MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT AUTOMATIQUE

MEMOIRE DE MAGISTER

en ***Automatique***

Option : **Traitement d'Images et Reconnaissance de Formes**

présenté par :

M^{elle} NAIT BELAID Ouiza

ingénieur U.M.M.T.O

Thème

Apprentissage des systèmes multi-agents

Mémoire soutenu le : 06/01/2011 devant le jury d'examen composé de :

BENFDILA Arezki	Professeur à l'U.M.M.T.O	**Président**
DIAF Moussa	Professeur à l'U.M.M.T.O	**Rapporteur**
AHMED OUAMAR Rachid	M.C.A à l'U.M.M.T.O	**Examinateur**
OUANES Mohand	M.C.A à l'U.M.M.T.O	**Examinateur**
BENSIDHOUM Mohand-Outahar	M.C.B à l'U.M.M.T.O	**Examinateur**

Avant-propos

Ce mémoire a été effectué au laboratoire « robotique et vision » du département Automatique de l'UMMTO.

Mes vifs remerciements vont tout d'abord à Monsieur DIAF Moussa, professeur à l'UMMTO pour m'avoir proposé le thème de ce mémoire et m'avoir dirigée, aidée et conseillée tout le long de notre travail.

Nous exprimons notre reconnaissance à Monsieur BENFDILA Arezki, professeur à l'UMMTO, pour m'avoir fait honneur en acceptant de présider le jury de ce mémoire.

Nos remerciements vont également à Monsieur AHMED-OUAMAR Rachid, Maître de Conférences classe A à l'UMMTO pour avoir accepté de faire partie du jury d'examen de notre mémoire.

Nous exprimons également nos remerciements à Monsieur *OUANES Mohand*, Maître de conférences classe A à l'UMMTO pour avoir accepté de faire partie du jury d'examen de notre mémoire.

Que Monsieur BENSIDHOUM Mohand-Outahar, Maître de Conférences classe B et chef du département Automatique trouve ici nos remerciements pour avoir accepté de faire partie du jury d'examen de notre mémoire.

Nous ne manquerons pas de remercier Monsieur Amir NAKIB, chercheur au laboratoire LISSI de l'université Paris XII, Val-de-Marnes pour m'avoir conseillée et mise sur l'application relative au problème de tournée de véhicules.

À mes chers parents
À toute ma famille, mes ami(e)s
et à tout ceux que je porte dans mon cœur.

Sommaire

Introduction générale 1

Chapitre 1 : Apprentissage des systèmes multi-agents.....3

1. Introduction.....3

2. Notion d'agent.....4

    2.1. Différents types d'agents.....5

    2.2. L'environnement.....6

3. Les systèmes multi-agents.....7

    3.1. Interactions dans les systèmes multi-agents.....8

    3.2. Approches de modélisation d'un système multi-agent.....9

4. Domaines d'application des systèmes multi-agents 11

5. L'apprentissage dans les systèmes multi-agents.....12

    5.1. Les algorithmes d'apprentissages multi-agents.....12

    5.2. Discussion sur l'apprentissage des SMA.....19

6. Etat d'art sur l'apprentissage des systèmes multi-agents.....19

7. Conclusion.....25

Chapitre 2 : le problème de tournées de véhicules 26

1. Introduction.....26

2. Le Problème de Tournées de véhicules 27

2.1.Tournées de véhicules avec contraintes de capacité.....27

2.2 Tournées de véhicules dynamiques.....27

2.3 Tournées de véhicules avec livraison et collecte.....28

2.4 Tournée de véhicules périodique.....28

2.5 Tournées de véhicules avec collecte et livraison.....28

2.6 Tournées de véhicules multi-dépôts.....29

2.7 Problème du voyageur de commerce.....29

2.8 Tournées avec des véhicules à flotte variable VRPHF.....29

2.9 Tournées de véhicules ouvert.....29

2.10 Tournées de véhicules classiques.....29

2.11 Tournées de véhicules avec fenêtres de temps.....30

3. Les méthodes de résolution du VRPTW 31

3.1. Les méthodes exactes.....31

3.2 Les méthodes approchées 33

4. Applications des systèmes multi-agents.....36

6. Conclusion.....38

Chapitre3: Modélisation et résolution du problème posé.....39

1. Introduction.....39

2. Présentation du problème.....40

3. Découpage du problème en sous-problèmes.....41

4. Modélisation du problème par des agents.....45

4.1. Comportements des agents dans le système.....48

4.2. Les interactions entre les agents du système.....56

5. Conclusion.....60

Chapitre 4:Implémentation de la méthode et résultats.....61

1. Introduction.....61

2. Choix de la plateforme multi-agent.....62

3. Implémentation des agents du système.....62

4. Résultats de simulation.....66

5. Influence des paramètres sur le Q-learning.....72

6. Conclusion.....75

Conclusion générale.....76

BIBLIOGRAPHIE

ANNEXE

Introduction Générale

Les limites de l'approche classique de l'intelligence artificielle qui s'appuie sur une centralisation de l'expertise au sein d'un système unique, ont incité les chercheurs à créer une nouvelle discipline : l'Intelligence Artificielle Distribuée qui a visé la distribution de l'expertise et qui a introduit le concept de système multi-agents qui porte sur le modèle d'agents, dotés de capacités de perception, d'actions sur l'environnement, et qui doivent posséder une certaine autonomie de comportement et dont les caractéristiques sont la coopération, la coordination et la communication, etc. Les systèmes multi-agents apportent de nouvelles solutions pour traiter des problèmes complexes, mais il est presque impossible de concevoir des solutions *«optimales»* qui répondraient à toutes les problématiques et situations qu'on pourrait rencontrer. Cela nécessiterait d'avoir des agents capables d'apprendre et de s'adapter à ces nouvelles situations. Plusieurs algorithmes d'apprentissage ont été développés pour les agents apprenant, afin d'améliorer leur performances tout en leur permettant d'évoluer et la technique la plus utilisée est l'apprentissage par renforcement. Dans Cette méthode les agents apprennent à partir de leur expérience au moyen d'une évaluation par pénalité/récompense sans avoir besoin de spécifier comment la tâche doit être remplie. Dans ce présent travail, nous nous sommes intéressés à ce type d'apprentissage et, pour mieux cerner le fonctionnement et le comportement des agents apprenant, nous procédons au choix d'un domaine d'application afin de tester cette méthode. Les domaines d'applications des SMA sont nombreux et diversifiés. Dans notre cas, nous avons opté pour le problème de tournées de véhicules qui est devenu un des problèmes majeur qui préoccupe un grand nombre de sociétés. Il s'agit, plus exactement, d'une variante de ce problème, à savoir le problème de tournées

de véhicules avec fenêtres de temps avec un certain nombre de contraintes additionnelles. Schématiquement, le problème vise à générer un plan de transport pour transiter un flux entre différentes agences à l'aide d'une flotte minimale de véhicules, et chaque agence dispose d'une fenêtre de temps à l'intérieur de laquelle elle peut être desservie. Le flux transite (ou non) par un ou plusieurs hubs, en respectant un certain nombre de contraintes. Le problème est décomposé en sous-problèmes où différents agents coopèrent, collaborent, communiquent et apprennent en vue de prendre une décision en se servant de l'apprentissage. Pour cet apprentissage, le Q-Learning, une des techniques d'apprentissage par renforcement des systèmes multi-agents, a permis de trouver les meilleurs hubs à insérer dans les itinéraires d'un véhicule allant d'une ville de départ pour servir une destination donnée en respectant les contraintes. Le nombre de hubs varie en cours de l'apprentissage, ce qui nous donne la dynamique des actions du Q-Learning.

Ainsi notre travail est organisé comme suit :

- *Le premier chapitre* est une introduction aux SMA. Les différentes définitions relatives au concept « *agent* » et « *systèmes multi-agents* », leurs méthodes d'apprentissage et un état d'art sur les travaux réalisés dans l'apprentissage des systèmes multi-agents est présenté.
- Le *deuxième chapitre* est dédié à l'application choisie. Le problème de tournées de véhicules avec ses différentes variantes, les méthodes de résolutions fréquemment utilisées dans la littérature sont présentées. Le chapitre est terminé par une présentation des cas d'applications systèmes multi-agents *pour la* résolution du problème de tournées de véhicules à fenêtre de temps.
- *Le troisième chapitre* est consacré à la modélisation du problème par les systèmes multi-agents en décrivant les comportements, l'interaction des agents et l'application du Q-learning afin de résoudre le problème.
- Le *quatrième chapitre* présente les différents résultats des agents, notamment les résultats des agents d'apprentissage.

Chapitre 1

Apprentissage des systèmes multi-agents

1. Introduction

Si l'informatique centralisée est aujourd'hui considéré assez statique et ayant atteint ses limites avec la croissance de la taille et de la complexité de nouvelles applications, la décentralisation impliquant la coopération entre modules et logiciels est devenue un besoin important. Après la programmation orientée objet qui décentralisait déjà les fonctionnalités dans des composants autonomes, mais nécessairement réalisés de la même façon, avec l'informatique distribuée, on donne plus d'autonomie et d'initiative aux différents modules logiciels permettant ainsi à des applications hétérogènes de tourner sur des plates-formes hétérogènes et de communiquer facilement entre elles. C'est en réponse à ces enjeux importants que le concept de système multi-agent a été proposé. Un système multi-agent (SMA) est, en fait, un système composé d'un ensemble d'agents situés dans un certain environnement et interagissant selon certaines relations.

Un SMA qui est donc l'un des aspects de l'Intelligence Artificielle Distribuée, constitue un moyen intéressant pour modéliser des comportements de

groupes dans différents domaines d'applications allant de l'industrie jusqu'aux sciences humaines et ce, à l'aide d'un ensemble d'agents intelligents fonctionnant en commun et dotés de capacités de perception et d'actions sur l'environnement. Les SMA évoluent généralement dans des environnements complexes autrement dit, larges, ouverts, dynamiques et non prévisibles. Pour de tels environnements, il est très difficile et parfois impossible de définir correctement et complètement les systèmes *a priori*, c'est-à-dire, lors de la phase de conception, bien avant leur utilisation. En effet, ceci exigerait de connaître à l'avance toutes les conditions environnementales qui vont survenir dans le futur tel que: quels agents seront disponibles à ce moment et comment les agents disponibles devront réagir et interagir en réponse à ces conditions. Une manière de gérer ces difficultés est de donner à chaque agent l'habileté d'améliorer ses propres performances et celles du groupe auquel il appartient, ce qui est réalisé en utilisant des algorithmes d'apprentissage qui dépendent des connaissances dont dispose le concepteur sur le domaine. Ainsi, dans ce chapitre, il s'agit pour nous de présenter les fondements des SMA, quelques unes de leurs applications, leur apprentissage et nous citerons quelques travaux déjà effectués dans le cadre de cet apprentissage.

2. Notion d'agent

Le terme d'agent a été utilisé dans différents domaines d'application comme la physique, la sociologie, la psychologie, l'économie, l'informatique etc. La définition d'agent varie et elle est définie différemment selon la discipline. Parmi les premières définitions, on retient celle de Ferber [1], qui définit un agent comme étant une entité physique ou virtuelle qui:

- est capable d'agir dans un environnement,
- peut communiquer directement avec d'autres agents,
- est mue par un ensemble de tendances sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser,
- possède ses propres ressources,

- perçoit, de manière limitée, son environnement,
- possède des compétences et offre des services,
- peut éventuellement se reproduire,
- a un comportement qui tend à satisfaire ses objectifs en tenant compte des ressources et des compétences dont elle dispose et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

De leur côté, Jennings, Sycara et Wooldridge [2], la définition d'un agent est proposée comme suit : *«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. »*

Les notions "situé", "autonome" et "flexible" prennent différentes significations. L'agent est *situé* s'il est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. En étant *autonome*, cela signifie qu'il est capable d'agir sans l'intervention d'un autre agent ou d'un operateur humain et il contrôle ses propres actions ainsi que son état interne. Si l'agent est *flexible,* dans ce cas, il est capable de *répondre à temps* et il est *proactif* et *social*. Répondre à temps, c'est être capable de percevoir son environnement et d'élaborer une réponse dans les temps requis. L'agent doit, aussi, exhiber un comportement proactif et opportuniste tout en étant capable de prendre l'initiative au "bon" moment. Pour être social, l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

2.1. Différents types d'agents

En fonction de la taille d'un agent, de sa complexité, de ses connaissances et de son raisonnement, nous pouvons classer les agents en trois grandes catégories : cognitif, réactif ou hybride.

Les agents réactifs sont des agents passifs qui réagissent seulement à un stimulus (fig.1). Ce type d'agent ne dispose pas de module de raisonnement interne.

Fig.1 Agent réactif

Les agents cognitifs disposent d'une capacité de raisonnement sur une base de connaissances et d'une aptitude relative à la gestion des interactions avec les autres agents et avec l'environnement (fig.2.). L'une des architectures cognitives les plus connues est, surtout, l'architecture BDI (*Believe Desire Intention*) qui, comme son nom l'indique, consiste à élaborer un processus décisionnel qui utilise une logique basée sur les notions de croyances, de désir et d'intention [3]. Les croyances correspondent à ce que l'agent connaît de son environnement, les désirs aux états possibles envers lesquels l'agent peut vouloir s'engager et les intentions aux états envers lesquels l'agent s'est engagé et envers lesquels il a engagé des ressources.

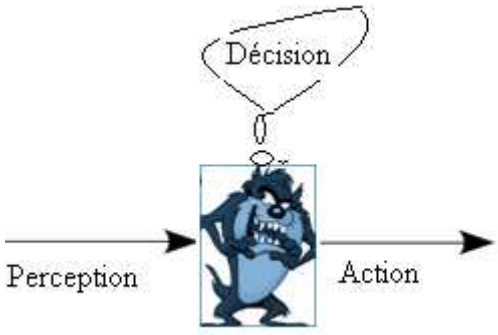

Fig.2 Agent cognitif

Quant aux *agents hybrides*, ils possèdent des composants réactifs et des composants cognitifs pour garantir un raisonnement de qualité.

2.2. L'environnement

L'environnement est le contexte dans lequel les agents évoluent. C'est un support commun aux actions des agents permettant ainsi l'interaction dans le système et il constitue une source d'information à laquelle les agents

peuvent accéder au travers de leur perception. Dans [4], les auteurs proposent un certain nombre de propriétés pour l'environnement et ils les classent de sorte que l'environnement soit accessible ou inaccessible, déterministe ou non déterministe, statique ou dynamique et discret ou continu. Ainsi, un environnement est accessible lorsque, pour un agent, il est possible d'obtenir la totalité des informations qui concernent son état. Dans le cas contraire il est inaccessible. Si l'environnement est déterministe, une même action aura toujours le même effet et sans aucune incertitude quant à son résultat. Dans le cas contraire il est non déterministe. L'environnement est statique s'il ne change que sous l'impulsion des actions effectuées par les agents. Dans le cas contraire, l'environnement définit aussi ses propres processus d'évolution et son état peut être modifié sans qu'aucune action ne soit intervenue. Cet environnement est discret lorsqu'il existe un nombre fixé de perceptions et d'actions possibles sur lui. Dans le cas contraire il est continu.

3. Les systèmes multi-agents

Plusieurs définitions ont été proposées pour un système multi-agents. D'après Mandiau [5], c'est «*un ensemble d'entités qui coordonnent leurs connaissances, buts, expériences et plans, pour agir ou résoudre des problèmes, incluant le problème de coordinations inter agents lui-même* ». Dans [1], Ferber le définit d'une façon plus détaillé comme étant un système composé d'un environnement et d'un ensemble (fig.3):

- d'objets passifs pouvant être perçus, créés, détruits et modifiés par des agents,
- d'agents actifs,
- de relations qui relient des objets entre eux,
- d'opérations ou de compétences offrant la possibilité aux agents de percevoir, produire, consommer, transformer et manipuler des lois de l'environnement,
- de lois appelées «les lois de l'univers » constituées d'opérateurs chargés de représenter l'application des actions des agents sur le monde et la réaction de ce monde par rapport à ces actions.

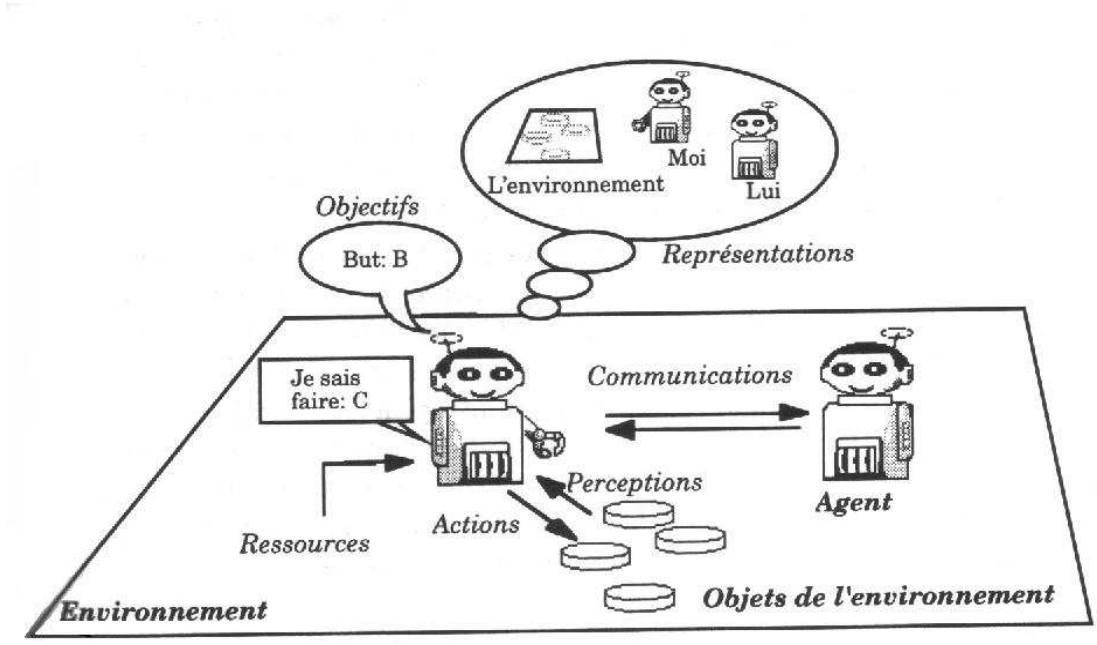

Fig.3. Représentation d'un système multi-agents d'après Ferber [1].

3.1. Interactions dans les systèmes multi-agents [6]

L'interaction entre agents représente l'une des plus importantes facettes d'un SMA et elle englobe la coopération, la coordination, la négociation, la collaboration, etc. Cependant, elle est, principalement, représentée par la coordination en faisant référence à tous les processus utilisés dans la prise de décision et au comportement global d'un ensemble d'agents. Ainsi, la coordination est définie comme la propriété d'un système composé d'au moins deux agents exécutant des actions dans un environnement partagé. Cette notion d'environnement ou de ressources partagées nécessite de la coordination des actions individuelles entre les agents pour atteindre l'objectif global du groupe. Cette coordination permet d'éviter les situations de conflits par la négociation pour les agents antagonistes, ayant des buts et des objectifs contradictoires, et d'améliorer l'efficacité de chacun des agents par la coopération pour les agents non antagonistes. Pour les agents antagonistes, la négociation permet à plusieurs agents d'arriver à un accord mutuel pour entreprendre une action donnée d'une certaine manière. Ceci est effectué grâce à un processus de communication, d'échange d'informations et de concessions mutuelles comme la relaxation de buts initiaux ou l'usage de mensonges ou de

menaces. La négociation peut être considérée comme une méthode de résolution de conflits et de recherche de consensus. Dans le cas d'agents non antagonistes, il s'agit d'une coopération entre ces agents pour se satisfaire mutuellement sans se gêner. Cette coopération, initiée par un échange d'information, est souvent associée à la notion de collaboration qui est une forme d'interaction qui étudie la manière de répartir le travail et, par conséquent, l'allocation de tâches entre plusieurs agents [7].

En ce qui concerne la communication dans les SMA, les agents peuvent communiquer *par mémoire partagée* qui nécessite une structure de données commune à l'ensemble des agents, comprenant tous les éléments nécessaires à la résolution d'un problème, Les agents déposent et lisent les informations sur cette zone de données communes appelée "*tableau noir*". La communication peut s'effectuer aussi par *envoi de messages* asynchrones suivant les protocoles de communication bien définis. Les connaissances et les mécanismes de traitement pour la résolution d'un problème sont distribués dans les agents. La communication s'établit, alors point par point, ou par diffusion. La communication peut aussi se réaliser par l'intermédiaire de l'*environnement* par le fait que les agents laissent des traces ou des signaux de leur présence qui peuvent être perçus par d'autres agents. Ce mode de communication s'applique à des domaines comme l'éthologie ou la robotique.

3.2. Approches de modélisation d'un système multi-agent

Plusieurs travaux de recherche se sont concentrés, ces dernières années, sur l'approche de modélisation et de conception d'un SMA. Différentes approches ont été proposées. Parmi ces approches, nous citons l'approche AALAADIN [8][9] et l'approche VOYELLES [10]. Dans l'approche *AALAADIN,* le modèle organisationnel est basé sur l'association de trois concepts clés : l'agent, le groupe et le rôle utilisés simultanément pour décrire des organisations concrètes d'agents (fig.4). Ainsi, l'agent est une entité autonome communicante qui joue des rôles au sein de différents groupes. Le groupe est un terme générique qui qualifie une communauté d'agents en relation par interaction, par partage d'un environnement, par un

but ou une ontologie commune, etc. Le rôle est une représentation abstraite d'une fonction, d'un service ou de l'identification d'un agent au sein d'un groupe particulier.

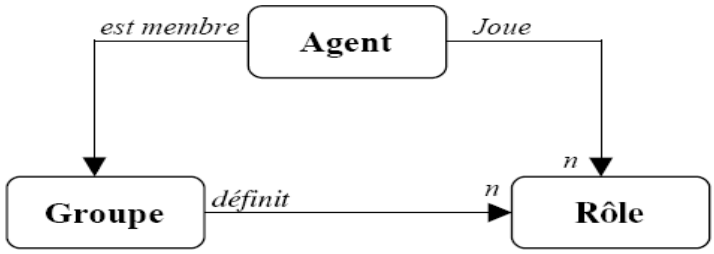

Fig.4. Le modèle organisationnel d'AALAADIN [8]

Notons que, dans le modèle AGR, un même agent peut jouer différents rôles et il participe alors à des interactions spécifiques à chacun de ses rôles. De même, un même rôle peut être joué par plusieurs agents.

L'approche VOYELLES, proposée par Yves Demazeau [10], consiste à considérer que l'analyse, le design, l'implémentation et le déploiement d'un système multi-agents peuvent être étudiés en fonction des quatre composantes fondamentales suivantes : Agents, Environnement, Interaction et Organisation (les voyelles A,E,I,O). Ainsi, la composante *Agent* représente l'ensemble des fonctionnalités de l'agent. Elle consiste à décrire l'architecture interne d'un agent à savoir, les mécanismes qu'il met en œuvre pour raisonner et produire son comportement, ses représentations de la connaissance, ses buts, etc. La composante *Environnement* est le milieu dans lequel sont plongés les agents et qui représente l'ensemble des éléments grâce auxquels les entités (agents) vont pouvoir agir ensemble, être mises en relation. La composante *Interaction* concerne les infrastructures, les langages et les protocoles d'interactions entre agents, depuis de simples interactions physiques à des interactions langagières par actes de langage. La *composante Organisation* représente les moyens utilisés pour structurer l'ensemble des entités.

Cette approche permet un jeu d'écriture par associativité afin d'exprimer le type d'approche choisie pour le développement d'un système. C'est ainsi que, à l'informatique, il a été attribué une approche (((*A*+*E*)+*I*)+*O*) dans laquelle les agents et l'environnement sont les priorités de développement (approche

centrée agent/environnement), alors que les sciences de l'économie se caractériseraient, plutôt, comme (((*O*+*I*+*E*)+*A*).

4. Domaines d'application des systèmes multi-agents

Les domaines d'application des SMA sont nombreux et ils sont utilisés dans plusieurs catégories d'applications comme la résolution de problèmes au sens large, la robotique distribuée, la simulation multi-agent, etc.

Pour la *résolution de problèmes* au sens large, elle concerne en fait toutes les situations dans lesquelles des agents logiciels accomplissent des taches utiles aux êtres humains. La résolution concerne la résolution distribuée de problèmes et la résolution de problèmes distribués. En résolution distribuée de problèmes, l'expertise globale est distribuée parmi l'ensemble des agents. Les agents aux compétences complémentaires coopèrent entre eux à la poursuite d'un objectif commun, tel qu'un diagnostic médical, la conception d'un produit industriel [11], l'acquisition de connaissances, le diagnostic de réseaux [12], la reconnaissance de formes [13], le système de contrôle et de surveillance d'un réseau de télécommunications [14], etc. En résolution de problèmes distribués, le problème est lui-même distribué et les agents peuvent avoir des compétences semblables. Il n'y a pas seulement distribution de l'expertise mais aussi distribution du domaine. Il s'agit d'applications telles que l'analyse, l'identification, le diagnostic et la commande de systèmes physiquement répartis pour lesquels il est difficile d'avoir une vision totalement centralisée.

