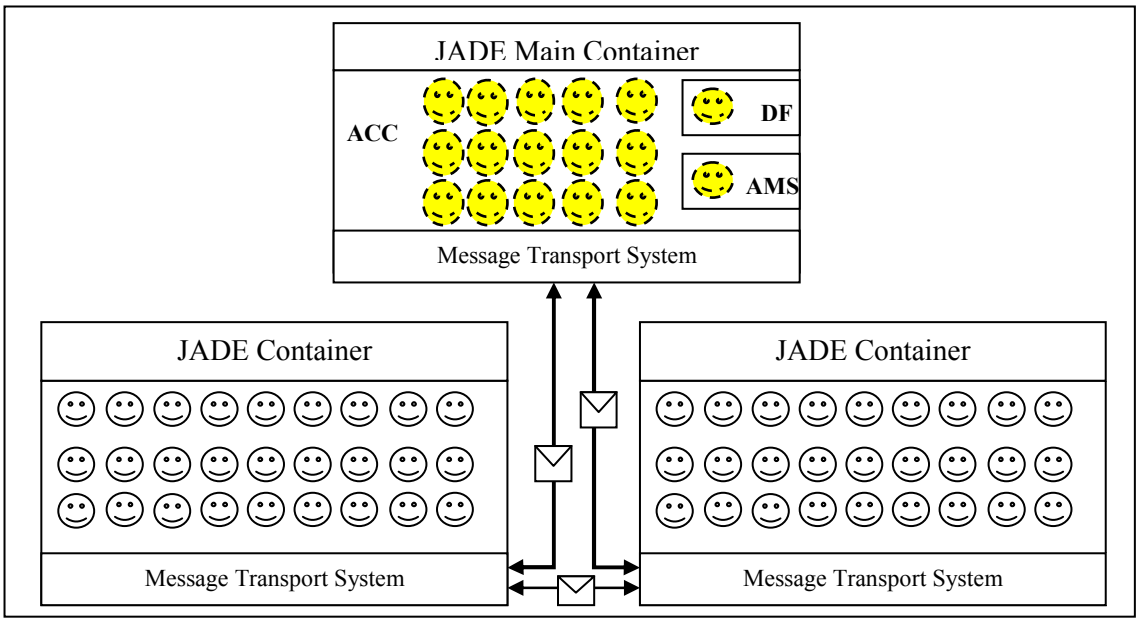


Figure III.13: L'Architecture de plateforme Jade

3.1.3. Base de données « Access »

Microsoft Access est un système de gestion de base de données relationnel édité par Microsoft. MS Access est composé de plusieurs programmes : le moteur de base de données Microsoft Jet, un éditeur graphique, une interface de type Query by Example pour manipuler les bases de données, et le langage de programmation Visual Basic for Applications. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde tel que : Oracle, SQL, … etc.

3.1.4. Les caractéristiques techniques

Notre application est développée dans un ordinateur dont les caractéristiques techniques, sont les suivantes :

Composant	Description
Processeur	Intel « I3 »
RAM	4 Go
Disque dur	500

Tableau IV.1: Les Caractéristiques Technique de l'Ordinateur de Développement.

3.2. L'interface Graphique

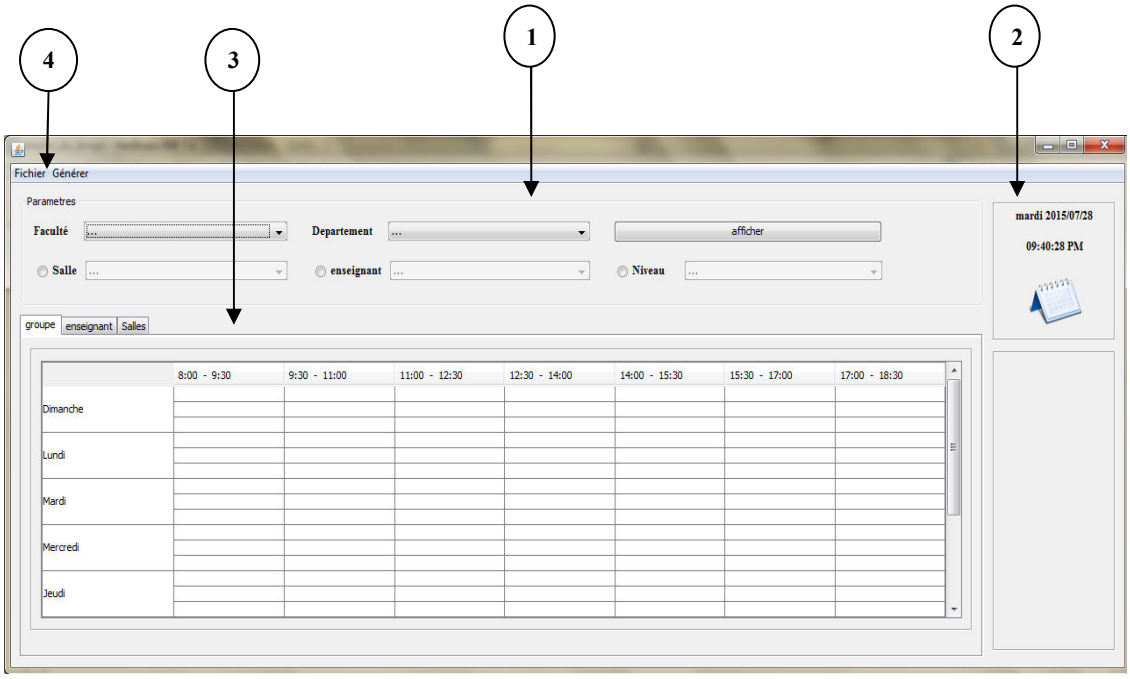
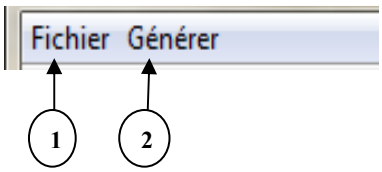


