Année : **2008**

THÈSE

présentée à

L'U.F.R. DES SCIENCES ET TECHNIQUES
DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

pour obtenir le

GRADE DE DOCTEUR DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

en **AUTOMATIQUE**
(Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques)

Synthèse d'agents adaptatifs et coopératifs par apprentissage par renforcement

Application à la commande d'un système distribué de micromanipulation

par
Laëtitia MATIGNON

Soutenue le 4 Décembre 2008 devant la Commission d'Examen :

Rapporteurs :

M. **Abdel-Illah MOUADDIB**	Professeur, Université de Caen Basse-Normandie–GREYC
M. **Joël QUINQUETON**	Professeur, Université Paul Valéry–LIRMM, Montpellier III

Examinateurs :

M. **François CHARPILLET**	Directeur de recherche, LORIA–INRIA, Nancy I
M. **Philippe GAUSSIER**	Professeur, Université de Cergy-Pontoise–ETIS–ENSEA
M. **Yves-André CHAPUIS**	Maître de Conférences, Université Louis Pasteur–InESS, Strasbourg
Mme **Nadine LEFORT-PIAT**	Professeur, E.N.S.M.M. Besançon
M. **Guillaume LAURENT**	Maître de Conférences, E.N.S.M.M. Besançon

A mon père.

Remerciements

Le manuscrit que vous avez entre les mains est le résultat d'un travail ponctué de rencontres, d'échanges et parfois de doutes. Ce travail n'aurait pu arriver à terme sans l'aide de nombreuses personnes, et en premier lieu ma famille qui m'a soutenu tout au long de ces années d'étude. Je remercie donc particulièrement mes parents et mon frère qui m'ont encouragé à poursuivre dans cette voie.

Dans un second temps, mes remerciements et ma gratitude vont à mes deux directeurs de thèse, Madame Nadine Le Fort-Piat, professeur à l'École Nationale Supérieure de Mécanique et des Microtechniques (ENSMM), et Monsieur Guillaume Laurent, maître de Conférences à l'ENSMM. J'ai particulièrement apprécié tout au long de ce travail le très bon encadrement, la disponibilité, le soutien et la bonne humeur qui ont caractérisé notre collaboration. J'aurais difficilement pu souhaiter meilleures relations avec mes encadrants et cette thèse ne serait pas ce qu'elle est sans leur confiance et leur aide scientifique.

Ces travaux de thèse ont été réalisés au département Automatique et Systèmes Micro-Mécatroniques de l'institut FEMTO-ST. Je tiens donc aussi à remercier les directeurs successifs, Monsieur Alain Bourjault puis Monsieur Nicolas Chaillet, pour m'avoir permis de réaliser mes travaux de thèse au sein de ce laboratoire. L'excellente ambiance de travail qui y règne est un atout incontestable pour le bon déroulement de travaux de thèse et c'est pourquoi je remercie l'ensemble des personnels du département, compagnons de travail parfois fonctionnaires mais souvent précaires, certains déjà partis d'autres arrivant tout juste. Je remercie particulièrement Messieurs Joël Agnus et David Guibert pour leur soutien technique lors de ces derniers mois de travail.

Une autre assemblée que je me dois de remercier est celle qui a participé à mon comité de thèse. Tout d'abord, Messieurs Abdel-Illah Mouaddib et Joel Quinqueton, qui m'ont fait l'honneur de rapporter ce travail. Je tiens également à remercier Messieurs Philippe Gaussier, François Charpillet et Yves-André Chapuis pour avoir accepté de faire parti du jury. Je remercie particulièrement Monsieur Yves-André Chapuis pour avoir mis à notre

disposition le prototype de surface active.

Enfin, je ne terminerai pas ces remerciements sans avoir une pensée sympathique pour mes camarades à raquettes « fous du volant » et mes compagnons de « rueda » qui m'ont permis de me défouler sur les terrains ou sur la piste ...

Laëtitia

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Glossaire des notations

Notations introduites au chapitre 2

s	état du système
$\mathcal{S}$	ensemble fini des états
a	commande (ou action)
$\mathcal{A}$	ensemble fini des commandes
r	récompense immédiate (scalaire)
R	fonction de récompense
T	fonction de transition entre états
α	coefficient d'apprentissage
γ	coefficient d'atténuation
G	gain (somme γ-pondérée des récompenses futures)
V	fonction de valeur définie sur les états
Q	fonction de valeur définie sur les couples état-action
π	politique (distribution de probabilités sur les couples état-action)
V^*	fonction de valeur optimale
Q^*	fonction d'utilité optimale
π^*	politique optimale
ϵ	taux d'exploration de ϵ-greedy
τ	paramètre de température dans la méthode softmax
λ	taux de diffusion
e	fonction d'éligibilité définie sur les couples état-action
t	temps ou période d'échantillonnage
$\mathbb{R}$	ensemble des réels

Notations introduites au chapitre 3

Q_i	valeurs initiales de la fonction Q
Q_∞	borne de Q_i
d	distance minimax en nombre de transitions
d_g	distance minimax au but
d_M	distance manhattan
F	fonction de récompense additionnelle
h	fonction heuristique définie sur les états
φ	estimateur de progrès
β	amplitude des fonctions d'influence gaussienne
δ	niveau d'exploration loin de la cible dans les fonctions d'influence gaussienne
σ	écart-type des fonctions d'influence gaussienne

Notations introduites au chapitre 4

m	nombre d'agents
$\mathcal{A}_i$	ensemble des actions pour l'agent i
$\boldsymbol{\mathcal{A}}$	ensemble des actions jointes
$\boldsymbol{\mathcal{A}}_{-i}$	ensemble des actions jointes de tous les agents sauf de l'agent i
$\Delta(\mathcal{A}_i)$	ensemble des stratégies pour l'agent i dans un jeu matriciel
$\Delta(\mathcal{S}, \mathcal{A}_i)$	ensemble des stratégies pour l'agent i dans un jeu de Markov
a_i	action ou stratégie pure de l'agent i
$\boldsymbol{a}$	action jointe ou stratégie pure jointe
$\boldsymbol{a}_{-i}$	stratégie pure jointe de tous les agents sauf de l'agent i
R_i	fonction de récompense de l'agent i
$\langle a_i, \boldsymbol{a}_{-i} \rangle$	stratégie pure jointe où l'agent i choisit a_i et les autres suivent $\boldsymbol{a}_{-i}$
π_i	stratégie de l'agent i
$\boldsymbol{\pi}_{-i}$	stratégie jointe des agents excepté l'agent i
$\boldsymbol{\pi}$	stratégie jointe
$\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$	stratégie jointe où l'agent i choisit pi_i et les autres suivent $\boldsymbol{\pi}_{-i}$
Ω_i	ensemble des observations de l'agent i
$\boldsymbol{\Omega}$	ensemble des observations jointes des agents
o_i	observation de l'agent i

$\mathcal{O}$	fonction d'observation
u_i	gain espéré d'un agent i dans un jeu matriciel
U_i	espérance de récompense d'un agent i dans un jeu de Markov

Notations introduites au chapitre 5

$\mathsf{Q}^{\boldsymbol{\pi}}$	fonction de valeur globale sous une politique jointe $\boldsymbol{\pi}$
$\mathsf{Q}_i^{\boldsymbol{\pi}}$	fonction de valeur locale d'un agent i sous une politique jointe $\boldsymbol{\pi}$
Q^*	fonction de valeur globale optimale d'une politique jointe optimale
$\boldsymbol{a}^*$	actions jointes optimales

Notations introduites au chapitre 6

Q_i	fonction de valeur individuelle d'un agent i
$\mathsf{Q}_{max,i}$	fonction de valeur individuelle d'un agent optimiste i
C_i	table pour mémoriser le nombre de fois qu'une action a été choisie
$\mathsf{C}_{Q_{max},i}$	table pour mémoriser le nombre de fois où la récompense maximale a été reçue après avoir effectué une action
F_i	fréquence d'occurrence d'un agent i
E_i	fonction d'évaluation d'une action d'un agent i
c	paramètre de réglage de l'importance de l'heuristique dans la méthode FMQ
δ_{perdre}	coefficient d'apprentissage pour l'ajustement de la politique
δ_{gagner}	coefficient d'apprentissage pour l'ajustement de la politique
$\bar{\pi}$	politique moyenne

Notations introduites au chapitre 7

β	coefficient d'apprentissage pour la décroissance des Q-valeurs
ψ	coefficient d'exploration globale
α_f	coefficient d'apprentissage de la fréquence d'occurrence
G_i	fréquence distante d'un agent i
α_g	coefficient d'apprentissage de la fréquence distante

Notations introduites au chapitre 8

F_D	force de traînée ou force de convoyage

F_L	force de portance
F_V	force de viscosité
ρ	densité d'un fluide
η	coefficient de viscosité du fluide
C_D	coefficient de traînée
C_L	coefficient de portance
S	surface frontale de référence
$\vec{f}_{i,n}$	force de convoyage élémentaire exercée par un jet d'air de l'orifice i sur une face n d'un polygone
$v_{i,n}$	vitesse relative du jet d'air
r_{acc}	récompense accumulée au cours d'une macro-action
γ_{acc}	coefficient d'atténuation accumulé au cours d'une macro-action
$\mathcal{S}_{i,non-controle}$	ensemble des observations « non-contrôlées » d'un agent i

Glossaire des abréviations

AAJ	agent percevant les actions jointes
AI	agent indépendant
ANR	Agence Nationale de la Recherche
BOSAR	Bibliothèque d'Outils pour Matlab-Simulink pour l'Apprentissage par Renforcement
FMQ	Frequency Maximum Q-value
GLIE	greedy in the limit with infinite exploration
IAD	intelligence artificielle distribuée
PDM	processus décisionnel markovien
PDSM	processus décisionnel semi-markovien
PHC	Policy Hill Climbing
PIO	actions individuelles potentiellement optimales
SMA	système multi-agent
SOoN	Swing between Optimistic or Neutral
WoLF	Win or Learn Fast

Chapitre 1

Introduction

Cette thèse se situe à la confluence de deux champs de recherches : l'apprentissage par renforcement d'une part et les systèmes multiagents d'autre part. Ces deux branches de l'intelligence artificielle sont présentées dans cette introduction, avant de s'intéresser au problème autour duquel va se construire notre travail : comment réunir ces deux approches dans un contexte coopératif et quelles problématiques vont en résulter ? Les objectifs et le cadre applicatif dans lequel se situe cette thèse sont aussi introduits. Finalement, le plan d'organisation de ce mémoire est établi.

1.1 Contexte

1.1.1 Apprentissage par renforcement

Prenons tout d'abord un exemple simple de comportement pouvant être modélisé par un processus de renforcement. Supposons que vous appreniez seul à faire du ski. Il est fort probable que la première descente d'une piste soit périlleuse avec, par exemple, une chute à chaque virage et d'éventuelles collisions malencontreuses. Avec un peu de chance, certains virages seront tout de même réussis et vous permettront de comprendre la technique du « chasse-neige ». Après quelques descentes, vous commencerez à mieux maîtriser cette technique, et vous serez probablement capable de rattraper un déséquilibre suite à une bosse mal négociée. Au bout de plusieurs semaines de pratique, le slalom n'aura plus aucun secret pour vous et peut-être même que vos skis commenceront à rester parallèle. Vous serez de plus capable de vous adapter à de nouvelles pistes. Vous aurez donc appris à maîtriser la technique du ski qui était au départ inconnue pour vous.

Cette faculté est une capacité essentielle pour trouver une stratégie d'action dans un environnement ou une situation complètement ou partiellement inconnus. L'apprentissage est le résultat d'un ensemble d'expériences qui se sont soldées par des réussites (virages bien manoeuvrés) ou des échecs (chutes, sapins). Ce critère de satisfaction est à la base de cet apprentissage par essais et erreurs.

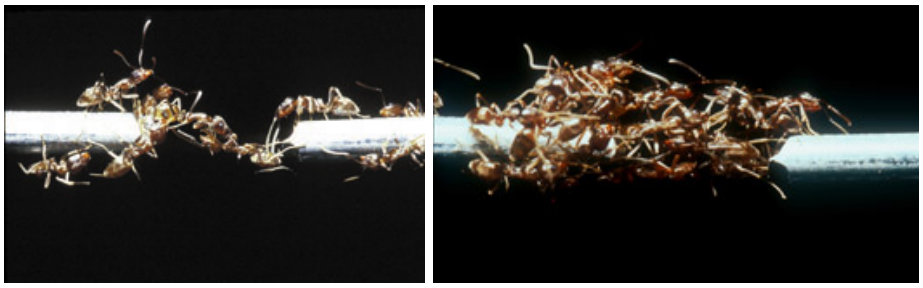


FIGURE 1.1 – Formation d'un pont vivant chez la fourmi d'Argentine « Linepithema humile » (illustrations issues de la photothèque du CNRS).

Développée depuis les années 1980, la théorie de l'apprentissage par renforcement a pour objectif de reproduire ce mécanisme. C'est une méthode de contrôle adaptative qui ne nécessite pas de connaître le système mais simplement un critère de satisfaction. Un agent interagissant avec un environnement teste différentes actions et doit apprendre un comportement grâce à un système de récompenses et de punitions. Ainsi, on ne dit pas explicitement à l'agent ce qu'il doit faire mais une évaluation de sa situation lui est donnée.

Le comportement optimal d'un agent se définit comme la séquence d'actions qui va lui permettre d'optimiser un critère sur le long terme. Par exemple, le critère à maximiser peut être la somme pondérée des récompenses futures. Le comportement est appris par l'expérience et correspond à un apprentissage par essais et erreurs.

De nombreux problèmes peuvent s'exprimer dans le cadre de l'apprentissage par renforcement. Le formalisme associé est celui des **processus décisionnels de Markov**. Les méthodes d'apprentissage par renforcement ne nécessitent aucune connaissance *a priori* sur la dynamique du système. De plus, les processus commandés peuvent être non linéaires et stochastiques.

Les domaines d'application de l'apprentissage par renforcement sont par exemple la robotique, les jeux vidéo où l'ordinateur va progresser en même temps que le joueur, ou encore la microrobotique et plus particulièrement la micromanipulation. En effet, les nombreuses difficultés rencontrées à l'échelle du micromonde recoupent les principaux intérêts de l'apprentissage. On peut notamment relever à ces échelles :

- un comportement dynamique souvent non ou mal connu étant donnés les problèmes liés à la modélisation des interactions. En effet, les forces de surface et de contact sont prépondérantes à l'échelle micrométrique et complexes à modéliser,
- un comportement en général non linéaire,
- un comportement pouvant être considéré comme stochastique.

Néanmoins, la durée de l'apprentissage peut être longue. En effet, l'agent adapte son comportement en fonction des récompenses ou punitions reçues, mais, au début de sa phase d'apprentissage, il ne possède généralement pas de connaissances. Il explore alors de manière plus ou moins aléatoire son environnement. Une de nos contributions est de

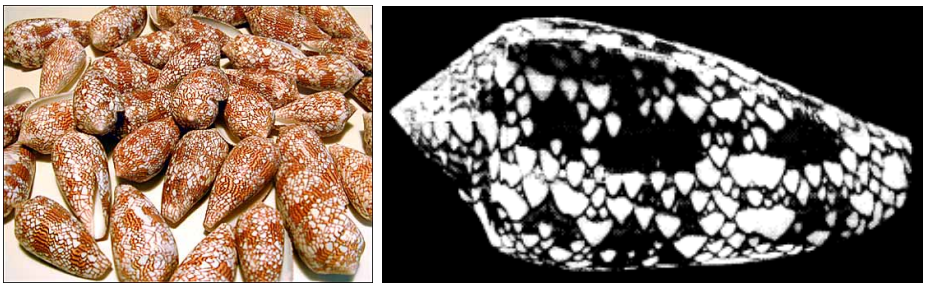


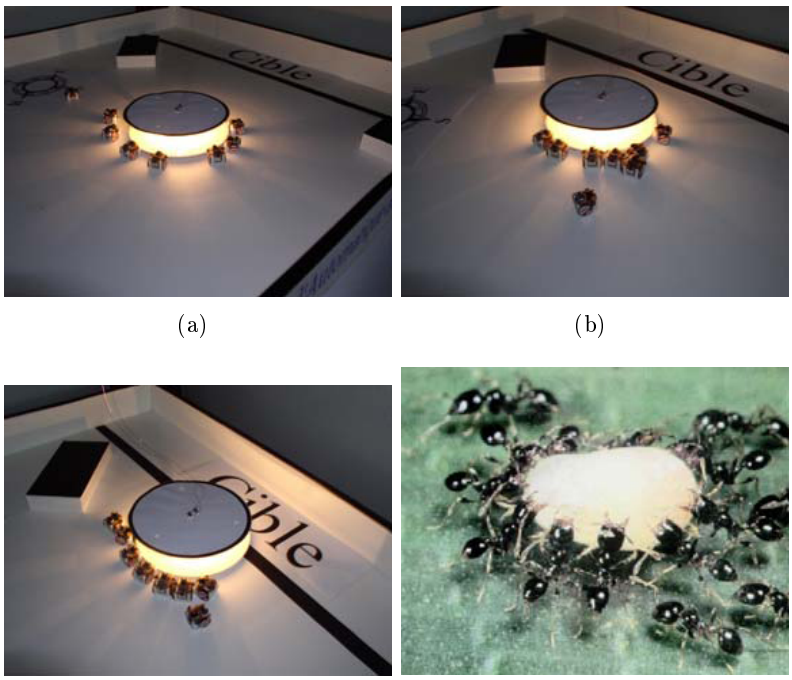
FIGURE 1.2 – Apparition spontanée sur des coquillages de motifs complexes similaires à certains automates cellulaires [Wol02].

permettre d'introduire de la connaissance *a priori* que l'on aurait sur une tâche donnée afin d'accélérer l'apprentissage de l'agent. Celui-ci sera ainsi guidé lors de l'exploration de son environnement.

1.1.2 Systèmes multiagents

En observant les sociétés d'insectes (cf. figure 1.1), notre système immunitaire, le fonctionnement du cerveau ou encore les automates cellulaires (cf. figure 1.2) tels que le **jeu de la vie**, imaginé par J. H. Conway en 1970, on s'aperçoit que le regroupement d'entités ou d'individus dotés d'une intelligence peu évoluée peut aboutir à l'émergence d'un comportement collectif beaucoup plus sophistiqué. Ce comportement d'ensemble plus complexe est le produit de l'interaction de plusieurs entités autonomes qui associent leurs capacités limitées pour accroître leurs possibilités. C'est sur la base de telles idées que se sont développées ces dernières décennies des travaux sur la thématique des systèmes multiagents. La mise en interaction d'entités autonomes relativement simples et la complexité qui en émerge ont donné naissance à une nouvelle discipline : l'intelligence artificielle distribuée. Cette vision décentralisée a été très vite confortée par le développement des réseaux informatiques et a rapidement trouvé de multiples applications, notamment en robotique distribuée. La robotique distribuée recouvre deux domaines distincts qui résument les différentes utilisations possibles des systèmes multiagents :

- la robotique mobile où plusieurs robots doivent coordonner leurs déplacements et coopérer pour accomplir une tâche commune telle que le nettoyage d'une surface, le rangement de marchandises ou l'exploration d'un environnement. Les systèmes multiagents sont ici une **nécessité** car plusieurs entités sont physiquement indépendantes. Les phénomènes d'émergence et de coopération sont notamment étudiés à travers l'analogie entre les comportements d'animaux sociaux et un groupe de robots mobiles. Par exemple, des travaux ont été menés au laboratoire en minirobotique mobile coopérative [Ado05]. Ces applications mettent en oeuvre un groupe d'entités robotiques minimalistes pouvant réaliser une tâche coopérative telle que la poussée d'objets lourds à la manière des fourmis (*cf.* figure 1.3),



(a) (b)

(c) (d)

FIGURE 1.3 – Des minirobots ALICE ($21 \times 19 \times 18mm$) [CES02] poussent un objet cylindrique d'une position initiale (a) vers la cible à atteindre (c) [Ado05]. Le déplacement de l'objet nécessite une quantité minimum de minirobots. d) Tâche coopérative de transport d'objets par un groupe de fourmis.

– la robotique cellulaire où chaque composant d'un robot peut être considéré comme un agent. La réalisation du mouvement est alors le résultat de la coordination de l'ensemble de ces agents (par exemple, chaque élément d'un bras manipulateur ou chaque doigt d'un préhenseur peut être considéré comme un agent). Dans ce cas, les systèmes multiagents sont utilisés comme un **moyen** pour améliorer la vitesse de résolution ou d'exécution du système en le divisant en plusieurs entités afin de diminuer sa complexité. Notre application à la commande d'un **système distribué de micromanipulation** se place dans ce cadre (*cf.* §1.2.3). En effet, un système distribué de micromanipulation est une matrice de microactionneurs, chacun pouvant être considéré comme un agent. Un exemple est donné à la figure 1.4.

Les systèmes multiagents permettent d'une part de mettre en oeuvre des systèmes robustes grâce à la redondance des agents, et d'autre part d'étudier les phénomènes sociaux qui y prennent place tels que la collaboration, la coordination, ...

Dans un système multiagent, les objectifs des agents peuvent être de diverses natures : identiques, conflictuels ou neutres. **Notre intérêt dans ce mémoire se porte vers les situations d'interaction faisant appel à la coopération des agents**. Dans un

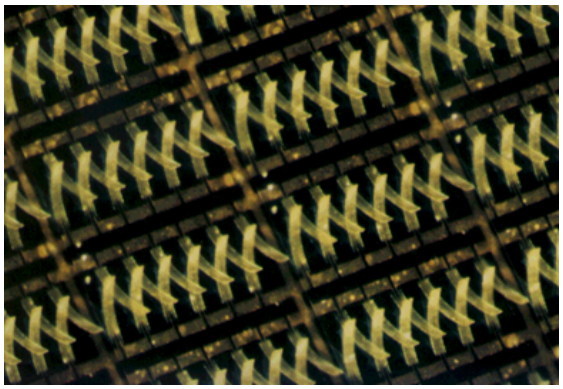


FIGURE 1.4 – Système distribué de micromanipulation « à contact » constitué de microactionneurs en polyimide dont l'effet thermique reproduit le mouvement des cils des organismes vivants [AOTF94].

système multiagent coopératif, les agents forment une équipe et donc partagent un but commun. Un exemple évident est une équipe de football dans laquelle les joueurs ont le même objectif qui est de gagner la partie. Cependant, pour atteindre cet objectif, les joueurs doivent coordonner leurs actions individuelles pour que celles-ci résultent en une action jointe optimale pour le groupe (ici, mettre un but). La coordination est un mécanisme complexe auquel les systèmes multiagents coopératifs font appel.

1.1.3 Union des deux approches

D'une part, les systèmes multiagents permettent de résoudre certains problèmes en adoptant une vision décentralisée et en utilisant des agents autonomes d'intelligence réduite en interaction. D'autre part, l'apprentissage par renforcement permet à un agent d'apprendre un comportement dans un environnement incertain et inconnu. L'union de ces deux outils permettrait alors d'associer leurs avantages : des agents simples et autonomes apprenant à résoudre de manière décentralisée des problèmes complexes en s'adaptant à ceux-ci.

1.2 Orientations de recherche

1.2.1 Problématique

Notre objectif premier est donc d'utiliser conjointement l'apprentissage par renforcement et les systèmes multiagents. Cependant, la combinaison de ces deux méthodes soulève plusieurs problèmes. Les systèmes multiagents sont caractérisés par une exécution décentralisée, donc chaque agent décide individuellement des actions qu'il souhaite entreprendre mais tous les agents contribuent à l'évolution du système au niveau global. Le comportement à adopter en présence d'autres agents va donc dépendre des congénères. Par exemple, dans une équipe de football, il ne suffit pas de réunir de bons joueurs

FIGURE 1.5 – Tâche nécessitant la coopération des robots [HHK+02].

pour obtenir une équipe performante. Les joueurs, mêmes excellents, doivent apprendre à s'adapter aux comportements des autres, par exemple en coordonnant leurs actions pour organiser leur jeu, se faire des passes et définir les rôles de chacun. C'est ici la clé de l'apprentissage d'agents autonomes dans un système coopératif.

Ainsi, lorsque plusieurs agents se trouvent dans le même environnement, le problème de leur apprentissage se complexifie car il est nécessaire de coordonner les agents pour que les performances soient acceptables. La complexité de coordination provient tout d'abord du nombre d'interactions possibles qui croît exponentiellement avec le nombre d'agents. Une autre raison est que la décision d'un agent peut influencer l'ensemble du système et donc ses congénères. Enfin, la coordination est complexe car les agents ont rarement accès à l'ensemble des informations sur l'environnement. Nous allons préciser cette problématique.

Un agent a accès à des informations sur son environnement appelées perceptions. Il exploite ses perceptions pour raisonner et décider de façon autonome comment agir selon chaque perception. Avec une vision décentralisée, chaque agent récupère ses perceptions à un niveau local. Si les perceptions ne donnent accès à l'agent qu'à une information incomplète, la prise de décision sera alors plus complexe. Un agent peut avoir des informations limitées à différents niveaux :

- prenons l'exemple d'une équipe de robots footballeurs. Si les robots sont équipés d'une simple caméra, ils ont alors un champ de vision limité et chaque robot ne percevra pas précisément ses coéquipiers éloignés. En quelque sorte, un robot pourra déterminer dans quelle partie du terrain sont ses coéquipiers, mais il n'aura pas accès avec précision à cette information. On parle alors d'*observabilité partielle*. Il sera donc difficile pour un robot footballeur de se coordonner avec ses coéquipiers éloignés en faisant une passe précise par exemple,
- la connaissance des actions des autres peut aussi être un paramètre important pour la prise de décision individuelle. Prenons le cas des trois robots de la figure 1.5 qui portent au bout de leur bras une même plaque. Supposons que leur objectif est d'orienter cette plaque à l'horizontale mais que les robots ne peuvent pas communiquer entre eux. Étant donnée l'inclinaison de la plaque dans la situation de la figure 1.5, le robot du fond va alors descendre son bras. Mais en même temps, les deux robots de devant vont monter les leurs. La plaque n'est toujours pas horizontale et l'objectif n'est pas atteint. La difficulté pour chaque agent est alors de s'adapter à la stratégie des autres sans savoir exactement ce que les autres décident

mais en observant l'effet de leurs actions sur leur environnement. Ici les actions combinées des agents ont un effet sur l'inclinaison de la plaque qui est observable. Cette adaptation se complique lorsque de nombreux agents sont impliqués,
- la communication entre les agents est un paramètre important qui peut améliorer la résolution d'une tâche coopérative lorsque les perceptions sont limitées. Si les minirobots peuvent s'envoyer des informations, les capacités perceptives de chacun seront améliorées, et par là même, leur coordination. Toutefois, la communication ne résout pas complètement les difficultés de coordination.

Le choix des différentes informations disponibles par chaque agent est donc vaste. Il doit cependant être précisé car le cadre formel choisi pour l'étude va en dépendre.

1.2.2 Objectifs

Dans cette thèse, nous nous intéressons à la **coopération d'agents adaptatifs**. En particulier, nous tentons de répondre à la question suivante : des agents adaptatifs et coopératifs peuvent-ils se coordonner sans communiquer mais en supposant que l'observabilité individuelle de l'environnement est totale ?

Les agents mis en oeuvre et étudiés dans ce mémoire sont donc les suivants :
- ils apprennent par apprentissage par renforcement,
- ils n'ont pas de modèle de la dynamique du système,
- ils ont des buts compatibles et sont donc coopératifs,
- ils ne communiquent pas,
- ils ont une observabilité individuelle totale du système,
- ils sont *indépendants*, *i.e.*ils n'ont pas accès aux actions des autres mais simplement à leur action individuelle.

Le coeur de notre travail va donc être de développer des méthodes décentralisées d'apprentissage par renforcement, pour de tels agents autonomes et indépendants, situés dans le cadre de systèmes multiagents coopératifs. Le formalisme choisi pour ces méthodes est celui des jeux de Markov d'équipe. Une de nos contributions sera donc de proposer une analyse des enjeux soulevés par ce problème. Suite à une étude des méthodes existantes dans ce cadre et de leurs difficultés face à ces enjeux, deux algorithmes seront alors exposés pour l'apprentissage par renforcement d'agents autonomes, indépendants et coopératifs. Une extension de ces travaux à un cas d'observabilités partielles sera enfin développée et appliquée à la commande décentralisée d'un système distribué de micromanipulation, appelé *smart surface*.

1.2.3 Cadre applicatif

L'application proposée est la commande d'un système distribué de micromanipulation, appelé *smart surface*, qui se place dans le cadre d'un projet de l'agence nationale de la recherche (ANR PSIROB) (ANR_06_ROBO_009). Ce projet s'est donné pour objectif « la conception, le développement et le contrôle d'un système micro-robotique distribué pour le convoyage, le positionnement et le tri de micro-pièces à l'échelle méso-

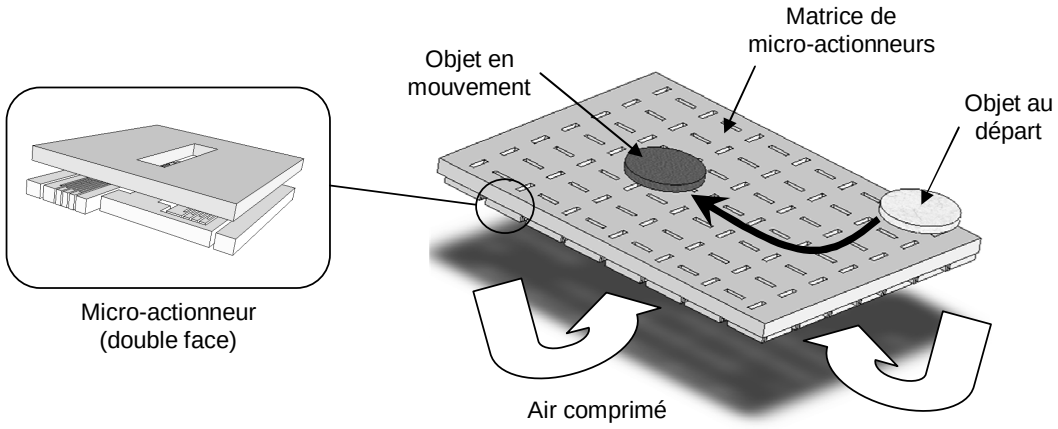


FIGURE 1.6 – Dispositif de micromanipulation basé sur une matrice de microactionneurs pneumatiques [FCMF06].

scopique (μm au mm) ». Le système choisi est une matrice de micro-actionneurs pneumatiques distribués, dont une illustration est donnée à la figure 1.6. Ce dispositif est expliqué plus en détail au chapitre 8 de ce mémoire. Les applications visées par un tel système sont les fonctions d'alimentation et de convoyage de composants dans le cadre de la micromanipulation ou du micro-assemblage automatisé de mini et microproduits. Ceci constitue actuellement un challenge important lié aux dimensions de ces pièces.

Ce projet ANR rassemble plusieurs laboratoires et ses directions de recherches peuvent être classées en 4 thématiques :

- la première concerne la conception et la fabrication de la matrice intégrant les micro-actionneurs et les capteurs (FEMTO-ST, InESS, LIMMS),
- la deuxième s'intéresse au traitement et à la gestion distribuée des informations dans le système. Il faut alors gérer les communications de proche en proche (LAAS, LIFC),
- la troisième concerne la commande décentralisée et adaptative de la surface (FEMTO-ST),
- la dernière porte sur les méthodes de conception et de validation (LIFC, InESS).

C'est donc dans la troisième thématique que les travaux de cette thèse peuvent s'appliquer. L'objectif est « la conception d'algorithmes d'apprentissage permettant la commande décentralisée de la *smart surface* et la coopération entre les actionneurs pour convoyer et positionner les micropièces ». Chaque actionneur est considéré comme un agent à capacité limitée dont les actions se résument à orienter un jet d'air à la surface de la matrice. La matrice distribuée est donc un système multiagent dans lequel le résultat complexe de convoyage ou de positionnement émerge de l'interaction de jets d'air élémentaires produit par des actionneurs indépendants.

1.3 Plan

Ce mémoire se compose de neuf chapitres. Outre l'introduction et la conclusion, deux parties se distinguent : la première, constituée des chapitres 2 et 3, concerne l'apprentissage par renforcement mono-agent ; la seconde, constituée des chapitres 4 à 8, s'intéresse à l'apprentissage par renforcement dans le cadre multiagent.

Le chapitre 2 présente l'apprentissage par renforcement. Il commence par en développer les fondements psychologiques et historiques, puis décrit la théorie et énumère les principaux algorithmes qui nous semblent les plus pertinents par rapport à nos problématiques.

Dans le troisième chapitre, différentes approches d'injection de connaissances dans le cas de problèmes de plus court chemin stochastique sont étudiées. La connaissance que l'on a *a priori* sur une tâche donnée peut être injectée dans les valeurs initiales ou dans la fonction de récompense afin d'accélérer l'apprentissage d'un agent. Une de nos contributions, présentée dans ce chapitre, est de proposer une fonction d'influence générique efficace et non risquée.

Le chapitre 4 a pour objectif de faire la liaison entre l'apprentissage par renforcement et les systèmes multiagents. Ces derniers sont tout d'abord introduits avec une mise en avant des différents concepts associés à ces systèmes multiagents. L'extension du cadre de l'apprentissage par renforcement aux systèmes multiagents conduit ensuite à préciser diverses notions nouvelles. Ces notions sont nécessaires à l'exposition des différents formalismes dans lesquels se situe l'apprentissage par renforcement multiagent. A l'issue de ce chapitre, une synthèse fait le lien entre les concepts propres aux systèmes multiagents et ces formalismes.

Dans le chapitre 5 est exposée une contribution qui est l'étude des enjeux que doivent surmonter des agents indépendants apprenant par renforcement dans les jeux de Markov d'équipe. Nous nous intéressons notamment à préciser les objectifs du groupe et ceux de chaque agent, et nous détaillons un certain nombre de difficultés rencontrées dans ce contexte, en particulier les facteurs de non-coordination. La fonction de ce chapitre est l'établissement des problématiques théoriques abordées dans la partie multiagent de cette thèse.

Le chapitre 6 est consacré à un état de l'art des travaux concernant l'apprentissage par renforcement d'agents indépendants dans le cadre des jeux de Markov d'équipe. Une notation uniformisée est utilisée, permettant de faire apparaître des points communs entre certains d'entre eux. Une synthèse est aussi proposée concernant les enjeux (présentés au chapitre précédent) surmontés par chacun de ces algorithmes.

Le chapitre 7 expose deux approches développées qui concernent l'apprentissage par renforcement d'agents indépendants dans le cadre des jeux de Markov d'équipe. La première approche, appelée Q-learning hystérétique, repose sur une extension du Q-learning décentralisé utilisant des agents « à tendance optimiste réglable ». La seconde contribution est un algorithme, appelé *Swing between Optimistic or Neutral* (SOoN), qui permet à des agents indépendants d'adapter automatiquement leurs évaluations à la stochasticité de l'environnement. Le fonctionnement de ces deux algorithmes et la démarche pour

y aboutir sont décrits en détail. Ils sont validés par des expérimentations présentées en fin de chapitre.

Le chapitre 8 présente l'application de ces travaux à la commande d'un système distribué de micromanipulation. Il détaille en particulier le prototype de surface active pneumatique et le modèle du convoyage d'objets en 2D que nous avons réalisé au cours de cette thèse. L'approche par apprentissage par renforcement décentralisé est comparée à une méthode classique de contrôle de systèmes distribués de micromanipulation. Une extension de nos méthodes à la commande décentralisée de la *smart surface* dans un cas partiellement observable est enfin proposée.

Le dernier chapitre conclue cette thèse, tirant un bilan des travaux qui y sont présentés et dévoilant les perspectives et travaux futurs qui se dessinent.

Chapitre 2

Apprentissage par renforcement

Ce chapitre présente les origines psychologiques et historiques de l'apprentissage par renforcement, ainsi que ses fondements théoriques. Il décrit le cadre fondamental de l'apprentissage par renforcement ainsi que les objectifs et principes utilisés pour la résolution. Les méthodes de résolution sont détaillées avec tout d'abord une brève présentation de la programmation dynamique puis les algorithmes d'apprentissage par renforcement basés sur la méthode des différences temporelles et en rapport avec nos travaux (Q-learning, Sarsa, Q(λ)).

2.1 Introduction

Développé depuis les années 1980, l'apprentissage par renforcement est une méthode de contrôle automatique qui ne nécessite pas de connaître le modèle du système mais simplement un critère de satisfaction, appelé renforcement, comme par exemple la satisfaction d'atteindre la consigne ou de réussir la tâche demandée. L'intérêt de cette approche est de pouvoir réaliser un contrôleur capable d'apprendre à commander un système inconnu sans avoir à spécifier comment la tâche doit être réalisée. Le contrôleur apprend par essais et erreurs, c'est-à-dire à partir d'expériences où il teste les commandes appliquées au système. L'objectif est de trouver la commande adéquate pour chaque situation. En d'autres termes, l'apprentissage par renforcement permet la synthèse de contrôleurs dans le cas où l'on ne dispose pas d'assez d'informations (par exemple pas de modèle du système) pour utiliser les approches classiques de l'automatique. Les applications de cette méthode sont nombreuses, notamment en robotique mobile ainsi qu'en microrobotique, compte tenu de la complexité du micromonde.

Ce chapitre se compose de trois parties. Nous nous sommes inspirés des ouvrages de référence de Leslie Pack Kaelbling, Michael L. Littman et Andrew W. Moore [KLM96] et de Richard S. Sutton et Andrew G. Barto [SB98] ainsi que des références [Lau02, Gar04] pour écrire ce chapitre. Nous commençons par replacer l'apprentissage par renforcement

dans son contexte originel, c'est-à-dire les recherches menées en psychologie comportementale sur l'apprentissage du comportement animal ainsi que la théorie de la commande optimale. Nous présentons ensuite les fondements théoriques avec le cadre général des processus décisionnels de Markov, qui permettent de formaliser les algorithmes d'apprentissage par renforcement. Ceux-ci sont exposés dans la troisième partie où sont principalement étudiées les méthodes basées sur les différences temporelles.

2.2 Origines psychologiques et historiques

Le concept d'apprentissage par renforcement est né à la fin des années 1980 de la rencontre entre deux domaines de recherche. L'un s'occupait des problèmes de commande optimale, l'autre de la psychologie animale. Puis, émanant des premiers travaux sur l'intelligence artificielle, des modèles informatiques basés sur l'apprentissage par essais et erreurs se sont développés et ont conduit à la formulation actuelle des modèles théoriques de l'apprentissage par renforcement.

2.2.1 Commande optimale

A la fin des années 1950 émerge la théorie du **contrôle optimal**, dont l'objectif est de concevoir des contrôleurs capables de minimiser une mesure de performance du contrôle d'un système dynamique. Une approche développée par Richard Bellman utilise le concept d'état du système dynamique et de fonction de valeur pour définir une équation fonctionnelle à résoudre, appelée **équation de Bellman**. La classe des méthodes pour résoudre les problèmes de contrôle optimal en résolvant cette équation est connue sous le nom de **programmation dynamique** [Bel57a]. Richard Bellman [Bel57b] introduit aussi la version stochastique discrète du problème de contrôle optimal appelée **Processus Décisionnel de Markov** (PDM) qui est le cadre habituel des algorithmes d'apprentissage par renforcement. Il propose une méthode de résolution par **itérations sur les valeurs** [Bel57a] et en 1960, Ronald Howard publie une méthode de résolution par **itérations de politiques** [How60]. Ces méthodes sont à la base des algorithmes d'apprentissage par renforcement.

2.2.2 Psychologie animale

L'autre source d'inspiration de l'apprentissage par renforcement fut les théories empiriques sur l'apprentissage animal, et notamment les modèles relevant de la tendance béhavioriste en psychologie. Le *béhaviorisme*[1] est une approche de la psychologie qui étudie l'interaction de l'individu avec son milieu. On distingue en psychologie animale plusieurs modèles d'apprentissage, notamment le conditionnement opérant de Buhrrus F. Skinner à qui nous devons la notion de **renforcement**. Ces modèles rendent compte de la manière dont une association entre des stimuli et des réflexes ou conséquences peut être renforcée.

1. Du terme anglais *behavior* qui signifie comportement

Conditionnement classique

Le conditionnement classique, appelé aussi **conditionnement pavlovien**, a été initié par le physiologiste russe Ivan Pavlov [Pav27] au début du XX^{ème} siècle. Au cours de ses travaux sur la salivation des chiens, il remarque qu'un chien salive lorsqu'il voit le plat où l'on met habituellement sa nourriture ou lorsqu'il sent l'odeur de la viande. Il établit alors un protocole expérimental destiné à montrer la création d'une association dans le cerveau de l'animal entre le réflexe salivaire et un stimulus neutre (par exemple le son d'une cloche) qui n'a rien à voir avec la nourriture. Pour mettre en évidence le phénomène de conditionnement pavlovien, il suffit d'associer temporellement un stimulus initialement neutre avec un second stimulus induisant un réflexe, appelé stimulus inconditionnel (par exemple l'apport de nourriture). Après un certain nombre d'expositions à la succession de ces deux stimuli, l'association se forme et il suffit de présenter uniquement le premier stimulus, devenu alors stimulus conditionnel, pour déclencher le réflexe conditionné. Ainsi, le simple son d'une cloche suffit à faire saliver le chien sans qu'aucune nourriture ne soit servie. Ivan Pavlov a ainsi mis en évidence un **processus d'apprentissage** dû à l'association entre un réflexe conditionné et un stimulus de l'environnement.

La loi de l'effet

Le premier à exprimer l'idée d'une sélection du comportement d'un animal en le récompensant ou non fut Edward L. Thorndike [Tho11]. Il formula une loi dite **loi de l'effet** établissant qu'une réponse est plus susceptible d'être reproduite si elle entraîne une satisfaction pour l'organisme et d'être abandonnée s'il en résulte une insatisfaction. Ainsi, les conséquences d'un comportement auront un effet sur la probabilité que ce comportement se reproduise ou non. Bien que parfois controversée, cette méthode a principalement inspiré l'**apprentissage par essais et erreurs** car elle renferme deux de ses principes : la sélectivité et l'associativité. La sélectivité permet la recherche d'alternatives nouvelles, tandis que l'associativité permet d'associer chaque solution trouvée à une situation particulière.

Conditionnement opérant

Le psychologue américain Buhrrus F. Skinner [Ski38] approfondit les travaux de Thorndike et élabora au milieu du XX^{ème} siècle le concept de **conditionnement opérant**. Celui-ci se distingue du conditionnement classique par le fait que la réponse ne soit pas un réflexe de l'organisme. Skinner a observé que la pression exercée par un animal sur un levier s'accroît de manière considérable lorsqu'elle entraîne la distribution de nourriture. Il s'établit ainsi un processus d'apprentissage de relations entre un comportement et ses conséquences. Les conséquences rendent plus ou moins probables la reproduction du comportement selon que ce soit des **récompenses** ou des **punitions**.

2.2.3 Modélisation informatique

Les premières spéculations sur l'intelligence artificielle datent des années 1950 avec Alan Turing. Il fut le premier à s'intéresser à la programmation d'une machine automatique pour reproduire le fonctionnement du cerveau. Dans l'article *L'ordinateur et l'intelligence* [Tur50], Turing explore le problème de l'intelligence artificielle et pose la question : « Les machines peuvent-elles penser ? ». Il propose une expérience, maintenant célèbre sous le nom de « test de Turing », qui consiste à dire que si, au cours d'un échange composé de questions et de réponses, on ne peut distinguer si l'interlocuteur est un ordinateur ou un humain, alors il est indéniable que l'ordinateur fasse preuve d'intelligence. Cet article est le document fondateur du vaste domaine de l'**intelligence artificielle**.

De ces premiers travaux sur l'intelligence artificielle émane l'idée de programmer un ordinateur pour apprendre une tâche par essais et erreurs. Dans les années 1960, de nombreux chercheurs travaillent alors sur l'apprentissage par essais et erreurs appliqué à des problèmes de contrôle. En 1961, Donald Michie [Mic61] décrit un système capable d'apprendre à jouer au morpion par essais et erreurs. Puis avec R. Chambers, ils écrivent en 1968 un programme capable d'apprendre à maintenir un pendule inversé en équilibre [MC68], exemple désormais classique de l'apprentissage par renforcement.

Parallèlement à ces recherches, Arthur Samuel [Sam59] réalise un logiciel qui apprend à jouer aux dames en utilisant une notion de différence temporelle. Cette notion introduit l'idée d'apprendre progressivement en utilisant la différence entre les estimations successives d'une même quantité. La synthèse entre les deux courants, apprentissage par essais et erreurs et différence temporelle, est réalisée par Harry Klopf [Klo72, Cd75]. Dans la lignée de ces travaux, Andrew G. Barto, Richard S. Sutton et Charles W. Anderson [BSA88] proposent une méthode connue sous le nom d'**AHC-learning**[2], qui est considérée comme la première méthode d'apprentissage par renforcement. Elle repose sur une architecture acteur-critique. Dans ces architectures, la structure de mémoire est séparée pour représenter la fonction qui sélectionne les actions (l'acteur) et celle qui les évalue (le critique).

Enfin, la formalisation mathématique des algorithmes d'apprentissage par renforcement telle que nous la connaissons aujourd'hui date de 1988 lorsque Richard S. Sutton [Sut88] et Christopher J.C.H. Watkins [Wat89] publient les algorithmes du **TD(λ)** et du **Q-Learning**. Ils font ainsi le lien entre la différence temporelle et l'apprentissage par essais et erreurs et utilisent le cadre théorique de la commande optimale par **itérations sur les valeurs**.

2.3 Fondements théoriques

2.3.1 Point de départ

L'apprentissage par renforcement s'inscrit dans le cadre de l'**apprentissage par essais et erreurs**. Cela peut se définir comme le problème d'apprendre, à partir d'ex-

2. Adaptative Heuristic Critic learning

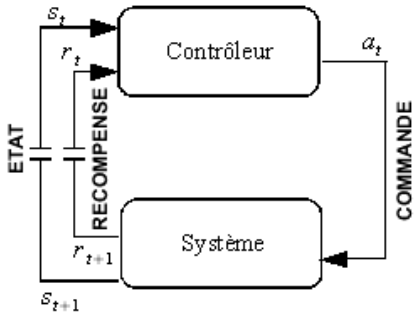


FIGURE 2.1 – Boucle sensori-motrice

périences, ce qu'il faut faire en chaque situation. L'objectif est de maximiser au cours du temps un signal scalaire émis par l'environnement.

Le modèle standard de l'apprentissage par renforcement est donné par une boucle sensori-motrice échantillonnée, présentée à la figure 2.1. Deux entités interagissent : le **contrôleur** [3] et le **système** [4]. Le contrôleur teste des **actions** [5] dans son environnement pour en évaluer les conséquences. A chaque instant t, le contrôleur perçoit l'état du système s_t et décide d'une action a_t à exécuter. Cette action modifie l'état du système. Le système fournit alors au contrôleur le nouvel état s_{t+1} ainsi qu'un signal de récompense r_{t+1}. Ce signal de **récompense** [6] peut être positif ou négatif. Il est utilisé par le contrôleur pour évaluer les actions effectuées dans un état. Le but du contrôleur est donc d'apprendre la meilleure séquence d'actions qui va le conduire dans un état final donné et lui permettre de récolter les meilleures évaluations. Par exemple, dans chaque état, le contrôleur peut choisir une action lui permettant de maximiser la somme des récompenses qu'il va recevoir sur le long terme. Autrement dit, le contrôleur apprend un *mapping* de ses états vers ses actions permettant de maximiser la somme des récompenses reçues.

2.3.2 Processus décisionnels de Markov

Comme nous l'avons vu dans l'historique, la plupart des algorithmes d'apprentissage par renforcement se situent dans le cadre des **processus décisionnels de Markov**.

Définition 2.1 *Un processus décisionnel de Markov (PDM) [Bel57a, SB98] fini est un quadruplet* $\langle \mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R} \rangle$ *où :*

- $\mathcal{S}$ *est un ensemble fini d'états,*
- $\mathcal{A}$ *est un ensemble fini d'actions,*

3. Ou *agent*.
4. Ou *environnement*.
5. Ou *commandes*.
6. Ou *renforcement*

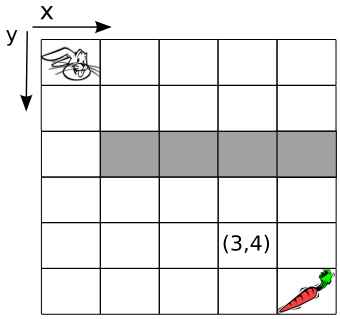


FIGURE 2.2 – L'agent (lapin) doit atteindre la carotte.

- $\mathsf{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0;1]$ *est une fonction de transition définissant une distribution de probabilité sur les états futurs telle que :*

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \sum_{s' \in \mathcal{S}} \mathsf{T}(s, a, s') = 1,$$

- $\mathsf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ *est une fonction de récompense qui associe une récompense immédiate*[7] *à chaque transition.*

On se restreint à des PDM stationnaires, c'est-à-dire que les fonctions T et R sont indépendantes du temps. Dans le cas contraire, on suppose généralement que la dérive de stationnarité est très lente (beaucoup plus lente que la vitesse d'apprentissage).

Un PDM permet de prendre des décisions dans un environnement lorsque l'on a une incertitude sur l'état dans lequel on se trouve ou en présence d'incertitude sur l'effet des actions. Les processus décisionnels de Markov modélisent donc des problèmes de décision en environnement **stochastique**. $\mathsf{T}(s, a, s')$ est la probabilité d'atteindre l'état s' lorsque l'on effectue l'action a depuis l'état s. La même information sera aussi notée sous forme de probabilité conditionnelle de l'évènement s' sachant que s et a sont vrais, *i.e.* $P(s'|s, a)$. Dans le cas particulier du cadre déterministe, pour tout couple (s, a), la probabilité d'être dans l'un des états suivants s' vaut 1 pour un état particulier et 0 pour tous les autres.

L'agent et l'environnement interagissent à chaque pas de temps. Dans chaque état, l'agent doit décider quelle action exécuter. Donc à chaque pas de temps t, l'agent perçoit l'état $s_t \in S$ dans lequel il se trouve, effectue une action $a_t \in A$ et au pas de temps suivant $t+1$, l'agent se trouve dans un nouvel état $s_{t+1} \in S$ selon une probabilité de transition $\mathsf{T}(s_t, a_t, s_{t+1})$ et reçoit une récompense immédiate $r_{t+1} = \mathsf{R}(s_t, a_t, s_{t+1}) \in \mathbb{R}$. La fonction de récompense utilise parfois uniquement l'état s_t et l'action effectuée a_t ($\mathsf{R}(s_t, a_t)$), ou tout simplement le fait d'être dans un état s_{t+1} ($\mathsf{R}(s_{t+1})$).

Exemple

Un exemple classique de PDM est le problème de recherche de plus court chemin dans un labyrinthe (*cf.* figure 2.2). L'agent (ici un lapin) peut se déplacer dans quatre directions

7. On parlera aussi de *renforcement* ou de *gain*.

(Nord, Sud, Est, Ouest) et doit atteindre un but (ici une carotte). Le lapin ne peut pas sortir de la grille ni franchir les obstacles (ici représentés par des cases grises). Ainsi, $\mathcal{S}$ est l'ensemble des positions possibles du lapin ($|\mathcal{S}| = 26$) et $\mathcal{A} = \{\text{Nord, Sud, Est, Ouest}\}$ l'ensemble de ses actions. L'objectif à atteindre est la carotte qui est dans l'état s_g, appelé **état final**. Dans ce cas, l'état final est un **état absorbant**, *i.e.* c'est un état s tel que $\forall a \in \mathcal{A}\ \mathsf{T}(s, a, s) = 1$. La fonction de récompense peut être par exemple :

$$\mathsf{R}(s, a, s') = \begin{cases} 1 & \text{si } s \neq s_g \text{ et } s' = s_g \\ 0 & \text{sinon} \end{cases} . \tag{2.2}$$

La fonction de transition peut être stochastique, c'est-à-dire que l'exécution d'une action a à partir d'un état s peut mener à différents états s'. Par exemple, dans l'état initial $s_{(0,0)}$, le lapin peut aller 9 fois sur 10 dans la direction souhaitée et sinon rester sur place, *i.e.* :

$$\begin{cases} \mathsf{T}(s_{(0,0)}, \text{Est}, s_{(1,0)}) = 0,9 \\ \mathsf{T}(s_{(0,0)}, \text{Est}, s_{(0,0)}) = 0,1 \end{cases} . \tag{2.3}$$

Propriété fondamentale de Markov

La fonction de transition définit une distribution de probabilité sur les états futurs qui vérifie la **propriété fondamentale de Markov**. Si on note h_t l'historique à la date t du processus, soit $h_t = (s_0, a_0, ..., s_{t-1}, a_{t-1}, s_t, a_t)$. La probabilité d'atteindre un nouvel état s_{t+1} suite à l'exécution d'une action a_t ne dépend que de l'état précédent s_t et de l'action effectuée a_t, et pas de l'historique h_t. On a :

$$\forall t, h_t, s_{t+1},\ P(s_{t+1}|h_t) = P(s_{t+1}|s_t, a_t) \tag{2.4}$$

Cette propriété dit simplement que la connaissance de l'**état courant** et de l'**action** est suffisante pour savoir comment le système va évoluer. Aucune information supplémentaire ne permet de mieux prédire le prochain état rencontré (étant donnée l'action choisie). Cette propriété peut paraître restrictive mais tout dépend de la manière dont sont définis les états du système. En effet, si ceux-ci contiennent toute l'information nécessaire à la prédiction de l'évolution du système, le PDM vérifie la propriété fondamentale de Markov.

Nous pouvons illustrer cette propriété sur le labyrinthe (*cf.* figure 2.2). L'état du système à chaque instant est la position de l'agent dans le labyrinthe. Il n'est pas nécessaire de connaître l'historique , c'est-à-dire les états et actions précédents, pour prédire l'état suivant de l'agent. La connaissance de l'état courant et de l'action choisie est suffisante pour prédire le comportement futur de l'agent. Le système vérifie alors la propriété fondamentale de Markov.

2.3.3 Objectif et politique

Critère à maximiser

Dans un PDM, après chaque décision, l'agent reçoit un signal de l'environnement destiné à le récompenser ou le pénaliser. Son comportement va dépendre de l'interprétation

de ce signal. Nous avons vu plus haut que l'agent va chercher dans chaque état à maximiser au cours du temps la somme des récompenses qu'il va recevoir sur le long terme. Dans le cas d'un horizon infini, on utilise généralement le critère γ-pondéré. Dans ce cas, le critère à maximiser est la somme pondérée des récompenses futures, dans laquelle les récompenses reçues loin dans le temps sont moins prises en compte que les récompenses proches. La pondération par un coefficient d'atténuation $\gamma \in [0;1[$ permet de calculer une récompense pondérée sur le long terme. Le critère γ-pondéré G à maximiser est alors :

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.5}$$

où r_t est la récompense obtenue à l'instant t. Le coefficient d'atténuation γ détermine l'influence du futur lointain par rapport au futur proche. En effet, plus γ se rapproche de 0 et plus les récompenses obtenues dans un futur lointain sont négligées. Si $\gamma = 0$, l'agent ne cherche à maximiser que la récompense immédiate.
D'autres critères d'optimalité existent mais le critère γ-pondéré est celui qui a reçu le plus d'attention. C'est pourquoi nous l'utiliserons dans ce mémoire.

Politique

L'objectif de l'agent est donc d'adopter un comportement lui permettant de maximiser un critère. Le comportement de l'agent est défini comme « choisir l'action à effectuer dans chaque état du système ». Pour cela, on définit une distribution de probabilité notée π, appelée **politique**, qui associe à chaque état s et chaque action a la probabilité $\pi(s,a)$ de choisir l'action a dans l'état s. π est donc une fonction de l'ensemble des couples d'état-action dans l'intervalle $[0;1]$: $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0;1]$ avec la contrainte suivante :

$$\forall s \in \mathcal{S},\ \sum_{a \in \mathcal{A}} \pi(s,a) = 1. \tag{2.6}$$

Une telle politique est dite de Markov car elle ne dépend que de l'état courant. C'est aussi une politique **stochastique**. Par exemple, pour le PDM représenté à la figure 2.2, une politique stochastique pourrait être de choisir dans l'état $(3,4)$ l'action *Est* et l'action *Sud* chacune avec une probabilité de 0,5, soit $\pi(s_{(3,4)}, \text{Est}) = \pi(s_{(3,4)}, \text{Sud}) = 0,5$.

Il est aussi possible de définir une politique déterministe $\pi : \mathcal{S} \mapsto \mathcal{A}$ qui associe à chaque état une action à effectuer. Une politique déterministe peut être considérée comme une politique stochastique où pour tout état s, $\pi(s,a)$ vaut 1 pour une action a particulière et 0 pour toutes les autres.

2.3.4 Fonctions de valeur

Afin que l'agent puisse comparer ses différentes politiques, une valeur est associée à chacune d'elle. Ces valeurs vont permettre d'établir un classement entre les différentes politiques d'un agent et de déterminer la meilleure d'entre elles. L'objectif d'un agent est de maximiser un critère de performance défini sur le long terme. La valeur d'une

politique π sera alors l'espérance du critère lorsque l'on part de l'état s et que l'on suit ensuite la politique π. Dans le cas du critère γ-pondéré, la fonction de valeur V^π d'un état s se définit alors comme suit :

$$\begin{aligned}\mathsf{V}^\pi(s) &= E_\pi \left\{ G_{t+1} | s_t = s \right\} \\ &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\}\end{aligned} \tag{2.7}$$

où E désigne l'espérance mathématique.

Une autre fonction de valeur représente la valeur d'un couple état-action (s, a) sous une politique π. C'est alors l'espérance du critère lorsque l'on exécute l'action a depuis l'état s et que l'on suit ensuite la politique π :

$$\mathsf{Q}^\pi(s,a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \tag{2.8}$$

Cette fonction Q, aussi appelée **fonction de valeur**, est essentielle pour les algorithmes d'apprentissage présentés à la section 2.5.

La fonction de valeur Q^π vérifie pour une politique π donnée, pour tout s et pour tout a :

$$\begin{aligned}\mathsf{Q}^\pi(s,a) &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \\ &= E_\pi \left\{ r_{t+1} + \sum_{k=1}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \\ &= E_\pi \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a \right\} \\ &= \sum_{s' \in \mathcal{S}} \mathsf{T}(s,a,s') \left[\mathsf{R}(s,a,s') + \gamma E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s' \right\} \right] \\ &= \sum_{s' \in \mathcal{S}} \mathsf{T}(s,a,s') \left[\mathsf{R}(s,a,s') + \gamma \mathsf{V}^\pi(s') \right]\end{aligned} \tag{2.9}$$

L'espérance de récompense en (s, a) mesurée par $\mathsf{Q}^\pi(s, a)$ est la somme de la récompense $\mathsf{R}(s, a, s')$ reçue immédiatement après avoir effectué a en s et de la valeur de la politique π à partir de l'état suivant s'.

$\mathsf{V}^\pi(s)$ est la moyenne sur les différentes actions possibles des $\mathsf{Q}^\pi(s, a)$:

$$\forall s \in \mathcal{S},\ \mathsf{V}^\pi(s) = \sum_{a \in \mathcal{A}} \pi(s,a) \mathsf{Q}^\pi(s,a). \tag{2.10}$$

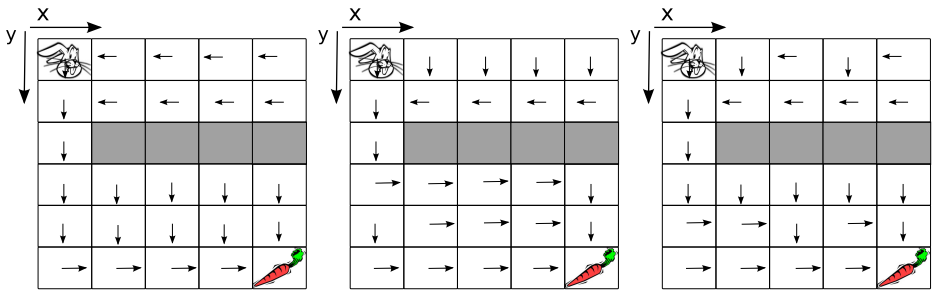


FIGURE 2.3 – Politiques optimales déterministes possibles pour atteindre l'état but.

2.3.5 Optimalité

Nous avons vu que l'objectif consiste à trouver la meilleure politique permettant de maximiser la récompense cumulée, appelée **politique optimale**. Par exemple, la figure 2.3 propose les politiques optimales déterministes possibles pour trouver le plus court chemin vers l'état final. Les fonctions de valeur permettent de définir un ordre partiel entre les politiques.

Définition 2.2 *Soit π_1 et π_2 deux politiques définies sur un PDM. On définit la relation $\leq$ telle que $\pi_1 \leq \pi_2$ si et seulement si $\forall s \in \mathcal{S}\ \forall a \in \mathcal{A},\ \mathsf{Q}^{\pi_1}(s,a) \leq \mathsf{Q}^{\pi_2}(s,a)$.*

On peut montrer qu'il existe au moins une politique optimale notée π^* dont la fonction de valeur dans n'importe quel état est au moins aussi bonne que celle de tout autre politique. La politique optimale π^* vérifie donc :

$$\forall \pi \quad \pi \leq \pi^*. \tag{2.11}$$

L'objectif de l'agent est donc de trouver une politique optimale. Il peut y avoir plusieurs politiques optimales dont les fonctions de valeur sont identiques. On note Q^* la fonction de valeur optimale d'une politique optimale définie par :

$$\forall s \in \mathcal{S}\ \forall a \in \mathcal{A} \quad \mathsf{Q}^*(s,a) = \max_{\pi} \mathsf{Q}^{\pi}(s,a). \tag{2.12}$$

La fonction de valeur optimale Q^* vérifie aussi une autre équation. En effet, pour un PDM fini et $\gamma \in [0;1[$, Q^* est l'unique solution de l'**équation d'optimalité de Bellman** suivante :

$$\mathsf{Q}^*(s,a) = \sum_{s' \in \mathcal{S}} \mathsf{T}(s,a,s') \left[\mathsf{R}(s,a,s') + \gamma \max_{b \in \mathcal{A}} \mathsf{Q}^*(s',b) \right]. \tag{2.13}$$

Cette équation traduit l'idée que quel que soit l'état initial s, la première action d'une politique optimale mène dans un nouvel état s' à partir duquel l'évolution ultérieure est optimale.

La relation entre la fonction de valeur optimale V^* et la fonction de valeur optimale Q^* est :

$$\forall s \in \mathcal{S},\ \mathsf{V}^*(s) = \max_{b \in \mathcal{A}} \mathsf{Q}^*(s,b). \tag{2.14}$$

Quand on connaît Q^*, on peut construire une politique optimale π^* à partir des actions gloutonnes sur Q^*.

Définition 2.3 *Une action* ***gloutonne*** *sur* Q *dans un état* s *est une action* a *dont la Q-valeur est la plus élevée,* i.e. *telle que* $a \in \arg\max_{b \in \mathcal{A}} \mathsf{Q}(s, b)$.

Définition 2.4 *Une politique* π ***gloutonne*** *sur* Q *est définie par :*

$$\forall s \in \mathcal{S} \forall a \in \mathcal{A}, \quad \pi(s,a) = \begin{cases} 1 & si\ a \in \arg\max_{b \in \mathcal{A}} \mathsf{Q}(s,b) \\ 0 & sinon \end{cases} . \tag{2.15}$$

Ainsi, quand on connaît Q^*, on peut construire une politique optimale π^* comme la politique gloutonne sur Q^*. Donc pour trouver une politique optimale π^*, une solution peut être de déterminer la fonction de valeur optimale Q^*. C'est l'objectif des méthodes de résolution présentées dans la section suivante.

2.4 Programmation dynamique

Dans la section précédente, on a explicité formellement ce qu'était un PDM, quels étaient son objectif et son critère de performance. Nous nous intéressons maintenant aux algorithmes qui permettent à un agent dans un PDM d'apprendre une politique optimale. A chaque interaction, l'agent connaît l'état et l'action du système, mais il n'a pas forcément accès aux transitions et aux récompenses. Dans le cas de la programmation dynamique, on suppose connues les probabilités de transition T et la fonction de récompense R. Nos travaux ne se placent pas dans ce cadre mais les algorithmes utilisés reprennent certaines idées de la programmation dynamique. C'est pourquoi nous présentons en premier lieu les méthodes issues de la programmation dynamique.

2.4.1 Principe

On considère que l'agent dispose de toutes les informations nécessaires sur le problème qu'il doit résoudre, *i.e.* que le modèle de l'environnement est connu. La résolution d'un PDM est alors un problème P-complet [PT87]. Deux étapes élémentaires se distinguent dans les algorithmes de programmation dynamique : l'**évaluation** d'une politique et l'**amélioration** d'une politique. L'évaluation consiste à déterminer la valeur de chacun des états pour un agent suivant une politique donnée. Par exemple, l'algorithme d'évaluation itérative d'une politique (*policy evaluation*) calcule itérativement la valeur d'une politique donnée. Il faut ensuite améliorer la politique jusqu'à déterminer une politique optimale. Deux algorithmes permettent de construire itérativement une politique déterministe optimale. L'algorithme d'itérations des politiques (*policy iteration*) [How60] évalue une certaine politique et, en fonction de cette évaluation, calcule une nouvelle politique qui est meilleure que l'ancienne et ainsi de suite. Le second algorithme

Algorithme 1 : Algorithme d'itération des valeurs (*value iteration*)

début
  Initialiser $\mathsf{Q}(s,a)$ arbitrairement pour tout (s,a) de $\mathcal{S}\times\mathcal{A}$
  répéter
    $\delta \leftarrow 0$
    pour tous les $(s,a)\in\mathcal{S}\times\mathcal{A}$ **faire**
      $q \leftarrow \mathsf{Q}(s,a)$
      $\mathsf{Q}(s,a) \leftarrow \sum_{s'\in\mathcal{S}} \mathsf{T}(s,a,s')\left[\mathsf{R}(s,a,s') + \gamma \max_{b\in\mathcal{A}} \mathsf{Q}(s',b)\right]$
      $\delta \leftarrow \max(\delta, |q - \mathsf{Q}(s,a)|)$
  jusqu'à $\delta < \beta$
fin

est l'algorithme d'itération des valeurs (*value iteration*) [Bel57a]. Cet algorithme aide à comprendre les algorithmes d'apprentissage présentés au paragraphe 2.5 ; nous allons donc le présenter plus en détail.

2.4.2 Algorithme d'itération des valeurs

Si on connaît le modèle du système, l'algorithme d'itération des valeurs permet d'évaluer itérativement la fonction de valeur optimale V^* ou la fonction de valeur optimale Q^*. L'algorithme 1 décrit le cas de l'évaluation successive de la fonction de valeur Q. Cet algorithme effectue simultanément les deux étapes d'évaluation et d'amélioration. Il est fondé sur le **théorème du point fixe de Banach** qui assure la convergence d'une suite vers l'unique fonction solution de l'équation de Bellman. L'appellation *value iteration* se justifie car une suite des approximations successives est générée et elle converge vers la fonction de valeur Q^* de la politique optimale. La méthode consiste à résoudre les équations d'optimalité de Bellman (2.13) de manière incrémentale. L'équation de mise à jour est la suivante :

$$\mathsf{Q}_{t+1}(s,a) \leftarrow \sum_{s'\in\mathcal{S}} \mathsf{T}(s,a,s')\left[\mathsf{R}(s,a,s') + \gamma \max_{b\in\mathcal{A}} \mathsf{Q}_t(s',b)\right] \tag{2.16}$$

Un critère d'arrêt classique utilise un seuil positif β. Les itérations sont stoppées lorsque le plus grand écart entre deux approximations successives est inférieur à ce seuil. Cet algorithme converge vers la fonction de valeur optimale Q^*.

2.5 Méthodes des différences temporelles

Dans cette section, nous allons présenter différents algorithmes d'apprentissage où l'agent apprend directement une politique optimale en expérimentant un système dont il ne connaît pas le modèle. Contrairement à la programmation dynamique vue précédemment, les algorithmes d'apprentissage par renforcement basés sur les méthodes temporelles sont donc exécutés en ligne au sein de la boucle sensori-motrice. Nous pré-

sentons dans cette section des méthodes qui utilisent dans leur fonction de mise à jour le principe de **différence temporelle** proposé par Richard S. Sutton [Sut88].

2.5.1 Q-learning

Cet algorithme est sûrement l'un des plus utilisés dans la communauté de l'apprentissage par renforcement en raison de sa simplicité, de sa robustesse et des preuves formelles de sa convergence dans les PDM. Le Q-learning, introduit par Christopher J.C.H. Watkins [Wat89], est une adaptation de *value iteration* (algorithme 1) qui remplace la mise à jour utilisant T et R par une mise à jour utilisant le principe de la différence temporelle. Comme nous l'avons vu précédemment, l'objectif est d'optimiser la fonction de valeur pour chaque couple état-action afin d'en déduire une politique optimale. L'algorithme Q-learning génère donc une suite de fonctions Q qui convergent vers Q^*.