Quant à la *simulation multi-agent*, elle 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 résultat des interactions d'un ensemble d'agents disposant de leur propre autonomie opératoire. L'intérêt de ces simulations est de pouvoir considérer aussi bien des paramètres quantitatifs, c'est-à-dire, des paramètres numériques que qualitatifs (des comportements individuels faisant éventuellement appel à des raisonnements stratégiques). Enfin, les SMA permettent la modélisation de situations complexes dont les structures globales émergentes des interactions entre individus, c'est-à-dire

de faire surgir des structures du niveau macro à partir de modélisations du niveau micro. Un des simulateurs les plus connus est le simulateur SIMPOP [15] qui modélise la dynamique de l'évolution d'un système de villes et, plus particulièrement, la genèse, le développement et la concentration des fonctions urbaines à différents niveaux durant une longue période de temps (environ 2000 ans).

En ce qui concerne la *robotique distribuée,* elle porte sur la réalisation d'un ensemble de robots qui coopèrent pour accomplir une mission. La robotique distribuée utilise des agents concrets qui se déplacent dans un environnement réel.

5. L'apprentissage dans les systèmes multi-agents

L'apprentissage d'un SMA a pour but d'améliorer les performances du système en tenant compte des ressources et des compétences dont il dispose. Elle est définie comme la construction de nouvelles connaissances ou amélioration de connaissances déjà existantes [16].

Le domaine de l'apprentissage a émergé suite à un réel besoin de techniques et de méthodes visant à améliorer les performances de ces systèmes tout en leur permettant d'évoluer. L'apprentissage dans un SMA comprend, d'une part, l'apprentissage dans un système mono-agent, parce qu'un agent impliqué dans un SMA peut apprendre de façon solitaire et indépendante des autres agents et, d'autre part, l'apprentissage dans le système multi-agent, parce que les agents dans un SMA peuvent apprendre de façon commune en échangeant des informations et en partageant des croyances.

5.1. Les algorithmes d'apprentissages multi-agents

Les algorithmes d'apprentissage multi-agents sont issus de la rencontre entre l'apprentissage artificiel et le modèle. Parmi les nombreux et variés algorithmes d'apprentissage, on retrouve :

- l'apprentissage *par renforcement* qui consiste, pour un agent, à apprendre un comportement optimal à partir d'interactions avec son environnement. L'agent et l'environnement interagissent à chaque séquence

discrète du temps, t = 0,1,2,3,.. Plus précisément, à chaque pas t l'agent reçoit une représentation G_t de l'état de l'environnement $e_t \in E$, où E est l'ensemble des états possibles. L'agent sélectionne alors une action $a_t \in A(e_t)$, où $A(e_t)$ est l'ensemble des actions possibles à l'état e_t. Une séquence ou un pas de temps après, suite à son action, l'agent reçoit une récompense numérique, G_{t+1} et se retrouve dans un nouvel état e_{t+1}. La figure 5 montre l'interaction agent-environnement.

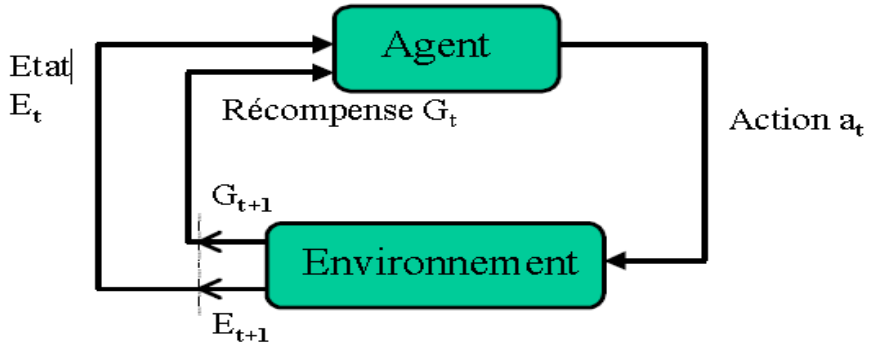

Fig.5. Interaction agent-environnement
dans un apprentissage par renforcement.

Le Processus de Décision de Markov ou MDP *(Markov Décision Process*) est le modèle généralement utilisé pour l'apprentissage par renforcement [17]. Ce modèle a, ensuite, été étendu de manière à prendre en compte l'aspect multi-agents. Le processus décisionnel de Markov est utilisé pour décrire l'environnement de l'agent ainsi que le but qu'il doit atteindre. Formellement, un MDP est un n-uplet *(S,A,T,R)* où S est un ensemble d'états, A, un ensemble d'actions, $T : S \times A \times S \rightarrow [0, 1]$, une fonction de transition qui donne la probabilité de passer entre le temps t et $t + 1$ de l'état s à l'état s' sachant que l'action choisie est a et $R : S \times A \times S \rightarrow R$, une fonction de récompense représentant le but de l'agent qui, à l'état courant, à une action donnée et à l'état suivant, associe une valeur $r \in R$. Cette valeur r est appelée récompense immédiate et définit implicitement le but de l'agent. Cette fonction peut être aussi définie sur $S \times A$ ou sur S sans que cela ne change le problème. Seules les équations pour la résolution seront légèrement différentes.

Notons qu'un MDP vérifie la propriété de Markov :

$$\forall s' \in S,\ P(s_{t+1} = s'/s_t, a_t) = P(s_{t+1} = s'/s_t, a_t, s_{t-1}, a_{t-1},\ldots,s_1, a_1) \quad (1)$$

Ceci signifie que le prochain état de l'environnement ne dépendra que de l'état courant et de l'action choisie par l'agent. L'objectif de l'agent est de maximiser le total des récompenses reçues. Dépendamment de l'horizon de calcul, il existe plusieurs manières de calculer un tel maximum. Dans le cas d'un horizon fini *T*, le total des récompenses reçues correspond à la somme des récompenses depuis le début des interactions jusqu'à l'horizon *T*, soit, $R=\sum_{t=0}^{T} r_{t+1}$, avec r_t, la récompense immédiate reçue à l'étape t. Toutefois, lorsque l'agent ne connaît pas l'horizon ou lorsque les interactions ne se terminent pas après un temps fini, il faut reformuler cette somme autrement. En effet, il est très facile de trouver une suite d'actions qui donnerait une somme de récompense infinie. C'est pourquoi, on introduit la notion de coefficient d'actualisation, noté γ et qui permet de donner plus d'importance aux récompenses proches plutôt qu'à celles données plus tard dans le temps. La somme des récompenses devient alors : $R = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, avec $0 \leq \gamma \leq 1$. Cette formulation est plus générale que la somme précédente car, dans le cas particulier où $\gamma = 0$, l'agent maximise la récompense immédiate alors que, lorsque $\gamma \to 1$, l'agent prend de plus en plus en compte les récompenses à long terme.

Le comportement d'un agent dans son environnement est défini par une politique π telle que : $\pi : S \times A \to [0, 1]$. Ceci constitue une politique qui est une fonction qui, à tout état $s \in S$ et toute action $a \in A$ associe la probabilité que l'agent joue l'action a dans l'état s. Pour calculer cette politique optimale, on introduit la notion de fonction de valeur *V*. La fonction de valeur $V^{\pi}(s)$ représente la qualité de l'état s en terme de récompenses espérées en suivant la politique π. Formellement, cette fonction de valeur est définie par :

$$V^{\pi}(s) = E_{\pi}[R_t / s_t = s] = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} / s_t = s\right] \quad (2)$$

La valeur d'un état en fonction de la valeur des autres états est donnée par l'équation de Bellman suivante :

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}[Rt \mid st = s] = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} / s_t = s\right] \\
&= E_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \middle| s_t = s\right] \\
&= E_{\pi}\left[r_{t+1} \middle| s_t = s\right] + E_{\pi}\left[\gamma\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \middle| s_t = s\right] \\
&= \sum_{a} \pi(s,a)\sum_{s'} T_{ss'}^{a} R_{ss'}^{a} + \sum_{a} \pi(s,a)\sum_{s'} T_{ss'}^{a} \left[\ \gamma E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \middle| s_{t+1} = s'\right]\ \right] \\
&= \sum_{a} \pi(s,a)\sum_{s'} T_{ss'}^{a} + \left[R_{ss'}^{a} + \gamma E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \middle| s_{t+1} = s\right]\right] \\
&= \sum_{a} \pi(s,a)\sum_{s'} T_{ss'}^{a} \left[\ R_{ss'}^{a} + \gamma V^{\pi}(s')\ \right] \qquad (3)
\end{aligned}
$$

On définit aussi $Q^{\pi}(s,a)$ comme l'espérance de récompense si l'agent choisit l'action a dans l'état s et s'il suit la politique π par la suite.

$$Q^{\pi}(s, a) = E_{\pi}[R_t / s_t = s\ a_t = a] = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} / s_t = s, a_t = a\right] \qquad (4)$$

Trouver une solution à un problème d'apprentissage par renforcement, c'est trouver une politique qui maximise la somme des récompenses sur le long terme. Pour savoir si une politique π‘ est meilleure qu'une politique π, on compare les fonctions de valeur pour tous les états. Dans ce cas, π’ est meilleure que π ($\pi' \geq \pi$) si et seulement si $V^{\pi}(s) \geq V^{\pi'}(s)$ pour tout s. Il existe une ou plusieurs politiques qui sont meilleures ou équivalentes à toutes les autres politiques. Il s'agit de la ou des politiques optimales π^{*}. La valeur de cette politique optimale, notée V^{*}, est définie par $V^{*}(s) = \max_{\pi} V_{\pi}(s)$ pour tout s. De même, on définit la fonction de valeur état-action optimale, notée Q^{*}, par $Q^{*}(s,a) = \max_{\pi} Q^{*}(s, a)$ pour tout s et tout a. On peut écrire $Q^{*}(s,a)$ en fonction de $V^{*}(s)$ comme suit (l’équation(5)):

$$Q^{*}(s, a) = E\left[r_{t+1} + \gamma V^{*}(s_{t+1}) \middle| s_t = s, a_t = a\right] \qquad (5)$$

L'équation optimale de Bellman pour $V^{*}(s)$ décrit le fait que la valeur optimale d'un état est égal à la récompense attendue pour la meilleure action dans cet état. Formellement, ceci s’écrit tel que:

$$V^*(s)=\max Q^*(s, a),=\max \sum_{s'} T_{ss'}^{a}\left[R_{ss'}^{a}+\gamma V^*(s')\right] \quad (6)$$

De même, on définit l'équation optimale de Bellman pour Q^* par l'équation(7) :

$$Q^*(s, a) = \sum_{s'} T_{ss'}^{a}\left[R_{ss'}^{a}+\gamma \max Q^*(s',a')\right] \quad (7)$$

L'un des principaux algorithmes de calcul de la politique optimale Q*, est le Q-Learning qui est la méthode utilisée dans notre travail, sera vue en détails dans le troisième chapitre et proposée par Watkins 1989 [18], repris par S. Sehad [19] comme méthode d'apprentissage par renforcement des systèmes multi agents dans le cas d'un MDP quand le modèle d'évolution est inconnu.

Les méthodes d'apprentissage par renforcement à base de processus de décision de Markov ne prennent en compte que des actions d'un seul agent. Cependant, plusieurs approches ont été proposées pour prendre en compte plusieurs agents.

Pour le principe de base, Boutilier [20] a proposé une méthode, appelée Processus de Décision de Markov Multi-agents (MMDP) inspirée des MDP, qui prend en compte un nombre quelconque d'agents qui partagent tous la même fonction d'utilité.

Ainsi, un MMDP est défini par un tuple < α, $A_{i\in \alpha}$, S , Pr , R > où :

- α désigne le nombre d'agents du système.
- chaque agent i ∈ α dispose d'actions individuelles décrites par A_i
- on définit A = ×A_i l'espace des actions jointes.
- *S* désigne l'espace d'état ;
- *Pr* la matrice de transition. *Pr : S × A × S → [0, 1]*
- *R* la fonction de récompense *R : S × A → R*

Pour la résolution, on suppose que tous les agents connaissent le modèle du monde et des récompenses. Chaque agent peut alors résoudre individuellement ce problème par planification en supposant qu'il contrôle le comportement de l'ensemble des agents du système. Résoudre un MMDP est équivalent à résoudre un MDP < S',A', T',R'> pour lequel :

- *S'* est égal à *S* ;
- *A'* correspond à l'espace des actions jointes ;

- *T'* correspond à *Pr* ;
- *R'* correspond à *R.*

En utilisant les algorithmes de résolution des MPD, chaque agent peut donc construire une politique optimale π*:S→A' comme s'il contrôlait intégralement les actions émises par tous les agents. Si tous les agents disposent du même modèle du monde, ils peuvent alors décider à chaque instant quelle action émettre en supposant que les autres agents aient effectué la même planification. Néanmoins, puisque l'exécution est décentralisée, un problème de coordination se pose quand deux actions jointes ont la même valeur. Dans ce dernier cas, chaque agent hésite entre plusieurs actions jointes et la réponse individuelle qu'il doit fournir. Si les agents choisissent des actions jointes différentes, cela peut grandement nuire à la performance du système.

Boutilier avance plusieurs propositions pour effectuer le choix d'une action jointe commune quand il y a ambiguïté. Il propose d'utiliser des *synchronisations par apprentissage*, *des conventions* et *de la communication.* A l'exécution, chaque agent évolue selon une certaine politique individuelle et apprend les réactions des autres agents lors des situations de conflit jusqu'à ce que l'ensemble des agents se synchronise sur une action jointe optimale qu'ils répéteront dans le futur. En ce qui concerne les *conventions*, les agents connaissent des règles qui permettent de lever l'ambiguïté sur l'action jointe à choisir en cas de conflit. Les agents peuvent communiquer en s'échangeant des messages pour se mettre d'accord sur l'action jointe à choisir dans les situations de conflit.

- l'apprentissage par *algorithmes génétiques* largement utilisé en Intelligence Artificielle Distribuée [21]. Le principe de l'algorithme est simple et repose sur un codage des problèmes et de leurs solutions sous la forme de chaînes d'éléments de base. Le but est de rechercher un extrema d'une fonction définie sur un espace de données. Les principaux avantages de ces algorithmes sont que leur fonctionnement est très simple à appliquer, qu'ils sont plutôt rapides et performants. De plus, ils n'exigent aucune connaissance sur la manière dont résoudre le problème puisqu'il est

seulement nécessaire de pouvoir évaluer la qualité d'une solution. Enfin, ils présentent à la fois des capacités de robustesse et d'adaptation, c'est à dire qu'ils ont la capacité d'évoluer dans le but d'une recherche satisfaisante mais sont aussi capable d'être attirés par des solutions et d'y rester.

- L'apprentissage par *réseau de neurone* qui consiste à ajuster les valeurs des poids synaptiques du réseau pour que la réponse du réseau soit celle désiré. L'algorithme d'apprentissage connexionniste le plus courant est l'algorithme rétro propagation du gradient [21]. L'avantage de cette architecture est quelle est très efficace pour certaines applications. De plus, elle incorpore le mécanisme de comportement apprenant. L'inconvénient majeur est quelle constitue une « boite noire ». C'est à dire que le mécanisme interne est caché et complexe, on ne comprend pas ce qu'on apprend.

- Le méta-apprentissage (MA) qui est un concept qui a fait son apparition dans les années 80 [22]. Il peut être décrit comme la capacité d'un système d'apprendre à apprendre. Par définition, le but du MA est de construire un système d'apprentissage auto-adaptatif qui s'adapte constamment à son environnement. Ce système améliore dynamiquement son fonctionnement en adaptant sa stratégie d'apprentissage en accord avec les expériences acquises, ou les méta-connaissances accumulées. L'un des buts du méta-apprentissage est, donc, de faire apprendre à l'apprenant comment régler lui-même les différents paramètres régissant son apprentissage. L'une des approches envisagées consiste à transférer la connaissance acquise en apprenant une tâche vers un système en train d'apprendre une autre tâche. Une autre approche, qui se veut encore plus générale, est proposée par [23] qui propose un algorithme de méta-apprentissage possédant le potentiel d'apprendre à apprendre à apprendre à etc. Dans son formalisme, le système apprenant évalue les performances de différentes stratégies d'apprentissage pour raisonner à leur sujet et choisir celles qui sont le mieux adaptées à la situation présente. A chaque évaluation des performances, l'apprenant peut garder sa stratégie actuelle, la modifier ou revenir à une stratégie antérieure qui s'avère, finalement, plus performante. Le processus d'évaluation est lui même une stratégie d'apprentissage qui peut alors être évaluée et modifiée.

5.2. Discussion sur l'apprentissage des SMA

Les algorithmes d'apprentissages ont certaines des caractéristiques plus ou moins intéressantes en fonction du problème à résoudre. Cependant, trois caractéristiques importantes se retrouvent et sont primordiales dans le cas d'apprentissage dans les SMA à savoir l'*environnement,* les *caractéristiques des agents* et la *communication.* En effet, l'environnement dans lequel évoluent les agents, est-il dynamique ou statique? Les agents ont-ils une connaissance partielle ou totale de leur environnement ? Les agents sont-ils de caractère statique ou dynamique ? Sont-ils indépendants les uns des autres ? Ont-ils les mêmes modèles de comportement ? De plus, les agents communiquent-ils pendant le processus d'apprentissage? Si oui, quelles informations sont échangées et quand sont-elles échangées.

6. Etat d'art sur l'apprentissage des systèmes multi-agents

Dans un environnement multi-agents, les agents peuvent apprendre grâce aux autres, et, d'un autre côté, ils peuvent aussi apprendre à propos des autres. Un autre côté important est l'apprentissage des interactions ou coordination, collaboration, communication, etc. entre les agents. Deux types d'apprentissage peuvent être distingués à savoir l'apprentissage des compétences et l'apprentissage de l'organisation. L'apprentissage des compétences, c'est-à-dire des connaissances sur le domaine, peut s'effectuer de manière isolée ou collective. Dans le premier cas, c'est un agent, qui, de sa propre initiative, crée de nouveaux corpus de connaissances. Dans le second cas, les agents d'un groupe échangent de la connaissance ou s'observent dans leurs activités respectives. Quant à l'apprentissage de l'organisation, il porte sur les liens d'interaction possibles entre agents pour leur permettre de mieux se coordonner lors de la résolution d'un problème donné. Il s'agit d'un apprentissage collectif qui consiste "à apprendre les interactions entre agents afin d'améliorer celles-ci" [24].

La grande majorité des travaux porte sur le premier type d'apprentissage, l'apprentissage des compétences, et les méthodes les plus couramment utilisées reposent sur les mécanismes d'apprentissage par renforcement.

Dans la méthode d'apprentissage proposé par Sian [25], aucun agent ne possède une information complète, donc les agents coopèrent pour posséder une information plus complète et bénéficier de l'expérience des autres agents. Le modèle présenté, appelé MALE (*Multi-Agent Learning Environment*) et écrit en PROLOG, a été appliqué au marché des matières premières [25]. Le système est constitué de trois agents qui ont des connaissances concernant différents pays : l'Afrique, l'Amérique et l'Asie. Le but des agents est de prédire les prix des matières premières en fonction de différents facteurs. Ces agents communiquent au moyen d'un tableau noir. Le mécanisme d'apprentissage est essentiellement basé sur l'interaction entre agents. Pour apprendre, ils échangent des opinions et des jugements de valeurs sur des hypothèses. A chaque hypothèse est associée une valeur de confiance basée sur le nombre d'instances utilisées pour la générer. L'apprentissage a lieu quand tous les agents ont réussi à trouver un consensus sur l'hypothèse considérée. Les trois phases du cycle d'apprentissage sont la génération d'hypothèses initiales par un des agents, :
- la génération d'hypothèses initiales par un des agents. Différents critères peuvent être définis pour le choix des hypothèses initiales à prendre en compte pour l'apprentissage, la coopération des différents agents pour arriver à des hypothèses acceptée par tous et l'intégration des informations ainsi apprises par les agents. La coopération des différents agents est vue en termes de représentation des connaissances, d'évaluation d'hypothèses et d'unification de toutes les opinions reçues. Quant à l'intégration des informations ainsi apprises par les agents, le problème de la révision des croyances se pose pour intégrer les nouvelles connaissances.

De leur coté, Sekaran et Sen [26], intéressés par le domaine coopératif où deux agents travaillent ensemble sur une tâche commune sans partage explicite de connaissances ou d'informations, Mahendra Sekaran et Sandip Sen étendent leur approche à un domaine non coopératif, où les agents ont des buts conflictuels. Leur modèle présenté dans [27] utilise l'algorithme du Q-learning, L'originalité de ce travail est qu'il n'y a ni échange de connaissance explicite entre les agents ni dépendance de relations. Cette méthode est appliquée au monde des blocs dans lequel deux agents

autonomes ayant différentes capacités et aucune connaissance sur le domaine, doivent pousser un bloc d'une position initiale à une position finale. A chaque étape, chaque agent applique une force suivant un certain angle et le bloc est déplacé. La session est terminée dès que le bloc atteint l'une des deux positions finales ou lorsqu'il est hors du champ de jeu. Chaque agent apprend, par l'intermédiaire de tentatives répétées, une politique optimale pour atteindre la position voulue. Les auteurs montrent que les agents peuvent utiliser des schémas d'apprentissage de renforcement pour atteindre leur but, qu'ils soient coopératifs ou conflictuels, et sans avoir de modèle l'un de l'autre. Dans le travail de Mataric [28], le SMA étudié est composé d'un groupe de 4 robots mobiles réalisant une tâche de fourragement. Chaque robot apprend individuellement à sélectionner le meilleur comportement pour chaque condition, de sorte à trouver et à ramener le plus de palets dans sa maison. A cet effet, quatre comportements sont fixé comme suit : droite-ligne (déplacement en avant en évitant les collisions), dispersion (respect d'une distance minimum entre les robots), repos et retour-maison. Apprendre à sélectionner des comportements est un problème d'apprentissage par renforcement car il est fondé sur une corrélation entre les comportements effectués et les résultats qu'il reçoit en retour. La stratégie d'apprentissage par renforcement utilise le Q-learning et est strictement individualiste, *ie*. les agents apprennent par eux-mêmes. Le mécanisme d'apprentissage est identique et distribué dans chacun des robots. Les événements suivants produisent des renforcements positifs immédiats : saisir un palet, déposer un palet à la maison et dormir à la maison. Les événements suivants produisent au contraire des renforcements négatifs immédiats : déposer un palet hors de la maison, dormir hors de la maison. Quant à Shaw [29], il a travaillé sur les SMA cognitifs qui communiquent constamment par envoi de messages. La méthode est basée sur deux processus: un processus d'offre de résolution de tâches et un processus de transformation génétique. Le processus d'offre est une extension du réseau contractuel de Smith et Davis [30] dans lequel les agents travaillent par partage de tâches en respectant le cycle suivant: décomposition d'une tâche en sous-tâches, allocation des sous-tâches aux

agents, résolution des sous-tâches par les agents choisis et intégration des solutions partielles. L'extension consiste à récompenser les agents sélectionnés pour résoudre une tâche au cours du processus de résolution. Cette récompense permet d'augmenter la "force" de l'agent. La force quantifie donc le service rendu par l'agent à la collectivité, elle reflète sa capacité à résoudre des tâches. Elle est utilisée ensuite pour déterminer quels sont les caractéristiques importantes des agents. Grâce au processus de détection des caractéristiques désirables, des algorithmes génétiques de mutation ou de croisement sont utilisés pour remplacer les agents "faibles" par des agents "forts". La performance globale du système est ainsi augmentée. Pour cela, chaque agent possède une représentation génétique de ses caractéristiques et de ses capacités. Cette méthode a été mise en œuvre dans le cadre d'un système de fabrication. Chaque agent est un intervenant au niveau de la fabrication et les chromosomes représentent tout ce que l'agent sait faire. Dans ce système, l'apprentissage concerne à la fois les connaissances des agents (puisque les agents qui vont être modifiés intègrent les caractéristiques des agents les plus sollicités) et à la fois l'organisation (puisque cette modification entraîne une réorganisation du système par le biais d'interactions futures). Pour sa part, Weiβ [31] a travaillé sur les SMA réactifs qui communiquent constamment avec leur environnement et entre eux par envoi de messages. Les agents connaissent typiquement seulement une fraction de leur environnement et l'information connue peut différer d'un agent à un autre. La méthode d'apprentissage distribuée préconisée vise une meilleure coordination entre les agents de la société. Pour cela, deux algorithmes d'apprentissage, ACE (*Action Estimation*) et AGE (*Action Group Estimation*) sont proposés. Le traitement réalisé par ces deux algorithmes peut être décrit de manière simplifiée comme l'exécution répétée du cycle suivant : la détermination de l'action, la compétition et la répartition des crédits. En effet, pour la détermination de l'action chaque agent détermine, en première étape, suivant ce qu'il connaît de l'état de l'environnement, l'ensemble des actions qu'il peut effectuer. Les agents concourent aussi pour le droit de devenir actif. Cette compétition englobe le calcul et l'annonce de l'offre ainsi que la sélection des actions qui sont

actuellement effectuées. Dans cette dernière étape de répartition des crédits, les agents attribuent un crédit à chacun des autres agents selon l'estimation de leurs actions. Les deux algorithmes ACE et AGE proposés différent par leur manière de mettre en œuvre la compétition. Dans la méthode ACE, les agents concourent pour pouvoir effectuer des actions individuelles. Chaque agent fait une offre pour chacune des actions qu'il peut effectuer et l'annonce à tous les autres agents qui sélectionnent alors l'offre la plus élevée parmi celles proposées. L'agent qui possède cette dernière est autorisé à exécuter l'action choisie. Tous les agents suppriment ensuite toutes les offres associées à des actions incompatibles avec l'action sélectionnée. Ce traitement est réitéré jusqu'à ce qu'il n'y ait plus d'action ayant une offre non nulle à sélectionner. La méthode AGE est très proche de la méthode ACE si ce n'est que les agents concourent pour effectuer des groupes d'actions et non plus des actions individuelles. Pour cela, un agent estime la pertinence d'un but ou d'une action non seulement sur la dépendance entre sa connaissance et l'état courant de l'environnement mais aussi sur la dépendance des contextes des activités possibles. Une telle séquence nécessite la connaissance des autres et la possession d'un même algorithme d'apprentissage notamment pour le calcul des offres. Il est essentiel que les agents soient rationnels et non égoïstes. Il ne faut pas que les agents insistent pour exécuter une action incompatible avec les actions précédemment exécutées. Le choix d'une offre parmi toutes celles proposées induit également un contrôle centralisé ce qui supprime l'autonomie de l'agent pour l'apprentissage. Ces deux algorithmes ont été étudiés expérimentalement dans le monde des blocs. Les agents doivent à partir d'une configuration initiale aboutir à une configuration finale. Chaque agent est responsable d'un cube. Tout en ayant une vue limitée de l'état de l'environnement, les agents parviennent à agencer les cubes pour atteindre la configuration désirée. Cette application montre que l'algorithme AGE fournit de meilleurs résultats que l'algorithme ACE.