Figure III.14: L'Interface graphique de l'application

L'interface graphique est compose on :

- 1 : région de paramètres.
- 2 : Date et l'heure.
- 3 : l'affichage de l'emploi du temps.
- 4 : La barre de menu.

3.2.1. Barre de menu



- **1 :** Fichier.
- **2 :** Générer

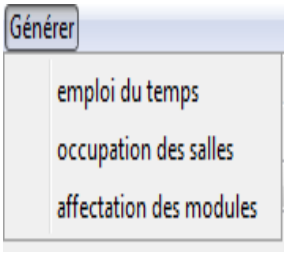
Figure III.15: Le menu de l'application

3.2.2. Menu « Fichier »

- Enregistrer :
- Importer :
- Exporter :

Figure III.16: Le sous_menu « Fichier »

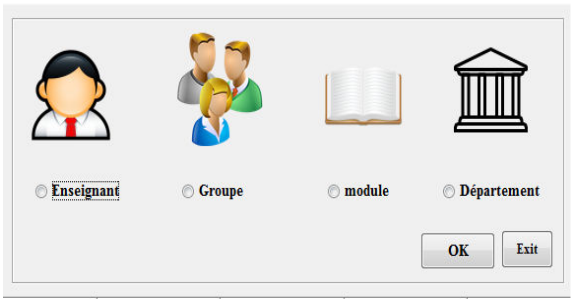
3.2.3. Menu « Générer »



- Emploi du temps
- Occupation des salles
- Affectation des modules.

Figure III.17: Le menu « Générer »

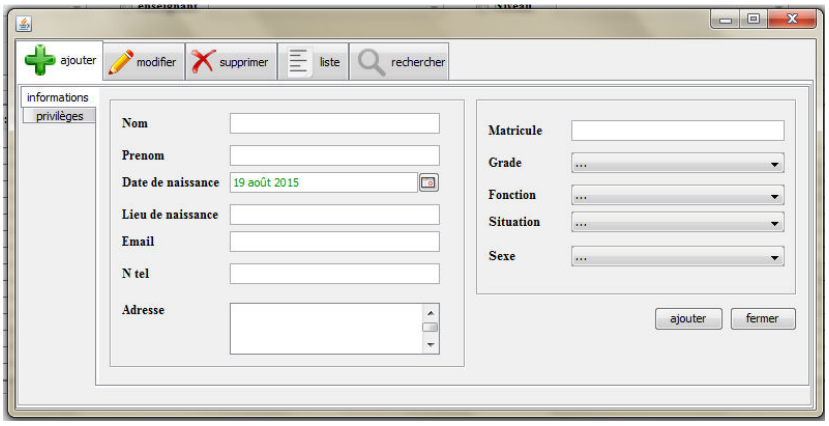
3.2.4. Sous Menu « Enregistrer »



- **Enseignant :** gestion des enseignants.
- **Groupe :** gestion des groupes.
- **Module :** gestion des modules.
- **Département :** gestion des départements.