Mise à jour

A chaque période d'échantillonnage t, l'agent effectue une action a depuis un état s ; il se retrouve alors dans un nouvel état s' et reçoit une récompense r. L'agent dispose alors des informations courantes s, a, s' et r pour mettre à jour la fonction de valeur pour le couple état-action (s, a). La nouvelle estimation de la valeur d'un couple état-action (s, a) est notée $\mathsf{Q}_{t+1}(s, a)$. Elle se partage entre l'estimation précédente et un facteur correctif selon :

$$\mathsf{Q}_{t+1}(s, a) = \mathsf{Q}_t(s, a) + \delta. \tag{2.17}$$

Ce facteur correctif noté δ est appelé **différence temporelle** et correspond à la différence entre l'amélioration et l'ancienne estimation de la fonction de valeur $\mathsf{Q}(s, a)$. L'ancienne estimation est notée $\mathsf{Q}_t(s, a)$. Pour l'amélioration, on reprend l'équation de mise à jour de *value itération* (2.16). La correction s'exprime alors par :

$$\begin{aligned} \delta &= \text{amélioration} - \text{ancienne estimation} \\ &= \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma \max_{b \in A} Q_t(s', b) \right] - Q_t(s, a). \end{aligned} \tag{2.18}$$

Si l'environnement est déterministe, la correction est alors :

$$\delta = r + \gamma \max_{b \in A} Q_t(s', b) - Q_t(s, a) \tag{2.19}$$

Afin d'approximer la « vraie » correction et de s'affranchir de l'utilisation des fonctions T et R, on introduit dans la correction exprimée en environnement déterministe un facteur α pour prendre en compte le cas non-déterministe. α signifie que s' n'est pas l'unique état successeur possible pour le couple (s, a). Au final, le Q-learning met à jour la fonction de valeur en utilisant les informations courantes à chaque période d'échantillonnage s,

Algorithme 2 : Algorithme du Q-learning

début
  Initialiser $\mathsf{Q}(s,a)$ arbitrairement pour tout (s,a) de $\mathcal{S}\times\mathcal{A}$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    $a \leftarrow$ Décision(Q, s)
    Exécuter l'action a et observer le nouvel état s' et la récompense r
    $\mathsf{Q}(s,a) \leftarrow (1-\alpha)\mathsf{Q}(s,a) + \alpha\left[r + \gamma \max_{b\in\mathcal{A}} \mathsf{Q}(s',b)\right]$
    $s \leftarrow s'$
fin

a, s' et r selon :

$$\begin{aligned}\mathsf{Q}_{t+1}(s,a) &\leftarrow \mathsf{Q}_t(s,a) + \alpha\delta \\ &\leftarrow \mathsf{Q}_t(s,a) + \alpha\left[r + \gamma \max_{b\in\mathcal{A}} \mathsf{Q}_t(s',b) - \mathsf{Q}_t(s,a)\right] \\ &\leftarrow (1-\alpha)\mathsf{Q}_t(s,a) + \alpha\left[r + \gamma \max_{b\in\mathcal{A}} \mathsf{Q}_t(s',b)\right] \end{aligned} \tag{2.20}$$

$\alpha \in]0;1]$ est appelé **taux d'apprentissage**.

De là, on en déduit l'algorithme du Q-learning présenté à l'algorithme 2. Cet algorithme ne stocke que la fonction de valeur Q. Couramment, un tableau de dimension $|\mathcal{S}| \times |\mathcal{A}|$ est utilisé. Les valeurs de cette table sont les évaluations courantes de chaque couple état-action.

Convergence

On peut prouver que sous certaines conditions, l'algorithme du Q-learning converge vers la solution optimale Q^*. Ces conditions sont [JJS94] :

- que $\mathcal{S}$ et $\mathcal{A}$ soient finis,
- que tous les couples état-action du PDM soient visités un nombre infini de fois,
- que (α_n) satisfasse les hypothèses suivantes :

$$\sum_n \alpha_n(s,a) = \infty \quad \text{et} \quad \sum_n \alpha_n^2(s,a) < \infty, \tag{2.21}$$

- que $0 \leq \gamma < 1$.

Ces hypothèses sont assez fortes. Ainsi, en théorie, α doit décroître à chaque nouveau passage par le couple état-action (s,a). En pratique, pour éviter de stocker des informations supplémentaires ou pour que l'apprentissage soit permanent (par exemple pour s'adapter à un changement de dynamique du système), on utilise un coefficient fixe. La valeur couramment utilisée est $\alpha = 0,1$. Ainsi, même si certaines hypothèses ne sont pas complètement vérifiées, on observe que l'algorithme se comporte de manière très robuste et converge vers des stratégies quasi-optimales.

Une autre hypothèse de convergence exige que tous les couples état-action soient visités un nombre infini de fois. En pratique, une exploration partielle est généralement suffisante. Il faut alors faire la balance entre l'exploration et l'exploitation.

Méthode de décision : dilemme exploration-exploitation

La fonction *Décision* permet de choisir une action dans chaque état. Cette fonction de décision de l'action doit donc trouver un compromis entre l'exploitation, qui consiste à choisir les actions gloutonnes (*cf.* définition 2.3), et l'exploration, qui consiste à choisir une action au hasard. Tant que l'agent n'a pas exploré la totalité de son environnement, il n'est pas certain que sa politique gloutonne (*cf.* définition 2.4) actuelle soit optimale. L'exploration permet donc d'approfondir ses connaissances pour améliorer la récompense cumulée, mais au risque d'une perte de temps si la politique optimale a été trouvée et si l'agent explore trop. Il faut ainsi faire la balance entre exploitation et exploration.

Le compromis couramment utilisé consiste à choisir l'action gloutonne la plupart du temps tout en explorant plus ou moins souvent. On classe les méthodes de décision en deux catégories : les méthodes dirigées et non-dirigées. Les méthodes non-dirigées utilisent uniquement les valeurs courantes de la table Q pour décider de l'action. Deux méthodes non-dirigées sont couramment employées :

- **méthode ϵ-greedy** [8] **:** Cette méthode de décision n'utilise qu'un paramètre ϵ et choisi une action au hasard avec une probabilité ϵ et l'action gloutonne avec une probabilité $1-\epsilon$. S'il existe plusieurs actions gloutonnes dans un état, le choix sera aléatoire parmi celles-ci. Souvent, le paramètre ϵ est choisit égal à $0,1$ afin de satisfaire le compromis exploration-exploitation.
- **méthode softmax :** Cette méthode assigne une probabilité à chaque action possible en fonction de sa Q-valeur. La probabilité de sélection d'une action a dans un état s est calculée selon une distribution de Boltzmann :

$$\pi(s,a) = \frac{e^{\frac{\mathsf{Q}(s,a)}{\tau}}}{\sum_{b\in\mathcal{A}} e^{\frac{\mathsf{Q}(s,b)}{\tau}}} \tag{2.22}$$

  où τ est un paramètre de température. Plus τ est grand et plus le choix des actions sera équiprobable et donc le taux d'exploration important. Si τ est proche de zéro, la probabilité associée à l'action de plus forte valeur sera la plus importante ; on s'approche donc d'une exploitation.

Dans ces méthodes, les paramètres ϵ et τ permettent de déterminer le niveau d'exploration. Le choix de ces paramètres est appelé **stratégie d'exploration**. Par exemple, une stratégie d'exploration stationnaire consiste à attribuer une valeur fixe au paramètre ϵ ou τ. Une troisième méthode non-dirigée consiste à initialiser à de fortes valeurs les Q-valeurs pour favoriser l'exploration. Cette méthode sera expliquée plus en détail dans le chapitre 3.

8. Ou ϵ-glouton ou exploration pseudo-stochastique.

Les méthodes dirigées utilisent des informations supplémentaires pour guider l'exploration. Par exemple, des bonus d'exploration peuvent être calculés [RP03]. Ce sont des heuristiques basées sur des informations acquises au cours de l'apprentissage. Ils vont encourager le contrôleur à favoriser certaines actions [Thr92, MB99]. Par exemple, le nombre de fois où une action a été effectuée dans un état peut être mémorisé pour favoriser les actions les moins fréquemment utilisées. Ou encore, le contrôleur peut préférer choisir les actions pour lesquelles la valeur de la fonction Q change le plus en calculant la différence absolue entre deux approximations successives des Q-valeurs. Les méthodes d'exploration dirigée sont d'autres exemples basés sur les actions les moins récentes ou sur celles dont la Q-valeur a la plus grande variance.

2.5.2 Sarsa

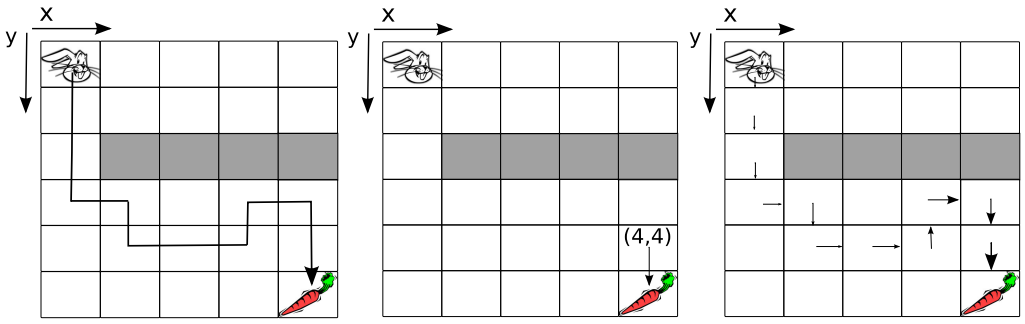
Nous allons maintenant présenter un algorithme d'apprentissage par renforcement qui va nous permettre d'aborder les notions d'algorithmes *on-policy*[9] et *off-policy*[10]. L'algorithme du Q-learning est dit *off-policy* car la valeur du prochain état s' est estimée sans prendre en considération l'action exécutée. En effet, dans l'algorithme 2, on estime la valeur de l'état s' dans la fonction de mise à jour grâce au maximum que l'on peut obtenir dans cet état. L'algorithme Sarsa est dit *on-policy* car il utilise les actions choisies par la méthode de décision pour estimer la valeur du prochain état s'. Dans l'état s, l'action $a \leftarrow \text{Décision}(\mathsf{Q}, s)$ est effectuée et l'agent reçoit la récompense r et observe le nouvel état s'. L'action $a' \leftarrow \text{Décision}(\mathsf{Q}, s')$ va être choisie depuis l'état s' en suivant la méthode de décision. La formule de mise à jour du couple (s, a) par l'algorithme Sarsa est [RN94] :

$$\mathsf{Q}_{t+1}(s,a) \leftarrow (1-\alpha)\mathsf{Q}_t(s,a) + \alpha\left[r + \gamma \mathsf{Q}_t(s',a')\right]. \tag{2.23}$$

Ainsi, la méthode de décision a une influence directe sur la politique apprise dans l'algorithme Sarsa. La conséquence est que la politique obtenue tient compte de la stratégie d'exploration suivie et est donc plus sure. Par exemple, si certaines actions d'exploration génèrent des pénalités, la politique obtenue va les prendre en compte et va compenser ces situations dangereuses en choisissant un chemin sous-optimal mais plus sûr (ne pas prendre un chemin près d'un précipice si on risque d'y tomber dans l'exemple du *cliff-walking* donné par Richard S. Sutton et Andrew G. Barto [SB98]). Le Q-learning ne prenant pas en considération l'exploitation, la politique obtenue est optimale mais risquée (prendre le chemin le plus court même s'il est près du précipice).

Des résultats sur la convergence des algorithmes *on-policy* selon différentes stratégies d'exploration sont donnés dans [SJLS00]. Les auteurs introduisent notamment la stratégie d'exploration appelée *greedy in the limit with infinite exploration* (GLIE) ou « gloutonne à la limite ». Cette stratégie consiste à utiliser une exploration intensive au début de l'apprentissage et à décroître ensuite l'exploration de sorte à utiliser exclusivement les actions gloutonnes à l'infini.

9. Ou *on-line*
10. Ou *off-line*



(a) chemin parcouru (b) couple état-action mis à jour avec le Q-learning (c) couples état-action mis à jour avec la trace d'éligibilité

FIGURE 2.4 – Intérêt des traces d'éligibilité.

2.5.3 TD(λ)

Les algorithmes que nous avons présentés jusqu'à maintenant (Q-learning et Sarsa) présentent le défaut de ne mettre à jour que le dernier état visité. Par exemple, dans le cas du labyrinthe (*cf.* figure 2.4a), l'agent reçoit une récompense pour avoir effectué l'action *bas* dans l'état $(4, 4)$ et alors, seule la Q-valeur de ce couple état-action est mise à jour (*cf.* figure 2.4b). Il faudra que l'agent parcourt à nouveau plusieurs fois ce chemin pour que cette récompense se propage jusqu'à l'état initial. En attendant cette propagation, les autres valeurs sont inchangées en début d'apprentissage et le comportement de l'agent est aléatoire. Afin de faire "remonter" cette information plus rapidement le long du chemin parcouru, une méthode consiste à propager cette récompense en parcourant la mémoire des transitions en marche arrière depuis la récompense reçue. Introduit par Richard S. Sutton [Sut88], le *TD(λ)* recouvre une classe d'algorithmes basés sur cette méthode.

Trace d'éligibilité

Les algorithmes TD(λ) utilisent la notion de *trace d'éligibilité*. L'idée est de modifier les couples état-action visités selon leur ancienneté par rapport à la dernière transition effectuée (*cf.* figure 2.4c). Beaucoup plus de Q-valeurs sont alors mises à jour à chaque transition. La fonction d'éligibilité est mémorisée dans une table $\mathsf{e} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ qui mémorise pour chaque couple état-action l'ancienneté de la dernière visite sous forme d'un nombre réel. Un nouveau paramètre $\lambda \in [0; 1]$ appelé *taux de diffusion* permet de doser la remontée d'information. A chaque passage par un couple état-action (s, a), la trace est remise à jour soit par $\mathsf{e}(s, a) \leftarrow 1$ dans le cas d'*éligibilité remplaçante*, soit par $\mathsf{e}(s, a) \leftarrow \mathsf{e}(s, a) + 1$ dans le cas d'*éligibilité accumulatrice*. De plus, à chaque période d'échantillonnage, les valeurs d'éligibilité pour tous les couples diminuent de façon exponentielle d'un facteur $\gamma\lambda$. Ainsi, plus l'éligibilité est faible, plus le passage est ancien. D'après Richard S. Sutton et Satinder P. Singh [SS96], l'éligibilité remplaçante peut parfois améliorer la vitesse de convergence.

Algorithme 3 : Algorithme du $\mathsf{Q}(\lambda)$ de C. Watkins

début
  Initialiser $\mathsf{Q}(s,a)$ arbitrairement et $\mathsf{e}(s,a)$ à 0 pour tout (s,a) de $\mathcal{S}\times\mathcal{A}$
  Initialiser l'état initial s et choisir au hasard une action a
  tant que *s n'est pas un état absorbant* **faire**
    Exécuter l'action a et observer le nouvel état s' et la récompense r
    $a' \leftarrow$ Décision(Q, s')
    $a^* \in \arg\max_{b\in\mathcal{A}} \mathsf{Q}(s',b)$
    $\delta \leftarrow r + \gamma\mathsf{Q}(s',a^*) - \mathsf{Q}(s,a)$
    $\mathsf{e}(s,a) \leftarrow \mathsf{e}(s,a) + 1$ ou $\mathsf{e}(s,a) \leftarrow 1$ ▷ `éligibilité accumulatrice ou remplaçante`
    pour tous les $(u,b) \in \mathcal{S}\times\mathcal{A}$ **faire**
      $\mathsf{Q}(u,b) \leftarrow \mathsf{Q}(u,b) + \alpha\delta\mathsf{e}(u,b)$
      $\mathsf{e}(u,b) \leftarrow \begin{cases} \gamma\lambda\mathsf{e}(u,b) & \text{si } \mathsf{Q}(s',a') = \mathsf{Q}(s',a^*) \\ 0 & \text{sinon} \end{cases}$
    $s \leftarrow s', a \leftarrow a'$
fin

$\mathsf{Q}(\lambda)$: un algorithme basé sur les traces d'éligibilité

Christopher J.C.H. Watkins [Wat89] a présenté une version baptisé $\mathsf{Q}(\lambda)$ donnée à l'algorithme 3. Le $\mathsf{Q}(\lambda)$ utilise les traces d'éligibilité pour mettre à jour les Q valeurs à chaque itération. Notons que si $\lambda = 0$, l'algorithme $\mathsf{Q}(\lambda)$ est équivalent au Q-learning. L'erreur de différence temporelle δ corrige l'estimation de chaque couple état-action en fonction de la valeur de $\mathsf{e}(s,a)$. De plus, les traces d'éligibilité sont remises à zéro si une action d'exploration est choisie.

Le $\mathsf{Q}(\lambda)$ converge sous les mêmes hypothèses que le Q-learning [Day92, DS94] et trouve très rapidement une politique relativement bonne. Néanmoins, dans certains cas, une fois qu'il a trouvé une bonne politique, il est plus long que le Q-learning à trouver la politique optimale [Lau02]. De plus, il est nécessaire de mémoriser en plus de la table Q une seconde table de dimension $|\mathcal{S}| \times |\mathcal{A}|$ pour les valeurs d'éligibilité.

Une version de l'algorithme Sarsa utilisant les traces d'éligibilité est connue sous le nom de **Sarsa(λ)** [RN94].

2.5.4 Autres algorithmes

Les algorithmes présentés précédemment (Q-learning, Sarsa et $\mathsf{Q}(\lambda)$) appartiennent à la classe des méthodes *directes*[11]. Ces méthodes recherchent directement la fonction de valeur optimale en expérimentant dans le système. Une autre classe de méthodes est celle des méthodes *indirectes*[12] qui construisent et mémorisent un modèle de l'environnement au fur et à mesure des états visités. Par exemple, le Dyna-Q construit un modèle partiel et déterministe de l'environnement qui mémorise pour chaque couple état-action visité l'état suivant et la récompense observés. Grâce à ce modèle, la fonction de valeur est optimisée

11. Ou méthodes sans modèle.
12. Ou méthodes avec modèle.

à chaque période d'échantillonnage ou même indépendamment lorsque la boucle sensori-motrice est stoppée par exemple. Les algorithmes indirects les plus connus sont le *Dyna-Q* et le *Priority Sweeping*.

2.6 Conclusion

Dans ce chapitre ont été présentés l'apprentissage par renforcement et ses principaux algorithmes. Ces derniers présentent l'avantage de trouver une politique proche de l'optimale. Toutefois, deux inconvénients majeurs limitent leur utilisation. Le premier est l'emploi de tableaux pour stocker les valeurs de Q qui n'est pas adapté à des espaces d'états de grandes tailles. Le second est la faible vitesse de convergence des algorithmes. Mais des solutions consistent à utiliser des fonctions d'approximations, des méthodes indirectes ou à injecter de la connaissance *via* des heuristiques dans l'apprentissage. Ce dernier point est mis en lumière dans le chapitre suivant.

Chapitre 3

Injection de connaissances dans le cas de problèmes de plus court chemin stochastique

Ce chapitre aborde les approches d'injection de connaissance dans l'apprentissage par renforcement. Après une présentation générale des différentes méthodes pour injecter la connaissance, il se focalise sur le choix de la fonction de récompense et des valeurs initiales de la fonction de valeur. Les méthodes existantes dans le cadre de l'apprentissage par renforcement sont exposées et de nouvelles idées sont proposées. Notamment, une fonction d'influence générique est introduite. Elle initialise la fonction de valeur et accélère l'apprentissage pour les problèmes de plus court chemin stochastique.

3.1 Introduction

Une des principales limitations des algorithmes d'apprentissage par renforcement est leur **lenteur de convergence**. En effet, les méthodes d'apprentissage par essais et erreurs nécessitent un grand nombre d'itérations afin que l'agent puisse explorer tous les états du système. Par exemple, dans le cas du Q-learning, la convergence de l'algorithme est garantie si l'agent visite une infinité de fois tous les couples état-action. L'exploration du système est donc un paramètre important mais qui demande du temps, ce qui limite les méthodes classiques d'apprentissage telles que le Q-learning à des petits espaces d'état-action. Leur utilisation pour des applications réelles est alors difficile car le temps d'expérimentation peut être extrêmement long. Améliorer la vitesse de convergence des algorithmes d'apprentissage par renforcement est donc une problématique importante. De nombreuses méthodes fondées sur des stratégies d'injection de connaissance *a priori* ou de conseils ont été proposées. Parmi elles, le choix de certains paramètres de l'apprentissage joue un rôle majeur sur la performance des algorithmes. Ce chapitre est consacré

à l'influence du choix de certains paramètres de l'apprentissage sur sa vitesse dans le cas de problèmes de plus court chemin stochastique. Dans ce cadre, l'objectif principal est de *proposer une méthode pour correctement choisir la fonction de renforcement et les valeurs initiales de l'apprentissage par renforcement en vue d'obtenir le comportement optimal désiré en un minimum de temps.*

Sont tout d'abord présentées dans ce chapitre les différentes méthodes d'injection de connaissance classées par type d'apprentissage : apprentissage par imitation, progressif ou intuitif. Les choix de l'initialisation des Q valeurs et de la fonction de récompense sont ensuite abordés successivement. Pour chacun de ces paramètres, les méthodes existantes sont exposées et de nouvelles fonctions sont aussi proposées.

3.2 Injection de connaissances

Dans la vie courante, il est rare que l'on aborde une nouvelle tâche sans aucune idée concernant la façon dont elle peut être accomplie. On a souvent une intuition, des instructions ou une démonstration de la part d'une personne extérieure pour nous aider dans notre apprentissage. Dans le cas de l'apprentissage par renforcement, on peut aborder la problématique de la même manière. Il peut être possible d'accéder à une certaine forme de connaissance sur le système, que ce soit grâce à une expertise humaine, à des contrôleurs développés auparavant ou à des intuitions. On peut alors utiliser ces informations pour accélérer l'apprentissage. On parle alors d'injection de connaissances.

Les méthodes classiques d'apprentissage humain sont classées en différentes catégories [Con99] dont s'inspirent les stratégies d'injection en apprentissage par renforcement : l'apprentissage par imitation qui fait intervenir la démonstration d'un professeur, l'apprentissage progressif qui structure l'apprentissage en différentes étapes et enfin l'apprentissage intuitif qui s'appuie principalement sur des heuristiques que l'élève doit vérifier ou améliorer.

3.2.1 Apprentissage par imitation

Un agent peut apprendre en « regardant » et en imitant le comportement d'un agent expérimenté. En terme d'apprentissage par renforcement, cette méthode consiste en l'intervention d'un expert qui fournit des informations au contrôleur [BK96]. S'il sait piloter seul le système, l'expert peut diriger au départ l'agent apprenant. C'est la phase de **démonstration**. Celle-ci peut prendre diverses formes : l'expert peut diriger directement le contrôleur par l'intermédiaire d'un joystick par exemple [SK02, BAC06] ou être simplement observé [HD94, KII94]. Stefan Schaal [Sch97] propose ainsi d'accélérer la vitesse d'apprentissage dans le cas du pendule inversé grâce à cette méthode. Pour les cinq premiers essais, c'est un expert qui dirige le contrôleur et qui réalise avec succès la mise en équilibre du pendule en position droite à partir de différents états initiaux. Puis l'agent apprenant doit reproduire cette mise en équilibre. C'est la seconde phase dite d'**imitation**.

L'expert peut aussi jouer simplement le rôle d'un conseiller si son expertise sur le système est partielle. On parle alors de **critique grossier**. Les informations fournies par le critique grossier ne sont pas forcément optimales et l'agent apprenant doit affiner les connaissances fournies par le critique grossier ; l'agent apprenant est donc appelé **critique fin**. Cette architecture est une architecture multi-expert [DMK00, MRIB04]. Par exemple, le critique grossier peut avoir appris une tâche simplifiée comme « mettre un but seul » dans le cas de robots footballeurs et devenir un conseiller pour un robot devant jouer face à un attaquant. Se pose alors pour l'agent apprenant le choix du critique à suivre : ne pas prendre en compte le conseil reçu et agir selon son propre discernement, écouter le conseil et accepter qu'il influence le choix sans pour autant le suivre à la lettre, ou enfin considérer l'avis du critique grossier comme un ordre. Pour résoudre ce dilemme, une fonction de confiance envers le conseil reçu peut être utilisée [MDP03].

3.2.2 Apprentissage progressif

En psychologie comportementale, une méthode d'apprentissage progressif nommée *shaping* consiste à présenter tout d'abord un problème simple à l'élève, puis à lui proposer des exercices de difficultés croissantes. On s'inspire intuitivement de cette méthode lorsque l'on apprend à faire du vélo et que l'on débute avec l'utilisation de petites roues. En apprentissage par renforcement, façonner l'environnement peut permettre d'augmenter la complexité des tâches à apprendre. Par exemple, on peut éloigner petit à petit un robot mobile de son objectif ou simplifier au départ un labyrinthe (en enlevant par exemple les obstacles) [DC94, PH96]. L'utilisation de l'apprentissage sur simulation avant l'apprentissage sur système réel est aussi une forme d'apprentissage progressif [CDB96]. Par exemple, Andrew Y. Ng *et al.* [NCD+04] réalisent le vol « inversé », c'est-à-dire à l'envers, d'un hélicoptère autonome *via* l'apprentissage par renforcement progressif. Grâce aux informations collectées pendant le vol normal de l'hélicoptère sous contrôle manuel, un modèle dynamique de celui-ci est obtenu. Une première phase d'apprentissage par renforcement du vol inversé est ensuite réalisée sous simulation grâce à ce modèle. Le contrôleur va ensuite affiner son apprentissage lors des essais en vols réels. Andres El-Fakdi et Marc Carreras effectuent de même l'apprentissage d'une politique initiale par simulation avant l'affinage de cette politique sur le système réel, qui est un robot sous-marin [EFC08]. Une autre application est proposée par Guillaume Laurent [Lau02] dans sa thèse dont l'objectif est de concevoir un contrôleur par apprentissage par renforcement d'un système de manipulation. Dans les tests expérimentaux réalisés, il a recours à une simulation logicielle afin de pouvoir entraîner les algorithmes avant de les utiliser sur le système réel. L'objectif de la simulation n'est pas de reproduire fidèlement le système, mais d'avoir un outil pour valider un algorithme ; la simulation est utilisée comme un **premier exercice d'approximation**. Il constate que l'apprentissage sur le système réel après un apprentissage sur simulation est plus rapide que l'apprentissage direct (sans simulation au préalable). Cela suppose cependant que l'on dispose d'une simulation, même simplifiée, du système.

3.2.3 Apprentissage intuitif ou *biaisé*

Les travaux en apprentissage par renforcement injectant de l'intuition dans le système sont nombreux. L'intuition peut porter sur des pré-requis sur l'environnement. Par exemple, dans le cas d'un labyrinthe, une manière intuitive de guider un agent est de lui conseiller de diminuer la distance entre son état courant et l'état but. Bien entendu, cela requiert d'avoir accès à cette information ! De plus, l'agent peut être près du but spatialement mais très loin du but du point de vue du nombre de pas (par exemple, si un mur les sépare). Cela illustre l'utilisation d'heuristique avec les méthodes intuitives. Il subsiste toujours un risque de mener l'agent dans une impasse. L'intuition peut aussi être plus vague et simplement favoriser ou non certains comportements comme l'exploration. Par exemple, les méthodes d'exploration dirigée, présentées dans la section précédente, utilisent des heuristiques pour guider l'exploration de l'agent. Le choix de fortes valeurs initiales de la table Q est aussi une technique élémentaire pour encourager l'exploration [SB98].

Discussion

De nombreuses techniques d'injection de connaissance consistent à incorporer l'heuristique directement dans la fonction de récompense ou la fonction de valeur. En effet, le choix et l'initialisation de ces paramètres ont un impact très important sur la performance des algorithmes d'apprentissage [KS96]. Ainsi, S. Behnke et M. Bennewitz [BB05] associent méthode par imitation et initialisation de la fonction de valeur en proposant de donner accès à l'agent apprenant aux valeurs de la table Q d'un agent expérimenté. G. Hailu et G. Sommer [HS99] codent la connaissance dans des matrices de croyance [1] qui initialisent les tables Q en début d'apprentissage.

Cependant, malgré leur rôle important dans l'apprentissage, on choisit souvent en pratique la fonction de récompense de manière intuitive et les valeurs initiales de la table Q de manière arbitraire [SB98]. Nous proposons donc une analyse générique des effets des paramètres de l'apprentissage par renforcement sur la stratégie optimale dans le cas de problèmes de plus court chemin stochastique [2]. Nous validons notre étude avec l'algorithme du Q-learning et l'exemple du labyrinthe. L'objectif principal est de *proposer une méthode pour correctement choisir la fonction de récompense et les valeurs initiales de l'apprentissage par renforcement en vue d'obtenir le comportement optimal désiré en un minimum de temps, dans le cadre de problèmes de plus court chemin stochastique.*

3.3 Initialisation des Q valeurs

Nous étudions tout d'abord le cas d'une **fonction de récompense binaire**, qui est la plus largement utilisée dans la littérature et permettra d'extrapoler nos résultats. Nous supposons que deux tendances se démarquent lors de l'apprentissage : **une stratégie globale** et **un comportement spécifique en début d'apprentissage**. Dans cette

1. En anglais *Belief matrices.*
2. En anglais *goal-directed.*

partie, nous allons préciser ces deux tendances qui dépendent du choix de la fonction de récompense et de l'initialisation de la table Q.

3.3.1 Récompenses binaires

Une fonction de **récompense binaire** est telle que la récompense reçue est toujours r_∞ excepté si le nouvel état est l'état cible à atteindre. Dans ce cas, la récompense est r_g. On a ainsi :

$$\forall s \in \mathcal{S}\ \forall a \in \mathcal{A} \quad \mathsf{R}(s,a,s') = \begin{cases} r_g \text{ si } s' = s_g \\ r_\infty \text{ sinon} \end{cases} \tag{3.1}$$

où s' est l'état obtenu en effectuant l'action a depuis l'état s, et s_g l'état cible. D'après Sven Koenig et Reid G. Simmons [KS96], deux fonctions de récompense binaire sont principalement utilisées en apprentissage par renforcement. Une fonction *goal-reward* dans laquelle l'agent reçoit une récompense positive ($r_g = 1$) lorsqu'il atteint l'état but et sinon, n'est ni pénalisé, ni récompensé ($r_\infty = 0$). L'autre fonction, *action-penalty*, a une structure plus dense car l'agent est pénalisé pour toutes les actions exécutées ($r_\infty = -1$) sauf celle qui mène au but ($r_g = 0$). Ces deux fonctions sont des cas particuliers de choix d'une récompense binaire que nous utiliserons dans certaines de nos expérimentations. Toutefois, l'étude qui suit est basée sur le choix plus général d'une fonction de récompense binaire quelconque.

3.3.2 Stratégie globale

Nous nous intéressons en premier lieu aux différentes stratégies globales qui se démarquent lors de l'apprentissage. Pour cela, nous définissons tout d'abord la distance au but en nombre de transitions.

Distance au but

Nous définissons la **distance minimax** $\mathsf{d}(s,s')$ comme le nombre minimal de transitions nécessaires à un agent pour atteindre l'état s' depuis l'état s dans le pire des cas. Dans un environnement déterministe, cette distance minimax est :

$$\forall s,s' \in \mathcal{S} \quad \mathsf{d}(s,s') = \begin{cases} 0 & \text{si } s = s' \\ 1 + \min\limits_{\substack{a \in \mathcal{A} \\ \mathsf{T}(s,a,s'')=1}} \mathsf{d}(s'',s') & \text{sinon} \end{cases} . \tag{3.2}$$

La **distance à l'état but** depuis l'état s est alors définie comme $\mathsf{d}_g(s) = \mathsf{d}(s,s_g)$. Pour éviter le cas où cette distance est infinie, les environnements **à exploration sûre** sont définis.

Définition 3.1 *Un environnement est* ***à exploration sûre***[3] *si la distance minimax au but* $\mathsf{d}_g(s)$ *pour tout état* $s \in \mathcal{S}$ *est finie, c'est-à-dire* $\mathsf{d}_g < \infty$ *[KS96].*

3. En anglais *safely explorable*.

Stratégie optimale

Dans le cas où toutes les récompenses sont identiques ($r_g = r_\infty$), la solution de l'équation de Bellman (2.13) est une constante notée Q_∞ :

$$\forall s \ \forall a \quad \mathsf{Q}^*(s,a) = Q_\infty = \frac{r_\infty}{1-\gamma}. \tag{3.3}$$

En d'autres termes, pendant le processus d'apprentissage, les valeurs de la table Q pour tous les couples d'état-action convergent vers Q_∞. Si $r_g \neq r_\infty$, on peut exprimer Q^* comme une fonction de la distance à l'état but d_g. En effet, dans le cas de problèmes de plus court chemin stochastique, la trajectoire optimale consiste à atteindre l'état absorbant s_g depuis l'état s en $\mathsf{d}_g(s)$ transitions. Dans un environnement déterministe, on obtient alors :

$$\begin{aligned}
\forall s \ \forall a \quad \mathsf{Q}^*(s,a) &= \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s, a_t = a \right) \\
&= \sum_{k=0}^{\mathsf{d}_g(s)-1} \gamma^k r_\infty + \gamma^{\mathsf{d}_g(s)} r_g \\
&= \frac{1-\gamma^{\mathsf{d}_g(s)}}{1-\gamma} r_\infty + \gamma^{\mathsf{d}_g(s)} r_g \\
&= (1-\gamma^{\mathsf{d}_g(s)}) Q_\infty + \gamma^{\mathsf{d}_g(s)} r_g.
\end{aligned} \tag{3.4}$$

Ainsi, la limite de la fonction de valeur Q^* pour les états dont la distance au but d_g tend vers l'infini est Q_∞. Donc, selon les valeurs de r_g et de Q_∞, les états sont **de plus en plus** ou **de moins en moins attirants** lorsque l'on se rapproche de l'état cible.

Soit s' un état plus proche de l'état cible que s. D'où, avec $\gamma \in [0;1[$ le coefficient d'atténuation :

$$\begin{aligned}
\mathsf{d}_g(s) &> \mathsf{d}_g(s') \\
\gamma^{\mathsf{d}_g(s)} &< \gamma^{\mathsf{d}_g(s')}
\end{aligned} \tag{3.5}$$

Soit $r_g > Q_\infty$, on a alors pour tout a :

$$\begin{aligned}
(\gamma^{\mathsf{d}_g(s)} - \gamma^{\mathsf{d}_g(s')}) Q_\infty &> (\gamma^{\mathsf{d}_g(s)} - \gamma^{\mathsf{d}_g(s')}) r_g \\
(1-\gamma^{\mathsf{d}_g(s')}) Q_\infty + \gamma^{\mathsf{d}_g(s')} r_g &> (1-\gamma^{\mathsf{d}_g(s)}) Q_\infty + \gamma^{\mathsf{d}_g(s)} r_g \\
\mathsf{Q}^*(s',a) &> \mathsf{Q}^*(s,a)
\end{aligned} \tag{3.6}$$

Donc, si $r_g > Q_\infty$, les valeurs des couples d'état-action menant à l'état cible sont de plus en plus attirantes lorsque l'on se rapproche de l'état cible. La stratégie optimale globale est **le plus court chemin vers** s_g. D'un autre côté, si $r_g < Q_\infty$, les valeurs des couples d'état-action menant à l'état cible sont de moins en moins attirantes lorsque l'on se rapproche de l'état cible. La stratégie optimale est donc **une répulsion locale**

TABLE 3.1 – Stratégie optimale en fonction des valeurs relatives de r_g et Q_∞.

$r_g > Q_\infty$	Plus court chemin vers s_g
$r_g < Q_\infty$	Répulsion locale de s_g

de s_g.

Évidemment, le plus court chemin vers l'état cible est la stratégie optimale recherchée dans le cas de problèmes de plus court chemin stochastique (*cf.* table 3.1). Donc r_g **doit toujours être supérieur à** Q_∞.

3.3.3 Initialisation uniforme des Q valeurs

De même que la fonction de récompense, les valeurs initiales Q_i de la fonction de valeur influencent la stratégie, mais seulement au début du processus d'apprentissage. Une tendance générale se distingue pendant les premiers épisodes de l'apprentissage.

Étudions les valeurs de la table Q en début d'apprentissage. Si nous calculons la première mise à jour d'un couple état-action (s, a) avec l'équation (2.20), tel que l'état suivant s' ne soit pas l'état cible et n'ait jamais été mis à jour, on obtient :

$$\begin{aligned} \mathsf{Q}(s,a) &\leftarrow Q_i + \alpha\left[r_\infty + (\gamma - 1)Q_i\right] \\ &\leftarrow Q_i + \alpha(1-\gamma)(Q_\infty - Q_i)\ . \end{aligned} \tag{3.7}$$

Donc la valeur discriminante de Q_i est aussi Q_∞. En fonction des valeurs de Q_i par rapport à Q_∞, **les états déjà visités seront plus ou moins attractifs** tant que l'agent n'a pas atteint un grand nombre de fois l'état cible.

- **si** $Q_i > Q_\infty$ **:** les états déjà visités auront une valeur inférieure à celle des états non visités ($\mathsf{Q}(s,a) < Q_i$). En d'autres termes, les états non visités seront donc plus attirants, ce qui induit l'agent à explorer. Cette **exploration est plus systématique** en début d'apprentissage que dans le cas de l'exploration aléatoire. Nous nommons ce comportement l'**exploration systématique**.
- **si** $Q_i < Q_\infty$ **:** les états déjà visités auront une valeur supérieure à celle des états non visités ($\mathsf{Q}(s,a) > Q_i$). C'est-à-dire que les états déjà visités seront plus attirants. Ceci induit une moins grande exploration de l'agent en début d'apprentissage. Ce comportement, que nous nommons **piétinement**, ralentit considérablement l'apprentissage. Il est donc préférable de l'éviter.
- **si** $Q_i = Q_\infty$ **:** les états déjà visités auront une valeur identique à celle des états non visités ($\mathsf{Q}(s,a) = Q_i$). Le comportement sera **purement aléatoire** en début d'apprentissage.

Ainsi, le choix d'une initialisation uniforme des Q valeurs doit être fait de manière appropriée selon le comportement désiré en début d'apprentissage, résumé en table 3.2.

TABLE 3.2 – Comportement en début d'apprentissage en fonction des valeurs relatives de Q_i et Q_∞.

$Q_i > Q_\infty$	Exploration systématique
$Q_i < Q_\infty$	Piétinement
$Q_i = Q_\infty$	Purement aléatoire

Discussion

Ces résultats sont en accord avec l'étude en début d'apprentissage sur les « représentations » de Sven Koenig et Reid G. Simmons [KS96]. Une représentation détermine le choix de la fonction de récompense et de l'initialisation des Q valeurs. Le résultat principal de leur étude est que le choix d'une représentation peut avoir un impact sur la performance d'un algorithme d'apprentissage par renforcement en début d'apprentissage. Ils étudient la complexité du Q-learning, mesurée en nombre de transitions, selon différentes représentations. Une représentation avec $Q_i = 0$ et une fonction de récompense *goal-reward* entraîne un nombre moyen de transitions pour atteindre le but la première fois qui peut croître exponentiellement avec le nombre d'états. En effet, une telle représentation correspond à un comportement purement aléatoire. Sven Koenig et Reid G. Simmons [KS96] préconisent donc deux autres représentations : $Q_i = 0$ et une fonction de récompense *action-penalty* ou $Q_i = 1$ et une fonction de récompense *goal-reward* qui correspondent toutes deux à un comportement d'exploration systématique. La complexité en nombre de transitions pour atteindre le but la première fois est alors diminuée. Il serait intéressant de généraliser ces travaux sur le calcul de la complexité aux récompenses binaires.

Expérimentations

Afin d'illustrer comment le comportement de l'agent est influencé par l'initialisation des Q valeurs, nous choisissons, pour des raisons de simplicité et de clarté, un labyrinthe non-déterministe. Le système est représenté par un lapin évoluant dans un damier (*cf.* figure 3.1). A chaque position du lapin correspond un état discret. Quand le lapin atteint l'état cible (la carotte), l'épisode se termine et le lapin est replacé sur sa case initiale. Le lapin a le choix entre quatre actions, selon ses intentions de se déplacer dans l'une des quatre directions cardinales (N,E,S,O). Si une action entraîne le lapin dans un mur, celui-ci reste dans son état courant. Ce système est non-déterministe : le lapin atteint l'état désigné avec une probabilité de 0,6, et sinon, il se retrouve de façon équiprobable dans l'un des quatre états voisins de l'état désigné. Pour tous les épisodes, l'algorithme du Q-learning est paramétré avec un coefficient d'apprentissage α de 0,1, un coefficient d'actualisation γ de 0,9, une table de Q initialisée uniformément à la valeur Q_i. L'agent suit une méthode ϵ-greedy où l'action gloutonne est choisie avec une probabilité de 0,9 ($\epsilon = 0{,}1$).

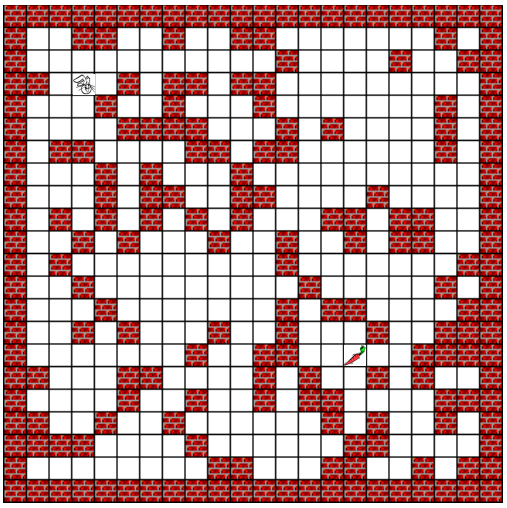
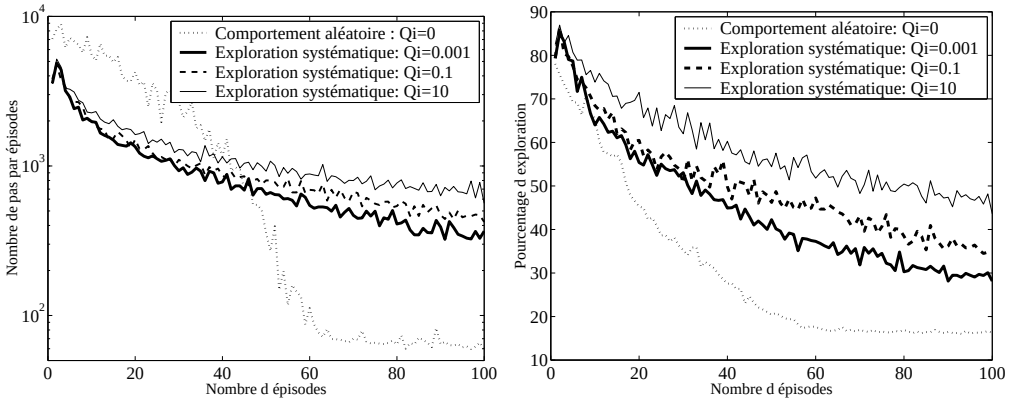


FIGURE 3.1 – Labyrinthe non-déterministe de 20×20 cases avec un état initial (état $[2, 2]$) et un état cible (la carotte) (état $[14, 14]$).

Nous choisissons une fonction de récompense binaire (3.1) telle que la stratégie optimale soit le plus court chemin vers l'état cible, *i.e.* $r_g = 1$ et $r_\infty = 0$. La valeur discriminante de Q_i est 0 (Q_∞=0).

Nous testons différentes valeurs de Q_i afin d'induire un comportement aléatoire ou différents comportements d'exploration systématique en début d'apprentissage. Concernant le comportement de piétinement, nous ne soumettons pas d'expérimentations car l'agent tourne en rond pendant un grand nombre de pas. Les résultats des autres comportements sont donnés en figure 3.2. La courbe du comportement aléatoire converge vers une limite qui est le nombre de pas minimum si l'agent suit le plus court chemin vers l'état cible. Avec un comportement d'exploration systématique, le nombre de pas par épisodes est inférieur en début d'apprentissage au nombre de pas nécessaire avec un comportement aléatoire. En effet, l'agent visite un plus grand nombre d'états et l'état cible est découvert plus rapidement. Le comportement d'exploration systématique favorise donc l'accélération de l'apprentissage pendant les premiers épisodes. De plus, nous avons utilisé différentes valeurs de Q_i pour expérimenter divers comportements d'exploration systématique. Nous remarquons que plus Q_i est supérieur à Q_∞, plus l'agent explore. Donc *plus la différence entre Q_i et Q_∞ est importante, plus le comportement caractéristique de début d'apprentissage s'accentue.*

Cependant, comme l'agent est incité à explorer, il s'éloigne du plus court chemin vers l'état cible pour effectuer des actions d'exploration. C'est pour cela que les courbes d'exploration systématique convergent plus lentement que la courbe du comportement aléatoire. Plus Q_i est supérieur à Q_∞, plus la convergence vers le nombre de pas minimum est lente. Donc plus le comportement d'exploration systématique est accentué, plus ce comportement de début d'apprentissage persiste. Afin de régler ce problème de



(a) Nombre de pas par épisodes pour atteindre l'état cible

(b) Pourcentage d'exploration de l'espace d'états induit par différentes valeurs de Q_i

FIGURE 3.2 – Expériences sur le labyrinthe moyennées sur 50 essais indépendants avec différentes valeurs de Q_i.

convergence, une méthode consiste à régler la pérennité de l'influence des valeurs initiales. Pour cela, on peut utiliser la limite de précision du calculateur. En effet, si l'on initialise les Q-valeurs juste « au-dessus » de Q_∞ (par exemple, $Q_i = Q_\infty + \epsilon$ où ϵ est très proche de 0), il suffira de quelques mises à jour de la valeur d'un couple état-action pour que cette valeur soit alors égale à Q_∞ et que le biais disparaisse. Le comportement d'exploration sera donc éphémère. Les résultats obtenus avec des valeurs initiales de Q_i proche de Q_∞ sont tracés en figure 3.3. Le comportement d'exploration systématique accélère l'apprentissage pendant les premiers épisodes et permet à l'agent d'explorer un grand nombre d'états. Ce comportement disparaît rapidement et l'agent suit alors sa politique gourmande et trouve rapidement le chemin optimal car un grand nombre de valeurs ont déjà été mises à jour.

3.3.4 Initialisation avec un biais : fonction d'influence

Étant donnée l'influence de l'initialisation de Q, nous proposons maintenant une initialisation non homogène de la fonction de valeur par une **fonction d'influence**. G. Hailu et G. Sommer [HS99], dans leur étude sur l'influence de la qualité et de la quantité d'un biais dans l'apprentissage, proposent une méthode nommée *goal directed bias* ou biais dirigé vers le but. Elle consiste à biaiser l'agent, si sa destination est connue, avec une politique initiale dirigée vers le but. Afin d'initialiser la table Q avec un biais dirigé vers le but, il faut instaurer un gradient ajustable sur les valeurs des états et dirigé vers le but. Ce gradient doit établir que les états près de l'état cible sont *a priori* de plus en plus attirants que les états loin de l'état cible. Nous suggérons la **fonction d'influence gaussienne** suivante :

$$Q_i(s,a) = \beta e^{-\frac{\mathrm{h}(s)^2}{2\sigma^2}} + \delta + Q_\infty \tag{3.8}$$

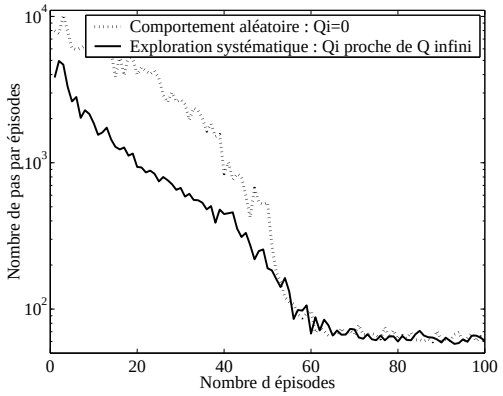


FIGURE 3.3 – Réglage de la persistance du comportement de début d'apprentissage. Expériences sur le labyrinthe moyennées sur 50 essais indépendants.

où δ fixe le niveau d'exploration systématique loin de l'état cible, β l'amplitude de la fonction d'influence, σ la zone d'influence et h est une heuristique définie sur les états pour guider l'agent. Le comportement non désiré de piétinement est évité car $Q_i \geq Q_\infty$. Afin d'éviter trop d'exploration systématique, δ et β doivent être choisis très petits par rapport à 1.

Expérimentations

Par exemple, prenons le labyrinthe présenté précédemment et une fonction de récompense binaire telle $r_\infty = 0$ et $r_g = 1$. L'heuristique h peut alors être la distance Manhattan entre le nouvel état $s = [x, y]$ et l'état final $s_g = [x_g, y_g]$ définie par :

$$\mathsf{d}_M(s, s_g) = |x - x_g| + |y - y_g| \,. \tag{3.9}$$

La figure 3.4 illustre le gradient dirigé vers le but sur les valeurs initiales des états du labyrinthe imposé par cette heuristique. La figure 3.5 expose les résultats obtenus avec une fonction d'influence sur le labyrinthe. Nous choisissons $\delta = 0$ de sorte à avoir en début d'apprentissage un comportement aléatoire loin de la cible. Une exploration systématique est induite dans une zone autour de l'état cible. La fonction d'influence conduit à un processus d'apprentissage plus rapide. Dès le dixième épisode, le nombre de pas nécessaire pour atteindre l'état cible est divisé par 6 par rapport à un comportement aléatoire. Le paramètre β a été initialisé à 0,001 donc la persistance de l'exploration systématique est assez élevée. C'est pourquoi le nombre de pas par épisodes avec la fonction d'influence est légèrement supérieur après 50 épisodes au nombre de pas avec le comportement aléatoire. Pour diminuer la persistance de ce phénomène, il suffit d'initialiser β plus proche de 0, comme expliqué précédemment.

Il est intéressant de remarquer que l'heuristique peut être fausse. Par exemple, dans le cas du labyrinthe, le biais dirigé vers le but peut être trompeur s'il y a présence d'obstacles

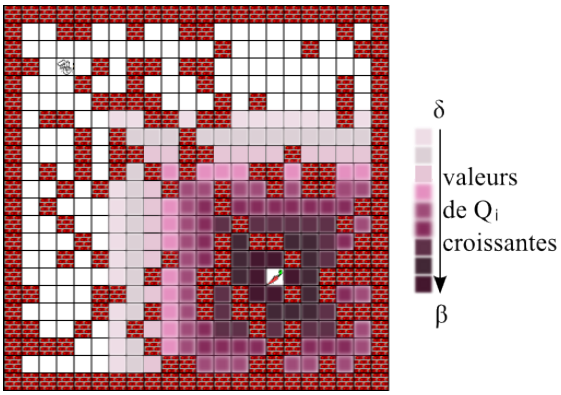


FIGURE 3.4 – Fonction d'influence gaussienne dirigée vers l'état cible. On utilise un code de couleur pour illustrer les différentes valeurs initiales de Q_i choisies pour chaque état.

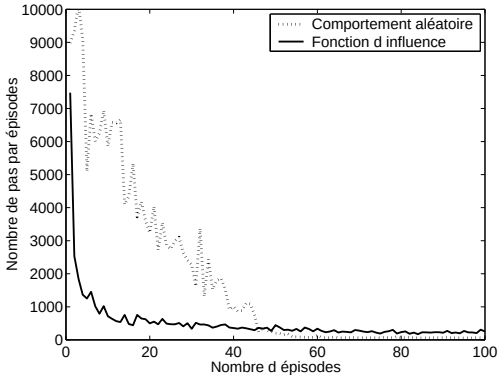


FIGURE 3.5 – Récompenses binaires avec $r_\infty = 0$ et $r_g = 1$. Le comportement aléatoire correspond à $Q_i = 0$. La fonction d'influence est $Q_i(s,a) = 0.001e^{-\frac{\mathrm{d}_M(s,s_g)^2}{2\times 13^2}}$. Expériences sur le labyrinthe moyennées sur 20 essais indépendants.

entre l'agent et l'état absorbant. L'heuristique utilisant la distance Manhattan ne les prend pas en compte. Si l'agent suit le gradient, il risque alors de rester bloqué dans une impasse. Néanmoins, aucun problème de cycles dans lesquels l'agent pourrait se bloquer n'apparaît, même si la fonction d'influence est fausse. En réalité, l'effet de la fonction d'influence est éphémère. Elle conseille l'agent **seulement en début d'apprentissage**.

3.3.5 Conclusion

Tout ceci met en évidence **l'importance des valeurs initiales de la table Q**. Le choix de Q_i n'est pas trivial et doit être fait en accord avec le comportement désiré. Dans les cas où le système s'éloigne naturellement de l'état cible, à cause par exemple d'instabilité dans cet état, l'exploration systématique peut être préférée afin d'accélérer

l'apprentissage au début. En effet, l'exploration systématique force le système à explorer des états nouveaux, et ainsi à se rapprocher de l'état cible. Par contre, il faut éviter une trop forte persistance de ce biais. Si certaines informations sur l'objectif sont connues, une fonction d'influence peut être utilisée pour instaurer un gradient dans les valeurs initiales de la table Q.

3.4 Choix d'une fonction de récompense

Afin d'élargir le cadre de notre étude, nous proposons maintenant d'étudier le cas des fonctions de récompense continue. Les méthodes qui consistent à modéliser la fonction de récompense selon des connaissances sont souvent appelées *reward shaping*.

3.4.1 Estimateurs de progrès

M. J. Mataric [Mat94] propose une méthode pour choisir les fonctions de récompense en utilisant les connaissances implicites sur l'environnement afin d'accélérer l'apprentissage par renforcement. Cette méthode statue que les agents doivent être autant que possible récompensés de manière locale, de sorte à avoir des informations lors de chaque action et non pas uniquement lorsque le but est atteint. Ainsi, M. J. Mataric [Mat94] constate que l'apprentissage de robots mobiles peut être accéléré de manière significative si un gradient sur les récompenses dirigé vers l'objectif est utilisé. Les robots sont récompensés lorsqu'ils choisissent des actions suivant le gradient, plutôt que seulement pour avoir atteint leur objectif. Ce gradient de récompense est appelé un **estimateur de progrès**.

Définition 3.2 *Un estimateur de progrès, noté φ, est associé à des objectifs précis et fournit une mesure de progrès relativement à ces objectifs. Il ne procure pas une information complète mais seulement un* ***conseil*** *partiel [Mat94].*

Les fonctions de potentiels, utilisées par [NHR99], peuvent être considérées comme des estimateurs de progrès. Une fonction de potentiel est définie sur les états et à valeurs réelles $\varphi : \mathcal{S} \mapsto \mathbb{R}$. Les fonctions de potentiel sont utilisées par exemple dans les applications de la théorie des schémas à la robotique. Cette méthode repose sur la génération d'un champ de potentiel englobant tout l'espace de navigation du robot [Ark89]. Les obstacles génèrent des champs répulsifs alors que le but génère un champ attractif. Les champs modélisés suivent généralement les lois de la gravitation ou de l'électromagnétisme : l'intensité du champ est inversement proportionnelle au carré de la distance.

Par exemple, dans le cas des problèmes de plus court chemin stochastique, l'agent peut être récompensé de manière positive si une action diminue sa distance au but et de manière négative si une action augmente sa distance au but. Ou alors, l'estimateur de progrès peut être la distance Manhattan (3.9) entre le nouvel état s' et s_g, c'est-à-dire $\varphi(s') = \mathsf{d}_M(s', s_g)$. Dans le cas du labyrinthe, l'objectif de l'agent est alors de minimiser

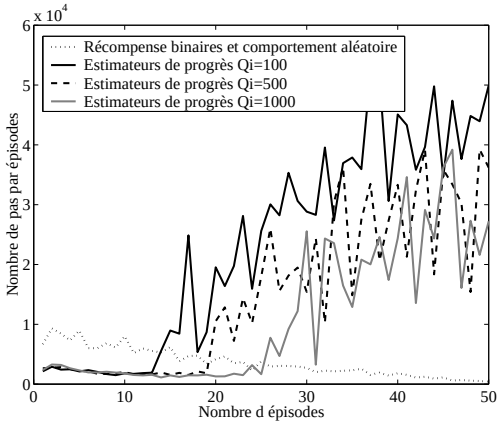


FIGURE 3.6 – Phénomène de désapprentissage engendré par les estimateurs de progrès. Expériences sur le labyrinthe moyennées sur 20 essais indépendants.

cette fonction et les paramètres peuvent être :

$$\begin{cases} \mathsf{R}(s,a,s') = -\varphi^2(s') = -\mathsf{d}_M^2(s', s_g) \\ Q_i = 0 \end{cases} . \tag{3.10}$$

Ainsi, l'agent est de moins en moins puni lorsqu'il s'approche de l'état cible. La stratégie globale est le plus court chemin vers l'état cible. Cependant, étant donné notre labyrinthe, cette forme de récompense est trompeuse pour l'agent car il y a de nombreux murs entre l'état initial et l'état cible. En effet, un biais dirigé vers le but est trompeur dans ce labyrinthe. Et cela engendre un **phénomène de désapprentissage** après quelques épisodes illustré à la figure 3.6. Le gradient attire l'agent vers le but mais, en se rapprochant du but, l'agent se retrouve bloqué dans une impasse. Il ne pourra en sortir qu'en explorant car les états sont de plus en plus attirants vers l'état cible. Au départ, l'agent trouve l'état but et donc ne suit pas le gradient car l'initialisation des Q valeurs engendre une **exploration systématique** en début d'apprentissage. Il est donc possible pour l'agent de sortir d'une impasse. Mais après quelques épisodes, ce biais disparaît et revenir en arrière est équivalent à choisir une valeur de Q moins attirante étant donné le gradient de récompense. Cela est possible seulement si plusieurs actions d'exploration se succèdent, c'est-à-dire **rarement**. On remarque de plus sur la figure 3.6 que plus les valeurs de Q_i sont élevées, plus le comportement d'exploration systématique persiste, et plus le phénomène de désapprentissage apparaît tardivement durant l'apprentissage. Cela confirme donc que *plus Q_i s'éloigne des bornes, plus le comportement de début d'apprentissage engendré est pérenne.*

Les méthodes par estimateurs de progrès sont donc risquées. Il est préférable d'utiliser ces approches avec circonspection dans la mesure où elles peuvent conduire à un comportement pernicieux.

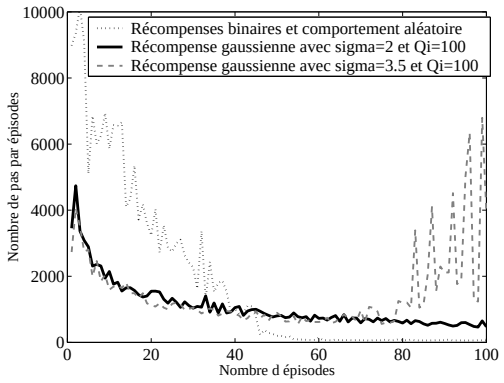


FIGURE 3.7 – Fonction de récompense gaussienne sous forme d'estimateur de progrès $\varphi(s') = 10e^{-\frac{\mathrm{d}_M(s',s_g)^2}{2\sigma^2}}$. Expériences sur le labyrinthe moyennées sur 20 essais indépendants.

3.4.2 Fonction de récompense gaussienne

Afin d'éviter le phénomène de désapprentissage, les récompenses doivent être **uniformes** pour un certain nombre d'états loin de l'état cible et le gradient de récompense doit être actif uniquement dans une zone proche de l'état cible. Nous suggérons un estimateur de progrès inspiré de **la fonction gaussienne** :

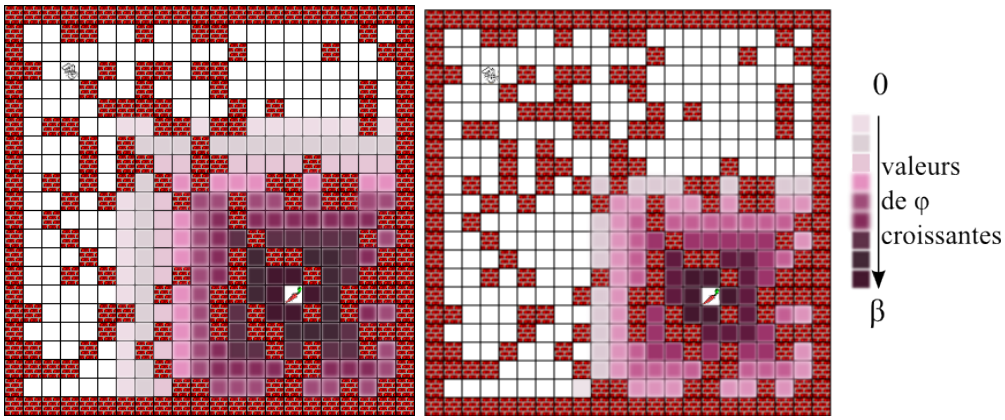
$$\varphi(s) = \beta e^{-\frac{\mathrm{h}(s)^2}{2\sigma^2}}. \tag{3.11}$$

Cette fonction de récompense gaussienne a l'avantage de générer un gradient sur les récompenses **ajustable**. Les valeurs de Q_i sont uniformes, β ajuste l'amplitude de la fonction et σ, l'écart type, caractérise la **zone d'influence du gradient de récompense**. Bien entendu, le comportement de piétinement devra être évité, c'est-à-dire $Q_i \geq \frac{\beta}{1-\gamma}$.

Expérimentations

Pour nos expériences avec le labyrinthe, nous avons choisi $Q_i = 100$, $\beta = 10$ et $\mathsf{R}(s,a,s') = \varphi(s')$. Sur la figure 3.7, le phénomène de désapprentissage est mis en évidence dès 80 épisodes avec $\sigma = 3.5$, c'est-à-dire que tous les états éloignés de plus de 10 pas de l'état cible ont une récompense identique. La zone d'influence du gradient généré par l'estimateur de progrès est illustrée en figure 3.8 par un code de couleur. Cette zone d'influence est trop large et quelques impasses y sont incluses dans lesquelles l'agent se bloque une fois que son exploration systématique s'atténue. A l'opposé, si la zone d'influence du gradient n'est activée que pour les états distants de 6 pas de s_g ($\sigma = 2$), nous n'avons pas constaté de phénomène de désapprentissage et le processus d'apprentissage est accéléré.

Une telle fonction de récompense continue est **ajustable** afin d'éviter un comportement nuisible et permet, si elle est correctement choisie, d'accélérer l'apprentissage.



(a) $\sigma = 3.5$ (b) $\sigma = 2$

FIGURE 3.8 – Zone d'influence du gradient de récompense généré par un estimateur de progrès $\varphi(s') = 10e^{-\frac{\mathrm{d}_M(s',s_g)^2}{2\sigma^2}}$. On utilise un code de couleur pour illustrer les différentes valeurs de l'estimateur de progrès φ selon l'état.

Somme toute, comme l'**influence est permanente**, il subsiste toujours un risque qu'une mauvaise heuristique génère un comportement à éviter tel qu'un phénomène de désapprentissage. C'est pourquoi la meilleure approche est de pouvoir influencer de manière éphémère l'apprentissage.

3.4.3 *Potential-based shaping*

Afin d'accélérer la vitesse de l'apprentissage, des approches sont basées sur l'utilisation d'un retour supplémentaire pour le contrôleur. En d'autres termes, cela consiste à appliquer une **structure de récompense additionnelle** qui aura un rôle de conseiller. Cette récompense supplémentaire va aider à guider le contrôleur vers une politique optimale. Pour cela, elle encourage un comportement qui pourrait éventuellement mener aux buts, ou en décourage un qui pourrait être regretté plus tard. L'objectif est d'apprendre une stratégie de commande pour un processus décisionnel de Markov donné $M = \langle \mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R} \rangle$. L'algorithme d'apprentissage est influencé par des récompenses additionnelles qui vont accélérer son apprentissage d'une politique de commande optimale. Pour cela, le contrôleur apprend pour un processus décisionnel de Markov transformé $M' = \langle \mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R}' \rangle$ où $\mathsf{R}' = \mathsf{R} + \mathsf{F}$ est la fonction de récompense du nouveau processus, F est la fonction de récompense additionnelle et R la fonction de récompense classique. Dans la figure 3.9, l'agent reçoit ainsi R provenant de l'environnement et F fourni par un observateur.

Il est important de comprendre l'impact d'une récompense additionnelle sur l'apprentissage.En effet, les récompenses additionnelles vont altérer le système de sorte que la stratégie de commande devienne optimale avec l'incorporation de ces nouvelles récompenses. Mais le nouveau comportement peut alors être assez différent de celui voulu au

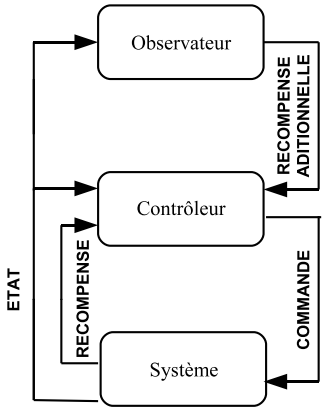


FIGURE 3.9 – Apprentissage influencé par des récompenses additionnelles.

départ. Prenons l'exemple du robot apprenant à jouer au football. Son objectif principal est de mettre un but et la récompense classique est donc de lui fournir un retour positif lorsqu'il rentre la balle dans la cage. Néanmoins, la possession de balle étant aussi importante, on peut ajouter une récompense additionnelle qui le récompense lorsqu'il touche la balle. Le robot va alors se détourner de son objectif premier qui est de mettre un but et va rester constamment proche de la balle, essayant de la toucher le plus fréquemment possible. Cette stratégie n'est clairement pas optimale pour notre processus décisionnel de Markov original. Cette simple récompense additionnelle distrait le contrôleur de son but premier et lui fait apprendre une tâche qu'il va répéter de manière cyclique.

Andrew Y. Ng, Daishi Harada et Stuart Russell [NHR99] étudient l'influence d'une structure de récompense additionnelle sur l'optimalité de la politique et propose le « façonnage » basé sur les fonctions de potentiel. On retrouve dans ces approches un problème de cycles oscillants proche de ce qui a été exposé précédemment. En effet, l'utilisation de champs de potentiel continus pose le problème de **cycles oscillants**, qui font entrer l'agent dans des cycles de boucles sans fin.

Afin de conserver l'invariance de la stratégie de commande, Andrew Y. Ng, Daishi Harada et Stuart Russell [NHR99] définissent une récompense additionnelle basée sur une **différence de potentiels**. La récompense additionnelle basée sur les potentiels[4] pour une transition de l'état s à s' en effectuant a est :

$$\mathsf{F}(s, a, s') = \gamma\varphi(s') - \varphi(s) \tag{3.12}$$

4. En anglais *potential-based shaping function*.

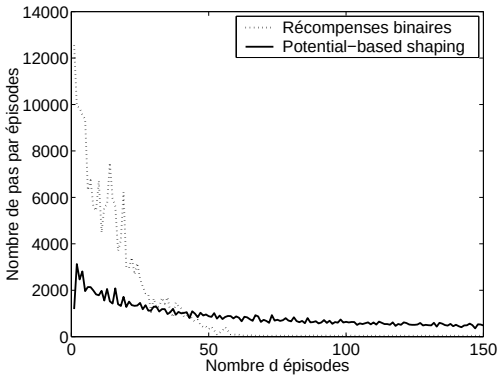


FIGURE 3.10 – *Potential-based shaping* avec $\varphi(s) = -\mathsf{d}_M^2(s, s_g)$ et $Q_i = 100$. Expériences sur le labyrinthe moyennées sur 20 essais indépendants.

où γ est le coefficient d'atténuation. Le critère, ou gain G (2.5), à maximiser est alors :

$$
\begin{aligned}
G &= \sum_{k=0}^{\infty} \gamma^k \mathsf{R}'_k(s_k, a_k, s_{k+1}) \qquad (3.13)\\
&= \sum_{k=0}^{\infty} \Big[\gamma^k (\mathsf{R}_k(s_k, a_k, s_{k+1}) + \mathsf{F}_k(s_k, a_k, s_{k+1}))\Big]\\
&= \sum_{k=0}^{\infty} \Big[\gamma^k (\mathsf{R}_k(s_k, a_k, s_{k+1}) + \gamma\varphi(s_{k+1}) - \varphi(s_k))\Big]\\
&= \sum_{k=0}^{\infty} \gamma^k \mathsf{R}_k(s_k, a_k, s_{k+1}) + \sum_{k=0}^{\infty} \gamma^{k+1}\varphi(s_{k+1}) - \sum_{k=0}^{\infty} \gamma^k \varphi(s_k)\\
&= \sum_{k=0}^{\infty} \gamma^k \mathsf{R}_k(s_k, a_k, s_{k+1}) - \varphi(s_0)
\end{aligned}
$$

Le gain à maximiser pour le PDM M' est le même à $-\varphi(s_0)$ près que celui du PDM M donc *cette récompense additionnelle garantit l'invariance de la stratégie de commande.* Toute politique optimale dans M' l'est aussi dans M et inversement.