D'autres travaux sur l'apprentissage des SMA ont été effectués comme celui de *Nagendra Prasad* [32] qui propose une méthode d'apprentissage sur l'organisation visant à prouver l'utilité de l'auto-organisation dans un

système composé d'agents ayant recours à des interactions complexes. Elle fournit aux agents un moyen de manipuler de manière efficace et sûre des tâches coopératives. Un rôle est décrit en tant que tâche ou ensemble de tâches à réaliser, en utilisant éventuellement plusieurs opérateurs, dans le but d'atteindre un objectif. Les rôles sont attribués par rapport à un modèle. Un agent n'a pas à tenir tous les rôles. Le but de l'apprentissage consiste, pour l'agent, à apprendre le rôle qu'il doit tenir. La méthode d'apprentissage proposée est appliquée au système L_TEAM [32].dans lequel chaque agent est dédié à la résolution d'un sous problème et à l'obtention d'une solution partielle. *Terabe et al* [33], quant à eux, proposent un modèle qu'ils appliquent au problème d'allocation de tâches. Le but organisationnel de l'allocation de tâches est d'obtenir une organisation permettant de minimiser les coûts d'exécution des tâches et les coûts de communication.

Dans le cas de l'apprentissage est utilisé pour apprendre à communiquer notamment l'apprentissage par renforcement, dans [34] les auteurs ont montré que lorsque la communication est coûteuse et qu'il est important pour les agents de déterminer s'ils doivent ou non envoyer un message à chaque prise de décision. La communication est donc vue comme une action et l'agent doit déterminer dans chaque état s'il doit ou non l'effectuer en utilisant des outils de planification. Dans l'article [35], les auteurs cherchent aussi à déterminer quand communiquer. Pour cela ils utilisent un algorithme de Q-Learning et le fait d'envoyer un message est assimilé à une action de coordination. À chaque prise de décision l'agent doit déterminer s'il utilise ses actions propres ou s'il doit se coordonner. En cas de coordination, il envoie un message et utilise les informations récupérées *via* cet outil pour déterminer la nouvelle action à effectuer. Ces travaux montrent que l'agent peut apprendre à déterminer quand envoyer ses actes de communication. Cependant les messages y sont déterminés à l'avance : l'agent connaît le contenu et le destinataire du message. Quant à l'article [36], les auteurs proposent un algorithme pour la communication entre agents dans le but d'apprendre quoi, à qui et quand communiquer. Afin de construire les actes de communication, ils ont utilisé une solution basée sur le pouvoir d'introspection des agents et leur capacité à raisonner sur leurs actions afin

que l'agent apprenant puisse construire les actes de communication dont il a besoin. Ces actes de communication ainsi créés sont alors considérés comme des actions afin de leur appliquer un algorithme d'apprentissage par renforcement.

7. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les concepts d'agent, d'environnement, d'interaction et de communications. Cela nous a permis de voir que chacun de ces concepts étaient nécessaires pour la conception d'un SMA. Nous avons ensuite exposé une des composantes importantes des SMAs, à savoir, leur apprentissage et les méthodes d'apprentissage, notamment l'apprentissage par renforcement qui est l'une des méthodes les plus utilisée dans la littérature et qui fait l'objectif de notre présent mémoire. La principale objectif de cet apprentissage est de permettre à l'agent, d'apprendre un comportement optimal sans connaitre les détails de l'environnement mais, en interagissant simplement, avec lui. Par ailleurs, nous avons montré que les domaines où les SMA sont appliqués sont nombreux. En ce qui concerne l'application que nous avons choisie pour, particulièrement, simuler l'apprentissage par renforcement des systèmes multi-agents, elle s'inscrit dans la cadre du problème de tournée de véhicules qui va fait l'objet du prochain chapitre.

Chapitre 2

Le problème de tournées de véhicules

1. Introduction

De nos jours, le transport terrestre incluant la livraison de marchandises à des magasins, l'ordonnancement de bus scolaires, la distribution de courriers ou de journaux etc. constitue un véritable problème aux entreprises modernes sur le plan contraintes spatiales, temporelles et économiques. Dans certains cas, la solution devient, très vite, extrêmement difficile à trouver. C'est ainsi que nombreux chercheurs se sont spécialisés dans ce domaine de transport à la recherche de solutions aux différents problèmes posés. Ce qui a conduit, d'ailleurs, à la dénomination théorique du problème sous forme de Problème de Tournées de Véhicules (PTV ou *VRP* en anglais) domaine dans lequel nous appliquerons les Systèmes Multi-Agents (SMA), thème de notre présent travail. Ainsi, dans ce chapitre nous allons présenter le VRP et ses différentes variantes et nous allons situer notre problème par rapport à ces variantes et énumérer les différentes méthodes de résolution utilisées. Nous présenterons, ensuite, quelques applications des SMA dans le problème du VRP et, notamment, le VRP avec

fenêtres de temps pour lequel nous nous sommes, particulièrement, intéressés.

2. Le Problème de Tournées de Véhicules

Le VRP peut être décrit comme un problème de conception d'itinéraires d'un ou plusieurs véhicules qui doivent partir d'un dépôt, visiter un ensemble de clients, collecter ou livrer des biens, et revenir au dépôt [37][38]. Plusieurs objectifs peuvent être considérés pour le VRP. Les plus typiques sont la minimisation du coût global de transport, la minimisation du nombre de véhicules et l'équilibrage des itinéraires. La minimisation du coût global de transport dépend de la distance globale parcourue ou temps global de voyage et des coûts liés à l'utilisation des véhicules et aux personnels conducteurs. L'équilibrage des itinéraires est relatif au temps de voyage et à la charge de véhicules.
Dans le VRP, les problèmes prennent plusieurs et différentes formes dont les principales sont les suivantes:

2.1 Tournées de véhicules avec contraintes de capacité

Le problème de tournées de véhicules avec contraintes de capacité CVRP (*Capacitated Vehicle Routing Problem* en anglais) consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité finie [39]. Ce véhicule commence d'un dépôt central unique et termine sa tournée au niveau de ce dépôt.

2.2 Tournées de véhicules dynamiques

Contrairement au VRP classique où toutes les demandes des clients et les informations sont connues à l'heure de la planification des itinéraires, le problème de tournées de véhicules dynamique ou DVRP (*Dynamic Vehicle Routing Problem* en anglais) illustre mieux la réalité par les apparitions de nouveaux clients, de nouvelles demandes ou de pannes des véhicules de transport [40][41]. Dans ce genre de situation, le décideur doit réorganiser,

en un temps acceptable, la planification des itinéraires de sa flotte de véhicules pour répondre aux nouvelles demandes urgentes.

2.3 Tournées de véhicules avec livraison et collecte

Le problème de tournées de véhicules avec livraison et collecte ou VRPB (*Vehicle Routing Problem with Backhauls*) est une prolongation du VRP classique. Il comporte des clients auxquels des produits doivent être livrés et des fournisseurs dont les marchandises doivent être amenées au dépôt [42]. Dans une tournée, les livraisons doivent être effectuées avant les collectes et la capacité du véhicule doit être respectée en tout point. Notons qu'une tournée ne peut être constituée que de collectes. De plus tous les véhicules disponibles doivent être utilisés.

2.4 Tournée de véhicules périodique

Dans le problème de tournées de véhicules périodique ou PVRP (*Periodic Vehicle Routing Problem*), chaque client est servi un certain nombre de fois [37]. Ce nombre constitue une donnée du modèle. L'objectif est de minimiser les coûts de transport tout en servant chaque client autant de fois qu'il est nécessaire. À chaque client correspond un ensemble de séquences de livraisons selon les jours de livraison. Dans ce type de problème, on distingue l'opération d'affectation des séquences de livraisons aux clients et la résolution du problème de tournées « classique » par jour.

2.5 Tournées de véhicules avec collecte et livraison

Le problème de collecte et de livraison ou VRPPD (*Vehicle Routing Problem with Pickup and Delivery*) a les mêmes propriétés que le VRP, avec, en plus, le fait que chaque client introduit deux positions géographiquement différentes [43]. La première, pour le ramassage du produit et la deuxième pour la livraison de ce dernier. Ceci va directement induire une contrainte de précédence à ajouter au problème du VRP classique sachant que, dans une tournée, chaque opération de livraison doit être précédée par l'opération de ramassage respective.

2.6 Tournées de véhicules multi-dépôts

Dans ce type de problème appelé Multi-Depot Vehicle Routing Problem en anglais (MDVRP), plusieurs dépôts sont géographiquement distribués. Dans cette tournée, un véhicule part et revient au même dépôt initial [44].

2.7 Problème du voyageur de commerce

Le problème du voyageur de commerce ou TSP (*Travelling Salesman Problem*) consiste à trouver le chemin le plus court pour passer par l'ensemble des villes données une fois et une seule et revenir à la ville de départ [45]. C'est donc un cas particulier du VRP sans contrainte de capacité et avec un seul véhicule.

2.8 Tournées avec des véhicules à flotte variable VRPHF

La seule différence entre un problème de tournées avec des véhicules à flotte variable VRPHF (*Vehicle Routing Problem with Heterogeneous Fleet)* et un VRP est que la flotte de véhicules est hétérogène, autrement dit, les véhicules sont de types différents mais différenciés par leurs coûts de transport, leurs capacités de transport, leurs vitesses, leurs tailles, etc. [46].

2.9 Tournées de véhicules ouvert

La différence entre un problème de tournées de véhicules ouvert OVRP (*Open Vehicle Routing Problem)* et un VRP est que, dans le premier cas, les véhicules ne sont pas tenus de retourner au dépôt. Dans le cas où ils le sont, ils parcourent le même chemin en revisitant les clients qui leurs sont affectés dans l'ordre inverse [47].

2.10 Tournées de véhicules classiques

A la différence des problèmes de tournées de véhicules classiques, le problème de tournées de véhicules classiques ou SDVRP (*Split Delivery Vehicle Routing Problem*) accepte qu'une demande soit satisfaite par plusieurs tournées. Pour cela, un client peut être visité plusieurs fois si cela est nécessaire [48]. Dans ce type de problème et contrairement aux autres, la demande d'un ou plusieurs clients peut excéder la capacité du véhicule [48].

2.11 Tournées de véhicules avec fenêtres de temps

Le problème de tournées de véhicules avec fenêtres de temps ou VRPTW (*Vehicle Routing Problem with Time Windows*) constitue une généralisation du VRP du moment que, en plus, des contraintes temporelles sur le service sont introduites [38]. Chaque client dispose d'une fenêtre de temps à l'intérieur de laquelle il peut être desservi. Le dépôt central possède également une fenêtre de temps que l'on désigne couramment comme l'Horizon de Service et son rôle est de fixer une plage horaire durant laquelle les véhicules peuvent réaliser leur tournée. Ce type de problème temporel peut intervenir dans la modélisation de nombreux services. On peut notamment citer le service postal, le service bancaire, les coopératives agricoles, le ramassage scolaire etc. Dans le cas du service postal, il s'agit d'établir des tournées pour les facteurs de manière à distribuer le courrier le plus efficacement possible et réaliser, aussi, des tournées pour effectuer le ramassage du courrier dans les boites aux lettres. Dans les coopératives agricoles, il s'agit de mettre en place des tournées de véhicules pour collecter la production des différents exploitants agricoles tels que le lait, les produits céréaliers, les produits maraichers, les animaux etc. Dans le service bancaire, on détermine des routes pour les convoyeurs de fond de manière à réaliser la collecte des recettes de commerçantes et des grandes surfaces. Pour le ramassage scolaire, on établit les routages de bus pour amener les écoliers des points d'arrêt vers l'˝ecole le matin et, inversement, le soir.

Ces exemples montrent que le problème de tournées de véhicules avec fenêtres de temps intéressent plusieurs d activités dans la vie quotidienne. On constate au passage quelques aspects importants, notamment le fait que l'on travaille avec un type de ressource unique. Les ressources sont acheminées depuis une source vers une destination assimilée au dépôt central, moyennant éventuellement quelques manipulations du modèle. On impose souvent un dépôt central, donc unique, mais le modèle peut être généralisé au cas multi-dépôts.

Dans ce même type de problèmes de tournées avec fenêtres de temps, on peut encore citer le cas où la dynamique intervient ou DVRPTW (*Dynamic*

Vehicle Routing Problem with Time Windows) [37] ou le cas de tournées de véhicules avec livraison et collecte et fenêtres temps ou PDPTW (Pickup and Delivery Problem with Time Windows) qui est une variante du VRPTW [37]. En effet, en plus de l'existence des contraintes de fenêtres de temps, ce problème possède un ensemble de clients et un ensemble de fournisseurs tel que, à chacun de ces clients, correspond un et un seul fournisseur. Les véhicules ne doivent alors passer par un client qu'après avoir visité son fournisseur.

3. Les méthodes de résolution du VRPTW

Plusieurs méthodes de résolution applicables au problème de tournées de véhicules ont été proposées en visant, de plus en plus, à obtenir de meilleures solutions en des temps de calcul réduits. Comme dans d'autres domaines, des tentatives d'obtenir des solutions exactes au problème de tournées de véhicules ont permis de résoudre uniquement des problèmes de taille limitée, ce qui a conduit à l'élaboration d'heuristiques et, par la suite, de métaheuristiques pour résoudre des problèmes de plus grandes tailles. Dans ce qui suit nous donnerons une courte description des méthodes de résolution qui apparaissent fréquemment dans la littérature. Notre étude concernera surtout, le problème de tournées de véhicules avec fenêtres de temps ou VRPTW (*Vehicle Routing Problem with Time Windows*), cas général du VRP car c'est de type de problème qu'il s'agit dans ce mémoire.

3.1. Les méthodes exactes

Les approches basées sur des méthodes exactes ont tardé à apparaître à cause surtout de la difficulté à trouver un modèle suffisamment performant pour exprimer le VRPTW puisque les contraintes de capacité et de fenêtres temps reposent sur un mécanisme récursif *ie*. L'application des contraintes sur un client dépend du résultat sur le client précédent. Les différentes approches proposées dans ce cas peuvent être regroupées en les quatre grandes catégories que sont la programmation dynamique, la génération de colonnes, la décomposition lagrangienne et la relaxation

lagrangienne. En programmation dynamique, Kolen et *al* [49] sont à l'origine du premier algorithme d'optimisation basé sur un Branch and Bound pour le VRPTW, un des travaux les plus anciens sur le VRPTW. La méthode s'appuie sur le calcul de bornes inférieures par une programmation dynamique et une relaxation de l'espace d'état. Les décisions de branchement portent sur l'affectation des clients aux tournées. Cette approche a permis, à l'époque, de résoudre d'une manière optimale des problèmes comptant jusqu'à 15 clients. En génération de colonnes, Desrochers et *al* [50] proposent l'application de la technique de génération de colonnes sur le VRPTW exprimé comme problème de partitionnement, et de résoudre ce dernier par la méthode de génération de colonnes en assimilant une tournée à une colonne. Cette approche, très performante, autorise la résolution d'instances comprenant jusqu'`à 100 clients. L'approche décomposition lagrangienne repose sur un schéma de partage des variables. Ceci permet d'obtenir des bornes inférieures pour la solution que l'on peut ensuite intégrer dans un mécanisme de Branch and Bound. Jornsten et *al* [51], Madsen [52, 53], Halse [54] et Fisher [55] proposent chacun un moyen différent de réaliser la décomposition lagrangienne. Des instances de 100 clients ont ainsi pu être résolues. La relaxation lagrangienne a été utilisée par Kohl et Madsen [56] en proposant une relaxation lagrangienne d'après un modèle du plus court chemin. Ce dernier présente une modélisation originale du VRPTW dont la relaxation des contraintes aboutit à un problème de *k-trees*. Cette approche permet de résoudre quelques problèmes à 100 clients. Plus récemment, Larsen et al. [57] ont implantés une approche combinant la relaxation lagrangienne utilisée par [56] avec la décomposition généralisée de Dantzig-Wolfe proposée par Desrochers et al. [50]. Dans une première phase, la relaxation lagrangienne est utilisée afin de tirer avantage de la convergence rapide, puis, dans une deuxième phase, les colonnes obtenues en phase une, sont utilisées pour initialiser la procédure. Jusqu'à présent, la plupart des méthodes proposées sont capables de résoudre des instances à 100 clients. Cependant, il apparait que l'approche de Desrochers [50] demeure la plus performante.

3.2 Les méthodes approchées

On rappelle que les heuristiques classiques ont pour objectif de produire des solutions acceptables dans des temps raisonnables. Parmi ces heuristiques, on trouve les heuristiques de construction de tournées, les heuristiques d'amélioration de tournées, les heuristiques composites, les heuristiques basées sur un principe exact et les métaheuristiques.

- *Heuristiques de construction de tournées*

Jusqu'en 1987, les méthodes proposées dans ce cadre comme, par exemple, la méthode de Clark et Wright (1964), ne fonctionne pas de façon satisfaisante vue la présence de la fenêtre temps dans le VRPTW. En 1987, Solomon [58] a introduit une nouvelle heuristique de construction, mieux adaptée aux problèmes avec fenêtre de temps reposant sur l'insertion séquentielle des clients. Potvin et Rousseau [59], puis Russell [60] proposent des méthodes de construction de tournées en parallèle. A l'heure actuelle, la quasi-totalité des méthodes de construction pour le VRPTW reposent sur les heuristiques dites de *meilleure insertion.*

- *Heuristiques d'amélioration de tournées*

Ces méthodes partent d'une solution initiale obtenue par une heuristique du type précédent et s'améliorent progressivement. Les différents opérateurs proposés sont presque tous basés sur le principe du voisinage K-opt [61], tout en intégrant les contraintes de fenêtre de temps et de capacité. On peut citer en particulier les travaux de Baker et Schaffer [62], Or [63], Potvin et Rousseau [64], Solomon et *al* [65] ou encore Thompson et Psaraftis [66].

- *Heuristiques composites*

À la différence des heuristiques d'amélioration de tournées qui nécessitent la connaissance de la solution initiale, les heuristiques composites sont auto-démarrantes. Elles combinent une phase de construction et une phase d'amélioration des tournées. Elles sont d'inspiration relativement récente bien qu'on puisse remonter aux travaux de

Gillett et Miller [67]. On pourra se référer aux travaux de Derigs et Grabenbauer [68], Kontoravdis et Bard [69], ou encore Russel [70].

- *Heuristiques basées sur un principe exact*

Elles forment une classe à part dans la mesure où elles sont déduites de méthodes exactes. Elles reprennent en fait un schéma d'optimisation exact, avec quelques aménagements, en particulier au niveau des critères décisionnels. Ainsi Koskosidis [71] utilise des critères heuristiques pour réaliser l'assignation des clients aux tournées. Après décomposition, ces dernières sont ensuite obtenues via une méthode exacte de routage.

- *Les métaheuristiques*

Les métaheuristiques tentent d'identifier les régions les plus prometteuses de l'espace solution afin de mieux les explorer. La qualité des solutions proposées par les métaheuristiques est supérieure à celles des heuristiques. Cependant, le temps de calcul requis est plus élevé. Différentes métaheuristiques sont utilisées. Ainsi, la recherche taboue est une méthode qui a fait l'objet de plusieurs travaux parmi lesquels on peut citer la méthode de diversification et d'intensification probabiliste de Rochat et Taillard[72] qui consiste à choisir de façon aléatoire les tournées utilisées pour construire une nouvelle solution, mais en pondérant la probabilité associée à chaque tournée par un facteur lié a l'efficacité de la solution dont elle a été extraite. Chiang et Russel [73], eux, décrivent un mécanisme qui encapsule la recherche tabou, appelé *Reactive Tabu Search*. Il s'agit de faire varier la taille de la liste taboue en fonction de la diversité des solutions visités. Le voisinage est construit à partir de deux opérations : échanges *2-opt* entre tournées différentes et déplacement d'un client d'une tournée à une autre. Semet et Taillard [74] ont également utilisé un algorithme de recherche tabou, mais sur un problème légèrement différent, incluant quelques contraintes supplémentaires. Une autre méthode hybride est proposée par Gambardella et *al*. [75], en se basant sur deux colonies de fourmis avec des phéromones (donc la mémoire) distinctes. La première colonie minimise le nombre de véhicule utilisés pendant que la seconde minimise la distance

totale. Les phéromones restent différents d'une colonie à l'autre mais avec des échanges d'informations. Par ailleurs, les méthodes évolutionnaires ont été récemment introduites. Notamment, il est difficile de concevoir un operateur de croisement qui produise une solution satisfaisant toutes les contraintes de fenêtres temps. Depuis 2001, des algorithmes génétiques efficaces pour le *VRPTW* ont vu le jour. Berger et Barkaoui [76] ont proposé en 2004 un algorithme génétique hybride parallélisé. L'opération de croisement consiste à combiner un sous-ensemble des tournées d'un individu avec un ensemble de demandes d'un autre individu ; ces demandes sont choisi sur un critère de distance pas rapport aux tournées héritées du premier individu.les demandes non planifiées après croisement sont réintroduites via une heuristique constructive de plus proche voisin. L'operateur de mutation utilisé consiste en fait en des recherches locales dans six voisinages différents. L'aspect parallèle sert à diviser les taches de recombinaison et mutation sur plusieurs processeurs. Par ailleurs, un algorithme génétique « pur » est décrit dans l'article de Tavares et *al.* [77]. Il s'agit d'une adaptation de l'algorithme GVR (Genetic Vehicle Representation) aux fenêtres de temps. La difficulté à produire, par le croisement, une solution satisfaisant les fenêtres temps est réglée en créant une nouvelle tournée si une fenêtre temps n'est pas respectée. Les méthodes les plus récentes essayent de combiner plusieurs de ces métaheuristiques, comme par exemple Homberger et Gehring [78] qui proposent une méthode en deux phases : minimiser le nombre de véhicules utilisés puis minimiser la distance totale parcourue avec une recherche tabou. D'autres méthodes hybrides pour le VRPTW de grande taille ont vu le jour, comme par exemple la métaheuristique de Bouthillier et Crainic [79] ou la méthode de Mester ou Braysy appelée *Active Guided Evolution Strategies* [80]. Plus récemment, Olli Braysy et Gendreau [81] arrivent à la conclusion que les méthodes actuellement les plus efficaces sont les algorithmes évolutionnaires, hybridés avec des méthodes de recherche locale, comme par exemple la recherche tabou.