Figure III.18: Le sous_menu « Enregistrer »

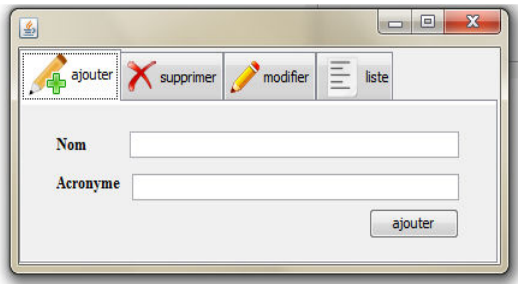
3.2.4.1. Enseignant



- Ajouter enseignant.
- Modifier enseignant.
- Supprimer enseignant.
- Liste enseignant.
- Rechercher enseignant.

Figure III.19: Le sous_menu « Enseignant »

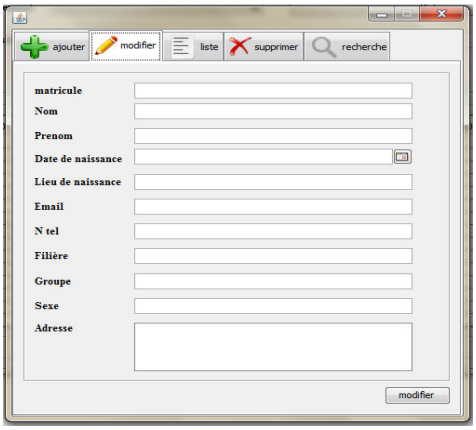
3.2.4.2. Département



- Ajouter département.
- Modifier département.
- Supprimer département.
- Liste département.

Figure III.20: Le sous_menu « Département»

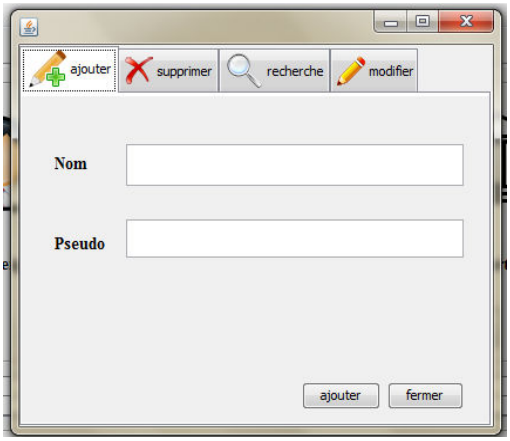
3.2.4.3. Etudiant



- Ajouter étudiant.
- Modifier étudiant.
- Supprimer étudiant.
- Liste étudiant.

Figure III.21: Le sous_menu « Etudiant»

3.2.4.4. Module



- Ajouter Module.
- Modifier Module.
- Supprimer Module.
- Liste Module.

Figure III.22: Le sous_menu « Etudiant»

3.2.5. Sous Menu « Importer »

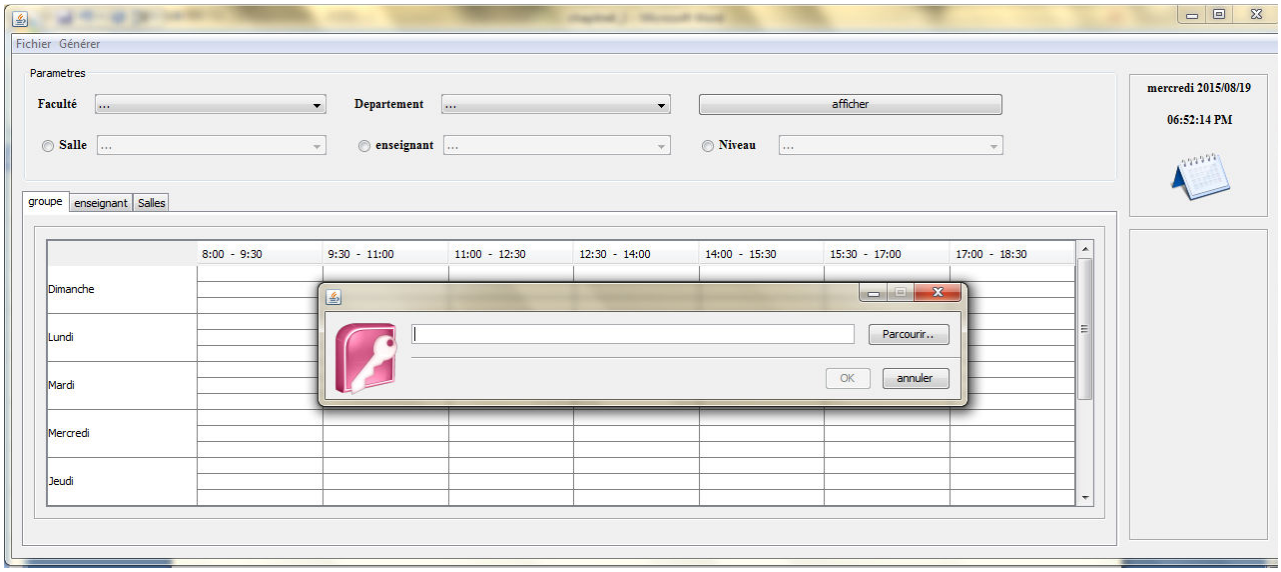


Figure III.23: Le sous_menu « Importer »

3.2.6. Aperçu des Résultats d'exécution

Pour la visualisation des résultats, on a proposé trois types de générations possibles :

- Emploi du temps concernant l'enseignant.
- Emploi du temps concernant la salle.
- Emploi du temps concernant le groupe des étudiants.