Expérimentations

Nous testons cette méthode sur le labyrinthe. Pour cela, nous choisissons la fonction de potentiel $\varphi(s) = -\mathsf{d}_M^2(s, s_g)$ pour calculer la récompense additionnelle et une fonction de récompense de type *goal reward* pour la fonction de récompense classique. Les résultats sont donnés en figure 3.10. Cette méthode accélère nettement l'apprentissage dès le début. Par contre, un comportement d'exploration systématique doit être induit en début d'apprentissage pour éviter que l'agent ne se bloque dans les impasses durant

les premiers épisodes. Une autre fonction de potentiel pourrait aussi permettre d'éviter ce comportement pernicieux.

Un exemple classique de cette méthode est aussi proposé par J. Randløv et P. Alstrøm [RA98] concernant l'efficacité d'une fonction de récompense additionnelle sur l'apprentissage de la conduite d'un vélo. La récompense supplémentaire entraîne un retour local qui récompense les progrès du contrôleur vers le but.

3.4.4 Conclusion

Nous avons suggéré dans cette partie une fonction de récompense continue ajustable et moins risquée que les estimateurs de progrès. Toutefois, nous conseillons de rester prudent vis-à-vis des méthodes par *reward shaping* telles que les estimateurs de progrès et fonctions de potentiel, qui peuvent entraîner des comportements non désirés. La meilleure approche reste de conseiller de manière éphémère. E. Wiewiora [Wie03] démontre notamment certaines similarités entre les méthodes basées sur les fonctions de potentiel et l'initialisation de la table Q.

3.5 Conclusion

Une des limitations principales des algorithmes d'apprentissage par renforcement concerne la vitesse d'apprentissage, qui peut être en partie améliorée grâce à un choix pertinent de la fonction de récompense et des valeurs initiales de la table Q. Ces choix ont un impact majeur sur la performance des algorithmes d'apprentissage. Notre étude a abouti à l'élaboration de règles pour évaluer correctement ces paramètres selon le comportement désiré. Notamment, certaines valeurs de Q_i peuvent amener l'agent à avoir un comportement non désiré qui doit être évité. Nous avons de plus confirmé la présence de limites qui démarquent différents comportements de début d'apprentissage. Il est important de noter que plus Q_i s'éloigne de ces bornes, plus le comportement caractéristique est marqué.

Le choix des valeurs initiales de la fonction Q est une méthode efficace et peu risquée car elle est éphémère. Elle est notamment à préférer aux méthodes injectant de la connaissance dans la fonction de récompense qui peuvent entraîner des comportements non désirés. Nous développons dans notre étude une fonction d'influence générique qui initialise les Q valeurs et améliore les performances de l'apprentissage pour les problèmes de plus court chemin stochastique. La table 3.3 récapitule les choix que nous préconisons concernant la fonction de récompense et l'initialisation de la fonction de valeur pour des tâches de plus court chemin stochastique. Des résultats sur le problème du pendule inversé avec différents choix de paramètres pour un espace d'état continu sont exposés dans [MLFP06].

TABLE 3.3 – Choix préconisés pour la fonction de récompense et les valeurs initiales de Q dans le cadre de problèmes de plus court chemin stochastique.

FONCTION DE RÉCOMPENSE BINAIRE POUR UN ESPACE D'ÉTATS DISCRET
$\mathsf{R}(s,a,s') = \begin{cases} r_g \text{ si } s' = s_g \\ r_\infty \text{ else} \end{cases}$ Choix de r_g et r_∞ : $r_g \geq \frac{r_\infty}{1-\gamma}$
Choix des valeurs initiales uniformes de Q : $\mathsf{Q}_i = \frac{r_\infty}{1-\gamma} + \delta$
Choix des valeurs initiales influencées de Q : $\mathsf{Q}_i(s) = \beta e^{-\frac{\mathsf{h}(s)^2}{2\sigma^2}} + \delta + \frac{r_\infty}{1-\gamma}$
$\delta \geq 0$ fixe le niveau d'exploration systématique loin de l'état cible. $\beta > 0$ ajuste l'amplitude du gradient. σ fixe la zone d'influence du gradient et γ est le coefficient d'actualisation. s est l'ancien état, s' le nouveau, $\mathsf{h}(s)$ une heuristique.

Chapitre 4

Processus décisionnels de Markov et systèmes multiagents

Ce chapitre détaille le contexte de nos travaux en établissant les différents liens existant entre l'apprentissage par renforcement et les systèmes multiagents. Les concepts fondamentaux associés aux systèmes multiagents sont tout d'abord présentés. Ensuite, les notions nécessaires à l'extension de l'apprentissage par renforcement au cadre multiagent et les différents formalismes multiagents issus des processus décisionnels de Markov sont successivement exposés. A l'issu de ce chapitre, une synthèse établit des parallèles entre les concepts multiagents et ces formalismes.

4.1 Introduction

Les systèmes multiagents (SMA) sont un domaine de recherche issu de l'**intelligence artificielle distribuée**. L'une des grandes sources d'inspiration des SMA a été l'étude des comportements sociaux de certaines familles d'insectes, telles que les fourmis. A ce titre, les SMA forment un type intéressant de modélisation de sociétés. Leurs champs d'application sont larges : sciences humaines, théorie des jeux, économie, applications réelles telles que le contrôle de trafic aérien, la gestion de réseau ou la robotique. Les approches SMA s'intéressent aux **interactions** entre entités **autonomes**. Une situation d'interaction largement étudiée dans les SMA est la **coopération** qui fait appel à des mécanismes complexes tels que la coordination des actions.

Pour résoudre ces mécanismes, nous formalisons le problème comme un ensemble d'agents apprenants autonomes en interaction. Ces agents doivent apprendre à se coordonner afin de coopérer pour réaliser leur objectif. Ainsi, cette approche combine deux domaines : les systèmes distribués pour les aspects d'**interaction** entre agents et les techniques de l'apprentissage artificiel (*Machine Learning*) associées à un point de vue décentralisé pour les aspects de prise de décision [Les99, SV00, Vla03, PL05]. Pour cela, nous

utilisons le formalisme des processus décisionnels de Markov dans le cadre multiagent. Cette extension soulève différents problèmes, détaillés dans la suite de ce chapitre, parmi lesquels :

- le traitement des perceptions locales des agents,
- la gestion des communications entre agents,
- la décomposition du problème en sous-tâches et les récompenses associées.

Ce chapitre se compose de quatre parties. Nous commençons par introduire les SMA et les concepts d'agents, d'émergence et d'interaction qui sont indissociables des SMA. Le choix du cadre des processus décisionnels de Markov pour la conception de SMA implique de définir différentes notions qui seront détaillées dans une seconde partie. La troisième partie sera l'occasion de présenter les formalismes auxquels nous nous intéressons et qui étendent les processus décisionnels de Markov aux SMA. Enfin, une synthèse permettra de faire le lien entre les concepts propres aux SMA et ces formalismes.

4.2 Les systèmes multiagents

4.2.1 Intelligence artificielle distribuée

Depuis l'apparition de l'intelligence artificielle dans le domaine de la modélisation informatique (*cf.* §2.2.3) et la fameuse question d'Alan Turing : « Les machines peuvent-elles penser ? » [Tur50], les chercheurs se sont intéressés à recréer un système intelligent calqué sur l'intelligence humaine, c'est-à-dire une intelligence centralisée. Ainsi, l'intelligence artificielle classique modélise le comportement intelligent d'une seule entité. Au début des années 80, certains chercheurs ont tenté de remédier aux insuffisances présentées dans certains domaines par l'intelligence artificielle classique en proposant une distribution de l'expertise sur un groupe d'entités. Une nouvelle discipline est née : l'**intelligence artificielle distribuée** (IAD) [EHRLR80, BG88, JSW98]. L'IAD s'intéresse à des comportements intelligents qui sont le produit de l'**interaction** de plusieurs entités capables en quelque sorte d'associer leurs efforts pour accroître leur intelligence collective. Cette vision est très vite confortée par le développement des réseaux dans l'industrie informatique qui vont connaître une croissance exponentielle. En effet, ces réseaux imposent une vision décentralisée de l'implémentation et du contrôle. Cette vision est reprise en 1980 dans le premier système d'IAD avec Hearsay II et le modèle du tableau noir [1] pour la reconnaissance de la parole [EHRLR80]. Les techniques du tableau noir permettent à des modules appelés *sources de connaissance* de coopérer par partage d'informations. Un autre système à l'origine de l'IAD est le système DVMT [2] [LC88] pour le repérage des mouvements de véhicules à partir d'un réseau de capteurs séparés géographiquement. Ce système d'analyse de trafic routier est basé sur le regroupement d'observations fournies par ces capteurs. DVMT est donc un exemple d'application dans lequel la connaissance est distribuée. Les informations sont acheminées d'un sous-ensemble de capteurs à des-

1. En anglais *blackboard*.
2. De l'anglais *Distributed Vehicle Monitoring Testbed*.

tination de noeuds participant à la résolution du problème. Les noeuds coopèrent pour construire une image globale du trafic routier.

L'intérêt d'une distribution de l'intelligence est mis en évidence par Jacques Ferber [Fer95]. Tout d'abord, les problèmes à résoudre sont souvent physiquement et fonctionnellement distribués, par exemple dans le cas d'une équipe de robots footballeurs. De même, les réseaux comme internet ou le contrôle aérien sont des problèmes intrinsèquement distribués. De plus, chaque entité n'a souvent qu'une vision locale de l'ensemble du système, qui est soit imposée par la configuration du système, soit nécessaire pour simplifier la résolution de problèmes trop vastes pour être résolus globalement. La complexité des problèmes peut ainsi imposer des solutions basées sur des approches locales. Les différentes entités vont alors mettre leurs connaissances en commun. Enfin, les systèmes distribués constituent des architectures flexibles et modulables aptes à s'adapter à des modifications de structures ou d'environnement.

Le passage du comportement individuel aux comportements collectifs est considéré comme une extension mais aussi comme un enrichissement de l'intelligence artificielle, d'où émergent de nouvelles propriétés et de nouveaux comportements, tels que la coopération, la coordination d'actions, l'émergence. L'idée de base de l'IAD est que l'intelligence peut provenir de l'interaction d'un ensemble d'entités, ensemble appelé **système multiagents** (SMA). Ainsi, l'IAD peut être définie comme « l'analyse, la conception et l'étude de systèmes multiagents » [Bon94, Wei99].

4.2.2 Des agents aux systèmes multiagents

Avant de définir les systèmes multiagents (SMA), il faut déjà commencer par définir ce qu'est un **agent**. Une définition orientée vers les systèmes multiagents et souvent citée en référence est celle de Jacques Ferber :

Définition 4.1 *« On appelle agent une entité physique ou virtuelle*
- *qui est capable d'agir dans un environnement,*
- *qui peut communiquer directement avec les autres agents,*
- *qui 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),*
- *qui possède des ressources propres,*
- *qui est capable de percevoir (mais de manière limitée) son environnement,*
- *qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),*
- *qui possède des compétences et offre des services,*
- *qui peut éventuellement se reproduire,*
- *dont le comportement 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. » [Fer95]*

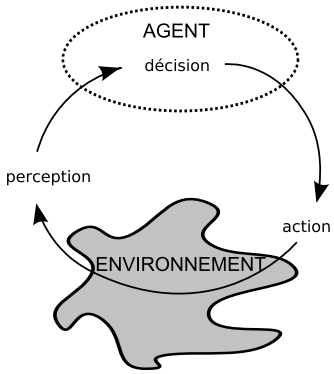


FIGURE 4.1 – Boucle sensori-motrice entre un agent et son environnement.

Cette définition évoque de nombreux concepts que l'auteur décrit dans son ouvrage et dont certains sont spécifiques à son domaine d'étude. Cette notion d'agent présente dans la littérature de multiples autres définitions [WJ95, FG96, RN03]. Nous dégageons de ces nombreuses définitions les notions principales communes à la plupart d'entre-elles pour aboutir à notre propre définition.

Définition 4.2 *Un agent est une entité capable de :*
- *percevoir, au moins partiellement, son environnement,*
- *agir sur son environnement,*
- *décider de manière autonome,*
- *communiquer avec d'autres agents.*

Par perception de l'environnement, un agent peut récolter des informations afin de mettre à jour ses représentations internes de l'environnement et/ou des autres agents. L'agent exploite alors ses perceptions pour raisonner et décider de façon autonome comment agir. Enfin, toute exécution d'une action modifie d'une certaine manière l'environnement et donc les perceptions de l'agent, qui décide alors d'une nouvelle action, et ainsi de suite. Ces trois phases forment donc un cycle par le bouclage de la phase action sur la phase perception. Ce cycle est représenté en figure 4.1. Il est très proche de la boucle sensori-motrice modélisant l'apprentissage par renforcement (*cf.* figure 2.1).

La définition d'un agent étant posée, nous pouvons maintenant nous intéresser aux SMA. De la même façon qu'il existe une multitude de définitions d'un agent, plusieurs définitions des SMA ont été données [Fer95]. Nous rappellerons ici la définition de Michael Wooldridge [Woo02].

Définition 4.3 *Un système multiagent est composé d'agents qui interagissent entre eux [Woo02].*

Jacques Ferber propose en 1994 [Fer94] le terme de **kénétique** pour désigner la science qui étudie les organisations artificielles et les interactions entre entités, qu'elles soient

informatiques, physiques, biologiques ... Ainsi, « l'action et l'interaction sont les éléments moteurs de la structuration d'un système multiagent dans son ensemble » [Fer95]. L'interaction entre les agents constitue donc la clé de voûte des SMA. En effet, c'est de l'**interaction** de comportements individuels qu'**émerge** un comportement global. Ainsi, les concepts d'interaction et d'émergence sont indissociables des SMA.

4.2.3 Concept d'émergence

L'apparition de phénomènes issus des interactions entre les agents dans un SMA se définit au travers de la notion d'**émergence**, notion fondamentale dans la recherche sur les SMA. Ainsi, « les systèmes multiagents prennent le parti de l'émergence » [Fer95]. Une définition d'un SMA orientée émergence est la suivante.

Définition 4.4 *Les SMA orientés émergence considèrent des ensembles d'agents autonomes interagissant pour résoudre un problème donné dont la difficulté est au-delà de leurs capacités ou connaissances individuelles.*

A la base de ce concept d'émergence, on trouve le postulat suivant : « Le tout est plus que la somme des parties ». Un exemple classique de cette notion s'inspire des comportements collectifs d'insectes sociaux tels que les fourmis. Les entomologistes ont toujours été étonnés par la capacité d'une colonie de fourmis à trouver le plus court chemin de son nid à une source de nourriture. Ils constatent un phénomène d'**émergence du plus court chemin**, phénomène lié à l'interaction des fourmis et non à la compétence propre d'un individu. On trouve aussi de nombreux comportements émergents dans les automates cellulaires, et particulièrement dans le **jeu de la vie** imaginé par J. H. Conway en 1970. Ce jeu se présente sous la forme d'un espace à deux dimensions ou damier généralement toroïdal de sorte à ne pas avoir de bords. Chaque case de l'espace contient une cellule qui peut prendre deux états, dénommées « vie » ou « mort ». Le jeu de la vie est basé seulement sur trois règles élémentaires :

- une cellule morte entourée d'exactement trois cellules vivantes « naît »,
- une cellule vivante entourée de deux ou trois voisines vivantes reste en vie,
- dans tous les autres cas, la cellule « meurt » ou reste morte.

En dépit de la simplicité de ces règles, il est possible d'engendrer une multitude de structures stables ou dynamiques, cohérentes et autonomes. On peut ainsi voir apparaître une configuration particulière de cinq cellules vivantes se reproduisant toutes les quatre générations à une cellule de distance, appelée planeur (*cf.* figure 4.2). Ce phénomène illustre bien le fait qu'un comportement complexe peut émerger de l'interaction entre des entités très simples. Ces exemples sont présentés plus en détail dans l'ouvrage de Jacques Ferber [Fer95] et dans celui de Jean-Philippe Rennard [Ren02]. Un aperçu des différentes manières d'utiliser le concept d'émergence dans certains domaines de l'intelligence artificielle est donné dans [Qui06].

4.2.4 Interactions dans les systèmes multiagents

L'**interaction** entre les agents est une notion centrale dans l'étude des SMA. Jacques Ferber en donne la définition suivante.

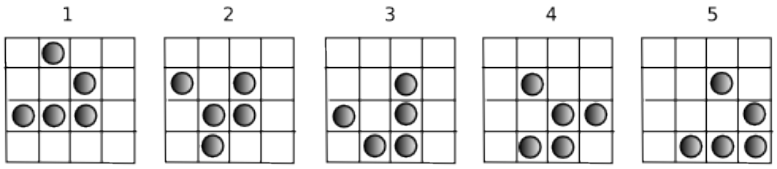


FIGURE 4.2 – Un planeur dans le jeu de la vie. Un rond représente une cellule vivante.

Définition 4.5 *« L'interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. » [Fer95]*

Les interactions sont les conséquences d'actions effectuées en même temps par plusieurs agents et sont un élément nécessaire à l'émergence de nouvelles fonctionnalités dans le groupe. Pour exister, les interactions supposent l'existence de **situations d'interaction**, telles qu'un point de rencontre entre agents ou le partage de ressources.

Définition 4.6 *« On appellera situation d'interaction un ensemble de comportements résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs en tenant compte des contraintes provenant des ressources plus ou moins limitées dont ils disposent et de leurs compétences individuelles. » [Fer95]*

Il existe différentes situations d'interaction que la kénétique se propose de définir et de classer selon trois critères principaux [Fer95] :

- **la nature des objectifs :** dans un SMA, les objectifs des agents peuvent être de diverses natures. Ainsi, les joueurs d'une équipe de football ont des objectifs compatibles dans le sens où lorsqu'un des joueurs met un but, toute l'équipe est satisfaite. Par contre, si l'on se place du point de vue de la partie de football, les objectifs des deux équipes participantes sont incompatibles car la satisfaction de l'une entraîne l'insatisfaction de l'autre. Ce critère permet une première classification : les agents sont dans une situation de **coopération** si leurs objectifs sont compatibles et dans une situation de **compétition** s'ils sont incompatibles. Si les agents peuvent atteindre leurs objectifs indépendamment des autres, ils sont dans une situation d'**indifférence**.
- **l'accès aux ressources :** si des robots mobiles doivent se déplacer d'une pièce à une autre et que ces pièces sont séparées par une porte ne permettant le passage que d'un seul robot à la fois, la porte devient alors une zone de conflit où la quantité de ressources en espace est limitée (*cf.* figure 4.3). Cette situation conflictuelle pourra alors être résolue si les robots coordonnent leurs actions. Le critère d'accès limité aux ressources (ressources énergétiques, financières, en temps, en espace, ...) détermine si la situation d'interaction nécessite des mécanismes de **coordination**.
- **les compétences des agents :** ce troisième critère concerne les capacités des agents par rapport aux tâches. La question qui se pose est alors la suivante : « Est-ce qu'un agent peut réaliser seul une tâche ou bien a-t-il besoin des autres pour parvenir à son but ? ». Avec le concept d'émergence, nous avons vu que des problèmes dont les difficultés sont au-delà des compétences individuelles des agents

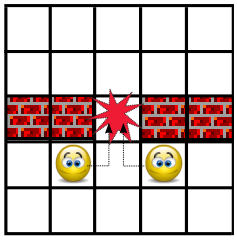
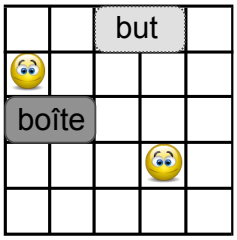
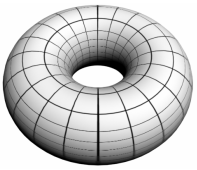


FIGURE 4.3 – Situation d'accès limité aux ressources.



(a) Tâche de *box-pushing*



(b) Monde toroïdal

FIGURE 4.4 – Situation de buts compatibles, ressources suffisantes et compétences insuffisantes nécessitant de la coordination pour une tâche de *box-pushing* [MC92] dans un monde toroïdal.

peuvent être résolus. Les compétences individuelles sont donc parfois insuffisantes pour mener à bien l'objectif. Dans ce cas, les agents doivent **collaborer** pour bénéficier de fonctionnalités émergentes.

Discussion

Selon les critères de Jacques Ferber, une situation de buts compatibles, ressources suffisantes et compétences insuffisantes requiert uniquement des mécanismes de collaboration par répartition de tâches. Or, des mécanismes de coordination peuvent aussi être nécessaires dans ces situations. Prenons l'exemple d'une situation de *box-pushing* où deux agents doivent déplacer une boite à un but précis dans un monde toroïdal et sans obstacles (*cf.* figure 4.4). C'est alors une situation de **coopération** car les buts des agents sont compatibles. Un agent seul ne peut pas déplacer l'objet donc les **compétences** sont insuffisantes. Les **ressources** en temps, en espace, ... ne sont quant à elles pas limitées. Néanmoins, les agents doivent tout de même mettre en place des mécanismes de coordination pour **synchroniser** leurs actions dans le temps et l'espace de sorte à déplacer l'objet à l'endroit désiré. Ainsi, la coordination est nécessaire quel que soit le type d'accès aux ressources.

Situations de coopération

La coopération est la forme générale d'interaction la plus étudiée dans les SMA. Dans le cadre de notre travail, nous nous intéressons aux situations de coopération, définies par Jacques Ferber [Fer95].

Définition 4.7 *Des agents sont dans une situation de coopération si leur objectifs sont compatibles mais que les ressources ou les compétences d'un ou plusieurs agents sont insuffisantes [Fer95].*

Ces situations de coopération font appel à des mécanismes de résolution tels que la collaboration par répartition de tâches et la coordination d'actions.

La **collaboration** consiste à travailler à plusieurs sur une tâche commune. Elle désigne l'ensemble des techniques permettant à des agents de se répartir le travail, les informations et les ressources lorsque leurs compétences individuelles sont insuffisantes. La **coordination** désigne quant à elle la manière dont les actions des différents agents doivent être organisées dans le temps et l'espace de manière à réaliser une tâche.

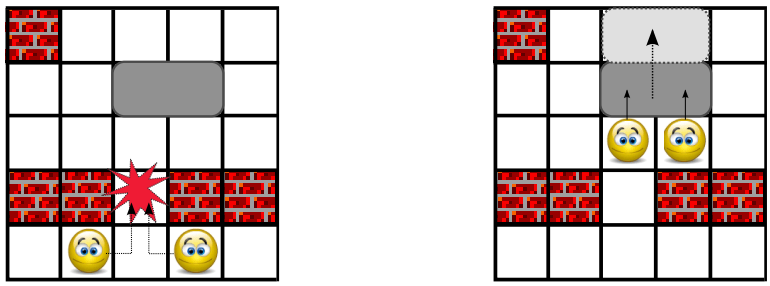
Définition 4.8 *« La coordination des actions, dans le cadre de la coopération, peut donc être définie comme l'articulation des actions individuelles accomplies par chacun des agents de manière à ce que l'ensemble aboutisse à un tout cohérent et performant. (...) La coordination des actions est donc l'une des principales méthodes pour assurer la coopération entre agents autonomes. » [Fer95]*

Pour illustrer ces différents sous-problèmes posés par les situations de coopération, prenons tout d'abord l'exemple de véhicules qui se croisent à un carrefour. Les agents doivent coordonner leurs actions les uns par rapport aux autres de manière à éviter les collisions. On est alors dans un cas typique de situation d'**encombrement** où les ressources en espace sont insuffisantes et demandent donc de la coordination, mais les agents n'ont pas besoin les uns des autres et donc ne collaborent pas. Les situations de coopération les plus complexes sont celles nécessitant une **collaboration coordonnée**. Prenons l'exemple d'une autre situation de *box-pushing*, illustrée en figure 4.5. Le but des agents est de pousser et déplacer une boîte à un endroit précis. On est donc dans une situation de coopération car les objectifs sont compatibles. La coordination est nécessaire lors de la première étape qui consiste pour les agents à atteindre la boîte sans se gêner, se heurter ou se bloquer mutuellement (*cf.* figure 4.5a). Ensuite, pour déplacer la boîte (*cf.* figure 4.5b), la **collaboration** est requise étant donné que les compétences individuelles sont insuffisantes car un agent seul ne peut pas déplacer la boîte. De plus, les agents doivent « additionner » leurs compétences au même moment, donc se synchroniser, afin que la résultante de leurs actions entraîne le déplacement de la boîte. Il s'agit donc ici d'une situation de coopération nécessitant collaboration et coordination d'actions, appelée **collaboration coordonnée**.

4.2.5 Conclusion

Nous avons défini dans cette section les notions d'agent et de système multiagent propres à notre étude. Nous avons aussi développé les concepts d'interaction et d'émergence indissociables des SMA. Les situations de coopération sont une forme d'interaction largement étudiée dans les SMA et qui apparaissent lorsque les agents ont des objectifs compatibles. Dans ce mémoire, nous nous situons dans le cadre des systèmes multiagents coopératifs. Le comportement collectif émerge du couplage à l'exécution des différents comportements des agents mis en présence. Nous avons détaillé les différentes situations de coopération et les mécanismes de résolution auxquels la coopération fait appel. La coordination des actions est ainsi une difficulté majeure à laquelle peuvent être confrontés des agents lors de la réalisation d'objectifs communs.

Notre objectif est d'utiliser des techniques d'apprentissage associées à un point de vue décentralisé pour résoudre les sous-problèmes posés par la coopération. Pour cela,



(a) Coordination nécessaire car ressources en espace insuffisantes

(b) Collaboration coordonnée pour que la résultante des actions individuelles entraîne le déplacement de la boîte

FIGURE 4.5 – Situation de coopération nécessitant des mécanismes de collaboration coordonnée pour une tâche de *box-pushing* [MC92].

le cadre mathématique des processus décisionnels de Markov, utilisé en apprentissage mono-agent, propose un formalisme intéressant qu'il est toutefois nécessaire d'étendre aux SMA.

4.3 Processus décisionnels de Markov et systèmes multiagents

L'extension des processus décisionnels de Markov (PDM) à des systèmes multiagents (SMA) implique de définir certaines notions nouvelles et soulève aussi quelques problèmes. Notamment, les états, actions et fonctions de récompense d'un PDM peuvent être définis à différents niveaux :

- un niveau **global** ou collectif dans lequel on observe la dynamique globale du système et les phénomènes émergents. C'est à ce niveau que l'on peut constater l'avancement de la résolution du problème, l'état global du système ainsi que l'atteinte de l'objectif global.
- un niveau **local** ou individuel dans lequel les agents perçoivent leur environnement et prennent leurs décisions.

De plus, selon le niveau dans lequel sont construites les politiques, le point de vue est centralisé ou décentralisé. Ces deux architectures de contrôle présentent chacune des avantages et des inconvénients qui seront détaillés dans ce paragraphe.

Nous allons tout d'abord développer les notions de **perception** de l'état et de l'action. En effet, les agents récupèrent des informations à un niveau local. Leur connaissance de l'état global du système peut donc être partielle, de même que l'accès aux actions de tous les agents du SMA. Afin d'avoir accès à des informations concernant ses congénères, et donc pour augmenter ses capacités de perception, un agent peut aussi **communiquer**. Les différentes manières de définir la **fonction de récompense** dans un SMA sont

exposées par la suite. Enfin, nous présenterons les différentes **architectures de contrôle** possibles dans les SMA.

4.3.1 Perception de l'état

Dans un SMA, les agents sont considérés à un niveau local. Ces contraintes de localité soulèvent des problématiques liées aux **observabilités partielles**.

Définition 4.9 *L'observabilité*[3] *d'un environnement caractérise l'ensemble des informations qui sont accessibles à un agent.*

Définition 4.10 *Une observabilité est partielle quand toutes les informations nécessaires à la connaissance de l'état du système ne sont pas accessibles.*

Différents types d'observabilités sont distingués selon que l'on se place à un niveau local (agent) ou global (SMA). David V. Pynadath et Milind Tambe [PT02] distinguent les quatre types suivant d'observabilités :

- **observabilité individuelle totale :** l'agent connaît l'état global du système à partir de ses perceptions individuelles.
- **observabilité individuelle partielle :** l'agent ne connaît pas l'état global du système à partir de ses perceptions individuelles.
- **observabilité collective totale :** l'état global du système peut être déduit de l'agrégation des observabilités individuelles de tous les agents.
- **observabilité collective partielle :** il n'est pas possible de déduire l'état global du système à partir de toutes les observations individuelles des agents.

Dans un PDM, un ensemble d'états $\mathcal{S}$ est défini. Ces états permettent de définir l'état global du système. Si tous les agents d'un SMA ont des observabilités individuelles totales, ils ont accès à cet état global. Par contre, dans le cas d'observabilités individuelles partielles, un agent i n'a accès qu'à des observations notées o_i. L'ensemble des observations d'un agent i est alors Ω_i et l'agrégation des observations de m agents est l'ensemble $\mathbf{\Omega} \equiv \Omega_1 \times ... \times \Omega_m$.

Reprenons par exemple le cas du *box-pushing*, illustré en figure 4.5. L'état global du système peut être la position de la boîte et des deux agents. Si chaque agent a accès à cet état global, les observabilités individuelles sont totales. Par contre, un agent peut avoir une perception locale limitée du monde qui l'entoure. Par exemple, il peut ne percevoir la boîte que selon deux critères de direction (*Nord*, *Sud*, *Est*, *Ouest*) et de proximité (Est-elle sur l'une des huit cases entourant l'agent ou non ?). Ainsi, différentes positions de la boîte peuvent entraîner les mêmes perceptions pour l'agent. La perception de l'autre agent peut être nécessaire afin par exemple d'éviter les collisions.

3. On parle aussi de *perception*.

4.3.2 Perception de l'action

Lorsque l'on étend les PDM à des SMA, il faut définir pour chaque agent i l'ensemble de ses **actions individuelles** $\mathcal{A}_i$. L'ensemble des **actions jointes** pour m agent est alors $\boldsymbol{\mathcal{A}} \equiv \mathcal{A}_1 \times ... \times \mathcal{A}_m$. Un agent i peut percevoir ses actions localement, c'est-à-dire uniquement l'action individuelle $a_i \in A_i$, ou percevoir les actions jointes $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, c'est-à-dire son action et celles des autres. Dans l'exemple du *box-pushing* (*cf.* figure 4.5), la perception des actions de l'autre agent est importante pour faciliter la coordination des actions afin d'éviter les collisions ou pour déplacer la boîte. Si l'agent ne perçoit que son action individuelle, il doit alors prendre sa décision et se coordonner malgré ce manque d'informations. Cette question de perception des actions individuelles ou des actions jointes dépend de la possibilité de communication entre les agents.

4.3.3 Communication

Percevoir la position de ses congénères ou leurs actions ouvre donc des possibilités pour améliorer la coopération et la coordination, comme nous venons de le voir dans le cas du *box-pushing*. Cet échange d'informations entre agents est réalisé au moyen de la communication. L'information échangée peut être de diverses natures : perceptions, actions, politiques ... C'est une forme d'interaction qui permet d'agrandir les capacités perceptives des agents et peut aussi améliorer la coopération et la coordination. La communication fait notamment partie des méthodes indispensables à la coopération d'après Jacques Ferber [Fer95].

Une information couramment échangée entre agents est l'action choisie par chacun. On peut ainsi classer les agents selon le type de perceptions d'actions auxquelles ils ont accès. Caroline Claus et Craig Boutilier [CB98a] distinguent deux types d'agents apprenants. Un **agent percevant les actions jointes** [4] (AAJ) perçoit son action individuelle ainsi que les choix d'actions des autres. Un AAJ perçoit donc l'action jointe. Dans ce cas, un AAJ peut par exemple modéliser le comportement des autres en calculant pour chacun de ses congénères la probabilité qu'ils choisissent une de leurs actions. Cette distribution de probabilités sur les actions des autres peut alors aider au choix d'une action coordonnée. Lorsque les agents entretiennent des croyances empiriques individuelles sur les politiques suivies par les autres agents, on parle de jeux fictifs [5] [Bro51]. Ces agents sont aussi appelés agents de niveau 1 [6] [VD98]. Le second type d'agent est l'**agent indépendant** [7] (AI) qui n'a accès qu'à sa propre action individuelle. Ce type d'agent est aussi appelé agent de niveau 0 [8] en tant qu'agent ne maintenant pas de modèles explicites de ses congénères. Dans ce cas, les actions des autres sont inaccessibles. Il faut donc faire des hypothèses sur leurs choix d'actions ou apprendre à se coordonner sans ces informations.

4. En anglais *joint action learner.*
5. En anglais *fictitious play.*
6. En anglais *1-level agent.*
7. En anglais *independent learner.*
8. En anglais *0-level agent.*

Ainsi, permettre une certaine forme de communication entre agents, pour par exemple connaître les actions des autres, amène des possibilités pour améliorer la coordination et peut aussi accélérer l'apprentissage des agents mais aux dépens de certains inconvénients. En effet, la communication peut compliquer les algorithmes [Kla03] et avoir un impact sur la performance du système [BA94]. Notamment, les informations échangées doivent être efficaces pour que la communication soit bénéfique aux agents [Tan93, SST95]. De plus, les ressources mémoires nécessaires peuvent aussi être plus importantes. Ainsi, avec des AAJ, celles-ci sont exponentielles avec le nombre d'agents pour certains algorithmes d'apprentissage par renforcement. Enfin, les problèmes de coordination ne sont pas complètement résolus lorsque les actions jointes sont perçues, comme le montrent [CB98a].

4.3.4 Distribution des récompenses

Nous nous plaçons dans le cadre de SMA où l'objectif est défini au niveau global. Par exemple, l'objectif pour un groupe d'agents peut être de déplacer une boîte à un endroit précis dans le cas du *box-pushing*, de gagner un match de football pour une équipe de robots footballeurs ou d'éviter des collisions entre des avions pour le trafic aérien. La mesure de performance ne s'effectue pas au niveau du comportement local d'un agent mais au niveau du groupe. Dans un SMA, les comportements locaux des agents et leurs récompenses associées doivent donc permettre d'assurer l'objectif défini de manière globale. C'est le problème de la **distribution des récompenses** : celles-ci doivent-elles être distribuées localement ou globalement ?

- l'objectif étant défini de manière globale, la récompense peut être définie à ce niveau par une fonction commune à tous les agents $\mathsf{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ selon l'état global $s \in \mathcal{S}$ et les actions jointes $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$. Une première méthode consiste à répartir cette récompense globale parmi les agents responsables de l'avancement de la résolution. Il est alors nécessaire dans ce cas de savoir quels agents récompenser. C'est ce qu'on appelle le problème du *credit assignment*. La récompense globale peut aussi être distribuée à tous les agents du SMA. Un agent peut alors recevoir une récompense parce que d'autres agents du groupe ont permis l'avancement de la résolution, et non pas grâce à son propre comportement. Il doit donc savoir dans quelle mesure son comportement est lié à la récompense qu'il reçoit. De nombreux formalismes de SMA coopératifs utilisent cette distribution de la récompense globale à tous les agents (*cf.* §4.4). Il faut alors distribuer à un niveau local cette récompense commune définie globalement, ce qui peut nécessiter de la communication ou une certaine forme de centralisation.
- la récompense peut aussi s'exprimer de manière locale, c'est-à-dire comme une fonction individuelle pour chaque agent i qui dépend de l'action jointe et des perceptions de chaque agent. La fonction de récompense est définie par $\mathsf{R}_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$. Chaque agent met à jour son comportement selon le retour local. Par contre, un agent qui effectue une action peut réduire ou augmenter la récompense reçue par un autre agent. On peut alors se poser la question suivante : « Maximiser ses satisfactions individuelles conduit-il forcément à maximiser la satisfaction du groupe ? ». Selon Garrett Hardin [Har68], la maximisation individuelle des satisfactions lo-

cales peut conduire à la ruine du groupe. C'est ce qu'il appelle la **tragédie des biens communs**[9]. Prenons l'exemple de la décroissance des sociétés de pêche pour illustrer ce phénomène. La mer constitue un « bien commun », une réserve de poissons commune. La quantité de poissons dans la mer est juste suffisante pour permettre aux bateaux de pêche d'avoir des prises individuelles correctes. Le fait d'augmenter la quantité de poisson par prises individuelles diminuerait la capacité de renouvellement des populations de poissons. Pourtant, il semble que chaque pêcheur ait avantage à augmenter la taille de ses prises individuelles. En effet, les pêcheurs sont soumis à une pression financière qui les pousse à capturer davantage de poissons. Du point de vue de chaque pêcheur, le gain est important et la perte faible car la diminution des stocks de poissons se répercute sur l'ensemble des pêcheurs et n'est donc pas un réel inconvénient à son échelle. Mais la situation devient tragique à partir du moment où tous les pêcheurs font la même chose : les stocks de poissons dans la mer diminuent et sont incapables de se renouveler.

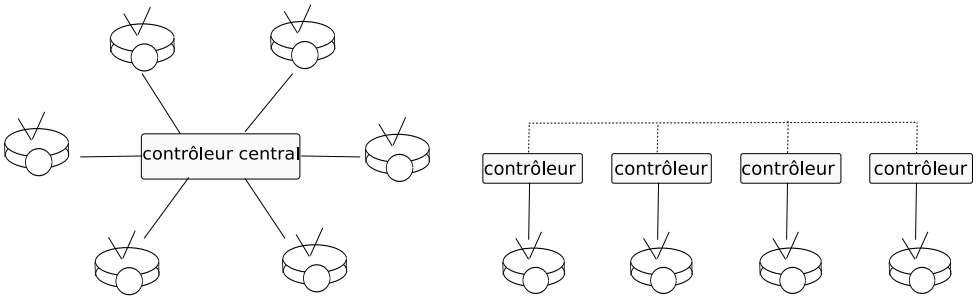
Peu de solutions sont proposées pour répondre à ces problèmes. Concernant la tragédie des communs, Kagan Tumer et David Wolpert [TW00] essaient de trouver comment, à partir d'un problème donné, il est possible de définir les fonctions de valeur locales des agents afin de s'assurer que de bons comportements individuels induisent un bon comportement collectif. Cette méthode est appelée **méthode COIN** pour *COllective INtelligence*. Néanmoins, il semble peu évident de pouvoir réellement appliquer cette méthode. Ainsi, *distribuer collectivement à tous les agents la récompense globale* et laisser à chacun d'eux le soin de déterminer dans quelle mesure leur comportement est lié à cette récompense *est une solution intéressante dans le cadre des SMA coopératifs*. C'est cette solution que nous allons mettre en oeuvre dans le cadre de l'apprentissage par renforcement dans les SMA (*cf.* §6). Chaque agent apprend si la récompense reçue est une conséquence de son comportement ou de celui des autres.

4.3.5 Architectures de contrôle

Résoudre un problème de décision dans un système multiagent consiste à calculer une **politique jointe** $\boldsymbol{\pi} =< \pi_1, ..., \pi_m >$ où π_i correspond à la politique de l'agent i et m est le nombre d'agents. La politique d'un agent i est une fonction notée $\pi_i : \mathcal{S} \times \mathcal{A}_i \mapsto [0; 1]$ qui définit une distribution de probabilités sur la perception $s \in \mathcal{S}$ de l'agent (ou sur l'état global si celui-ci est individuellement observable) et son action $a_i \in \mathcal{A}_i$. Comme dans le cas mono-agent, un contrôleur récupère l'ensemble des informations nécessaires (état, récompense), calcule la politique de l'agent et décide de l'action à effectuer. Lors de l'extension d'un PDM aux SMA, le processus qui construit les politiques individuelles peut être défini à différents niveaux :

- un niveau global : on parle alors d'**architecture centralisée** (*cf.* figure 4.6a). Un contrôleur central dispose de l'ensemble des informations : l'état global du système $s \in \mathcal{S}$, l'action jointe $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$ et la récompense R. Il calcule la politique jointe $\boldsymbol{\pi}$

9. En anglais *The Tragedy of the Commons*.



(a) centralisée (b) hétérarchique ou décentralisée

FIGURE 4.6 – Architectures de contrôle possibles pour un groupe de robots mobiles.

et distribue les commandes individuelles $a_1, ..., a_m$ parmi les m agents selon cette politique. Il détient donc le pouvoir de décision et maintient l'information globale sur l'ensemble des agents.

– un niveau local : on parle alors d'**architecture hétérarchique ou décentralisée** (*cf.* figure 4.6b). La notion essentielle de cette structure est la considération d'agents totalements autonomes. Chaque agent i a son propre contrôleur qui construit une politique individuelle π_i à partir d'informations locales : la perception locale de l'agent, son action individuelle a_i et la récompense individuelle R_i. Les agents sont donc tous considérés au même niveau, collectent de l'information locale et agissent sur une partie du système. Ce sont souvent des structures coopératives regroupant de multiples unités de contrôle en interaction afin de réaliser un objectif global.

Imaginons un système composé de plusieurs robots mobiles qui doivent effectuer une tâche commune. Deux approches sont alors possibles. La première consiste à utiliser un ordinateur qui contrôle l'ensemble des robots et a accès à l'état global du système, grâce par exemple à une caméra qui filme toute la scène. Le système utilise donc un contrôleur unique centralisé ; on parle alors d'**architecture centralisée**. Une seconde approche dote chaque robot de capacités de réflexion, de perception et de décision propres. Dans ce cas, chaque robot est autonome et l'architecture est appelée **hétérarchique** ou **décentralisée**.

Ces deux architectures de contrôle présentent chacune certains avantages et inconvénients que nous détaillons ici de manière non exhaustive :

– **architecture centralisée :** Les principaux inconvénients d'une architecture centralisée sont qu'elle est **peu robuste** en cas de défaillance du contrôleur central et difficilement modifiable à cause de la **non modularité**. Concernant les perceptions, le contrôleur central doit disposer à chaque instant de l'**information globale** sur le système, ce qui n'est pas toujours réaliste. Enfin, la construction d'une politique jointe de manière centralisée est un problème très complexe [BGIZ02]. Les avan-

tages sont qu'un nombre limité d'unités de contrôles et de moyens de traitement sont nécessaires. De plus, **aucun mécanisme de coordination** ne doit être implémenté car le contrôleur central calcule directement la politique jointe optimale. Concernant les communications, le contrôleur central doit **communiquer** à tous les agents les actions individuelles.

- **architecture hétérarchique ou décentralisée :** Les avantages sont une amélioration de la **modularité** du système avec la possibilité d'ajouter ou de retirer facilement des agents et une **plus grande robustesse**. De plus, les agents autonomes ont aussi souvent des tâches simples à résoudre et le comportement global émerge de leurs interactions. La politique locale a aussi souvent besoin de considérer moins de variables. Nous pouvons toutefois souligner quelques inconvénients de cette structure, tels que l'accès à des **perceptions partielles**. La construction des politiques individuelles se fait alors à partir d'observations parfois incomplètes. La **communication** entre les agents peut être nécessaire pour que chacun ait accès aux actions jointes ou pour partager des perceptions locales. Enfin, la **coordination** reste la difficulté majeure des architectures décentralisées. Outre la difficulté de calculer des politiques individuelles optimales avec des perceptions limitées, s'ajoute le problème d'assurer que ces politiques individuelles optimales définissent une politique jointe optimale.

L'extension des PDM aux SMA conduit naturellement aux **architectures décentralisées**. En effet, ces approches considèrent des entités totalement autonomes, ce qui est en accord avec notre définition d'agents et de SMA. On remarque d'ailleurs l'emploi abusif du terme système multiagent pour des systèmes qui ne sont pas vraiment décentralisés. Par exemple, dans le cas des robots mobiles, ce système est appelé système multiagent car plusieurs robots sont recensés dans l'environnement. Néanmoins, si une architecture centralisée est choisie, ce système n'est pas multiagent au sens où nous l'avons défini précédemment car il n'y a qu'une unité de contrôle centrale. Pour que ce système soit réellement multiagent, il faut que chaque robot dispose de sa propre capacité de réflexion et de décision.

4.3.6 Conclusion

Dans cette section, l'ensemble des notions propres à un PDM ont été redéfinies dans le cas d'une extension à des SMA. Les différentes possibilités d'extension nécessitent le choix du niveau de définition de ces notions de perception de l'état, de distribution des récompenses et de communication. Les différents formalismes utilisés dans le cadre de l'apprentissage par renforcement multiagent permettent de recouvrir l'ensemble de ces possibilités. Nous allons maintenant présenter les formalismes supposant une perception individuelle totale de l'état par chaque agent indépendant, ce qui correspond au cadre de notre étude.

4.4 Différents modèles multiagents issus des processus décisionnels de Markov

Nous présentons tout d'abord les jeux matriciels, formalisme multiagent et sans états issu de la théorie des jeux. L'extension de ce modèle à des environnements à plusieurs états a abouti aux jeux de Markov. Ceux-ci modélisent des problèmes de décision entre plusieurs agents dans un environnement avec des transitions stochastiques. Au départ principalement étudiés dans la théorie des jeux, les jeux de Markov ont pris de l'importance récemment dans le cadre de l'apprentissage par renforcement multiagent. Les objectifs à atteindre dans ces jeux seront précisés. Nous conseillons en lecture complémentaire la thèse de Michael Bowling [Bow03] et l'ouvrage [PDM08].

4.4.1 Jeux matriciels

Les **jeux matriciels** sont souvent utilisés pour l'étude de problèmes de décision multiagents car ils modélisent de manière abstraite les interactions entre des agents qui coexistent dans un environnement et prennent des décisions simultanément ou de manière séquentielle. Cette section définit le cadre des jeux matriciels, leur classification et les principaux concepts utilisés pour leur résolution.

Jeu matriciel

Un jeu matriciel est un formalisme à plusieurs agents et sans états. Il peut aussi être vu comme un PDM ayant un seul état, un grand vecteur d'action et de multiples fonctions de renforcement.

Définition 4.11 *Un jeu matriciel*[10] *[NM44, OR94] est un n-uplet* $\langle m, \mathcal{A}_1, \ldots, \mathcal{A}_m, \mathsf{R}_1, \ldots, \mathsf{R}_m \rangle$ *où*

- m *est le nombre d'agents appelés aussi joueurs,*
- $\mathcal{A}_i$ *est l'ensemble des actions pour l'agent* i *(et* $\mathcal{A}_1 \times \ldots \times \mathcal{A}_m \equiv \boldsymbol{\mathcal{A}}$ *est l'ensemble des actions jointes),*
- $\mathsf{R}_i : \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ *est la fonction de récompense*[11] *de l'agent* i *qui dépend de l'action jointe des agents.*

Ces jeux sont appelés « jeux matriciels » car les fonctions de récompense R_i peuvent être représentées sous forme matricielle, comme dans l'exemple de la figure 4.7. On parle alors de **jeu en forme stratégique**.

Types de jeux matriciels

Les jeux matriciels peuvent être classés selon les caractéristiques de leur fonction de récompense.

10. En anglais *matrix game*.
11. Généralement appelée *fonction de gain* dans le cas des jeux matriciels. En anglais *payoff function*.

Définition 4.12 *On appelle* ***jeu d'équipe***[12] *un jeu matriciel tel que tous les agents ont la même fonction de récompense,* i.e. *:*

$$\forall \mathbf{a} \in \boldsymbol{\mathcal{A}} \; \forall i \; \forall j \quad \mathsf{R}_i(\mathbf{a}) = \mathsf{R}_j(\mathbf{a}). \tag{4.1}$$

Définition 4.13 *On appelle* ***jeu à somme nulle***[13] *un jeu matriciel tel que la somme des fonctions de récompense de tous les agents est nulle,* i.e. *:*

$$\forall \mathbf{a} \in \boldsymbol{\mathcal{A}} \quad \sum_{i=1}^{m} \mathsf{R}_i(\mathbf{a}) = 0. \tag{4.2}$$

Définition 4.14 *On appelle* ***jeu à somme générale***[14] *un jeu matriciel qui n'est ni un jeu d'équipe ni un jeu à somme nulle.*

Par exemple, les jeux Pierre-Feuille-Ciseau (*cf.* figure 4.7) et pile ou face (*cf.* figure 4.9) sont des jeux à somme nulle car $\forall \boldsymbol{a} \in \boldsymbol{\mathcal{A}} \quad \mathsf{R}_1(\boldsymbol{a}) + \mathsf{R}_2(\boldsymbol{a}) = 0$. Dans le cas des jeux d'équipe, les récompenses sont représentées par une matrice unique étant donné qu'elles sont identiques pour tous les agents.

Un jeu matriciel est **déterministe** si pour toute action jointe $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$ et pour tout agent i, la probabilité de la récompense $\mathsf{R}_i(a)$ est de 1. Dans un jeu matriciel **stochastique**, des récompenses différentes peuvent être reçues avec diverses probabilités pour une même action jointe et par un même agent.

Les jeux dits **statiques**[15] se jouent en un seul tour pendant lequel tous les joueurs choisissent simultanément leur coup. Lorsque l'on s'intéresse à l'apprentissage d'agents au sein d'un groupe, il faut faire interagir les agents les uns avec les autres de manière répétitive. C'est le principe des jeux **répétés**[16] qui consistent en la répétition d'un même jeu statique par les mêmes agents.

Stratégie

Dans les jeux matriciels, l'objectif des agents est d'apprendre une stratégie qui permette à l'agent de maximiser ses récompenses. Une stratégie dans un jeu matriciel est l'équivalent de la politique dans un PDM, c'est pourquoi nous utiliserons la même notation π pour une stratégie.

Définition 4.15 *Une* ***stratégie*** *pour un agent i est une fonction $\pi_i : \mathcal{A}_i \mapsto [0;1]$ qui définit une distribution de probabilités sur les actions de l'agent,* i.e. *:*

$$\forall a_i \in \mathcal{A}_i \quad P(a_i) = \pi_i(a_i). \tag{4.3}$$

12. En anglais *team game.*
13. En anglais *zero-sum game.*
14. En anglais *general-sum game.*
15. Ou jeux *simultanés.*
16. Ou jeux *itérés.*

		Agent 2		
		P	**F**	**C**
Agent 1	**P**	0	-1	1
	F	1	0	-1
	C	-1	1	0

(a) Matrice du joueur 1 (R_1)

		Agent 2		
		P	**F**	**C**
Agent 1	**P**	0	1	-1
	F	-1	0	1
	C	1	-1	0

(b) Matrice du joueur 2 (R_2)

FIGURE 4.7 – Le jeu Pierre-Feuille-Ciseau sous sa forme stratégique. Ce jeu se joue à deux joueurs. Les lignes et colonnes correspondent aux actions possibles respectivement du premier et du second agent. Les valeurs dans les cases des matrices indiquent les récompenses de chaque joueur selon l'action jointe effectuée. Une récompense de 1 indique que le joueur gagne et de -1 qu'il perd.

		Agent 2	
		dénoncer	**nier**
Agent 1	**dénoncer**	-2	0
	nier	-5	-1

(a) Matrice du joueur 1 (R_1)

		Agent 2	
		dénoncer	**nier**
Agent 1	**dénoncer**	-2	-5
	nier	0	-1

(b) Matrice du joueur 2 (R_2)

FIGURE 4.8 – Le dilemme du prisonnier représente une situation où deux prisonniers sont complices d'un délit. Si un seul des deux prisonniers dénonce l'autre, il sort tout de suite de prison et l'autre prend un maximum de 5 ans de prison. Si les deux dénoncent, ils passent chacun 2 ans en prison. Si les deux se taisent, ils feront chacun un an de prison.

		Agent 2	
		pile	**face**
Agent 1	**pile**	1	-1
	face	-1	1

(a) Matrice du joueur 1 (R_1)

		Agent 2	
		pile	**face**
Agent 1	**pile**	-1	1
	face	1	-1

(b) Matrice du joueur 2 (R_2)

FIGURE 4.9 – Le jeu du pile ou face (*matching pennies*).

		Agent 2	
		ciné	**foot**
Agent 1	**ciné**	2	0
	foot	0	1

(a) Matrice du joueur 1 (R_1)

		Agent 2	
		ciné	**foot**
Agent 1	**ciné**	1	0
	foot	0	2

(b) Matrice du joueur 2 (R_2)

FIGURE 4.10 – La guerre des sexes (*battle of the sexes*). Un couple doit choisir où il va passer sa soirée. Ils souhaitent tous deux être ensemble, mais chacun a une préférence : la femme voudrait aller au cinéma et l'homme préférerait aller voir un match.

On a la contrainte suivante :

$$\sum_{a_i \in \mathcal{A}_i} \pi_i(a_i) = 1 \tag{4.4}$$

On utilise la notation $\boldsymbol{\pi}_{-i}$ pour désigner la stratégie jointe de tous les agents excepté l'agent i et $\boldsymbol{\pi}$ la stratégie jointe de tous les agents. La notation $\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$ désigne la stratégie jointe des agents pour laquelle l'agent i suit la stratégie π_i et les autres agents leur stratégie respective définie dans $\boldsymbol{\pi}_{-i}$. Nous noterons $\Delta(\mathcal{A}_i)$ l'ensemble de toutes les distributions de probabilité qui recouvre l'ensemble $\mathcal{A}_i$, *i.e.* l'ensemble de toutes les stratégies pour l'agent i.

Définition 4.16 *Une stratégie est dite **pure** si et seulement si elle est déterministe.*

Définition 4.17 *Une stratégie est dite **mixte** si et seulement si elle est non déterministe.*

Par abus de notations, nous désignerons par $a_i \in \mathcal{A}_i$ la **stratégie pure** pour un agent i. Dès lors, $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$ est une combinaison des actions de tous les agents, appelée **stratégie pure jointe** ou **action jointe**. Conformément aux notations précédentes, on désigne par $\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}$ l'ensemble de toutes les actions choisies par les agents sauf celle de l'agent i. Enfin, nous utiliserons la notation $\langle a_i, \boldsymbol{a}_{-i} \rangle$ pour désigner la stratégie pure jointe où l'agent i choisit a_i tandis que les autres suivent la stratégie $\boldsymbol{a}_{-i}$.

Par exemple, pour le jeu du dilemme du prisonnier (*cf.* figure 4.8), une stratégie pure peut être pour les deux agents de dénoncer l'autre. Dans certains cas, les stratégies pures sont insuffisantes. Prenons l'exemple du jeu du pile ou face représenté en figure 4.9. Si un agent joue une des actions de façon déterministe (soit *pile*, soit *face*), alors l'autre agent peut être garanti de gagner en jouant l'action appropriée. Une stratégie pure est par conséquent insuffisante et les agents doivent alors suivre une stratégie mixte.

Équilibres

Nous définissons tout d'abord le **gain espéré** d'un agent comme l'espérance de récompense étant données sa stratégie et celles des autres agents.

Définition 4.18 *Le* ***gain espéré*** *de l'agent i selon la stratégie jointe $\boldsymbol{\pi} = \langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$, noté $\mathsf{u}_{i,\boldsymbol{\pi}}$, est :*

$$\begin{aligned}\mathsf{u}_{i,\boldsymbol{\pi}} &= E_{\boldsymbol{\pi}}\{\mathsf{R}_i(\mathbf{a})\} \\ &= \sum_{\mathbf{a}\in\boldsymbol{\mathcal{A}}} \boldsymbol{\pi}(\mathbf{a})\mathsf{R}_i(\mathbf{a}) \\ &= \sum_{a_i\in\mathcal{A}_i} \sum_{\mathbf{a}_{-i}\in\boldsymbol{\mathcal{A}}_{-i}} \mathsf{R}_i(\langle a_i, \mathbf{a}_{-i}\rangle)\pi_i(a_i)\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}).\end{aligned} \tag{4.5}$$

Par exemple, dans le dilemme du prisonnier, soit la stratégie mixte où l'agent 1 dénonce l'autre avec une probabilité de $0,4$ (et donc il nie avec une probabilité de $0,6$) et l'agent 2 dénonce avec une probabilité de $0,8$ (et donc il nie avec une probabilité de $0,2$). Le gain espéré de l'agent 1 est obtenu avec la fonction u_1 suivante :

$$\begin{aligned}u_{1,\langle\pi_1,\pi_2\rangle} &= \pi_1(\textit{dénoncer})\pi_2(\textit{dénoncer})\mathsf{R}_1(\langle \textit{dénoncer}, \textit{dénoncer}\rangle) + \\ &\quad \pi_1(\textit{dénoncer})\pi_2(\textit{nier})\mathsf{R}_1(\langle \textit{dénoncer}, \textit{nier}\rangle) + \\ &\quad \pi_1(\textit{nier})\pi_2(\textit{dénoncer})\mathsf{R}_1(\langle \textit{nier}, \textit{dénoncer}\rangle) + \pi_1(\textit{nier})\pi_2(\textit{nier})\mathsf{R}_1(\langle \textit{nier}, \textit{nier}\rangle) \\ &= 0,4 \times 0,8 \times (-2) + 0,4 \times 0,2 \times 0 + 0,6 \times 0,8 \times (-5) + 0,6 \times 0,2 \times (-1) \\ &= -3,16\end{aligned} \tag{4.6}$$

Dans la théorie des jeux, la solution optimale est définie à partir de deux concepts. Le premier concept a conduit au grand développement des jeux matriciels et de la théorie des jeux. C'est la notion d'**équilibre de Nash** [Nas50, OR94].

Définition 4.19 *Une stratégie jointe $\boldsymbol{\pi}^*$ définit un équilibre de Nash si, pour chaque agent i, on a :*

$$\forall \pi_i \in \Delta(\mathcal{A}_i) \quad \mathsf{u}_{i,\langle \pi_i^*, \boldsymbol{\pi}_{-i}^*\rangle} \geq \mathsf{u}_{i,\langle \pi_i, \boldsymbol{\pi}_{-i}^*\rangle}. \tag{4.7}$$

Autrement dit, aucun agent ne peut améliorer son gain espéré en déviant unilatéralement d'un équilibre de Nash. Ainsi, un équilibre de Nash a une propriété de « stabilité » qui est satisfaite pour chacun des agents, c'est pourquoi on parle d'« équilibre ». Un jeu matriciel peut avoir plusieurs équilibres de Nash. Il peut aussi ne comporter aucun équilibre de Nash en stratégies pures. Par contre, on a le théorème suivant :

Théorème 4.1 *Tout jeu en forme stratégique fini admet au moins un équilibre de Nash en stratégies mixtes [Nas51].*

Par exemple, dans le jeu du dilemme du prisonnier (*cf.* figure 4.8), l'équilibre de Nash en stratégie pure correspond au cas où les deux agents choisissent de dénoncer. En effet, si un des deux agents dévie et décide de nier, il rallongera sa peine de prison. Dans le jeu du pile ou face (*cf.* figure 4.9), il n'y a pas d'équilibre de Nash en stratégies pures. L'équilibre en stratégies mixtes dans ce jeu est :

$$\forall i \ \forall j \quad \pi_i(a_j) = \frac{1}{2} \tag{4.8}$$

qui correspond au fait que chaque agent choisit une de ses actions de manière équiprobable. Dans le jeu de la guerre des sexes (*cf.* figure 4.10), les deux équilibres de Nash en stratégies pures sont les cas où les deux agents choisissent d'aller au cinéma ou d'aller voir un match de foot. L'équilibre en stratégie mixte est :

$$\pi_1(cine) = \frac{2}{3} \quad \pi_1(foot) = \frac{1}{3} \quad \pi_2(cine) = \frac{1}{3} \quad \pi_2(foot) = \frac{2}{3}. \tag{4.9}$$

Cet équilibre correspond à un gain espéré pour chaque agent de :

$$\begin{aligned} \mathsf{u}_{1,\langle\pi_1,\pi_2\rangle} &= \pi_1(cine)\pi_2(cine)\mathsf{R}_1(\langle cine, cine\rangle) + \pi_1(foot)\pi_2(foot)\mathsf{R}_1(\langle foot, foot\rangle) \\ &= \frac{2}{3}. \\ \mathsf{u}_{2,\langle\pi_1,\pi_2\rangle} &= \pi_1(cine)\pi_2(cine)\mathsf{R}_2(\langle cine, cine\rangle) + \pi_1(foot)\pi_2(foot)\mathsf{R}_2(\langle foot, foot\rangle) \\ &= \frac{2}{3}. \end{aligned} \tag{4.10}$$

L'équilibre de Nash étant un équilibre, on peut se demander si la solution donnée par cet équilibre est celle qui maximise les gains de tous les agents. Dans l'exemple du dilemme du prisonnier, l'équilibre de Nash consiste pour les deux agent à dénoncer et donc à subir chacun une peine de prison de deux ans. Or, en suivant la stratégie jointe où chaque agent nie, ils n'écopent chacun que d'une année de prison ! Cette stratégie est donc meilleure. Pour cela, les concepts de **Pareto dominance** et d'**optimum de Pareto** sont utiles.

Définition 4.20 *Une stratégie jointe* $\hat{\boldsymbol{\pi}}$ domine au sens de Pareto *une autre stratégie jointe* $\boldsymbol{\pi}$ *si et seulement si :*

- *chaque agent* i *suivant* $\hat{\pi}_i$ *reçoit au moins le même gain espéré qu'en suivant* π_i *et,*
- *au moins un agent* j *suivant* $\hat{\pi}_j$ *reçoit un gain espéré strictement supérieur à ce qu'il recevrait s'il suivait* π_j*,*

c'est-à-dire, formellement :

$$\hat{\boldsymbol{\pi}} > \boldsymbol{\pi} \Leftrightarrow \forall i \quad \mathsf{u}_{i,\hat{\boldsymbol{\pi}}} \geq \mathsf{u}_{i,\boldsymbol{\pi}} \quad et \quad \exists j \quad \mathsf{u}_{j,\hat{\boldsymbol{\pi}}} > \mathsf{u}_{j,\boldsymbol{\pi}}. \tag{4.11}$$

Définition 4.21 *Si une stratégie jointe* $\hat{\boldsymbol{\pi}}^*$ *n'est dominée au sens de Pareto par aucune autre stratégie, alors* $\hat{\boldsymbol{\pi}}^*$ *est* Pareto optimal.

Une stratégie jointe est Pareto optimale si aucun agent ne peut améliorer son gain espéré sans qu'au moins un autre agent ne voit alors son gain espéré diminuer. Il faut retenir qu'un équilibre de Nash n'est pas nécessairement un optimum de Pareto. Ainsi, dans le jeu du dilemme du prisonnier, la stratégie jointe $\langle \text{nier}, \text{nier}\rangle$ est Pareto optimale mais n'est pas un équilibre de Nash. Par contre, dans le cas où un jeu matriciel comporte de multiples équilibres de Nash, un équilibre peut en dominer un autre au sens de Pareto.

4.4.2 Jeux de Markov

Les **jeux de Markov**[17] modélisent les problèmes de décision de plusieurs agents dans un environnement à plusieurs états. Ce sont donc une extension des jeux matriciels à des environnements à plusieurs états ou une extension des PDM au cadre multiagent.

Définition 4.22 *Un* jeu de Markov *[Sha53] est défini comme un n-uplet* $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R}_1, ..., \mathsf{R}_m \rangle$ *où :*

- m *est le nombre d'agents,*
- $\mathcal{S}$ *est un ensemble fini d'états,*
- $\mathcal{A}_i$ *est l'ensemble des actions pour l'agent* i *(et* $\mathcal{A}_1 \times ... \times \mathcal{A}_m \equiv \boldsymbol{\mathcal{A}}$ *est l'ensemble des actions jointes),*
- $\mathsf{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \mapsto [0; 1]$ *est une fonction de transition définissant une distribution de probabilité sur les états futurs,*
- $\mathsf{R}_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ *est la fonction de récompense pour l'agent* i*.*

Dans un jeu de Markov, les caractéristiques principales sont que chaque agent a une connaissance complète de l'état s (observabilité individuelle totale) et la fonction de récompense est propre à chaque agent. A chaque tour de jeu, étant donné l'état courant s auquel tous les agents ont accès, ceux-ci choisissent leurs actions simultanément. Chaque agent obtient alors une récompense $\mathsf{R}_i(s, \boldsymbol{a})$ selon l'action jointe $\boldsymbol{a}$. Le système passe alors dans un nouvel état s' en suivant la transition $\mathsf{T}(s, \boldsymbol{a}, s')$. Ainsi, dans chaque état du jeu de Markov, on retrouve un jeu matriciel définissant les récompenses de chaque agent selon l'action jointe (*cf.* figure 4.11).

Jeux de Markov d'équipe

La même classification est utilisée pour les jeux matriciels et pour les jeux de Markov selon les fonctions de récompense. Ainsi, si tous les agents ont la même fonction de récompense, le jeu de Markov est dit **jeu d'équipe**.

Différentes dénominations sont utilisées pour les formalismes des jeux de Markov d'équipe où les agents ont chacun une observabilité individuelle totale. Ils sont appelés **jeux stochastiques à récompenses identiques**[18] (IPSG) par [PKMK00]. Craig Boutilier propose quant à lui le terme de **processus décisionnels de Markov multiagents**[19] (MMDP) [Bou96, Bou99]. Un MMDP est définit par un n-uplet $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R} \rangle$ où m, $\mathcal{S}$, $\boldsymbol{\mathcal{A}}$ et T sont définis comme pour un jeu de Markov. Les MMDP modélisent uniquement des jeux d'équipe donc la fonction de récompense est globale ; elle est « partagée » par tous les agents et est définie par une fonction unique $\mathsf{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$. Ces modèles sont donc une spécialisation des jeux de Markov dans laquelle la fonction de récompense est la même pour tous les agents.

17. Ou *jeux stochastiques.*
18. En anglais *identical payoff stochastic games.*
19. En anglais *Multi-agent Markov Decision Processes.*

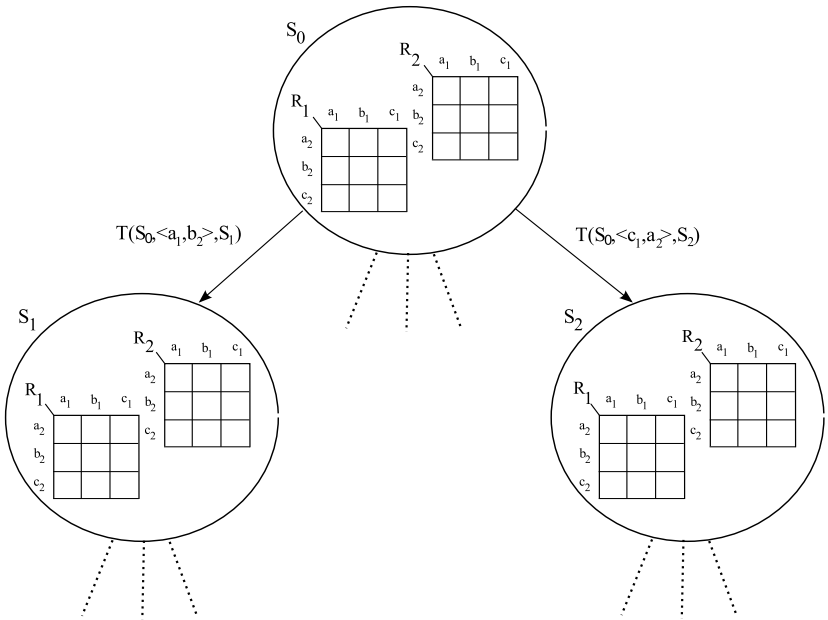


FIGURE 4.11 – Jeu de Markov à deux agents et trois actions chacun où chaque état peut être vu comme un jeu matriciel définissant les récompenses reçues par chaque agent selon l'état global et l'action jointe effectuée. Les transitions entre états sont aussi fonction de l'action jointe.

Politique

Définition 4.23 *Une politique pour un agent i dans un jeu de Markov est une fonction $\pi_i : \mathcal{S} \times \mathcal{A}_i \mapsto [0;1]$ qui définit une distribution de probabilité sur les états et actions de l'agent,* i.e. *:*

$$\forall a_i \in \mathcal{A}_i \quad P(a_i|x) = \pi_i(x, a_i). \tag{4.12}$$

On a la contrainte suivante :

$$\forall s \in \mathcal{S} \quad \sum_{a_i \in \mathcal{A}_i} \pi_i(s, a_i) = 1. \tag{4.13}$$

Comme pour les jeux matriciels, on note $\boldsymbol{\pi}_{-i}$ la politique jointe des agents exceptée celle de l'agent i. $\boldsymbol{\pi} = \langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$ désigne la politique jointe de tous les agents où l'agent i suit π_i et les autres suivent $\boldsymbol{\pi}_{-i}$. Nous noterons aussi $\Delta(\mathcal{S}, \mathcal{A}_i)$ l'ensemble de toutes les distributions de probabilités qui recouvre l'ensemble $\mathcal{S} \times \mathcal{A}_i$, *i.e.* l'ensemble de toutes les politiques pour l'agent i.

Les définitions 4.16 et 4.17 de politique déterministe et mixte sont identiques dans les jeux de Markov.