4. Applications des systèmes multi-agents

Plusieurs approches multi-agents ont été adoptées dans la littérature pour le problème de tournées de véhicule et ses variantes. Dans ce qui suit, nous citons de nombreux travaux d'applications des systèmes multi-agents (SMA) pour le VRP et le VRPTW. Ainsi, dans Thangiah et *al.* [82] et dans Kohout et Erol [83], les auteurs proposent une architecture multi-agents pour résoudre un VRP et un VRP à dépôts multiples pour le premier et un problème de transport à la demande pour le second. Le principe est le même, celui de distribuer une heuristique d'insertion suivi d'une étape de post optimisation. Dans Thangiah et *al.* [82], les clients sont traités séquentiellement. Diffusés à tous les véhicules, ces derniers proposent des offres et le meilleur est retenu. Dans la seconde étape, les véhicules échangent des clients afin d'améliorer leurs solutions, chaque véhicule connaissant tous les autres agents véhicule du système. Les véhicules s'exécutant en parallèle, les auteurs envisagent d'appliquer des heuristiques et des méta-heuristiques différentes pour chaque véhicule, sans changer d'architecture. Quant à Kohout et Erol [83], ils ont proposé le système *In-Time* qui est réalisé dans le cadre d'une application réelle d'ordonnancement de véhicules d'une société de transport en aéroport. Le système est composé d'agents Client et d'agents Véhicule. L'agent client s'annonce et tous les agents Véhicule calculent son coût d'insertion dans leurs itinéraires et renvoient des offres à l'agent Client. Ce dernier sélectionne l'offre la moins chère. Le mécanisme proposé par les auteurs représente une recherche locale distribuée. En effet, ils permettent à un client de demander d'une manière stochastique d'annuler son ordonnancement actuel et de se ré-annoncer au système dans l'objectif d'avoir un meilleur marché. Les travaux de B. Chaib-Draa et *al.* reflètent les efforts réalisés pendant plusieurs années dans le domaine du trafic aérien [84][85]. Ces travaux ont été poursuivis dans le domaine du transport urbain. Une approche multi-agent a été utilisée pour la résolution du problème de planification et de négociation dans les systèmes de transport de marchandises par des camions [86]. Dans cette approche multi-agent, les camions conducteurs sont considérés comme

des agents intelligents, autonomes et rationnels. Ces agents négocient leur proposition, via leur compagnie de transport (modélisation hiérarchique), en effectuant des achats et des ventes en vue d'atteindre un compromis qui minimise le coût du plan commun. Pour améliorer la performance de leur DMA, ils associent à ce commerce simulé, des heuristiques de recherche permettant d'améliorer, entre autres, la qualité et le temps d'obtention de la solution. Ainsi, ils utilisent les algorithmes génétiques pour la négociation entre les agents afin d'évaluer les plans les moins coûteux pour un agent (camion-conducteur). De leur coté, Fischer et *al.* [87] ont présenté MARS qui modélise un ordonnancement coopératif dans une compagnie maritime sous la forme d'un SMA. La solution du problème de l'ordonnancement global émerge de la prise de décision locale et des stratégies de résolution de problèmes. Le système utilise une extension du Contract Net Protocol (CNP) [88] et montre qu'il peut être utilisé pour obtenir de bonnes solutions initiales aux problèmes complexes d'allocation de ressources. Le SMA profite d'une structuration organisationnelle *a priori*, puisque chaque véhicule est associé à une société particulière et ne peut traiter que les clients de cette société. Kefi et Ghdédira [89] proposent un modèle multi-agent pour la résolution du problème *VRPTW*. L'objectif est de déterminer un ensemble de tournées de véhicules satisfaisant différents clients, respectant des contraintes spatiotemporelles prédéfinies et ayant un coût total minimal. Dans ce modèle seul des agents clients coordonnent, coopèrent et négocient en formant des coalitions afin de résoudre au mieux ce problème. M Zargayouna et *al.* [90], quant à eux, proposent une approche multi-agents pour le problème de tournées de véhicule avec des fenêtres de temps *(VRPTW)*. Ils traitent la version dynamique du problème où tous les clients ne sont pas connus avant le début du processus de solution. Ils ont concentré leur présentation sur des agents de véhicule et des agents de client, et sur leur enchaînement dynamique. Ils comptent sur le modèle de *Property-Based Coordination*, permettant de limiter l'interaction aux seuls agents qui pourraient atteindre une solution. L'interaction globale suit le protocole *Contract Net.* Pour ce qui concerne le processus de calcul, ils proposent une

nouvelle heuristique, basée sur la mesure du champ de la perception des agents au-dessus de leur environnement, ce qui favorise les futures possibilités d'insertion au lieu de l'optimisation immédiate de coût. Zheng-Gang Dan et *al.* [91] présentent un modèle basé sur le protocole *Contract-Net.* Dans l'article [92], les auteurs ont proposé un modèle d'auto-organisation orienté-agent pour le VRPTW dynamique se fondant sur les zones d'action des agents. Ces dernières reflètent la couverture spatio-temporelle des agents véhicule de l'environnement. Ils ont utilisé la variation de ces zones d'action comme une nouvelle métrique afin de réduire le comportement myope des métriques traditionnelles. En optimisant la couverture spatio-temporelle de l'environnement par les agents Véhicule, notre modèle permet au SMA de s'auto-organiser en affichant une répartition spatio-temporelle équilibrée de ses agents Véhicule, et de diminuer ainsi le nombre de véhicules mobilisés pour la desserte des clients.

6. Conclusion

Dans ce chapitre nous avons présenté le problème de tournée de véhicules et ses variantes et présenté un état de l'art par rapport à l'application relative à ce présent mémoire et qui constitue un problème d'optimisation de plan de transport de marchandises considéré comme un VRPTW à multiples dépôts. Cet état a montré que des spécialistes appliquent différents concepts théoriques pour résoudre ce type de problèmes dont le cahier des charges varie lui-même d'une application à une autre. Le problème étant d'optimisation et vue sa complexité, d'autres concepts théoriques tels les métaheuristiques et les SMAs sont appliqués pour tenter de trouver la solution qua-optimale. Intéressés par la résolution d'un problème réel de tournée de véhicules, nous nous sommes investis dans les SMAs en vue d'y apporter la solution en respectant les différentes contraintes posées. Ceci fait l'objet du chapitre suivant.

Chapitre 3

Modélisation et résolution du problème posé

1. Introduction

Dans ce chapitre, il s'agit du problème de tournées de véhicules avec fenêtres de temps que nous nous sommes proposé de résoudre. Pour modéliser le problème, nous avons utilisé les Systèmes Multi-Agents (SMAs) basés sur la coopération, la coordination et l'apprentissage des différents agents. Pour la conception du SMA, l'approche *AALAADIN* a été adoptée et, pour représenter les comportements et interactions des agents, les diagrammes d'AUML (Agent Unified Modeling Language) sont utilisés. Notre démarche commence par le découpage du problème en sous-problèmes pour déduire une architecture globale. Ensuite, nous procédons à la spécification des agents de notre système, leurs comportements et les interactions qui existent entre eux. Le Q-Learning, une des méthodes d'apprentissage des SMAa été utilisée pour trouver les meilleurs centres à insérer dans les itinéraires d'un véhicule allant d'une ville de départ pour servir une

destination donnée en respectant les contraintes. Le nombre de centres varie en cours de l'apprentissage, ce qui nous donne la dynamique des actions du Q-Learning.

Ainsi, en section 2, nous présentons le problème posé avec les différentes contraintes. Le découpage du problème en sous-problèmes fait l'objet de la section 3. En section 4, il s'agit de la modélisation du problème par des agents.

2. Présentation du problème

Notons que problème de tournées de véhicules que nous proposons de résoudre est avec fenêtres de temps ou VRPTW (*Vehicle Routing Problem with Time Windows*). Il s'agit particulièrement d'optimiser un plan de transport de marchandises et qui consiste à générer un plan de transport avec le minimum de véhicules pour transiter un ensemble de flux entre des agences de départ et des agences d'arrivée. Les flux transitent (ou non) par un ou plusieurs centres ou hubs tout en respectant un certain nombre de contraintes à savoir:

- tout le flux doit être acheminé en vingt quatre heures,
- les agences ont une heure d'ouverture et une heure de fermeture,
- le nombre maximum de transits dans un hub est limité à 3,
- la capacité des véhicules est limitée,
- le véhicule retourne à son agence de départ après avoir servi sa destination.

Ainsi, ce problème peut être considéré comme une généralisation du VRP où les agences de départ peuvent être considérées comme des dépôts et avec, en plus, le fait qu'un véhicule peut être amené à faire plusieurs tournées pour servir une destination donnée si la quantité à transmettre dépasse la capacité du véhicule.

L'itinéraire compte trois types d'agences à savoir, l'agence de départ et d'arrivée, l'agence d'arrivée et l'agence hub.

Le rôle de chaque agence de départ est de servir sa destination tout en essayant d'utiliser un minimum de véhicules, d'éviter les véhicules à moitié vide et de retourner à leurs départs en respectant les contraintes. Les flux à transiter sont représentés sous formes d'une matrice dont les lignes et les colonnes représentent les agences. Le tableau.1 présente un exemple des flux à transiter entre les agences (R,T,A,B,C,D,E) avec (A,B,C,D,E) comme agences de départ et d'arrivée au même temps et R et T comme agences d'arrivée seulement. L'intersection des agences des lignes avec les agences des colonnes nous donne les flux. Ainsi, à l'intersection de l'agence de la ligne A avec l'agence de la colonne C par exemple, on trouve 489, cela signifie que la quantité de marchandise ou flux à transiter entre l'agence A et l'agence C est de 489.

Tableau.1.Exemple de flux à transiter entre agences

	R	**T**	**A**	**B**	**C**	**D**	**E**
A	230	100	0	300	489	450	1246
B	90	600	234	0	970	678	2450
C	145	0	2045	1900	0	120	120
D	300	445	890	5355	680	0	670
E	609	280	1100	55	600	356	0

3. Découpage du problème en sous-problèmes :

Pour résoudre le problème avec les systèmes multi-agents, nous avons opté à la division du problème en deux sous problèmes et cela, en subdivisant la matrice flux en la somme de deux matrice M_1 et M_2 de sorte que *Matrice flux*=M_1+M_2 où M_1 est la matrice flux$_1$ dont la quantité *mod* (capacité camion dans notre cas 100)=0 et M_2**,** la matrice flux$_2$ dont la quantité est inférieure à 100. Si on prend la matrice flux de l'exemple du tableau 1 précédent, la matrice M_1 sera celle du tableau 2 et la matrice M2 celle du tableau 3.

Tableau 2. Exemple de flux M_1 calculé du flux de la table 1

	R	**T**	**A**	**B**	**C**	**D**	**E**
A	200	100	0	300	400	400	1200
B	0	600	200	0	900	600	2400
C	100	0	2000	1900	0	100	100
D	300	400	800	5300	600	0	600
E	600	200	1100	0	600	300	0

Tableau 3. .Exemple de flux M_2 calculé du flux de la table 1

	R	**T**	**A**	**B**	**C**	**D**	**E**
A	30	0	0	0	89	50	46
B	90	0	34	0	70	78	50
C	45	0	45	0	0	20	20
D	0	45	90	55	80	0	70
E	9	80	0	55	0	56	0

Le premier sous-problème consiste à faire transiter le flux de la matrice M_1. Dans ce sous-problème, les véhicules sont remplis, donc transportent directement la marchandise de l'agence de départ vers l'agence d'arrivée. Néanmoins, il reste juste à définir, pour chaque agence de départ, le nombre de véhicules qu'elle utilise pour transporter sa marchandise et, pour chaque véhicule, le nombre d'allers et retours possibles en respectant les contraintes, et tester si l'agence de d'arrivée est un hub pour que le véhicule prenne de la marchandise lors de son retour. Le deuxième sous-problème permet de transiter le flux de la matrice M_2. Dans ce sous-problème, les véhicules ne sont pas remplis, donc ils peuvent, en allant vers leurs destinations, passer par les hubs et prendre leurs marchandises afin de servir la même destination. Le problème qui se pose est quels hubs choisir, sachant qu'on ne peut pas en choisir plus de trois (d'après les contraintes du problème) pour une destination donnée. C'est dans la résolution de ce sous-problème qu'une des méthodes d'apprentissage des SMA qui est le Q-learning est utilisée pour nous permettre de choisir les meilleurs hubs à insérer dans les itinéraires d'un véhicule allant d'une ville de départ pour

servir une destination donnée tout en respectant les contraintes. Le nombre de hubs varie en cours de l'apprentissage, ce qui nous donne la dynamique des actions du Q-Learning. Nous rappelons que la méthode du *Q-Learning* est une méthode d'apprentissage par renforcement des SMA basée sur la fonction de *Q-valeur* et converge vers la *Q-valeur optimale*. Cet algorithme fonctionne par interaction avec l'environnement de la manière suivante (fig.1)

Algorithme1 Q-learning.
Initialiser :
Q = Fonction de valeur arbitraire
For** all épisodes **do
- *Initialiser s*
- ***Repeat***
  - *Choisir a à partir de s en utilisant la politique π dérivée de Q*
  - *Faire l'action a, observer r et le prochain état s'*
  - *Mettre à jours la fonction QValeurs :*
    $Q(s, a) \leftarrow (1 - a)Q(s, a) + a[r + \gamma \max_{a' \in A} Q(s', a')]$
  - $s \leftarrow s'$
- ***Until** s soit terminal*

End for

Fig.1. Algorithme du Q-Learning

A chaque étape, dans l'état courant *s*, l'agent choisit une action *a* en fonction d'une politique *π* dérivée des valeurs courantes de *Q*. Il exécute l'action *a*, reçoit la récompense *r* et observe l'état suivant *s'*. Ensuite, il met à jour les valeurs de *Q(s,a)* selon l'équation(1) :

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left[r + \gamma \max_{a' \in A} Q(s', a') \right] \quad (1)$$

où α est le taux d'apprentissage et γ le facteur d'escompte.

Selon cette formule, la mise à jour consiste à enlever une partie de l'ancienne valeur *Q(s,a)* et à ajouter une partie de la nouvelle estimation de la valeur avec la récompense immédiate et la valeur de l'état suivant. Pour illustrer le fonctionnement général du *Q-Learning*, une étape du cycle d'apprentissage est illustrée en figure 2. L'algorithme conserve donc en mémoire la table de Q-Valeurs *Q(s,a)*, le dernier état s_t du système et la dernière action effectuée a_t, car il les réutilise dans les étapes (1) et (3).

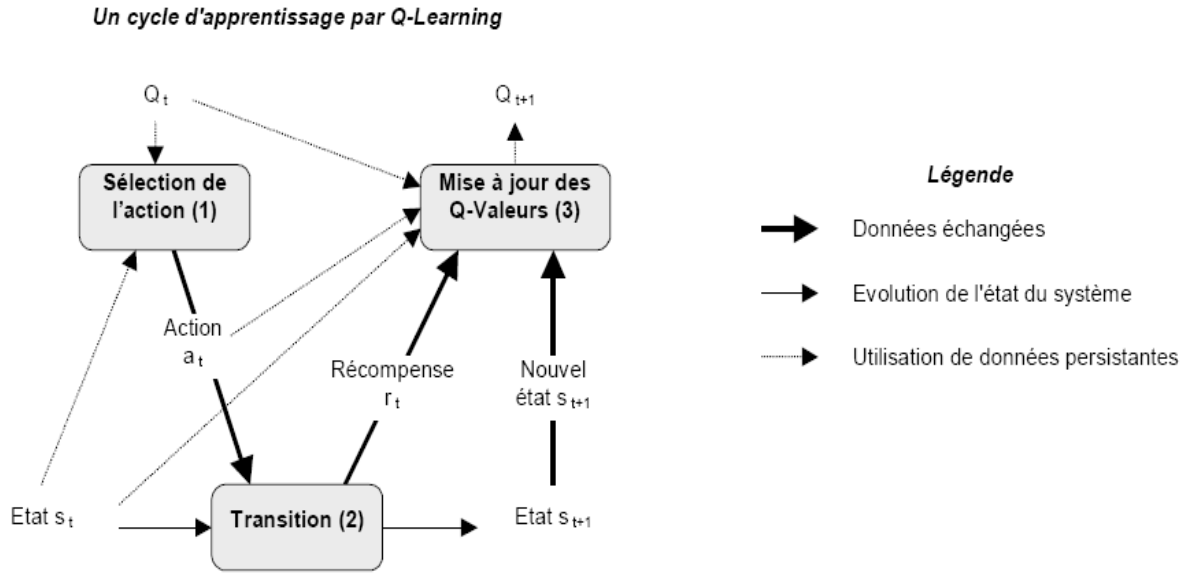

Fig.2. Etape pour l'algorithme du Q-Learning [93]

Il a été prouvé que l'algorithme Q-learning converge presque sûrement vers la *Q-valeur* optimale Q^*[94] à condition que la suite α_t correspondant au taux d'apprentissage qui diminue au cours du temps, soit :

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ et } \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

Pour apprendre la politique optimale rapidement, la manière de choisir l'action à chaque étape est très importante. Ce problème est connu sous le nom de dilemme *exploration/exploitation* et peut s'énoncer ainsi : est-ce que l'agent doit choisir une action qu'il a déjà effectuée et qui donne de bonnes récompenses espérées ou bien est ce qu'il doit explorer de nouvelles actions pour voir s'il n'est pas possible d'avoir des meilleures récompenses. Il existe plusieurs techniques pour résoudre ce dilemme de manière classique. On peut notamment citer la technique gloutonne, la technique ξ-gloutonne, la technique Softmax et la technique Boltzmann. La technique gloutonne consiste à toujours choisir l'action estimée comme la meilleure alors que la technique ξ-gloutonne consiste à choisir l'action gloutonne avec une probabilité ξ et à choisir une action au hasard avec une probabilité 1- ξ. La technique Softmax consiste à choisir une action en fonction de sa qualité relativement à la qualité des autres actions dans le même état, soit en

associant, par exemple, à chaque action la probabilité d'être émise selon l'équation(2):

$$\Pr[a_t, s_t] = \frac{Q(s_t, a_t)}{\sum_{a \in A(s_t)} Q(s_t, a)} \quad (2)$$

La technique Boltzmann est un cas particulier de la sélection Softmax. Ici, la probabilité est calculée avec l'équation(3) qui produit une distribution dite de Boltzmann :

$$\Pr[a_t, s_t] = \frac{e^{Q(s,a)/\Gamma}}{\sum_{a \in A(s_t)} e^{Q(s,a')/\Gamma}} \quad (3)$$

où rest un réel positif. Si rest grand, cette méthode tend vers la méthode gloutonne et si rest proche de 0, elle tend alors vers une 0-gloutonne. Généralement, on utilise cette méthode en faisant varier r au cours de l'apprentissage, en commençant avec une grande valeur assurant l'exploration de l'espace des actions, puis en la diminuant progressivement pour augmenter l'exploitation de l'apprentissage déjà effectué.

4. Modélisation du problème par des agents

La modélisation multi-agents d'un problème consiste à définir une répartition des compétences et des connaissances (les agents), à fournir un outil de recomposition (l'interaction) et à préciser les règles de la recomposition (l'organisation). L'approche multi-agents propose des modèles multi-agents ouverts qui permettent d'avoir une granularité assez fine. Ceci présente beaucoup d'avantages par rapport aux approches classiques de modélisation de transport. Parmi ces avantages, nous citons la modularité, la vitesse et la fiabilité. La modularité permet de rendre la programmation plus simple. Elle permet, de plus, aux SMA d'être facilement extensibles, parce qu'il est plus facile d'ajouter de nouveaux agents à un SMA que d'ajouter de nouvelles capacités à un système monolithique. La vitesse est principalement due au parallélisme, car plusieurs agents peuvent travailler en même temps pour la résolution d'un problème. La fiabilité qui peut être

également atteinte, dans la mesure où le contrôle et les responsabilités étant partagés entre les différents agents, le système peut tolérer la défaillance d'un ou de plusieurs agents. Si une seule entité contrôle tout, alors une seule défaillance de cette entité fera en sorte que tout le système tombera en panne. La modélisation et la conception d'un SMA sont des problèmes majeurs et très délicats [95]. En effet, la communauté SMA ne trouve pas encore de consensus pour définir ce qu'est un SMA. Ainsi, les différentes visions d'un SMA résident essentiellement dans l'approche et la méthodologie de modélisation et de conception de ce dernier. Malgré l'existence de nombreuses approches et méthodologies, le concepteur se pose toujours la question sur la meilleure façon de concevoir son SMA. Les tentatives d'unification des méthodologies de conception sont les bienvenues. Pour la conception d'un SMA, le choix d'une approche ou d'une méthodologie est l'une des difficultés. Pour le problème de transport avec fenêtre de temps, nous avons choisi l'approche *AALAADIN* à cause de la grande liberté donnée au concepteur pour concevoir son SMA. Dans ce cadre méthodologique, les agents sont identifiés par leur rôle au sein du groupe dans un modèle agent-groupe-rôle. Dans un premier temps, nous commençons par la définition des différents agents. Nous donnons ensuite leur comportement dans le système. Enfin, nous décrivons les différentes interactions possibles et les différentes communications entres les différents agents du système et afin de mieux représenter les interactions entre agents, nous avons choisi d'utiliser la notation AUML (Agent Unified Modeling Language) [96][97]. Actuellement, les extensions d'UML permettent de décrire les notions relatives aux agents comme la représentation des interactions entre agents. Dans l'approche basée sur les SMA que nous avons utilisée, les différents agents qui composent le système sont les suivants (fig.3). Ces agents coopèrent, collaborent, communiquent et apprennent en utilisant une méthode d'apprentissage pour la résolution du problème.

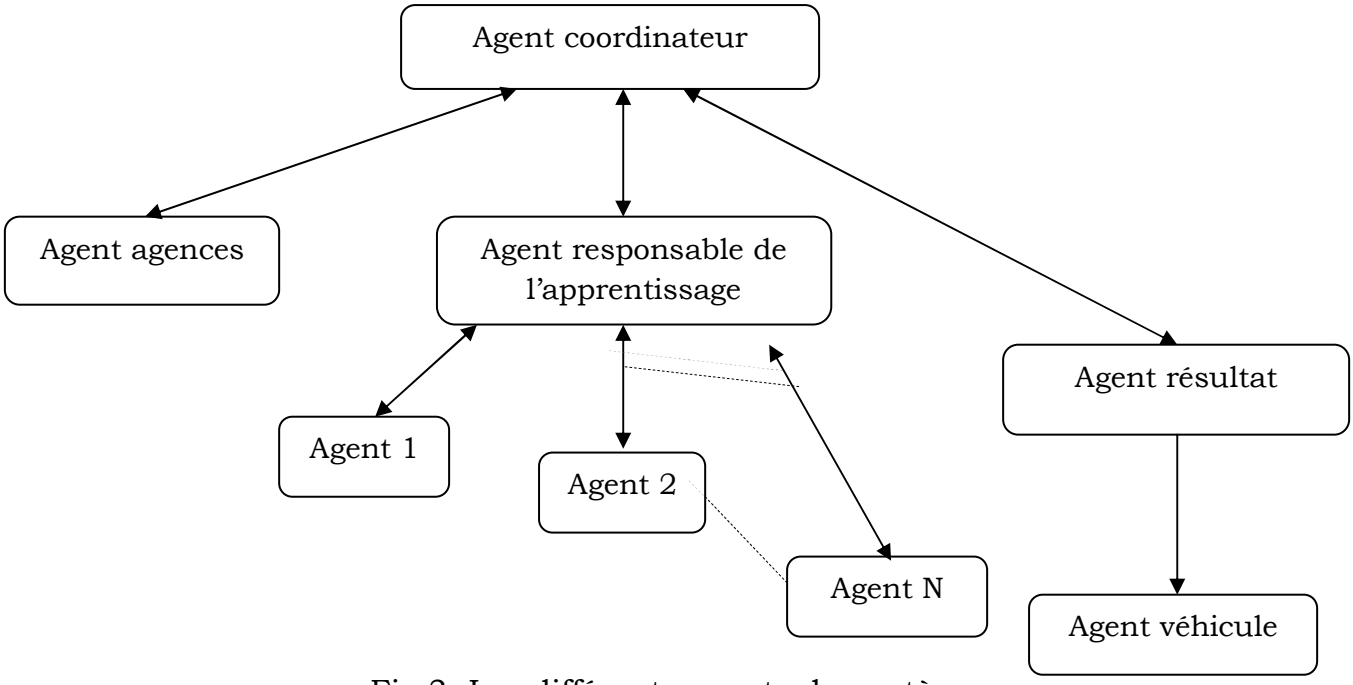

Fig.3. Les différents agents du système

Ainsi, l'agent coordinateur est l'interface utilisateur. Il s'occupe du chargement des données, de la distribution des rôles pour chacun des agents du système et de la réception des résultats ainsi du lancement d'un agent pour leurs affichages. L'agent agences sert à définir pour chaque agence de départ, le nombre de véhicules nécessaires pour servir une destination donnée et, pour chaque véhicule, son nombre de tournées et vérifie si son retour est rempli ou pas. L'agent responsable de l'apprentissage s'occupe de l'apprentissage en lançant des agents l'un à la suite de l'autre pour qu'il effectue l'apprentissage à différentes agences de départ pour chaque destination. A chaque fois qu'un agent termine l'apprentissage, il envoie les résultats à l'agent d'apprentissage qui mettra à jour ses données et lance un autre agent jusqu'à la fin de l'apprentissage de la destination. Les agents d'apprentissage sont lancés par l'agent responsable de l'apprentissage. Ils exécutent la méthode de Q-learning. Leur nombre n'est pas connu d'avance. Tout dépend de la destination et, à la fin de l'apprentissage, ils envoient les résultats à l'agent responsable de l'apprentissage. L'agent résultat reçoit les résultats de l'agent coordinateur et affiche ces résultats et lance un agent véhicule. L'agent véhicule simule le

déplacement d'un véhicule d'une agence de départ vers une agence d'arrivée et, si l'agence de départ a fait un apprentissage pour l'agence de destination choisie, le véhicule passe par les hubs choisis lors de l'apprentissage.