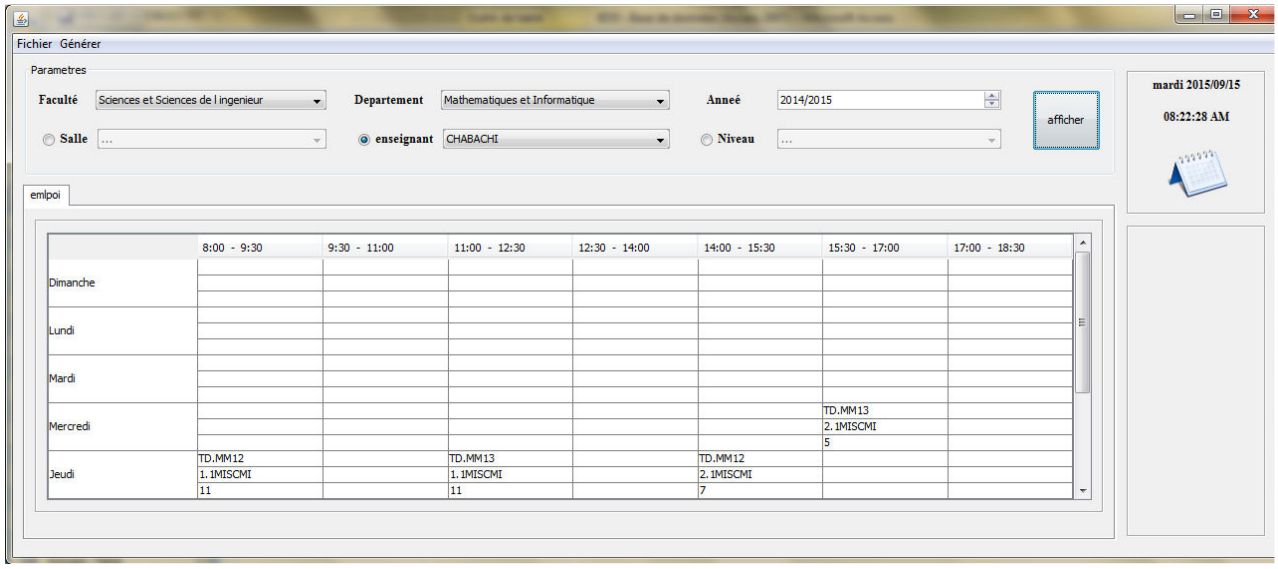


Figure III.24: Visualisation de l'emploi du temps de « enseignant ».