Équilibres

Dans un jeu de Markov, le gain immédiat espéré de l'agent i suivant π est défini pour chaque état s de la même manière que dans les jeux matriciels, *i.e.* :

$$\begin{aligned} \mathsf{u}_{i,\boldsymbol{\pi}}(s) &= E_{\boldsymbol{\pi}}\left\{\mathsf{R}_i(s,\boldsymbol{a})\right\} \\ &= \sum_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \boldsymbol{\pi}(s,\boldsymbol{a})\mathsf{R}_i(s,\boldsymbol{a}) \end{aligned} \tag{4.14}$$

Dans ce cas, l'espérance de gain à long terme (et pondérée) pour l'agent i à partir de l'état s et si tous les agents suivent la politique $\boldsymbol{\pi}$ est :

$$\mathsf{U}_{i,\boldsymbol{\pi}}(s) = E_{\boldsymbol{\pi}}\left\{\sum_{k=0}^{\infty}\gamma^k \mathsf{u}_{i,\boldsymbol{\pi}}(s)_{t+k+1}|s_t = s\right\}. \tag{4.15}$$

Comme pour les PDM, $\gamma \in [0;1[$ est un coefficient d'atténuation.

De même que dans les jeux matriciels, les concepts d'équilibre de Nash et de Pareto optimalité sont définis dans un jeu de Markov.

Définition 4.24 *Une politique jointe $\boldsymbol{\pi}^*$ définit un équilibre de Nash dans un jeu de Markov si pour chaque agent i on a :*

$$\forall \pi_i \in \Delta(\mathcal{S},\mathcal{A}_i)\ \forall s\in\mathcal{S} \quad \mathsf{U}_{i,\langle \pi_i^*,\boldsymbol{\pi}_{-i}^*\rangle}(s) \geq \mathsf{U}_{i,\langle \pi_i,\boldsymbol{\pi}_{-i}^*\rangle}(s). \tag{4.16}$$

Définition 4.25 *Une politique jointe $\hat{\boldsymbol{\pi}}$* domine au sens de Pareto *une autre politique jointe $\boldsymbol{\pi}$ si et seulement si, dans tous les états :*

- *chaque agent i suivant $\hat{\pi}_i$ a au moins la même espérance de gain qu'en suivant π_i et,*
- *au moins un agent j suivant $\hat{\pi}_j$ a une espérance de gain strictement supérieure à ce qu'elle serait s'il suivait π_j,*

c'est-à-dire, formellement :

$$\hat{\boldsymbol{\pi}} > \boldsymbol{\pi} \Leftrightarrow \forall i,\ \forall s\in\mathcal{S} \quad \mathsf{U}_{i,\hat{\boldsymbol{\pi}}}(s) \geq \mathsf{U}_{i,\boldsymbol{\pi}}(s) \quad et \quad \exists j \quad \mathsf{U}_{j,\hat{\boldsymbol{\pi}}}(s) > \mathsf{U}_{j,\boldsymbol{\pi}}(s). \tag{4.17}$$

Définition 4.26 *Si une politique jointe $\hat{\boldsymbol{\pi}}^*$ n'est dominée au sens de Pareto par aucune autre politique, alors $\hat{\boldsymbol{\pi}}^*$ est* Pareto optimal.

4.4.3 Objectifs

Dans les jeux matriciels comme dans les jeux de Markov, les concepts d'équilibre de Nash et de Pareto optimalité sont utiles à la résolution. Mais les objectifs atteints par ces équilibres ne sont pas les mêmes. Il est donc nécessaire de préciser quel équilibre répond au mieux à nos objectifs.

Résoudre un problème de décision dans un système multiagent consiste à calculer une **politique jointe** $\boldsymbol{\pi} =< \pi_1, ..., \pi_m >$ où π_i correspond à la politique de l'agent i et m est le nombre d'agents. Comme dans un processus décisionnel de Markov, chaque agent cherche une politique individuelle qui lui permet de maximiser un critère de performance. Ce critère est le plus souvent la somme pondérée de ses récompenses futures. Les jeux de Markov à somme générale ont des fonctions de récompense qui peuvent être différentes pour chaque agent. Dans certains cas, il peut alors être impossible de trouver des politiques qui maximisent ce critère pour tous les agents. C'est pourquoi *dans les jeux de Markov à somme générale, un point d'équilibre est recherché.* Cet équilibre est une situation dans laquelle aucun agent ne pourra améliorer son critère de performance s'il est le seul à changer sa politique. On retrouve ici la définition d'un équilibre de Nash ainsi que la propriété de « stabilité » qu'il satisfait : chaque agent joue la meilleure réponse aux stratégies des autres. C'est pourquoi l'équilibre de Nash est fréquemment utilisé dans les jeux de Markov à somme générale comme équilibre de meilleure réponse mutuelle. L'équilibre de Nash permet notamment de répondre au mieux aux **objectifs individuels** des agents.

Néanmoins, certains s'interrogent sur la nécessité de converger vers un équilibre de Nash [SPG04, CG05]. En effet, cet équilibre n'est pas toujours unique et la multiplicité des équilibres de Nash requiert alors des processus de coordination. Pour s'affranchir de ce problème, de nombreux travaux dans les jeux de Markov supposent l'unicité de l'équilibre de Nash [Lit01a, Lit01b, HW03]. Un autre inconvénient de cet équilibre est qu'il ne correspond pas toujours à la meilleure politique du point de vue de maximiser les gains de tous les agents. Par exemple, dans le dilemme du prisonnier (*cf.* figure 4.8), l'équilibre de Nash fait subir aux agents deux ans de prison alors que l'équilibre Pareto optimal ne leur fait subir qu'un an. Ainsi, *l'équilibre de Nash peut conduire à des solutions sous-optimales dans les jeux de Markov à somme générale.* Certains travaux utilisent donc d'autres équilibres pour atteindre une solution : l'équilibre de Stackelberg [Kön03, LC05] ou l'équilibre corrélé [GH03] par exemple. L'objectif à atteindre dans les jeux de Markov à somme générale est donc difficile à définir.

Par contre, dans les **jeux de Markov d'équipe**, tous les agents reçoivent la même fonction de récompense. Ainsi, d'une part, les équilibres Pareto optimaux sont aussi des équilibres de Nash. D'autre part, les politiques qui définissent ces équilibres maximisent la somme pondérée des récompenses futures pour tous les agents. En effet, si une politique procure à un agent la somme pondérée maximale des récompenses futures, elle maximise aussi dans un jeu d'équipe la somme pondérée des autres agents. *Dans un jeu de Markov (ou matriciel) d'équipe, nous définissons donc l'objectif comme atteindre un équilibre de Nash Pareto optimal ; dans ce cas, les agents maximisent la somme pondérée de leurs récompenses futures.* Toutefois, ces équilibres de Nash Pareto optimaux peuvent être multiples donc des mécanismes de coordination sur un unique équilibre sont nécessaires.

4.4.4 Conclusion

Dans cette section a été présenté le formalisme des jeux de Markov qui constitue le cadre dans lequel se placent nos travaux. Les jeux matriciels peuvent être vus comme une « réduction » des jeux de Markov à un état. Nous les utiliserons principalement en première étude car les situations d'interaction qui y prennent place sont souvent plus faciles à repérer et à interpréter. Les différents concepts utiles à la résolution de ces jeux ont été détaillés, et en particulier deux équilibres principaux : l'équilibre de Nash qui maximise le plus souvent un critère individuel et l'équilibre Pareto optimal qui répond quant à lui à l'intérêt collectif. Les objectifs ont enfin été précisés. Notamment, dans les jeux d'équipe, les équilibres Pareto optimaux sont aussi des équilibres de Nash et maximisent le critère de performance individuel et global.

4.5 Liens entre concepts et formalismes

Dans cette section est proposée une synthèse présentant les liens entre les concepts introduits lors de l'étude des SMA et les formalismes des processus décisionnels de Markov multiagents. Ces formalismes recouvrent les notions suivantes :

- une connaissance complète ou partielle de l'état du système par chaque agent,
- une fonction de récompense définie globalement ou propre à chaque agent.
- la possibilité ou non de communiquer entre agents.

4.5.1 Perceptions de l'état

Le premier concept introduit est celui d'observabilité totale ou partielle. Le formalisme des jeux de Markov suppose que chaque agent a accès à l'état global du système, c'est-à-dire que l'état du système est individuellement totalement observable. Cette hypothèse correspond au cadre de notre étude mais n'est pas toujours vérifiée dans les SMA (*cf.* §4.3.1). Il faut alors utiliser des modèles avec observabilités partielles. Par souci d'exhaustivité, nous présentons donc ici des processus décisionnels de Markov multiagents pour des agents avec des **observabilités partielles**. Cette présentation reste succincte car nous n'utiliserons pas ces formalismes dans la suite de ce mémoire.

Dans le cadre mono-agent, le formalisme des **processus décisionnel de Markov partiellement observable** (POMDP) [SS73, KLC98] modélise des PDM mono-agent dans lesquels l'agent n'a accès qu'à des observations partielles. Son extension dans le cadre des SMA a donné lieu aux PDM décentralisés et aux **jeux stochastiques partiellement observables** (POSG). Un POSG [HBZ04] est une extension des jeux de Markov dans laquelle les agents ont des observabilités partielles et la fonction de récompense est propre à chaque agent. Un PDM décentralisé est quant à lui une extension des MMDP aux cas d'observabilités partielles.

Définition 4.27 *Un Processus Décisionnel de Markov décentralisé partiellement observable (DEC-POMDP) [BZI00, BGIZ02] est défini par un n-uplet* $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \Omega_1, ..., \Omega_m, \mathcal{O}, \mathsf{R}\rangle$ *où :*

- *le n-uplet* $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R} \rangle$ *est un jeu de Markov d'équipe (MMDP),*
- Ω_i *est l'ensemble des observations* o_i *de l'agent* i *(et* $\Omega = \Omega_1 \times ... \times \Omega_m$ *est l'ensemble des observations des agents),*
- $\mathcal{O} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \times \Omega \mapsto [0; 1]$ *est une fonction d'observation définissant une distribution de probabilité sur les actions et observations jointes.* $\mathcal{O}(s, \mathbf{a}, s', o_i)$ *correspond à la probabilité que l'agent* i *observe* o_i *lorsque les agents exécutent l'action jointe* $\mathbf{a}$ *à partir de* s *et que le système arrive dans* s'*.*

La fonction de récompense est définie sur les actions jointes et est unique comme dans les MMDP. Si on a une observabilité collective totalement observable à partir des perceptions locales, le DEC-POMDP devient un **processus décisionnel de Markov décentralisé** (DEC-MDP). Il est à noter que cette propriété n'implique pas que chaque agent ait une observabilité locale totale. Dans le cas ou l'état est individuellement totalement observable, le DEC-MDP se ramène à un MMDP.

Des modèles très proches se retrouvent dans la littérature, par exemple le modèle de **jeu stochastique à récompenses identiques partiellement observables** (POIPSG) [PKMK00].

4.5.2 Coopération et distribution des récompenses

La récompense dans un SMA peut être distribuée localement ou globalement. Si tous les agents reçoivent la même récompense, on parle de jeux d'équipe. Les jeux de Markov (ou matriciels) d'équipe sont aussi souvent appelés jeux coopératifs [20]. Cependant, dans ce mémoire, nous ferons la distinction entre **jeux coopératifs** et **jeux d'équipe** en nous basant sur la définition du concept de coopération introduite en début de ce chapitre. Dans un jeu coopératif, les agents sont dans une situation de coopération, c'est-à-dire que leurs objectifs sont compatibles. Dans un jeu d'équipe, tous les agents ont la même fonction de récompense, donc une action jointe dans le meilleur intérêt d'un agent est aussi dans le meilleur intérêt de tous les agents. Un jeu d'équipe est donc un jeu coopératif car maximiser sa récompense revient à maximiser celle du groupe, *i.e.* que la satisfaction individuelle est aussi celle du groupe. Par contre, la réciproque n'est pas toujours vraie. En effet, un jeu coopératif n'est pas forcément un jeu d'équipe. Comme nous l'avons vu dans la section sur la distribution des récompenses, les agents peuvent être récompensés de manière locale : on détermine ceux qui sont responsables de l'avancement de la résolution et ils sont récompensés. Dans ce cas, tous les agents ne reçoivent pas la même récompense malgré que ce soit une situation de coopération.

4.5.3 Communication

La possibilité ou non de communiquer entre les agents n'est pas forcément introduite directement dans le formalisme. Ainsi, le choix d'agents indépendants ou d'agents percevant les actions jointes, et donc le choix de permettre à un agent de communiquer ses

20. En anglais *cooperative games.*

TABLE 4.1 – Différents formalismes.

		Distribution des récompenses	
		locale	globale
Perception de l'état	état global individuellement observable	Jeux de Markov	Jeux de Markov d'équipe MMDP IPSG
	observabilité collective totale	POSG	DEC-MDP POIPSG
	observabilité collective partielle	POSG	DEC-POMDP POIPSG

actions aux autres, est indépendant du formalisme choisi. Mais il existe aussi des formalismes qui, par exemple, étendent les DEC-POMDP afin de permettre la modélisation de la communication entre les agents [GZ03].

4.5.4 Synthèse

Les différents formalismes des processus décisionnels de Markov multiagents recouvrent les notions de perception de l'état et de distribution locale ou globale des récompenses. La table 4.1 récapitule ces formalismes qui permettent des prises de décisions individuelles ou collectives avec des observabilités totales ou partielles.

La **complexité** de la résolution optimale de ces différents problèmes varie selon le nombre d'agents dans le SMA et selon l'observabilité de l'état global du système. Pour plus d'informations, nous conseillons la lecture de la thèse d'Aurélie Beynier [Bey06] ainsi que des articles [PT87, BGIZ02].

4.6 Conclusion

Ce chapitre a été l'occasion d'introduire les notions et formalismes dont il sera fait usage dans la suite de ce mémoire. Nous avons notamment discuté :

- de la notion d'agent et de celle de système multiagent vu comme un ensemble d'agents autonomes qui interagissent entre eux. De ces interactions émerge un comportement global,
- des situations de coopération qui font appel à des mécanismes de résolution tels que la collaboration et la coordination.

Notre objectif est d'utiliser l'apprentissage par renforcement dans des modèles de processus décisionnels de Markov adaptés aux SMA. Le point de vue adopté est décentralisé. Les différents formalismes recouvrent les notions de perception de l'état et de

distribution de récompense. Une synthèse a notamment été proposée dans ce chapitre.

Dans le cadre de notre étude, nous nous intéressons à des agents qui ont accès uniquement à leur action individuelle et ont des observabilités individuelles totales. La distribution des récompenses est quant à elle collective. Les formalismes satisfaisant ces hypothèses ont été détaillés dans ce chapitre :

- **les jeux matriciels :** issu de la théorie des jeux, ce formalisme à plusieurs agents et sans états modélise les interactions entre agents. Les algorithmes d'apprentissage par renforcement pour SMA sont souvent évalués en premier lieu dans les jeux matriciels car les situations d'interaction qui compliquent l'apprentissage y sont plus facilement repérables et interprétables.
- **les jeux de Markov :** ce formalisme est une extension des jeux matriciels à des environnements à plusieurs états. Dans chaque état du jeu de Markov se retrouve un jeu matriciel qui définit les récompenses individuelles de chaque agent.

Ce chapitre a aussi été l'occasion de *préciser les solutions que nous allons mettre en oeuvre dans le cadre de l'apprentissage par renforcement dans les SMA ainsi que les objectifs*. Nous nous intéressons à des agents indépendants coopératifs, et plus spécifiquement aux jeux d'équipe. La principale difficulté posée par la coopération des agents est d'assurer leur *coordination*. Lors de l'apprentissage d'agents indépendants, certains facteurs responsables des difficultés de coordination peuvent être identifiés. Le chapitre suivant sera donc consacré à une étude de ces enjeux de coordination. Nous proposerons ensuite dans le chapitre 6 un état de l'art des méthodes par apprentissage par renforcement permettant à des agents d'apprendre indépendamment à se coordonner dans les **jeux de Markov d'équipe**.

Chapitre 5

Enjeux de l'apprentissage par renforcement pour des agents indépendants dans les jeux de Markov d'équipe

Ce chapitre propose une étude des problématiques liées à l'apprentissage par renforcement d'agents indépendants dans les jeux de Markov d'équipe. Après avoir précisé les objectifs du groupe et ceux de chaque agent lors de l'apprentissage, ce chapitre détaille certaines difficultés rencontrées par des agents telles que les facteurs de non-coordination ou l'impact de l'exploration.

5.1 Introduction

Le domaine de l'apprentissage par renforcement mono-agent est aujourd'hui mature dans le sens où les résultats théoriques sont bien compris et où de nombreuses techniques pratiques existent [SB98]. A l'inverse, le domaine de l'apprentissage par renforcement dans les systèmes multiagents (SMA) est beaucoup moins évolué. D'une part, c'est un domaine de recherches assez récent, comme le montrent les états de l'art des articles [SPG04, YG04, BBS06b]. D'autre part, comme nous l'avons vu dans le chapitre précédent, l'apprentissage dans les SMA est soumis à plusieurs difficultés, notamment dues aux interactions et à la présence d'autres agents qui doivent être prises en compte. Afin de mieux appréhender les techniques d'apprentissage dans les SMA, nous proposons donc tout d'abord d'*étudier certains des enjeux que des agents indépendants doivent surmonter dans les jeux de Markov d'équipe.*

Dans ce chapitre sont donc analysées trois difficultés majeures qui compliquent l'apprentissage par renforcement d'agents indépendants. Ces difficultés sont clairement iden-

tifiables dans des jeux où le nombre d'agents est faible. Dans le cas de problèmes plus larges, un enjeu tel que la coordination est très complexe à décomposer en différents facteurs clairement définis. Notre étude permet néanmoins *une meilleure compréhension des difficultés rencontrées lors de l'application d'algorithmes d'apprentissage par renforcement dans les jeux de Markov d'équipe*, ce qui sera l'objet du chapitre suivant. Une telle étude présente un intérêt majeur pour la compréhension de certains phénomènes et n'a à notre connaissance jamais été proposée dans le cadre d'agents indépendants.

Tout d'abord, les objectifs du groupe et ceux de chaque agent lors de l'apprentissage d'agents indépendants doivent être précisés. Pour cela, nous allons définir les **fonctions de valeur locales et globales**. Nous établirons de plus une condition nécessaire d'optimalité au niveau local issue d'une hypothèse optimiste. Une fois les objectifs posés, nous nous intéressons à certaines difficultés rencontrées lors de l'apprentissage par des agents indépendants. Une première difficulté vient d'une **propriété des processus locaux**. En effet, dans certains cas, la **propriété fondamentale de Markov** n'est plus nécessairement vérifiée au niveau local. Les garanties théoriques de convergence des algorithmes d'apprentissage par renforcement issus du cadre mono-agent sont alors remises en cause dans le cas multiagent décentralisé. Un autre enjeu auquel nous nous intéressons pour des agents indépendants dans un jeu de Markov d'équipe est d'assurer leur **coordination**. Nous nous focalisons tout d'abord non pas sur comment améliorer la coordination, mais plutôt sur certains **facteurs** qui peuvent compliquer la coordination d'agents indépendants. L'impact de l'**exploration** sera notamment discuté. Une fois ces facteurs isolés et compris, le développement et l'application de techniques de coordination dans les algorithmes d'apprentissage par renforcement seront facilités.

5.2 Objectifs

Nous allons tout d'abord définir précisément les objectifs du groupe et ceux de chaque agent dans le cadre des jeux de Markov d'équipe. Pour cela, nous devons définir la notion de processus décisionnel local ainsi que celles de fonctions de valeur globales et locales. Ensuite, grâce à une hypothèse qui peut être posée dans le cadre coopératif, nous établissons une condition nécessaire d'optimalité au niveau local pour des agents indépendants.

5.2.1 Processus local

Nous allons tout d'abord définir le **processus décisionnel local** à un agent dans le cas d'un groupe d'agents déterministes.

Définition 5.1 *Soit* $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R} \rangle$ *un jeu de Markov d'équipe à* m *agents et* $\pi_i : \mathcal{S} \times \mathcal{A}_i \mapsto [0;1]$ *les politiques* déterministes *des agents. On définit le processus*

décisionnel local à l'agent i par le tuple $\langle \mathcal{S}, \mathcal{A}_i, \mathsf{T}_i, \mathsf{R}_i \rangle$ *où pour tout* $(s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$ *:*

$$\mathsf{T}_i(s, a, s') = \mathsf{T}(s, \langle a_1, ..., a_m \rangle, s') \text{ avec } a_j = \begin{cases} a \text{ si } j = i \\ \arg\max_b \pi_j(s, b) \text{ sinon} \end{cases} \tag{5.1}$$

et :

$$\mathsf{R}_i(s, a) = \mathsf{R}(s, \langle a_1, ..., a_m \rangle) \text{ avec } a_j = \begin{cases} a \text{ si } j = i \\ \arg\max_b \pi_j(s, b) \text{ sinon} \end{cases} \tag{5.2}$$

Dans le **cas général**, c'est-à-dire lorsque les politiques des agents peuvent être stochastiques, on a la définition suivante :

Définition 5.2 *Soit* $\langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R} \rangle$ *un jeu de Markov d'équipe à m agents et* $\pi_i : \mathcal{S} \times \mathcal{A}_i \mapsto [0; 1]$ *les politiques des agents. On définit le processus décisionnel local à l'agent i par le tuple* $\langle \mathcal{S}, \mathcal{A}_i, \mathsf{T}_i, \mathsf{R}_i \rangle$ *où pour tout* $(s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$ *:*

$$\mathsf{T}_i(s, a, s') = \sum_{\mathbf{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \boldsymbol{\pi}(s, \langle a, \mathbf{a}_{-i} \rangle) \mathsf{T}(s, \langle a, \mathbf{a}_{-i} \rangle, s') \tag{5.3}$$

et :

$$\mathsf{R}_i(s, a) = \sum_{\mathbf{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \boldsymbol{\pi}(s, \langle a, \mathbf{a}_{-i} \rangle) \mathsf{R}(s, \langle a, \mathbf{a}_{-i} \rangle) \tag{5.4}$$

En effet, d'après le théorème des probabilités totales, on a pour tout $(s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$:

$$\begin{aligned} & P_r(s_{k+1} = s' | s_k = s, a_k = a) \\ & \quad = \sum_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}} P_r(s_{k+1} = s' | s_k = s, a_k = a, \boldsymbol{a}_{k,-i} = \boldsymbol{a}_{-i}) P_r(\boldsymbol{a}_{k,-i} = \boldsymbol{a}_{-i} | s_k = s, a_k = a) \\ & \quad = \sum_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{T}(s, \langle a, \boldsymbol{a}_{-i} \rangle, s') \boldsymbol{\pi}(s, \langle a, \boldsymbol{a}_{-i} \rangle) \end{aligned} \tag{5.5}$$

5.2.2 Fonctions de valeur globales et locales

Dans ce paragraphe, les fonctions de valeur locales et globales dans un jeu de Markov d'équipe sont définies. Ces fonctions de valeur permettent notamment de préciser les objectifs de l'apprentissage.

Fonction de valeur globale

La **fonction de valeur globale** sous une politique jointe $\boldsymbol{\pi}$ est notée $\mathsf{Q}^{\boldsymbol{\pi}}$ et vérifie :

$$\forall (s, \boldsymbol{a}) \in \mathcal{S} \times \boldsymbol{\mathcal{A}} \quad \mathsf{Q}^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = E_{\boldsymbol{\pi}} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \, | s_t = s, \boldsymbol{a}_t = \boldsymbol{a} \right\}. \tag{5.6}$$

On note alors Q^* la fonction de valeur globale optimale d'une politique jointe optimale définie par :

$$\forall (s, \boldsymbol{a}) \in \mathcal{S} \times \boldsymbol{\mathcal{A}} \quad \mathsf{Q}^*(s, \boldsymbol{a}) = \max_{\boldsymbol{\pi}} \mathsf{Q}^{\boldsymbol{\pi}}(s, \boldsymbol{a}). \tag{5.7}$$

Les actions jointes optimales $\boldsymbol{a}^*$ qui maximisent la somme pondérée sur le long terme des récompenses (*cf.* équation 5.6) peuvent alors être construites à partir des actions gloutonnes sur la table globale Q^* (*cf.* définition 2.3) de la manière suivante :

$$\forall s \in \mathcal{S} \quad \boldsymbol{a}^* \in \arg\max_{\boldsymbol{u} \in \boldsymbol{\mathcal{A}}} \mathsf{Q}^*(s, \boldsymbol{u}). \tag{5.8}$$

Ces actions $\boldsymbol{a}^*$ vérifient donc aussi l'égalité suivante :

$$\forall s \in \mathcal{S} \quad \mathsf{Q}^*(s, \boldsymbol{a}^*) = \max_{\boldsymbol{u} \in \boldsymbol{\mathcal{A}}} \mathsf{Q}^*(s, \boldsymbol{u}). \tag{5.9}$$

L'objectif est donc pour l'ensemble des agents de trouver une action jointe qui vérifie cette équation sur la fonction de valeur globale.

Fonction de valeur locale

Soit i un agent indépendant dans un jeu de Markov d'équipe $\mathcal{M}$. Chaque agent i a donc accès à l'ensemble des états $\mathcal{S}$ et à l'ensemble de ses actions individuelles $\mathcal{A}_i$. La **fonction de valeur locale** d'un agent i sous une politique jointe $\boldsymbol{\pi}$ est notée $\mathsf{Q}_i^{\boldsymbol{\pi}}$ et vérifie :

$$\forall (s, a_i) \in \mathcal{S} \times \mathcal{A}_i \quad \mathsf{Q}_i^{\boldsymbol{\pi}}(s, a_i) = E_{\boldsymbol{\pi}} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s, a_{i,t} = a_i \right\}. \tag{5.10}$$

L'objectif d'un agent est de trouver une action individuelle a_i qui permet de maximiser la somme pondérée de ses récompenses futures.

5.2.3 Hypothèse optimiste

Dans le cadre coopératif, l'**hypothèse optimiste** peut être posée. Elle consiste pour chaque agent à supposer que les autres agissent de manière optimale. Avec cette hypothèse, chaque agent i peut alors choisir, pour évaluer chaque couple état-action individuelle, la valeur maximale de la fonction de valeur globale, *i.e.* :

$$\begin{aligned} \forall (s, a_i) \in \mathcal{S} \times \mathcal{A}_i \quad \mathsf{Q}_i^*(s, a_i) &= E_{\boldsymbol{\pi}^*} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_{i,t} = a_i, \boldsymbol{a}_{-i,t} = \boldsymbol{a}_{-i}^* \right\} \\ &= \mathsf{Q}^*(s, \langle a_i, \boldsymbol{a}_{-i}^* \rangle) \\ &= \max_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{Q}^*(s, \langle a_i, \boldsymbol{a}_{-i} \rangle) \end{aligned} \tag{5.11}$$

Pour illustrer cette hypothèse optimiste, prenons la fonction de valeur globale optimale tracée à la figure 5.1. Cette courbe représente les évaluations des actions jointes de

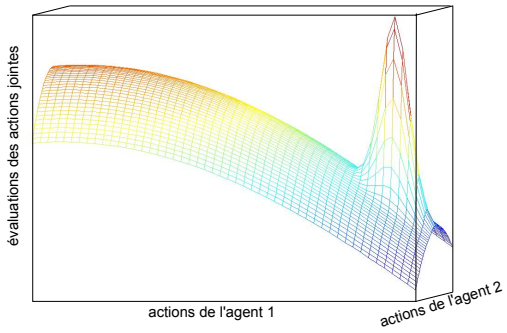
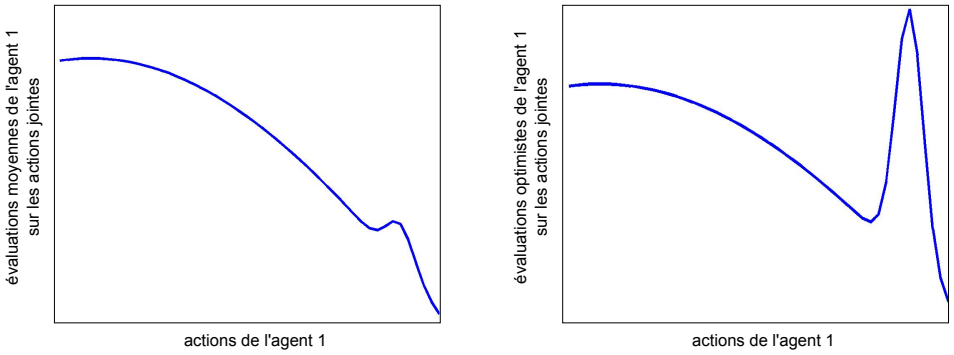


FIGURE 5.1 – Évaluations des actions jointes de deux agents. Cette courbe présente deux pics de différentes tailles. Le pic le plus haut est un équilibre de Nash Pareto optimal. Le diamètre de ce pic est très faible, donc si un des deux agents dévie de cet équilibre, les évaluations diminuent très rapidement. Le second pic est moins haut mais de diamètre plus large. C'est un équilibre de Nash sous-optimal.



(a) moyenne (b) optimiste

FIGURE 5.2 – Différentes évaluations des actions individuelles par des agents indépendants dans le cas d'actions jointes évaluées à la figure 5.1.

deux agents dans un état. Sont présents dans ce jeu un équilibre de Nash Pareto optimal et un équilibre sous-optimal. La courbe de la figure 5.2a montre l'évaluation moyenne des actions individuelles de l'agent 1 pour différentes actions choisies par l'autre agent, c'est-à-dire :

$$\mathsf{Q}_i(s, a_i) = \frac{\sum\limits_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{Q}(s, \langle a_i, \boldsymbol{a}_{-i} \rangle)}{|\boldsymbol{\mathcal{A}}_{-i}|}. \tag{5.12}$$

Ces évaluations sont faites par chaque agent sans poser aucune hypothèse sur le comportement de l'autre agent. Dans ce cas, le pic correspondant à l'équilibre de Nash Pareto optimal diminue fortement en amplitude. L'action gloutonne correspond alors à l'équilibre sous-optimal.

Par contre, si l'agent 1 est optimiste et donc suppose que l'autre agent agit de manière optimale, il évalue ses actions individuelles selon l'équation 5.11. Ces évaluations optimistes sont tracées à la figure 5.2b. Sous cette hypothèse, l'action individuelle gloutonne sur la fonction de valeur locale correspond bien à l'équilibre de Nash Pareto optimal.

Nous allons maintenant montrer que cette hypothèse optimiste permet d'établir une condition nécessaire d'optimalité au niveau local.

5.2.4 Condition nécessaire d'optimalité

Si la fonction de valeur globale Q^* est optimale, alors, Q^* vérifie l'équation d'optimalité de Bellman suivante :

$$\mathsf{Q}^*(s, \boldsymbol{a}) = \sum_{s' \in \mathcal{S}} \mathsf{T}(s, \boldsymbol{a}, s') \left[\mathsf{R}(s, \boldsymbol{a}, s') + \gamma \max_{b \in \boldsymbol{\mathcal{A}}} \mathsf{Q}^*(s', b) \right]. \tag{5.13}$$

D'après l'équation 5.9, si $\boldsymbol{a}$ est une action jointe optimale, on peut alors écrire :

$$\begin{aligned} \max_{b \in \boldsymbol{\mathcal{A}}} \mathsf{Q}^*(s, b) &= \sum_{s' \in \mathcal{S}} \mathsf{T}(s, \boldsymbol{a}, s') \left[\mathsf{R}(s, \boldsymbol{a}, s') + \gamma \max_{b \in \boldsymbol{\mathcal{A}}} \mathsf{Q}^*(s', b) \right] \\ \max_{u_i \in \mathcal{A}_i} \max_{\boldsymbol{u}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{Q}^*(s, \langle u_i, \boldsymbol{u}_{-i} \rangle) &= \sum_{s' \in \mathcal{S}} \mathsf{T}(s, \boldsymbol{a}, s') \left[\mathsf{R}(s, \boldsymbol{a}, s') + \gamma \max_{u_i \in \mathcal{A}_i} \max_{\boldsymbol{u}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{Q}^*(s', \langle u_i, \boldsymbol{u}_{-i} \rangle) \right]. \end{aligned} \tag{5.14}$$

On suppose que l'hypothèse optimiste est vérifiée. On peut donc insérer dans l'équation précédente l'équation caractéristique de l'hypothèse optimiste (*cf.* équation 5.11) et le fait que les autres agents agissent de manière optimale :

$$\max_{u_i \in \mathcal{A}_i} \mathsf{Q}_i^*(s, u_i) = \sum_{s' \in \mathcal{S}} \mathsf{T}(s, \langle a_i, \boldsymbol{a}_{-i}^* \rangle, s') \left[\mathsf{R}(s, \langle a_i, \boldsymbol{a}_{-i}^* \rangle, s') + \gamma \max_{u_i \in \mathcal{A}_i} \mathsf{Q}_i^*(s', u_i) \right]. \tag{5.15}$$

Soit le processus décisionnel local à l'agent i défini par le tuple $\langle \mathcal{S}, \mathcal{A}_i, \mathsf{T}_i, \mathsf{R}_i \rangle$, on a alors pour tout $(s, a_i, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$ (*cf.* définition 5.2) :

$$\begin{aligned} \mathsf{T}_i(s, a_i, s') &= \sum_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{T}(s, \langle a_i, \boldsymbol{a}_{-i} \rangle, s') \boldsymbol{\pi}(s, \langle a_i, \boldsymbol{a}_{-i} \rangle) \\ &= \mathsf{T}(s, \langle a_i, \boldsymbol{a}^*_{-i} \rangle, s') \end{aligned} \tag{5.16}$$

car les autres agissent toujours de manière optimale d'après l'hypothèse optimiste. On fait de même pour la fonction de récompense et on aboutit alors à :

$$\mathsf{Q}^*_i(s, a_i) = \sum_{s' \in \mathcal{S}} \mathsf{T}_i(s, a_i, s') \left[\mathsf{R}_i(s, a_i, s') + \gamma \max_{u_i \in \mathcal{A}_i} \mathsf{Q}^*_i(s', u_i) \right] \tag{5.17}$$

où a_i est une action gloutonne sur la fonction locale Q^*_i. Ainsi, Q^*_i *est la solution optimale pour le processus décisionnel local à l'agent* i.

Si l'hypothèse optimiste est posée, il est alors nécessaire que chaque agent i trouve sa fonction de valeur locale optimale Q^*_i pour obtenir des actions jointes optimales. En effet, si chaque agent i choisit une action individuelle gloutonne a^*_i selon sa fonction de valeur locale suivant :

$$\forall s \in \mathcal{S} \quad a^*_i \in \arg\max_{u \in \mathcal{A}_i} \mathsf{Q}^*_i(s, u) \tag{5.18}$$

alors, a^*_i appartient à une action jointe optimale sur Q^*. Cependant, l'action jointe résultante n'est pas forcément optimale car si plusieurs agents ont le choix parmi différentes actions individuelles optimales, la combinaison de toutes ces actions individuelles n'est pas forcément optimale. Ce problème de sélection d'équilibres est évoqué au §5.4.2.

5.3 Propriété d'un processus local

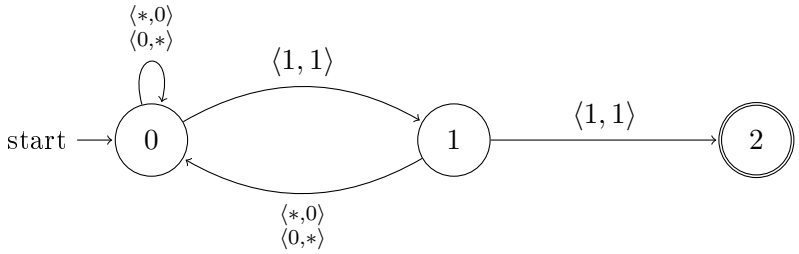
Une difficulté à laquelle des agents indépendants apprenants sont confrontés dans un jeu de Markov concerne la remise en cause **au niveau local** d'une propriété fondamentale des processus décisionnels de Markov (PDM). Le cadre mono-agent classique des PDM vérifie la *propriété fondamentale de Markov* (*cf.* §2.3.2), qui est une condition nécessaire de convergence vers des optima globaux des algorithmes d'apprentissage par renforcement mono-agent. Cette propriété dit simplement que la probabilité d'atteindre un nouvel état s_{k+1} suite à l'exécution d'une action a_k à l'instant k ne dépend que de l'état précédent s_k et de l'action effectuée a_k. Lors de l'extension du cadre des PDM à un formalisme multiagent, cette propriété n'est alors plus nécessairement vérifiée au niveau local.

Un processus décisionnel local a la propriété suivante :

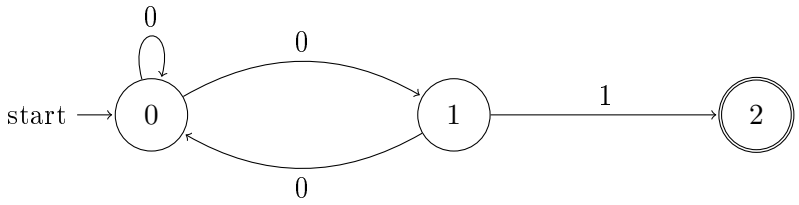
Propriété 5.1 *Un processus local peut ne pas être markovien si les autres agents sont « apprenants ».*

Ainsi, dans un jeu de Markov d'équipe où les agents indépendants apprennent simultanément, du point de vue d'un agent, le processus local peut ne pas vérifier la propriété de Markov. Nous montrons cela à l'aide du contre exemple qui suit.

Preuve 5.1 Soit le jeu de Markov à 2 agents défini par :
- 2 actions possibles par agent : $\boldsymbol{\mathcal{A}} = \{0,1\} \times \{0,1\}$
- 3 états : $\mathcal{S} = \{0,1,2\}$
- la fonction de transition déterministe T définie par le diagramme suivant



- la fonction de récompense commune R définie par le diagramme suivant



On suppose que l'agent 2 suit l'algorithme du Q-learning (*cf.* algorithme 2). Nous allons montrer que dans ces conditions, le processus local à l'agent 1 n'est pas markovien.

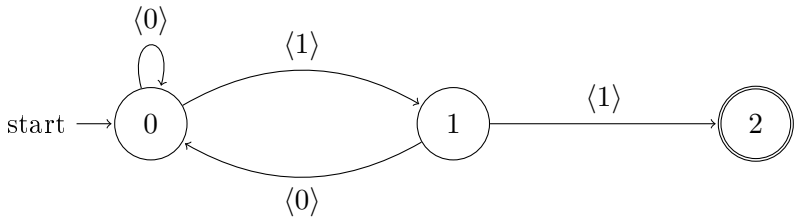
L'agent 2 construit une fonction de valeur notée Q_2. Il suit une politique gloutonne par rapport à Q_2. On utilisera $\gamma = 0.9$ et $\alpha = 0.6$ pour la mise à jour de Q_2.

Du point de vue de l'agent 1, le processus local est défini par (*cf.* définition 5.1) :
- 3 états : $\mathcal{S} = \{0,1,2\}$
- 2 actions : $\mathcal{A}_1 = \{0,1\}$
- une fonction de transition déterministe $\mathsf{T}_1(s,a,s') = \mathsf{T}(s, \langle a, \arg\max_b \mathsf{Q}_2(s,b)\rangle, s')$
- une fonction de récompense $\mathsf{R}_1(s,a) = \mathsf{R}(s, \langle a, \arg\max_b \mathsf{Q}_2(s,b)\rangle)$

A l'instant k, on suppose que les valeurs de la fonction Q_2 de l'agent 2 sont optimales, *i.e.* :

$$\begin{aligned}
\mathsf{Q}_{2,k}(0,1) &= 0.9 \\
\mathsf{Q}_{2,k}(0,0) &= 0.81 \\
\mathsf{Q}_{2,k}(1,1) &= 1 \\
\mathsf{Q}_{2,k}(1,0) &= 0.81
\end{aligned}$$

On suppose aussi que l'on se trouve dans l'état 0 à l'instant k. Le processus du point de vue de l'agent 1 à l'instant k est donc :



Soit, par exemple :

$$\mathsf{T}_{1,k}(0,1,1) = \mathsf{T}(0,\langle 1,1\rangle,1) = 1 \tag{5.19}$$

Scénario 1 A l'instant k, l'agent 1 effectue l'action 0, puis à nouveau l'action 0 à l'instant $k+1$. Le déroulement est le suivant :

- instant k : on se trouve dans l'état 0, l'agent 1 effectue l'action 0, l'agent 2 effectue $\arg\max_b \mathsf{Q}_{2,k}(0,b) = 1$.
- instant $k+1$: on se trouve dans l'état 0, l'agent 2 met sa fonction de valeur à jour :

$$\mathsf{Q}_{2,k+1}(0,1) = (1-0.6)*0.9 + 0.6*(0+0.9*0.9) = 0.846. \tag{5.20}$$

L'agent 1 effectue l'action 0, l'agent 2 effectue $\arg\max_b Q_{2,k}(0,b) = 1$

- instant $k+2$: on se trouve dans l'état 0, l'agent 2 met sa fonction de valeur à jour :

$$Q_{2,k+2}(0,1) = (1-0.6)*0.846 + 0.6*(0+0.9*0.846) = 0.79524 \tag{5.21}$$

La probabilité de transition du point de vue de l'agent 1 pour l'état 0 et l'action 1 est alors :

$$P_r(s_{k+3}=1|s_{k+2}=0,a_{k+2}=1) = \mathsf{T}_{1,k+2}(0,1,1) = \mathsf{T}(0,\langle 1,0\rangle,1) = 0 \tag{5.22}$$

On a donc :

$$P_r(s_{k+3}=1|s_{k+2}=0,a_{k+2}=1) \neq P_r(s_{k+1}=1|s_k=0,a_k=1) \tag{5.23}$$

Les probabilités de transitions du processus local ont donc changé. Ceci n'est pas une simple non stationnarité de la fonction de transition T_1. En effet, si les actions de l'agent 1 sont différentes, T_1 peut rester inchangée comme le montre le scénario suivant.

Scénario 2 A l'instant k, l'agent 1 effectue l'action 1, puis l'action 0, à l'instant $k+1$. Le déroulement est le suivant :

- instant k : on se trouve dans l'état 0, l'agent 1 effectue l'action 1, l'agent 2 effectue $\arg\max_b \mathsf{Q}_{2,k}(0,b) = 1$.
- instant $k+1$: on se trouve dans l'état 1, l'agent 2 met sa fonction de valeur à jour :

$$\mathsf{Q}_{2,k+1}(0,1) = (1-0.6)*0.9 + 0.6*(0+0.9*1) = 0.9. \tag{5.24}$$

L'agent 1 effectue l'action 0, l'agent 2 effectue $\arg\max_b Q_{2,k}(1,b) = 1$

– instant $k+2$: on se trouve dans l'état 0, l'agent 2 met sa fonction de valeur à jour :

$$Q_{2,k+2}(1,1) = (1-0.6) * 1 + 0.6 * (0 + 0.9 * 0.9) = 0.886 \tag{5.25}$$

La probabilité de transition du point de vue de l'agent 1 pour l'état 0 et l'action 1 est alors :

$$P_r(s_{k+3} = 1 | s_{k+2} = 0, a_{k+2} = 1) = \mathsf{T}_{1,k+2}(0,1,1) = \mathsf{T}(0, \langle 1,1 \rangle, 1) = 1 \tag{5.26}$$

On a donc :

$$P_r(s_{k+3} = 1 | s_{k+2} = 0, a_{k+2} = 1, a_k = 0) \neq P_r(s_{k+3} = 1 | s_{k+2} = 0, a_{k+2} = 1, a_k = 1) \tag{5.27}$$

Les probabilités de transition à l'instant k dépendent bien de l'action passée a_k de l'agent 1. **Le processus local n'est donc pas markovien**.

Ainsi, dans certains cas, la **propriété fondamentale de Markov** n'est plus vérifiée au **niveau local**. Les garanties théoriques de convergence des algorithmes d'apprentissage par renforcement issus du cadre mono-agent sont alors remises en cause dans le cas multiagent décentralisé.

5.4 La coordination d'agents indépendants

Comme nous l'avons au chapitre précédent, la difficulté principale pour des agents indépendants apprenant de manière décentralisée dans un système multiagent (SMA) coopératif est la coordination. Dans le cadre général des SMA, nous avons défini la coordination comme l'« articulation des actions individuelles accomplies par chacun des agents de manière à ce que l'ensemble aboutisse à un tout cohérent et performant » (définition 4.8). Nous nous intéressons dans cette section à la coordination d'agents indépendants dans le cadre des jeux de Markov d'équipe. La coordination peut alors être considérée comme le processus par lequel les décisions individuelles des agents indépendants conduisent à des décisions jointes optimales pour le groupe d'agents [Vla03]. Formellement :

Définition 5.3 *Dans un jeu de Markov d'équipe, résoudre le problème de la coordination multiagent consiste à garantir que, dans chaque état du jeu, les politiques individuelles définissent une politique jointe optimale.*

La politique jointe optimale a été définie dans le chapitre précédent comme une politique définissant un équilibre de Nash Pareto optimal dans les jeux de Markov d'équipe.

Afin d'étudier les difficultés de coordination d'agents indépendants dans les SMA, nous nous intéressons dans cette section aux facteurs responsables de ce problème de coordination. La coordination est un problème complexe qui résulte de l'action combinée

d'un grand nombre de facteurs. La classification proposée ici n'est donc sans doute pas exhaustive. Elle résulte d'une étude des jeux matriciels avec un faible nombre d'agents où l'identification et l'interprétation de certains facteurs est plus évidente. Les facteurs retenus sont évidemment transposables aux jeux à plusieurs agents ainsi qu'aux jeux de Markov, mais ils y sont moins bien identifiables et souvent combinés à d'autres phénomènes non traités ici. Nancy Fulda et Dan Ventura [FV07] distinguent deux facteurs principaux : les **équilibres cachés** [1] et la **sélection d'équilibre** [2]. Nous ajoutons deux autres facteurs, le **bruit de l'environnement** et l'**exploration** des autres agents, qui sont responsables d'observations bruitées perçues par un agent indépendant.

5.4.1 Équilibre caché

Définition 5.4 *Un équilibre défini par une politique $\bar{\pi}$ est **caché** par une politique $\hat{\pi}$ dans un état s si et seulement si :*

$$\exists i \quad \exists \pi_i \quad \mathsf{U}_{\langle \pi_i, \bar{\boldsymbol{\pi}}_{-i} \rangle}(s) < \min_{j, \pi_j} \mathsf{U}_{\langle \pi_j, \hat{\boldsymbol{\pi}}_{-j} \rangle}(s) \tag{5.28}$$

Une structure particulière de jeu illustrant ce concept peut être un jeu avec un équilibre de Nash Pareto optimal caché. Par exemple, prenons le cas du jeu *Climbing* (*cf.* figure 5.4). Il a deux équilibres de Nash : un équilibre Pareto optimal $\langle a, a \rangle$ et un équilibre sous-optimal $\langle b, b \rangle$. Ces deux équilibres de Nash sont cachés car des pénalités sont associées à une non-coordination sur $\langle a, b \rangle$ ou $\langle b, a \rangle$. De plus, il n'y a pas de pénalités de non-coordination associées à l'action c. L'action jointe $\langle c, c \rangle$ est donc l'unique équilibre non caché dans ce jeu *Climbing*. $\langle c, c \rangle$ parait donc intéressant pour des agents indépendants.

Si l'agent 2 explore en suivant par exemple une politique π_2 où il choisit ses actions de manière équiprobable, les gains immédiats espérés de l'agent 1 sont alors :

$$\begin{aligned} \mathsf{u}_{1,\langle a, \pi_2 \rangle} &= -6,33 \\ \mathsf{u}_{1,\langle b, \pi_2 \rangle} &= -5,67 \\ \mathsf{u}_{1,\langle c, \pi_2 \rangle} &= 1,67 \end{aligned} \tag{5.29}$$

L'action individuelle optimale est l'action a mais le gain immédiat espéré le plus élevé est celui de l'action c. En effet, la récompense maximale de 11 reçue pour l'action jointe $\langle a, a \rangle$ est cachée par la forte pénalité de -30 reçue en $\langle a, b \rangle$.

Ainsi, pour une même action individuelle a_i, un agent indépendant i est incapable de faire la distinction entre différentes actions jointes $\langle a_i, \boldsymbol{a}_{-i} \rangle$ et donc de distinguer les différentes récompenses reçues dues à différents comportements des autres agents. *Un mécanisme est donc nécessaire pour que l'apprentissage d'un agent indépendant ne soit pas faussé par des équilibres cachés.*

1. En anglais *action shadowing*.
2. En anglais *equilibrium selection*.

		Agent 2		
		a	b	c
Agent 1	a	5	3	1
	b	4	1	1
	c	1	1	0

FIGURE 5.3 – Un jeu matriciel d'équipe simple vis-à-vis de la coordination

		Agent 2		
		a	b	c
Agent 1	a	11	-30	0
	b	-30	7	6
	c	0	0	5

FIGURE 5.4 – Le jeu *Climbing* [CB98a].

		Agent 2		
		a	b	c
Agent 1	a	11	-30	0
	b	-30	14/0	6
	c	0	0	5

FIGURE 5.5 – Le jeu *Climbing* partiellement stochastique [KK02]. La probabilité de chaque récompense stochastique est 50%.

		Agent 2		
		a	b	c
Agent 1	a	10/12	5/-65	8/-8
	b	5/-65	14/0	12/0
	c	8/-8	12/0	10/0

FIGURE 5.6 – Le jeu *Climbing* stochastique [KK02]. La probabilité de chaque récompense stochastique est 50%.

		Agent 2		
		a	b	c
Agent 1	a	10	0	k
	b	0	2	0
	c	k	0	10

FIGURE 5.7 – Le jeu du *penalty* ($k \leq 0$) [CB98a].

5.4.2 Sélection d'équilibres

Un problème de sélection d'équilibres se manifeste lorsqu'il existe plusieurs équilibres optimaux. Dans ce cas, certains agents doivent choisir parmi plusieurs politiques individuelles optimales. Cependant, seules certaines combinaisons de politiques individuelles optimales définissent un équilibre optimal. Le problème de sélection d'équilibres se résume alors à la coordination des politiques individuelles des agents de sorte que la résultante soit optimale.

Dans le cas des stratégies pures dans les jeux matriciels, la notion d'ensemble des **actions individuelles potentiellement optimales** (PIO) a été introduite par Craig Boutilier dans son article sur la coordination dans le formalisme des MMDP [Bou96].

Définition 5.5 *L'ensemble des actions individuelles potentiellement optimales $PIO(i, s)$ pour l'agent i dans l'état s est l'ensemble des actions $a_i \in A_i$ qui appartiennent à au moins une des actions jointes optimales pour l'état s. On dit que l'état s est* faiblement dépendant *pour l'agent i s'il y a plus d'une action dans l'ensemble $PIO(i, s)$ [Bou96].*

Ainsi, un état s est faiblement dépendant s'il y a plusieurs équilibres de Nash Pareto optimaux dans cet état. Nous pouvons alors définir le problème de sélection d'équilibres à partir de cette notion. En effet, les problèmes de sélection d'équilibres apparaissent dès qu'au moins deux agents doivent choisir parmi plusieurs équilibres de Nash Pareto optimaux.

Définition 5.6 *Il y a un problème de sélection d'équilibres dans un état s si cet état est faiblement dépendant pour au moins deux agents.*

Prenons le cas du jeu *penalty* (*cf.* figure 5.7) qui présente deux équilibres de Nash Pareto optimaux : $\langle a, a\rangle$ et $\langle c, c\rangle$ et un équilibre de Nash sous-optimal $\langle b, b\rangle$. L'ensemble des actions individuelles potentiellement optimales pour le premier agent est alors $\{a, c\}$. Le second agent a le même ensemble. Le jeu *penalty* est donc faiblement dépendant pour les deux agents qui sont alors face à un problème de sélection d'équilibres. Si chaque agent choisit son action parmi son ensemble PIO, il n'y a aucune garantie que l'action jointe résultante soit optimale car quatre actions jointes sont alors possibles dont deux seulement sont optimales : $\langle a, a\rangle$, $\langle c, c\rangle$, $\langle a, c\rangle$ et $\langle c, a\rangle$.

Dans le jeu *penalty*, le paramètre k est souvent choisi inférieur à 0 de sorte à combiner le facteur de sélection d'équilibre et celui d'équilibre caché. En effet, si les deux équilibres de Nash Pareto optimaux sont cachés par de fortes pénalités, l'unique équilibre non risqué dans ce jeu est alors un équilibre de Nash sous-optimal défini par $\langle b, b\rangle$.

Donc, trouver ses actions individuelles optimales pour un agent indépendant ne suffit pas à trouver les actions jointes optimales ; un mécanisme de sélection d'équilibres est nécessaire.

5.4.3 Environnement bruité

Dans un SMA, les agents peuvent être confrontés à un environnement bruité. Les informations bruitées perçues par un agent indépendant peuvent être dues à différents phénomènes : récompenses bruitées, par exemple dans les jeux matriciels stochastiques (*cf.* figure 5.5 et figure 5.6), éléments de l'espace d'état non-observables, incertitudes, ... Cependant, lors de l'apprentissage, les divers comportements des autres agents peuvent aussi être interprétés comme du bruit par un agent indépendant. Par exemple, dans le jeu *Climbing* partiellement stochastique, un agent indépendant doit distinguer si les récompenses distinctes reçues pour avoir effectué l'action individuelle b sont dues à des comportements différents de l'autre agent ou à une fonction de récompense stochastique. L'agent doit donc apprendre que la récompense moyenne pour l'action jointe $\langle b, b\rangle$ est 7 et que la récompense moyenne pour l'action jointe $\langle a, a\rangle$ est supérieure.

Ainsi, la difficulté pour un agent indépendant dans des jeux bruités est de *faire la distinction entre le bruit dû à l'environnement et le bruit dû à l'exploration des autres agents*. Ce comportement d'exploration est notamment caractérisé par des actions d'exploration.

5.4.4 Exploration

L'exploration est un enjeu important de l'apprentissage par renforcement d'agents indépendants. En effet, comme dans le cas mono-agent, chaque agent indépendant d'un SMA choisit des actions d'exploration ou d'exploitation selon sa méthode de décision. Les actions d'exploration sont nécessaires pour explorer l'ensemble de l'environnement. Cependant, un agent indépendant ne sait pas si les actions de ses congénères sont des actions d'exploitation ou d'exploration. Or, l'exploration de l'autre peut entraîner la « destruction » de la stratégie d'un agent. Par exemple, si l'on reprend l'exemple de la preuve 5.1, l'agent 2 a une stratégie gloutonne optimale à l'instant k :

$$\pi_{2,k}(0,0) = 0 \quad \pi_{2,k}(0,1) = 1 \quad \pi_{2,k}(1,0) = 0 \quad \pi_{2,k}(1,1) = 1. \tag{5.30}$$

Si l'agent 1 explore en choisissant par exemple aux instants k et $k+1$ l'action 0, comme dans le scénario 1, alors à l'instant $k+2$, la stratégie gloutonne de l'agent 2 est :

$$\pi_{2,k+2}(0,0) = 1 \quad \pi_{2,k+2}(0,1) = 0 \quad \pi_2, k+2(1,0) = 0 \quad \pi_{2,k+2}(1,1) = 1. \tag{5.31}$$

La stratégie optimale a été « détruite » par le comportement d'exploration de l'autre agent. Dans un jeu où les équilibres optimaux sont risqués, la stratégie optimale est encore plus soumise à l'exploration de l'autre. L'exploration de l'autre et les pénalités que cette exploration engendre dans un jeu où sont présents des équilibres risqués peuvent donc amener l'agent à suivre une stratégie sûre mais non-optimale et peut aussi entraîner la destruction de sa stratégie optimale. Avec une stratégie sûre, aucune pénalité ne sera reçue quelle que soit l'exploration de l'autre. L'exploration d'agents indépendants entraîne donc du « bruit » qui peut être néfaste à l'apprentissage.

Pour diminuer ce « bruit » dû à l'exploration des autres agents, il faut tout d'abord éviter l'exploration simultanée des agents. Caroline Claus et Craig Boutilier [CB98a] appellent cela diminuer le risque **d'exploration « concurrente »** . Une méthode couramment employée avec l'apprentissage par renforcement multiagent est l'utilisation d'une stratégie d'exploration GLIE [3] [SJLS00] qui consiste à décroître le taux d'exploration au fur et à mesure de l'apprentissage. Ainsi, au début de l'apprentissage, les agents explorent. Puis, l'exploration diminue de sorte à éviter que les agents explorent en même temps. Chacun pourra alors trouver la meilleure réponse au comportement des autres.

Illustrons l'intérêt de cette méthode sur le jeu *Climbing* (*cf.* figure 5.4). Nous rappelons qu'à cause des pénalités de non-coordination et donc des équilibres risqués que sont $\langle a, a\rangle$ et $\langle b, b\rangle$, l'action jointe sûre $\langle c, c\rangle$ est attractive pour des agents ne sachant pas surmonter les facteurs de non-coordination. Au début de l'apprentissage, les agents explorent car ils suivent une stratégie d'exploration GLIE. Si les deux agents explorent en choisissant leurs actions de manière équiprobable, les gains immédiats espérés sont alors :

$$\begin{array}{cc} \mathsf{u}_1(a) = -6,3333 & \mathsf{u}_2(a) = -6,3333 \\ \mathsf{u}_1(b) = -5,6667 & \mathsf{u}_2(b) = -7,6667 \\ \mathsf{u}_1(c) = 1,67 & \mathsf{u}_2(c) = 3,6667 \end{array}$$

Les stratégies apprises par les deux agents lors de cette première phase définissent donc l'action jointe sûre $\langle c, c\rangle$. Ensuite, le taux d'exploration diminue. Supposons que l'agent 2 exploite sa stratégie, *i.e.* qu'il joue l'action gloutonne c. L'agent 1 cherche alors la meilleure réponse à ce comportement en variant des actions d'exploration et d'exploitation : l'action b permet de recevoir une plus forte récompense et l'action gloutonne de l'agent 1 devient l'action b. Les rôles sont ensuite inversés : l'agent 1 joue uniquement son action gloutonne sans explorer et l'agent 2 cherche une meilleure réponse au nouveau comportement de l'agent 1. Alors la nouvelle action gloutonne de l'agent 2 est l'action b. L'équilibre $\langle b, b\rangle$ peut donc être atteint en évitant l'exploration concurrente grâce à une stratégie GLIE. Cet équilibre n'est pas optimal mais est un équilibre de Nash ; le résultat est donc plus satisfaisant que la convergence vers l'action jointe sûre.

Cependant, cette technique consistant à éviter **l'exploration « concurrente »** est difficilement applicable. En effet, les paramètres de décroissance choisis doivent assurer que le bruit dû à l'exploration concurrente soit évité. Cette méthode requiert un grand nombre d'épisodes pour l'apprentissage et est de plus très difficile à appliquer à cause du réglage des paramètres de décroissance qui demande une forte expertise sur le comportement des agents. La difficulté d'application de cette méthode avec des algorithmes d'apprentissage sera présentée dans le chapitre suivant.

Un autre inconvénient provient du fait que même si les agents n'explorent pas simultanément, lorsqu'un des agents explore, les autres apprennent. Cette exploration de

3. En anglais *greedy in the limit with infinite exploration*.

l'un d'entre eux peut alors détruire la stratégie de ceux qui exploitent, comme expliqué dans le premier paragraphe. Une première solution est de quantifier le bruit perçu par un agent indépendant et causé par l'exploration des autres agents. Cette solution sera détaillée au §7.2.3 de ce mémoire. Une autre solution pourrait être de permettre à chaque agent de savoir si les actions de ses congénères sont des actions d'exploration ou d'exploitation (sans savoir quelles sont ces actions). De cette manière, les agents pourraient par exemple ne pas apprendre lorsqu'un de leur congénère explore. Une pénalité serait alors due uniquement à une non-coordination ou à une récompense bruitée. Cette idée de suspendre la mise à jour des Q-valeurs dans le cas d'une d'exploration est testée par [dCLR06] avec des agents percevant les actions jointes.

5.4.5 Conclusion

Nous avons présenté dans cette section quatre facteurs responsables de la non coordination d'agents indépendants dans des jeux d'équipe. Ces facteurs sont responsables de deux difficultés : les agents doivent apprendre quelles politiques individuelles sont optimales malgré les phénomènes d'équilibre caché et de bruit et ils doivent de plus, s'ils ont plusieurs choix, coordonner leurs politiques individuelles optimales pour s'accorder sur une unique politique jointe résultante optimale. Résoudre ces deux difficultés et donc surmonter les facteurs de non-coordination est un enjeu principal pour les algorithmes d'apprentissage par renforcement dans les SMA. De plus, un autre facteur clé est la robustesse face à l'exploration des autres agents.

5.5 Conclusion

Dans ce chapitre, nous avons rappelé les objectifs de l'apprentissage en terme de fonctions de valeur locales et globales. De plus, trois enjeux soulevés par l'apprentissage par renforcement d'agents indépendants ont été détaillés et mis en lumière sur des exemples de jeux matriciels d'équipe. Le premier d'entre eux concerne la remise en cause de la propriété fondamentale de Markov au niveau local. La seconde difficulté étudiée est la coordination et plus précisément, nous avons identifié dans les jeux où le nombre d'agents est faible **quatre facteurs principaux de non-coordination**. Notamment, **trouver un algorithme robuste à la stratégie d'exploration des autres agents** est un enjeu important qui est très peu évoqué dans la littérature et rarement pris en compte dans les algorithmes d'apprentissage par renforcement décentralisés qui vont être présentés dans le chapitre suivant. Les difficultés rencontrées avec ces algorithmes seront mieux appréhendées grâce à l'étude proposée dans ce chapitre.

Chapitre 6

État de l'art des méthodes par apprentissage par renforcement pour agents indépendants dans les jeux de Markov d'équipe

Ce chapitre présente un état de l'art des méthodes par apprentissage par renforcement pour agents indépendants dans les jeux de Markov d'équipe. Notamment, il détaille les algorithmes décentralisés basés sur le Q-learning et ses variantes et met en avant leurs points communs grâce à une notation uniforme. Des résultats de simulation obtenus avec des jeux matriciels sont présentés. Ce chapitre propose enfin une synthèse des enjeux surmontés par chacun de ces algorithmes.

6.1 Introduction

Ce chapitre est consacré à un état de l'art des travaux concernant l'apprentissage par renforcement d'agents indépendants dans le cadre des jeux de Markov d'équipe. Souhaitant éviter une conception centralisée, nous nous intéressons aux algorithmes décentralisés. Les travaux précurseurs sont ceux de Ming Tan sur le jeu des prédateurs et des proies [Tan93] et ceux de Sandip Sen *et. al.* sur une tâche de *box-pushing* [SSH94]. Dans le chapitre précédent ont été présentés les principaux enjeux de l'apprentissage d'agents indépendants. Outre les problèmes de la coordination et de l'exploration, les résultats théoriques de l'apprentissage par renforcement mono-agent ne peuvent pas s'appliquer directement à des algorithmes décentralisés, étant donné que le processus local à un agent peut ne pas être markovien si les autres agents sont « apprenants » (cf. propriété 5.1). Sont aussi présents des problèmes d'explosion de la taille de l'espace d'état et de

complexité de calculs.

Malgré ces difficultés, de nombreuses applications réussies de l'apprentissage par renforcement décentralisé ont été obtenues, comme le contrôle d'une flotte d'ascenseurs [CB98b], d'un bras manipulateur [BBS06a] ou une application d'équilibrage de charges [1] [VNPT07]. La gestion du trafic aérien [TA07] et le transport de données par une constellation de satellites de communication [WST01] sont d'autres résultats encourageants de l'application de l'apprentissage par renforcement à des agents indépendants coopératifs.

Ainsi, plusieurs approches ont été proposées dans la littérature pour résoudre les enjeux de l'apprentissage d'agents indépendants. Un état de l'art des différents algorithmes situés dans le cadre des jeux de Markov d'équipe est ici présenté. Notamment, nous mettons l'accent sur les algorithmes basés sur le Q-learning (*cf.* §2.5.1) et ses variantes. Pour la plupart des algorithmes présentés ci-après, une notation uniformisée est utilisée qui permet notamment de faire apparaître des points communs entre certains d'entre eux. De plus, les facteurs de non-coordination que chaque algorithme est capable d'éviter et les mécanismes utilisés sont détaillés. Quelques résultats obtenus avec ces algorithmes dans les jeux matriciels des figures 5.3 à 5.7 seront aussi exposés afin d'illustrer nos propos. Bien que ces jeux matriciels n'aient seulement que deux agents et un faible nombre d'actions, ils sont beaucoup étudiés dans la littérature car l'apprentissage d'agents indépendants y est difficile à cause de ces enjeux qu'ils doivent surmonter. En effet, on y retrouve notamment les trois facteurs de non-coordination mis à jour dans le chapitre précédent. De plus, l'interprétation de ces facteurs est plus évidente. Tester ces algorithmes dans ces jeux est donc un préambule à leur étude dans le cadre des jeux de Markov et permet surtout une meilleure compréhension des problèmes que connaît l'apprentissage par renforcement d'agents indépendants.

6.2 Q-learning décentralisé

L'algorithme du Q-learning est un des algorithmes les plus utilisés en mono-agent pour sa simplicité et sa robustesse. Ce sont aussi des raisons pour lesquelles il a été l'un des premiers algorithmes à être appliqué de manière décentralisée aux systèmes multiagents [Tan93]. L'équation de mise à jour pour un agent i dans l'état s, ayant effectué l'action individuelle a_i, dont l'action jointe $\boldsymbol{a} = \langle a_i, \boldsymbol{a}_{-i} \rangle$ fait évoluer le système dans un nouvel état s' et où tous les agents reçoivent la récompense $\mathsf{R}(s, \boldsymbol{a})$ est :

$$\mathsf{Q}_i(s, a_i) \leftarrow (1 - \alpha)\mathsf{Q}_i(s, a_i) + \alpha(\mathsf{R}(s, \langle a_i, \boldsymbol{a}_{-i} \rangle) + \gamma \max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s', b)). \tag{6.1}$$

L'algorithme du Q-learning décentralisé est donné à l'algorithme 4. Dans cet algorithme (et ceux qui suivent), si la méthode de décision n'est pas précisée, son choix est alors laissé à l'utilisateur parmi les méthodes existantes (ϵ-greedy, softmax, ...). Malgré le

1. En anglais *load balancing*.

Algorithme 4 : Algorithme du Q-learning décentralisé pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s, a_i)$ arbitrairement pour tout (s, a_i) de $\mathcal{S} \times \mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i ▷ `étape de décision`
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $\mathsf{Q}_i(s, a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(s, a_i) + \alpha \left[r + \gamma \max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s', b) \right]$
    $s \leftarrow s'$
fin

TABLE 6.1 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le Q-learning décentralisé selon la stratégie d'exploration (moyenne sur 1000 épisodes indépendants). Un épisode consiste en 3000 répétitions. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale. La méthode de décision softmax est utilisée avec différentes stratégies d'exploration : stationnaire (τ) ou GLIE ($\tau = \tau_{ini} \times \delta^t$ où t est le nombre de répétitions).

Stratégie stationnaire	$\tau = 1000$	$\tau = 200$	$\tau = 10$	$\tau = 1$
Jeu *penalty* ($k = -100$)	$62,6\%$	$65,3\%$	$62,8\%$	$65,1\%$

Stratégie GLIE	$\tau_{ini} = 5000$ $\delta = 0.9$	$\delta = 0.99$	$\delta = 0.995$	$\delta = 0.997$	$\tau_{ini} = 500$ $\delta = 0.995$	$\tau_{ini} = 100$ $\delta = 0.995$
Jeu *penalty* ($k = -100$)	0%	$61,7\%$	84%	$96,6\%$	$85,3\%$	$84,8\%$

manque de justifications théoriques dans le cadre multiagent, il a été appliqué avec succès à certaines applications [SSH94, SS98, BBS06a, WdS06]. Il est cependant important de remarquer que cet algorithme est peu robuste face à la stratégie d'exploration (choix des paramètres de la méthode de décision) ainsi que face aux facteurs de non-coordination.

6.2.1 Robustesse face à la stratégie d'exploration

Dans le Q-learning décentralisé, la méthode de décision ainsi que la stratégie d'exploration ne sont pas précisées. Cependant, les différents résultats disponibles dans la littérature ainsi que les essais que nous avons effectués montrent que **la stratégie d'exploration joue un rôle important dans la réussite du Q-learning décentralisé**. La plupart des travaux sur le Q-learning décentralisé utilisent une stratégie d'exploration GLIE[2] [SJLS00] qui consiste à décroître le taux d'exploration au fur et à mesure de

2. En anglais *greedy in the limit with infinite exploration*.

l'apprentissage. Cette méthode est couramment choisie pour assurer la convergence vers une politique optimale déterministe des algorithmes *on-policy* mono-agent tel que l'algorithme Sarsa. Néanmoins, peu d'auteurs justifient l'utilisation de cette stratégie dans le cadre multiagent. Selon Caroline Claus et Craig Boutilier [CB98a], cette méthode est plus efficace qu'une exploration constante car elle permet de réduire l'exploration concurrente (*cf.* §5.4.4). Une stratégie GLIE peut donc permettre dans certains cas d'améliorer les résultats obtenus avec les algorithmes décentralisés, et notamment avec le Q-learning décentralisé. La table 6.1 donne les différents résultats de convergence possibles selon les paramètres de la stratégie d'exploration. Avec une stratégie stationnaire, le choix du taux d'exploration a peu d'influence sur les résultats de convergence. Par contre, les paramètres de décroissance d'une stratégie GLIE ont beaucoup d'influence sur la réussite du Q-learning décentralisé. Notamment, avec une stratégie GLIE correctement choisie, on peut obtenir 96% d'épisodes réussis dans le jeu du *penalty* avec de fortes pénalités ($k = -100$). *Le Q-learning décentralisé est donc peu robuste face à la stratégie d'exploration qui influe fortement sur la convergence.* Il faut insister sur le fait que le choix des paramètres de décroissance n'est pas évident. Il nécessite une très bonne expertise du comportement de l'algorithme d'apprentissage, comportement qui varie selon le type de jeu. **Le réglage d'une stratégie GLIE rend donc le Q-learning décentralisé difficilement utilisable dans la pratique**.