4.1. Comportements des agents dans le système

Les comportements des différents agents du système sont décrits dans ce qui suit. Chaque agent est représenté sous la forme d'un rectangle divisé en compartiments. L'identité (communauté, groupe et rôle) de chaque agent est définie au sommet d'un rectangle. En-dessous, sont représentés ses attributs relatifs à son état. Les actions que l'agent peut exécuter sur son environnement sont représentées dans le troisième compartiment.

- *L'agent Coordinateur*

C'est l'agent interface, c'est le responsable de tous les agents du système, cet agent est identifié comme suit (fig.4) :

Identité	Communauté="transport " Groupe=" coordinateur" Rôle="responsables du projet"
Etat	Libre ou occupé
Actions	Lancer les agents du système Lecture des données Découper les données Transmettre à chaque agent ses données Recevoir les résultats

Fig.4. Agent Coordinateur

Le comportent de l'agent coordinateur est illustré par le diagramme d'activité de la figure.5. Il lance les agents du système comme l'agent responsable de l'apprentissage et l'agent agence. Il se préoccupe de la lecture des données à savoir la matrice distance, la matrice flux, de la subdivision de la matrice flux en deux sous matrices et de la transmission des matrices aux agents agences et à l'agent responsable de l'apprentissage. L'agent coordinateur reçoit de chacun des ces agents les résultats à la fin du traitement et il lance un agent résultats qui affiche l'agence de départ et l'agence d'arrivée sur l'interface de l'agent coordinateur, et qui affiche les résultats en lançant, à

son tour, un agent véhicule qui simule le déplacement d'un véhicule d'une agence de départ vers une agence d'arrivée. Si l'agence de départ a effectué un apprentissage pour la destination, il affiche les hubs choisis.

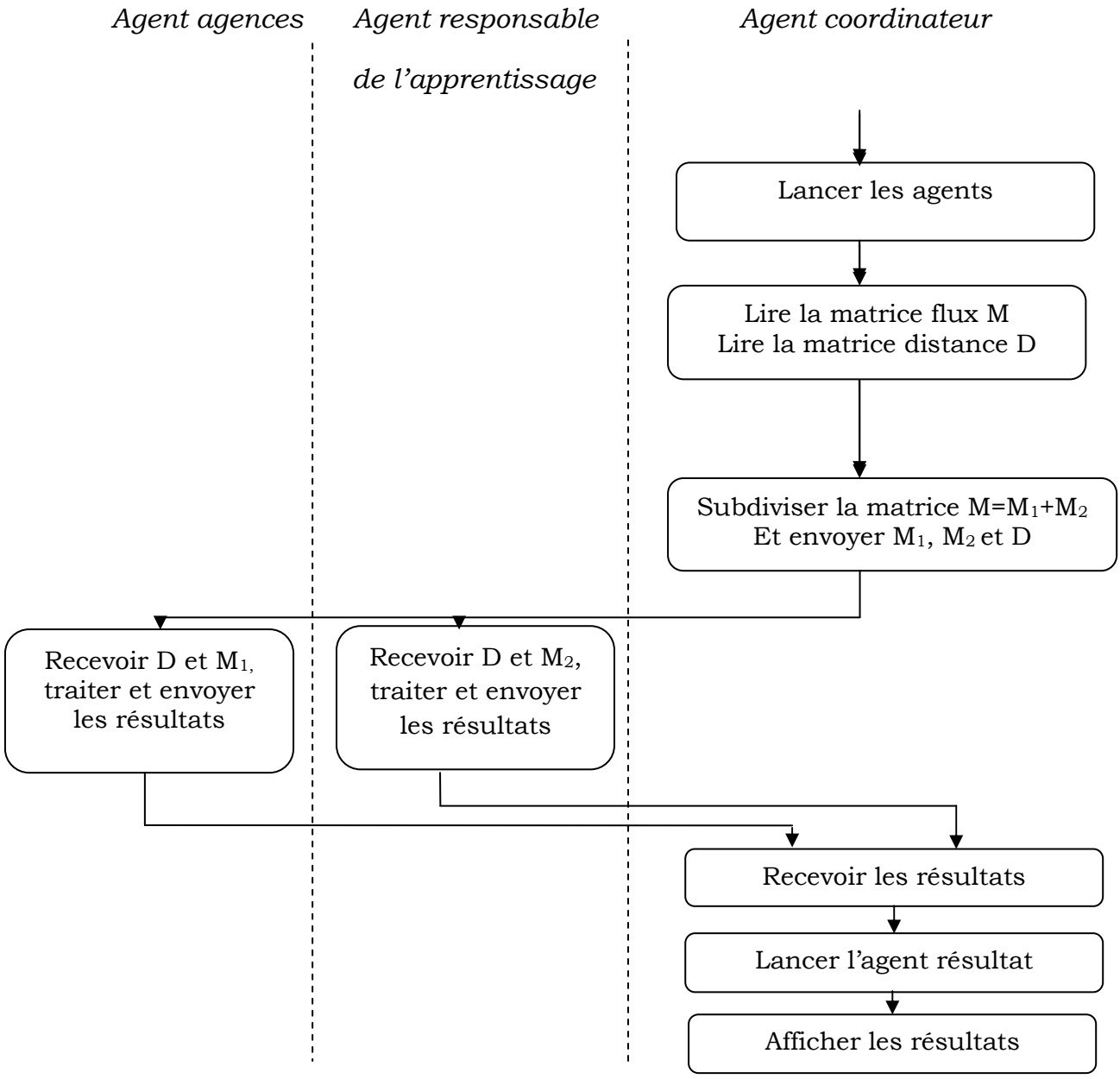

Fig.5. Comportement de l'agent coordinateur

• *L'agent agences* reçoit, de l'agent coordinateur, la matrice flux M_1, la matrice distance qui nous donne les distances entre toutes les agences du système et accède aussi à la base de données du problème qui nous donne

les caractéristiques des agences (hub ou pas, d'arrivée ou de départ). Cet agent est identifié comme suit (fig.6):

Identité	Communauté="transport " Groupe=" Agence de départ" Rôle="servir client "
état	Libre ou occupé
actions	Recevoir données de l'agent coordinateur Lecture de la base de données sur les agences Réduire le nombre de véhicule Transmettre les résultats a l'agent coordinateur

Fig.6. Agent agences

Pour chaque agence de départ, on cherche à réduire le nombre de véhicules qu'elle utilise pour servir une agence d'arrivée donnée. Pour cela un même véhicule est utilisé pour servir une agence d'arrivée tant qu'il y a de la marchandise à transmettre entre ces agences dans le temps de service, et, si l'agence est une agence hub, alors de la marchandise sera prise au retour. Sachant que le nombre de véhicules initial d'une agence de départ pour servir une destination donnée et égal à la quantité de marchandise à transmettre entre cette agence de départ et la destination divisé par la capacité du véhicule comme le montre la procédure de la figure 7.

On calcule aussi le temps que prend chaque agence de départ pour servir une agence d'arrivée en utilisant la matrice des distances, la vitesse moyenne des véhicules qui est supposée à 80 km/h et le temps de chargement/déchargement qui est supposé à 900s. Pour calculer le temps T[x] [y] que prend un véhicule d'une agence x pour servir une agence y, la formule suivante a été utilisée :

$$T[x][y] = \left(\frac{\text{distance}[x][y]}{80}\right) \times 3600 + 900 \quad (4)$$

Ainsi, L'agent agence réduit le nombre de véhicule pour chaque agence de départ et pour chacune de ses destinations, et il définit le nombre de véhicules nécessaires pour servir chaque destination, pour chaque véhicule,

son nombre de tournées en respectant les contraintes et pour chaque véhicule il définit si son retour est rempli ou pas.

Soit *Num véhicule=numéro de véhicules.*
Soit Nb vehiculeInitial= le nombre de véhicules initial
Pour chaque destination ;
Initialise Num véhicule =0 ;
Nb véhiculeInitial=flux[départ][destination]/capacité du véhicule
Tant que (flux [départ] [destination] !=0)
- *Num véhicule=Num véhicule+1 ;*
- *Calcul du temps nécessaire pour servir la destination,*
- *Calcul du nombre de tournées possible pour le véhicule en 24h.*
- *Si nombre de tournées >1 alors Réduire le nombre de véhicule initial.*
- *Mise à jour du flux.*

Fin de Tant que.
Si la destination est un Hub alors
Prendre de la marchandise au retour
Réduire le nombre de véhicule initial.

Fig.7. Procédure de réduction du nombre de véhicules

• *L'agent responsable de l'apprentissage* est un agent qui contient une interface graphique, il reçoit de l'agent coordinateur la matrice flux M_2, la matrice distance D, cet agent est identifié comme suit (fig.8):

Identité	Communauté="transport " Groupe="responsable de l'apprentissage" Rôle="Apprentissage "
état	Libre ou occupé
actions	Recevoir données de l'agent coordinateur Lecture de la base de données sur les agences Lancer des agents d'apprentissage Afficher résultats d'apprentissage Transmettre les résultats a l'agent coordinateur

Fig.8. Agent responsable de l'apprentissage

Le comportement de cet agent est illustré par le diagramme d'activité de la figure (9). Comme la quantité de marchandise à transmettre entre une agence de départ et une agence d'arrivée est inférieure à la capacité du véhicule, pour une agence de départ et pour une agence de destination donnée, l'agent d'apprentissage essaie de trouver les meilleurs hubs à insérer dans la liste des tournées d'un véhicule tant que la quantité est

inferieure ou égale à la capacité du véhicule. Pour chacune des agences de destination, l'agent responsable de l'apprentissage sélectionne une agence de départ et lance un agent d'apprentissage pour cette agence. Cet agent applique la méthode Q-learning pour trouver les meilleurs hubs et les paramètres de la méthode (nombre d'itérations, l'amortissement, le taux d'apprentissage et l'exploration) sont les entrées de l'interface graphique de l'agent responsable d'apprentissage.

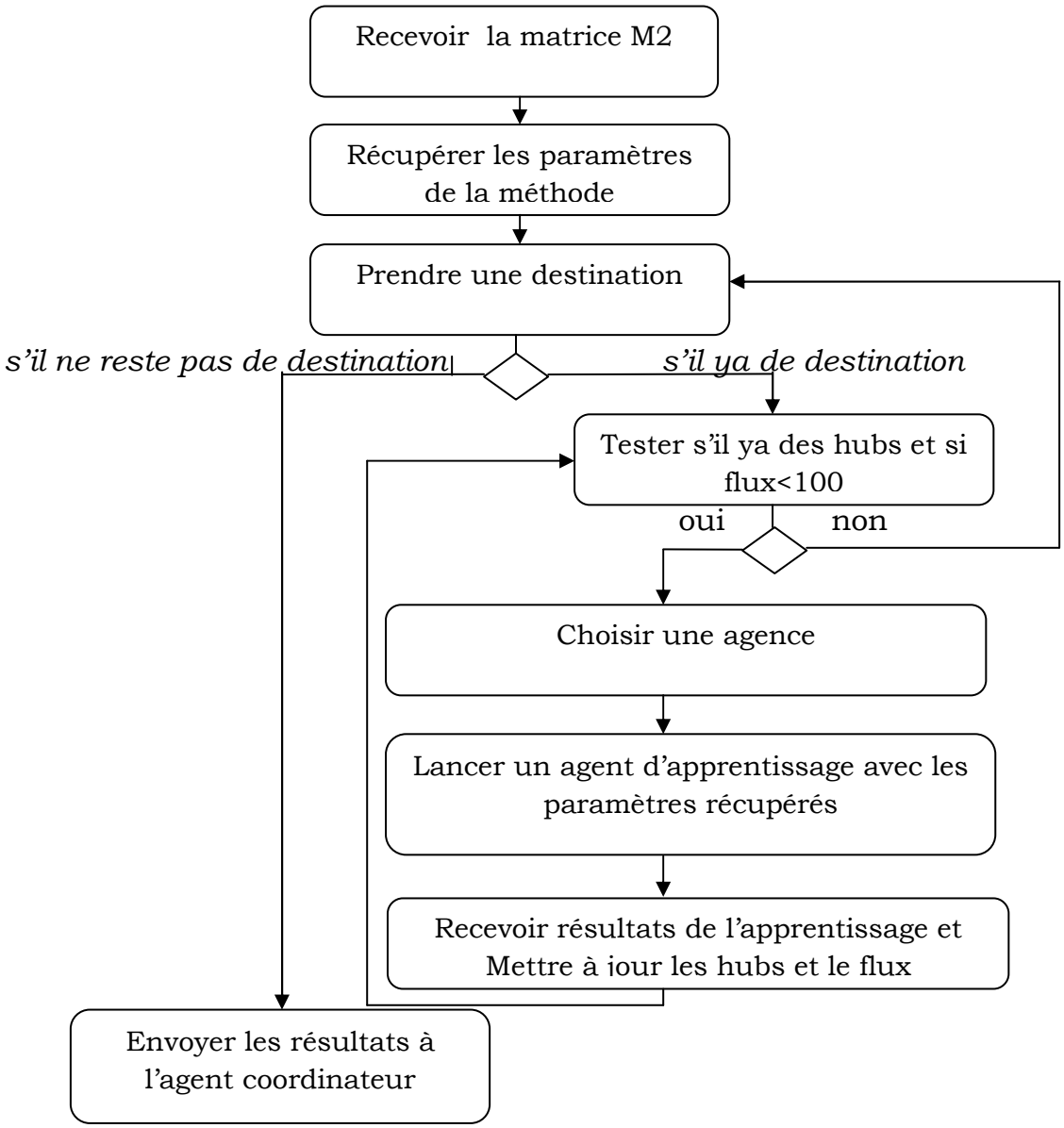

Fig.9. Comportement de l'agent responsable d'apprentissage

Dès que l'agent termine l'apprentissage, il envoie les résultats à l'agent responsable de l'apprentissage qui mettra à jour la liste des hubs (les actions de l'apprentissage) en enlevant les hubs choisis par l'agent qui a effectué l'apprentissage. Ainsi, une autre agence de départ est choisie et un autre agent est lancé pour lui effectuer l'apprentissage, ainsi de suite jusqu'à ce qu'on n'ait plus de hubs à insérer pour la destination. L'apprentissage étant terminé pour une destination donnée, nous appliquons la même procédure pour toutes les autres destinations. Le maximum de hubs que l'on peut choisir pour une destination donnée est trois. Ainsi, dés que l'agent responsable d'apprentissage reçoit les résultats d'un agent d'apprentissage il vérifie si, lors du passage par les hubs choisis, les contraintes ne sont pas violées. Sinon, il enlève les hubs dépassant le temps de service.

- *Les agents d'apprentissage* appartiennent au groupe d'apprentissage, ils sont lancés par l'agent responsable de l'apprentissage. Leur rôle est d'effectuer l'apprentissage pour différentes agences en utilisant le Q-learning pour choisir les meilleurs hubs à insérer dans un itinéraire d'un véhicule allant d'une agence de départ vers une agence d'arrivée. Dans ce qui suit, on définit les états, les actions et la fonction de récompense pour notre stratégie d'apprentissage, basée sur le Q-Learning pour la résolution du problème de transport de marchandises avec fenêtre de temps. Soient I , le nœud courant d'un véhicule, J , le nœud d'arrivée d'un véhicule et *a,* une mesure prise à un instant donné. Les états sont l'ensemble (i,j) et les actions A(i,j), l'ensemble des candidats pour le prochain nœud (dans notre cas c'est l'ensemble des agences hub candidates à être insérées dans la tournée). Pour la fonction de récompense R [(i,j),a] : en étant dans un état (i,t) et en choisissant l'action *a,* on obtient la récompense R qui est définie par l'expression (5) :

$$R[(i,j),a] = \text{temps}[i][a] \tag{5}$$

Ceci correspond au temps que prend le véhicule pour aller de l'agence *i* vers l'agence *a* en ayant pour destination l'agence *j*.

*Les Qvaleurs Q[(i,j),a] c*onstituent une matrice ayant, pour lignes, les agences, et pour colonnes, les actions (agences *hubs*).

Initialement, on prend :

$$\text{QValues}[(i,j),a] = \text{temps}[a][j] \quad (6)$$

où *i* est l'état courant, a, l'action, et j, la destination

La fonction Q est mise à jour selon l'équation (1). Pour le choix des actions**,** plusieurs stratégies sont utilisées classiquement. Dans notre application, les deux stratégies ξ-gloutonne et Softmax ont été testées.

En cours de l'apprentissage le nombre d'agences hub à insérer varie. Ainsi, les actions du Qlearning ne sont pas statiques, elles sont dynamiques . C'est ainsi qu'une agence hub ne peut être insérée que si la somme du flux de départ avec son flux est inferieure à la capacité du véhicule. Les agents d'apprentissage effectuent l'apprentissage en suivant l'algorithme de la figure (10) qui est l'algorithme de Q-learning appliqué à notre problème.

Initialiser les Qvaleurs
Pour tout épisode
- *Initialiser s*
- *Initialiser le nombre d'actions*
- *Initialiser les actions*
- ***Répéter***
  - *Choisir une action* a *avec la* stratégie ξ-gloutonne
  - *Faire l'action* a*, observer* r *et le prochain état* s'
  - *Mettre à jours la fonction QValeurs :*

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a' \in A} Q(s',a')\right]$$

  - $s \leftarrow s'$
  - *Mettre à jour le nombre d'actions et les actions*

Jusqu'a *(nombre actions=0)*
Fin de pour

Fig.10. Algorithme de Q-learning utilisé par les agents d'apprentissage

Ainsi, chaque agent d'apprentissage lancé, exécute cet algorithme de Q learning. La sortie finale de cet algorithme est une matrice des Q valeurs et, à partir de cette matrice, en étant dans un état donné, on cherche l'état (hub) à prendre pour atteindre la destination. Une fois les hubs choisis, ils sont transmis à l'agent responsable de l'apprentissage qui vérifie la faisabilité des solutions trouvées, autrement dit, si le véhicule ne viole pas les contraintes en passant par les hubs sélectionnés pour servir sa destination. Dans le cas de viol de contrainte, l'agent responsable d'apprentissage enlève les hubs un à un jusqu'à ne plus garder que ceux qui ne violent aucune contrainte.

- *L'agent résultat* est un agent avec une interface graphique, Il reçoit les résultats de l'agent coordinateur, en choisissant une agence de départ quelconque et une agence de destination. L'agent récupère les coordonnées GPS des agences et leurs caractéristiques. Il vérifie s'il s'agit d'une agence de départ, d'arrivée ou d'une agence hub. Ensuite il affiche les agences qu'elle soit de départ ou d'arrivé et s'il y a les hubs sur l'interface graphique de l'agent coordinateur, et lance un agent véhicule pour simuler le déplacement de l'agence de départ vers l'agence d'arrivée affiché. L'agent résultat affiche sur son interface les données suivantes :

  le flux qui est la quantité de marchandise à transporter de l'agence de départ vers l'agence de destination choisie,

  le N° véhicule : relatif au nombre de véhicules nécessaires pour transporter le flux,

  le nombre de tournées pour chaque véhicule, en lui mentionnant le nombre d'allers et retours qu'il fera pour servir la destination,

  le nombre de retours remplis qui est le nombre de retours où le véhicule ne revient pas vide à son départ vide en prenant de la marchandise en retour si la destination est un hub,

l'apprentissage qui nous informe si l'agent responsable de l'apprentissage a lancé un agent pour effectuer un apprentissage pour servir la destination choisie,

l'affichage des hubs choisis lors de l'apprentissage par ordre de passage si la zone apprentissage est à «OUI ».

- *L'agent véhicule* simule le déplacement d'une agence de départ vers une agence d'arrivée. Il prend les résultats à partir de l'agent résultat, et si l'agence de départ a lancé un agent d'apprentissage, le véhicule passe par les hubs choisis pour prendre leurs marchandises afin de servir la destination. Le déplacement du véhicule se fait sur l'interface graphique de l'agent coordinateur.

4.2. Les interactions entre les agents du système

Les différents agents du système coopèrent et coordonnent entre eux pour résoudre le problème. il n'existe pas de concurrences entre les différents agents du système. Chacun des agents joue un rôle dans la résolution du problème. Les différentes interactions entre les agents qui composent le système sont décrites dans ce qui suit :

- *Interaction de l'agent coordinateur et les agents du système:* l'agent coordinateur interagit avec les différents agents du système comme le montre le diagramme de séquence de la figure (11).

- *Interaction de l'agent responsable de l'apprentissage et les agents d'apprentissage: l*'agent responsable de l'apprentissage lance des agents d'apprentissage l'un a la suite de l'autre, pour chaque destination jusqu'à ce qu'on n'ait plus de hubs à insérer pour la destination ou le véhicule est rempli. Le diagramme de séquence de la figure(13) montre les différentes interactions entre ces agents pour une destination donnée.

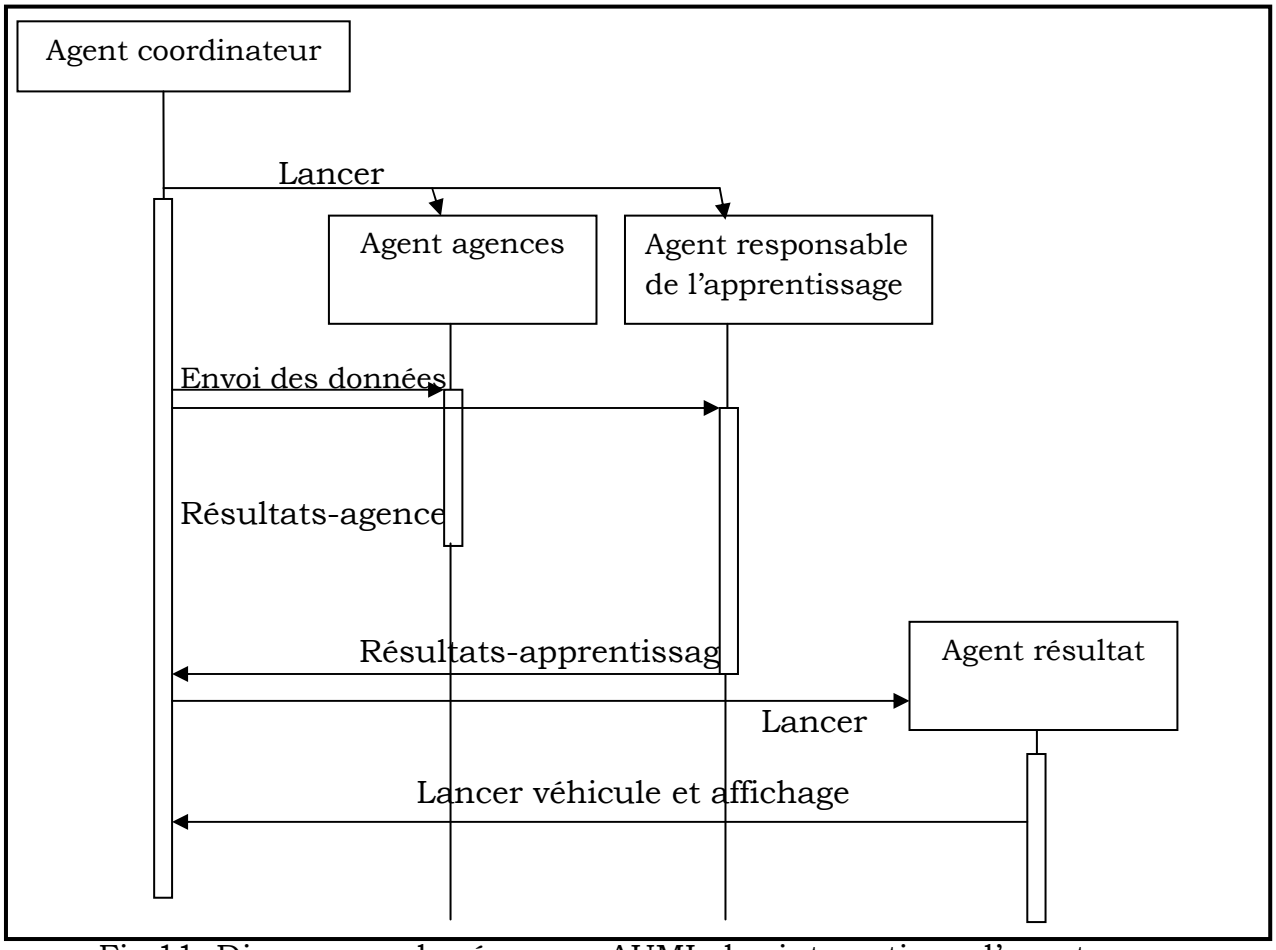

Fig.11. Diagramme de séquence AUML des interactions d'agent coordinateur avec les agents du système.