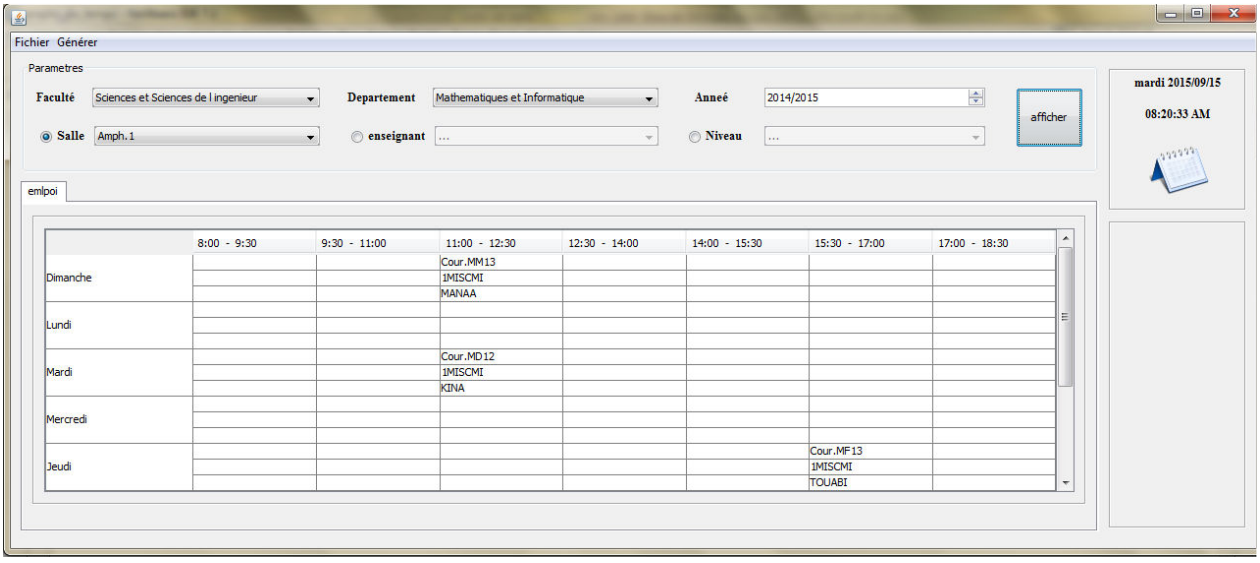


Figure III.25: Visualisation de l'emploi du temps de « Salle ».

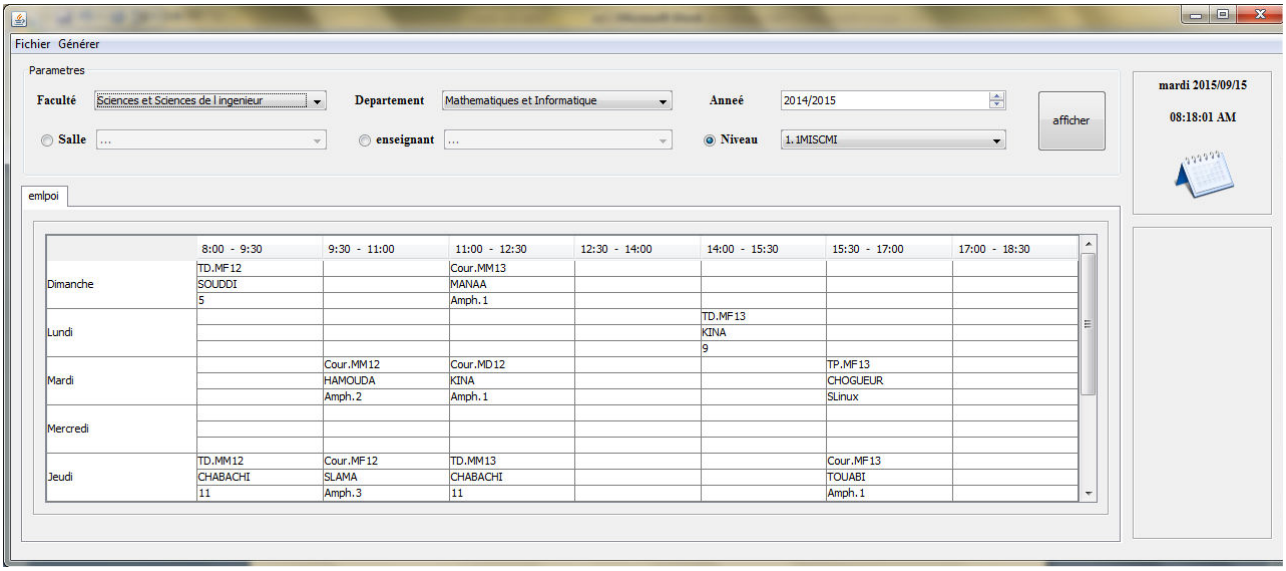


Figure III.26: Visualisation de l'emploi du temps de « groupe ».

3.3. Expérimentation

Pour tester notre système, on a choisi un niveau parmi les niveaux de MI. Les ressources nécessaires de traitement sont subdivisées comme suite :

- **Salles :** **1. Salles cours** {Amphi1, Amphi2, Amphi3, Amphi4}
  2 . Salles TD {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  3 . **Salles TP** {205, 206, S.Linux}
- **Enseignant : {** MAMOUNI, HAMOUDA, CHOGUEUR, TOUABI, MANAA, SLAMA, SOUDDI, CHABACHI, MEDIANI, …etc.**} .**
- **Modules : {** Mécanique du point, Technique d'expression, Electricité, Electronique, Algebre1, Informatique1**}**