6.2.2 Robustesse face aux facteurs de non-coordination

La structure du jeu et plus particulièrement la présence de facteurs de non-coordination agit aussi sur la convergence du Q-learning décentralisé. Les résultats de la table 6.2 illustrent ce phénomène. Dans un jeu matriciel d'équipe simple (par exemple celui de la figure 5.3), le Q-learning décentralisé converge vers l'action jointe optimale si la stratégie d'exploration est bien choisie. Mais dans le jeu *Climbing*, la convergence sur l'équilibre de Nash Pareto optimal n'est pas réalisée car le Q-learning décentralisé ne surmonte pas la difficulté de l'équilibre caché. Dans le jeu matriciel *penalty*, la convergence du Q-learning décentralisé sur une des deux actions jointes optimales peut être réalisée avec une stratégie d'exploration très bien choisie. Aucun mécanisme n'est explicitement intégré dans le Q-learning décentralisé pour assurer la sélection d'équilibres, mais celui-ci semble y parvenir dans le jeu *penalty*, notamment avec de faibles pénalités. La stratégie d'exploration GLIE qui permet d'éviter le bruit dû à l'exploration concurrente joue implicitement un rôle de mécanisme de sélection d'équilibres. Par contre, plus la valeur des pénalités est élevée, et donc plus les équilibres optimaux sont risqués, et plus la convergence est difficile (*cf.* figure 6.1).

6.2.3 Synthèse

Ainsi, le Q-learning décentralisé ne surmonte aucun des enjeux de l'apprentissage d'agents indépendants. Il n'est robuste ni à la stratégie d'exploration ni aux facteurs de non coordination. Cependant, il est à noter que dans certains jeux et avec une stratégie d'exploration bien choisie, le Q-learning décentralisé peut tout de même donner de bons

TABLE 6.2 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le Q-learning décentralisé (moyenne sur 500 épisodes indépendants). Un épisode consiste en 3000 répétitions. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale. La méthode de décision softmax est utilisée avec une stratégie d'exploration GLIE ($\tau = 5000 \times 0.997^t$ où t est le nombre de répétitions) de sorte à obtenir les meilleurs résultats possibles de convergence.

	Jeu simple	Jeu *penalty* ($k = -100$)	Jeu *Climbing* déterministe
Q-learning décentralisé	100%	96%	3%

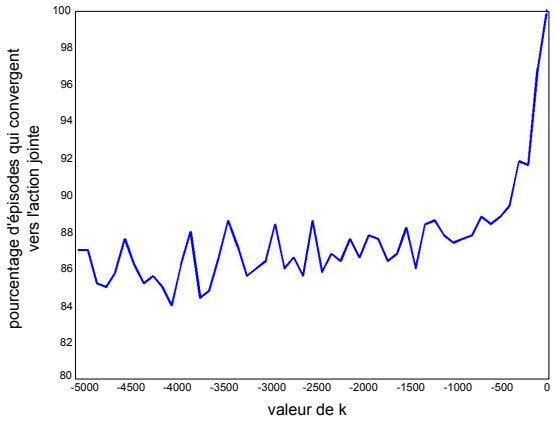


FIGURE 6.1 – Pourcentage d'épisodes convergeant vers l'une des deux actions jointes optimales dans le jeu *penalty* selon la valeur de k avec le Q-learning décentralisé (moyenne sur 500 épisodes indépendants). Un épisode consiste en 3000 répétitions. La méthode de décision softmax est utilisée avec une stratégie d'exploration GLIE ($\tau = 5000 \times 0.997^t$ où t est le nombre de répétitions). A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

résultats.

6.3 Q-learning distribué

Martin Lauer et Martin Riedmiller [LR00] proposent un algorithme décentralisé pour agents indépendants dans le cadre des jeux de Markov d'équipe appelé Q-learning distribué [3]. Les politiques calculées par cet algorithme sont des politiques déterministes. Cet algorithme est capable de résoudre les deux facteurs de non-coordination présents dans les environnements déterministes. Le phénomène d'équilibres cachés est évité grâce

3. En anglais *Distributed Q-learning*.

à l'emploi d'**agents indépendants optimistes** qui apprennent des politiques individuelles optimales. Dans le cas où différents équilibres optimaux existent, un **mécanisme de sélection d'équilibres** est utilisé pour régler le second facteur de non-coordination.

6.3.1 Agents indépendants optimistes

La caractéristique d'agents optimistes est d'ignorer dans leur équation de mise à jour les pénalités reçues qui sont souvent dues à une non-coordination des agents. Ainsi, un agent optimiste met à jour la valeur d'un couple état-action selon le Q-learning seulement si la mise à jour entraîne une amélioration de la valeur courante de ce couple. L'environnement est déterministe par hypothèse donc le coefficient d'apprentissage α est choisi égal à 1. Un agent optimiste i maintient les valeurs de sa table individuelle Q_i pour un couple état-action (s, a_i) selon l'équation de mise à jour du Q-learning distribué, *i.e.* :

$$\mathsf{Q}_i(s, a_i) \leftarrow \max \left\{ \mathsf{Q}_i(s, a_i), \mathsf{R}(s, \langle a_i, \boldsymbol{a}_{-i} \rangle) + \gamma \max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s', b) \right\}. \tag{6.2}$$

Les tables Q_i d'agents optimistes vérifient l'hypothèse optimiste (*cf.* équation 5.11). En effet, on a le théorème suivant :

Théorème 6.1 *[LR00] Soit $\mathcal{M} = \langle m, \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_m, \mathsf{T}, \mathsf{R} \rangle$ un jeu de Markov d'équipe **déterministe**. Si :*

- $\forall s \in \mathcal{S} \quad \forall \mathbf{a} \in \boldsymbol{\mathcal{A}} \quad \mathsf{R}(s, \mathbf{a}) \geq 0$,
- $\forall i \quad \forall s \in \mathcal{S} \quad \forall a_i \in \mathcal{A}_i$ *les valeurs des tables* $\mathsf{Q}_i(s, a_i)$ *sont initialisées à* 0,
- $\forall i$ *les fonctions locales* Q_i *sont mises à jour selon l'équation 6.2,*
- *la fonction globale* Q *est mise à jour selon l'équation du Q-learning et initialisée à* 0,
- *les tables* Q *et* Q_i *sont mises à jour avec les mêmes séquences d'états et d'actions,*

alors, à chaque itération :

$$\forall i \quad \forall s \in \mathcal{S} \quad \forall a_i \in \mathcal{A}_i \quad \mathsf{Q}_i(s, a_i) = \max_{\mathbf{u}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} \mathsf{Q}(s, \langle a_i, \mathbf{u}_{-i} \rangle). \tag{6.3}$$

Preuve 6.1 [LR00]

- A l'état initial $t = 0$, les deux tables sont initialisées à 0 donc l'équation 6.3 est vraie.
- L'égalité étant supposée vraie à l'instant t, nous devons démontrer l'égalité 6.3 à l'instant $t + 1$. Soit un agent j dans l'état s_t à l'instant t, ayant effectué l'action individuelle $a_{j,t}$, dont l'action jointe $\boldsymbol{a}_t = \langle a_{j,t}, \boldsymbol{a}_{-j,t} \rangle$ fait évoluer le système dans un nouvel état s_{t+1} et où tous les agents reçoivent la récompense $\mathsf{R}(s_t, \boldsymbol{a}_t)$. On a :

$$\mathsf{Q}_{j,t}(s_t, a_{j,t}) = \max_{\boldsymbol{u}_{-j} \in \boldsymbol{\mathcal{A}}_{-j}} \mathsf{Q}_t(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j} \rangle). \tag{6.4}$$

La fonction Q_j est mise à jour selon l'équation du Q-learning distribué (équation 6.2) donc :

$$\begin{aligned}\mathsf{Q}_{j,t+1}(s_t, a_{j,t}) &= \max\left\{\mathsf{Q}_{j,t}(s_t, a_{j,t}), \mathsf{R}(s_t, \boldsymbol{a}_t) + \gamma \max_{b\in\mathcal{A}_j} \mathsf{Q}_{j,t}(s_{t+1}, b)\right\} \\ &= \max\left\{\max_{\boldsymbol{u}_{-j}\in\boldsymbol{\mathcal{A}}_{-j}} \mathsf{Q}_t(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle), \mathsf{R}(s_t, \boldsymbol{a}_t) + \gamma \max_{\mathbf{v}\in\boldsymbol{\mathcal{A}}} \mathsf{Q}_t(s_{t+1}, \mathbf{v})\right\}\end{aligned} \tag{6.5}$$

d'après l'hypothèse de récurrence (équation 6.4).
La fonction globale Q est mise à jour selon l'équation du Q-learning centralisé en environnement déterministe donc :

$$\mathsf{Q}_{t+1}(s_t, \boldsymbol{a}_t) = \mathsf{R}(s_t, \boldsymbol{a}_t) + \gamma \max_{\mathbf{v}\in\boldsymbol{\mathcal{A}}} \mathsf{Q}_t(s_{t+1}, \mathbf{v}). \tag{6.6}$$

On obtient alors :

$$\begin{aligned}\mathsf{Q}_{j,t+1}(s_t, a_{j,t}) &= \max\left\{\max_{\boldsymbol{u}_{-j}\in\boldsymbol{\mathcal{A}}_{-j}} \mathsf{Q}_t(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle), \mathsf{Q}_{t+1}(s_t, \boldsymbol{a}_t)\right\} \\ &= \max\left\{Q_t(s_t, \boldsymbol{a}_t), \max_{\substack{\boldsymbol{u}_{-j}\in\boldsymbol{\mathcal{A}}_{-j} \\ \boldsymbol{u}_{-j}\neq\boldsymbol{a}_{-j,t}}} \mathsf{Q}_t(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle), \mathsf{Q}_{t+1}(s_t, \boldsymbol{a}_t)\right\}\end{aligned} \tag{6.7}$$

Étant donnée l'équation de mise à jour de la fonction Q en environnement déterministe, la positivité de la fonction R et l'initialisation à zéro de Q pour tout couple du système, on en déduit que les valeurs de la fonction Q sont monotones par rapport à t, *i.e.* $\mathsf{Q}_t(s, \boldsymbol{a}) \leq \mathsf{Q}_{t+1}(s, \boldsymbol{a}) \quad \forall (s, \boldsymbol{a}) \in \mathcal{S} \times \boldsymbol{\mathcal{A}}$. D'où :

$$\mathsf{Q}_{j,t+1}(s_t, a_{j,t}) = \max\left\{\max_{\substack{\boldsymbol{u}_{-j}\in\boldsymbol{\mathcal{A}}_{-j} \\ \boldsymbol{u}_{-j}\neq\boldsymbol{a}_{-j,t}}} \mathsf{Q}_t(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle), \mathsf{Q}_{t+1}(s_t, \boldsymbol{a}_t)\right\} \tag{6.8}$$

De plus, la valeur de la fonction Q pour le couple $(s_t, \boldsymbol{u}_t)$ avec $\boldsymbol{u}_t \neq \boldsymbol{a}_t$ n'est pas modifiée entre l'instant t et l'instant $t+1$, donc :

$$\begin{aligned}\mathsf{Q}_{j,t+1}(s_t, a_{j,t}) &= \max\left\{\max_{\substack{\boldsymbol{u}_{-j}\in\boldsymbol{\mathcal{A}}_{-j} \\ \boldsymbol{u}_{-j}\neq\boldsymbol{a}_{-j,t}}} \mathsf{Q}_{t+1}(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle), \mathsf{Q}_{t+1}(s_t, \boldsymbol{a}_t)\right\} \\ &= \max_{\boldsymbol{u}_{-j}\in\mathcal{A}_{-j}} \mathsf{Q}_{t+1}(s_t, \langle a_{j,t}, \boldsymbol{u}_{-j}\rangle)\end{aligned} \tag{6.9}$$

On a bien l'égalité 6.3 vraie à l'instant $t+1$. Le théorème 6.1 est donc vérifié.

Ainsi, **tout agent optimiste est capable de déterminer la fonction de valeur locale optimale dans un jeu de Markov d'équipe déterministe**. Dans ce cas, les actions individuelles gloutonnes sur la fonction de valeur locale optimale d'un agent appartiennent aux actions jointes optimales.

Cependant, nous avons précisé au §5.2.4 que cette condition n'était pas suffisante. *Un mécanisme de sélection d'équilibres doit être mis en place pour éviter l'ambiguïté sur le choix d'un équilibre.*

6.3.2 Mécanisme de sélection d'équilibres pour agents optimistes

Un mécanisme additionnel de sélection d'équilibres est utilisé pour les agents optimistes par Martin Lauer et Martin Riedmiller [LR00]. Ce mécanisme, inspiré par les travaux de Craig Boutilier [Bou96], consiste à poser des règles permettant de lever l'ambiguïté sur le choix d'une action individuelle optimale. Un tel mécanisme est appelé une *convention* [ST92].

Une convention est donc utilisée pour mettre à jour la politique π_i de chaque agent, maintenue parallèlement aux tables Q_i. La contrainte imposée par la convention est que chaque agent i mette à jour sa politique courante déterministe π_i dans un état s seulement si l'action choisie a_i entraîne une amélioration dans l'évaluation de la valeur $\mathsf{Q}_i(s, a_i)$. Dans ce cas, la probabilité de choisir cette action dans l'état s est mise à 1. Ainsi, lors de la première occurrence d'une action jointe optimale, les actions individuelles optimales correspondantes sont mémorisées par les politiques et celles-ci ne sont plus jamais modifiées.

La loi de mise à jour de chaque politique individuelle est donnée à l'algorithme 5 qui détaille le Q-learning distribué pour un agent. Étant donnée l'initialisation, nous supposons que les récompenses sont positives. *Toute politique jointe déterministe $\boldsymbol{\pi}$ calculée par ce mécanisme de coordination avec des agents indépendants optimistes est assurée d'être une politique gloutonne sur la fonction globale* Q, *i.e.* :

$$\forall s \in \mathcal{S} \quad \mathsf{Q}(s, \boldsymbol{\pi}(s)) = \max_{\boldsymbol{u} \in \boldsymbol{\mathcal{A}}} \mathsf{Q}(s, \boldsymbol{u}). \tag{6.10}$$

Une preuve est disponible dans [LR00].

Un des enjeux de l'apprentissage d'agents indépendants, évoqué au §5.4.4, est de maîtriser l'influence de l'exploration sur l'apprentissage des politiques. Or, l'intérêt principal du mécanisme de coordination du Q-learning distribué est que la politique optimale apprise par un agent optimiste est indépendante de l'exploration car elle ne peut pas être « détruite » par des actions d'exploration des autres, contrairement au Q-learning décentralisé par exemple (*cf.* §6.2.1). **L'algorithme du Q-learning distribué est donc robuste face à l'exploration** dans le sens où l'apprentissage d'une politique peut être simultanée à l'exploration.

Algorithme 5 : Algorithme du Q-learning distribué pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_{max,i}(s,a_i)$ à 0 et $\pi_i(s,a_i)$ arbitrairement pour tout (s,a_i) de $\mathcal{S}\times\mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i selon la méthode de décision ϵ-greedy basée sur π_i
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $q \leftarrow r + \gamma \max_{u\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s',u)$
    si $q > \mathsf{Q}_{max,i}(s,a_i)$ **alors**
      $\mathsf{Q}_{max,i}(s,a_i) \leftarrow q$ ▷ `mise à jour optimiste`
    si $\mathsf{Q}_{max,i}(s, \arg\max_{u\in\mathcal{A}_i} \pi_i(s,u)) \neq \max_{u\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s,u)$ **alors**
      Sélectionner $a_{max} \in \arg\max_{u\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s,u)$ aléatoirement
      pour tous les $u \in \mathcal{A}_i$ ▷ `sélection d'équilibres`
      faire
        si $u = a_{max}$ **alors**
          $\pi_i(s,u) \leftarrow 1$
        sinon
          $\pi_i(s,u) \leftarrow 0$
    $s \leftarrow s'$
fin

6.3.3 Résultats sur des jeux matriciels

Le Q-learning distribué est donc assuré de **surmonter les facteurs de non-coordination présents dans les jeux déterministes**. L'autre intérêt de cet algorithme est que grâce à la mise à jour optimiste, la fonction de valeur d'un agent ne peut pas être « détruite » par l'exploration d'un autre agent. La stratégie d'exploration est donc simple ; il suffit de permettre un minimum d'exploration pour que tous les couples du système soient visités. Si c'est le cas, l'action jointe optimale aura été choisie au moins une fois et mémorisée par les politiques individuelles. **Le Q-learning distribué est donc robuste à l'exploration**.

Des résultats sont donnés à la table 6.3. La convergence est effective dans les jeux *penalty* et *Climbing* déterministe où sont présents les facteurs d'équilibres cachés et de sélection d'équilibres. Naturellement, les agents échouent dans le jeu *Climbing* partiellement bruité. En effet, pour l'action jointe $\langle b,b\rangle$, les récompenses 7 et 14 sont obtenues de manière équiprobable. La récompense moyenne pour cette action jointe est donc inférieure à celle de l'action jointe optimale $\langle a,a\rangle$. Cependant, des agents optimistes évaluent la valeur de l'action jointe $\langle b,b\rangle$ par la récompense maximale, c'est-à-dire 14, et donc se coordonnent sur cet équilibre sous-optimal.

TABLE 6.3 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le Q-learning distribué (moyenne sur 500 épisodes indépendants). Un épisode consiste en 3000 répétitions et on prend $\epsilon = 0.05$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

	Jeu simple	Jeu *Penalty* ($k = -100$)	Jeu *Climbing*	
			déterministe	partiellement bruité
Q-learning distribué	100%	100%	100%	7%

6.3.4 Synthèse

L'algorithme du Q-learning distribué est donc capable de résoudre les deux facteurs de non-coordination présents dans les environnements déterministes : les équilibres cachés et la sélection d'équilibres. De plus, l'enjeu de l'exploration est surmonté grâce au mécanisme de sélection d'équilibres. Chaque agent suit sa politique avec un minimum d'exploration, mais les problèmes d'exploration concurrente et de destruction de politique sont ici évités. Donc, *cet algorithme est assuré de converger vers une action jointe optimale dans tous jeux déterministes tout en étant robuste à l'exploration et aux facteurs de non-coordination.*

6.4 Frequency Maximum Q-value

6.4.1 L'heuristique Frequency Maximum Q-value

Spiros Kapetanakis et Daniel Kudenko [KK02, KK04] proposent d'influencer la probabilité de choisir une action dans leur algorithme Frequency Maximum Q-value (FMQ). Pour cela, ils utilisent la méthode de décision softmax qui assigne une probabilité à chaque action en fonction de l'évaluation de celle-ci. La méthode softmax classique évalue les actions avec leurs Q-valeurs, calculées selon le Q-learning. Spiros Kapetanakis et Daniel Kudenko proposent d'évaluer une action en prenant en compte sa Q-valeur ainsi qu'une heuristique. Cette heuristique est fonction de la récompense maximale reçue et de la fréquence d'occurrence de celle-ci lorsqu'est effectuée une action. Plusieurs tables sont donc nécessaires :

- une table C pour mémoriser le nombre de fois qu'une action a été choisie,
- une table Q_{max} pour mémoriser la récompense maximale reçue jusqu'alors après chaque action,
- une table $\mathsf{C}_{Q_{max}}$ pour mémoriser le nombre de fois que la récompense maximale a été reçue après avoir effectué une action.

La fréquence d'occurrence F de la récompense maximale associée à une action a est alors $\mathsf{F}(a) = \frac{\mathsf{C}_{Q_{max}}(a)}{\mathsf{C}(a)}$. L'évaluation $\mathsf{E}(a)$ d'une action a est définie par :

$$\mathsf{E}(a) = \mathsf{Q}(a) + c\ \mathsf{F}(a)\ \mathsf{Q}_{max}(a) \tag{6.11}$$

TABLE 6.4 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le FMQ (moyenne sur 500 épisodes indépendants). Un épisode consiste en 3000 répétitions. La stratégie d'exploration choisie est GLIE ($\tau = 499e^{-0.006t} + 1$ où t est le nombre de répétitions). On pose $\alpha = 0,1$ et $c = 10$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

	Jeu simple	Jeu *Penalty* ($k = -100$)	Jeu *Climbing*		
			déterministe	partiellement bruité	bruité
FMQ	100%	100%	100%	98%	21%

où c est un paramètre contrôlant l'importance de l'heuristique dans l'évaluation d'une action. Si $c = 0$, le FMQ est équivalent au Q-learning décentralisé. L'algorithme du Frequency Maximum Q-value (FMQ) est donné à l'algorithme 6. Étant donnée l'initialisation, nous supposons que les récompenses sont positives. On peut remarquer l'utilisation dans le FMQ de la table Q_{max} pour mémoriser les évaluations optimistes. Cette table est la même que celle du Q-learning distribué.

Les auteurs de cet algorithme utilisent la méthode de décision softmax avec les évaluations E et une stratégie d'exploration GLIE. Le choix de ce type d'exploration est une fois de plus non justifié alors qu'il est crucial pour la réussite du FMQ. La robustesse de cette heuristique face à l'exploration est étudiée dans le chapitre suivant.

6.4.2 Résultats sur des jeux matriciels

Les résultats obtenus avec le FMQ dans divers jeux matriciels sont intéressants (*cf.* table 6.4). En effet, si la stratégie d'exploration est bien réglée, les résultats sont équivalents à ceux du Q-learning distribué dans des jeux déterministes. De plus, alors que ce dernier échoue dans des jeux stochastiques, le FMQ montre de bons résultats dans des jeux partiellement bruités.

Par contre, le FMQ ne surmonte pas la difficulté de jeux fortement bruités comme le *Climbing* bruité. Ce jeu présente la caractéristique d'avoir une récompense bruitée à 50% pour **toutes** les actions jointes (*cf.* figure 5.6). La moyenne des deux récompenses stochastiques possibles pour chaque action jointe est équivalente à la valeur de la récompense déterministe dans le jeu *Climbing*. Cependant, l'heuristique utilisée avec le FMQ prend en compte uniquement la récompense maximale reçue pour chaque action et sa fréquence (F Q_{max}). Or, la récompense maximale pour l'action jointe optimale $\langle a, a\rangle$ est 12 alors qu'elle est de 14 pour l'action jointe sous-optimale $\langle b, b\rangle$. Et la fréquence est ici de 0.5 pour chaque récompense maximale si les actions jointes sont choisies de manière quasiment équiprobable. Après la première phase d'exploration induite par la stratégie GLIE, l'action jointe sous-optimale $\langle b, b\rangle$ présente donc une récompense moyenne de 7 selon l'heuristique alors que l'action jointe optimale $\langle a, a\rangle$ a une récompense moyenne de 6. Le FMQ peut donc être faussement biaisé dans certains jeux bruités.

Algorithme 6 : Algorithme du Frequency Maximum Q-value pour un agent indépendant i dans un jeu matriciel d'équipe

début
  Initialiser $\mathsf{Q}_i(a_i)$, $\mathsf{Q}_{max,i}(a_i)$, $\mathsf{C}_i(a_i)$, $\mathsf{C}_{Q_{max},i}(a_i)$ et $\mathsf{E}_i(a_i)$ à 0, $\mathsf{F}_i(a_i)$ à 1 $\forall a_i \in \mathcal{A}_i$
  Initialiser $\pi_i(a_i)$ arbitrairement $\forall a_i \in \mathcal{A}_i$
  répéter
    Choisir l'action a_i selon la politique π_i
    Exécuter l'action a_i et observer la récompense r
    $\mathsf{C}_i(a_i) \leftarrow \mathsf{C}_i(a_i) + 1$ ▷ `compteur d'occurrences d'une action`
    $\mathsf{Q}_i(a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(a_i) + \alpha r$
    si $r > \mathsf{Q}_{max,i}(a_i)$ **alors**
      $\mathsf{Q}_{max,i}(a_i) \leftarrow r$ ▷ `mise à jour optimiste`
      $\mathsf{C}_{Q_{max},i}(a_i) \leftarrow 1$ ▷ `compteur d'occurrences de la récompense maximale pour une action`
    sinon si $r = \mathsf{Q}_{max,i}(a_i)$ **alors**
      $\mathsf{C}_{Q_{max},i}(a_i) \leftarrow \mathsf{C}_{Q_{max},i}(a_i) + 1$
    $\mathsf{F}_i(a_i) \leftarrow \frac{\mathsf{C}_{Q_{max},i}(a_i)}{\mathsf{C}_i(a_i)}$ ▷ `fréquence d'occurrences de la récompense maximale`
    $\mathsf{E}_i(a_i) \leftarrow \mathsf{Q}_i(a_i) + c\, \mathsf{F}_i(a_i)\, \mathsf{Q}_{max,i}(a_i)$ ▷ `heuristique`
    pour tous les $b \in \mathcal{A}_i$ **faire** $\pi_i(b) \leftarrow \frac{e^{\frac{\mathsf{E}_i(b)}{\tau}}}{\sum_{u \in \mathcal{A}_i} e^{\frac{\mathsf{E}_i(u)}{\tau}}}$
  jusqu'à *la fin des répétitions*
fin

Enfin, la sélection d'équilibres est réussie dans le jeu *penalty*, même si aucun mécanisme n'est explicitement utilisé pour cela. Comme dans le cas du Q-learning décentralisé, la stratégie d'exploration GLIE joue ce rôle implicitement.

6.4.3 Synthèse

Ainsi, concernant les facteurs de non-coordination, *le FMQ surmonte l'enjeu des équilibres cachés notamment grâce au calcul de la fréquence d'occurrence de la récompense maximale*. Cette fréquence permet aussi au FMQ de **faire la distinction entre le bruit dû à la non-coordination des agents et le bruit dû à l'environnement**, mais uniquement lorsque ce bruit est faible. *L'intérêt du FMQ est donc principalement situé dans le cadre stochastique.* Cet algorithme présente des limitations, les principales étant d'une part qu'il n'est proposé que pour des jeux matriciels, et d'autre part qu'il est peu robuste à l'exploration. En effet, une stratégie d'exploration GLIE doit être utilisée et le choix de ses paramètres est important. Une analyse détaillée de cet algorithme et de sa robustesse face à l'exploration est proposée dans le chapitre suivant.

6.5 Agents indulgents

Liviu Panait *et. al.* [SPBL06, PSL06] introduisent la notion d'agents indulgents dans leurs travaux sur les jeux matriciels coopératifs. Les agents indulgents sont en fait équivalents aux agents optimistes. Par contre, les auteurs s'intéressent à **varier le degré**

d'indulgence, ou d'optimisme, des agents. La stratégie d'exploration choisie est une stratégie GLIE. Elle consiste donc à explorer fortement au début de l'apprentissage, *i.e.* que les actions choisies par les agents sont quelconques. Il est donc justifié d'ignorer au début de l'apprentissage les pénalités reçues qui sont dues à un comportement d'exploration des agents. Cela permet aussi d'identifier les actions potentiellement prometteuses. Néanmoins, être optimiste peut aussi amener les agents à surestimer leurs actions et à fausser leurs évaluations dans les environnements stochastiques. Liviu Panait *et. al.* suggèrent donc de diminuer le degré d'indulgence des agents au fur et à mesure de leur apprentissage afin d'estimer la valeur réelle de leurs actions.

L'idée générale de l'algorithme est d'associer une température à chaque action de sorte que le degré d'indulgence soit inversement proportionnel à la température. Au début de l'apprentissage, les températures sont initialisées à de fortes valeurs et les agents sont optimistes. A chaque sélection d'une action, la température associée à cette action décroît et l'agent est alors plus enclin à incorporer les pénalités reçues dans l'évaluation de l'action.

Dans [SPBL06, PSL06], cet algorithme est comparé au Q-learning décentralisé et au FMQ sur les versions déterministes et stochastiques du jeu matriciel *Climbing* ainsi que sur le jeu *penalty*. Les résultats sont équivalents à ceux du FMQ pour un algorithme beaucoup plus difficile à utiliser car il nécessite le réglage de nombreux paramètres. En effet, il faut à la fois gérer la décroissance du degré d'indulgence et celle de la stratégie d'exploration. La robustesse de l'algorithme face à ces paramètres est de plus très faible.

6.6 WoLF-PHC

Michael Bowling [BV02, Bow03] propose deux algorithmes dans le cadre des jeux de Markov qui combinent le Q-learning décentralisé avec un ajustement de la politique courante issu des techniques de montée de gradient. Le premier algorithme qui utilise la montée de gradient et l'apprentissage par renforcement multiagent est l'*Infinitesimal Gradient Ascent* (IGA) [SKM00]. L'IGA a été conçu et étudié dans le cas des jeux matriciels à deux agents et deux actions et nécessite de plus que l'agent connaisse la politique courante de l'autre. Michael Bowling propose des adaptations « pratiques » de l'IGA pour le formalisme des jeux de Markov et pour des agents indépendants.

6.6.1 Le *Policy Hill Climbing*

Le premier algorithme est le *Policy Hill Climbing* (PHC) (*cf.* algorithme 7). **Le PHC est une simple modification du Q-learning décentralisé pour la recherche de politiques stochastiques.** Des tables individuelles Q_i sont mises à jour selon l'équation 6.1. La politique d'un agent est modifiée en augmentant la probabilité de choisir l'action gloutonne sur Q_i et en assurant la validité de la distribution de probabilité. Cette « mon-

Algorithme 7 : Algorithme du PHC pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s, a_i)$ à 0, $\pi_i(s, a_i)$ à $\frac{1}{|\mathcal{A}_i|}$ pour tout (s, a_i) de $\mathcal{S} \times \mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i selon la politique π_i avec une certaine exploration
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $\mathsf{Q}_i(s, a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(s, a_i) + \alpha \left[r + \gamma \max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s', b) \right]$ ▷ `Q-learning décentralisé`
    Sélectionner $a_{max} \in \arg\max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s, b)$ aléatoirement
    pour tous les $b \in \mathcal{A}_i$ ▷ `politique stochastique`
    faire
      si $b \neq a_{max}$ **alors**
        $\pi_i(s, b) \leftarrow -\delta_{sb}$
      sinon
        $\pi_i(s, b) \leftarrow \sum_{u \neq b} \delta_{su}$
      avec $\delta_{sb} = \min\left(\pi_i(s, b), \frac{\delta}{|\mathcal{A}_i| - 1}\right)$
    $s \leftarrow s'$
fin

tée de pente »[4] est réglée par un coefficient d'apprentissage $\delta \in]0; 1]$. Il est intéressant de noter que si $\delta = 1$, l'algorithme du PHC est équivalent au Q-learning décentralisé car les politiques associent une probabilité de 1 à l'action gloutonne dans chaque état. Les politiques construites sont alors gloutonnes sur Q_i et un agent suivant sa politique exécutera toujours l'action de plus forte Q-valeur dans chaque état.

L'algorithme PHC peut ne pas converger vers une politique stationnaire, comme le montrent Michael Bowling et Manuela Veloso [BV02].

6.6.2 L'heuristique WoLF

Une extension de PHC est donc développée par Michael Bowling et appelée WoLF-PHC. Le principe de l'heuristique WoLF, pour *Win or Learn Fast*, est d'utiliser un coefficient d'apprentissage variable pour l'ajustement de la politique. Deux coefficients δ_{perdre} et δ_{gagner} sont utilisés respectivement selon que l'agent perde ou gagne. Ceci est déterminé en comparant la valeur espérée de la politique courante, π, et la valeur espérée de la politique moyenne, $\bar{\pi}$. Si la valeur courante est supérieure, alors l'agent gagne ; sinon, il perd. Lorsque l'agent gagne, un coefficient d'apprentissage faible est utilisé pour ajuster sa politique courante de sorte à laisser du temps aux autres agents pour s'adapter à ce changement de politique. Lorsque l'agent perd, un coefficient d'apprentissage fort est utilisé afin qu'il puisse s'adapter rapidement aux changements de comportements des autres joueurs lorsque ces changements lui sont défavorables. On choisit donc $\delta_{perdre} > \delta_{gagner}$. Dans cet algorithme, la politique moyenne est la moyenne des politiques d'un

4. En anglais *hill climbing*.

Algorithme 8 : Algorithme du WoLF-PHC pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s,a_i)$ et $\mathsf{C}_i(s)$ à 0, $\pi_i(s,a_i)$ et $\bar{\pi}_i(s,a_i)$ à $\frac{1}{|\mathcal{A}_i|}$ pour tout (s,a_i) de $\mathcal{S}\times\mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i selon la politique π_i avec une certaine exploration
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $\mathsf{Q}_i(s,a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(s,a_i) + \alpha\left[r + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_i(s',b)\right]$ ▷ `Q-learning décentralisé`
    $\mathsf{C}_i(s) \leftarrow \mathsf{C}_i(s) + 1$ ▷ `compteur`
    $\bar{\pi}_i(s,b) \leftarrow \bar{\pi}_i(s,b) + \frac{1}{\mathsf{C}_i(s)}(\pi_i(s,b) - \bar{\pi}_i(s,b)) \quad \forall b \in \mathcal{A}_i$ ▷ `politique moyenne`
    si $\sum_{b\in\mathcal{A}_i} \pi_i(s,b)\mathsf{Q}_i(s,b) > \sum_{b\in\mathcal{A}_i} \bar{\pi}_i(s,b)\mathsf{Q}_i(s,b)$ **alors**
      $\delta \leftarrow \delta_{gagner}$ ▷ `ajustement du pas`
    sinon
      $\delta \leftarrow \delta_{perdre}$
    Sélectionner $a_{max} \in \arg\max_{b\in\mathcal{A}_i} \mathsf{Q}_i(s,b)$ aléatoirement
    pour tous les $b \in \mathcal{A}_i$
    faire
      si $b \neq a_{max}$ **alors**
        $\pi_i(s,b) \leftarrow \pi_i(s,b) - \delta_{sb}$ ▷ `ajustement de la`
      sinon
        $\pi_i(s,b) \leftarrow \pi_i(s,b) + \sum_{u\neq b} \delta_{su}$ `politique courante`
      avec $\delta_{sb} = \min\left(\pi_i(s,b), \frac{\delta}{|\mathcal{A}_i|-1}\right)$
    $s \leftarrow s'$
fin

joueur depuis le début de l'apprentissage. Cette politique moyenne est censée représenter la politique à suivre pour atteindre l'équilibre de Nash.

Michael Bowling a montré la convergence de cette heuristique WoLF appliqué à l'algorithme IGA (WoLF-IGA) vers un équilibre de Nash dans le cas où il y a deux joueurs à deux actions chacun dans un jeu à somme générale [BV01]. L'algorithme WoLF-PHC (*cf.* algorithme 8) est l'application de l'heuristique WoLF à l'algorithme PHC. Plusieurs variantes de cet algorithme existent : PD-WoLF [5] [BP03], GIGA-WoLF [Bow05] ou encore CoLF [6] [MLR06], pour la plupart destinés aux jeux à somme nulle.

La convergence de l'algorithme du WoLF-PHC a été observée en pratique par Michael Bowling dans des jeux matriciels et des jeux de Markov [BV02]. Les benchmarks utilisés sont des jeux à somme nulle et à somme générale, aucun d'entre eux ne présentant de facteurs de non-coordination. Pour étudier le comportement de cet algorithme face à ces difficultés, nous l'appliquons aux jeux matriciels des figures 5.3 à 5.7.

5. Pour *Policy Dynamics based WoLF*.
6. Pour *Change or Learn Fast*.

TABLE 6.5 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le WoLF-PHC (moyenne sur 200 épisodes indépendants). Un épisode consiste en 50000 répétitions. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale. Un taux d'exploration de 5% est choisi. Les paramètres sont $\alpha = 0,1$, $\delta_{gagner} = 5.10^{-5}$, $\delta_{perdre} = 2\delta_{gagner}$.

	Jeu simple	Jeu *penalty*			Jeu *Climbing*
		$k = 0$	$k = -10$	$k = -20$	déterministe
WoLF-PHC	100%	100%	82%	0,5%	0%

6.6.3 Résultats sur des jeux matriciels

La table 6.5 donne les résultats de convergence avec le WoLF-PHC pour différents jeux matriciels d'équipe. Pour le choix des coefficients d'apprentissage et du nombre de répétitions, nous prenons des valeurs similaires à celles utilisées par Michael Bowling pour ses expérimentations dans [BV02]. L'algorithme converge vers l'action jointe optimale dans le jeu simple ainsi que dans le jeu *penalty* avec de faibles pénalités. Il surmonte donc la difficulté de la sélection d'équilibres. Par contre, l'enjeu des équilibres cachés semble difficile à résoudre.

6.7 Le Q-learning « indicé »

L'idée principale de cet algorithme, proposé par [LR04], est de permettre à des agents indépendants de distinguer les différentes actions jointes mais de manière non-explicite. C'est ce que les auteurs appellent le principe de la **coordination implicite**.

Chaque agent i maintient pour chaque état s une liste $\mathsf{L}_i(s)$. Cette liste contient pour chaque état s un ensemble de n-uplets $\langle l, a_i, \mathsf{Q}\rangle$ (*cf.* figure 6.2), tel que :

- $l \in [1, l_{max}]$ est un indice correspondant à une action jointe. l_{max} est donc le nombre d'actions jointes,
- a_i est l'action individuelle de l'agent i appartenant à l'action jointe indicée par l,
- Q correspond à la valeur actuelle selon l'équation du Q-learning de l'indice l, c'est-à-dire à la valeur de l'action jointe (implicite) indicée par l dans l'état s.

Cette liste doit donc être construite avant l'apprentissage avec toutes les combinaisons possibles d'actions jointes. Si à chaque itération, tous les agents choisissent leur action individuelle a_i selon le même indice $\mathsf{L}_i(s).x$, alors la récompense reçue par chaque agent est $\mathsf{R}(s, \boldsymbol{a})$ où $\boldsymbol{a} = \langle a_1, ..., a_m\rangle$ est l'action jointe indicée par $\mathsf{L}_i(s).x$. Donc la valeur de Q pour cet indice est la valeur de cette action jointe dans l'état s. On peut alors montrer par récursivité que les listes des valeurs $\mathsf{L}_j(s).\mathsf{Q}$ sont identiques pour tous les agents j.

Chaque agent maintient donc les évaluations de toutes les actions jointes de manière implicite étant donnée que la différenciation se fait grâce aux indices. La condition nécessaire est que tous les agents choisissent le même indice à chaque itération.

$\mathsf{L}_i(s)$

l	a_i	Q_i
1		
2		
...	...	...
l_{max}		

FIGURE 6.2 – Liste de n-uplets dans l'état s pour un agent i.

Pour cela, une solution est d'utiliser le même générateur de nombre aléatoire pour tous les agents. Pour des raisons d'efficacité, les listes sont triées par valeurs décroissantes des indices. Ainsi, les actions gloutonnes sont en début de liste. Une méthode de sélection de l'indice peut donc être de choisir le plus souvent les indices en début de liste tout en conservant une part d'exploration.

L'algorithme de cette approche par « liste complète » et sa preuve théorique de convergence vers la politique jointe optimale dans les jeux de Markov d'équipe est disponible dans [LR04]. **L'intérêt majeur de cet algorithme est de converger dans les jeux de Markov stochastiques**, donc de faire la distinction entre le bruit dû à l'environnement et celui dû à différents comportements des autres agents. Ceci est possible grâce à la distinction des différentes actions jointes. Cependant, une limitation évidente est l'explosion combinatoire de la taille des listes. Une version par « liste réduite » est donc proposée par [LR04]. Elle utilise une taille limite pour les listes et réinitialise une partie de la liste après un certain nombre d'épisodes. Seuls les indices avec les meilleures évaluations sont donc conservés. Cet algorithme a été testé expérimentalement avec succès sur le jeu *Climbing* bruité. Malgré cela, son utilisation dans des jeux de Markov semble difficile. C'est pourquoi les auteurs proposent l'utilisation de fonctions d'approximation avec cet algorithme dans [GR06].

6.8 Bibliothèque d'outils pour l'apprentissage par renforcement

L'ensemble des expérimentations effectuées dans ce mémoire ont été réalisées avec une **bibliothèque d'outils pour Matlab-Simulink pour l'apprentissage par renforcement**, nommée **BOSAR**. Cette bibliothèque est sous licence GNU GPL et a été développée en C/C++ durant cette thèse. Elle est disponible en libre accès sur le site http ://www.lab.cnrs.fr/openblockslib/. C'est une bibliothèque de fonctions qui permet de développer et d'étudier les algorithmes d'apprentissage par renforcement. L'ensemble des algorithmes présentés dans ce mémoire ainsi que tous les *benchmarks* qui y sont utilisés sont implantés dans BOSAR et documentés dans une aide en ligne. Ces *benchmarks* sont aussi détaillés à l'annexe B.

TABLE 6.6 – Caractéristiques des algorithmes décentralisés d'apprentissage par renforcement pour agents indépendants dans les jeux d'équipe. « NT » signifie que la caractéristique n'a pas été testée.

	Jeux matriciels	Jeux de Markov	Sélection d'équilibres	Équilibres cachés	Environnement stochastique	Stratégie d'exploration
Q-Learning décentralisé [WD92]	✓	✓	✓ avec GLIE		✓ partiellement	GLIE
Q-Learning distribué [LR00]	✓	✓	✓	✓		stationnaire
Agents indulgents [PSL06]	✓		NT	NT	NT	GLIE
WoLF PHC [BV02]	✓	✓	✓		NT	stationnaire
FMQ [KK02]	✓		✓ avec GLIE	✓	✓ partiellement	GLIE
Q-learning indicé [LR04]	✓		✓	✓	✓	stationnaire

6.9 Conclusion

Dans ce chapitre a été proposé un état de l'art des algorithmes d'apprentissage par renforcement pour agents indépendants dans le cadre des jeux de Markov d'équipe. Les caractéristiques de ces algorithmes sont synthétisées à la table 6.6. Notamment sont précisés les facteurs de non coordination que chaque algorithme surmonte parmi les trois facteurs identifiés dans le chapitre précédent. De plus, la stratégie d'exploration utilisée avec chaque algorithme est précisée. En effet, un des enjeux de l'apprentissage d'agents indépendants est la robustesse face à l'exploration. Nous avons vu dans ce chapitre qu'une stratégie d'exploration GLIE est difficile à régler car elle nécessite le plus souvent une forte expertise du comportement de l'algorithme d'apprentissage. **La robustesse des algorithmes d'apprentissage utilisant une stratégie GLIE est donc faible**. De plus, une telle stratégie est aussi difficilement utilisable dans la pratique.

Au vu de cette synthèse, nous pouvons mettre en avant certains de ces algorithmes qui vont se révéler intéressants pour la suite de notre travail. Tout d'abord, le Q-learning distribué est le seul algorithme applicable dans les jeux de Markov d'équipe déterministes et qui possède une preuve de convergence vers un équilibre de Nash Pareto optimal dans ce cadre. Il est de plus robuste face à l'exploration. Son unique handicap est de ne pas surmonter la difficulté d'environnements stochastiques. Les deux algorithmes capables de maîtriser cet obstacle sont le Q-learning indicé et le FMQ. Le premier n'est pas applicable dans les jeux de Markov à cause d'une explosion combinatoire de la taille des listes

utilisées. Le second a quant à lui deux limitations principales : son manque de robustesse face à la stratégie d'exploration et son cadre d'application limité aux jeux matriciels. Ces deux limitations seront étudiées dans le chapitre suivant.

L'état de l'art et la notation uniforme qui ont été proposés dans ce chapitre ont aussi permis de mettre en avant de nombreux points communs entre certains algorithmes, et notamment entre le Q-learning distribué et le FMQ. Nous nous intéressons donc dans le chapitre suivant à ces deux algorithmes, et plus particulièrement au FMQ, *dont l'intérêt majeur est d'être capable de faire la distinction entre le bruit dû aux récompenses stochastiques et le bruit dû aux comportements des autres agents dans des jeux partiellement bruités.*

Chapitre 7

SOoN et Q-learning hystérétique

Ce chapitre décrit deux algorithmes d'apprentissage par renforcement pour agents indépendants développés dans le cadre des jeux de Markov d'équipe : le Q-learning hystérétique *et le* Swing between Optimistic or Neutral *(SOoN). La démarche de développement de ces deux méthodes est détaillée, notamment à travers l'étude du* Frequency Maximum Q-value *et l'étude de la robustesse de ces algorithmes face aux différents enjeux de l'apprentissage. Les résultats expérimentaux obtenus sur divers jeux de Markov sont également présentés.*

7.1 Introduction

Ce chapitre est consacré à l'étude de nouveaux algorithmes d'apprentissage par renforcement multiagents situés dans le cadre des jeux de Markov d'équipe. Les hypothèses retenues dans ce chapitre sont que les agents sont indépendants, ont une observabilité totale et doivent apprendre à se coordonner. Pour cela, ils doivent notamment faire face à différents enjeux de l'apprentissage présentés précédemment dans ce mémoire (*cf.* chapitre 5). Parmi ces difficultés, nous nous intéressons aux facteurs de non-coordination et à la robustesse face à l'exploration. Les environnements stochastiques sont particulièrement étudiés dans ce chapitre. *L'objectif est donc de trouver des algorithmes robustes face à l'exploration et capables de surmonter les facteurs de non-coordination dans les jeux de Markov d'équipe.*

L'état de l'art proposé au chapitre précédent a mis en évidence deux algorithmes : le Q-learning distribué qui possède une preuve de convergence vers un équilibre de Nash Pareto optimal dans tous jeux de Markov d'équipe déterministes, et l'algorithme du *Frequency Maximum Q-value* (FMQ), capable de découpler le bruit dû à la non-coordination des agents du bruit dû à l'environnement dans les jeux matriciels. Cet algorithme propose donc une technique intéressante de détection de la stochasticité d'un jeu. Afin d'atteindre notre objectif, nous nous intéressons à ces deux algorithmes qui présentent de nombreux points communs et dont la réunion des caractéristiques de chacun va permettre de faire

face à différents enjeux de l'apprentissage.

Deux algorithmes sont proposés dans ce chapitre. Le premier repose sur une extension du Q-learning décentralisé. La modification apportée consiste à utiliser deux vitesses d'apprentissage de sorte à obtenir des agents « à forte tendance optimiste », appelés **agents hystérétiques**. Le second algorithme est issu d'une étude de l'algorithme du FMQ. En effet, celui-ci présente diverses limitations qui doivent être améliorées afin d'envisager une extension au cadre multi-état. Cette étude aboutit au développement d'une version modifiée du FMQ plus robuste à l'exploration. Elle associe une nouvelle heuristique à une fréquence détectant la stochasticité du jeu. Grâce à ses modifications, une extension aux jeux de Markov est proposée. L'étude de cette nouvelle heuristique et du calcul adapté de la fréquence selon le cadre choisi est l'élément fondamental de l'algorithme proposé. Ces deux algorithmes sont testés sur différentes applications multiagents.

7.2 Le Q-learning hystérétique

Dans cette section est présenté un algorithme basé sur le Q-Learning décentralisé et sur des agents « à forte tendance optimiste ». A travers cet algorithme, nous recherchons d'une part à améliorer les performances du Q-learning décentralisé, qui présente des difficultés dans certains jeux où les équilibres optimaux sont cachés. D'autre part, nous voulons aussi surmonter l'incapacité des agents optimistes à se coordonner sur l'équilibre optimal dans des jeux stochastiques.

7.2.1 Agents indépendants hystérétiques

Comme nous l'avons vu précédemment, la fonction de récompense dans un jeu de Markov dépend des actions choisies par tous les agents. Ainsi, un agent peut parfois être puni à cause d'un mauvais choix d'un de ses congénères, et ce, même si l'action individuelle qu'il a choisie est optimale. Le mauvais choix d'un congénère peut être dû à une action d'exploration ou à une non-coordination des agents. Prenons pour exemple le jeu *Climbing*. Si l'agent 1 joue l'action optimale a mais que l'agent 2 explore ou n'est pas coordonné et joue l'action b, la récompense résultant de l'action jointe $\langle a, b\rangle$ est une pénalité de -30. L'agent 1 est donc pénalisé même s'il a joué son action individuelle optimale. Ces pénalités de non-coordination font partie des facteurs responsables des difficultés de convergence du Q-learning décentralisé (*cf.* §6.2.2).

Partant de ce fait, il peut donc être préférable pour un agent d'ignorer ces pénalités. C'est ici l'idée des agents indépendants optimistes qui ignorent dans leur équation de mise à jour les pénalités reçues qui sont souvent dues à une non-coordination des agents. Ces agents ont été présentés au §6.3. En étant aveugles aux pénalités, ils évaluent une action jointe par sa valeur maximale et non pas par sa valeur moyenne. Il a été démontré que tout agent optimiste est capable de déterminer la fonction de valeur locale

Algorithme 9 : Algorithme du Q-learning hystérétique pour un agent indépendant i dans un jeu de Markov d'équipe.

début
  Initialiser $\mathsf{Q}_i(s, a_i)$ arbitrairement pour tout (s, a_i) de $\mathcal{S} \times \mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i ▷ étape de décision
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $q \leftarrow r + \gamma \max_{b \in \mathcal{A}_i} \mathsf{Q}_i(s', b)$
    ▷ mise à jour hystérétique
    si $q \geq \mathsf{Q}_i(s, a_i)$ **alors**
      $\mathsf{Q}_i(s, a_i) \leftarrow (1 - \alpha)\mathsf{Q}_i(s, a_i) + \alpha q$
    sinon
      $\mathsf{Q}_i(s, a_i) \leftarrow (1 - \beta)\mathsf{Q}_i(s, a_i) + \beta q$
    si $\mathsf{Q}_i(s, \arg\max_{u \in A_i} \pi_i(s, u)) \neq \max_{u \in A_i} \mathsf{Q}_i(s, u)$ **alors**
      Choisir $a_{max} \in \arg\max_{u \in A_i} \mathsf{Q}_i(s, u)$ aléatoirement
      $\forall b \in A_i \quad \pi_i(s, b) \leftarrow \begin{cases} 1 & \text{si } b = a_{max} \\ 0 & \text{sinon} \end{cases}$ ▷ sélection d'équilibres
    $s \leftarrow s'$
fin

optimale dans un jeu de Markov d'équipe déterministe. Cependant, dans des environnements stochastiques, ces agents ne parviennent pas à se coordonner. En effet, dans ces environnements stochastiques, une pénalité n'est pas toujours due à un mauvais comportement d'un autre agent mais peut aussi être le résultat d'une fonction de récompense bruitée.

Pour remédier à cette limitation, nous proposons d'utiliser des agents « à forte tendance optimiste », *i.e.* qui accordent peu d'importance aux pénalités reçues mais ne sont pas totalement aveugles face à elles. Ces agents sont appelés des **agents hystérétiques**. Ces agents accordent peu d'importance à une pénalité reçue après avoir choisi une action leur ayant amenée satisfaction dans le passé, au cas où cette pénalité serait due à un mauvais comportement d'un ou plusieurs de ses congénères. Cependant, cette pénalité n'est pas totalement ignorée car elle pourrait être la conséquence d'une récompense bruitée.

7.2.2 Q-learning hystérétique

Afin de mettre en oeuvre ces agents hystérétiques, deux coefficients d'apprentissage sont utilisés pour la croissance et la décroissance des Q_i-valeurs, de sorte que la courbe de croissance ne se superpose pas à la courbe de la décroissance. Ce décalage entre les deux courbes est à l'origine du nom donné à ces agents. L'agent hystérétique apprend donc « à deux vitesses » selon qu'il soit récompensé ou puni.

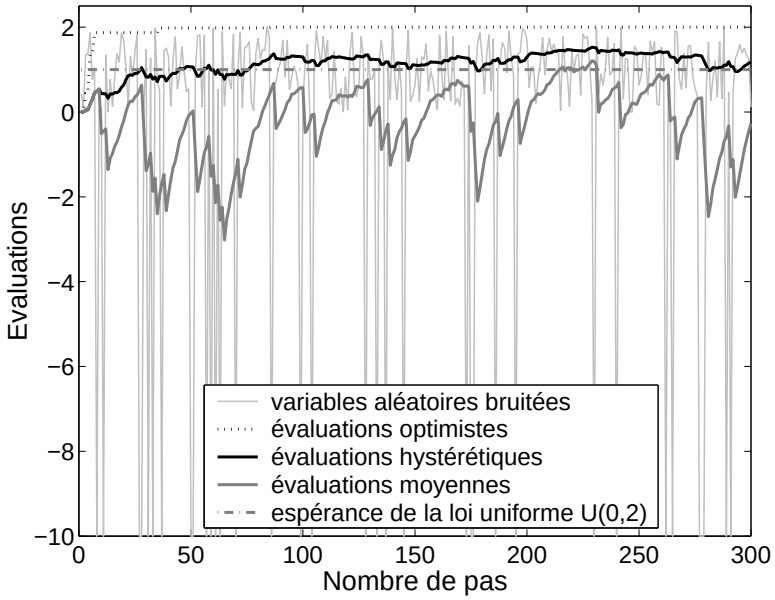


FIGURE 7.1 – Différentes évaluations de l'espérance de la loi uniforme « dénaturée » dans $[0;2]$ avec $\alpha = 0,1$.

Le Q-learning hystérétique pour agents indépendants dans les jeux de Markov d'équipe est donné à l'algorithme 9. α et β sont respectivement les coefficients de croissance et de décroissance des Q-valeurs ($(\alpha, \beta) \in [0;1]^2$). Les agents hystérétiques sont « à forte tendance optimiste » donc $\alpha \geq \beta$. Plus le rapport $\frac{\alpha}{\beta}$ est grand et plus la tendance optimiste des agents sera forte. Si $\alpha = \beta$, le Q-learning hystérétique est équivalent au Q-learning décentralisé. Si $\alpha = 1$ et $\beta = 0$, il est équivalent au Q-learning distribué.

Le principe de la mise à jour est le suivant. Si une action a est effectuée dans l'état s et que la transition immédiate entraîne une amélioration ou une stagnation de la valeur courante $\mathsf{Q}(s,a)$, alors la valeur de ce couple va être mise à jour avec le coefficient de croissance α. Par contre, si la transition entraîne une diminution de la valeur courante $\mathsf{Q}(s,a)$, la valeur de ce couple va être mise à jour avec le coefficient de décroissance β. Un mécanisme de sélection d'équilibres similaire à celui utilisé par [LR00] peut être utilisé. Dans ce cas, la méthode de décision ϵ-greedy basée sur π_i est suivie. Si la méthode de décision *softmax* est choisie, ce mécanisme de sélection d'équilibres ne peut pas être suivi. Une stratégie d'exploration GLIE [1] correctement choisie pourra alors jouer implicitement un rôle de sélection d'équilibres.

Afin d'illustrer le fonctionnement de cet algorithme et le principe des agents hystérétiques, nous choisissons un exemple simple qui met en défaut les agents optimistes et les agents effectuant une évaluation moyenne. Soit une suite de n variables aléatoires

1. En anglais *greedy in the limit with infinite exploration*.

indépendantes $U_1, U_2, ..., U_n$ de loi uniforme dans $[0; 2]$. Cette suite est « dénaturée », c'est-à-dire qu'avec une probabilité de $0,1$, le nombre généré est en fait une pénalité égale à -10. Le signal généré par les variables aléatoires représente une récompense stochastique perçue par un agent indépendant. Les pénalités illustrent quant à elles un mauvais comportement d'un congénère. On peut donc considérer que ce signal est similaire à ce que recevrait un agent indépendant pour un couple état-action, dans le cas d'une récompense stochastique et de mauvais comportements de la part de ses congénères. L'objectif de l'agent indépendant est d'évaluer l'espérance de la récompense tout en ignorant les pénalités reçues car ce n'est pas son comportement qui en est responsable.

On cherche donc à évaluer l'espérance de cette loi uniforme dans $[0; 2]$ malgré les pénalités qui « dénaturent » le signal. A chaque période d'échantillonnage t, une variable U_t est reçue et la nouvelle estimation de l'espérance, notée X_t, est calculée selon la mise à jour suivante :

$$\begin{aligned} \text{Si} \quad & U_t \geq X_{t-1} \\ & X_t \leftarrow (1-\alpha)X_{t-1} + \alpha U_t \\ \text{Sinon} \quad & \\ & X_t \leftarrow (1-\beta)X_{t-1} + \beta U_t \end{aligned}$$

où α et β sont respectivement les coefficients de croissance et de décroissance de l'évaluation. On retrouve donc ici la loi de mise à jour hystérétique. Différentes évaluations sont comparées :

- une évaluation optimiste avec $\alpha = 1$ et $\beta = 0$,
- une évaluation moyenne avec $\beta = \alpha$,
- une évaluation hystérétique avec $\beta = \frac{\alpha}{10}$.

Les résultats sont donnés à la figure 7.1. Précisons tout d'abord qu'une évaluation moyenne estime très bien l'espérance si celle-ci n'est pas dénaturée par des pénalités. Par contre, avec les pénalités, l'espérance est sous-estimée par une évaluation moyenne car cette évaluation accorde autant d'importance aux pénalités reçues qu'à la dispersion des variables aléatoires. Concernant l'évaluation optimiste, elle est mise en défaut car elle n'est pas adaptée aux environnements stochastiques (*cf.* §6.3). Elle surestime la valeur de l'espérance. **La valeur réelle de l'espérance se situe donc entre son évaluation optimiste et son évaluation moyenne**. L'évaluation hystérétique évalue quant à elle de manière assez fiable l'espérance. L'espérance n'est pas sous-estimée car les pénalités sont peu prises en compte. Elle n'est pas trop sur-estimée car la dispersion des variables aléatoires est tout de même considérée. *L'estimation faite par un agent hystérétique se situe donc entre les évaluations optimistes et moyennes, et est donc assez proche de l'évaluation réelle.*

7.2.3 Choix de β et exploration globale

Exploration globale

Lors de notre étude sur les enjeux de l'apprentissage, nous avons évoqué le phénomène d'exploration concurrente, c'est-à-dire le bruit perçu par un agent indépendant et causé par l'exploration des autres agents (*cf.* §5.4.4). Afin de quantifier ce bruit dû à l'exploration, une notion importante dans les systèmes multiagents est celle d'**exploration globale**. L'exploration globale peut être définie comme la probabilité qu'au moins un des agents du système explore. Elle peut être exprimée en fonction de l'exploration individuelle de chaque agent. L'exploration individuelle d'un agent est la probabilité que l'agent explore. Elle correspond par exemple à la valeur du paramètre ϵ avec la méthode de décision ϵ-greedy.

Propriété 7.1 *Soit un système à n agents dans lequel chaque agent explore avec une probabilité ϵ. Alors, la probabilité qu'au moins l'un d'entre eux explore est $\psi = 1-(1-\epsilon)^n$. ψ est appelé l'exploration globale.*

Cette exploration globale ne doit pas dépasser une certaine valeur afin d'éviter un bruit trop important. Typiquement, nous suggérons de limiter l'exploration globale à une valeur limite de $0,2$.

Choix de β

Le rôle du coefficient d'apprentissage β est de prendre en compte la diminution de la valeur courante d'une action au cas où la récompense serait bruitée, mais ce coefficient doit rester assez faible car cette diminution peut aussi être causée par de mauvais comportements des congénères. Notamment, lorsqu'un des autres agents explore, cela peut entraîner une pénalité que l'agent ne doit pas considérer. Donc, plus l'exploration des autres agents est élevée, plus le bruit dû à leurs comportements est important, et donc plus l'agent doit être optimiste. Si une stratégie d'exploration stationnaire est choisie avec le Q-learning hystérétique, on peut alors préconiser de choisir un rapport $\frac{\alpha}{\beta}$ d'autant plus grand que l'exploration globale sera élevée. Typiquement et après expérimentations, un rapport de $\frac{\alpha}{\beta} = 10$ est conseillé pour une exploration globale de $\psi = 0,1$. Cependant, une étude plus approfondie du choix de ce paramètre pourrait être envisagée.

7.2.4 Résultats sur des jeux matriciels

Nous testons le Q-learning hystérétique sur les jeux matriciels des figures 5.3 à 5.7. La stratégie d'exploration choisie est GLIE. Les résultats sont donnés à la table 7.1. Si la stratégie d'exploration est bien réglée, le Q-learning hystérétique converge vers l'une des actions jointes optimales dans les jeux déterministes. Ses résultats sont donc équivalents à ceux du Q-learning distribué et dépassent ceux du Q-learning décentralisé, qui ne converge dans le jeu *Climbing* déterministe que dans 3% des cas.

Dans le jeu *Climbing* partiellement bruité, ses résultats sont supérieurs à ceux obtenus avec le Q-learning décentralisé ou le Q-learning distribué. 82% des épisodes convergent

TABLE 7.1 – Pourcentage d'épisodes convergeant vers l'action jointe optimale avec le Q-learning hystérétique (moyenne sur 100 épisodes indépendants). Un épisode consiste en 3000 répétitions. La stratégie d'exploration choisie est GLIE. On pose $\alpha = 0,1$, $\beta = 0,01$ et $\tau = 5000e^{-0,003t}$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale (x% de convergence vers un équilibre de Nash Pareto optimal) ou sous-optimale (y% de convergence vers un équilibre de Nash sous-optimal). Les résultats sont notés x% [y%].

	Jeu simple	Jeu *Penalty* ($k = -100$)	Jeu *Climbing*		
			déterministe	partiellement bruité	bruité
Q-learning hystérétique	100%	100%	100%	82%[18%]	0%[100%]

vers l'équilibre de Nash Pareto optimal et 18% vers l'équilibre de Nash sous-optimal. Les évaluations hystérétiques semblent donc plus proches des valeurs réelles.

Par contre, dans le jeu *Climbing* bruité, le Q-learning hystérétique converge à 100% vers l'équilibre de Nash sous-optimal.

Le Q-learning hystérétique surpasse donc le Q-learning décentralisé en déjouant la difficulté d'équilibres cachés grâce à ses deux vitesses d'apprentissage. De plus, dans les jeux partiellement bruités, le fait de ne pas ignorer totalement les pénalités permet aux agents hystérétiques d'obtenir de meilleurs résultats que les agents optimistes.

D'autres tests du Q-learning hystérétique seront effectués sur divers jeux de Markov multiagents à la section 7.6.

7.2.5 Conclusion

Grâce à deux coefficients d'apprentissage, l'algorithme du Q-learning hystérétique donne la possibilité d'influencer différemment les vitesses de croissance et de décroissance des valeurs de Q_i. En d'autres termes, il procure l'avantage de pouvoir « doser » l'optimisme des agents. De cette manière, le Q-learning hystérétique peut surmonter la difficulté d'équilibres cachés. Il réussit aussi la coordination d'agents indépendants dans certains jeux partiellement bruités.

Cependant, le Q-learning hystérétique, de même que le Q-learning décentralisé, est sensible à la stratégie d'exploration choisie, qui influe fortement sur la convergence. Dans les jeux matriciels, la convergence de l'algorithme dépend d'une stratégie d'exploration GLIE bien choisie. Les meilleurs résultats dans les jeux matriciels ont été obtenus avec une stratégie GLIE. Le choix du paramètre β doit aussi être fait en fonction de l'exploration globale. Or, avec une stratégie GLIE, ce choix peut s'avérer complexe alors qu'il pourrait être précisé de manière simple avec une stratégie stationnaire.

Cet algorithme ne satisfait donc pas à l'objectif de robustesse fixé au départ. *La robustesse de cet algorithme face à l'exploration doit donc être améliorée.*

Malgré ses limitations et les améliorations qu'il reste à apporter à cet algorithme, le principe des agent hystérétiques est une manière simple de mettre en oeuvre une évaluation qui se situe entre l'optimiste et la moyenne. Les résultats obtenus sur les jeux matriciels ainsi que sur les jeux de Markov (*cf.* §7.6) confirment la pertinence de cette évaluation hystérétique. Le principe des agents hystérétiques pourrait aussi être étendu à d'autres algorithmes décentralisés d'apprentissage par renforcement. Particulièrement, il pourrait être appliqué à des versions décentralisées des algorithmes TD(λ).

7.3 Étude du *Frequency Maximum Q-value*

Au regard de l'état de l'art, l'intérêt principal du *Frequency Maximum Q-value* (FMQ), présenté au chapitre précédent (*cf.* §6.4), réside dans sa capacité à surmonter la difficulté de récompenses faiblement bruitées dans les jeux matriciels. Néanmoins, une de ses limitations est son manque de robustesse face à l'exploration, qui est un des enjeux de l'apprentissage d'agents indépendants. Cette limitation doit notamment être surmontée dans la perspective d'une application au cadre multi-état. Nous analysons donc ici **l'influence de la stratégie d'exploration et du choix de certains paramètres sur la convergence du FMQ.**

7.3.1 Amélioration de la robustesse face à la stratégie d'exploration

Lien entre la convergence du *Frequency Maximum Q-value* et l'exploration

Spiros Kapetanakis et Daniel Kudenko [KK02] utilisent avec le *Frequency Maximum Q-value* (FMQ) la méthode de décision softmax associée à une stratégie d'exploration GLIE. Plus spécifiquement, pour la décroissance de l'exploration associée à cette stratégie GLIE, ils proposent la fonction de température exponentielle suivante :

$$\tau_k = \tau_{max} e^{-\delta k} + \tau_\infty \tag{7.1}$$

où k est le nombre de répétitions du jeu matriciel, δ contrôle la décroissance de l'exponentielle, et τ_{max} et τ_∞ permettent de régler les valeurs de température en début et en fin des répétitions. Comme nous l'avons vu dans les chapitres précédents, l'utilisation d'une telle stratégie d'exploration implique de choisir correctement l'ensemble des paramètres de décroissance. Les résultats de la table 7.2 illustrent la difficulté du choix de ces paramètres. Le changement d'un seul des paramètres de la stratégie d'exploration peut faire fortement chuter le taux de réussite du FMQ dans un même jeu. Ainsi, **le FMQ est très peu robuste à la stratégie d'exploration**, comme cela est aussi précisé dans le cas des systèmes multiagents coopératifs hétérogènes [KK04].

Pour étudier ce lien entre la convergence du FMQ et l'exploration, nous traçons à la figure 7.2 une fonction de température exponentielle typiquement utilisée avec le FMQ.

TABLE 7.2 – Pourcentage d'épisodes convergeant vers l'action jointe optimale dans le jeu *Climbing* avec la méthode sofmax et selon différentes stratégies d'exploration avec le FMQ (moyenne sur 500 épisodes indépendants). Un épisode consiste en 5000 répétitions et on pose $\alpha = 0,1$ et $c = 10$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

			FMQ
Stratégie d'exploration	GLIE	$\tau = 499e^{-0,006t} + 1$	100%
		$\tau = 100e^{-0,006t} + 1$	59%
		$\tau = 499e^{-0,03t} + 1$	86%
		$\tau = 100 \times 0,997^t$	70%
	Stationnaire	$\tau = 20$	23%

En parallèle de cette courbe, on trace aussi la moyenne des récompenses reçues par des agents indépendants apprenant dans le jeu *Climbing* avec le FMQ et une méthode d'exploration softmax suivant cette fonction de température. Deux phases peuvent être identifiées lors de cet apprentissage. La première phase est une phase d'**exploration** pendant laquelle la valeur de la température est élevée. La moyenne des récompenses est alors constante car les agents choisissent toutes les actions jointes possibles de manière quasiment équiprobable. La deuxième phase est constituée de la décroissance de la température. Les agents apprennent alors à se coordonner jusqu'à ce que la température atteigne sa valeur limite. Les agents suivent alors leurs actions gloutonnes qui sont ici optimales car la moyenne des récompenses reçues est 11, ce qui correspond à l'équilibre de Nash Pareto optimal dans le jeu *Climbing*. Cette phase de décroissance de l'exploration est appelée la phase de **coordination**.

Ces deux phases successives d'exploration et de coordination sont nécessaires à la convergence du FMQ. En effet, si ces deux phases sont mal réglées, le pourcentage d'épisodes convergeant vers l'action jointe optimale peut diminuer. Par exemple, si l'on choisit une stratégie d'exploration stationnaire où ces deux phases sont alors inexistantes, le taux de réussite du FMQ diminue fortement (*cf.* table 7.2). Ainsi, **le FMQ est peu robuste à l'exploration**, dans le sens où l'apprentissage d'une politique ne peut pas être simultanée à l'exploration car deux phases successives d'exploration et de coordination sont nécessaires, contrairement au Q-learning distribué par exemple (*cf.* §6.3.2).

Instabilité de la fréquence face à l'exploration

Ce manque de robustesse du FMQ face à l'exploration et donc ce lien entre la convergence et l'exploration est dû à une instabilité de la fréquence d'occurrence F *calculée par le FMQ.* Afin d'illustrer ce phénomène, reprenons le cas du jeu *Climbing*. Nous choisissons d'étudier *l'instabilité de la fréquence d'occurrence de la récompense maximale associée à l'action* a *pour un agent,* i.e. $\mathsf{F}_1(a)$*, face à l'exploration de l'autre agent.* Nous avons choisi l'action a car c'est l'action individuelle optimale. La fréquence d'occurrence de la

récompense maximale associée à l'action a doit donc tendre vers 1.