Primitive	Désignation
Lancer	l'agent coordinateur lance l'agent
Envoi des données	l'agent coordinateur envoi le flux M1 et M2 et D
Résultats-agence	l'agent agences renvoi les résultats a l'agent coordinateur
Résultats-apprentissage	l'agent responsable de l'apprentissage envoi les résultats a l'agent coordinateur
Lancer véhicule et affichage	l'agent résultat affiche les agences et lance un agent véhicule qui simule le déplacement sur l'interface de l'agent coordinateur.

Fig.12. Primitives d'interactions *:*

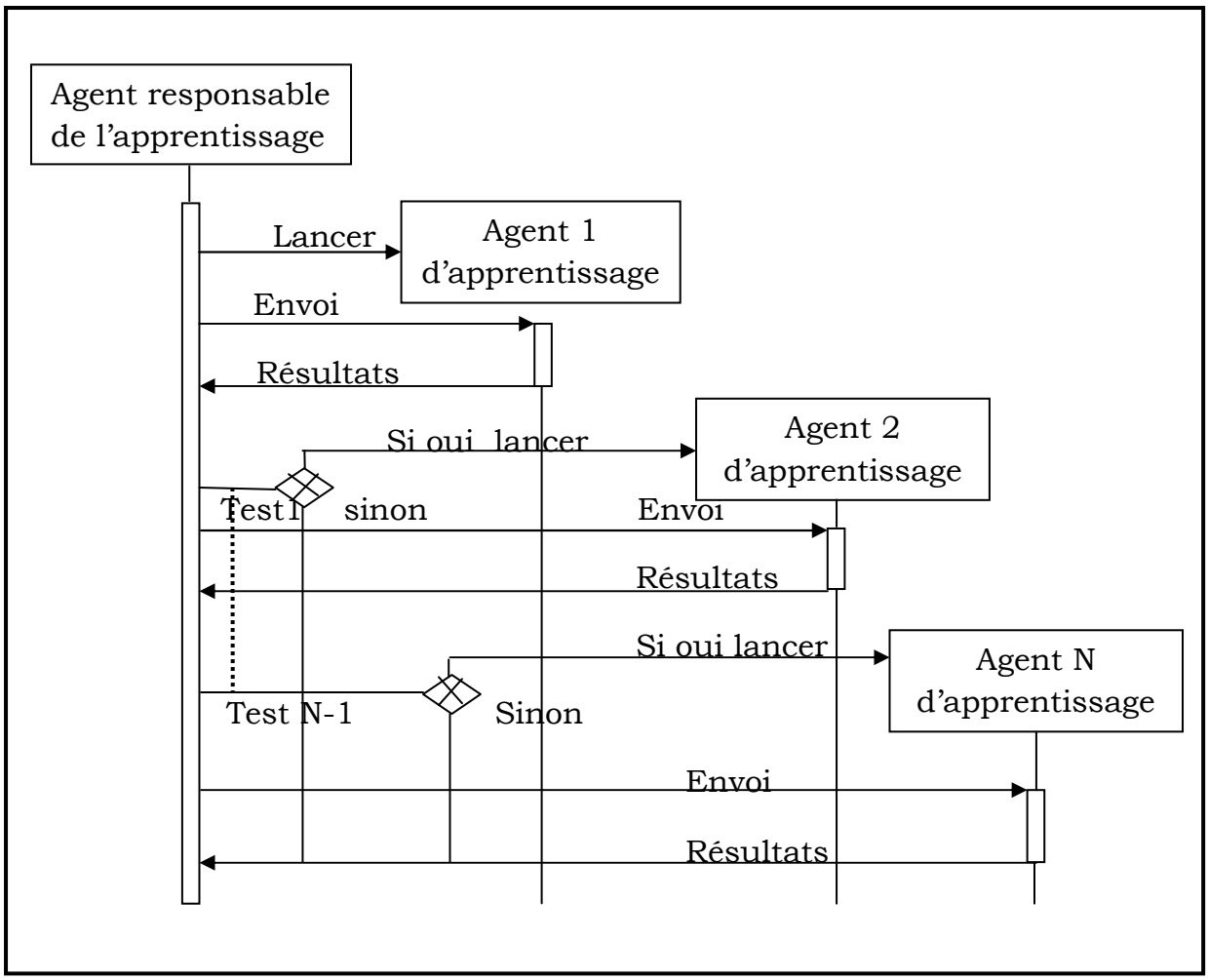

Fig.13. Diagramme de séquence AUML des interactions d'agent responsable d'apprentissage et agent d'apprentissage pour une destination donnée.

Primitive	**Désignation**
Lancer	L'agent responsable de l'apprentissage lance l'agent d'apprentissage
Envoi	L'agent responsable de l'apprentissage envoi les données à l'agent d'apprentissage (listes des hubs, paramètres du Q-learning)
Résultats	L'agent d'apprentissage renvoi les résultats(les hubs choisis) à l'agent responsable de l'apprentissage
Test (1…N-1)	L'agent responsable de l'apprentissage verifie s'il ya des hubs restants et si le véhicule n'est pas encore rempli
Si oui lancer	L'agent responsable de l'apprentissage lance un agent d'apprentissage si le test est vrai (il reste des hubs et le véhicule n'est pas rempli.
sinon	L'agent responsable d'apprentissage affiche les résultats sur sa propre interface et envoi les résultats à l'agent coordinateur

Fig.14. Primitives d'interactions

- *Interaction de l'agent résultat et agent véhicule :* l'agent résultat lance un agent véhicule et lui transmet des informations sur la destination, le départ et sur les hubs si la destination a effectué un apprentissage pour servir la destination. Les interactions entres les deux agents sont illustrées dans le diagramme de séquence de la figure (15).

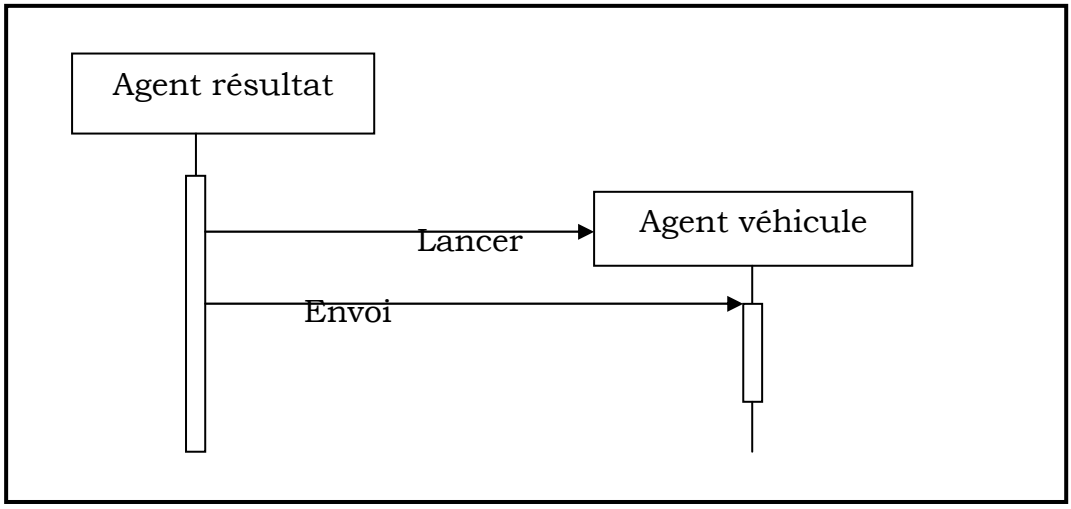

Fig.15. Diagramme de séquence AUML des interactions d'agent résultat et agent véhicule.

Primitive	**Désignation**
Lancer	l'agent résultat lance l'agent véhicule
Envoi	l'agent transmet toute information nécessaire pour l'agent véhicule afin de simuler le déplacement

Fig.16. Primitives d'interactions

- *Communication :* la communication entre les différents agents du système se fait par envoi de message d'un agent émetteur vers un agent récepteur. Les différents messages effectués sont illustrées dans les diagrammes de séquence fait précédemment.

Dans notre travail, les messages échangés ont la structure suivante :

« La communauté, le groupe, le rôle, l'information » où :

- *la communauté* : c'est la communauté de l'agent récepteur.
- *le groupe* : c'est le groupe de l'agent récepteur.

- *le rôle* : c'est le rôle de l'agent récepteur.
- *l'information* : c'est l'information envoyé entre l'agent émetteur vers l'agent récepteur.

5. Conclusion

Dans ce chapitre, nous avons présentés l'application concernant le problème de tournée de véhicules avec fenêtres de temps. Nous avons adopté la modélisation systèmes multi-agents constituée de six types d'agents. Chacun des agents participe, interagit avec les autres afin de résoudre le problème. L'agent d'apprentissage applique la méthode de Q-learning pour chaque agence sélectionnée de la part de l'agent responsable de l'apprentissage et trouve le chemin optimal pour servir la destination et cela on sélectionnant les meilleurs hubs.
Le chapitre suivant est consacré à l'implémentation des différents agents du système et à l'illustration de leurs différents résultats.

Chapitre 4

Implémentation de la méthode et résultats

1. Introduction

Rappelons que l'application visée dans ce mémoire est le problème de transport de marchandises avec fenêtres de temps. Il s'agit d'un problème d'optimisation qui consiste à générer un plan de transport avec le minimum de véhicules pour transiter un ensemble de flux entre des agences de départ et des agences d'arrivée. L'itinéraire compte trois types d'agences à savoir, l'agence de départ et d'arrivée, l'agence d'arrivée et l'agence hub. Les flux transitent (ou non) par un ou plusieurs ou hubs tout en respectant les contraintes suivantes:

- tout le flux doit être acheminé en vingt quatre heures,
- les agences ont une heure d'ouverture et une heure de fermeture,
- le nombre maximum de transits dans un hub est limité à 3,
- la capacité des véhicules est limitée,
- le véhicule retourne à son agence de départ après avoir servi sa destination.

Ainsi, dans ce chapitre, nous allons procéder à l'implémentation de la méthode basée sur le choix du logiciel qui servira de plate forme de développement du système qui transforme la modélisation multi-agents obtenue précédemment en un Système Multi-Agent (SMA) directement exécutable. Après cette implémentation des différents agents du système et les différents résultats obtenus, nous présentons l'influence des paramètres du Q-learning sur les résultats de l'apprentissage.

2. Choix de la plateforme multi-agent

Le besoin de mettre en œuvre des systèmes à plusieurs composantes autonomes nécessite une infrastructure de logiciels utilisée comme environnement pour le déploiement et l'exécution d'un ensemble d'agents. Cette infrastructure est appelée *plate-forme de développement des systèmes multi-agents*.

Dans notre travail nous avons implémenté les différents agents sous *MADKIT* version4.2.0 qui est une plate-forme multi-agents écrite en Java version 6 basée sur un modèle organisationnel à savoir le modèle agent-groupe-rôle. Cette plate-forme d'exécution des SMA utilisant un micro-noyau agent est développée par O.Gutknecht et J.Ferber [8]. Le modèle organisationnel est basé sur l'approche AALAADIN décrite dans le chapitre précédent. Dans cette approche, les agents sont identifiés par leur rôle au sein du groupe dans un modèle agent-groupe-rôle.

3. Implémentation des agents du système

Dans ce qui suit nous présentons les différents agents du système sachant que c'est l'agent coordinateur le responsable sur tous ces agents.

- *L'agent Coordinateur*

C'est un agent qui contient une interface graphique comme le montre la figure.1 qui nous donne un menu nous permettant de lancer l'agent Agences, agent responsable de l'apprentissage et l'agent Résultat tout comme il nous permet aussi de lire la matrice flux, la matrice distance et de

quitter l'application. C'est dans cette interface que l'agent Résultat affiche les agences avec leurs numéros et lance un véhicule qui simule le déplacement d'une agence Départ vers une agence Arrivée en passant sur des hubs si l'agence a effectué un apprentissage.

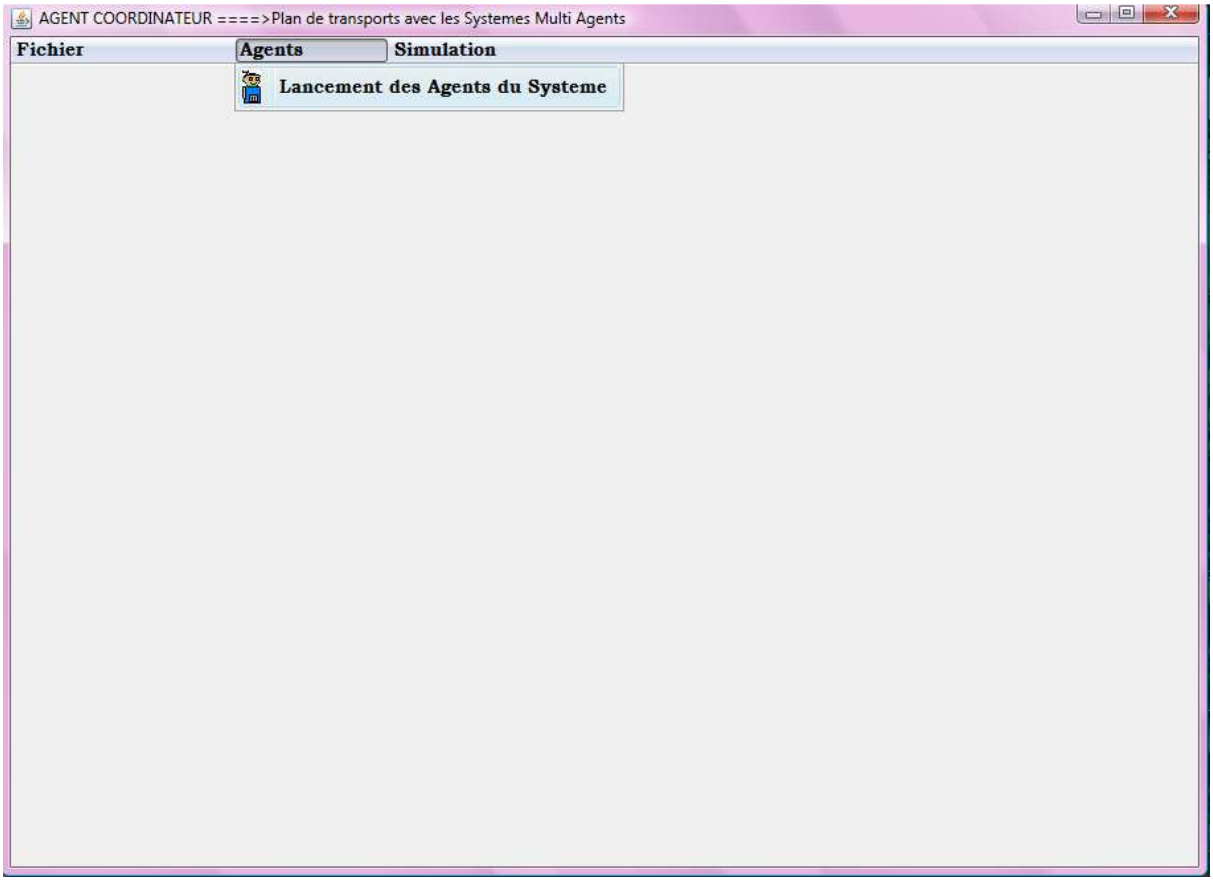

Fig.1 Interface graphique de l'agent coordinateur

- *L'agent Agences*

L'agent agences ne contient pas une interface graphique (fig.2). Il reçoit la matrice M_1 et la matrice D, et il fait son traitement et il envoie les résultats à l'agent Coordinateur.

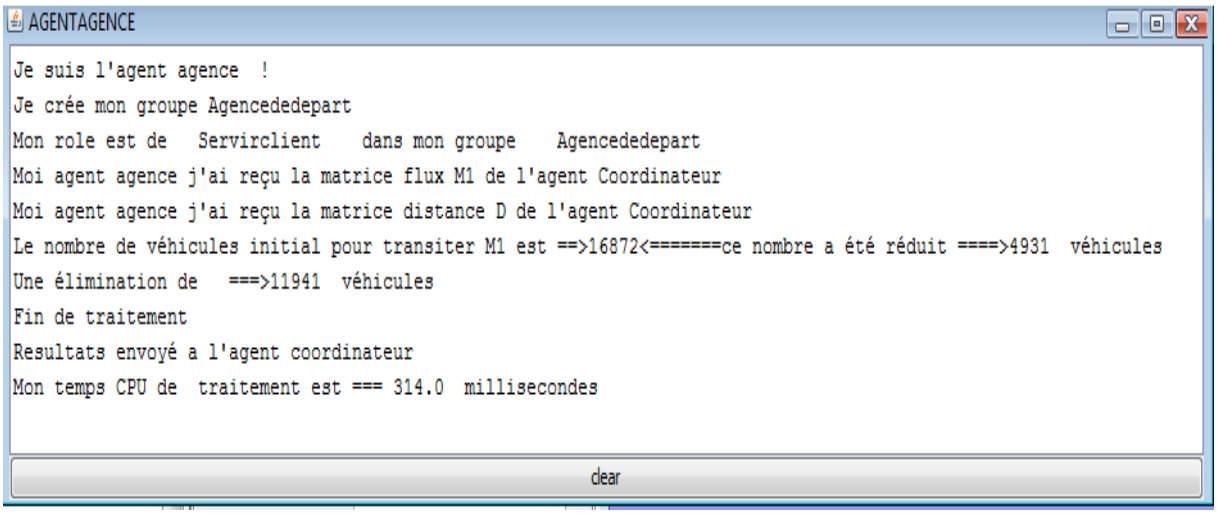

Fig.2. Agent Agence

- *Agent responsable de l'apprentissage*

C'est un agent avec une interface graphique (fig.3). Il possède quarte entrées qui sont les paramètres de du Q-learning à savoir, le nombre d'itérations, l'amortissement, le taux d'apprentissage et l'exploration. Un bouton pour lancer l'apprentissage et l'affichage des résultats dans la zone d'écriture ainsi qu'un bouton pour les envoyer a l'agent coordinateur sont prévus.

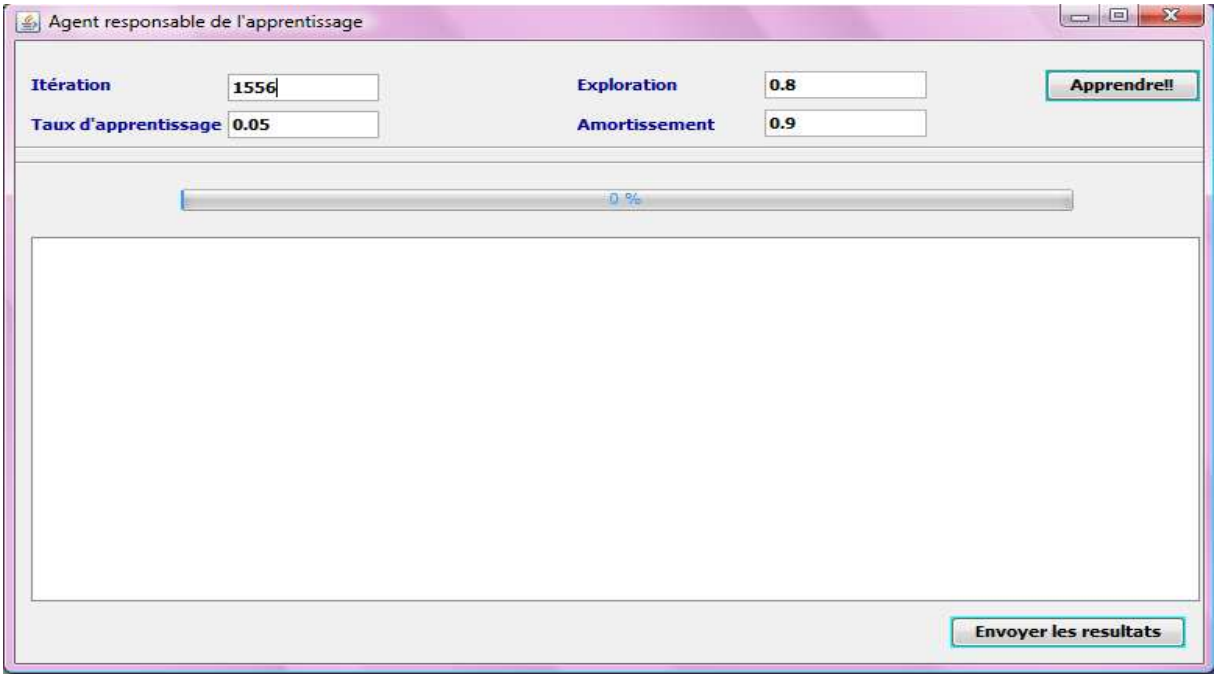

Fig.3. Interface graphique de l'agent responsable de l'apprentissage

- *Les agents d'apprentissage*

Ces agents du groupe d'apprentissage sont lancés l'un à la suite de l'autre par l'agent responsables de l'apprentissage. Leurs rôles est d'effectuer l'apprentissage en utilisant le Q-learning pour différentes agences de départ. Leurs nombres n'est pas connu et ils ne sont pas visibles pour l'utilisateur.

- *Agent Résultat*

Cet agent se préoccupe de l'affichage des résultats en choisissant le départ et la destination comme (fig.4).

Fig.4. Interface de l'agent résultat avant affichage des résultats.

Ensuite cet agent affiche les résultats comme le montre l'exemple résultats de la figure 5 que l'agent affiche. Cet agent affiche aussi les agences et lance l'agent véhicule sur l'interface de l'agent Coordinateur.

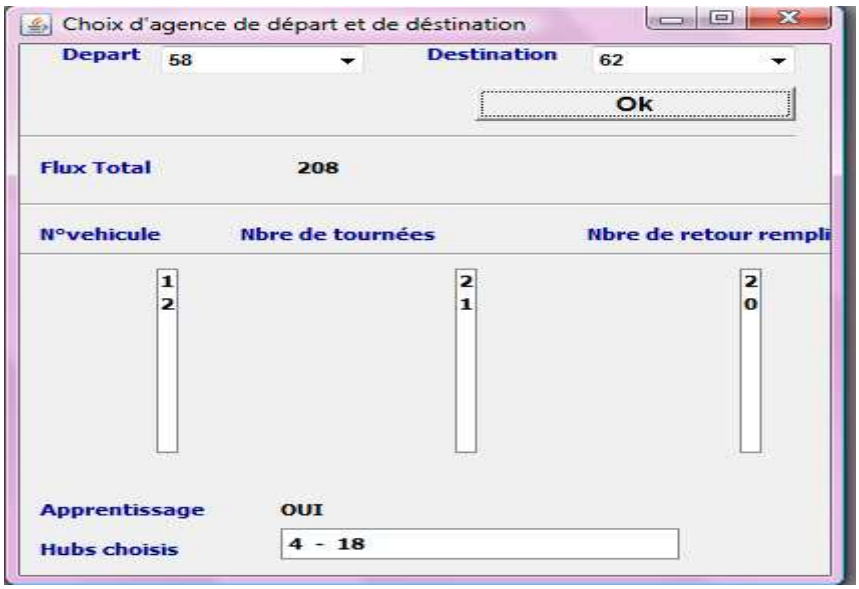

Fig.5. interface de l'agent résultat après affichage des résultats

4. Résultats de simulation :

Le fichier de données utilisé comprend 81 agences (matrice de 81*81) dont 29 sont des agences de départ et parmi ces agences de départ, il y a celles qui sont au même temps des hubs. Cette matrice contient les flux à transiter entre l'agence de départ qui est sur la ligne et l'agence d'arrivée qui est sur la colonne. Nous disposons aussi d'une matrice de distances qui nous renseigne sur les distances entre les différentes agences et d'une base de données qui contient les caractéristiques des agences à savoir, si c'est des agences de départ, d'arrivée ou des hubs, leurs coordonnées GPS, etc.…

Pour l'agent Agences, il définit :

- pour chaque agence de départ et sa destination, le nombre de véhicules,
- pour chaque véhicule son nombre de tournées et le nombre de retour rempli.

En effet, prenons l'exemple où les agences sont :

N° Agence	Caractéristique Agences
11	Agence de départ et d'arrivé
22	Agence de départ et d'arrivé
30	Agence de départ et d'arrivé
41	Agence d'arrivée
52	Agence HUB
58	Agence HUB
59	Agence de départ et d'arrivé
62	Agence HUB
68	Agence HUB
69	Agence d'arrivée
70	Agence d'arrivée

et où la matrice flux M de nos données est donnée ci-après (fig.6) :

	22	**30**	**41**	**52**	**62**	**68**	**69**	**70**
11	189	2944	210	941	1233	2999	1269	3427
30	6	0	2	33	389	180	36	105
58	6	28	71	104	208	159	148	339
59	3	2	0	1	11	8	3	3

Fig.6. Exemple de matrice flux M pris dans la matrice flux de la base de données

La matrice M_1 de cet exemple est celle présentée en figure 7 et la matrice M_2 celle présenté en figure 8.