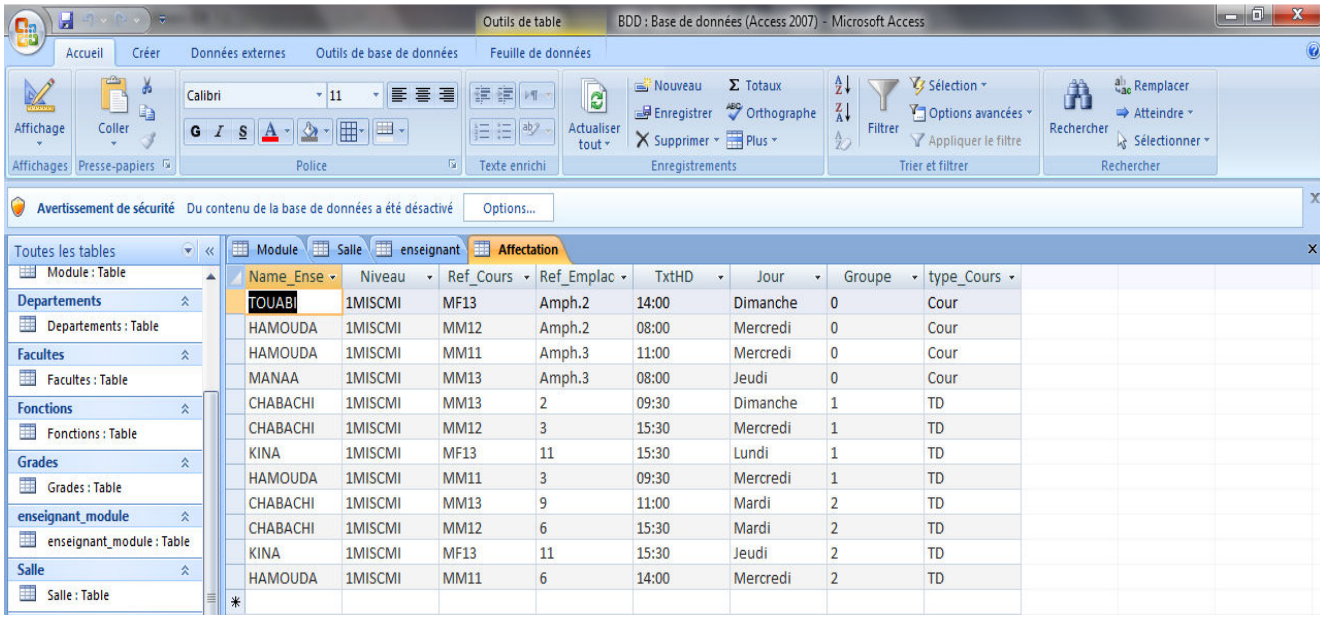


Figure III.27: Le résultat de test.

4.Conclusion

Au niveau de ce chapitre, On a expliqué l'architecture générale de notre système. Tout d'abord on a donné une brève explication sur les étapes de gestion des ressources pédagogiques puis on a détaillé l'approche qu'on a choisie pour planifier le temps concernant les établissements universitaire qui se base sur l'approche décentralisé, ensuite on a passé à la phase de l'implémentation et expérimentation de notre système.

Conclusion générale

L'automatisation d'un emploi du temps prend une place fondamentale entre les problèmes qui ont reçu une grande attention dans la communauté scientifiques, alors plusieurs efforts ont été déployés mais malheureusement tous les tentatives de modélisations mathématiques telle que : théorie des graphes, programmation linéaire… etc. ont échoué et ont montré que ces méthodes exactes ne suffisent plus dans la plupart des cas réels. La raison est que ces problèmes ont de complexité très élevées.

Ce mémoire présente le problème d'emploi du temps d'enseignement précisément les établissements universitaires, ce genre des problèmes est installé dans le domaine de l'intelligence Artificielle, qui a plusieurs méthodes de résolution comme l'algorithme génétique, Système expert, Recuit stimuli …etc. De notre coté le travail est situé dans le domaine de l'intelligence artificielle distribué qui se base sur un ensemble des agents coopératifs sous un architectures multi-agents.

La stratégie de ce mémoire est combiné la partie théorique avec la partie pratique dont le but de bâtir un système capable de générer un emploi du temps répond au maximum les besoins. C'est pour cela elle est devisé en deux parties : partie théorique, partie pratique.

- **Partie Théorique :** La première partie de ce mémoire est composé en deux chapitres, chapitre I est sous titre « L'IAD et Système multi-agents » qui introduire le domaine de l'intelligence artificielle distribué dont laquelle on s'est utilisé dans ce système et donner une vue générale sur le système multi-agents. La seconde chapitre « Le problème d'emploi du temps universitaire » ou on a donné une description détaillé sur ce problème et sa complexité avec une présentation les méthodes proposées pour la résolution du problème.
- **Partie Pratique :** La deuxième partie possède un seule chapitre « Modélisation et implémentation » dans laquelle on a expliqué la méthode utilise dans le système qui s'appuie sur l'approche décentralisé ou le SMA puis on a essayé de présenter l'implémentation de cette approche pour la réalisation de l'application.