Nous supposons que les agents sont en début d'une phase d'apprentissage. La fonction de fréquence $\mathsf{F}_1(a)$ est donc initialisée à 1. On trace l'évolution de la fréquence d'occurrence de la récompense maximale associée à l'action a pour l'agent 1, c'est-à-dire $\mathsf{F}_1(a)$ (*cf.* figure 7.3 en courbe pointillée). En haut de cette figure, un diagramme donne les actions successivement choisies par les agents. Précisons que les agents ne suivent pas leurs Q-valeurs mais une politique « manuelle » choisie pour l'exemple. Sur l'ensemble de l'épisode, l'agent 1 choisit toujours l'action a.

Au début de l'apprentissage, l'action jointe $\langle a, c\rangle$ est jouée. La récompense reçue est donc 0 et les compteurs d'occurrences de l'action $\mathsf{C}_1(a)$ et de la récompense maximale reçue $\mathsf{C}_{\mathsf{Q}_{max},1}(a)$ s'incrémentent du nombre de répétitions effectuées. Lorsque l'action jointe optimale $\langle a, a\rangle$ est jouée pour la première fois (suite à une action d'exploration de l'agent 2), une nouvelle récompense maximale est reçue donc :

- la récompense maximale reçue jusqu'alors est modifiée, $\mathsf{Q}_{max,1}(a) \leftarrow 11$,
- le compteur d'occurrences de la récompense est réinitialisé, $\mathsf{C}_{\mathsf{Q}_{max},1}(a) \leftarrow 1$,
- le compteur d'occurrences de l'action $\mathsf{C}_1(a)$ est simplement incrémenté.

Alors, la valeur de la fréquence mise à jour selon $\mathsf{F}_1(a) \leftarrow \frac{\mathsf{C}_{\mathsf{Q}_{max},1}(a)}{\mathsf{C}_1(a)} = \frac{1}{\mathsf{C}_1(a)}$ diminue fortement, comme illustrée à la $15^{\text{éme}}$ répétition (courbe pointillée). La conséquence de cette chute de la valeur de la fréquence est que l'évaluation de l'action $\mathsf{E}_1(a)$ peut alors être inférieure aux évaluations des autres actions. Dans ce cas, l'action optimale a est alors choisie uniquement lors de décisions d'exploration. L'algorithme peut alors ne pas converger sur l'action jointe optimale et de nombreuses décisions d'exploration sont alors nécessaires pour que la valeur de la fréquence de l'action optimale $\mathsf{F}_1(a)$ augmente assez pour permettre la coordination sur l'action jointe optimale.

Il faut noter que plus l'action jointe optimale est jouée la première fois après un grand nombre de répétitions, plus la chute de la valeur du compteur $\mathsf{C}_1(a)$ est importante, et donc plus le nombre de décisions d'exploration nécessaires à la coordination est élevé. Pour éviter ce phénomène, il faut que l'action jointe optimale soit jouée le plus tôt possible. C'est pourquoi la première phase d'exploration pendant laquelle toutes les actions jointes sont choisies est importante dans le FMQ. *Si cette première phase d'exploration est mal réglée, la fréquence est alors instable face à l'exploration.*

Calcul récursif de la fréquence

Afin d'obtenir une fréquence plus robuste face à l'exploration, le compteur d'occurrences $\mathsf{C}(a)$ d'une action a doit être réinitialisé à 1 quand une nouvelle récompense maximale est reçue pour cette action. Ainsi, la fréquence F est aussi remise à 1, comme illustrée à la $15^{\text{éme}}$ répétition sur la courbe (ligne continue) de la figure 7.3. Contrairement au cas précédent, la valeur de la fréquence diminue alors seulement en cas de non-coordination des agents. Cette non-coordination peut être due soit à une action d'exploration d'un des agents, soit à une récompense bruitée. La fréquence est alors

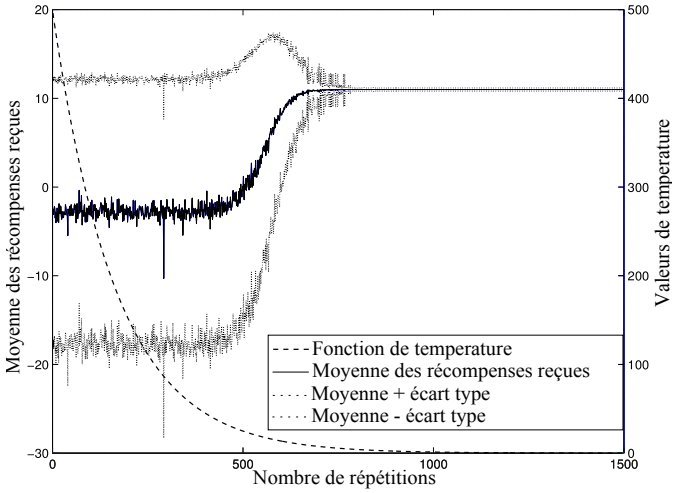


FIGURE 7.2 – Moyenne des récompenses reçues dans le jeu *Climbing* avec des agents apprenant selon le FMQ : $c = 10$ et $\tau = 499e^{-0.006k} + 1$.

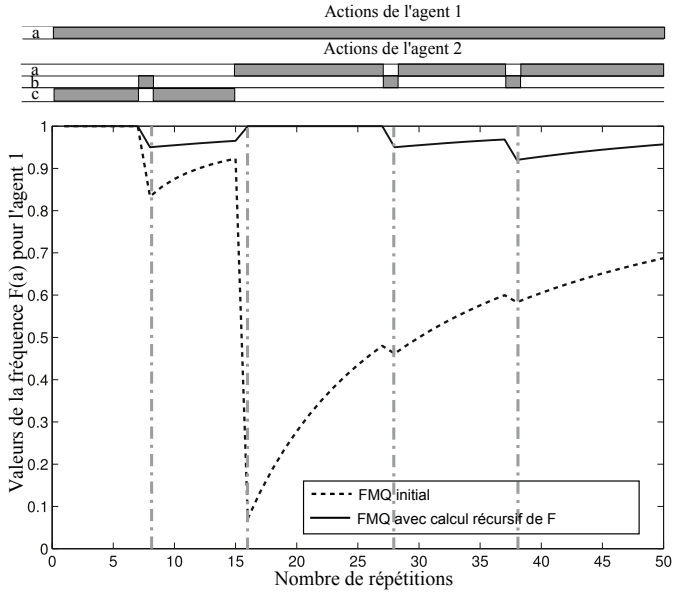


FIGURE 7.3 – Valeurs de la fréquence d'occurrence de la récompense maximale associée à l'action a pour l'agent 1 *vs.* le nombre de répétitions du jeu *Climbing*. L'algorithme utilisé est le FMQ. Les actions des deux agents suivent la politique « manuelle » choisie pour l'exemple et elles sont données dans le diagramme au-dessus de la courbe.

TABLE 7.3 – Pourcentage d'épisodes convergeant vers l'action jointe optimale dans le jeu *Climbing* selon différentes stratégies d'exploration avec le FMQ initial et le FMQ modifié (réinitialisation et calcul récursif de F) (moyenne sur 500 épisodes indépendants). Un épisode consiste en 5000 répétitions et on pose $\alpha = 0,1$, $\alpha_f = 0,05$ et $c = 10$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

			FMQ initial	FMQ avec modifications
Stratégie d'exploration	GLIE	$\tau = 499e^{-0,006^t} + 1$	100%	100%
		$\tau = 100e^{-0,006^t} + 1$	59%	100%
		$\tau = 499e^{-0,03^t} + 1$	86%	100%
		$\tau = 100 \times 0,997^t$	70%	100%
	Stationnaire	$\tau = 20$	23%	100%

moins instable face à l'exploration.

L'utilisation de compteurs incrémentaux C et $\mathsf{C}_{\mathsf{Q}_{max}}$ peut aussi être un facteur de dépendance à l'ancienneté. Nous appelons dépendance à l'ancienneté le fait que l'importance de la chute ou de la montée de la fréquence dépend du nombre de répétitions après lequel une action d'exploration ou une action gloutonne est jouée. Pour diminuer cette dépendance, nous introduisons donc le **calcul récursif de la fréquence d'occurrence** de la récompense maximale associée à une action a suivant :

$$\mathsf{F}(a) \leftarrow \begin{cases} (1-\alpha_f)\mathsf{F}(a) + \alpha_f & \text{si } r = \mathsf{Q}_{max}(a) \\ (1-\alpha_f)\mathsf{F}(a) & \text{sinon} \end{cases} \tag{7.2}$$

où $\alpha_f \in [0;1]$ est le coefficient d'apprentissage de la fréquence. On obtient alors une fréquence plus robuste face à l'exploration (*cf.* figure 7.3 ligne continue). Le choix du coefficient α_f sera discuté au §7.4.2.

Résultats sur un jeu matriciel

Nous vérifions maintenant la robustesse du FMQ lorsqu'il est modifié avec ces deux améliorations : une réinitialisation et un calcul récursif de la fréquence F. Pour cela, nous prenons le jeu matriciel *Climbing*. Nous testons le FMQ initial (*cf.* algorithme 6) et le FMQ modifié avec diverses stratégies d'exploration. Les résultats sont donnés dans la table 7.3.

On constate que contrairement au FMQ initial qui ne converge ici qu'avec une stratégie sur les cinq testées, le FMQ modifié obtient 100% de réussite avec toutes les stratégies testées. Notamment, il est intéressant de remarquer que les modifications effectuées permettent de s'affranchir d'une stratégie GLIE. Une stratégie stationnaire, qui est plus simple à régler, peut être choisie.

Ainsi, les modifications apportées au FMQ permettent d'obtenir un algorithme plus robuste à la stratégie d'exploration choisie. Cet algorithme modifié converge vers l'action jointe optimale dans le jeu *Climbing* avec une stratégie GLIE et avec une stratégie stationnaire, contrairement au FMQ initial. Comme nous l'avons vu dans les chapitres précédents, *une stratégie stationnaire apporte les avantages d'avoir moins de paramètres à régler et plus de stabilité face à l'exploration.* Elle nécessite uniquement un nombre de répétitions suffisant pour assurer une exploration complète de l'ensemble des actions jointes.

7.3.2 Nouvelle heuristique pour l'évaluation de l'action

Une seconde limitation du FMQ est l'utilisation d'un paramètre c qui pondère l'importance de l'heuristique dans l'évaluation des actions :

$$\mathsf{E}(a) = \mathsf{Q}(a) + c\ \mathsf{F}(a)\ \mathsf{Q}_{max}(a). \tag{7.3}$$

La convergence du FMQ dépend du choix de ce paramètre de pondération c [KK02]. De plus, aucune justification n'est donnée concernant le choix de cette heuristique pour évaluer les actions. En effet, quelles sont les justifications théoriques pour utiliser la somme des valeurs de Q et Q_{max} ? Nous proposons donc ici une autre heuristique. Elle présente d'une part l'avantage de ne plus avoir de paramètre de pondération à régler. D'autre part, une justification qualitative de la convergence de cette heuristique dans les jeux matriciels déterministes est proposée.

Nous rappelons tout d'abord le point commun entre l'algorithme du FMQ et celui du Q-learning distribué : tous deux utilisent une fonction de valeur Q_{max} pour mémoriser la récompense maximale reçue pour chaque action. Cette fonction donne la plupart du temps une **sur-estimation des valeurs des actions**, excepté **dans le cas des jeux déterministes où elle donne la valeur maximale exacte de chaque action**, comme le montrent [LR00] et notre étude du Q-learning distribué au §6.3 et la figure 7.1. Le FMQ utilise aussi la fonction Q du Q-learning décentralisé. Les évaluations fournies par Q sont une sous-estimation des valeurs des actions, car elles prennent en compte les pénalités reçues en cas de non-coordination. Donc, **les valeurs réelles de chaque action se situent entre celles des fonctions Q_{max} et Q**. Rappelons que si les actions des agents sont évaluées avec la fonction Q_{max} dans un jeu déterministe, les agents sont assurés de converger vers un équilibre optimal s'ils suivent un mécanisme de sélection d'équilibres adapté. Dans le cas d'environnements stochastiques, nous avons vu au chapitre précédent qu'aucun algorithme n'était capable d'assurer la convergence vers un équilibre optimal. Nous proposons donc d'utiliser les valeurs de la fonction Q pour évaluer les actions dans le cas stochastique.

Ainsi, si l'on dispose d'un moyen pour détecter la stochasticité d'un jeu, on peut alors évaluer les actions des agents avec la fonction Q_{max} si le jeu est déterministe et avec la fonction Q sinon. Pour évaluer la stochasticité d'un jeu, nous utilisons la fréquence d'occurrence F de la récompense maximale associée à une action. En effet, si le jeu est

déterministe et si les agents parviennent à se coordonner, cette fréquence tend alors vers 1. Nous proposons donc l'interpolation linéaire suivante pour évaluer une action a :

$$\mathsf{E}(a) = [1 - \mathsf{F}(a)]\, \mathsf{Q}(a) + \mathsf{F}(a)\ \mathsf{Q}_{max}(a). \tag{7.4}$$

Ainsi, les évaluations d'une action sont proches de Q_{max} dans un jeu déterministe. Dans le cas contraire, elles sont inférieures à Q_{max} et fluctuent entre les valeurs de Q_{max} et de Q selon la coordination des agents. En effet, quand la récompense maximale est reçue après avoir effectuée une action a, la fréquence d'occurrence de cette action est réinitialisée à 1 donc l'agent est en premier lieu optimiste. Ensuite, si le jeu est déterministe et que les agents se coordonnent, la fréquence de l'action a reste proche de 1 car l'agent va ensuite toujours recevoir la récompense maximale associée à cette action (sauf en cas d'actions d'exploration). Si la récompense est bruitée, la fréquence d'occurrence de cette action va diminuer et l'agent sera de moins en moins optimiste. Les évaluations des actions seront alors plus proches des Q valeurs. *Les évaluations fluctuent donc entre des évaluations optimistes suivant le Q-learning distribué et des évaluations moyennes suivant le Q-learning décentralisé selon la stochasticité détectée dans le jeu.*

Discussion

Cette proposition d'évaluation des actions est une heuristique. Elle peut donc être améliorée ou modifiée. Une autre possibilité serait de ne pas faire fluctuer les évaluations de manière linéaire entre les valeurs Q_{max} et Q, mais d'appliquer une sorte de « tout ou rien » entre ces valeurs. Il faudrait alors déterminer un seuil pour la fréquence au-dessus duquel les agents seraient optimistes ($\mathsf{E} = \mathsf{Q}_{max}$) et en-dessous duquel ils suivraient le Q-learning décentralisé ($\mathsf{E} = \mathsf{Q}$). Il faudrait néanmoins déterminer cette valeur seuil.

7.3.3 Conclusion

Cette étude du FMQ a mis en lumière deux limitations principales de celui-ci : son manque de robustesse face à la stratégie d'exploration et face au paramètre de pondération de l'heuristique. Des solutions ont donc été proposées. Tout d'abord, afin d'améliorer sa robustesse face à l'exploration, un calcul récursif de la fréquence d'occurrence a été proposé. Nous avons aussi introduit une nouvelle heuristique basée sur une interpolation linéaire entre les valeurs de Q_{max} et de Q. Ainsi, l'évaluation des actions fluctue entre le Q-learning distribué et le Q-learning décentralisé selon la stochasticité du jeu, qui peut être détectée avec la fréquence d'occurrence. Ce principe sera notamment étudié plus en détail dans la section suivante où nous proposons d'appliquer ces solutions à l'algorithme du FMQ afin d'en obtenir une version modifiée.

7.4 *Frequency Maximum Q-value* récursif

Dans cette section est proposée une version modifiée de l'algorithme du FMQ intégrant les solutions proposées précédemment. L'utilisation de la fréquence récursive pour détecter la stochasticité est aussi détaillée. Cette version modifiée est testée sur différents

Algorithme 10 : Algorithme du Frequency Maximum Q-value récursif pour un agent indépendant i dans un jeu matriciel d'équipe

début
  Initialiser $\mathsf{Q}_i(a_i)$, $\mathsf{Q}_{max,i}(a_i)$ et $\mathsf{E}_i(a_i)$ à 0, $\mathsf{F}_i(a_i)$ à 1 $\forall a_i \in \mathcal{A}_i$
  Initialiser $\pi_i(a_i)$ arbitrairement $\forall a_i \in \mathcal{A}_i$
  répéter
    Choisir l'action a_i selon la méthode de décision ϵ-greedy basée sur π_i
    Exécuter l'action a_i et observer la récompense r
    $\mathsf{Q}_i(a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(a_i) + \alpha r$
    si $r > \mathsf{Q}_{max,i}(a_i)$ **alors**
      $\mathsf{Q}_{max,i}(a_i) \leftarrow r$ ▷ `mise à jour optimiste`
      $\mathsf{F}_i(a_i) \leftarrow 1$ ▷ `réinitialisation de la fréquence`
    sinon si $r = \mathsf{Q}_{max,i}(a_i)$ **alors**
      $\mathsf{F}_i(a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(a_i) + \alpha_f$ ▷ `calcul récursif de la fréquence`
    sinon
      $\mathsf{F}_i(a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(a_i)$
    ▷ `heuristique par interpolation linéaire`
    $\mathsf{E}_i(a_i) \leftarrow [1-\mathsf{F}_i(a_i)]\mathsf{Q}_i(a_i) + \mathsf{F}_i(a_i)\mathsf{Q}_{max,i}(a_i)$
    si $\mathsf{E}_i(\arg\max_{u\in\mathcal{A}_i} \pi_i(u)) \neq \max_{u\in\mathcal{A}_i} \mathsf{E}_i(u_i)$ **alors**
      Sélectionner $a_{max} \in \arg\max_{u\in\mathcal{A}_i} \mathsf{E}_i(u)$ aléatoirement
      $\forall b \in \mathcal{A}_i \quad \pi_i(b) \leftarrow \begin{cases} 1 & \text{si } b = a_{max} \\ 0 & \text{sinon} \end{cases}$ ▷ `sélection d'équilibres`
  jusqu'à *la fin des répétitions*
fin

jeux matriciels et notamment, nous étudions son application dans le cas de jeux à plus de 2 agents.

7.4.1 Extension du *Frequency Maximum Q-value*

Nous intégrons les modifications proposées dans la section précédente à l'algorithme original du FMQ. La version modifiée est alors donnée à l'algorithme 10. Nous nommons cet algorithme le **FMQ récursif**. Une fonction de valeur Q_{max} est mise à jour de manière optimiste, *i.e.* selon le Q-learning distribué, pour mémoriser la récompense maximale reçue pour chaque action. En parallèle est aussi mise à jour la fonction de valeur Q selon le Q-learning décentralisé. Les valeurs de ces fonctions sont utilisées comme bornes pour évaluer les actions individuelles grâce à une nouvelle heuristique (*cf.* équation 7.4). Cette heuristique consiste en une interpolation linéaire réalisée entre ces bornes et basée sur la fréquence. Les évaluations d'une action fluctuent donc entre les valeurs de Q_{max} et de Q selon la stochasticité du jeu. Pour détecter la stochasticité, la fréquence d'occurrence de la récompense maximale associée à une action est utilisée. Elle est notée F. Elle est calculée de manière récursive et réinitialisée pour une action lorsqu'une nouvelle récompense maximale est reçue après avoir effectué cette action. Ainsi, si le jeu est déterministe et si les agents parviennent à se coordonner sur une action jointe optimale, la valeur de la fréquence associée à cette action sera proche de 1. Les agents vont donc évaluer leurs

actions de manière optimiste. Dans les autres cas, les évaluations fluctuent entre les valeurs optimistes et moyennes selon la fréquence. Le mécanisme de sélection d'équilibres proposé par [LR00] est utilisé. Il choisit aléatoirement parmi les actions qui maximisent les évaluations E.

Le principe de cet algorithme est proche de celui des agents indulgents [SPBL06, PSL06] (*cf.* §6.5) étant donné que le degré d'optimisme des agents peut changer. Cependant, concernant les agents indulgents, le degré d'optimiste diminue inévitablement, alors qu'ici, **si les agents réussissent à se coordonner dans un jeu déterministe, ils restent optimistes et vont donc converger vers l'action jointe optimale étant donnée la preuve de convergence du Q-learning distribué**.

7.4.2 Étude de l'influence de l'exploration globale sur la fréquence

Le rôle de la fréquence dans l'heuristique est de détecter la stochasticité du jeu. Elle doit donc être capable de distinguer le bruit causé par la non-coordination des agents du bruit de l'environnement. Pour cela, une condition nécessaire est que la fréquence ne soit pas sensible à l'exploration des agents. En effet, l'exploration d'un autre agent peut être interprétée comme du bruit par un agent indépendant. Ce découplage entre les différentes causes de bruit est d'autant plus complexe à réaliser que le nombre d'agents augmente. Si les agents sont nombreux, le bruit dû à leur exploration sera amplifié. L'impact de nombreux agents peut alors faire ressembler la fonction de récompense à une fonction bruitée du point du vue d'un agent indépendant. Il faut donc nous assurer que la fréquence est robuste à l'exploration des autres quel que soit le nombre d'agents présents dans le jeu. Pour cela, le choix du coefficient d'apprentissage $\alpha_f \in [0;1]$ de la fréquence récursive est important et doit être précisé. Il sera fonction de l'exploration globale dans le système (*cf.* définition 7.1).

Ce coefficient α_f règle la vitesse de croissance et de décroissance de la fréquence. **Avec $\alpha_f = 0$, le FMQ récursif est équivalent au Q-learning distribué**. Plus α_f est grand, plus la fréquence sera sensible aux actions d'exploration des agents. En effet, supposons que deux agents soient coordonnés dans un jeu déterministe. La fréquence d'occurrence de la récompense maximale pour l'action individuelle optimale de chaque agent est alors proche de 1. Si un des agents explore, entraînant alors une récompense reçue inférieure à la récompense maximale, la fréquence de l'action individuelle optimale de l'agent qui exploite décroît alors d'un facteur $(1 - \alpha_f)$. Donc, si le coefficient d'apprentissage α_f est trop élevé, les fréquences d'occurrence de l'action individuelle optimale peuvent être « détruites » par une action d'exploration d'un agent. *Le coefficient α_f doit donc être choisi d'autant plus faible que l'exploration globale sera élevée.*

A la table 7.4 sont donnés les choix préconisés pour le coefficient d'apprentissage de la fréquence α_f en fonction de la valeur de l'exploration globale ψ. Ces choix sont issus de l'expérience et permettent d'obtenir une fréquence récursive robuste face à l'exploration des autres. Notamment, une stratégie GLIE est déconseillée car cela entraînerait alors un

TABLE 7.4 – Choix préconisés pour le coefficient d'apprentissage de la fréquence α_f en fonction de la valeur de l'exploration globale ψ.

Exploration globale	Coefficient d'apprentissage de la fréquence récursive
$\psi = 0,1$	$\alpha_f = 0.01$
$\psi = 0,15$	$\alpha_f = 0.001$
$\psi = 0,2$	$\alpha_f = 0.0001$

TABLE 7.5 – Pourcentage d'épisodes convergeant vers l'une des actions jointes optimales avec le FMQ récursif (moyenne sur 500 épisodes indépendants). Un épisode consiste en 5000 répétitions. La stratégie d'exploration choisie est **stationnaire**. On pose $\alpha = 0,1$, $\alpha_f = 0,01$ et $\epsilon = 0,05$. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale (x% de convergence vers un équilibre de Nash Pareto optimal) ou sous-optimale (y% de convergence vers un équilibre de Nash sous-optimal). Les résultats sont notés x% [y%].

	Jeu simple	Jeu *Penalty* ($k = -100$)	Jeu *Climbing*		
			déterministe	partiellement bruité	bruité
FMQ récursif	100%	100%	100%	100%	56% [3%]

choix de α_f qui pourrait s'avérer très complexe. Ceci est en accord avec nos mises en garde précédentes contre l'utilisation de telles stratégies, étant données leur difficulté de réglage et leur influence sur la robustesse de l'algorithme. Par contre, une stratégie stationnaire est parfaitement adaptée. Dans ce cas, il faut assurer une exploration suffisante de l'espace d'état d'une part pour permettre aux agents de trouver l'action jointe optimale, et d'autre part pour permettre à la fréquence de converger vers sa valeur moyenne. L'exploration globale doit cependant être limitée pour éviter un bruit trop important.

7.4.3 Résultats sur des jeux matriciels d'équipe

Nous testons l'algorithme du FMQ récursif sur divers jeux matriciels à deux agents et plus. Ces jeux regroupent certains facteurs de non-coordination et permettent de tester si le FMQ récursif surmonte ces enjeux. Un paramètre important de ces essais concerne la stratégie d'exploration et le nombre de répétitions de chaque jeu. Comme précisé précédemment, nous utilisons une stratégie d'exploration stationnaire, plus simple à régler et permettant un choix de α_f moins complexe. Cependant, cela implique, lorsque les agents sont nombreux, un nombre important de répétitions du jeu. En effet, l'exploration de l'espace des actions jointes doit être suffisante pour que les agents trouvent le ou les actions jointes optimales parmi l'ensemble des actions jointes. Mais l'exploration globale doit être limitée pour éviter un bruit trop important. Or, le nombre d'actions

jointes est exponentiel avec le nombre d'agents. Le nombre de répétitions effectuées est donc croissant avec le nombre d'agents.

Résultats sur des jeux matriciels d'équipe à 2 agents

Tout d'abord, l'algorithme du FMQ récursif est testé sur les jeux matriciels des figures 5.3 à 5.7. Les résultats sont donnés à la table 7.5 (méthode de décision ϵ-greedy). Cette version récursive converge vers l'action jointe optimale dans les jeux déterministes car les agents sont optimistes et parviennent à se coordonner. Elle parvient aussi à surmonter la difficulté de récompenses partiellement bruitées. Par contre, elle ne surmonte pas toujours la difficulté de jeux fortement bruités tel que le *Climbing* bruité. Les raisons sont identiques à celles qui mettent en défaut le FMQ original dans ce jeu (*cf.* §6.4).

Ainsi, les résultats sont équivalents à ceux obtenus avec le FMQ initial. Mais l'intérêt majeur de cette version récursive est qu'elle est plus robuste face à l'exploration, grâce à un choix du coefficient d'apprentissage de la fréquence qui a été précisé selon l'exploration globale choisie. Cela permet d'assurer que la fréquence ne soit pas détruite par l'exploration.

Résultats sur des jeux matriciels d'équipe à $n > 2$ agents

Afin de vérifier que la fréquence d'occurrence découple bien les diverses causes de bruit, nous choisissons de tester l'algorithme sur des jeux matriciels à plus de 2 agents. En effet, lorsque le nombre d'agents augmente, le bruit dû à l'exploration des agents est alors plus important et donc plus difficile à distinguer du bruit dû à l'environnement. Nous proposons donc d'étendre l'étude au cas où plus de deux agents doivent se coordonner dans des jeux complexes présentant un ou plusieurs facteurs de non-coordination.

Nous utilisons une version dérivée du jeu *penalty* avec un nombre d'agents $n > 2$. Les agents ont chacun trois actions possibles a, b ou c. Si la moitié des agents ou plus jouent l'action a et que les autres jouent l'action c, ils reçoivent une récompense de 10. Si moins de la moitié des agents jouent l'action a et que les autres jouent l'action c, ils reçoivent une pénalité de -100 car ils ont échoué à se coordonner. Si la moitié des agents ou plus jouent l'action b et que les autres jouent l'action c, ils reçoivent 2. Dans tous les autres cas, une récompense de 0 est reçue. On retrouve dans ce jeu plusieurs équilibres de Nash Pareto optimaux qui correspondent au cas où la moitié des agents ou plus jouent l'action a et que les autres jouent l'action c. Ces équilibres optimaux sont cachés par les pénalités en cas de non-coordination. Sont aussi présents plusieurs équilibres de Nash sous-optimaux lorsque la moitié des agents ou plus jouent l'action b et que les autres jouent l'action c. Ce jeu présente donc plusieurs facteurs compliquant la coordination. De plus, la récompense peut être partiellement bruitée. Dans ce cas, au lieu de recevoir une récompense de 2, les récompenses 12 et 6 sont reçues de manière équiprobable.

TABLE 7.6 – Pourcentage d'épisodes convergeant vers l'une des actions jointes optimales dans le jeu matriciel *penalty* à $n > 2$ agents (moyenne sur 500 épisodes indépendants). Les paramètres choisis pour chaque expérience sont disponibles à l'annexe A. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale (x% de convergence vers un équilibre de Nash Pareto optimal) ou sous-optimale (y% de convergence vers un équilibre de Nash sous-optimal). Les résultats sont notés x% [y%].

		Q-learning décentralisé		Q-learning distribué		WoLF PHC		FMQ récursif	
$n = 3$	déterministe	100%	[0%]	100%	[0%]	100%	[0%]	100%	[0%]
	stochastique	98%	[2%]	0%	[100%]	87%	[13%]	99%	[1%]
$n = 4$	déterministe	100%	[0%]	100%	[0%]	100%	[0%]	91%	[9%]
	stochastique	6%	[94%]	0%	[100%]	0%	[100%]	92%	[8%]
$n = 5$	déterministe	100%	[0%]	100%	[0%]	100%	[0%]	97%	[3%]
	stochastique	10%	[90%]	0%	[100%]	1%	[99%]	94%	[6%]

Nous testons les algorithmes du Q-learning décentralisé, du Q-learning distribué, du WoLF-PHC et du FMQ récursif dans ce jeu. Les trois premiers algorithmes ont été présentés au chapitre 6. Pour le Q-learning décentralisé, nous utilisons une stratégie d'exploration GLIE car cela permet, si elle est bien paramétrée, d'obtenir de meilleurs résultats avec cet algorithme. Concernant les autres algorithmes, nous choisissons une stratégie stationnaire. Dans chaque cas, l'exploration doit permettre une visite uniforme de l'ensemble des actions jointes. A l'annexe A sont détaillées les valeurs choisies pour ces stratégies. Les résultats sont quant à eux donnés à la table 7.6.

Tout d'abord, le Q-learning décentralisé converge vers l'une des actions jointes optimales dans tous les jeux déterministes avec une stratégie GLIE correctement choisie. Mais lorsque le nombre d'agents augmente, ses performances dans les jeux stochastiques diminuent considérablement. Il est cependant difficile de déterminer si cela est dû à un mauvais choix de la stratégie GLIE ou à une non-robustesse de l'algorithme face à la difficulté de récompenses bruitées. De même, le WoLF-PHC est mis en défaut dès que des récompenses stochastiques sont utilisées. Il est cependant intéressant de remarquer que dans les environnements bruités, ces algorithmes convergent alors vers les équilibres de Nash sous-optimaux.

Les résultats obtenus avec le Q-learning distribué sont conformes à l'étude effectuée au §6.3 : les agents optimistes convergent vers une des actions jointes optimales dans tous les jeux déterministes. Par contre, dans les jeux stochastiques, ils convergent vers les équilibres de Nash sous-optimaux car ce sont ces équilibres qui ont la récompense (bruitée) maximale.

Enfin, le FMQ récursif obtient les meilleurs résultats sur l'ensemble des jeux testés : il converge vers l'une des actions jointes optimales plus de 9 fois sur 10 avec tous les jeux déterministes et stochastiques. Notamment, lorsque $n > 3$, on peut remarquer

que le FMQ surpasse largement le Q-learning décentralisé dans les jeux stochastiques. L'évaluation des actions n'est donc pas ici égale aux valeurs de Q mais se situe entre les bornes Q et Q_{max}. **L'interpolation linéaire pour évaluer les actions est donc une bonne heuristique de mesure des valeurs réelles des actions**.

7.4.4 Conclusion

L'algorithme du FMQ récursif proposé dans cette section présente de nombreux intérêts. Tout d'abord, contrairement au FMQ original, il peut être utilisé avec une stratégie d'exploration stationnaire, ce qui permet d'éviter le choix parfois complexe des paramètres de décroissance d'une stratégie GLIE. Cela évite de plus le risque de manque de robustesse face à ces paramètres. Ensuite, un moyen de découpler les diverses causes de bruit dans un jeu multiagent a été proposé et étudié. La fréquence récursive permet notamment de découpler le bruit dû à l'exploration des agents dans des jeux multiagents faiblement bruités. Le réglage du coefficient de cette fréquence a été précisé dans cette section en fonction de l'exploration globale choisie, de sorte à obtenir une fréquence robuste à l'exploration des autres.

L'algorithme du FMQ récursif obtenu est proche du Q-learning distribué dans les jeux déterministes. Il est de plus capable de surmonter l'enjeu de récompenses faiblement bruitées, surpassant ainsi les performances des autres algorithmes. *L'algorithme du FMQ récursif est donc un algorithme facile d'utilisation capable de résoudre des jeux matriciels d'équipe complexes.* Sa robustesse face à l'exploration et face à certains facteurs de non-coordination a été démontrée sur des jeux multiagents ($n > 2$). Cet algorithme est toutefois restreint au cadre des jeux matriciels. Une extension aux jeux de Markov est donc proposée dans la section suivante.

7.5 Algorithme *Swing between Optimistic or Neutral*

L'objectif de cette section est de proposer un algorithme robuste face à l'exploration et capable de surmonter les trois facteurs de non-coordination dans les jeux de Markov d'équipe. L'algorithme du FMQ récursif répond à la plupart de ces objectifs. Cependant, son utilisation est limitée aux jeux matriciels. L'extension directe du FMQ récursif aux jeux de Markov, proposée au début de cette section, n'est pas satisfaisante comme le montre l'étude d'un cas particulier. Une nouvelle fréquence mieux adaptée au cadre multi-état est alors proposée. Il en résulte un nouvel algorithme pour la coordination d'agents indépendants dans le cadre des jeux de Markov d'équipe. Nous l'avons baptisé l'algorithme *Swing between Optimistic or Neutral*, ou SOoN, car il évalue les couples état-action de manière optimiste ou moyenne selon une détection de la stochasticité de l'environnement.

Algorithme 11 : Algorithme du Frequency Maximum Q-value récursif pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s,a_i)$, $\mathsf{Q}_{max,i}(s,a_i)$ et $\mathsf{E}_i(s,a_i)$ à 0, $\mathsf{F}_i(s,a_i)$ à 1 $\forall (s,a_i) \in \mathcal{S} \times \mathcal{A}_i$
  Initialiser $\pi_i(s,a_i)$ arbitrairement $\forall (s,a_i) \in \mathcal{S} \times \mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i selon la méthode de décision ϵ-greedy basée sur π_i
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $\mathsf{Q}_i(s,a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(s,a_i) + \alpha(r + \gamma \max_{u \in A_i} \mathsf{Q}_i(s',u))$
    $q \leftarrow r + \gamma \max_{u \in \mathcal{A}_i} \mathsf{Q}_{max,i}(s',u)$
    si $q > \mathsf{Q}_{max,i}(s,a_i)$ **alors**
      $\mathsf{Q}_{max,i}(s,a_i) \leftarrow q$ ▷ `mise à jour optimiste`
      $\mathsf{F}_i(s,a_i) \leftarrow 1$ ▷ `réinitialisation de la fréquence`
    sinon si $q = \mathsf{Q}_{max,i}(s,a_i)$ **alors**
      $\mathsf{F}_i(s,a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(s,a_i) + \alpha_f$ ▷ `calcul récursif de la fréquence`
    sinon
      $\mathsf{F}_i(s,a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(s,a_i)$
    ▷ `heuristique par interpolation linéaire`
    $\mathsf{E}_i(s,a_i) \leftarrow [1-\mathsf{F}_i(s,a_i)]\mathsf{Q}_i(s,a) + \mathsf{F}_i(s,a_i)\mathsf{Q}_{max,i}(s,a_i)$
    si $\mathsf{E}_i(s, \arg\max_{u \in A_i} \pi_i(s,u)) \neq \max_{u \in A_i} \mathsf{E}_i(s,u)$ **alors**
      Choisir $a_{max} \in \arg\max_{u \in A_i} \mathsf{E}_i(s,u)$ aléatoirement
      $\forall b \in A_i \quad \pi_i(s,b) \leftarrow \begin{cases} 1 & \text{si } b = a_{max} \\ 0 & \text{sinon} \end{cases}$ ▷ `sélection d'équilibres`
    $s \leftarrow s'$
fin

7.5.1 Étude de l'extension directe du FMQ récursif aux jeux de Markov

L'extension directe aux jeux de Markov du FMQ récursif, donnée à l'algorithme 11, est assez évidente. La principale différence réside dans le calcul de la fréquence d'occurrence $\mathsf{F}_i(s,a)$ d'un agent i.

La fréquence immédiate ou « myope » $\mathsf{F}_i(a)$ calculée de façon récursive dans les jeux matriciels correspond à la probabilité de recevoir la récompense maximale après avoir exécuté une action a :

$$\mathsf{F}_i(a) = P_r\left\{r_{t+1} = \mathsf{Q}_{max,i}(a_t) | a_t = a\right\}. \tag{7.5}$$

Dans les jeux de Markov, on donne la définition suivante de la fréquence immédiate.

Définition 7.1 *Lorsque l'action a est effectuée dans l'état s, la fréquence immédiate $\mathsf{F}_i(s,a)$ pour l'agent i est la probabilité que la transition immédiate fournisse une récompense et un état dont l'évaluation, selon une politique gloutonne sur $\mathsf{Q}_{max,i}$, est égale à*

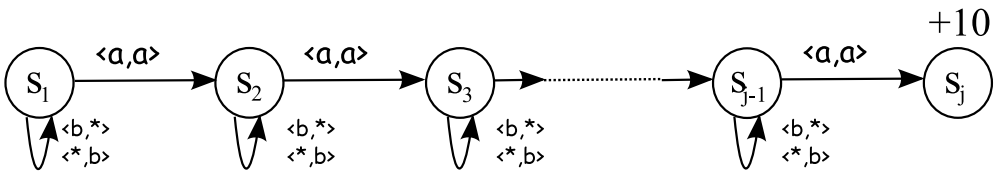


FIGURE 7.4 – Jeu de Markov d'équipe à deux agents.

la valeur maximale prévue $\mathsf{Q}_{max,i}(s,a)$, i.e. *:*

$$\mathsf{F}_i(s,a) = P_r\left\{r_{t+1} + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s_{t+1},b) = \mathsf{Q}_{max,i}(s_t,a_t) | s_t = s, a_t = a\right\}. \quad (7.6)$$

On retrouve dans l'extension directe du FMQ récursif les fonctions suivantes, définies pour chaque agent individuel i sur l'ensemble des couples état-action :

- la fonction de valeur locale Q_i mise à jour selon le Q-learning décentralisé,
- la fonction de valeur locale $\mathsf{Q}_{max,i}$ mise à jour selon le Q-learning distribué,
- la fréquence immédiate F_i calculée de la manière suivante : pour chaque transition, on calcule la valeur courante du couple selon le Q-learning distribué, que nous appellerons valeur courante optimiste. Cette valeur courante optimiste correspond à la somme de la récompense immédiate reçue et de la valeur optimiste de la politique gloutonne dans l'état suivant. Si la transition effectuée entraîne une amélioration de la valeur courante optimiste de ce couple, la fréquence est réinitialisée à 1. Si la valeur courante optimiste est égale à l'évaluation optimiste, la fréquence augmente. Dans le cas contraire, la fréquence diminue. Ainsi, l'estimation F_i correspond bien à sa définition 7.1,
- la fonction d'évaluation E_i qui utilise une heuristique basée sur une interpolation linéaire entre les valeurs de Q_i et $\mathsf{Q}_{max,i}$. Cette évaluation doit être égale à $\mathsf{Q}_{max,i}$ si le jeu est déterministe et si les agents se coordonnent. Dans le cas d'un jeu stochastique, les valeurs de la fonction E_i se rapprochent de celles de la fonction Q_i,
- la politique courante π_i mise à jour selon un mécanisme de sélection d'équilibres basé sur un choix aléatoire parmi les actions qui maximisent la fonction d'évaluation E_i.

Avec cette extension directe, la fréquence issue de la version en jeux matriciels est une mesure immédiate car elle ne prend en compte que la fréquence dans l'état courant. Ce raisonnement « myope » peut cependant être insuffisant dans le cas des jeux multi-états. Nous allons donc nous intéresser aux possibles limitations entraînées par cette vision « myope » de la fréquence.

7.5.2 Fréquence immédiate et limitations

Cette extension directe de l'algorithme du FMQ récursif présente des défaillances dans certains jeux de Markov. Afin d'illustrer ces défaillances, nous testons cet algorithme sur le jeu de Markov d'équipe à deux agents de la figure 7.4 et nous y étudions

notamment l'évolution de la fréquence immédiate. Ce jeu est particulièrement difficile car les agents doivent coordonner leurs actions dans chaque état. Chaque agent a le choix parmi deux actions a et b. Le jeu débute avec les deux agents dans l'état s_1. Les transitions sur la figure sont indiquées par un couple d'actions où se retrouvent respectivement l'action de l'agent 1 et celle de l'agent 2. Le sigle $*$ est un joker représentant une action quelconque. Si les deux agents se coordonnent sur l'action jointe $\langle a, a\rangle$ dans l'état s_k, ils se déplacent alors dans l'état suivant s_{k+1}. Si au moins un des agents joue l'action b, les deux agents restent dans leur état actuel. Aucune récompense n'est reçue sauf lorsque l'état absorbant s_j est atteint ; la récompense reçue est alors égale à 10.

Les valeurs optimales de la fonction $Q_{max,i}$ pour chaque agent i, dans l'état s_k, sont :

$$\begin{cases} \mathsf{Q}^*_{max,i}(s_k, a) = 10 \times \gamma^{j-k-1} \\ \mathsf{Q}^*_{max,i}(s_k, b) = 10 \times \gamma^{j-k} \end{cases} \tag{7.7}$$

Nous nous plaçons du point de vue d'un agent. Deux cas sont alors étudiés :

- l'agent joue l'action b dans l'état s_k, donc, quelle que soit l'action jouée par l'autre agent, l'état suivant s' est l'état s_k et pour chaque agent i :

$$\begin{aligned} q_i &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s', u) \\ &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s_k, u) \\ &= 10 \times \gamma^{j-k} \\ &= \mathsf{Q}^*_{max,i}(s_k, b) \end{aligned} \tag{7.8}$$

  Ainsi, du point de vue d'un agent, l'action b est sûre et la fréquence d'occurrence de la valeur optimiste Q_{max} pour l'action b, notée $\mathsf{F}_i(s_k, b)$, augmente à chaque fois que l'action b est jouée dans l'état s par un agent i. De plus, elle ne diminue jamais donc $\mathsf{F}_i(s_k, b)$ **tend vers 1**.
- l'agent joue l'action a dans l'état s_k. Si l'autre agent joue aussi l'action a, la coordination est réussie et l'état suivant s' est alors s_{k+1} :

$$\begin{aligned} q_i &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s', u) \\ &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s_{k+1}, u) \\ &= 10 \times \gamma^{j-k-1} \\ &= \mathsf{Q}^*_{max,i}(s_k, a) \end{aligned} \tag{7.9}$$

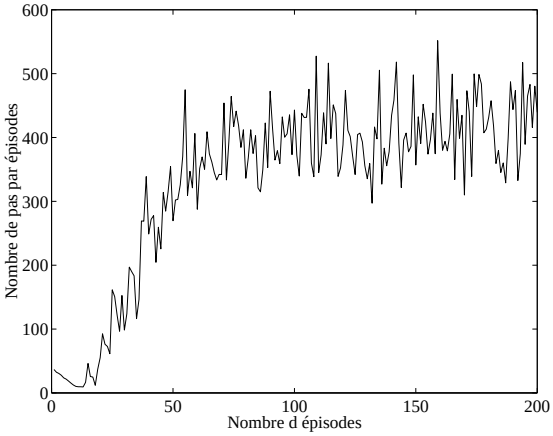


FIGURE 7.5 – Expériences sur le jeu de Markov d'équipe à deux agents de la figure 7.5 avec l'extension directe du FMQ récursif (moyennées sur 200 essais indépendants). Le jeu a 10 états et on choisit $\alpha = 0,1$, $\gamma = 0,9$, $\alpha_f = 0,05$, $\epsilon = 0,05$.

Si l'autre agent joue l'action b, il y a non-coordination et l'état suivant s' est alors s_k :

$$\begin{aligned} q_i &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s', u) \\ &= \gamma \max_{u=\{a,b\}} \mathsf{Q}_{max,i}(s_k, u) \\ &= 10 \times \gamma^{j-k} \\ &< \mathsf{Q}^*_{max,i}(s_k, a) \end{aligned} \tag{7.10}$$

Donc $\mathsf{F}_i(s_k, a)$ **diminue à chaque fois que l'autre agent joue** b.

D'une part, $\mathsf{F}_i(s, b)$ augmente à chaque fois que l'action b est jouée dans l'état s par un agent i et ne diminue jamais. D'autre part, $\mathsf{F}_i(s, a)$ diminue à chaque fois que les agents échouent à se coordonner dans l'état s. Donc, du point de vue d'un agent, comme $\mathsf{Q}^*_{max,i}(s, a)$ est assez proche de $\mathsf{Q}^*_{max,i}(s, b)$ ($\mathsf{Q}^*_{max,i}(s, b) = 0 + \gamma \mathsf{Q}^*_{max,i}(s, a)$), on a rapidement $\mathsf{E}(s, b) > \mathsf{E}(s, a)$. L'action b est mieux évaluée que l'action a. L'action jointe choisie en exploitation par les agents est $\langle b, b \rangle$; ils échouent à se coordonner et restent sur place ! Ce problème est illustré sur les résultats de la figure 7.5. Le nombre de pas par épisode augmente rapidement car les agents restent sur place et ont beaucoup de mal à se coordonner. Au début de l'apprentissage, les agents ont un comportement aléatoire donc le nombre de pas par épisode est faible durant les premiers épisodes.

Cet exemple a permis de mettre en avant une défaillance de l'extension directe du FMQ récursif aux jeux de Markov due à l'utilisation d'une mesure locale de la fréquence. La fréquence F_i calculée par l'algorithme est une fréquence immédiate ou « myope » étant donnée qu'elle ne prend en compte que la **valeur courante** d'un couple état-action. Ce

raisonnement « myope » centré sur l'état courant est insuffisant dans le cadre multiagent. La fréquence doit être considérée globalement en prenant en compte les fréquences des états futurs. En effet, la fréquence immédiate d'une action peut être élevée dans un état, mais cette action peut alors mener dans un état où l'action optimale a une fréquence immédiate faible. En d'autres termes, une action peut paraître intéressante dans un état selon sa fréquence immédiate mais peut être un mauvais choix en ce qui concerne le futur.

L'utilisation d'une mesure immédiate de la fréquence est donc dangereuse car elle peut induire en erreur l'algorithme. Nous allons donc maintenant proposer une méthode de calcul pour raisonner à long terme à partir de cette mesure immédiate.

7.5.3 Fréquence distante

L'utilisation d'une fréquence immédiate doit être évitée, comme nous l'avons illustré précédemment. Nous proposons donc de calculer une fréquence « sur le long terme » définie comme suit.

Définition 7.2 *Lorsque l'action a est effectuée dans l'état s, la* ***fréquence distante*** $\mathsf{G}_i(s,a)$ *pour l'agent i est la probabilité que* ***toutes*** *les transitions futures fournissent des récompenses et des états dont les évaluations, selon une politique gloutonne sur $\mathsf{Q}_{max,i}$, sont égales à la valeur maximale prévue $\mathsf{Q}_{max,i}$ pour ces états,* i.e. *:*

$$\mathsf{G}_i(s,a) = P_r^{\pi_{max}}\Big\{r_{t+1} + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s_{t+1},b) = \mathsf{Q}_{max,i}(s_t,a_t) \wedge \\ r_{t+2} + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s_{t+2},b) = \mathsf{Q}_{max,i}(s_{t+1},a_{t+1}) \wedge \ldots | s_t = s, a_t = a\Big\} \quad (7.11)$$

Nous allons maintenant proposer une estimation récursive de cette fréquence distante.

Nous faisons l'hypothèse que les évènements précédents sont indépendants, c'est-à-dire que la réalisation de l'un n'influe pas sur la probabilité de réalisation d'un autre. L'égalité précédente est donc équivalente à :

$$\mathsf{G}_i(s,a) = P_r^{\pi_{max}}\Big\{r_{t+1} + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s_{t+1},b) = \mathsf{Q}_{max,i}(s_t,a_t) | s_t = s, a_t = a\Big\} \times \\ P_r^{\pi_{max}}\Big\{r_{t+2} + \gamma \max_{b\in\mathcal{A}_i} \mathsf{Q}_{max,i}(s_{t+2},b) = \mathsf{Q}_{max,i}(s_{t+1},a_{t+1}) \wedge \ldots | s_t = s, a_t = a\Big\}. \quad (7.12)$$

Le premier terme de l'égalité est égal à $\mathsf{F}_i(s,a)$ d'après la définition 7.1. D'après le théorème des probabilités totales, on peut récrire le second terme de l'égalité précédente

selon :

$$P_r^{\pi_{max}}\left\{r_{t+2}+\gamma\max_{b\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s_{t+2},b)=\mathsf{Q}_{max,i}(s_{t+1},a_{t+1})\wedge\ldots|s_t=s,a_t=a\right\}=$$
$$\sum_{s'\in\mathcal{S}}P_r^{\pi_{max}}\left\{r_{t+2}+\gamma\max_{b\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s_{t+2},b)=\mathsf{Q}_{max,i}(s_{t+1},a_{t+1})\wedge\ldots|s_t=s,a_t=a,s_{t+1}=s'\right\}\times$$
$$P_r\left\{s_{t+1}=s'|s_t=s,a_t=a\right\}. \quad (7.13)$$

La politique π_{max} est suivie donc $a_{t+1}\in\arg\max_{b\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s_{t+1},b)$ et :

$$\mathsf{G}_i(s,a) \quad = \quad \mathsf{F}_i(s,a)\sum_{s'\in\mathcal{S}}P_r\left\{s_{t+1}=s'|s_t=s,a_t=a\right\}\mathsf{G}_i(s',\arg\max_{b\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s_{t+1},b)) \quad (7.14)$$

Si on note $P_{ss'}^a$ la probabilité de chaque état suivant possible s' étant donné un état s et une action a, on aboutit à :

$$\mathsf{G}_i(s,a)=\mathsf{F}_i(s,a)\sum_{s'\in\mathcal{S}}P_{ss'}^a\mathsf{G}_i(s',\arg\max_{u\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s',u)) \quad (7.15)$$

Comme $P_{ss'}^a$ est inconnue, nous proposons pour estimer la fréquence distante le calcul récursif suivant :

$$\mathsf{G}_i(s,a)\leftarrow(1-\alpha_g)\mathsf{G}_i(s,a)+\alpha_g\mathsf{F}_i(s,a)\max_{\substack{v\in\arg\max_{u\in\mathcal{A}_i}\mathsf{Q}_{max,i}(s',u)}}\mathsf{G}_i(s',v) \quad (7.16)$$

où $\alpha_g\in]0;1]$ est le coefficient d'apprentissage de la fréquence distante. Cette mise à jour de la fréquence distante repose sur le principe de différence temporelle. L'amélioration apportée à la fréquence distante à travers cette méthode est le produit de la fréquence immédiate du couple état-action effectué par la fréquence immédiate de l'action optimale dans l'état suivant. Ainsi, si une action a dans l'état courant une fréquence immédiate élevée mais peut mener dans un état où l'action optimale a une faible fréquence immédiate, alors cette action aura une fréquence distante faible.

L'idée générale est proche du principe de propagation des bonus d'exploration proposé par Nicolas Meuleau et Paul Bourgine [MB99]. Cette méthode permet notamment une exploration plus complète de l'environnement grâce à un mécanisme de rétro propagation de mesures locales de bonus d'exploration. Dans notre cas, ce sont les valeurs des fréquences immédiates qui sont propagées : pour chaque état visité, l'agent « regarde en avant » [2] vers les états futurs pour prendre en compte les valeurs de fréquences des actions optimales de ces états.

Algorithme 12 : Algorithme SOoN pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s,a_i)$, $\mathsf{Q}_{max,i}(s,a_i)$ et $\mathsf{E}_i(s,a_i)$ à 0, $\mathsf{F}_i(s,a_i)$ et $\mathsf{G}_i(s,a_i)$ à 1 $\forall (s,a_i) \in \mathcal{S} \times \mathcal{A}_i$
  Initialiser $\pi_i(s,a_i)$ arbitrairement $\forall (s,a_i) \in \mathcal{S} \times \mathcal{A}_i$
  Initialiser l'état initial s
  tant que *s n'est pas un état absorbant* **faire**
    Dans l'état s choisir l'action a_i selon la méthode de décision ϵ-greedy basée sur π_i
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $\mathsf{Q}_i(s,a_i) \leftarrow (1-\alpha)\mathsf{Q}_i(s,a_i) + \alpha(r + \gamma \max_{u \in A_i} \mathsf{Q}_i(s',u))$
    $q \leftarrow r + \gamma \max_{u \in \mathcal{A}_i} \mathsf{Q}_{max,i}(s',u)$
    si $q > \mathsf{Q}_{max,i}(s,a_i)$ **alors**
      $\mathsf{Q}_{max,i}(s,a_i) \leftarrow q$ ▷ `mise à jour optimiste`
      $\mathsf{F}_i(s,a_i) \leftarrow 1$ ▷ `réinitialisation de la fréquence immédiate`
      $\mathsf{G}_i(s,a_i) \leftarrow 1$ ▷ `réinitialisation de la fréquence distante`
    sinon si $q = \mathsf{Q}_{max,i}(s,a_i)$ **alors**
      $\mathsf{F}_i(s,a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(s,a_i) + \alpha_f$ ▷ `calcul récursif de la fréquence`
    sinon
      $\mathsf{F}_i(s,a_i) \leftarrow (1-\alpha_f)\mathsf{F}_i(s,a_i)$
    ▷ `mise à jour de la fréquence distante`
    $\mathsf{G}_i(s,a_i) \leftarrow [1-\alpha_g]\mathsf{G}_i(s,a_i) + \alpha_g \mathsf{F}_i(s,a_i) \max_{v \in \arg\max_{u \in \mathcal{A}_i} \mathsf{Q}_{max,i}(s',u)} \mathsf{G}_i(s',v)$
    ▷ `heuristique par interpolation linéaire`
    $\mathsf{E}_i(s,a_i) \leftarrow [1-\mathsf{G}_i(s,a_i)]\mathsf{Q}_i(s,a) + \mathsf{G}_i(s,a_i)\mathsf{Q}_{max,i}(s,a_i)$
    si $\mathsf{E}_i(s, \arg\max_{u \in A_i} \pi_i(s,u)) \neq \max_{u \in A_i} \mathsf{E}_i(s,u)$ **alors**
      Choisir $a_{max} \in \arg\max_{u \in A_i} \mathsf{E}_i(s,u)$ aléatoirement
      $\forall b \in A_i \quad \pi_i(s,b) \leftarrow \begin{cases} 1 & \text{si } b = a_{max} \\ 0 & \text{sinon} \end{cases}$ ▷ `sélection d'équilibres`
    $s \leftarrow s'$
fin

7.5.4 Algorithme *Swing between Optimistic or Neutral*

La mesure globale de la fréquence distante est mieux adaptée au cadre des jeux de Markov. Nous utilisons cette mesure de la fréquence dans un nouvel algorithme baptisé *Swing between Optimistic or Neutral* (SOoN) et décrit à l'algorithme 12. On y retrouve l'ensemble des fonctions définies au §7.5.1. La différence principale réside dans l'utilisation de la fréquence distante G définie précédemment (7.16) pour l'interpolation linéaire de l'heuristique d'évaluation des actions. Le principe de fluctuation des évaluations entre des évaluations optimistes selon le Q-learning distribué et des évaluations moyennes selon le Q-learning décentralisé est à l'origine du nom donné à l'algorithme.

Le test de cet algorithme SOoN sur le jeu de Markov d'équipe à deux agents de la figure 7.4, où la coordination de deux agents est particulièrement difficile, donne les

2. Principe du *forward-view* expliqué par [SB98].

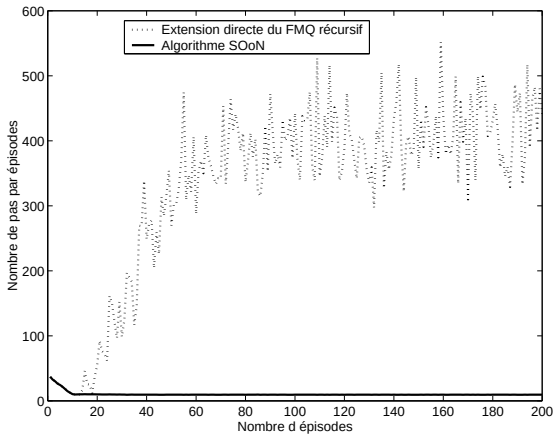


FIGURE 7.6 – Expériences sur le jeu de Markov d'équipe à deux agents de la figure 7.5 avec l'algorithme SOoN (moyennées sur 200 essais indépendants). Le jeu a 10 états et on choisit $\alpha = 0,1$, $\gamma = 0,9$, $\alpha_f = 0,05$, $\alpha_g = 0,3$, $\epsilon = 0,05$.

résultats de la figure 7.6. L'utilisation de la fréquence distante permet à des agents indépendants de réussir leur coordination dans ce jeu. En effet, même si dans un état s_k, la fréquence immédiate de l'action b est élevée, la fréquence distante de l'action optimale est faible et l'évaluation de l'action optimale n'est pas faussée.

Des résultats obtenus avec cet algorithme sur divers jeux de Markov multiagents seront détaillés à la section 7.6.

Discussion

Une forme d'exploration originale pourrait être appliquée à cet algorithme. En effet, la seule forme d'exploration actuelle consiste à tester des actions non-gloutonnes selon leur évaluation E_i. Les agents pourraient aussi tester de temps en temps des actions gloutonnes selon les tables $\mathsf{Q}_{max,i}$. Ainsi, un agent pourrait savoir si sa fréquence est anormalement trop faible et s'il ferait mieux d'évaluer ses actions de manière plus optimiste.

7.5.5 Conclusion

L'algorithme *Swing between Optimistic or Neutral* (SOoN) a été introduit et étudié dans cette section. Grâce au calcul d'une fréquence distante et à une nouvelle heuristique pour évaluer les couples d'état-action, cet algorithme est adapté aux jeux de Markov et surmonte les facteurs de non-coordination évoqués dans ce mémoire. Sa robustesse face à l'exploration a aussi été discutée. Dans les environnements déterministes et si les agents se coordonnent, l'algorithme SOoN est proche de celui du Q-learning distribué qui est assuré de converger dans les jeux de Markov déterministes. Les facteurs de non-coordination dans ces jeux sont donc surmontés. Par contre, dans les environnements stochastiques,

aucun algorithme ne réussit parfaitement la coordination. La solution proposée à travers l'algorithme SOoN est d'évaluer les état-actions à l'aide d'une interpolation linéaire entre les évaluations du Q-learning décentralisé et celles du Q-learning distribué. Ainsi, l'algorithme SOoN s'adapte automatiquement à travers le calcul de la fréquence à des évaluations optimistes ou moyennes.

7.6 Résultats sur des jeux de Markov

Nous testons maintenant les deux algorithmes présentés dans ce chapitre : le Q-learning hystérétique et le SOoN, dans divers jeux de Markov d'équipe. Nous comparons aussi les résultats obtenus avec les algorithmes existants présentés dans l'état de l'art (*cf.* §6). Pour chaque expérimentation, les algorithmes du Q-learning décentralisé, Q-learning distribué, WoLF-PHC, Q-learning hystérétique et SOoN sont comparés. Les jeux de Markov utilisés présentent différents niveaux de difficulté selon le nombre d'agents (2 et plus), les perceptions (totales ou partielles), et les facteurs de non-coordination (équilibres cachés, sélection d'équilibres, environnements déterministes ou stochastiques). Pour chacun d'entre eux, nous précisons les enjeux de l'apprentissage. Une description détaillée est aussi disponible à l'annexe B.

Pour chaque expérimentation et par souci d'équité, la méthode de décision ϵ-greedy et une stratégie stationnaire fixant une exploration globale de $\psi = 0,1$ sont utilisées pour tous les algorithmes. Cela permet notamment de vérifier la **robustesse face à l'exploration** des différentes méthodes.

7.6.1 Jeu de coordination de Boutilier

Le jeu de coordination à deux agents de Craig Boutilier [Bou99] est détaillé à l'annexe B.1. Ce jeu présente 2 agents et 6 états mais différents facteurs de non-coordination peuvent y être combinés. Le problème de sélection d'équilibres est présent car les agents doivent se coordonner et choisir la même action jointe optimale dans un des états. Ces équilibres optimaux peuvent de plus être cachés selon le choix de la pénalité k. Enfin, on peut choisir une version stochastique de ce jeu qui présente une récompense bruitée dans un des états absorbants.

Un épisode se termine lorsque les agents atteignent un état absorbant. Un cycle consiste en 50000 épisodes d'apprentissage. A la fin de chaque cycle, on détermine si la politique jointe gloutonne est optimale. Les résultats sont donnés à la table 7.7.

Dans le jeu déterministe avec $k = 0$, tous les algorithmes réussissent la coordination des agents indépendants sur une des politiques jointes optimales. De même, dans la version stochastique de ce jeu et avec $k = 0$, tous les algorithmes excepté le Q-learning distribué réussissent la coordination. Il est donc à noter que le Q-learning décentralisé et hystérétique notamment réussissent ici avec une stratégie stationnaire.

TABLE 7.7 – Pourcentage de cycles convergeant vers l'une des politiques jointes optimales dans le jeu de coordination de Boutilier (moyenne sur 50 cycles indépendants). On choisit $\alpha = 0,1$, $\gamma = 0,9$, $\delta_{perdre} = 0,006$, $\delta_{gagner} = 0,003$, $\alpha_f = 0,01$, $\alpha_g = 0,03$, $\epsilon = 0,05$. $\beta = 0,01$ dans les jeux déterministes et $\beta = 0,05$ dans les jeux stochastiques. A chaque fin d'un épisode, on détermine si l'action jointe gloutonne est optimale.

	Jeu de coordination de Boutilier			
	déterministe		stochastique	
	$k = 0$	$k = -100$	$k = 0$	$k = -100$
Q-learning décentralisé	100%	6%	100%	12%
Q-learning distribué	100%	100%	0%	0%
WoLF-PHC	100%	36%	100%	40%
Q-learning hystérétique	100%	100%	100%	30%[70%]
SOoN	100%	100%	100%	96%

Lorsque des pénalités de non-coordination sont introduites dans le jeu déterministe, le Q-learning décentralisé et le WoLF-PHC sont mis en défaut face aux équilibres cachés. Le Q-learning distribué, le Q-learning hystérétique et le SOoN réussissent la coordination des agents. Par contre, dans la version stochastique avec $k = -100$, le SOoN est l'unique algorithme qui approche les 100% de convergence. Le Q-learning hystérétique ne fait pas mieux que l'algorithme du WoLF-PHC.

Les meilleurs résultats de convergence sont donc obtenus avec le SOoN. Dans les jeux déterministes, il obtient les mêmes résultats que le Q-learning distribué. Dans le jeu stochastique sans pénalités, ses performances sont comparables à celles du Q-learning décentralisé. Dans le jeu stochastique avec pénalités, il surpasse largement tous les autres algorithmes. L'interpolation linéaire pour évaluer les actions est donc ici une bonne heuristique de mesure des valeurs réelles des actions.

7.6.2 Jeu de poursuite à deux agents

Dans le jeu de poursuite à deux agents [BJD86, KV04, JRK06], deux prédateurs doivent coordonner leurs actions à travers une combinaison spécifique de leurs actions individuelles afin de capturer une unique proie dans un labyrinthe toroïdal de taille 10×10. Ce jeu est détaillé à l'annexe B.2. Des pénalités sont associées à une non-coordination des actions. Les équilibres optimaux sont donc risqués. L'environnement est stochastique car la proie présente un comportement aléatoire. On teste deux instances de ce jeu : avec ou sans incertitude sur le résultat des actions des prédateurs.

Un épisode consiste en 1000 pas. On distingue les épisodes d'apprentissage des épisodes d'exploitation pendant lesquels les agents suivent la politique gloutonne apprise pendant les épisodes d'apprentissage. Après chaque épisode d'apprentissage est lancé un épisode d'exploitation. Pour chaque épisode d'exploitation, on calcule deux choses : le

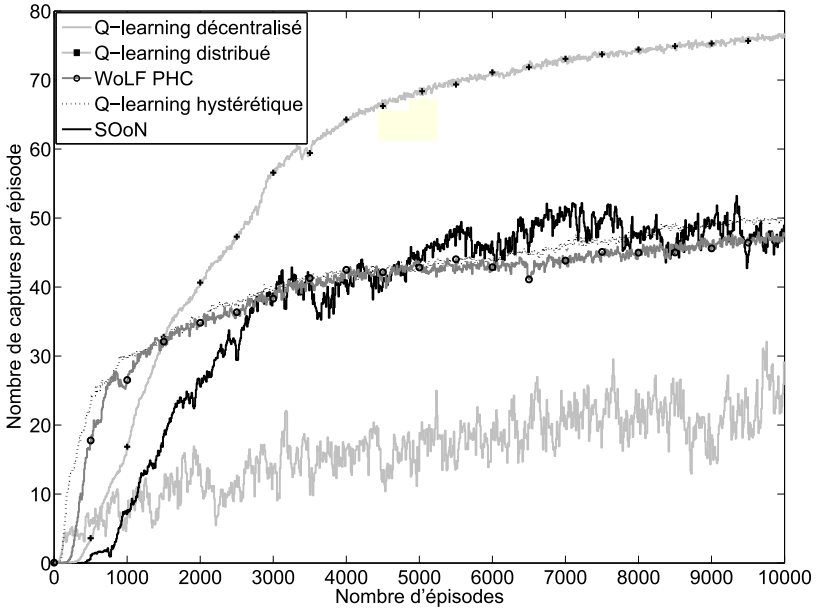
nombre de captures par épisode et la somme des récompenses reçues par un agent pendant l'épisode. Le nombre de captures par épisode permet de comparer les politiques gloutonnes apprises avec chaque algorithme. La somme des récompenses reçues permet de constater s'il y a beaucoup de collisions. Les résultats obtenus sur une série de 10000 épisodes d'exploitation avec tous les algorithmes sont tracés à la figure 7.7 (sans incertitude sur les actions) et à la figure 7.8 (avec incertitude sur les actions).

Dans la version sans incertitude sur le résultat des actions (*cf.* figure 7.7), on constate que :

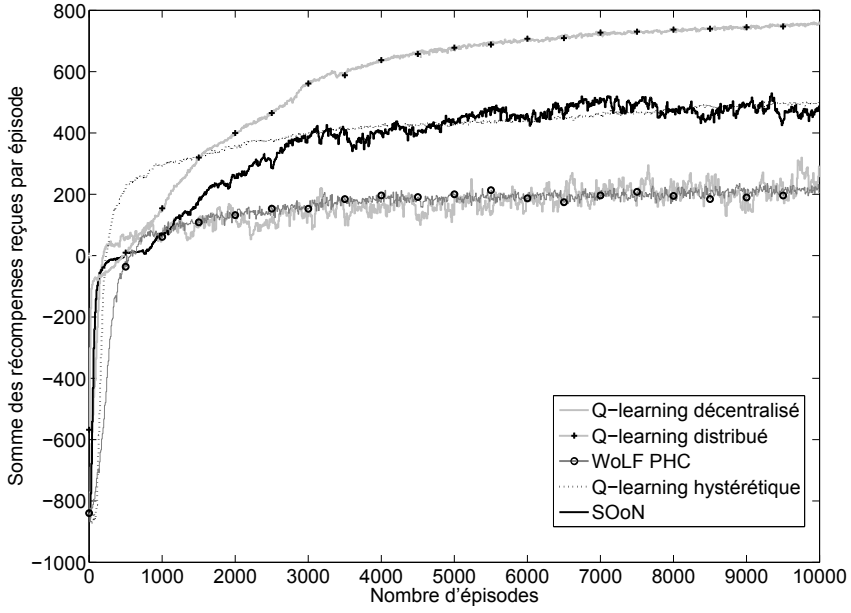
- les politiques gloutonnes apprises avec le **Q-learning décentralisé** sont d'une part bruitées, et d'autre part, elles réalisent difficilement la coordination. La *non-robustesse de cet algorithme face aux équilibres optimaux cachés* est ici illustrée,
- les politiques gloutonnes permettant le plus de captures par épisode sont celles apprises avec le **Q-learning distribué**. Or, des agents optimistes ne parviennent pas à se coordonner dans un environnement stochastique. *La stochasticité induite par le comportement aléatoire de la proie est trop faible pour mettre en défaut les agents optimistes*,
- les politiques apprises avec le WoLF-PHC, le Q-learning hystérétique et le SOoN permettent d'obtenir un nombre de captures par épisode similaire, avec un léger retard au début de l'apprentissage avec le SOoN. Cependant, pour un même nombre moyen de captures, les politiques apprises avec le **WoLF-PHC** engendrent une somme de récompenses inférieure. *Le* ***Q-learning hystérétique*** *et le* ***SOoN*** *réussissent mieux à éviter les collisions*. On peut notamment remarquer que les performances du Q-learning hystérétique et du SOoN sont très proches, avec des politiques gloutonnes plus stables concernant le Q-learning hystérétique.