	22	**30**	**41**	**52**	**62**	**68**	**69**	**70**
11	100	2900	200	900	1200	2900	1200	3400
30	0	0	0	0	300	100	0	100
58	0	0	0	100	200	100	100	300
59	0	0	0	0	0	0	0	0

Fig.7. Matrice flux M_1 de la matrice M

	22	**30**	**41**	**52**	**62**	**68**	**69**	**70**
11	89	44	10	41	33	99	69	27
30	6	0	2	33	89	80	36	5
58	6	28	71	4	8	59	48	39
59	3	2	0	1	11	8	3	3

Fig.8 Matrice flux M_2 de la matrice M

Le tableau 1, nous donne les résultats que l'agent Agence fournit à l'agent Coordinateur pour les agences de l'exemple afin de transiter le flux de la matrice M_1 de la figure 7. Sachant que le nombre total initial de véhicules était de 144 véhicules, grâce à l'agent Agence, ce nombre est réduit 65 véhicules, ce qui veut dire que, avec la stratégie de réduction du nombre de véhicules, nous avons pu éliminer 75 véhicules.

Au départ, le nombre de véhicules nécessaires pour faire transiter tout le flux de la matrice M_1 de la base de données (81 agences) des agences de départs vers les agences d'arrivées était de 16872 véhicules. Grâce à l'agent Agences, ce nombre a été réduit à 4931 véhicules, soit, une élimination de 11941 véhicules avec un temps de calcul de 0,314 secondes.

Départ	*Destination*	*N° véhicule*	*Nombre de tournées*	*Nombre de retour rempli*
11	22	1	1	0
11	30	1	5	0
11	30	2	5	0
11	30	3	5	0
11	30	4	5	0
11	30	5	5	0
11	30	6	4	0
11	41	1	2	0
11	52	1	2	2
11	52	2	2	2
11	52	3	2	2
11	52	4	2	2
11	52	5	1	1
11	62	1	3	3
11	62	2	3	3
11	62	3	3	3
11	62	4	3	3
11	68	1	2	2
11	68	2	2	2
11	68	3	2	2
11	68	4	2	2
11	68	5	2	2
11	68	6	2	2
11	68	7	2	2
11	68	8	2	2
11	68	9	2	2
11	68	10	2	2
11	68	11	2	2
11	68	12	2	2
11	68	13	2	2
11	68	14	2	2
11	68	15	1	1

Départ	*Destination*	*N° véhicule*	*Nombre de tournées*	*Nombre de retour rempli*
11	69	1	2	0
11	69	2	2	0
11	69	3	2	0
11	69	4	2	0
11	69	5	2	0
11	69	6	2	0
11	70	1	2	0
11	70	2	2	0
11	70	3	2	0
11	70	4	2	0
11	70	5	2	0
11	70	6	2	0
11	70	7	2	0
11	70	8	2	0
11	70	9	2	0
11	70	10	2	0
11	70	11	2	0
11	70	12	2	0
11	70	13	2	0
11	70	14	2	0
11	70	15	2	0
11	70	16	2	0
11	70	17	2	0
30	62	1	3	3
30	68	1	1	1
30	70	1	1	0
58	52	1	1	1
58	62	1	2	2
58	68	1	1	1
58	69	1	1	0
58	70	1	2	0
58	70	2	1	0

Tableau 1. Résultats de l'agent agence pour transiter la matrice flux M_1

L'agent responsable de l'apprentissage pour faire transiter le flux de la matrice M_2 lance des agents d'apprentissage, chaque agent applique le Q-learning avec les paramètres entrées sur l'interface graphique de l'agent responsable de l'apprentissage et ce, dans le but de sélectionner les meilleurs hubs à insérer dans l'itinéraire d'un véhicule non rempli allant d'un départ vers une destination donnée. La sortie de l'apprentissage est un tableau qui a pour lignes les agences et qui a une seule colonne. La colonne nous donne l'état que le véhicule peut choisir en étant dans l'état courant (l'agence de la ligne) et allant à la destination j. C'est à partir de ce tableau que l'agent d'apprentissage choisit ces hubs et envoie les résultats à l'agent responsable de l'apprentissage.

Le maximum de hubs qu'on peut choisir pour une destination donnée est trois. Dés que l'agent responsable d'apprentissage reçoit les résultats d'un agent d'apprentissage, il vérifie si, lors du passage par les hubs choisis, les contraintes ne sont pas violées, sinon il enlève les hubs pour ne garder que ceux qui ne dépassent pas le temps du service. La figure 9 montre le cas où l'agent responsable d'apprentissage affiche tous les résultats de l'apprentissage de toutes les agences ayant effectué l'apprentissage. Si, par exemple, on prend comme agence de départ l'agence 58 et comme destination l'agence 22, le tableau.3 montre les QValeurs finales après l'apprentissage en notant que seules les agences de départ sont prises pour les lignes. La sortie finale de l'apprentissage est donnée par le tableau.4. Dans l'état courant, 58 du tableau.3 plusieurs hubs (actions) peuvent être sélectionnés 4, 18, 29, 40, 52, 60,62 et 68. L'état sélectionné est 29 puisque c'est lui qui a la plus petite valeur des QValeurs. En allant à l'état 29, plusieurs hubs peuvent être sélectionnés, le 4, 18,40 ,52 et 68. Le hub 40 sera choisi puisqu'il contient la plus petite valeur des QValeurs et à l'état 40 le hub 68 sera choisi à son tour. Ainsi l'agent d'apprentissage envoie les hubs choisis à l'agent responsable de l'apprentissage. Seuls, les hubs satisfaisant les contraintes, donc la route optimale pour l'agence 58 vers la destination 22 (fig.9) sont choisis.

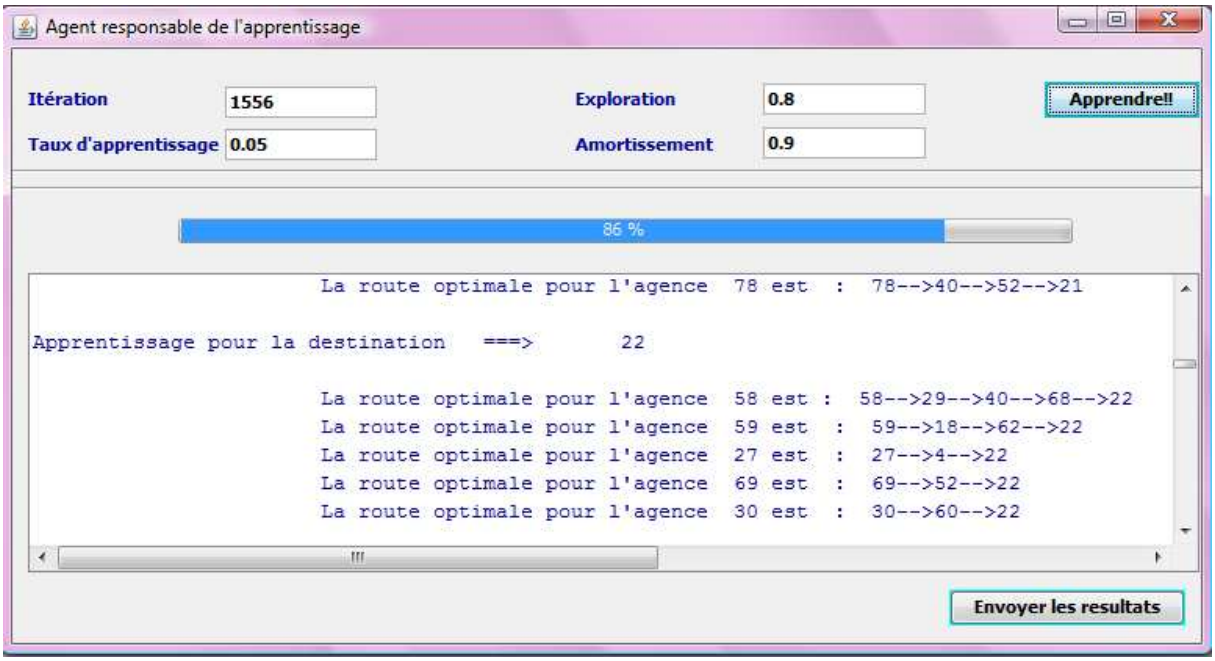

Fig.9.Résultats de l'apprentissage affiché par l'agent responsable de l'apprentissage.

Le tableau 2 nous donne les résultats que l'agent responsable de l'apprentissage fournit à l'agent Coordinateur pour les agences de l'exemple afin de transiter le flux de la matrice M_2 de la table 3 avec un temps de calcul de 3,342 secondes.

Chacun des agents Agences et responsable de l'apprentissage envoient les résultats à l'agent Coordinateur qui, à son tour, les envoie à l'agent Résultats qui s'occupe de l'affichage des résultats tel que le flux à transiter, le nombre de véhicules, le nombre de tournées pour chaque véhicule, le nombre de retours remplis et si l'agence a effectué l'apprentissage oui ou non. Si c'est oui, alors quels hubs sont choisis et qui lance l'agent véhicule simulant le déplacement de l'agence de départ vers l'agence d'arrivée en passant par les hubs en cas où ils existent (fig.10). Chacune des agences Départs, Arrivées, hubs est affichée selon ces coordonnées GPS en consultant la base de données.

Départ	Destination	Nombre de véhicule	Si l'agence de départ a effectué l'apprentissage	HUB 1	HUB 2	HUB 3
11	22	1	NON			
30	22	1	OUI	60		
58	22	1	OUI	29	40	68
59	22	1	OUI	18	62	
11	30	1	NON			
58	30	1	OUI	29		
59	30	1	NON			
11	41	1	OUI	4	29	
30	41	1	OUI	62		
58	41	1	OUI	52		
11	52	1	OUI	4	60	
30	52	1	OUI	40		
58	52	1	OUI	29		
59	52	1	OUI	18	62	
11	62	1	NON			
30	62	1	NON			
58	62	1	OUI	4	18	
59	62	1	NON			
11	68	1	NON			
30	68	1	NON			
58	68	1	OUI	29	52	68
59	68	1	OUI	18	60	62
11	69	1	OUI	29		
30	69	1	NON			
58	69	1	OUI	68		
59	69	1	OUI	4	60	62
11	70	1	OUI	62		
30	70	1	OUI	52		
58	70	1	OUI	29		

Tableau 2. Résultats de l'agent responsable de l'apprentissage pour transiter la matrice flux M_2

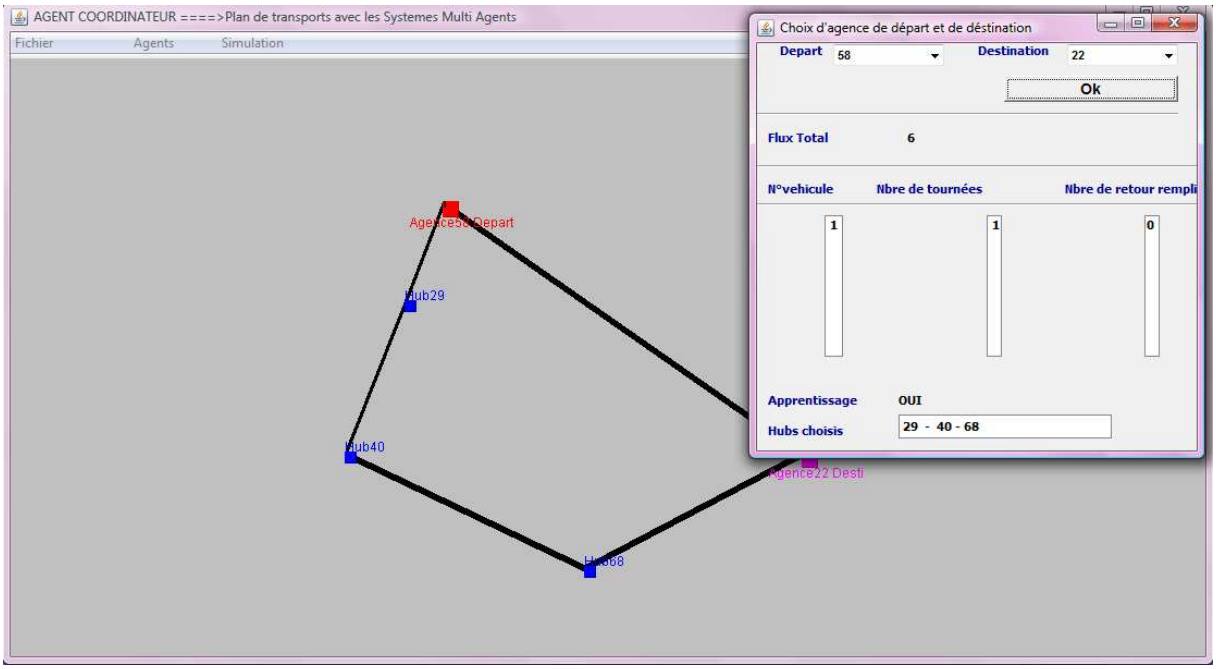

Fig.10. Affichage des résultats sur l'agent coordinateur et agent résultat

5. Influence des paramètres sur le Q-learning

Les paramètres d'entrées du Q-learning, comme mentionné précédemment, sont les entrées de l'interface graphique de l'agent responsable de l'apprentissage.

Pour le premier paramètre qui est nombre d'itérations, il a une influence sur les résultats et, surtout, sur la convergence de l'apprentissage vers la solution optimale. Dans notre cas, l'influence est aussi sur le nombre d'agents d'apprentissage lancés pour effectuer l'apprentissage. La figure 11 montre le nombre d'agents en fonction du nombre d'itérations. Nous remarquons que le nombre d'agents varie jusqu'à un certain nombre d'itérations. Le nombre ne change pas sensiblement car la solution optimale est presque atteinte à ce niveau.

Tableau.3.*Les $QValues[(i,22),a]$final du processus d'apprentissage (j=22)*

i	a							
	4	**18**	**29**	**40**	**52**	**60**	**62**	**68**
4		28722.30	39865.25	41564.16	48491.22	0.0	18337.61	32234.05
11								
18	29969.98		37024.63	37222.04	30389.12	0.0	13873.00	22150.95
22								
23								
24								
27								
29	41688.07	35140.83		19620.00	48603.11			35014.79
30								
33								
34								
35								
38								
40	49538.35	29104.29	26989.71		33657.08			25593.92
41								
45								
48								
50								
52	57130.86	24447.15	44465.99	37875.89			27699.19	16014.24
58	36493.24	40099.69	13455.00	31288.19	58143.71	56155.23	31384.98	43343.62
59								
60								
61								
62	23117.31	13810.90			35634.23			
67								
68	46700.30	17475.68	37802.25	36558.88	30119.17			
69								
78								
80								

i, l'agence courante; a, l'agence suivante sélectionnée (hub=action);j, agence de destination.

Tableau.4. *Illustration de l'agence suivante pour l'exemple du tableau3*

i	a
	22
4	62
11	
18	62
22	
23	
24	
27	
29	40
30	
33	
34	
35	
38	
40	68
41	
45	
48	
50	
52	68
58	29
59	
60	
61	
62	18
67	
68	18
69	
78	
80	

i,l'agence courante

j:destination

Le paramètre exploration est utilisé pour la sélection des actions dans le Q-learning. Comme cela est mentionné déjà plus haut, il existe plusieurs stratégies pour le choix de la prochaine action. Dans notre travail nous avons testé la stratégie ξ-gloutonne. Pour cela, le paramètre exploration est utilisé en générant un nombre aléatoire. Si le nombre est inferieur à la valeur du paramètre exploration, l'action est choisie aléatoirement et, si le nombre est supérieur, alors, dans ce cas, la stratégie gloutonne qui consiste à choisir la meilleure action est utilisée. Dans nos tests réalisés, nous avons remarqué que, tant que le nombre d'explorations est proche de 1, nous avons la possibilité de passer par toutes les actions. Dans notre travail nous avons optés pour 0.8.

Pour le paramètre amortissement, il s'agit d'un paramètre intrinsèque au Q-Learning. Idéalement, on doit donner une valeur plutôt forte (i.e. proche de 1.0) mais 0.9 est une valeur, par défaut, tout à fait convenable. En abaissant le l'amortissement, on prend le risque de "dérouter" l'apprentissage, en ignorant les actions considérées comme optimales.

En ce qui concerne le taux d'apprentissage, celui-ci symbolise, en quelque sorte, la capacité de mémorisation instantanée de l'agent. On pourrait être donc tenté de lui donner une valeur assez forte afin "d'accélérer l'apprentissage". Mais cela peut conduire l'agent à donner trop d'importance aux premiers choix qu'ils effectuent, qui peuvent s'avérer être de mauvais choix en termes de comportement optimal etc. Dans notre cas, nous avons utilisé la valeur faible de 0.05 qui rallonge un peu le temps de l'apprentissage. Ainsi, des résultats optimums ont été obtenus au bout d'un certain temps. Un essai a été effectué en faisant varier le taux d'apprentissage en en fonction du nombre de visites d'une action donnée. Comme les résultats ne diffèrent pas vraiment avec le choix de 0.05, nous avons opté pour cette valeur.

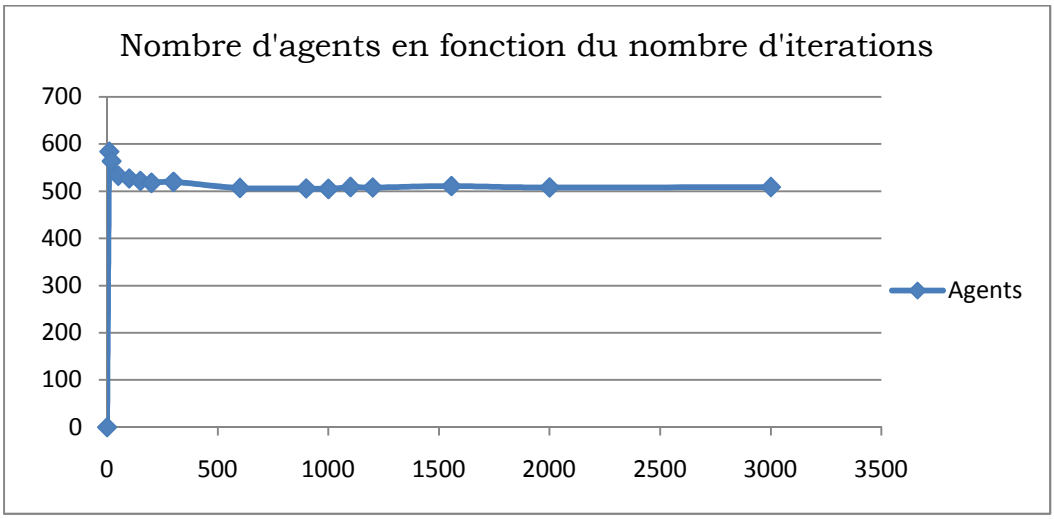

Fig.11.Nombre d'agents d'apprentissage en fonction du nombre d'itérations

6. Conclusion

L'implémentation des différents agents du système sous la plate forme Madkit nous a permis d'avoir de bon résultats en un temps de calcul raisonnable qui nous dépasse pas 3,656 secondes pour une base de données de 81 agences. L'agent Agence nous a permis de réduire le nombre de véhicules de 16872 à 4931 véhicules, tandis que l'agent responsable d'apprentissage réduit, du moins pour chaque agence de destination, un nombre de véhicules qui est égal au nombre de hubs insérés dans l'itinéraire de chaque véhicule des départs de la destination.

Le Q-Learning est très simple et sa convergence est assurée des lors que le monde est markovien et que les paramètres sont bien choisis, ce qui nous a permis d'avoir des solutions quasi-optimal. Un mauvais choix des paramètres peut conduire l'agent à avoir un comportement néfaste. Dans notre cas, les agents apprennent l'un à la suite de l'autre mais, dans le cas où les agents modifient l'environnement via leurs propres actions en même temps que l'agent apprenant, cela n'assure, en aucun cas, la convergence de la méthode.

Conclusion Générale

Dans ce travail, nous avons proposé un système multi-agents pour la résolution d'une généralisation d'un VRPTW (*Vehicle Routing Problem with Time Windows*). A la différence des approches classiques de résolution de problème, l'utilisation des Systèmes Multi-Agents nous a permis de subdiviser le problème en sous-problèmes, de le résoudre en coopérant, collaborant et communiquant plusieurs entités et surtout de développer une stratégie d'apprentissages de ces entités en utilisant Q-Learning avec des actions dynamiques. Ceci nous a permis de choisir les meilleurs tournées pour chaque agence de départ en un temps suffisamment réduit.

Dans la méthode d'apprentissage que nous avons utilisée, une synchronisation entre les agents apprenants est requise pour apprendre. Pour lancer un agent, il faut d'abord avoir les résultats de l'autre agent. Cela est dû aux contraintes de notre problème même si ceci peut nuire à l'autonomie de l'agent. D'après les tests menés avec les agents que nous avons utilisés sous la plateforme MADKIT, les réglages des paramètres du Q-learning ont un impact majeur sur la performance de ces algorithmes. En effet, le nombre d'itérations et l'exploration peuvent amener l'agent à avoir un comportement néfaste qui doit être évité. Le choix de ces paramètres doit donc être judicieux pour trouver des solutions optimales. Toutefois, des améliorations peuvent être apportées. D'une part, on peut prendre en considération des situations imprévisibles comme le cas où un véhicule tombe en panne lors du service et d'autre part, on peut traiter le même problème mais dans le cas où les données sont dynamiques et non statiques alors que dans notre cas les agences et les demandes sont connues d'avance.

Bibliographie

[1] J.Ferber., Les Systèmes Multi-Agents vers une intelligence collective.: InterEditions, 1995.

[2] N. R. Jennings, M. Wooldridge, et K. Sycara. “A roadmap of agent research and development”. In Journal of Autonomous Agents and Multi-Agent Systems, 1(1): pages 7-38, 1998.

[3] A.Rao, M.Geogreff, "An abstract architecture for rational agents," In Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, 1992, pp. 439–449.

[4] S.Russel , P.Norvig, "Artificial Intelligence: Modern Approach," in Prentice Hall., 1995.

[5] D.Danflous, « Déploiement national des Systèmes d’information multimodale : DELFI, l’exemple Allemand. Rapport technique, CERTU (centres d’études sur les réseaux, les transports, l’urbanisme et les constructions publiques), Aout 2002.

[6] MA.Kamoun, "Conception d’un système d’information pour l’aide au déplacement multimodal : Une approche multi-agents pour la recherche et la composition des itinéraires en ligne," Ecole Centrale de Lille, Thèse de Doctorat, Avril 2007.

[7] G. Weiss, "Multi-agent Systems, A Modern Approach to distributed Artificial Intelligence," in The MIT Press, 2000.

[8] O.Gutknecht, J.Ferber, "Vers une méthodologie organisationnelle de conception de system multi-agents," Rapport de Recherche du Laboratoire d’Informatique de Robotique et de Microélectronique de Montpellier N°99073, 1998.

[9] J.Ferber , M.Ferber ,O.Gutknecht, "MadKit :une architecture de plate-forme multi-agent générique," Rapport de Recherche du Laboratoire d’Informatique de Robotique et de Microélectronique de Montpellier N°00061, 1998.

[10] Y. Demazeau. Steps towards multi-agent oriented programming. In First International Workshop on Multi-Agent Systems (IWMAS’97), Boston, 1997.

[11] C.Iffenecker, J.Ferber, "Using Multi-Agent Architecture for Designing Electomechanical Products," in Actes de la conférence Avignon'92 sur les Systèmes Experts et leurs Applications, Avignon, EC2, 1992.

[12] J. M.Corera , I.Laresgoiti, N.Jennings, "Developing Industrial Multi-Agent Systems." ," in First International Conference on Multi-Agent Systems, San Francisco, V. Lesser (Ed.), MIT Press., 1995.

[13] O.Boissier, J-L.Koning, Y.Demazeau, "Using Interaction Protocols to Control Vision Systems," in IEEE International Conference on Systems, Man and Cybernetics, San Antonio., 1994.

[14] R.Weihmayer ,R.Brandau, "A Distributed AI Architecture for Customer Network Control," in GLOBECOM'90:IEEE Global Telecommunications Conference and Exhibition, San Diego, 1990.

[15] F.Guérin-Pace, H.Mathian , D.Pumain , L.Sanders ,S.Bura, "Multi-Agents Systems and the Dynamics of a Settlement System," in Simulating Societies Symposium, Siena, C. Castelfranchi (Ed.), 1993.

[16] M.Kaufmann ,M.Kodratoff Eds, "Machine Learning: An Artificial Intelligence Approach," Vol III, Morgan Kaufmann Publishers, San Mateo, 1990.

[17] R.S.Sutton ,A.G.Barto, "Reinforcement Learning : An Introduction," in MIT Press, Cambridge, MA., 1998.

[18] C.Watkins, "Learning from Delayed Rewards," University of Cambridge,England, PhD Thesis of the King's College,1989.

[19] S.Sehad, "Contribution à l'étude et au développement de modèles connexionnistes a apprentissage par renforcement: application a l'acquisition de comportements adaptatifs," Université de Montpellier II, Thèse de génie informatique et traitement du signal Montpelier 1996.

[20] C. Boutilier, 'Sequential Optimally and Coordination in Multiagent Systems', actes dans IJCAI' 99(the 16th International Joint Conference on Artificial Intelligence), Suéde, 1999.