Perspectives

Après la réalisation de l'application, on a fait un petit test pour tester sa robustesse, et généralement on a obtenir des résultats en courageux malgré avoir certain problèmes parmi eux : le problème d'insuffisance de la capacité et le problème du temps d'exécution .Comme perspectives, on peut citer quelques points qui peuvent améliorer la qualité de notre application dans le future :

- Utiliser la mode de communication indirecte qui se base sur l'utilisation d'une mémoire partagé au lieu de mode directe qui s'appuie sur l'envoi des messages.
- Essayer de hybrider la méthode décentralisé avec une autre méthode tel que : Algorithmique génétique a fin d'augmenter les performances.
- Adapter le système et ajouter des options supplémentaires comme la génération d'un emploi du temps concernant les examens.
- Remplacer l'ensemble des agents réactifs qui sont considéré comme des agents peu intelligents par un groupe agent cognitifs ou hybridés.
- Dépasser certains problèmes et trouver des solutions comme le problème de capacité concernant la machine.
- Essayer de générer un emploi du temps qui résoudre le problème d'insuffisances des salles.

Références Bibliographiques

[ABB et al, 06] : ABBAS Faicel et OMRI abd-elouahebe et COULIBALY Souleyman, *La mise en point d'un système de génération automatique de l'emploi du temps basé sur les systèmes multi agent*, Université 08 mai 1945, 2006.

[ALE, 06] : Alexander PAUCHET, *Modélisation cognitive d'interactions humaines dans un cadre de planification multi-agents,* thèse de DOCTORAT, Université Paris Nord, 2006.

[AME, XX]: Walter Amedzro St-Hilaire, *L'adaptation organisationnelle dans les théories managériales et sociales,* XX.

[BEL, XX] : BELAID Saad, *Integration des problèmes de satisfaction de contraintes distribués et sécurisés dans les systèmes d'aide à la décision à base de connaissances*, Université de PAUL Verlaine, XX.

[BRI et al, 01] : Jean-pierre Briot et Yves Demazeau, *Introduction aux agents : principes et architecture des systèmes multi-agents*, collection IC2 Hèrmes, 2001.

[BUR et al, 02]: Edmund Kieran Burke, Sanja Petrovic, *Recent research directions in automated timetabling*, University of Nottingham UK, 2002.

[CHA, 01] : Brahim Chaib-Draa, *Systèmes multi-agents : principes généraux et applications*, 2001.

[DAS et al, 04]: S.Daskalaki, T.Birbas, E.Housos, *An integer programming formulation for course study in university timetabling*, University de patras Gréque, 2004.

[DAV, 80] : Davis, *Rapport sur l'intelligence Artificielle Distribué*, SIGART Newsletter 73,1980.

[DOG, 93]: Alexis Drogoul, *De la simulation multi-agent à la résolution collectif des problèmes*, thèse doctorat, Université Paris VI, 1993.

[DUR et al, 90]: E. H. Durfee, T. A. Montgomery, *Hierarchical Protocol for Coordinating Multiagent Behaviors,* the 8th National Conference on Artificial Intelligence, 1990.

[FEH, 83]: M.Fehling and L.Erman, *Rapport sur le troisième atelier annuel sur DAI,* 1983.

[FER et al, 05] : FERRO Luca, KHIN Samith, SALMAN Nader, *Résolution pratique de problem NP_complets*, XX.

[FER, 95] : Jacques Ferber, *Les systèmes multi-agents : Vers une intelligence collective,* 1995, InterEditions.

[FER, 97] : Jacques Ferber, *Les systèmes multi-agents : un aperçu général*, LIRMM Université Montpellier II, 1997.

[FIN et al, 94]: Tim Finin, Don McKay, Rich Fritzson, Robin McEntire, *KQML: An Information and Knowledge Exchange Protocol,* 1994.

[FRA, 96]: Stan Franklin and Art Graesser, *Is it an agent or just a program?: A Taxonomy for Autonomous Agents*, 1996, University of Memphis.

[FRE et al, XX]: Louis Frécon, Okba Kazar, *Manuel d'intelligence Artificielle, université romandes*, XX.

[GIL, 97]: Don Gilbert, *Intelligent Agent: The information at the right Time*, IBM corporation, 1997.

[GLE, 05] : Benoit Glisse, *Schémas d'interactions dans les systèmes multi-agents*, Laboratoire d'informatique de paris VI, 2005.

[GLO, 89]: Fred Glover, *Tabu Search_partI* , ORSA journal of computing,1989.