Dans la version avec incertitude sur le résultat des actions (*cf.* figure 7.8), on constate que :

- les politiques gloutonnes apprises avec le **Q-learning décentralisé** réalisent très difficilement la coordination avec en moyenne 2 captures par épisode,
- le **WoLF-PHC** et le **Q-learning hystérétique** permettent d'obtenir un nombre équivalent de captures par épisode, mais le Q-learning hystérétique engendre moins de collisions,
- malgré la stochasticité induite par le comportement aléatoire de la proie et l'incertitude sur le résultat des actions, les politiques gloutonnes apprises avec le **Q-learning distribué** permettent de réaliser la coordination avec en moyenne 45 captures par épisodes. Cependant, les collisions sont peu évitées,
- les meilleurs résultats sont obtenus avec le **SOoN** qui, après 5000 épisodes, surpasse le Q-learning distribué. On distingue deux phases : une première phase pendant laquelle le SOoN se comporte de manière similaire au Q-learning distribué (500 premiers épisodes), puis de manière similaire au Q-learning décentralisé (2000 épisodes suivants). Cette première phase est en quelque sorte une *phase d'adaptation à l'environnement* où les évaluations « oscillent » entre des évaluations optimistes et moyennes. La deuxième phase correspond au cas où le SOoN s'est adapté et

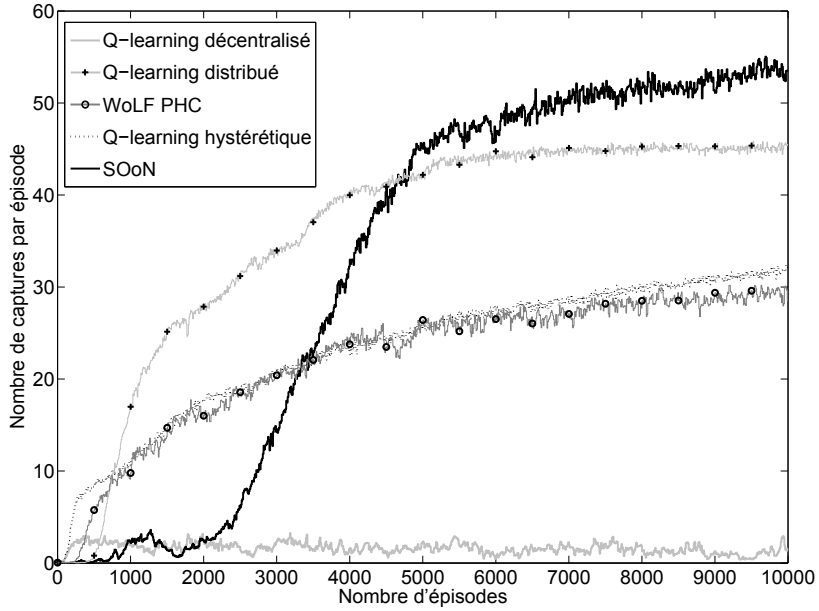


(a)

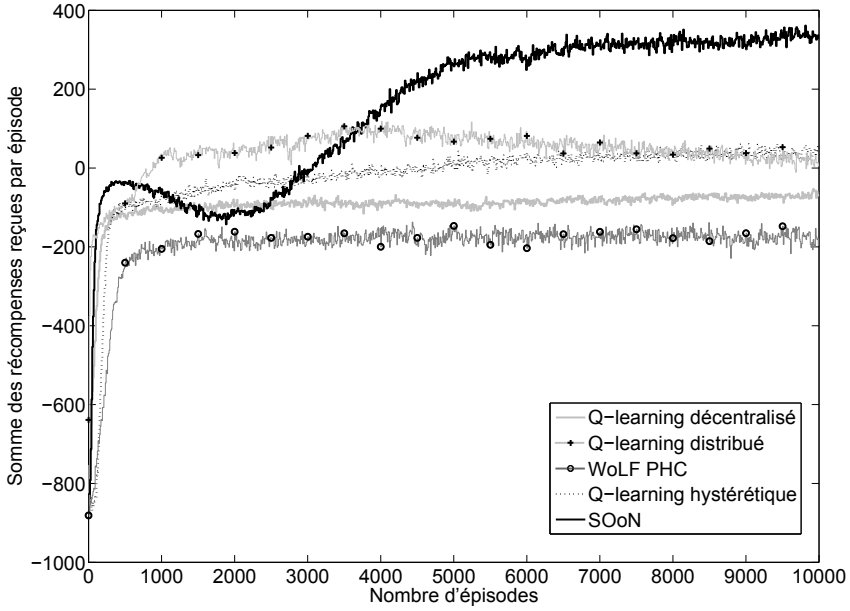


(b)

FIGURE 7.7 – Résultats obtenus avec le jeu de poursuite à 2 agents **sans incertitude sur le résultat des actions** (moyenne sur 20 essais indépendants) a) Nombre de pas par épisode. b) Somme des récompenses reçues par épisode par un agent. Les paramètres choisis sont $\alpha = 0,1$, $\beta = 0,01$, $\alpha_f = 0,01$, $\alpha_g = 0,3$, $\delta_{perdre} = 0,06$, $\delta_{gagner} = 0,03$ et les valeurs initiales de $\mathsf{Q}_{i,max}$ sont -100.



(a)



(b)

FIGURE 7.8 – Résultats obtenus avec le jeu de poursuite à 2 agents **avec incertitude su le résultat des actions** (moyenne sur 20 essais indépendants) a) Nombre de pas par épisode. b) Somme des récompenses reçues par épisode par un agent. Les paramètres choisis sont $\alpha = 0,1$, $\beta = 0,01$, $\alpha_f = 0,01$, $\alpha_g = 0,3$, $\delta_{perdre} = 0,06$, $\delta_{gagner} = 0,03$ et les valeurs initiales de $\mathsf{Q}_{i,max}$ sont -100.

réussit alors la coordination : le nombre de captures par épisode augmente alors et dépasse tous les autres résultats obtenus.

De manière générale, on peut remarquer que les politiques apprises avec le Q-learning décentralisé ne permettent pas un grand nombre de captures par épisode mais néanmoins, la somme des récompenses reçues est proche de celle obtenue avec le WoLF-PHC (qui réalise quant à lui de meilleurs résultats concernant le nombre de captures). Les politiques apprises avec le Q-learning décentralisé évitent donc mieux les collisions que celles apprises avec le WoLF-PHC.

Le SOoN permet une **adaptation automatique à l'environnement**. Dans le premier cas de figure, il fait mieux que le Q-learning décentralisé mais moins bien que le Q-learning distribué : il semble donc plus sensible à la stochasticité induite par le comportement aléatoire de la proie que le Q-learning distribué. Par contre, dans le second cas de figure, il réussit à surpasser le Q-learning distribué et le Q-learning décentralisé : la coordination est meilleure car elle permet plus de captures par épisodes et moins de collisions.

7.6.3 Jeu de poursuite à quatre agents

Dans le jeu de poursuite à quatre agents, quatre prédateurs doivent coordonner leurs actions afin d'encercler une unique proie dans un labyrinthe toroïdal de taille 7×7 [BJD86, Buf03]. Ce jeu est détaillé à l'annexe B.3. Les perceptions des agents souffrent d'une faible résolution, ce qui entraîne du bruit compliquant la coordination des agents.

Un épisode consiste en 1000 pas. On distingue les épisodes d'apprentissage des épisodes d'exploitation pendant lesquels les agents suivent la politique gloutonne apprise pendant les épisodes d'apprentissage. Après chaque épisode d'apprentissage est lancé un épisode d'exploitation. Pour chaque épisode d'exploitation, on trace la *courbe d'apprentissage* du nombre de captures par épisode. Cela permet de comparer les politiques gloutonnes apprises avec chaque algorithme. Les résultats obtenus sur une série de 80000 épisodes d'exploitation avec tous les algorithmes sont tracés à la figure 7.9.

Nous constatons plusieurs choses sur la courbe d'apprentissage de la figure 7.9 :
- le **Q-learning distribué** a beaucoup de difficultés à réaliser la coordination des agents. En effet, les agents optimistes réussissent à capturer la proie pour la première fois après 48000 épisodes, et après 80000 épisodes, ils capturent la proie environ 5 fois par épisodes. *La faible résolution des perceptions et la stochasticité que cela engendre dans l'environnement met en défaut les agents optimistes*,
- les agents utilisant le **Q-learning décentralisé** réussissent bien à se coordonner. Les premières captures sont réalisées après 5000 épisodes, et le nombre de captures par épisode augmente assez rapidement avec en moyenne 70 captures par épisode après 80000 épisodes. Malgré la faible résolution dans les perceptions, le Q-learning décentralisé se place en deuxième position. *Il permet donc d'obtenir dans cet en-*

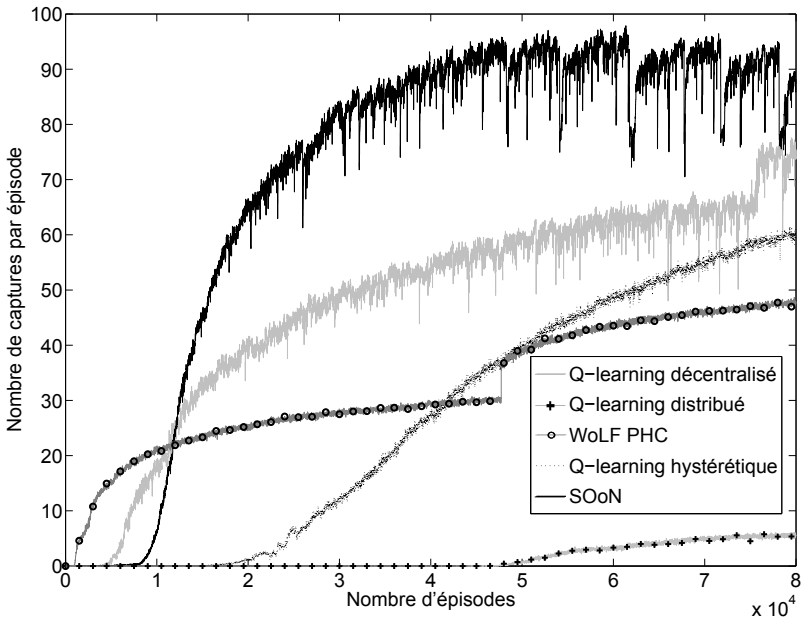


FIGURE 7.9 – Résultats obtenus avec le jeu de poursuite à 4 agents (moyenne sur 5 essais indépendants). Les paramètres choisis sont $\alpha = 0,3$, $\beta = 0,03$, $\alpha_f = 0,03$, $\alpha_g = 0,3$, $\delta_{perdre} = 0,06$, $\delta_{gagner} = 0,03$ et les valeurs initiales de $\mathsf{Q}_{i,max}$ sont 0.

vironnement stochastique des résultats de coordination satisfaisant comparés aux autres algorithmes,

- le **WoLF-PHC** est le plus rapide à réaliser des captures en début d'apprentissage. Cependant, la courbe stagne et le nombre de captures par épisode augmente très lentement,
- le **Q-learning hystérétique** réussit les premières captures après environ 17000 épisodes, ce qui est assez tard comparé aux autres algorithmes (excepté le Q-learning distribué). Par contre, sa courbe d'apprentissage augmente rapidement. Elle dépasse celle du WoLF-PHC et les agents hystérétiques rattrapent presque les agents utilisant le Q-learning décentralisé après 80000 épisodes avec en moyenne 60 captures par épisode,
- les agents utilisant le **SOoN** sont les plus performants. On distingue deux phases d'apprentissage : une première phase où les agents s'adaptent à l'environnement. Ils réalisent les premières captures assez tardivement comparé au Q-learning décentralisé et au WoLF-PHC car ils sont optimistes au début de leur apprentissage. Dans la seconde phase, leur apprentissage de la coordination est très rapide. Après 50000 épisodes, ils réussissent environ 90 captures par épisode, ce qui est le meilleur résultat obtenu avec tous ces algorithmes. *Ils se sont donc adaptés automatiquement à l'environnement stochastique* et réussissent ensuite la coordination de manière

plus performante que les autres algorithmes, et spécialement que le Q-learning décentralisé.

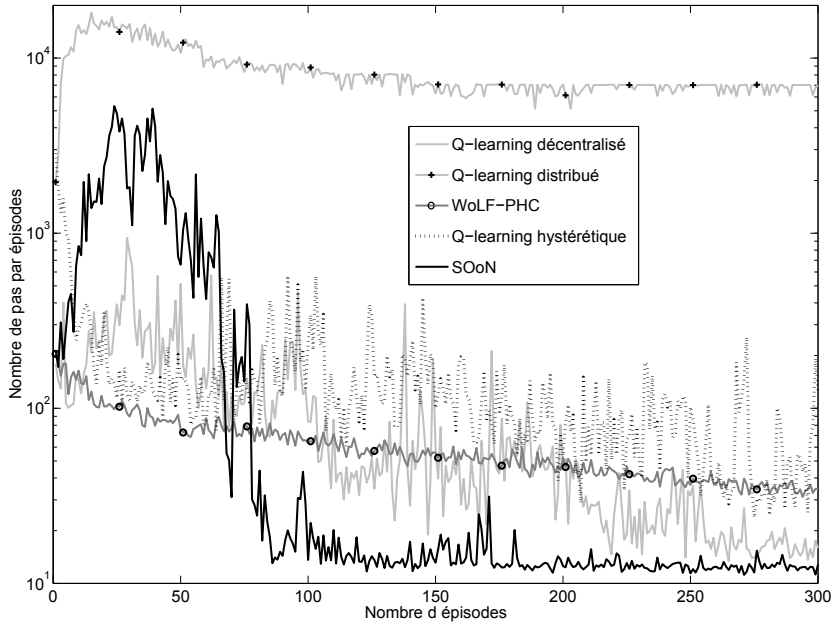
De manière générale, on peut aussi remarquer que les politiques gloutonnes apprises avec le SOoN et le Q-learning décentralisé sont bruitées, contrairement aux politiques apprises avec le WoLF-PHC ou le Q-learning hystérétique. Les agents utilisant le SOoN réalisent donc une coordination plus rapide mais ils sont aussi plus sensible à la stochasticité de l'environnement.

7.6.4 *Smart surface* discrète

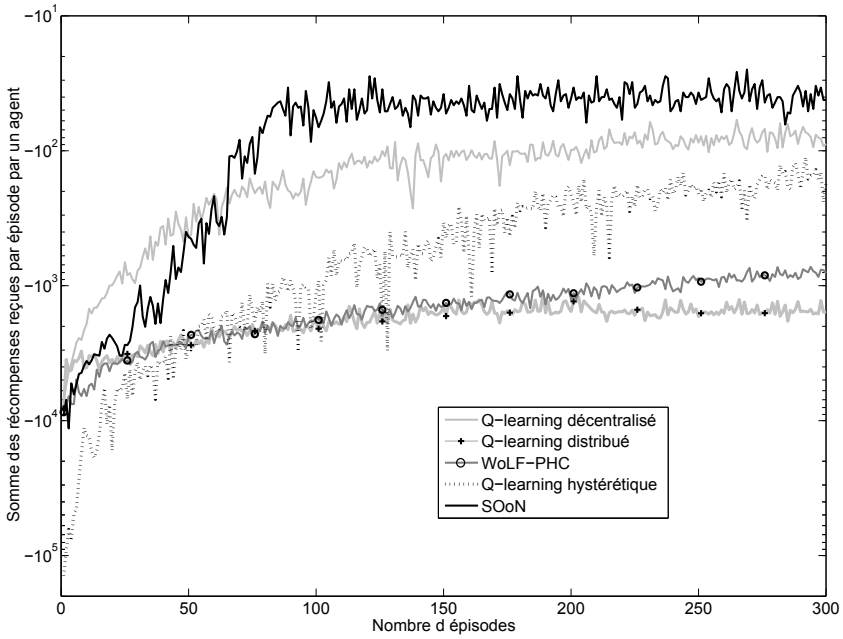
Dans ce *benchmark*, qui est une version discrète simplifiée de notre application réelle présentée au chapitre suivant, 270 agents (ou actionneurs) sont organisés en matrice. Ils doivent se coordonner afin de déplacer une pièce, située sur la surface, à une position donnée. Ce jeu est détaillé à l'annexe B.4. La pièce doit atteindre le plus rapidement possible la position désirée sans s'approcher des bords de la surface, ce qui entraîne une pénalité.

Un épisode débute avec la pièce dans sa position initiale et se termine lorsque la pièce atteint la position désirée (état absorbant). Pour chaque épisode, nous traçons le nombre de pas nécessaire à la pièce pour atteindre la position désirée, ainsi que la somme des récompenses reçues par un agent. Les *courbes d'apprentissage* obtenues sur 300 épisodes sont données à la figure 7.10. Le nombre de pas par épisodes permet de savoir si la politique apprise par les agents est proche de la meilleure politique obtenue « à la main ». La somme des récompenses permet notamment de vérifier si les bords de la surface sont bien évités, étant donné que se rapprocher de ces bords entraîne une pénalité. Nous précisons que les courbes tracées sont obtenues pendant des épisodes d'apprentissage ; elles prennent donc en compte les actions d'exploration des agents ($\psi = 0,1$). La politique optimale réalise 3 pas par épisodes et la somme des récompenses reçues est de 10 (l'incertitude sur les actions n'est pas considérée).

Nous constatons plusieurs choses avec les courbes d'apprentissage de la figure 7.10 :
- avec le **Q-learning distribué**, le nombre de pas par épisode diminue très peu et la somme des récompenses reçues augmente aussi très peu. D'une part, *l'environnement est stochastique ce qui met en défaut les agents optimistes* ; d'autre part, *les agents optimistes ignorent les pénalités reçues et donc n'apprennent pas à éviter les bords de la surface*,
- les agents qui suivent le **Q-learning décentralisé** effectuent après 300 épisodes un peu moins de 20 pas par épisodes. La somme des récompenses reçues par un agent est alors d'environ -100 : les bords sont donc approchés en moyenne 4 fois par épisodes. Néanmoins, les politiques apprises ne sont pas stables : sur l'ensemble des épisodes d'apprentissage, la courbe du nombre de pas par épisode est fortement bruitée et converge lentement, alors que la somme des récompenses reçues est beaucoup plus stable et diminue rapidement. En effet, *à cause des pénalités et de*



(a)



(b)

FIGURE 7.10 – Résultats obtenus avec la *smart surface* discrète (moyenne sur 100 essais indépendants) a) Nombre de pas par épisode. b) Somme des récompenses reçues par épisode par un agent. Les paramètres choisis sont $\alpha = 0,1$, $\beta = 0,01$, $\alpha_f = 0,01$, $\alpha_g = 0,3$, $\delta_{perdre} = 0,06$, $\delta_{gagner} = 0,03$ et les valeurs initiales de $\mathsf{Q}_{i,max}$ sont -50.

la non-robustesse face à l'exploration des agents, les politiques individuelles sont « détruites » . On constate d'ailleurs que les agents se coordonnent pour éviter les bords : la pièce tourne alors en rond dans la zone sans pénalités. La coordination pour amener la pièce à l'état final est plus difficile,

- les **agents hystérétiques** ont un comportement proche de celui des agents apprenant avec le Q-learning décentralisé. Cependant, comme ils accordent moins d'importance aux pénalités, la somme des récompenses reçues augmente plus lentement et ils sont moins performants pour éviter les bords de la surface, ce qui entraîne une convergence plus lente du nombre de pas par épisodes,
- les politiques apprises avec le **WoLF-PHC** sont les plus stables mais leurs convergences sont lentes : après 300 épisodes, environ 35 pas par épisodes sont effectués et un agent reçoit une somme de récompenses d'environ -800, *i.e.* à peine supérieure à celle des agents optimistes,
- avec le **SOoN** et après 300 épisodes, le nombre de pas par épisode est d'environ 12 (l'optimal est de 3 pas par épisodes), et la somme des récompenses reçues par un agent par épisode est autour de -40, *i.e.* qu'en moyenne, les bords ne sont approchés qu'une seule fois. De plus, les courbes d'apprentissage se stabilisent après environ 100 épisodes. *Ces agents sont donc les plus performants.* On distingue deux phases. Au début de l'apprentissage, le nombre de pas par épisodes augmente. En effet, les agents qui suivent le SOoN ne parviennent pas à réaliser leur coordination car les agents sont optimistes au début de l'apprentissage. Après une cinquantaine d'épisodes, le nombre de pas par épisodes diminue et les bords sont évités donc les agents sont de moins en moins optimistes. C'est donc ici une première phase d'*adaptation automatique à l'environnement.* Au final, le SOoN obtient de meilleures performances que le Q-learning décentralisé, donc les évaluations se sont automatiquement ajustées entre des évaluations optimistes et moyennes, et celles-ci sont les plus proches des optimales.

7.6.5 Synthèse sur les expérimentations en jeux de Markov d'équipe

Ces expérimentations ont permis d'illustrer le comportement des algorithmes du SOoN et du Q-learning hystérétique dans divers jeux de Markov d'équipe avec différents niveaux de difficultés. Notamment, nous avons pu constater l'**adaptation automatique à l'environnement** effectuée par le SOoN. Deux phases se distinguent lors de l'apprentissage. Une première phase d'adaptation où les évaluations sont tout d'abord optimistes, puis le sont moins si l'environnement est stochastique. Et une seconde phase où l'adaptation est réalisée, c'est-à-dire que les évaluations se sont ajustées entre des évaluations optimiste et moyennes, et où les agents réussissent alors à se coordonner de manière souvent plus performante qu'avec les autres méthodes. Il est important de préciser que ces deux phases sont automatiques. Ainsi, l'*interpolation linéaire pour les évaluations est une bonne heuristique proche des valeurs réelles.* Le SOoN obtient en général de meilleures performances que le Q-learning décentralisé ou que le Q-learning distribué.

Concernant le Q-learning hystérétique, les politiques apprises permettent de réaliser dans une certaine mesure la coordination, mais la convergence est lente par rapport au SOoN. Par contre, les politiques gloutonnes apprises sont plus stables que celles apprises avec le SOoN.

7.7 Conclusion et perspectives

L'objectif de ce chapitre était de proposer des algorithmes robustes face à l'exploration et capables de surmonter les facteurs de non-coordination dans les jeux de Markov d'équipe. Pour cela, une ligne directrice de ce chapitre est de permettre à un agent indépendant de correctement évaluer ses actions individuelles. En partant du fait que les valeurs réelles de chaque action se situent entre les valeurs optimistes et moyennes, deux algorithmes ont été développés. Le premier est le **Q-learning hystérétique**, qui permet de doser l'optimisme des agents grâce à deux vitesses d'apprentissage. Le second est l'algorithme ***Swing between Optimistic or Neutral*** (**SOoN**), capable de découpler les diverses causes de bruit dans un jeu de Markov d'équipe faiblement bruité. Une heuristique utilisant une fréquence d'occurrence permet aux évaluations des actions de se **régler automatiquement** entre les évaluations optimistes et moyennes selon la stochasticité détectée dans le jeu. L'algorithme SOoN surmonte ainsi les principaux facteurs de non-coordination évoqués dans ce mémoire et est de plus robuste face à l'exploration des autres. Des résultats sur de multiples *benchmarks* multiagents ont permis de comparer les performances de ces deux algorithmes aux principaux algorithmes de la littérature. Ils ont notamment illustré la première phase d'adaptation automatique des évaluations puis la phase de coordination. Ils ont aussi permis de confirmer que l'interpolation linéaire est une bonne heuristique d'évaluation des actions proche des valeurs réelles.

Dans ce chapitre a aussi été précisée une notion importante dans les systèmes multiagents : l'**exploration globale**. Celle-ci permet de quantifier le bruit dans le système dû à l'exploration de tous les agents. Ce bruit est un paramètre important qui peut notamment mettre en défaut un agent indépendant. En effet, le bruit dû à un comportement aléatoire des autres rend difficile, voire impossible si le bruit est trop important, une évaluation correcte des valeurs d'un état par un agent indépendant.

La **robustesse** des algorithmes d'apprentissage pour agents indépendants face à cette exploration globale est nécessaire ; c'est d'ailleurs un des enjeux de l'apprentissage par renforcement d'agents indépendants. En effet, si cette robustesse n'est pas assurée, on peut assister à des phénomènes de « destruction » des politiques apprises ou des fréquences d'occurrence par exemple. Des méthodes ont été proposées dans ce chapitre pour choisir correctement divers paramètres d'apprentissage afin d'assurer cette robustesse. Cependant, une autre solution se dégage de ce constat. Il serait intéressant de permettre aux agents de savoir si les actions des autres sont des actions d'exploration ou d'exploitation (sans connaître exactement quelles sont ces actions). Cette forme de communication est simple à mettre en oeuvre même avec un grand nombre d'agents :

d'une part, l'information à transmettre est binaire (« j'explore et donc j'émets » ou « je n'explore pas et j'écoute »), et d'autre part, elle ne nécessite qu'un seul canal de communication partagé par tous les agents (réseau de *diffusion*). Cette forme simple de communication entre agents leur permettrait de découpler aisément les diverses causes de bruit. Les agents qui exploitent pourraient alors suspendre leur mise à jour lorsqu'un ou plusieurs de leurs congénères explorent. Cette méthode pourrait de plus être appliquée facilement à de nombreux algorithmes.

Chapitre 8

Application à la commande d'un système distribué de micromanipulation

Dans ce chapitre une application de nos travaux à la commande d'un système distribué de micromanipulation, appelé smart surface*, est proposée. Le prototype de surface pneumatique déjà existant ainsi que le modèle de convoyage d'objets en 2D développé sont présentés. L'approche par apprentissage par renforcement est ensuite comparée à une méthode classique de contrôle des systèmes distribués de micromanipulation, et validée sur une tâche de positionnement. Enfin, une extension de nos algorithmes à la commande décentralisée de la* smart surface *dans un cas partiellement observable est proposée et testée sur une tâche de convoyage.*

8.1 Introduction

Ce chapitre présente l'application à la commande d'un système distribué de micromanipulation des méthodes décentralisées par apprentissage par renforcement développées dans ce mémoire. Cette application, appelée *smart surface*, se place dans le cadre d'un projet de l'agence nationale de la recherche (ANR PSIROB). L'objectif de ce projet est la conception, le développement et le contrôle d'un système de microconvoyage et micropositionnement sur coussin d'air de micropièces à l'échelle mésoscopique (μm au mm). Les applications visées par ce système sont la micromanipulation et le tri automatisés de mini et microproduits, dans lesquels les fonctions d'alimentation, de convoyage et de positionnement des composants constituent un challenge important lié aux dimensions de ces pièces.

Le projet *smart surface* s'est orienté vers la réalisation d'un système distribué constitué d'une matrice de microactionneurs pneumatiques, de capteurs et de modules de traitement. La réalisation concrète d'une *smart surface* intégrant des cellules capteur-

calcul-actionneur reste aujourd'hui un challenge. Le projet *smart surface* étant actuellement toujours en cours, nous ne disposons pas encore d'une telle surface totalement intégrée. Par conséquent, deux approches sont envisagées pour tester des méthodes de contrôle décentralisé. La première consiste à développer un modèle approché de la *smart surface* et à tester en simulation les algorithmes de contrôle. La seconde méthode est d'utiliser une *surface active*, c'est-à-dire une matrice distribuée d'actionneurs pneumatiques, déjà existante mais non-intégrée (donc sans capteurs et modules de calcul). Le contrôle décentralisé peut alors être « émulé » à l'aide d'une caméra, d'un mécanisme d'allocation individuelle des actionneurs et d'une unité centrale de calcul.

Ce chapitre est tout d'abord consacré à la présentation des systèmes distribués de micromanipulation de manière générale, puis il se focalise sur le prototype de surface active pneumatique mis à notre disposition. Le modèle de la *smart surface* développé est quant à lui détaillé en deuxième partie de ce chapitre. Avec ces deux méthodes à disposition, nous proposons alors d'expérimenter un contrôle par apprentissage par renforcement décentralisé pour positionner ou convoyer un objet sur la surface active pneumatique. Deux objectifs sont poursuivis :

- tout d'abord, nous souhaitons valider l'approche par apprentissage par renforcement en tant que méthode de contrôle adaptée à ce système. Pour cela, une approche usuelle de contrôle des systèmes distribués de micromanipulation est comparée à l'apprentissage par renforcement décentralisé. Nous nous plaçons pour cela dans le cadre des jeux de Markov d'équipe et les travaux effectués dans cette thèse sont appliqués,
- ensuite, une extension de nos travaux au cadre plus réaliste des observabilités partielles est proposée.

8.2 Les systèmes distribués de micromanipulation

8.2.1 Systèmes de micromanipulation

Les MEMS (Micro Electro Mechanical Systems) sont des microsystèmes électromécaniques actuellement en plein essor. Ils réalisent des fonctions de capteurs et/ou d'actionneurs. Ils sont utilisés dans certaines applications « de tous les jours » telles que les têtes d'imprimantes à jets d'encre, les capteurs *airbag* (coussins gonflables de sécurité) ou les capteurs de pression pour les pneus dits « intelligents » dans l'automobile, ou plus récemment les accéléromètres des manettes à détection de mouvement de la console Nintendo de jeu *Wii*.

De nombreuses voies d'application des MEMS restent cependant à explorer, et parmi les développements futurs, les structures « distribuées » font partie des plus prometteuses. Ce sont des systèmes ou structures constitués de réseaux de capteurs et/ou d'actionneurs distribués, appelés MEMS en réseaux ou **MEMS distribués**. Par exemple, la « libel-

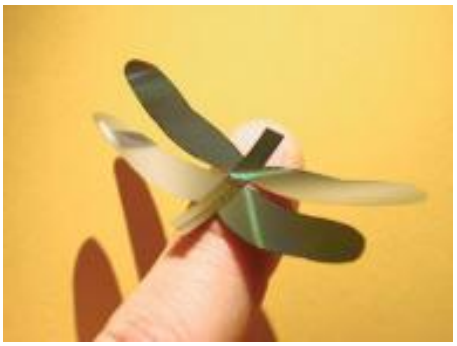


FIGURE 8.1 – Libellule artificielle de la société *SilMach*.

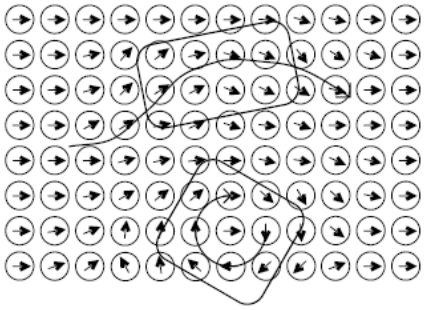


FIGURE 8.2 – Matrice distribuée pour la micromanipulation (translation et rotation illustrées ici) (image extraite de [LMC01]).

lule » artificielle de la société *SilMach*[1] de Besançon possède quatre ailes battantes mues par 180 000 muscles MEMS répartis et distribués en réseaux sur toute la surface de l'aile (*cf.* figure 8.1). Une autre application potentielle des MEMS distribués est l'utilisation en réseaux de microcapteurs sans fils qui permettent l'étude du comportement d'une structure, comme une aile d'avion par exemple.

L'application des MEMS distribués qui nous intéresse concerne les *systèmes de micromanipulation 2D*, et plus précisément la fonction d'alimentation en microcomposants. En anglais, on parle de fonction *feeding* qui consiste à déplacer un micro-objet et à le positionner pour par exemple alimenter une cellule de micromanipulation ou de microassemblage. Dans ce contexte de convoyage et positionnement de micro-objets, des méthodes basées sur des vibrations mécaniques ont été proposées [BGC+98, PHLR08]. L'utilisation de « surfaces actives » à base de réseaux de microactionneurs répond aussi efficacement aux enjeux de la micromanipulation 2D. Ces surfaces actives intègrent une densité importante de microactionneurs par unité de surface. Le principe est celui de la manipulation distribuée [LMC01] : on déplace un objet en combinant les actions individuelles exercées sur celui-ci en différents points de contact par plusieurs actionneurs. Un champ de forces est ainsi créé, comme illustré à la figure 8.2.

Notons ici l'analogie entre ce principe et les notions d'interaction et d'émergence dans les systèmes multiagents. En effet, la manipulation distribuée peut être vue comme un résultat complexe émergeant de l'interaction d'actions effectuées par un ensemble d'actionneurs (ou agents). L'objet manipulé est le support de couplage de ces interactions et le déplacement la résultante d'actions coordonnées de coopération.

Les systèmes distribués de micromanipulation ont de nombreux avantages comparés aux systèmes conventionnels. Tout d'abord, ce sont des systèmes flexibles et robustes

1. Spin-off du CNRS.

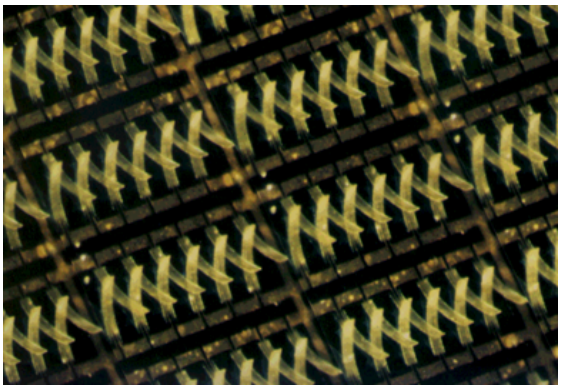


FIGURE 8.3 – Système distribué de micromanipulation « à contact » constitué de microactionneurs en polyimide dont l'effet thermique reproduit le mouvement des cils dans les organismes vivants [AOTF94].

dans le sens où, même si un des actionneurs est défectueux, les autres compenseront et la tâche pourra toujours être réalisée. Ensuite, les actionneurs distribués vont être capable d'assurer plusieurs tâches en parallèle comme le convoyage simultané de multiples objets.

Plusieurs dispositifs de microconvoyage ont été réalisés utilisant des actionneurs thermiques bimorphes [AOTF94, AMF07, ALB+ss], des actionneurs électrostatiques [PFH90, BDMM94, KF94, BDM96, BGC+98, FCMF06] ou encore des actionneurs magnétiques [NWM99]. Ces travaux illustrent la capacité de ces surfaces actives à réaliser des tâches de positionnement et de convoyage de micro-objets.

Deux types de surface active pour la micromanipulation 2D sont distinguées : les systèmes « à contact » (*contact systems*) et les systèmes « sans contact » (*contact-free systems*). Les micromanipulateurs 2D à contact se définissent comme un réseau de microactionneurs dont chaque unité vient en contact avec l'objet si celui-ci est à portée de l'actionneur. Ces systèmes sont généralement constitués de microactionneurs électrostatiques [BDM96] ou de microactionneurs en polyimide dont l'effet thermique reproduit le mouvement des cils dans les organismes vivants [AOTF94, AMF07, ALB+ss] (*cf.* figure 8.3). Les systèmes à contact présentent une bonne précision et une capacité de charge élevée mais leur vélocité est relativement faible.

Les systèmes sans contact se définissent comme un réseau de microactionneurs dont chaque unité produit une force à distance sur l'objet, évitant ainsi tout contact ou frottement mécanique de la surface avec l'objet manipulé. Les surfaces actives pneumatiques où un flux d'air est contrôlé par de multiples microvalves sont des exemples de systèmes sans contact [PFH90, KF94, FCMF06]. Ils ont une bonne vélocité et évitent les problèmes de friction. Ils sont donc potentiellement plus miniaturisables [FCMF06]. Mais leur précision est faible et ils nécessitent des approches de contrôle plus complexes [KWSS01, FCMF06, CZF+07]. Le contrôle de ces surfaces sans-contact sera détaillé au

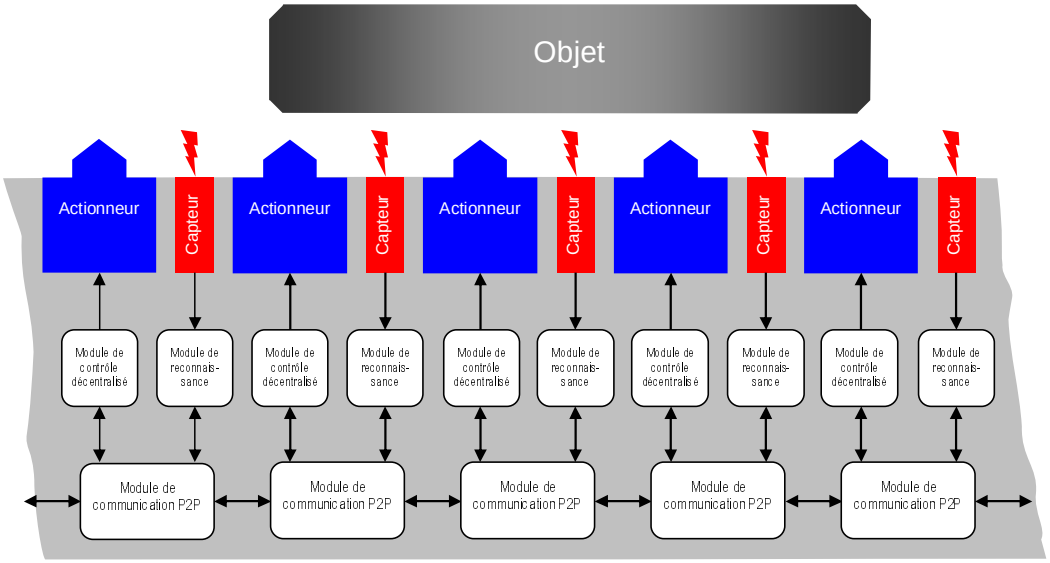


FIGURE 8.4 – Concept de *smart surface*.

§8.4.

Les micromanipulateurs 2D permettent d'atteindre de bonnes performances de fonctionnement, et l'évolution de ces technologies vers des applications industrielles nécessite désormais l'intégration d'éléments de contrôle et de capteurs. L'idée est d'aboutir à un dispositif à base de MEMS *intelligent*, *autonome*, *distribué* et intégrant des cellules capteur-calcul-actionneur. Ces microsystèmes distribués et autonomes sont donc composés de plusieurs cellules intelligentes qui peuvent partager des informations, communiquer entre elles, voire coopérer [Fuj96]. On parle alors de *smart surface* (cf. figure 8.4). Chaque cellule contrôle son propre actionneur et traite les informations de son capteur (présence ou non de la pièce au-dessus de lui par exemple) et les informations échangées avec les autres cellules. La coopération de nombreuses cellules aboutit alors à l'exécution de tâches complexes telles que le positionnement parallèle d'objets multiples. Une *smart surface* peut donc être considérée comme un système multiagent coopératif où chaque cellule est un agent autonome.

8.2.2 Prototype de *surface active* pneumatique

Un premier prototype de *surface active* pneumatique a été développé et mis à notre disposition mi-2008. C'est un dispositif à base de MEMS fabriqués sur silicium. Il a été conçu en 2005 au LIMMS [2]/CNRS-IIS [3] de l'Université de Tokyo au sein du laboratoire du Pr. Fujita (*Fujita Lab.*) par Yves-André Chapuis, maître de conférences HDR à l'Uni-

2. *Laboratory of Integrated Micro Mechatronic Systems.*
3. *Institute of Industrial Science.*

versité Louis Pasteur de Strasbourg.

La *surface active* pneumatique est constituée d'un réseau de microactionneurs organisés en matrice afin de réaliser des opérations de micromanipulation « sans contact », de type fluidique, comme le montre la figure 8.5. Des jets d'air bidirectionnels sont générés à la surface de la matrice. Ils permettent de déplacer un objet positionné dessus. Ces jets d'air orientables ont pour fonction la mise en lévitation de l'objet ainsi que son déplacement. La direction d'un jet d'air est contrôlée par un microactionneur électrostatique.

Un microactionneur est composé de deux couches (*cf.* figure 8.6) :
- une couche fonctionnelle supérieure composée d'un simple orifice. L'air provenant de sous la surface passe par cet orifice, créant à sa sortie un flot d'air (*cf.* figure 8.6a),
- une couche fonctionnelle inférieure composée d'une microvalve mobile actionnée par un microactionneur électrostatique (*cf.* figure 8.6b). L'actionneur électrostatique est composé de deux électrodes fixes qui ont pour objectif de générer des forces dans les deux sens. La microstructure est composée d'une électrode mobile et de poutres suspendues.

Ces deux couches sont fabriquées à partir de *wafers* SOI [4], composés de deux substrats en silicium de même épaisseur 100μm. La taille d'un microactionneur est approximativement de $1 \times 1\ mm^2$. Pour plus de détails sur le processus de fabrication de ces actionneurs, il est conseillé de se référer à l'article [YFF06].

La figure 8.6c montre la vue transversale d'un microactionneur dans deux cas de fonctionnement. Si aucune tension n'est appliquée, la microvalve est en position de « repos » : elle est alignée sur la forme de l'orifice de la couche supérieure. Un léger flot d'air vertical est alors produit qui maintient l'objet en lévitation. Si une tension est appliquée sur une des deux électrodes fixes du microactionneur électrostatique, la microvalve se déplace vers cette électrode. Elle dégage alors une partie de l'orifice supérieur et obstrue l'autre partie, ce qui génère un jet d'air orienté et met en mouvement l'objet. On distingue ainsi les actionneurs verticaux et horizontaux selon l'orientation de l'orifice.

Sur le prototype de surface active pneumatique, les lignes et colonnes de la matrice se composent de microactionneurs à deux directions, comme l'illustre la figure 8.7. Les microactionneurs sont organisés alternativement de manière verticale et horizontale, ce qui permet le convoyage d'un objet en 2 dimensions (2D). L'arrangement de la matrice distribuée se compose de deux parties latérales symétriques, composées de 23 lignes de 8 microactionneurs pneumatiques, et d'une partie centrale de 8 colonnes et 24 microactionneurs pneumatiques. Ce prototype n'est pas totalement décentralisé. Les microactionneurs des parties latérales sont commandés ligne par ligne, et ceux des parties centrales demi-colonne par demi-colonne. On a ainsi 128 entrées de commande. A chacune d'elles, la commande envoyée est distribuée aux 8 (ou 12) microactionneurs de la ligne ou de la

4. *Silicon On Insulator.*

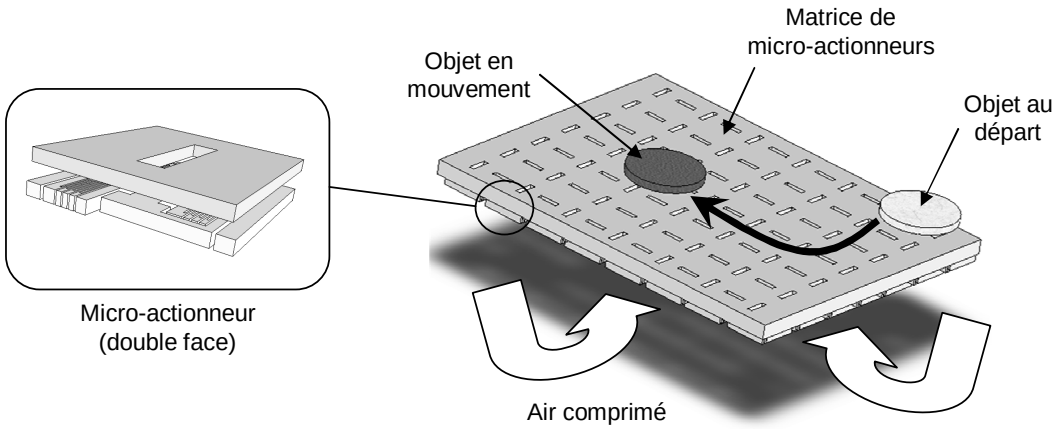


FIGURE 8.5 – *Surface active* pneumatique constituée d'un réseau de microactionneurs en technologie MEMS pour des opérations de micromanipulation « sans contact » de type fluidique.

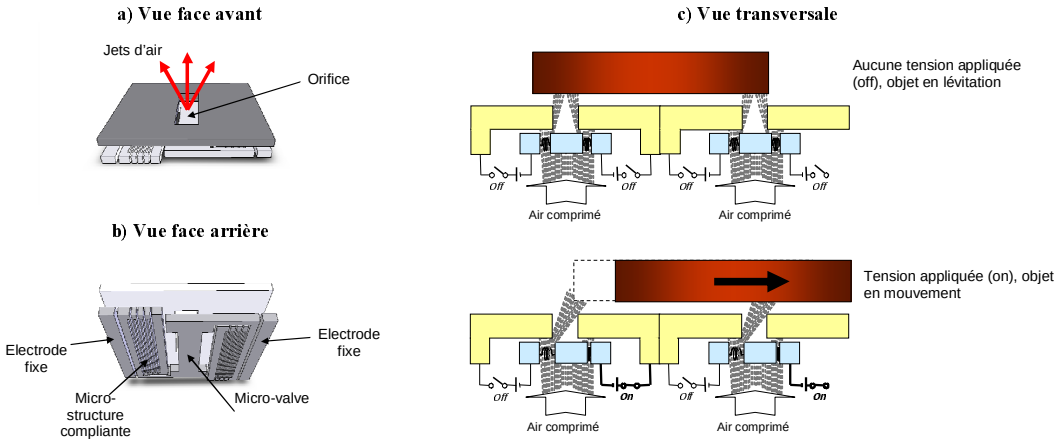


FIGURE 8.6 – Microactionneur sous différentes vues. a) Vue de dessus. b) Vue de dessous. c) Vue transversale dans deux cas de fonctionnement. (images extraites de [YFF06])

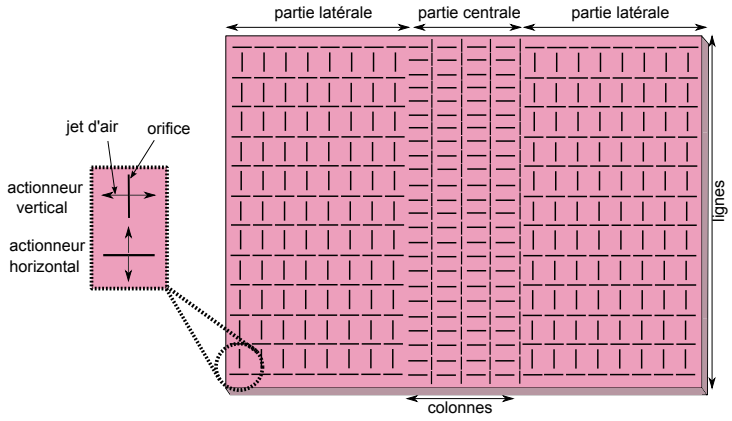


FIGURE 8.7 – Schéma d'organisation des microactionneurs sur la surface (vue de dessus).

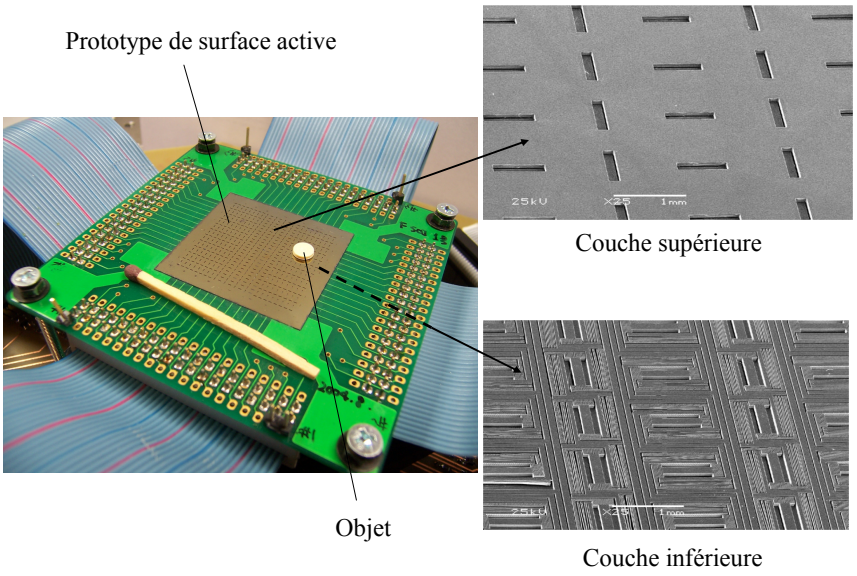


FIGURE 8.8 – Photo de la surface active pneumatique ainsi que les images MEB de la couche supérieure et de la couche inférieure.

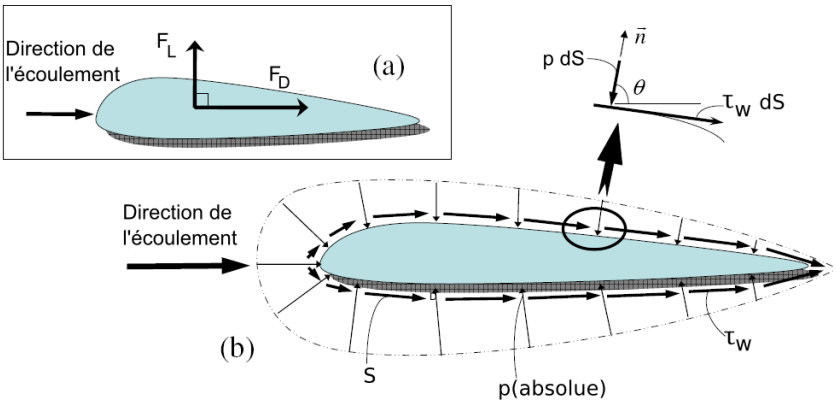


FIGURE 8.9 – Forces agissant sur un « corps à pointe » a) Résultantes des forces de « traînée » F_D et de « portance » F_L b) Forces de pression et tangentielles agissant sur une surface élémentaire.

colonne.

La figure 8.8 présente une photo du prototype de surface active pneumatique ainsi que les images des faces avant et arrière de la surface obtenues par microscope électronique à balayage (MEB) [CZFH08]. La matrice est constituée de 560 microactionneurs répartis sur une surface active d'environ $35 \times 35\ mm^2$.

8.3 Modèle simple du convoyage 2D de la *smart surface*

Afin de pouvoir tester rapidement et d'avoir une première approximation des performances de nos algorithmes d'apprentissage, un modèle simple de la *smart surface* a été développé. **L'objectif de ce modèle n'est pas d'être réaliste**, car une modélisation multi-domaine de ce dispositif à base de MEMS distribués reste très complexe [Zho07, CZFH08]. **Le modèle développé vise à générer des problématiques de contrôle équivalentes à celles que l'on pourrait rencontrer avec la *smart surface*.** Il se base sur un modèle simple des forces fluidiques exercées par un flux d'air sur un corps situé dans ce flux.

8.3.1 Introduction aux forces fluidiques

Un fluide peut exercer des forces et des moments mécaniques sur un corps dans plusieurs directions. On distingue généralement deux types de forces fluidiques, représentées sur la figure 8.9a :

- la force de « traînée », ou *drag force*, notée F_D, qui s'exerce sur un corps dans la direction de l'écoulement du fluide,
- la force de « portance », ou *lift force*, notée F_L, qui s'exerce sur un corps dans la direction normale à l'écoulement du fluide.

Ces deux forces sont les résultantes des effets combinés des forces de pression et des forces tangentielles (appelées aussi forces de surface mouillée (*wall shear*)). Ces forces de pression et tangentielles agissant sur une surface élémentaire dS d'un corps sont respectivement exprimées par $p.dS$ et $\tau_w.dS$, comme décrites à la figure 8.9b. Les forces résultantes de « traînée » et de « portance » exercées sur un corps sont alors exprimées par :

$$F_D = \int_S (-pcos\theta + \tau_w sin\theta) dS \tag{8.1}$$

$$F_L = -\int_S (psin\theta + \tau_w cos\theta) dS \tag{8.2}$$

où θ est l'angle entre la normale à la surface élémentaire dS et la direction positive de l'écoulement du fluide. Ainsi, les forces de pression et de surface mouillée contribuent en général toutes les deux à la traînée et à la portance.

En pratique, on utilise souvent une approximation des forces résultantes de traînée et de portance agissant sur l'ensemble du corps solide. Cette approximation dépend entre autres de la densité ρ du fluide (dans notre étude, on utilise la densité de l'air), de la vélocité du fluide appliqué au solide V, des dimensions du corps solide et d'un coefficient sans dimension de traînée C_D ou de portance C_L. Les forces sont alors exprimées par :

$$F_D = \frac{1}{2}\rho C_D V^2 S \tag{8.3}$$

$$F_L = \frac{1}{2}\rho C_L V^2 S \tag{8.4}$$

où S est la surface frontale de référence.

8.3.2 Forces de convoyage et de lévitation

Pour le modèle du convoyage d'un objet, nous décrivons les interactions fluidiques entre les jets d'air distribués sur la surface active et l'objet manipulé. Les microactionneurs pneumatiques sont représentés par des carrés de $1mm^2$. Au centre de chaque carré est situé un orifice par lequel sort un jet d'air. Cet orifice est orienté suivant $\vec{x}$ ou $\vec{y}$.

Une vue en coupe est donnée à la figure 8.10. Un objet de centre de gravité G lévite et est convoyé selon $\vec{x}$. Les jets d'air sont générés en sortie des orifices. En position de repos, le jet d'air est de direction verticale et de vélocité $v_{az(off)}$. En position active, le jet d'air peut prendre deux orientations suivant qu'on l'active à droite ou à gauche du plan de mouvement. On décompose alors sa vélocité selon :

$$\vec{v}_{a(on)} = v_{ax(on)}\vec{x} + v_{ay(on)}\vec{y} + v_{az(on)}\vec{z}. \tag{8.5}$$

Par l'application combinée des jets d'air au repos et actifs, des forces fluidiques vont s'exercer sur l'objet en son centre de gravité G. On distingue la **force résultante de**

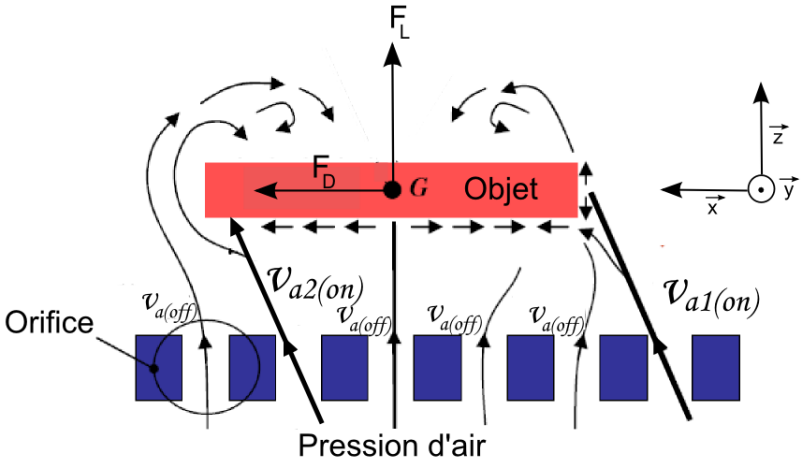


FIGURE 8.10 – Forces exercées sur l'objet dans la matrice de microactionneurs (image extraite de [CZFH08]).

lévitation F_L qui s'applique dans la direction $\vec{z}$ et la **force résultante de convoyage** F_D qui s'applique dans le plan $(\vec{x}, \vec{y})$. Un couple s'exerce aussi entraînant une rotation de l'objet autour de l'axe $\vec{z}$.

8.3.3 Modèle du convoyage 2D

Concernant notre modèle de convoyage en 2D, seuls les déplacements de l'objet dans le plan $(\vec{x}, \vec{y})$ sont considérés et sa lévitation n'est donc pas prise en compte. Les hypothèses sont alors les suivantes (*cf.* figure 8.10) :

- **nous modélisons uniquement la force de convoyage** F_D qui permet à l'objet de se déplacer dans le plan. Ainsi, l'objet est supposé être constamment en lévitation,
- nous supposons négligeables les composantes selon $\vec{x}$ et $\vec{y}$ de la force due aux jets d'air verticaux. Donc, on a :

$$\vec{v}_{a(off)} = v_{az(off)}\vec{z}. \tag{8.6}$$

  Les jets verticaux ne participent qu'à la lévitation dans notre modèle. Or, cette force de lévitation n'est pas modélisée, donc **les jets verticaux sont ignorés**,
- concernant les jets d'air orientés, seules **les composantes selon $\vec{x}$ et $\vec{y}$** sont modélisées donc :

$$\vec{v}_{a(on)} = v_{ax(on)}\vec{x} + v_{ay(on)}\vec{y}, \tag{8.7}$$

- pour les déplacements de l'objet dans le plan $(\vec{x}, \vec{y})$, seules **les interactions entre les jets d'air orientés et les bords de l'objet sont modélisées**. En effet, ces parties sont les plus exposées à la force de convoyage lors de la génération d'un jet d'air orienté [Zho07]. Les jets d'air orientés qui mouillent la surface inférieure de

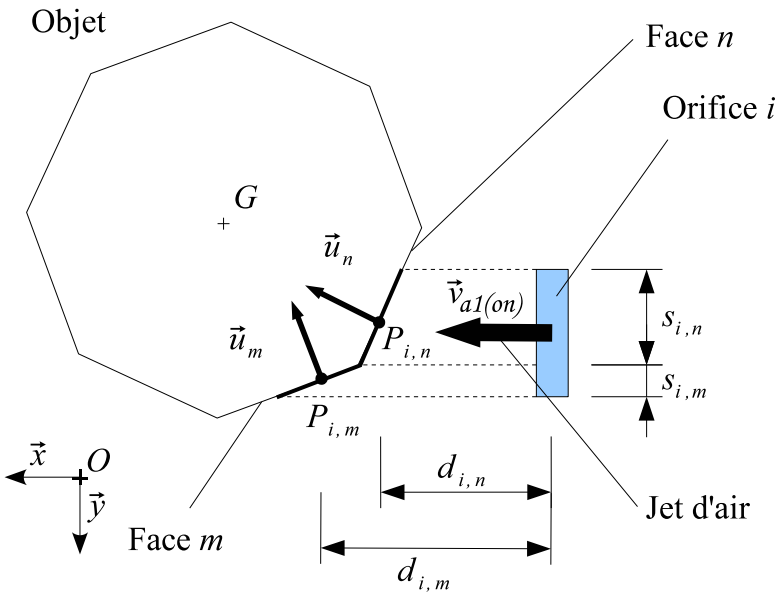


FIGURE 8.11 – Notations géométriques dans le cas d'un jet d'air orienté selon $\vec{x}$ et qui mouille la tranche de l'objet (vue de dessus).

l'objet ne sont donc pas pris en compte. Par exemple, à la figure 8.10, $\vec{v}_{a2(on)}$ est ignoré ; seul $\vec{v}_{a1(on)}$ participe au déplacement de l'objet,
- nous supposons enfin que **les jets d'air sont indépendants**, *i.e.* qu'il n'y a aucune interaction entre les jets.

Notations géométriques

Afin de modéliser la force de convoyage due aux jets d'air orientés, les notations géométriques données à la figure 8.11 sont nécessaires. On s'intéresse ici à un orifice normal à $\vec{x}$ et donc à un jet d'air parallèle à $\vec{x}$.

L'objet est représenté par un polygone à N faces. Son centre de gravité est noté G. L'abcisse de G est noté x et son ordonné est notée y. Tous les orifices sont numérotés de 1 à M. Lorsqu'un orifice est proche de l'objet et que le jet d'air est orienté dans la direction de l'objet (comme illustré à la figure 8.11), le jet d'air atteint alors une tranche de l'objet appelée « aire mouillée ». Nous faisons l'hypothèse que cette aire mouillée est la projection dans le plan $(O, \vec{y}, \vec{z})$ de la tranche atteinte par le jet d'air. La relation entre une face n de l'objet et un orifice i de la surface est décrite grâce aux variables suivantes :
- $\vec{u}_n$ est le vecteur normal à la face n,
- $s_{i,n}$ est la surface de l'aire mouillée de la face n,
- $P_{i,n}$ est le centre de gravité de l'aire mouillée de la face n,

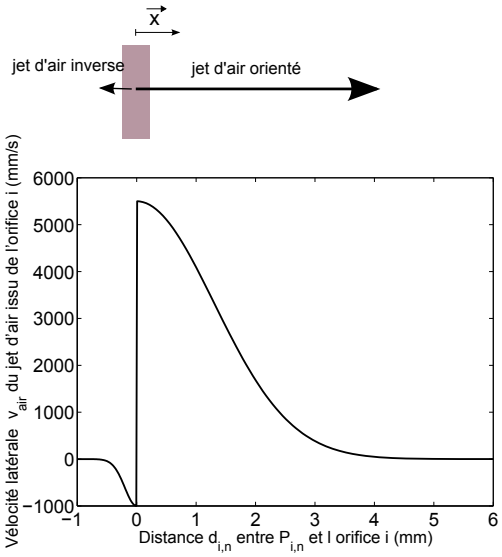


FIGURE 8.12 – Vélocité latérale d'un jet d'air en fonction de sa distance à l'orifice.

– $d_{i,n}$ est la distance entre $P_{i,n}$ et l'orifice i.

Ainsi, on peut déduire de l'équation 8.3 la force de convoyage élémentaire exercée par le jet d'air de l'orifice i sur la face n :

$$\vec{f}_{i,n} = \frac{1}{2}\rho C_D s_{i,n} v_{i,n}^2 \vec{u}_n. \tag{8.8}$$

Nous supposons que la force de convoyage exercée sur une face n est normale à cette face. $v_{i,n}$ est la vitesse relative du jet d'air au point $P_{i,n}$ donnée par :

$$v_{i,n} = \dot{x} - v_{air}(d_{i,n}) \tag{8.9}$$

où $\dot{x}$ est la vitesse de l'objet et v_{air} est une fonction qui donne la vélocité d'un jet d'air en fonction de sa distance à l'orifice (*cf.* figure 8.12 et annexe C). Cette fonction est issue de l'expérience exceptée la vitesse latérale du jet d'air en sortie de l'orifice ($v_{air}(0) = 5,5m/s$) dont la valeur est donnée dans une modélisation à base de calculs par éléments finis du microactionneur pneumatique réalisée par [FCMF06]. On remarque une vélocité négative du jet d'air pour une distance inférieure à 0. En effet, lorsqu'un jet d'air est généré en sortie d'un orifice dans le sens de $\vec{x}$ par exemple, un léger jet d'air inverse est aussi généré dans le sens contraire. Ceci illustre la constatation expérimentale suivante : un objet se déplaçant dans la direction de $\vec{x}$ et dont l'inertie est faible peut alors « rebondir » contre les jets d'air inverses (voir aussi les résultats au §8.4.2).

Étant donné le phénomène de dispersion dans le processus de fabrication des MEMS, un actionneur peut présenter des défaillances. Afin de modéliser cette incertitude sur l'effet d'un jet d'air, la force de convoyage élémentaire $\vec{f}_{i,n}$ exercée par un jet d'air d'un orifice i sur une face n est nulle avec une probabilité de $0,05$.

Modélisation

Pour aboutir à la force de convoyage résultante notée $\vec{F}_D$, qui s'applique au centre de gravité G de l'objet, toutes les forces de convoyage élémentaires exercées par les M jets d'air sur les N faces de l'objet sont sommées selon :

$$\vec{F}_D = \sum_{i=1}^{M} \sum_{n=1}^{N} \delta_{i,n} \vec{f}_{i,n} \tag{8.10}$$

où $\delta_{i,n}$ est égal à 1 si le jet d'air issu de l'orifice i atteint la face n et 0 sinon. $\delta_{i,n}$ est aussi égal à 0 si l'orifice i est situé en-dessous de l'objet. En effet, les forces fluidiques résiduelles exercées par les microactionneurs sur la face inférieure de l'objet sont négligées par hypothèse.

Le moment résultant de ces forces élémentaires est alors :

$$\vec{\mathcal{M}}_{F_D} = \sum_{i=1}^{M} \sum_{n=1}^{N} \delta_{i,n} (\vec{GP}_{i,n} \wedge \vec{f}_{i,n}). \tag{8.11}$$

Force de viscosité

Lorsque l'objet est en mouvement sur la surface, il peut être considéré comme un corps en mouvement dans un fluide (l'air). Il est donc soumis à des forces de frottement réparties sur sa surface. La résultante de ces forces de frottement est appelée la **force de viscosité** dans le cas des fluides. Cette force de viscosité s'exprime de la manière suivante :

$$F_V = -K\eta v \tag{8.12}$$

où K est un coefficient géométrique dépendant de la forme de l'objet, η le coefficient de viscosité du fluide et v la vitesse de l'objet.

Modèle dynamique du convoyage 2D

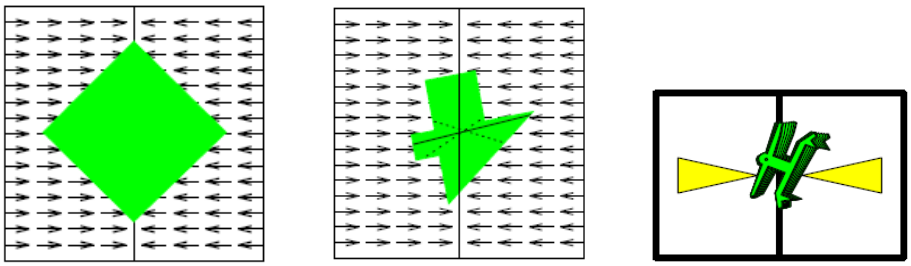
Au final, la dynamique du convoyage 2D de l'objet suit les relations suivantes :

$$\begin{cases} m\ddot{x} &= \vec{F}_D \bullet \vec{x} - K\eta\dot{x} \\ m\ddot{y} &= \vec{F}_D \bullet \vec{y} - K\eta\dot{y} \\ J\ddot{\theta} &= \vec{\mathcal{M}}_{F_D} \bullet \vec{z} \end{cases} \tag{8.13}$$

où les constantes sont précisées à l'annexe C et (x, y) sont les coordonnées du centre de gravité G de l'objet et $\dot{\theta}$ sa vitesse angulaire.

8.4 Validation de l'approche par apprentissage par renforcement multiagent

Dans cette section, l'approche par apprentissage par renforcement multiagent est validée en tant que méthode de contrôle de la surface active pneumatique. Pour cela,



(a) (b) (c)

FIGURE 8.13 – Divers positionnements réussis sur un équilibre stable avec la méthode par champ de vecteur programmable (illustrations issues de [BBR+00]).

deux approches de contrôle issues de la littérature sont présentées. Puis les raisons qui étayent l'approche par apprentissage par renforcement décentralisé sont détaillées. Enfin, une approche usuelle de contrôle des surfaces distribuées est comparée à l'apprentissage par renforcement sur une tâche de positionnement d'un objet sur la surface. Des résultats issus de tests avec le prototype et le modèle sont présentés.

8.4.1 Diverses approches de contrôle décentralisé

Contrôle en boucle ouverte par champ de vecteur programmable

Concernant les systèmes de micromanipulation « à contact », une approche de contrôle usuelle est le **champ de vecteur programmable**[5], proposé, étudié et appliqué par Karl-Friedrich Böhringer *et al.* [BDMM94, BDM96, fBRNM96, BBR+00, BDKL00]. C'est une méthode de contrôle par boucle ouverte pouvant être employée pour orienter, trier, assembler, positionner, etc ... un objet situé une surface de micromanipulation distribuée. Elle est couramment employée car son principe est simple : un champ de vecteur définit la commande (ou action) à assigner à chaque actionneur. Ainsi, quand un objet est placé sur la surface, le champ de vecteur induit par les actions de tous les actionneurs génère une force et un couple sur cet objet. Celui-ci va alors se déplacer jusqu'à atteindre une position d'équilibre stable, comme illustré à la figure 8.13.

Pour définir le ou les champs de vecteur à appliquer, il est nécessaire d'être capable de prédire le mouvement de l'objet dans ce champ. En particulier, il est important de déterminer les différentes positions d'équilibre stable que l'objet peut atteindre. Une analyse de ces équilibres est notamment décrite dans [BDMM94, BBR+00].

Cette approche présente cependant des difficultés d'application dans le cas où les équilibres stables sont difficiles à déterminer ou lorsque la dynamique de l'objet est complexe. En particulier, avec les systèmes de micromanipulation distribuée « sans contact », tels que les surfaces pneumatiques, l'approche par champ de vecteur programmable peut présenter de mauvaises performances de stabilité ou de précision, ou encore échouer à

5. En anglais *programmable vector field*.

positionner un objet, comme cela est illustré avec le prototype de surface pneumatique au §8.4.2. D'autres approches pour déterminer un point d'équilibre stable [ML04] ou utilisant une boucle fermée [KWSS01] doivent alors être développées.

Contrôle réactif en boucle fermée par détection du bord de l'objet

Concernant le prototype de surface pneumatique, un contrôle décentralisé en boucle fermée a été implémenté dans [FCMF06]. Il repose sur méthode simple de détection du bord de l'objet. Lorsqu'un actionneur détecte le bord de l'objet au-dessus de lui, et selon qu'il s'agisse du bord avant ou arrière, il applique immédiatement le jet d'air correspondant à la direction désirée.