[21] M.R Abdessemed," Proposition d'une méthode de classification dans un environnement de robotique collective", Université de Batna, Mémoire de Magister en Informatique, 2006.

[22] E.Pellerin," Méta-apprentissage des algorithmes génétiques", Université du Québec, maitrise en mathématique et informatique appliquées, 2005.

[23] J.Zhao, M.Wiering, J.Schmidhuber, "Simple Principles of Metalearning," Technical Report IDSIA, pp. 69-96, 1996.

[24] G.Weiß, ""Reader's Guide" in Distributed Artificial Intelligence Meets Machine Learning : learning in multi-agent environments," Lecture notes in artificial intelligence Gerhard Weiβ Editor, Springer-Verlag, vol. 1221 - ISBN : 3-540-62934-3 , pp. 1-10, 1997.

[25] S.S.Sian, Adaptation Based On Cooperative Learning In MAS, North Holland ed., Y. Demazeau & J-P Müller, Ed.: in Proceedings of the second workshop on Modelling Autonomous Agents in Multi-Agent World , 1991.

[26] M.Sekaran ,J.Hale S.Sen, "Learning to coordinate without sharing information," in Proceedings of the twelfth national conference on Artificial Intelligence , Seattle Washington , July 1994.

[27] M.Sekaran , S.Sen, "Multi-Agent Learning In Non Cooperative Domains," in ECAI 94,11th Europe Conference on Artificial intelligence, Amstedam, 1994.

[28] M.Mataric, "Interaction and Intelligent Behavior," Massachussetts Institute of Technology, PHD of Philosophy, May 1994.

[29] M. J.Shaw , A.B.Whinston, "Learning And Adaptation In DAI systems" in Gasser and Huhns , pp. 413-427, California, 1989.

[30] R.G.Smith , R.Davis, "Frameworks for Cooperation in Distributed Problem Solving," IEEE Transactions on Systems, Man, and Cybernetics , vol. SMC-11 , no. 1, pp. 61-70, January 1981.

[31] G.Weiß, "Learning To Coordinate Actions In Multi-Agent Systems," Proceedings of the International Joint Conference on Artificial Intelligence , pp. 311-316, August 1993.

[32] M.V.N.Prasad ,V.R.Lesser ,S.E.Lander, "Learning Organisational Roles for Negociated Search in Multi-agent System," Special Issue on Evolution and Learning in Multi-Agent Systems In the International Journal of Human-Computer Studies (IJHCS) , 1997.

[33] M.Terabe,T.Washio ,O.Katai , T.Sawaragi, "A study of organizational learning in Multi-agent systems," Distributed Artificial Intelligence Meets Machine Learning : learning in multi-agent environments - Lecture notes in artificial intelligence, vol. 1221 - ISBN : 3-540-62934-3 Gerhard Weiβ Editor, Springer-Verlag, , pp. 167-179, 1997.

[34] V.Lesser , S.Zilberstein, P.Xuan, "Communication decisions in multiagent cooperation : Model and experiments," In Proceedings of the Fifth International Conference on Autonomous Agents,ACM Press, pp. 616-623 , 2001.

[35] F. S.Melo , M.Veloso, "Learning of coordination : exploiting sparse interactions in multiagent systems," in AAMAS '09 : Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 773-780.

[36] S.Hoet, N.Sabouret, "Apprentissage par renforcement d'actes de communication dans un système multi-agent," RSTI - RIA, vol. 24, no. 2, pp. 159-188, 2010.

[37] R. Kammarti, "Approches évolutionnistes pour la résolution du 1-PDPTW statique et dynamique», université des sciences et technologies de lille, thèse de doctorat, 2006.

[38] Paolo toth,Daniel Vigo,"vehicule routing problem" , université degli Studi di Bologna, Bologna, Italie ,2002.

[39] T.K. Ralphs, L. Kopman, W.R. Pulleyblank and L.E. Trotter,"On the capacitated vehicle routing problem", Mathematical Programming 94, p343-359, 2001

[40] P. Kilby, P. Prosser, P. Shaw. "Guided Local Search for the Vehicle Routing Problem with Time Windows". Metaheuristics Advanced and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publisher, Boston, 1999.

[41] M. Gendreau,J.Y. Potvin. "Dynamic vehicle routing and dispatching" . Working paper, université de Montréal, 1998

[42] C. Jacobs-Blecha, M. Goetschalckx. "The vehicle Routing Problem with Backhauls : Properties and Solution Algorithms". School of Industrial and systems engineering, 1992-1998.

[43] R. Mechti, "Tournée de véhicules à la demande : problèmes et méthodes". Technical report, Laboratoire PRiSM, Université de Versaille-St. Quentin, France, 1995

[44] M. Fischetti, A. Lodi and P. Toth "A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem".département d'électronique et informatique, université di Padova, Italie, 1999.

[45] C. Rego et C. Roucairol. “Le problème de tournées de véhicules : Etude et Résolution Approchée”. Technical Report, inria, Février 1994.

[46] E.D. Taillard, “A heuristic column generation method for the heterogeneous fleet VRP”. RAIRO-Operations Research, 33, pages PP. 1-14, 1999.

[47] Z. Fu, R. W. Eglese and L. Li “A tabu search heuristic for the open vehicle routing problem”. LUMS working paper, 2003.

[48] C. Archetti, A. Hertz and U. Derigs “A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem”. Mathematics of Information Technology and Complex Systems, 2002

[49] A.W. Kolen,H.G. Rinnooy Kan , H.W.J.M.Trienekens, "Vehicle Routing with Time Windows," Operations Research, vol. 35, pp. 266-273, 1987.

[50] M. Desrochers,J. Desrosiers, M.Solomon, "A new Optimization Algorithm for the Vehicle Routing Problem with Time Windows," Operations Research, vol. 40, pp. 342-354, 1992.

[51] O.B.G. Madsen, B.Sorensen, K. Jornsten, "Exact solution of the Vehicle Routing and Scheduling Problem with Time Windows by Variable Splitting," in Research Report 5. IMSOR, The Technical University of Denmark, Lyngby, Denmark, 1986.

[52] O.B.G.Madsen, "Variable Splitting and Vehicle Routing Problems with Time Windows," in Preprint IB. IMSOR, The Technical University of Denmark, Lyngby, Denmark, 1988.

[53] O.B.G.Madsen, "Lagrangean Relaxation and Vehicle Routing”, Working paper, in IMSOR, The Technical University of Denmark, Lyngby, Denmark.

[54] K. Halse, "Modelling and solving complex Vehicle Routing Problems," IMSOR, The Technical University of Denmark, Lyngby, Denmark, Ph.D. thesis,1992.

[55] M.L.Fisher,K.Jornsten,O.B.G. Madsen, "Vehicle Routing with Time Windows : two optimization algorithms," Operations Research, vol. 45 (3), pp. 488-492, 1997.

[56] N. Kohl, O.B.G.Madsen, "An optimization algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation," Operations Research, vol. 45 (3), pp. 395-406, 1997.

[57] B.Larsen,J. Madsen, O.B.G.Kallehauge, "Lagrangean duality and non-differentiable optimization applied on routing with time windows-experimental results," in Internal report IMM-REP-2000-8, Department of Mathematical Modeling, Technical University of Denmark, Lyngby, Denmark, 2000.

[58] M.M.Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," Operations Research, vol. 35, , p. 254-265., pp. 254-265, 1987.

[59] J.-M.Rousseau, J.-Y.Potivin, "A parallel route building algorithm for the Vehicle Routing and Scheduling Problem with Time Windows," European Journal of Operational Research, vol. 66, pp. 331-340, 1993.

[60] R.A.Russell, "Hybrid heuristics for Vehicle Routing Problem with Time Windows," Transportation Science, vol. 29, pp. 156-166, 1995.

[61] S.Lin, "Computer solutions of the traveling salesman problem," Bell System Tech. J., vol. 44, pp. 2245-2269, 1964.

[62] J.R.Schaffer, E.K.Baker, "Solution improvement heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints," American Journal of Mathematical and Management Sciences, vol. 6, pp. 261-300, 1988.

[63] I.Or, "Traveling Salesman-Type combinatorial problems and their relation to the Logistics of Blood Banking," Department of Industrial Engineering and Management Science, Northwestern University , Evanston, IL, Ph.D. Thesis 1976.

[64] J.-M.Rousseau,J.-Y.Potvin, "An exchange heuristic for Routing Problems with Time Windows," Journal of the Operational Research Society, vol. 46, pp. 1433-1446, 1995.

[65] M.M.Solomon,E.K.Baker,J.R.Schaffer, "Vehicle Routing and Scheduling Problems with Time Window Constraints : efficient implementations of solution improvement procedures," in “Vehicle Routing : Methods and Studies” by B.L. GOLDEN and A.A. ASSAD, North Holland, Amsterdam, 1988, pp. 85-105.

[66] P.Thompson,H.Psaraftis, "Cyclic transfert algorithms for Multi-Vehicle Routing ans Scheduling Problems ," Operations Research, vol. 41, pp. 935-946, 1993.

[67] L.R.Miller, B.E.Gillett, "A heuristic algorithm for the Vehicle Dispatch Problem," Operations Research, vol. 22, pp. 340-349, 1974.

[68] G.Grabenbauer, U.Derigs, "INTIME – A new heuristic approach to the Vehicle Routing Problem with Time Windows with a Bakery Fleet Case," American Journal of Mathematical and Management Sciences, vol. 13, pp. 249-366, 1993.

[69] J.Bard, G.Kontoravdis, "Improved heuristics for the Vehicle Routing Problem withTime Windows," in Working paper, Operations Research Group, University of Texas at Austin, Austin, TX, 1992.

[70] R.A.Russell, "Hybrid heuristics for Vehicle Routing Problem with Time Windows," Transportation Science, vol. 29, pp. 156-166, 1995.

[71] W.B.Powell, M.M.Solomon, Y.A. Koskosidis, "An optimization based heuristic for Vehicle Routing and Scheduling with soft Time Window Constraints," Transportation Science, vol. 26, pp. 69-85, 1992.

[72] Y.Rochat , É.Taillard, “Probabilistic diversification and intensification in local search for Vehicle Routing”, Journal of Heuristics, Vol. 1, 1995, p. 147167.

[73] R.A. Russell, W.C. Chiang, "reactive tabu search metaheuristic for the vehicle routing problem with time windows," INFORMS Journal on Computing, pp. 417-430, 1997.

[74] E.Taillard, F.Semet, "Solving real-life Vehicle Routing Problems efficiently using tabu search," Annals of Operations Research , vol. 41, pp. 469-488, 1993.

[75] L-M.Gambardella, É.Taillard, G.Agazzi, “New Ideas Optimization”, chapitre MACS-VRPTW :A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in McGraw-Hill, 1999, pp. 63-76.

[76] M. Barkaoui, J. Berger, "A parallel hybrid genetic algorithm for the vehicle routing problem with time windows," Computers and Operations Research, vol. 31, pp. 2037-2053, 2004.

[77] F.B.Pereira,P.Machado, E.Costa, J.Tavares, "Gvr delivers it on time. in Lipo Wang Kay Chen Tan,Takeshi Furuhaski," in Jong-Hwan Kim,et Xin Yao editors,4th Asia-Pacific Conference on Simulated Evolution And Learning, 2002, pp. 745-749.

[78] H.Gehring, J.Homberger, "A two-phase hybrid metaheuristic for vehicle routing problem with time windows," European Journal of Operational Research, pp. 162:220-238, 2005.

[79] T.G.Crainic, A Le Bouthillier, "A cooperative parallel meta-heuristic for the vehicle routing problem with time windows," Computers and Operations Research, pp. 32:1685-1708, 2005.

[80] Bräysy D. Mester, "Active Guided Evolution Strategies for large-scale vehicle routing problems with time windows," Computers and Operations Research, pp. 32:1593-1614, 2005.

[81] Olli Braysy,M Gendreau, W Dullaert , "Evolutionary algorithms for vehicle routing problem with time windows," Journal of Heuristics, pp. 10:587-611, 2004.

[82] S. R.Thangiah,O.Shmygelska,W.Mennell, "An agent architecture for vehicle routing problems," in Proceedings of the 2001 ACM symposium on Applied computing (SAC '01), New York, NY (USA) : ACM Press, 2001, pp. 517–521.

[83] K. Erol, R.Kohout, "In-Time agent-based vehicle routing with a stochastic improvement heuristic," in Proceedings of the 16th national conference on Artificial intelligence and the 11th Innovative applications of artificial intelligence(AAAI'99/IAAI'99), , Menlo Park, CA (USA) : AAAI Press, 1999, pp. 864–869.

[84] P.Millot, B.Chaib-Draa, "A framework for cooperation work :an approach based on the intentionally," in journal of AI in Engineering, vol. 5, pp. 199-205, 1990.

[85] B.Chaib-Draa, "Coordination between agents in routine, familiar and unfamiliar Situations," in Rapport de recherche DUIL-RR-9401 du département d'informatique, Université de Laval ,Canada, 1994 .

[86] B.Chaib-draa, M.Sassi, " stratégies de négociation entre agents dans le domaine du transport ," in Journées Francophones en intelligence Artificielle Distribuée – Systèmes Multi-Agents (JFIADSMA'97),Editions Hermès, 1997.

[87] J.Muller ,M.Pischel,D.Schier, K.Fischer, "A model for cooperative transportation scheduling ," in V. R. LESSER & L. GASSER, Eds., Proceedings of the First International Conference on Multiagent Systems (ICMAS'95), 1995, pp. 109–116.

[88] R.G.Smith, "The contract net protocol : High-level communication and control in a distributed problem solver," IEEE Trans. on Comp., C-29(12), pp. 1104–1113, 1980.

[89] K.Ghédira, M.Kefi, "A multi-agent algorithm for vehicle routing problem with time window," in Proceedings of Urban Transport'04, Germany, 2004.

[90] F.Balbo,G.Scémama, M.Zargayouna, "A multi-agent approach for the dynamic VRPTW," in ESAW08,The 9th Annual International Workshop "Engineering Societies in the Agents World.france,2008.

[91] Lin-ning Cai, Li Zheng ,Zheng-gang Dan, "Real-Time Distribution System Based on Multi-Agent," in Proceedings of the International MultiConference of Engineers and Computer Scientists 2008, Vol I IMECS 2008, 19-21 March, Hong Kong, 2008.

[92] M. Zargayouna, B. Zeddini, "Auto-Organisation Spatio-Temporelle pour le VRPTW Dynamique ," in Actes de la 9ème Rencontre des Jeunes Chercheurs en Intelligence Artificielle (RJCIA'09), Plate-Forme de l'Afia 2009, Hammamet (Tunisie), mai 2009, pp. 247-261.

[93] R.Charton, " Des agents intelligents dans un environnement de communication multimédia : vers la conception de services adaptatifs," Université Henri Poincaré - Nancy 1, thèse de Doctorat en informatique, 2003.

[94] P.Dayan, C.Watkins, "Technical note: Q-learning," in Machine Learning, 1992, pp. 8(3): 279-292.

[95] S.Galland, "Approche multi-agents pour la conception et la construction d'un environnement de simulation en vue de l'élévation des performances des ateliers multi-sites," Université Jean Monet et Ecole Nationale Supérieure des Mines de Saint-Étienne, Thèse de doctorat en informatique 2001.

[96] B.Bauer, "UML Class Diagrams Revisited in the Context of Agent-Based Systems," in Wooldridge M., Ciancarini P.,Weiss G.,(eds),Proceeding of Agent-Oriented Software Engineering(AOSE'01), Montreal, Canada. Springer-Verlag, May 2001, pp. 1-8.

[97] H.Van Dyke Parunak,B.Bauer, J.Odell, "Representing Agent Interaction Protocols in UML," in Dans First International Workshop, AOSE 2000 on Agent-Oriented Software Engineering, Springer-Verlag, 2001, pp. 121–140.

ANNEXE

MADKIT

Madkit pour (**M**ulti **A**gents **D**éveloppement **KIT**) est une plate-forme d'exécution des systèmes multi-agents, développée en 1996 par Olivier GETKNECHT, Fabien MICHEL et Jacques FERBER au Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) de l'Université Montpellier II. Elle a été construite selon le modèle Agent/Rôle/Groupe, développé dans le contexte du projet Aalaadan.

L'architecture de cette plate-forme est basée sur 3 principes :

- ***Architecture à micro-noyau*** : le micro-noyau est un environnement d'exécution d'agents dont la taille est très réduite, gère les groupes, les rôles, le cycle de vie des agents et a responsabilité de l'aiguillage et de la distribution de messages entre agents. Le micro-noyau est en faite lui-même un agent (KernelAgent) qui est créé automatiquement à l'amorçage de la plate-forme.

- ***Agentification systématique des services :*** Tous les services de la plate-forme sont réalisés par des agents (même ceux du micronoyau étant donné que le micro-noyau est un agent). Un agent tenant plusieurs rôles peut, au fur et à mesure déléguer à de nouveaux agents certains de ses rôles.

- ***Une interface graphique componentielle :*** Chaque agent est responsable de sa propre interface graphique. Le micro-noyau ne fournit aucune interface graphique pour l'utilisateur de la plate-forme. Il est par contre possible de lui associer une application hôte qui la mettra en œuvre. Comme interface graphique de la plate-forme MadKit, il y a GBox, un environnement graphique complet de développement et de test qui aura la charge, au lancement d'un agent, de décider de la mise en place de l'objet graphique défini par ce dernier

L’agent dans Madkit est simplement décrit comme une entité autonome communicante caractérisé par une carte d’identité unique, son AgentAddress, qui joue des rôles au sein de différents groupes. Cette définition permet au développeur de choisir la représentation interne de l’agent appropriée à son domaine d’application. Il étend la classe AbstractAgent. Il hérite alors de nombreuses méthodes lui permettant de communiquer et de manipuler éventuellement une interface graphique et des flots d'entrée/sortie. L'état d'un agent est défini par quatre valeurs : création, activation, exécution et destruction. Aussi, l'agent doit implémenter les trois méthodes suivantes :

- activate : comportement de l'agent lorsqu'il est activé ;
- live : comportement de l'agent lorsqu'il vit ;
- end : comportement de l'agent lorsqu'il se termine.

Madkit associe un thread à chaque agent. Cependant, pour les simulations de systèmes comptant plusieurs milliers d'agents, il est possible d'utiliser des agents synchrones dont la politique d'exécution sera définie par des agents extérieurs.

La gestion des messages est assurée par le noyau. L’envoi et la réception de message à travers MadKit fonctionne comme le courrier postal, chaque agent possède une boîte à messages qui fonctionne comme une file FIFO, lorsqu’un agent envoie un message, il dépose en fait ce message dans la boite de l’agent à qui il veut envoyer. Madkit ne fournit qu’une classe de message qu’il faut dériver pour stocker les valeurs que l’on souhaite.

MadKit est un environnement de travail très intéressant dans le cadre des Systèmes Multi-Agents et de l'intelligence artificielle.sa force réside dans le fait qu'elle n'impose quasiment aucune contrainte sur l'architecture des agents. D'autre part, à partir de la version 4.0, MadKit se présente sous une forme totalement modulaire, ou plus exactement sous la forme d'un ensemble de plugins. Il est ainsi facile d'ajouter de nouvelles fonctionnalités à MadKit par l'intermédiaire de nouveaux plugins. Il est aussi possible de construire des versions allégées de MadKit en n'utilisant que les plugins désirés.

AUML

L'unification des différentes méthodologies de développement des objets a abouti à l'émergence du langage de modélisation des systèmes à base d'objets UML permettant la spécification, la construction, la visualisation et la documentation de logiciels.

UML fournit une base insuffisante pour modéliser les agents et les systèmes multi-agents. Fondamentalement, c'est dû à deux raisons: *Premièrement* comparé aux objets, les agents sont proactifs et autonomes. *Deuxièmement,* Les agents interagissent les uns avec les autres. C'est ainsi que *Odell* et d'autres chercheurs ont proposé des extensions à apporter à UML afin de modéliser et de spécifier des agents [O'dell et al. 00]. Ils l'ont appelé AUML(Agent Unified Modelling Language). Ces extensions permettent : de représenter des protocoles d'interactions entre agents comme le *Contrat net protocol,* de définir un concept de rôle comme une extension des acteurs dans les diagrammes d'interaction d'UML, qui permet ainsi de représenter un agent remplissant plusieurs rôles et d'intégrer l'aspect dynamique dans les classes d'agents.

Un diagramme n'est pas un modèle mais une représentation graphique de quelques éléments du modèle. Les diagrammes constituent l'expression visuelle et graphique. Il existe plusieurs types de diagrammes dynamiques dans UML : les diagrammes de déploiement, de collaboration, de séquences, de cas d'utilisation, d'états-transitions et d'activités. Un diagramme ne montre pas toutes les informations, plusieurs diagrammes sont nécessaires pour illustrer le modèle complet.

Des diagrammes ont été modifiés et améliorés pour donner une description dynamique complète des agents et leurs interactions, comme les diagrammes de séquences qui portent maintenant le nom diagramme de protocole [O'dell et al. 00] et correspondent à la représentation des protocoles d'interaction. Les diagrammes de classes ont aussi été modifiés. Il ne faut pas oublier qu'un agent diffère d'un objet par conséquent le diagramme de classes qui permet de représenter les classes constituant le système sera

aussi modifié. Le tableau 1 résume les concepts pris en compte dans la notation AUM

Concepts Modélisés	Notation AUML
Agent	Diagramme de classes : stéréotype *"Agent"* description d'états, actions, méthodes, capabilité, protocoles supportés, contraintes
Rôle	Diagramme de classes : stéréotype *"Role"*
Interaction	Spécification d'un template UML Diagramme d'interaction étendu d'une branche XOR pour prendre en compte l'indéterminisme
Message	Message asynchrone : flèche simple Message synchrone : flèche remplie
Action	Décrite au niveau du diagramme de séquences avec un rectangle aux angles arrondis

Tab.1. La notation AUML

Rappelons brièvement à quoi servent quelques représentations. Les diagrammes de séquence (protocole) décrivent les protocoles d'interaction utilisés par les agents. Ils exploitent la richesse des communications entre agents qui peuvent être asynchrone ou synchrone. Trois types dede branchements sont définit dans les diagrammes de séquence afin de prendre en compte l'indéterminisme du comportement d'un agent. Ces branchements voir figure 1) peuvent être AND (ET logique fig.1.a), OR (OU logique fig.1.b) XOR (OU exclusif fig.1.C).Le connecteur ET signifie que les messages doivent être envoyés concurremment, le connecteur OU, zéro ou plusieurs messages est choisis et XOR signifient qu'une décision entre plusieurs messages doit être faites.

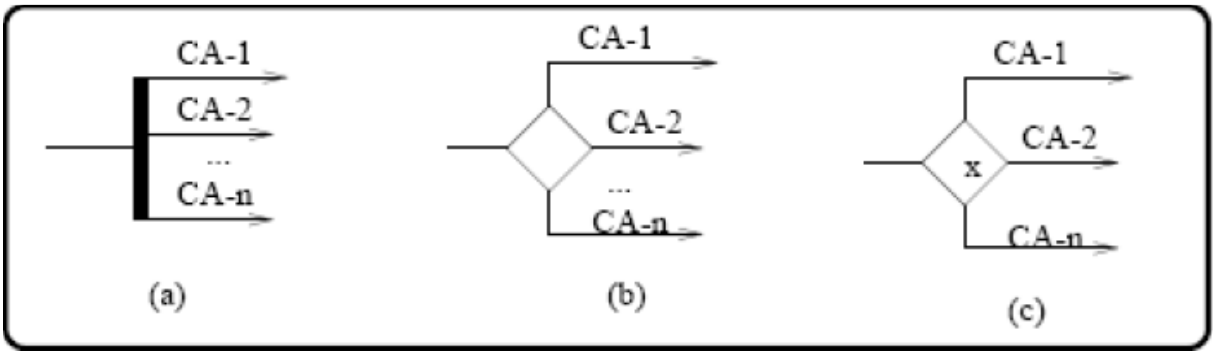

Fig.1.Les branchements définis dans AUML

Les diagrammes de classe modélisent les agents, une classe agent est une classe UML qui peut aussi spécifier un ensemble de rôles, des actions, des capacités, une description des services, un ensemble de protocoles, des contraintes et une référence vers un agent-head automata.

Les diagrammes d'activité traitent l'activité du système et décrivent les échanges les activités. Les diagrammes d'état ressemblent aux diagrammes d'activités si ce n'est qu'il ne s'agit pas de modéliser des activités mais des états du système. Les diagrammes d'état sont utilisés pour représenter le comportement du système. Les diagrammes de cas sont utilisés pour définir des cas d'utilisation du système et donnent une analyse du système.

Le langage de modélisation Agent UML fait maintenant partie de l'environnement du concepteur de systèmes multi-agents. Agent UML en est toutefois à ses débuts et de nombreux efforts doivent être menés afin d'adapter complètement Agent UML au domaine agent. A l'heure actuelle, il existe peu de travaux sur Agent UML et encore moins une application réelle d'Agent UML.