[GUE et al, XX] : Guessoum Zahia René Mandiau Philipe Mathieu Olivier Boissier Pièrre Glize A.Hamri S.Pesty Gauthier Picard Jean-Paul Sansonnet Catherine Tessier Erwan Tranvouez, *Systèmes multi-agent et simulation*, LIP6-université de paris 6 et CReSTIC-université de Reims LAMIH-universite valenciennes et du Hainaut-Cambrésis LIFL-université des sciences et technologie de Lille G2I-ENSM.SE,Saint Etienne IRIT,

[HOU et al, 14]: Houssem Eddine Nouri, Olfa Belkahla, *Résolution multi-agents du problème d'emploi du temps universitaire*, éditions universitaire européennes, 2014.

[HUH, 87]: M. N. Huhns, *Distributed Artificial Intelligence*. Pitman Publishing Morgan Kaufman, 1987.

[LAB, 93] : S.Labidi, w.Lejouad, *De l'intelligence Artificielle Distribuée aux systèmes multi-agents*, Unité de recherche INRIA_ Antipolis, 1993.

[MOH et al, 11]: Mohammed El-Helly, Yasser Abdelhamid, Mohamed Al-Wakeel, *A Multi-agent Pattern Based Timetabling System*, Université Tabuk et Laboratoire central KSA et Egypt, 2011.

[MOY et al, XX] : Thierry Moyaux, Brahim Chaib-draa, Sophie D'amours, *Satisfaction distribué de contraintes et son application à la génération d-un emploi du temps d'employés*, Université Laval, XX.

[MUS, XX]: Mushi, *Tabu search heuristic for university course timetabling problem*, université de Dar es esalaam, Tanzania, XX.

[NAD, XX] : Nadine Richard, *Description de comportements d'agents autonomes évoluant dans des mondes virtuels,* Thèse de doctorat, XX.

[OUM, 09] : OUMAR Kane, *Nouvelle approches pour résolution du problème d'ordonnancement de projet à moyens limité*, Thèse de doctorat, université TOULOUSE, 2009.

[RUS, 95] : Stuart J.Russell and Peter Norvig , *Artificial Intelligence A Modern Approach*, Library of congress cataloging-in- publication Data, 1995.

[SAM et al, XX]: Samuel Lukas, Arnold Aribowo, Milyandreana Murchi , *Solving timetable problem by Genetic Algorithm and Heuristic Search Case Study*, Universitée Pelita Harapan Indonesia, XX.

[TAI et al, 12]: P.Taillandier O.Therond B.Gaudou, *Une architecture d'agent BDI basé sur la théorie des fonctions de croyance : application à la simulation du comportement des agriculteurs,* université de Rouen université Toulouse, 2012 .

[TRO, 06]: Troudi Fatiha, *Résolution du problème de l'emploi du temps : Proposition d'un algorithme évolutionnaire multi objectif* ; Information & Computation ; Université Mentouri Constantine ; 2006.

[Web2]: http://www.fipa.org/, Site official de FIPA réalise par: Jonathan Dale, dernier modification : Mercredi 3 décembre 2014 17 :55 :53.

[Web1]:http://turing.cs.pub.ro/auf2/html/chapters/chapter1/chapter_1_1_1.html , Politechnica University of Bucharest – 2002, dernier modification: Jeudi 7 Janvier 1999 03 :38 :50.

[Web4]: http://www.infres.enst.fr, site de Département Informatique et Réseaux, dernier modification: mercredi 2 septembre 2015 05 :02 :36.

[Web3]: http://www.eila.univ-paris-diderot.fr/ , Site officiel de L'université Paris Diderot, dernier modification: mercredi 16 septembre 2015.

[WEI, 99]: Gerhad Weiss, *Multiagent Systems a modern Approach to distributed Modern Approach to Artificial Intelligence*, The MIT press LONDON, 1999.

[WEY, 04]: Danny Weyns, *Environment for multi agent Systems state-of-the-Art and Research challenges*, LIRMM Université Montpellier II, 2004.

[WOO, XX]: Michael Wooldrige, *An Introduction to MultiAgent Systems*, XX.

[YAN, 03] : Yann Secq, *une méthodologie pour les systèmes multi-agents ouverts,* thèse de Doctorat, UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE, 2003.

[YOK, 00]: MAKOTO yakoo, KATSUTSHI Hirayama, *Algorithme for distributed constraint satisfaction : A review, NTT communication science laboratoire*, 2000.

[ZAH et al, XX] : Zahia Guessoum, Thomas Meurisse et Jean-Pierre Briot, *Construction modulaire d'agents et de systèmes multi-agents adaptatifs en DIMA*, Laboratoire d'informatique de Paris VI (LIP6), XX.