Cette approche ne donne pas de bons résultats avec une pression constante car l'objet a alors une forte instabilité due aux effets fluidiques et à son inertie en lévitation. C'est pourquoi les auteurs utilisent des « pulses » périodiques d'air afin de réduire la dynamique de l'objet.

Contrôle adaptatif par apprentissage par renforcement multiagent

Nous proposons d'expérimenter une méthode de contrôle par apprentissage par renforcement décentralisé pour cette application de positionnement et convoyage d'un objet sur une *smart surface* pneumatique. Plus spécifiquement, les algorithmes décentralisés pour agents autonomes, indépendants, ayant une observabilité totale de l'espace d'état et situés dans un système multiagent coopératif ont été étudiés dans ce mémoire. Les raisons justifiant l'utilisation de tels algorithmes pour cette application sont les suivantes :

- la surface distribuée peut être considérée comme un **système multiagent**, où chaque actionneur est un agent autonome, apprenant sa propre stratégie de commande qui donne l'action à effectuer (gauche, droite ou rien si l'actionneur est vertical et haut, bas ou rien s'il est horizontal) selon l'observation (position du centre de gravité de la pièce par exemple),
- l'ensemble de ces agents autonomes ont un objectif compatible qui est le positionnement ou le convoyage d'un objet. Ils forment donc un **système multiagent coopératif**,
- l'ensemble des actionneurs proche de l'objet participent au déplacement de celui-ci. Les agents doivent donc apprendre à se **coordonner** afin de réaliser leur objectif commun,
- le système à commander est non-linéaire et stochastique. De plus, un **modèle** réaliste de ce dispositif à base de MEMS distribués reste très complexe à développer [Zho07, CZFH08].

Pour toutes ces raisons, une approche de contrôle adaptatif par apprentissage par renforcement, ne nécessitant pas de modèle du système et adaptée à des systèmes stochastiques et non-linéaires, peut être appliquée. En particulier, nous nous intéressons à valider l'approche décentralisée étudiée dans ce mémoire. Le cadre choisi pour l'apprentissage par renforcement est celui des jeux de Markov d'équipe.

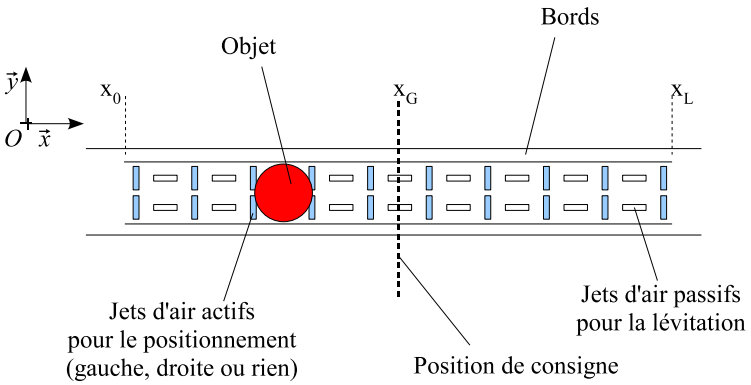


FIGURE 8.14 – Description de la tâche de positionnement.

8.4.2 Tâche de positionnement

Dans ce paragraphe, l'approche classique par champ de vecteur programmable et celle par apprentissage par renforcement décentralisé sont comparées sur une tâche de positionnement. L'objectif est ici de valider l'approche par apprentissage en tant que méthode de contrôle adaptée à notre système. Cette approche permet de contrôler la position et la vitesse de l'objet avec une pression d'air constante, donc avec une dynamique de l'objet complexe (contrairement à la méthode réactive par détection de bord). Elle permet aussi une stabilité de l'objet plus performante (en rapidité et précision) que la méthode par champ de vecteur programmable.

Description de la tâche

Nous souhaitons stabiliser un cylindre de plastique (2mm de diamètre et $0,5$mm d'épaisseur) à une position de consigne sur la surface. En simulation, ce cylindre est modélisé par un polygone à 21 faces. Nous cherchons à stabiliser la position dans une seule direction. Pour cela, un espace restreint de la surface est utilisé. Il contient 10 colonnes de 2 actionneurs chacune (*cf.* figure 8.14). La position de consigne est alors le milieu de cette zone. Tous les microactionneurs verticaux sont commandés, colonne par colonne, avec la direction du jet d'air (droite, gauche ou rien) requise par le système de contrôle. Les microactionneurs horizontaux envoient un jet d'air vertical de sorte à assurer simplement la lévitation de l'objet.

Montage expérimental

Le montage expérimental utilisé pour le prototype de surface pneumatique est composé d'un système de contrôle de l'alimentation en air, d'une interface de pilotage pour le multiplexage et l'alimentation haute tension, d'une interface de vision pour le traitement vidéo et d'un ordinateur pour l'interface de contrôle.

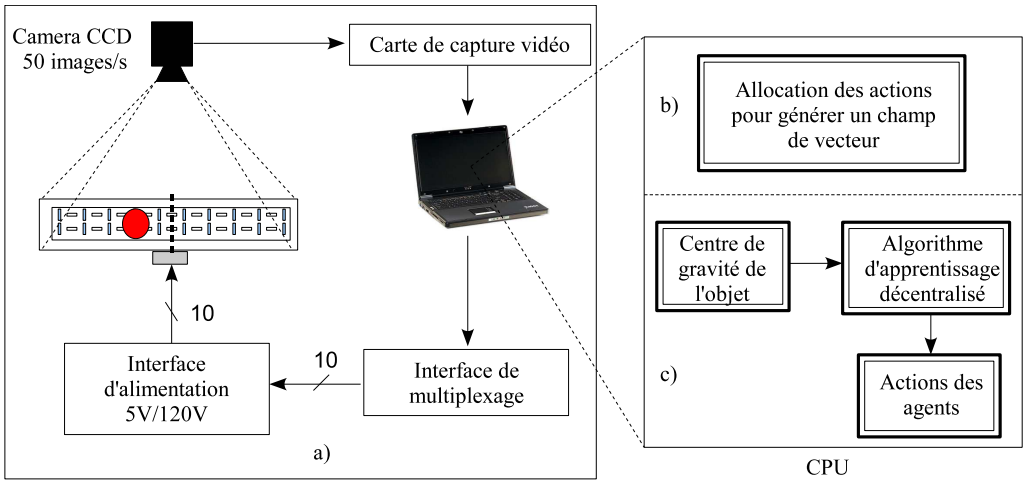


FIGURE 8.15 – a) Configuration matérielle pour le contrôle du positionnement. b) Interface de contrôle par champ de vecteur programmable. c) Interface de contrôle par apprentissage par renforcement.

Tout d'abord, la surface pneumatique est posée sur une plate-forme permettant d'ajuster la position d'équilibre de la surface. La source du flux d'air est fournie par de l'air compressé (3kPa pour assurer la lévitation) *via* les microvalves.

La figure 8.15a donne la configuration matérielle utilisée pour le contrôle du positionnement d'un objet. L'interface de vision se compose d'une caméra CCD et d'une carte d'acquisition permettant de récupérer des vidéos de la surface pneumatique. La position du centre de l'objet est calculée à partir de ces images dans l'unité centrale de calcul (CPU) d'un ordinateur. Des fonctions de seuillage et de détection de régions connexes sont utilisées à une vitesse de 50 images par seconde (la bibliothèque graphique utilisée est OpenCV).

La tension électrique appliquée à chaque actionneur électrostatique est approximativement de 90 VDC.

Les microactionneurs sont commandés colonne par colonne. La même action est donc distribuée à tous les actionneurs d'une même colonne. L'algorithme de contrôle décentralisé, implémenté dans le CPU, envoie donc une commande pour chaque colonne. Ce signal est transmis aux microactionneurs *via* une interface de multiplexage.

Résultats avec le champ de vecteur programmable

Concernant le contrôle par champ de vecteur programmable, un point d'équilibre stable est créé au centre de la surface (position égale à $x_G = 12,7$mm sur le prototype et à $x_G = 5$mm sur le modèle) grâce au champ de vecteur donné à la figure 8.16. Celui-ci suit les recommandations de Karl-Friedrich Böhringer *et al.* [fBRNM96, BDKL00]. Ce champ de vecteur programmable est testé en simulation et sur le prototype de surface pneumatique. La position du centre du cylindre en fonction du temps est tracée à la

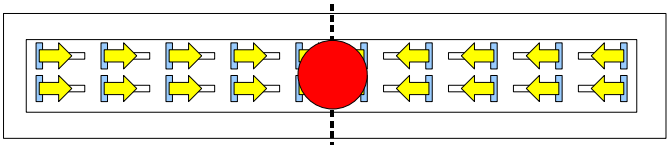


FIGURE 8.16 – Champ de vecteur.

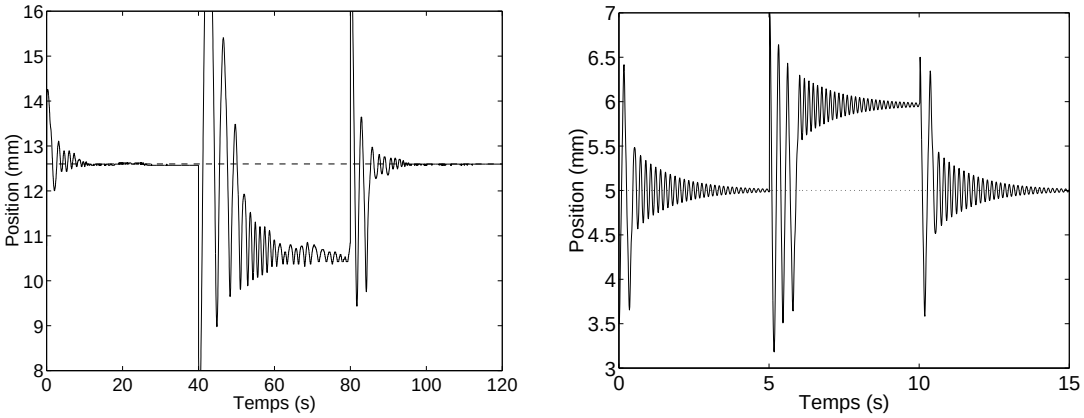
figure 8.17. Trois perturbations sont effectuées à intervalle de temps régulier. La position de consigne x_G est représentée par une ligne pointillée. Tout d'abord, on remarque que les échelles des réponses sont différentes entre le système simulé et le système réel. Le meilleur temps de réponse obtenu avec le système réel se situe autour de 10s, alors qu'il est de 5s avec le système simulé. Néanmoins, dans les deux cas, le déplacement de l'objet se caractérise par de fortes oscillations avant d'atteindre le point d'équilibre stable généré par le champ de vecteur. De plus, lors de certains essais, l'objet se stabilise sur une position différente de x_G. En effet, pendant les oscillations, le champ de vecteur a pour effet de réduire la vitesse de l'objet. Mais une fois que celui-ci est trop lent, il risque alors de « rebondir » contre les jets d'air inverses. Ce phénomène a été intégré dans notre modèle à travers la fonction donnant la vélocité d'un jet d'air (*cf.* figure 8.12). On constate d'ailleurs que les positions où l'objet se stabilise correspondent à des positions où les bords de l'objet sont en contact avec ces jets d'air inverses.

Ainsi, le contrôle en boucle ouverte par champ de vecteur programmable présente de faibles performances de stabilité et n'est pas adapté au contrôle de ce système.

Résultats avec l'apprentissage par renforcement décentralisé

Concernant le contrôle par apprentissage par renforcement décentralisé, on choisit le cadre des jeux de Markov d'équipe où :

- m est le nombre d'agents. Ici, tous les microactionneurs sont contrôlés colonne par colonne, donc le système nécessite autant d'agents que de colonnes impliquées dans la tâche de positionnement, *i.e.* $m = 10$. Chaque agent décide d'une action à effectuer et celle-ci est ensuite distribuée aux microactionneurs de chaque colonne,
- $\mathcal{S}$ est un ensemble fini d'états. La position du centre de gravité de l'objet est transmise par l'interface vidéo à chaque pas de temps et envoyée à chaque agent indépendant (*cf.* figure 8.15c). Afin de prendre en compte la dynamique de l'objet, la position au pas précédent est mémorisée. Ainsi, à chaque pas, l'information reçue par chaque agent est la position actuelle de l'objet notée x et sa position précédente notée $\tilde{x}$,
- $\mathcal{A}_i$ est l'ensemble des actions pour l'agent i. Chaque actionneur peut effecteur trois actions : orienter le jet d'air à droite, à gauche ou ne pas l'orienter, donc $|A_i| = 3$,



(a) Résultats avec le prototype de surface pneumatique (perturbations en 0s, 40s et 80s)

(b) Résultats avec la simulation de la surface (perturbations en 0s, 5s et 10s)

FIGURE 8.17 – Résultats du positionnement (après perturbations) avec la méthode par champ de vecteur programmable. La position de consigne est représentée par une ligne pointillée en a) $x_G = 12,7$mm b) $x_G = 5$mm.

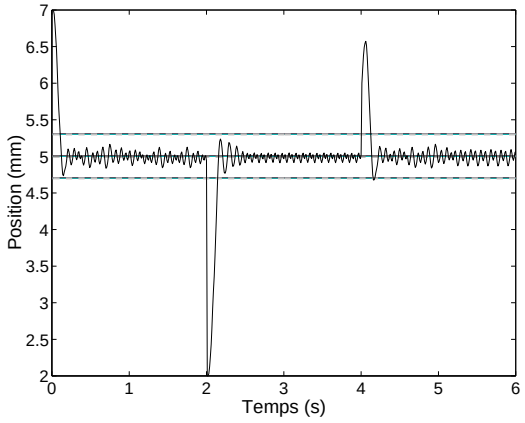


FIGURE 8.18 – Résultats du positionnement avec la simulation de la surface et la méthode par apprentissage par renforcement lorsque les politiques gloutonnes sont suivies. Perturbations en 0s ($x_{ini} = 7$mm), 2s ($x_{ini} = 2$mm) et 4s ($x_{ini} = 4$)mm). La position de consigne est représentée par une ligne pointillée en $x_G = 5$mm. Les deux lignes pointillées délimitent la marge définie pour la fonction de récompense.

Algorithme 13 : Algorithme du $\mathsf{Q}(\lambda)$ décentralisé pour un agent indépendant i dans un jeu de Markov d'équipe

début
  Initialiser $\mathsf{Q}_i(s,a_i)$ arbitrairement et $\mathsf{e}_i(s,a_i)$ à 0 pour tout (s,a_i) de $\mathcal{S}\times\mathcal{A}_i$
  Initialiser l'état initial s et choisir au hasard une action a_i
  tant que *s n'est pas un état absorbant* **faire**
    Exécuter l'action a_i et observer le nouvel état s' et la récompense r
    $a_i' \leftarrow$ Décision(Q_i, s')
    $a^* \in \arg\max_{b\in\mathcal{A}_i} \mathsf{Q}_i(s',b)$
    $\delta \leftarrow r + \gamma\mathsf{Q}_i(s',a^*) - \mathsf{Q}_i(s,a_i)$
    $\mathsf{e}_i(s,a_i) \leftarrow \mathsf{e}_i(s,a_i)+1$ ou $\mathsf{e}_i(s,a_i)\leftarrow 1$ ▷ `éligibilité accumulatrice ou remplaçante`
    pour tous les $(u,b)\in\mathcal{S}\times\mathcal{A}_i$ **faire**
      $\mathsf{Q}_i(u,b) \leftarrow \mathsf{Q}_i(u,b) + \alpha\delta\mathsf{e}_i(u,b)$
      $\mathsf{e}_i(u,b) \leftarrow \begin{cases} \gamma\lambda\mathsf{e}_i(u,b) & \text{si } \mathsf{Q}_i(s_i',a_i') = \mathsf{Q}_i(s_i',a^*) \\ 0 & \text{sinon} \end{cases}$
    $s\leftarrow s'$,$a_i \leftarrow a_i'$
fin

– la fonction de récompense choisie afin de stabiliser l'objet au milieu de la surface est la suivante :

$$\mathsf{R}(x,\tilde{x}) = \begin{cases} 1 \text{ si } (x,\tilde{x}) \in [x_G-\rho, x_G+\rho]^2 \\ -1 \text{ si } x < x_0 \text{ ou } x > x_L \\ 0 \text{ sinon} \end{cases} \tag{8.14}$$

où ρ est la marge choisie pour la récompense, x_0 l'abscisse minimale de la surface, x_L l'abscisse maximale de la surface et x_G la position de consigne (*cf.* figure 8.14). Les bords permettent de bloquer le mouvement de l'objet selon $\vec{y}$ mais aucun bord n'est présent pour empêcher l'objet de sortir de la surface en $\vec{x}$. Lorsque cela arrive, les agents sont pénalisés.

Dans le modèle, la position du centre de gravité de l'objet est $x \in [0;10]$mm. La position de consigne est le centre de la surface ($x_G = 5$mm) et la marge pour la fonction de récompense est $\rho = 0,3$mm. Pour appliquer l'apprentissage par renforcement dans la forme utilisée dans ce mémoire, les échelles de temps et de position doivent être discrétisées. Le pas d'échantillonnage est de 0,01s pour le contrôle et la simulation s'effectue à pas variables (méthode d'intégration ODE45). Pour l'espace d'état $(x,\tilde{x})$, une discrétisation de 41×41 est utilisée. Donc, chacun des 10 agents maintient une table de valeurs Q_i de taille $41\times 41\times 3$. A chaque début d'épisode, l'objet est placé aléatoirement à une position initiale $x_{ini} \in [2;8]$mm. Un épisode se termine si l'objet sort de la surface ($x < 0$mm ou $x > 10$mm) et dure au plus 10s.

Ainsi, de nombreux couples d'état-action doivent être mis à jour. Afin d'accélérer la mise à jour de ces nombreuses valeurs, le TD(λ) pourrait être appliqué. Seule une faible pénalité est utilisée dans le cas où l'objet sort de la surface. La non-coordination

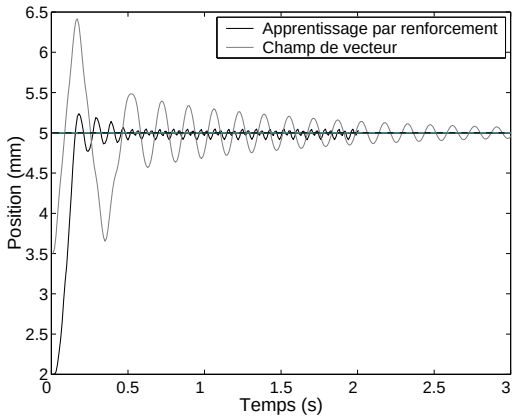


FIGURE 8.19 – Comparaison des deux méthodes de contrôle sur une tâche de positionnement avec la simulation de la surface. La position de consigne est représentée par une ligne pointillée en $x_G = 5$mm.

des agents est donc peu pénalisée. Dans ces conditions, le Q-learning décentralisé peut présenter de bonnes performances. De plus, l'objectif est ici de valider l'approche par apprentissage par renforcement décentralisé et non pas de comparer différents algorithmes multiagents. C'est pourquoi, pour cette tâche de positionnement, le Q-learning décentralisé combiné aux traces d'éligibilité est utilisé. L'algorithme est donc une version décentralisée pour agents indépendants du Q(λ) de Christopher J.C.H. Watkins [Wat89] (*cf.* §2.5.3). Il est donné à l'algorithme 13.

Cet algorithme est testé en simulation avec notre modèle de convoyage de la surface pneumatique. 300 épisodes d'apprentissage avec le Q(λ) décentralisé sont effectués avec les paramètres suivants : $\alpha = 0,1$, $\gamma = 0,9$, $\lambda = 0,7$ et une éligibilité remplaçante. La méthode de décision ϵ-greedy est utilisée avec une exploration globale de $\psi = 0,1$. De l'exploration systématique est induite avec une initialisation à 1 des fonctions de valeur Q_i. A la fin de cet apprentissage, plusieurs épisodes d'exploitation sont réalisés où les agents indépendants suivent leurs politiques gloutonnes. Chaque épisode d'exploitation dure 2s et différentes positions et vitesses initiales du centre de gravité de l'objet sont testées. A la figure 8.18 est tracée la position du centre de l'objet en fonction du temps avec les positions initiales successives suivantes : $x = 7$mm, $x = 2$mm et $x = 4$mm. Dans les deux premiers cas, la vitesse initiale de l'objet est nulle. Dans le dernier cas, la vitesse initiale de l'objet tend à l'éloigner du centre de la surface.

On constate que l'objet atteint rapidement la position de consigne et aucune oscillation dépassant la marge d'erreur est constatée. Les agents ont donc appris à accélérer l'objet pour que celui-ci atteigne rapidement le milieu de la surface. Cependant, contrairement au cas du champ de vecteur, certains agents envoient ici un jet d'air vertical. En effet, si tous les jets sont appliqués sur l'objet, celui-ci sera difficilement freiné et oscillera pendant un certain temps. Les agents ont donc appris à se coordonner pour « doser » la

force à appliquer sur l'objet afin d'être capables de le ralentir et ainsi d'éviter de fortes oscillations. Dans le dernier cas, les agents contrent très rapidement la vitesse initiale de l'objet. De plus, quelle que soit la position initiale, les agents n'échouent pas à positionner l'objet. A la figure 8.19, des positionnements par apprentissage par renforcement et par champ de vecteur programmable sont comparés. Après 3s, l'amplitude des oscillations avec la méthode par champ de vecteur est toujours supérieure à celle obtenue après 0,5s avec l'apprentissage.

8.4.3 Conclusion

Le contrôle par apprentissage par renforcement décentralisé est donc capable de stabiliser un objet rapidement et avec précision sur le modèle de surface pneumatique. *L'approche par apprentissage par renforcement multiagent est donc validée en tant que méthode de contrôle de la surface pneumatique.*

8.5 Commande décentralisée par apprentissage par renforcement avec observabilités partielles

Dans cette section, nous nous intéressons à la commande décentralisée par apprentissage par renforcement de la *smart surface* dans le cadre plus réaliste d'observabilités partielles. En effet, dans le cas d'une *smart surface* totalement intégrée, chaque agent n'a accès qu'à une information locale provenant des capteurs. Une approche par macro-actions est alors proposée. Elle permet d'étendre les algorithmes présentés dans ce mémoire au cadre des observabilités partielles. Des résultats obtenus avec la simulation du convoyage 2D d'un objet sur la *smart surface* sont présentés.

8.5.1 Macro-actions et processus décisionnels semi-Markovien

Macro-actions

Dans un processus décisionnel de Markov (PDM), chaque action a implicitement une même durée égale à un pas de temps : c'est ce qu'on appelle une **action primitive**. Cependant, il peut s'avérer utile de pouvoir modéliser des actions ayant des durées différentes. Par exemple, dans certaines régions de l'espace, une résolution temporelle fine peut être nécessaire pour obtenir la réactivité et la précision nécessaire au niveau du changement d'action. Par contre, dans d'autres régions de l'espace, la même action peut être répétée pendant un nombre de pas consécutifs. Une résolution temporelle moins fine dans ces états permet alors de réduire le nombre de décisions prises par les agents.

Pour cela, le concept de **macro-action** a été proposé. Il permet de regrouper plusieurs actions primitives. Par exemple, si les actions primitives d'un problème sont « faire un pas dans une direction donnée », une macro-action pourrait être « faire dix pas vers le nord suivis de deux pas vers l'ouest ». Les macro-actions permettent ainsi de représenter

un problème selon différents niveaux d'« abstraction temporelle » [Pre00]. Le concept de macro-action est formalisé à travers les processus décisionnels semi-markoviens.

Processus décisionnel semi-Markovien

Un processus décisionnel semi-markovien (PDSM) peut être vu comme un PDM avec la différence que les transitions peuvent avoir une durée de temps stochastique. Les actions dans les PDSM peuvent ainsi avoir des durées d'exécution différentes et sont destinées à modéliser des actions temporellement abstraites, ou macro-actions. Nous ne considérons ici que les PDSM dans lesquels le temps est discret.

Définition 8.1 *Un processus décisionnel semi-Markovien (PDSM) [Put94, Par98] fini est un quintuplet* $\langle \mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R}, \mathsf{F} \rangle$ *où :*

- $\mathcal{S}$ *est un ensemble fini d'états,*
- $\mathcal{A}$ *est un ensemble fini d'actions,*
- $\mathsf{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0;1]$ *est une fonction de transition définissant une distribution de probabilité sur les états futurs telle que :*

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \sum_{s' \in \mathcal{S}} \mathsf{T}(s,a,s') = 1,$$

- $\mathsf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ *est une fonction de récompense qui associe une récompense immédiate à chaque transition,*
- $\mathsf{F} : \mathbb{N} \times \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0;1]$ *est une fonction définissant une distribution de probabilités sur les états futurs et sur la durée de la transition. Ainsi,* $\mathsf{F}(t|s,a,s')$ *représente la probabilité que l'action* a*, exécutée à partir de* s*, mène en* s' *et dure* t *pas de temps.*

L'objectif des macro-actions est d'accélérer la phase d'apprentissage des algorithmes d'apprentissage traditionnels [MSF97, MS98, Gar04, STM05, JHS08]. La taille du problème est implicitement réduite en permettant aux agents d'atteindre le but avec moins de séquences de décisions. Les macro-actions ont été introduites dans l'apprentissage par renforcement de diverses manières, la plupart étant basées sur des *approches hiérarchiques*. Les contributions principales sont le formalisme des *options* [SPS99], les actions à pas multiples[6] (MSA) [SR02, SR03, SRI03], les hiérarchies de machines abstraites[7] (HAM) [Par98] et la méthode MAXQ [Die00].

Les deux dernières méthodes sont basées sur la notion de décomposition de la tâche en sous-tâches, chacune avec un sous-but. Les MSA sont spécifiquement destinées aux approches ne pouvant pas être décomposées en sous-problèmes. Une MSA est une séquence de la même action primitive appliquée pendant un nombre consécutif de pas. Enfin, le formalisme des *options* est devenu l'un des plus populaires parmi les méthodes d'abstraction temporelle, d'une part grâce à sa simplicité de mise en oeuvre, et d'autre part car ce formalisme généralise une grande partie des travaux sur les macro-actions.

6. En anglais *multi-step actions.*
7. En anglais *hierarchy of asbtract machines.*

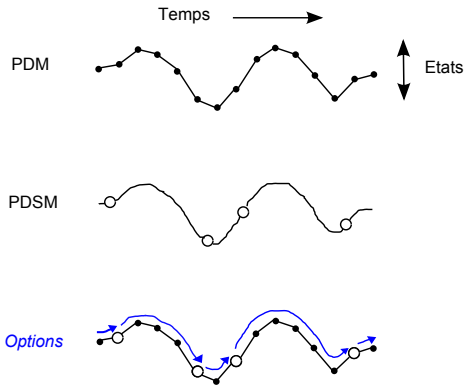


FIGURE 8.20 – Un PDM modélise des actions primitives. Un PDSM modélise des actions pouvant durer plusieurs pas de temps. Ces différentes échelles peuvent être superposées avec le formalisme des *options* (illustration issue de [SPS99]).

Options

Le formalisme des *options* [Pre00, SPS99] est une généralisation des actions qui permet de représenter des actions ayant des durées différentes. Une action primitive, *i.e.* qui dure un pas de temps, est une *option* particulière. Par rapport aux macro-actions (actions temporellement étendues), l'intérêt principal des *options* est que ce formalisme définit une représentation des actions temporellement étendues à travers une combinaison de **politiques** définies sur plusieurs pas et d'une **condition de terminaison**.

Si un PDM utilise des transitions s'effectuant sur un pas de temps, un PDSM utilise des transitions pouvant s'effectuer sur plusieurs pas de temps. Les *options* permettent de définir des transitions comme dans un PDSM, et donc en quelque sorte de « superposer » un PDSM sur un PDM, comme illustrée à la figure 8.20. Ainsi, pour tout PDM et pour tout ensemble fixé d'*options*, le processus de décision choisissant parmi ces *options* est un PDSM [SPS99, Pre00] :

$$\texttt{PDM} + \textit{options} = \texttt{PDSM}.$$

Une *option* est une politique avec une condition de terminaison. A chaque pas de temps, l'agent peut choisir entre une *option* ou une action primitive, à moins qu'il soit déjà en train d'exécuter une *option*. Une fois qu'une *option* est choisie, il sélectionne une action primitive en accord avec la politique de son *option* jusqu'à ce que la condition de terminaison soit satisfaite.

Définition 8.2 *Une* option *markovienne o [SPS99, Pre00] est un triplet* $\langle \mathcal{I}, \pi, \beta \rangle$ *où :*

- $\mathcal{I} \subseteq \mathcal{S}$ *est l'ensemble des états dans lesquels o peut commencer,*
- $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0;1]$ *est la politique suivie durant l'exécution de o,*
- $\beta : \mathcal{S} \mapsto [0;1]$ *est la condition de terminaison qui représente la probabilité de terminer l'option o pour chaque état.*

Une *option* $o = \langle \mathcal{I}, \pi, \beta \rangle$ est donc initialisée dans un état $s_t \in \mathcal{I}$. Une fois qu'une *option* o est choisie, les actions sont alors sélectionnées suivant la politique π jusqu'à ce que l'*option* se termine (de manière stochastique) selon β. Ainsi, dans l'état s_t, l'action a_t est choisie selon la politique $\pi(s_t, .)$ et l'environnement se retrouve alors dans un état s_{t+1}. Dans s_{t+1}, l'*option* peut soit se terminer avec une probabilité $\beta(s_{t+1})$, ou continuer, et alors, l'action a_{t+1} est choisie selon $\pi(s_{t+1}, .)$, et l'*option* se termine dans l'état s_{t+2} avec une probabilité $\beta(s_{t+2})$; et ainsi de suite. Une fois qu'une *option* se termine, l'agent a alors la possibilité de choisir une nouvelle *option* si son état le permet.

Par exemple, une *option* « ouvrir une porte » pourrait consister en :
- un ensemble $\mathcal{I}$ comprenant les états dans lesquels une porte est présente,
- une politique permettant d'atteindre, d'attraper et de tourner la poignée d'une porte,
- une condition de terminaison reconnaissant si la porte est ouverte.

États « non-contrôlés » et politiques « gelées »

Ronald E. Parr [Par98] propose une manière alternative de considérer un PDSM à travers les notions d'états « contrôlés » et « non-contrôlés ». Le processus peut passer à travers plusieurs états « non-contrôlés » avant que celui-ci ne retourne dans un état « contrôlé ». Dans ce cas, la fonction F d'un PDSM peut aussi être définie comme une distribution de probabilités sur le nombre de pas de temps avant le prochain état contrôlé. $\mathsf{F}(t|s, a, s')$ représente la probabilité que le prochain état contrôlé s' soit atteint en t pas de temps si l'action a est choisie dans l'état s.

Dans chaque état « non-contrôlé », une politique est définie qui est suivie par l'agent dès qu'il est dans un de ces états. Les politiques associées aux états « non-contrôlés » sont appelées des **politiques « gelées »** . Si les politiques gelées ne changent pas, les états « non-contrôlés » peuvent être extraits du processus. Le PDSM résultant est alors constitué des états « contrôlés » et d'actions temporellement étendues qui modélisent le passage de l'agent dans les états « non-contrôlés ». Il est alors possible pour un agent d'apprendre les politiques optimales pour les états « contrôlés » du PDSM résultant [Par98].

Par exemple, prenons un problème de navigation où un agent doit se déplacer dans quatre pièces reliées entre elles (*cf.* figure 8.21). Dans la pièce 2, une politique π_2 est fixée (politique gelée). Elle est suivie par l'agent dès qu'il est dans un état de cette pièce. Ces états sont donc « non-contrôlés ». Il est alors possible de retirer du processus les états associées à cette pièce. Ainsi, les transitions effectuées lorsque π_2 est suivie peuvent être considérées comme le résultat d'actions temporellement étendues définies dans un PDSM. En résumé, l'utilisation de politiques « gelées » et d'extraction d'états « non-contrôlés » peut être interprétée comme une transformation en un PDSM dans lequel le comportement dans les états extraits est défini par des actions temporellement abstraites.

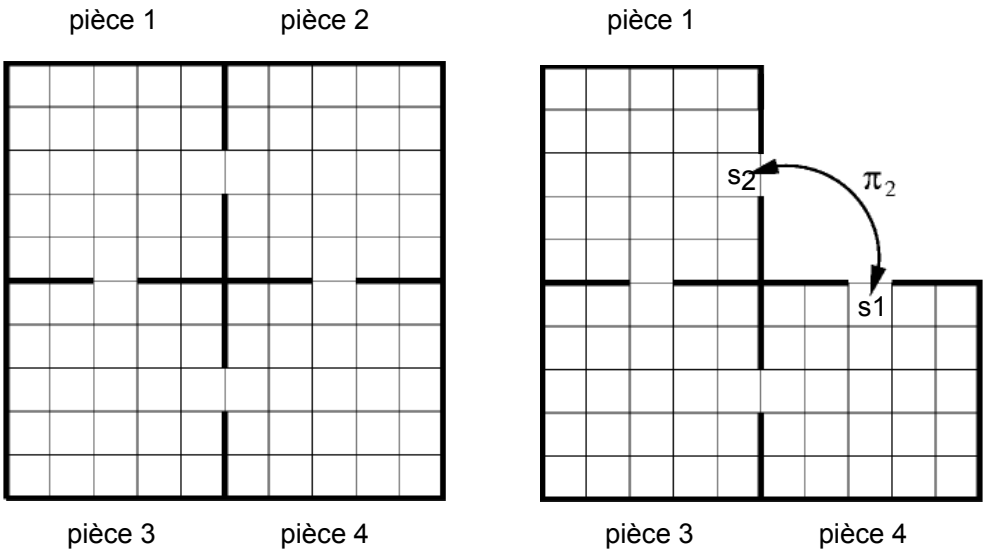


FIGURE 8.21 – a) Version à quatre pièces d'un problème de navigation. b) π_2 est une politique « gelée » et les états de la pièce 2 sont « non-contrôlés » (illustration issue de [Par98]).

Cette vision d'un PDSM avec des états « non-contrôlés » est équivalente à l'utilisation d'*options*. En effet, dans l'exemple de la figure 8.21, une *option* $o = \langle \mathcal{I}, \pi, \beta \rangle$ peut être définie telle que :

- l'ensemble d'entrée $\mathcal{I}$ comprenne les états s_1 et s_2,
- la politique de l'*option* soit égale à π_2 et détermine les actions de l'agent dans l'ensemble des états de la pièce 2,
- la condition de terminaison détermine une probabilité nulle de terminer pour tous les états de la pièce 2 et égale à 1 lorsque les états s_1 ou s_2 sont atteints.

8.5.2 Algorithme d'apprentissage par renforcement avec macro-actions

Les algorithmes classiques d'apprentissage par renforcement pour les PDM, que nous avons vu dans le chapitre 2, ont été adaptés pour les PDSM [BD95, PS97]. Nous ne présentons que l'adaptation du Q-learning, l'ensemble des autres adaptations étant triviales.

Dans un PDSM $\langle \mathcal{S}, \mathcal{A}, \mathsf{T}, \mathsf{R}, \mathsf{F} \rangle$, l'agent effectue dans l'état s_t une action a_t dont la durée est de τ pas de temps. L'agent se retrouve alors dans l'état $s_{t+\tau}$ et la mise à jour de la Q-valeur pour le couple (s_t, a_t) est alors :

$$\mathsf{Q}(s_t, a_t) \leftarrow (1-\alpha)\mathsf{Q}(s_t, a_t) + \alpha(r + \gamma^\tau \max_{u \in \mathcal{A}} \mathsf{Q}(s_{t+\tau}, u)) \tag{8.17}$$

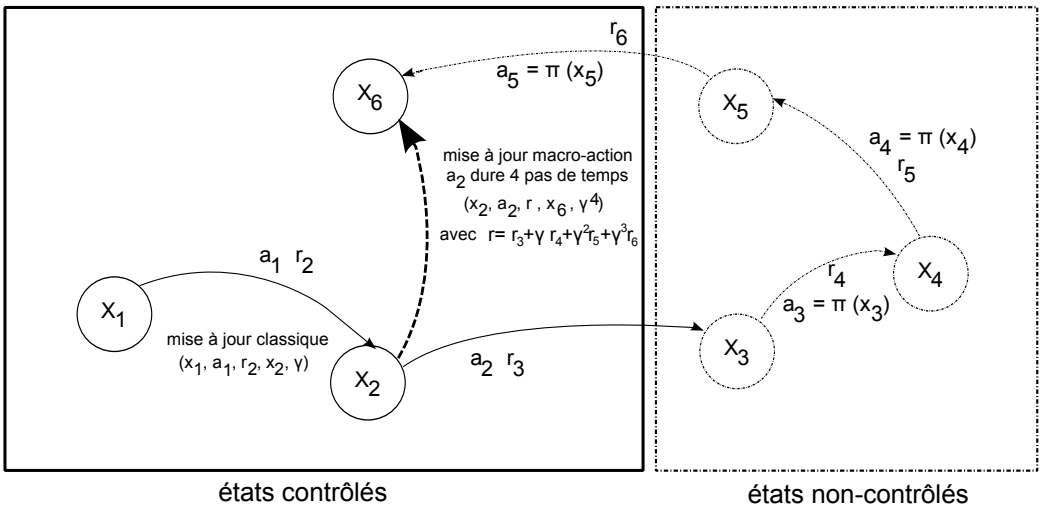


FIGURE 8.22 – PDSM Q-learning avec le formalisme des états « non-contrôlés ».

où r est l'ensemble des récompenses accumulées sur les transitions effectuées à une échelle de temps primitive, *i.e.* :

$$r = \sum_{k=0}^{\tau-1} \gamma^k r_{k+1}. \tag{8.18}$$

Nous appellerons cette adaptation du Q-learning au formalisme des PDSM le **PDSM Q-learning**. La convergence du PDSM Q-learning vers la fonction Q^* de la politique optimale du PDSM a été démontrée par [Par98].

Le PDSM Q-learning peut être utilisé dans le cas d'états « non-contrôlés » et de politiques « gelées ». Prenons l'exemple de la figure 8.22. Un agent, dans l'état initial x_1, parcourt 4 états jusqu'à atteindre un état final x_6. Les flèches modélisent les transitions effectuées par l'agent. L'action suivie et la récompense reçue sont indiquées à côté de ces flèches. Dans les états x_3, x_4 et x_5, l'agent suit une politique π. Ces états peuvent donc être considérés comme des états « non-contrôlés » dans lesquels π est une politique « gelée ». Le processus peut alors être formalisé comme un PDSM à trois états x_1, x_2 et x_6. L'action menant l'agent de l'état x_2 à l'état x_6 est alors une macro-action dont la durée est de 4 pas de temps. Les états du PDSM sont mis à jour selon le PDSM Q-learning (*cf.* équation 8.17). L'action a_1 est une action primitive donc le couple (x_1, a_1) est mis à jour selon le Q-learning classique. Par contre, dans l'état x_2, l'agent effectue l'action a_2 qui dure 4 pas de temps. Le couple (x_2, a_2) est donc mis à jour selon le PDSM Q-learning avec $\tau = 4$, $r = r_3 + \gamma r_4 + \gamma^2 r_5 + \gamma^3 r_6$ et $s_{t+\tau} = x_6$.

Le PDSM Q-learning est donc une version du Q-learning adaptée au formalisme des macro-actions et aux notions d'états « non-contrôlés » et de politiques « gelées ».

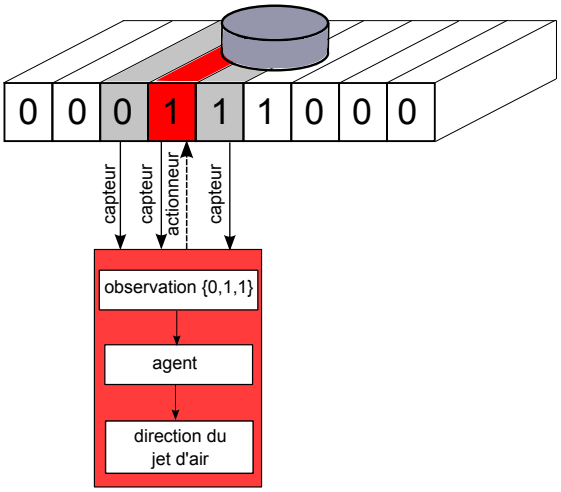


FIGURE 8.23 – Information d'un capteur partagée entre trois agents.

8.5.3 Algorithme décentralisé avec observabilités partielles

Nous proposons d'appliquer ces notions d'états « non-contrôlés » et de macro-actions à la *smart surface*. Pour cela, le point de vue local adopté avec la *smart surface* est détaillé et notamment le partage des informations des capteurs. Les observations des agents sont ensuite classées en deux catégories et les macro-actions sont introduites de sorte à étendre le formalisme des jeux de Markov d'équipe aux actions temporellement étendues. Une méthode générique pour utiliser nos algorithmes dans ce cadre est enfin proposée.

Point de vue local

La finalité du projet *smart surface* est de développer une surface totalement intégrée, composée de cellules dans lesquelles un capteur *tout-ou-rien* de présence de la pièce est associé à un actionneur, à un module de contrôle décentralisé et à un module de reconnaissance, comme illustré à la figure 8.4. Lorsque l'on se place du point de vue d'un système multiagent, chaque cellule est un agent pour lequel l'actionneur est son « bras » lui permettant d'agir sur la pièce si celle-ci est assez proche, et le capteur ses « yeux » lui permettant de savoir si la pièce est ou non au-dessus de lui. Chaque agent agit et perçoit donc de manière **locale**. L'état du système est donc **individuellement partiellement observable**.

Partage des informations des capteurs

Afin de ne pas limiter l'observation d'un agent à l'information locale retournée par son capteur, un **partage de cette information capteur** est envisagé. Ce partage permet de distribuer l'information des capteurs entre un certain nombre d'agents voisins. Ainsi, un agent pourra percevoir si la pièce est au-dessus de lui et au-dessus de ses voisins.

Prenons l'exemple illustré à la figure 8.23. La pièce recouvre trois capteurs qui sont donc activés (état 1). Les autres capteurs sont désactivés (état 0). Supposons que l'information d'un capteur est partagée par un agent et ses deux voisins latéraux. L'observation de chaque agent sera alors une combinaison des états de son capteur et de ceux de ses deux voisins. A partir de cette observation, chaque agent doit alors décider de l'action à effectuer et l'envoyer à son actionneur.

Avec qui partager ?

Le nombre d'agents avec lesquels un capteur partage son information doit être défini. Tout d'abord, tout partage d'information à un coût dû aux ressources que ce partage consomme. Un coût en capacité mémoire à allouer à chaque agent car le choix du partage détermine le nombre total d'observations d'un agent. Cette capacité mémoire est exponentielle avec le nombre de partages autorisés. Si l'information d'un capteur est partagée par un agent et ses deux voisins directs (à gauche et à droite par exemple), le nombre total d'observations par agent sera de 2^3. Si l'information est partagée avec ses huit voisins directs, le nombre total d'observations par agent sera alors de 2^9. Donc plus l'information est partagée avec un grand nombre d'agents, plus il y aura de couples observation-action par agent. Le partage à aussi un coût technique car il requiert d'intégrer des modules de partage d'information dans la *smart surface*, ce qui peut s'avérer complexe techniquement. *Il est donc préférable de limiter le nombre d'agents avec lesquels l'information d'un capteur est partagée.*

D'un autre côté, plus cette information est distribuée, plus un agent a accès à une connaissance précise de l'environnement. En d'autres termes, l'observabilité individuelle d'un agent devient **de plus en plus complète**. *La prise de décision est alors facilitée par une plus haute observabilité et donc par plus de partages.*

Ainsi, un compromis doit être trouvé entre l'utilité de partager des informations et le coût que ce partage entraîne, afin à la fois de limiter le nombre total d'observations mais aussi afin que l'observabilité individuelle ne soit pas trop partielle. *Le partage doit donc être limité au strict nécessaire.* Au §8.5.4, deux types de partages seront proposés et testés avec la simulation du convoyage 2D d'un objet sur la *smart surface*.

Algorithme décentralisé avec macro-actions

Chaque agent a accès à une observation individuelle qui est une combinaison d'informations provenant de différents capteurs. On peut alors choisir un partage qui permette à un agent d'avoir accès à assez d'informations sur le système uniquement lorsque ses actions ont une influence sur celui-ci. Par exemple, dans le cas de la *smart surface*, les jets d'air ont une portée limitée ; les actions d'un agent ont donc une influence sur un objet lorsque celui-ci se situe à une distance d'au plus 4 actionneurs voisins. Si l'objet est situé au-delà de 5 actionneurs voisins, l'agent ne pourra pas avoir d'influence dessus et donc partager l'information de ces capteurs éloignés n'apporte aucun intérêt. Par contre, partager l'information avec uniquement son voisin direct entraînera une observation in-

complète.

Les observations individuelles se classent alors en deux catégories :
- l'agent perçoit la pièce, *i.e.* que parmi tous les capteurs dont les informations sont partagées avec l'agent, au moins un est activé. Nous faisons alors l'hypothèse que l'observabilité individuelle est suffisante. Les observations sont dites « contrôlées » et l'agent peut apprendre une politique sur l'ensemble de ses observations « contrôlées »,
- l'agent ne perçoit pas la pièce, *i.e.* que tous les capteurs dont les informations sont partagées avec l'agent sont inactifs. L'agent n'a aucune information sur le système et ne peut donc ni apprendre, ni décider. Dans ce cas, l'observation est dite « non-contrôlée ». Une politique déterministe « gelée » est alors définie qui est suivie par l'agent lorsque son observation est « non-contrôlée ». Cette politique associe une probabilité de 1 à l'action « envoyer un jet d'air vertical (non-orienté) ». Ainsi, lorsque l'agent ne perçoit pas la pièce, son influence sur le système est nulle.

De manière équivalente à cette vision par observation « non-contrôlée », une *option* markovienne o_i peut être définie pour chaque agent i telle que :
- l'ensemble d'entrée de l'option soit l'ensemble des observations « non-contrôlées », noté $\mathcal{S}_{non-controle}$,
- la politique π_i suivie par un agent i durant l'exécution de l'option o_i définit une probabilité de 1 pour tout couple « observation non-contrôlée, envoyer un jet d'air vertical (non-orienté) »,
- la condition de terminaison associe une probabilité égale à 1 de terminer o_i pour toute observation « non-contrôlée ».

Les observations « non-contrôlées » peuvent être extraites du processus et le comportement d'un agent est alors défini par des macro-actions. Du point de vue d'un agent, le processus est constitué des observations « contrôlées » et d'actions temporellement étendues (ou macro-actions) qui modélisent le passage de l'agent dans les observations « non-contrôlées ». De plus, les observations « contrôlées » sont considérées par hypothèse comme des observabilités individuelles suffisantes.

Du point de vue global, le formalisme est donc un jeu de Markov d'équipe étendu aux macro-actions ou une extension des PDSM au cadre multiagent. Dans ce jeu de Markov étendu, on note $\mathcal{S}_i$ l'ensemble fini des observations « contrôlées » d'un agent i. L'ensemble fini des observations « non-contrôlées » d'un agent i est noté $\mathcal{S}_{i,non-controle}$. Une mise à jour semblable à celle du PDSM Q-learning peut alors être appliquée de manière décentralisée à chaque agent. Par exemple, si qu'un agent i ne perçoit pas la pièce pendant τ pas de temps entre les observations « contrôlées » $s_{i,t}$ et $s_{i,t+\tau}$. Cela correspond alors à une macro-action pendant laquelle l'agent reçoit la récompense accumulée r_{acc}. L'équation de mise à jour est alors :

$$\mathsf{Q}_i(s_{i,t}, a_{i,t}) \leftarrow (1-\alpha)\mathsf{Q}_i(s_{i,t}, a_{i,t}) + \alpha(r_{acc} + \gamma_{acc} \max_{u \in \mathcal{A}_i} \mathsf{Q}_i(s_{i,t+\tau}, u)) \tag{8.19}$$

Algorithme 14 : Algorithme générique pour l'extension des méthodes décentralisées aux macro-actions.

début
  Initialiser les tables nécessaires à l'apprentissage
  Initialiser l'observation initiale s_i
  tant que $s_i \in \mathcal{S}_{i,non\_controle}$ *et épisode non terminé* **faire**
    Exécuter l'action a_i selon la politique « gelée » ▷ `boucle « gelée »`
    Observer la nouvelle observation s'_i et la récompense r
    $s_i \leftarrow s'_i$
  répéter
    Choisir l'action a_i selon la méthode de décision
    Exécuter l'action a_i et observer la nouvelle observation s'_i et la récompense r
    $r_{acc} = r$
    $\gamma_{acc} = \gamma$
    tant que $s'_i \in \mathcal{S}_{i,non\_controle}$ *et épisode non terminé* **faire**
      Exécuter l'action a_i selon la politique « gelée » ▷ `boucle « gelée »`
      Observer la nouvelle perception s''_i et la récompense r
      $r_{acc} = r_{acc} + \gamma_{acc} r$
      $\gamma_{acc} = \gamma\gamma_{acc}$
      $s'_i \leftarrow s''_i$
    Mise à jour du couple (s_i, a_i) avec la récompense accumulée r_{acc}, le coefficient γ_{acc} et l'état suivant s'_i
    $s_i \leftarrow s'_i$
  jusqu'à *la fin de l'épisode*
fin

où $a_{i,t}$ est l'action effectuée lorsque l'agent observait $s_{i,t}$ et $\gamma_{acc} = \gamma^{\tau}$. On retrouve ici une version décentralisée pour agents indépendants du PDSM Q-learning (*cf.* équation 8.17) et aussi une extension du Q-learning décentralisé (*cf.* §6.2) aux macro-actions.

De manière semblable, les algorithmes étudiés dans ce mémoire dans le cadre des jeux de Markov d'équipe peuvent être adaptés aux macro-actions. L'algorithme 14 en donne la méthode générique. Les boucles « gelées » correspondent aux macro-actions lors du passage de l'agent dans les observations « non-contrôlées ». Dans ces boucles, l'agent suit la politique « gelée » et calcule la récompense accumulée r_{acc} et le coefficient d'atténuation accumulé γ_{acc} pour la macro-action débutée lors de la dernière observation « contrôlé ». Chaque transition d'une observation « contrôlée » à une autre entraîne une mise à jour avec la récompense et le coefficient d'atténuation accumulés et selon l'algorithme choisi.

8.5.4 Tâche de convoyage avec observabilités partielles

La méthode générique est maintenant appliquée en simulation à une tâche de convoyage d'un objet sur la *smart surface* dans le cas d'observabilités partielles.

Description de la tâche

Une matrice de 15×13 agents est utilisée (*cf.* figure 8.24). Le modèle est modifié de sorte à ce que chaque actionneur puisse envoyer un jet d'air orienté selon quatre directions ($\vec{x}$, $-\vec{x}$, $\vec{y}$ ou $-\vec{y}$) ou un jet d'air non-orienté (vertical). **Chaque actionneur est commandé indépendamment**. Cette surface ne possède pas de bords donc l'objet peut en sortir. Nous souhaitons convoyer « vers le nord » un cylindre (2,7mm de diamètre et 0,5mm d'épaisseur) modélisé par un polygone à 21 faces. L'objectif est que l'objet sorte de la surface au plus près du milieu du bord nord. Nous pouvons par exemple imaginer que cet objectif fasse partie d'une fonction de tri de minicomposants. Les pièces sont triées en envoyant les pièces cylindriques au nord et les autres à l'est.

Partage des informations des capteurs

Nous proposons d'étudier deux types de partages des informations des capteurs. Le premier est illustré à la figure 8.25a. L'information d'un capteur est partagée par un agent et ses quatre voisins directs (en haut, en bas, à gauche et à droite), soit 2^5 observations au total. Nous nommons ce partage le **partage direct**. Le second type de partage consiste à étendre la distribution jusqu'au quatrième voisin, ce qui correspond à la distance limite où le jet d'air n'est pas nul (*cf.* figure 8.12). Afin d'éviter une explosion combinatoire du nombre d'observations, un agent perçoit si la pièce est au-dessus de son voisin direct et si elle est au-dessus d'un au moins de ses trois voisins éloignés (*cf.* figure 8.25b). Cela correspond alors à 2^9 observations au total. Ce partage est appelé **partage étendu**.

Apprentissage par renforcement décentralisé avec macro-actions

Concernant le contrôle décentralisé par apprentissage par renforcement, on utilise la méthode générique de l'algorithme 14 dans le cadre des jeux de Markov d'équipe étendu aux macro-actions :

- $m = 13 \times 15$ est le nombre d'agents indépendants,
- chaque agent possède 5 actions : orienter le jet d'air à droite, à gauche, en haut, en bas ou ne pas l'orienter,
- chaque agent i a accès à une observation « contrôlée » notée s_i. L'ensemble des observations « contrôlées » d'un agent i est notée $\mathcal{S}_i$,
- une observation « non-contrôlée » pour un agent correspond au cas où tous les capteurs partagées avec cet agent sont inactifs (*cf.* figure 8.25a). Dans ce cas, la politique de l'agent est « gelée » ; il envoie un jet d'air vertical,
- la fonction de récompense est définie par :

$$\mathsf{R}(x_G, y_G) = \begin{cases} 10e^{-\frac{(x_G - \frac{x_L}{2})^2}{18}} \text{ si } y_G \geq y_L \\ -3 \text{ si } x_G < x_0 \text{ ou } x_G > x_L \text{ ou } y_G < y_0 \text{ ou } y_G > y_L \\ 0 \text{ sinon} \end{cases} \tag{8.20}$$

où x_0, y_0, x_L et y_L définissent les bords de la surface (*cf.* figure 8.24) et (x_G, y_G) sont les coordonnées du centre de gravité de l'objet. La récompense reçue quand

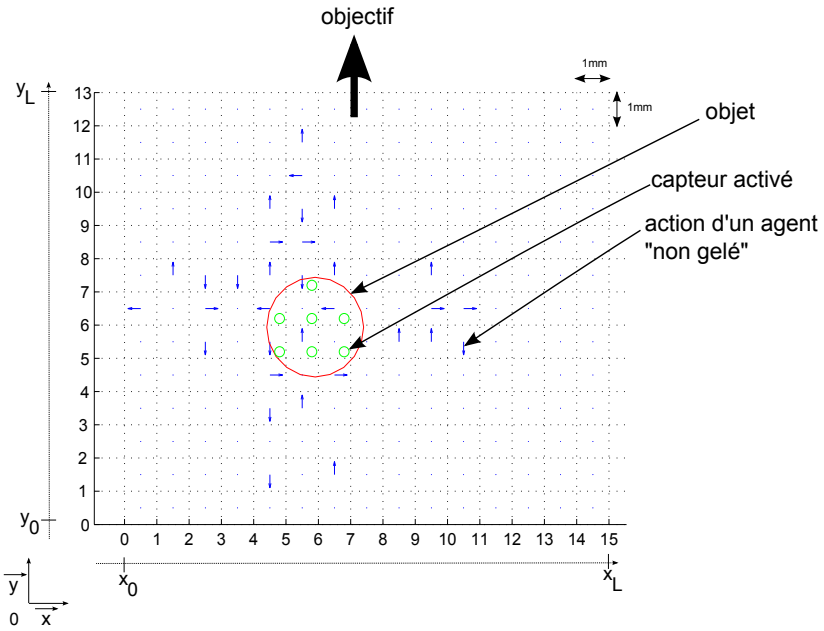
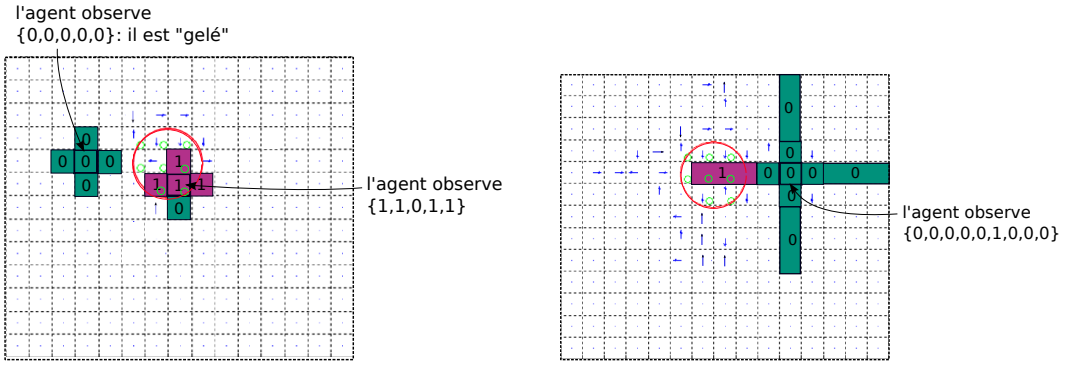


FIGURE 8.24 – Description de la tâche de convoyage (illustration issue de l'interface graphique de la simulation de notre modèle).



(a) Partage direct. (b) Partage étendu.

FIGURE 8.25 – Exemples de partage d'informations des capteurs (illustrations issues de l'interface graphique de la simulation de notre modèle).

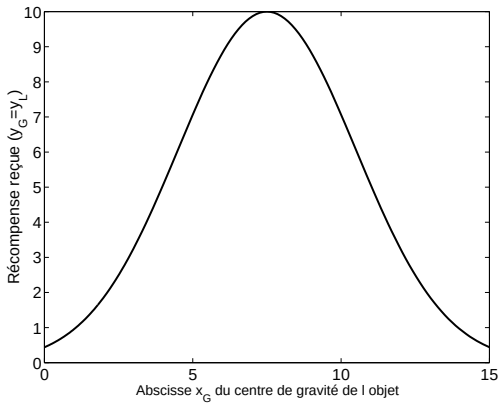


FIGURE 8.26 – Récompense reçue par les agents lorsque l'objet sort de la surface par le bord nord en fonction de l'abcisse de l'objet (x_G).

l'objet sort de la surface par le bord nord ($y_G \geq y_L$) est une fonction donnée à la figure 8.26. Nous pouvons donc remarquer que cette fonction de récompense dépend des coordonnées du centre de gravité de l'objet lorsqu'il sort de la surface.

Pour appliquer l'apprentissage par renforcement dans la forme utilisée dans ce mémoire, les échelles de temps et de position doivent être discrétisées. Le pas d'échantillonnage est de 0,02s pour le contrôle et la simulation s'effectue à pas variables (méthode d'intégration ODE45). A chaque début d'épisode, l'objet est placé en $(x_G, y_G) = (6; 6)$. Il a de plus une vitesse initiale de $\dot{x}_G = -1mm/s$. Un épisode se termine lorsque l'objet sort de la surface. L'objectif est de convoyer la pièce au milieu du bord nord de la surface. Un convoyage est réussi avec une marge ρ si, à la fin d'un épisode, on a :

$$y_G = y_L \text{ et } x_G \in [\frac{x_L}{2} - \rho, \frac{x_L}{2} + \rho]. \tag{8.21}$$

Nous testons la méthode générique de l'algorithme 14 appliquée au Q-learning décentralisé avec les deux types de partages (direct et étendu). Les algorithmes du Q-learning hystérétique et du SOoN sont aussi expérimentés avec le partage étendu. Un essai consiste en 400 épisodes avec $\alpha = 0,1$, $\beta = 0,01$, $\alpha_f = 0,01$, $\alpha_g = 0,3$ et $\gamma = 0,9$. La méthode de décision ϵ-greedy est utilisée avec une exploration globale de $\psi = 0,1$. Pour chaque algorithme, 5 essais indépendants sont réalisés. Pour chaque essai, on calcule le pourcentage d'épisodes où le convoyage a été réussi avec une marge d'erreur ρ. Les résultats sont donnés à la table 8.1.

Nous comparons tout d'abord les résultats obtenus avec les deux types de partage et la mise à jour du Q-learning décentralisé. On constate que le partage étendu permet de réussir un plus grand nombre de convoyages que le partage direct. En effet, les agents ont accès à plus d'informations provenant des capteurs. Leur observabilité individuelle est donc plus complète et la réponse des agents est mieux adaptée. Notamment, on constate en expérimentation qu'avec un partage direct, les agents ne parviennent pas à

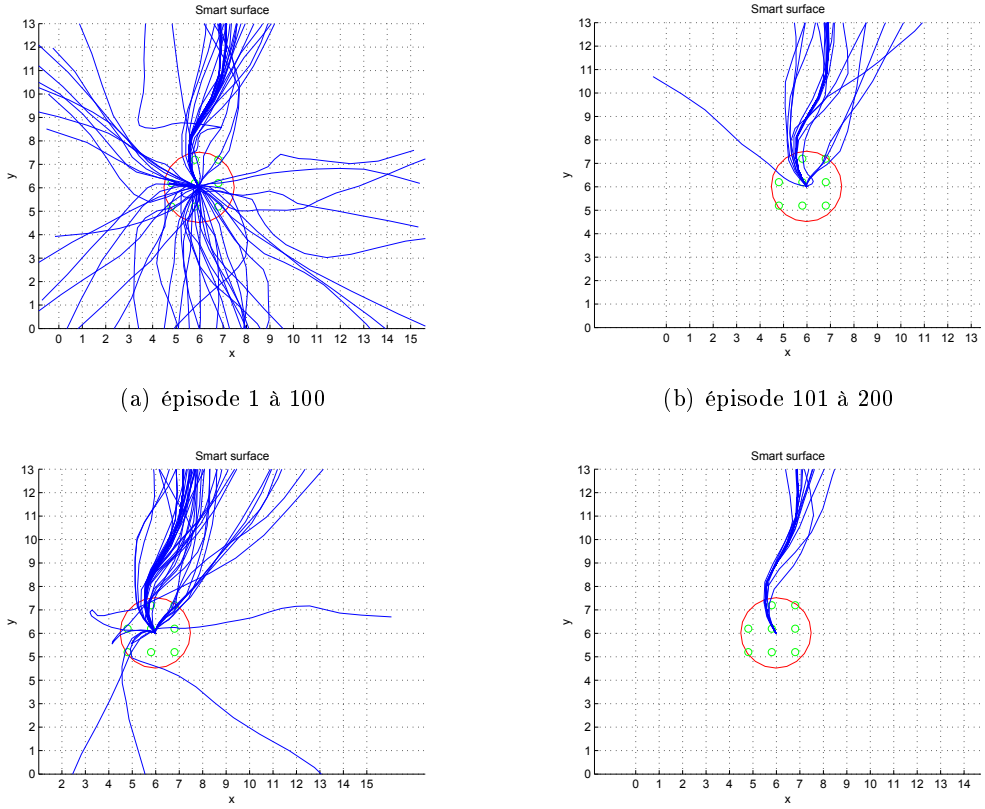
Mise à jour	Type de partage	Pourcentage d'épisodes réussis à chaque essai					Moyenne (écart-type)
Q-learning décentralisé	direct	74,5%	24,7%	2,75%	27%	67,5%	39,3% (30,6%)
Q-learning décentralisé	étendu	86%	55%	75,8%	25%	73,8%	63,1% (24%)
Q-learning hystérétique	étendu	10,6%	3,8%	5%	1,3%	73,5%	18,9% (30,7%)
SOoN	étendu	88%	82%	83,5%	71%	79,75%	80,9% (6,3%)

TABLE 8.1 – Pourcentage d'épisodes où le convoyage a été réussi (moyenne sur 5 essais indépendants). La marge d'erreur est $\rho = 2,5$mm.

« corriger » la trajectoire de l'objet car ils ne « voient » pas assez loin. Supposons par exemple que l'objet prenne une mauvaise trajectoire et que les agents doivent alors le ralentir en envoyant des jets d'air orientés sur sa tranche avant. Avec un partage direct, chaque agent n'oriente son jet d'air que lorsqu'il perçoit l'objet, *i.e.* lorsque l'objet active le capteur de son voisin direct. Par contre, avec un partage étendu, un agent pourra commencer à agir sur l'objet dès que celui-ci activera le capteur de son quatrième voisin. Le ralentissement de l'objet sera donc plus efficace dans le second cas.

Par contre, avec les deux types de partage, l'écart-type est élevé. D'un essai à l'autre, le nombre de réussites peut fortement varier. Par exemple, lors du quatrième essai, seul 25% des convoyages sont réussis. En effet, les agents semblent peu robustes à l'exploration des autres car on constate des phases successives d'apprentissage et de désapprentissage pendant les simulations.

Concernant les trois mises à jour testées avec un partage étendu, le meilleur résultat est obtenu avec le SOoN. 80,9% des convoyages sont réussis et l'écart type est faible. La figure 8.27 illustre les différentes étapes d'un apprentissage lors d'un essai. Les trajectoires de l'objet pour chaque centaine d'épisodes sont tracées. Lors des 100 premiers épisodes (*cf.* figure 8.27a), la plupart des convoyages sont insatisfaisants. En effet, c'est le début de l'apprentissage et étant donné l'initialisation choisie, les agents ont un comportement aléatoire. De plus, en début d'apprentissage, les agents sont optimistes dans un environnement stochastique. La coordination est donc difficile. Pendant les 200 épisodes suivants (*cf.* figure 8.27b et c), les agents effectuent leur adaptation automatique à l'environnement. La majorité des convoyages envoient l'objet au nord mais ces convoyages sont peu précis. Quelques convoyages échouent ; ils correspondent à des actions d'exploration des agents qui font suivre à l'objet une trajectoire difficilement corrigeable. Des trajectoires risquées et corrigées par les agents sont observées à la figure 8.27c. Enfin, lors des 100 derniers épisodes (*cf.* figure 8.27d), toutes les trajectoires sont réussies. Les agents se sont adaptés et réussissent à se coordonner pour convoyer précisément l'objet. De plus, les trajectoires risquées dues à des actions d'exploration sont rapidement corri-



(a) épisode 1 à 100

(b) épisode 101 à 200

(c) épisode 201 à 300

(d) épisode 301 à 400

FIGURE 8.27 – Trajectoires de l'objet lors de l'apprentissage des agents avec le SOoN et des observabilités partielles.

gées par les agents.

Ainsi, le SOoN peut être étendu à un cas particulier d'observabilités partielles grâce à l'introduction de macro-actions et d'observabilités « non-contrôlées ». Les deux phases caractéristiques du SOoN sont de plus distinguées : l'adaptation à l'environnement avec tout d'abord des agents optimistes qui ajustent ensuite leurs évaluations afin de réussir leur coordination.

8.6 Conclusion

Dans ce chapitre, l'application de nos travaux à un système distribué de micromanipulation a été présentée. Les deux approches envisagées pour cette application ont été exposées. Tout d'abord, le prototype de surface active pneumatique a été introduit, puis le modèle de convoyage 2D utilisé pour nos simulations a été détaillé. L'approche par apprentissage par renforcement a ensuite été validée en tant que méthode de contrôle adaptée à ce type de système non-linéaire et stochastique. Une comparaison avec la méthode par champ de vecteur programmable a été effectuée en simulation pour cette validation. Nous nous sommes alors intéressés au cas d'observabilités partielles en vue d'appliquer nos algorithmes à la *smart surface* totalement intégrée. Pour cela, des macro-actions et des observations « non-contrôlées » ont été utilisées afin de pouvoir considérer le système comme un jeu de Markov d'équipe étendu aux actions temporellement abstraites. Une méthode générique d'extension de nos algorithmes à ce cadre a été détaillée. Enfin, des premiers tests en simulation sur une tâche de convoyage ont été réalisés. Ils ont permis d'une part de comparer différents partages d'informations des capteurs. D'autre part, les premiers résultats obtenus avec le SOoN étendu aux observabilités partielles dégagent de nombreuses perspectives dans l'optique d'une commande décentralisée de la *smart surface*.

Afin d'améliorer ces performances, l'observation au pas précédent pourrait aussi être mémorisée afin de mieux prendre en compte la dynamique de l'objet. De plus, les champs de vecteur programmable pourraient être utilisés pour initialiser l'apprentissage. Ce biais serait intégré dans les valeurs initiales de la fonction de valeur afin d'être éphémère et d'éviter tout risque de comportements non désirés, comme expliqué au chapitre 3. Enfin, la méthode développée devra être appliquée à court terme sur le prototype de surface active. Afin d'avoir un point de vue local, les capteurs et le partage d'informations seront « émulés » à l'aide d'une caméra, d'un mécanisme d'allocation individuelle des actionneurs et d'une unité centrale de calcul.

Chapitre 9

Conclusion et perspectives

9.1 Bilan des travaux

L'objectif de cette thèse était l'étude de la coordination d'agents indépendants, adaptatifs et coopératifs dans le cas d'une observabilité individuelle totale de l'environnement. Le point de vue adopté a été d'utiliser conjointement l'apprentissage par renforcement et les systèmes multiagents afin d'apporter les quatre contributions principales suivantes.

Injection de connaissances dans le cas de problèmes de plus court chemin stochastique

Nous avons considéré en premier lieu le domaine de l'apprentissage par renforcement, ce qui nous a conduits à étudier différentes approches d'injection de connaissances dans le cas de problèmes de plus court chemin stochastique. Selon l'initialisation uniforme de la fonction Q, des comportements peuvent être distingués en début d'apprentissage : le comportement de piétinement doit toujours être évité alors que celui d'exploration systématique peut être choisi afin d'accélérer le début de l'apprentissage. Des choix concernant la fonction de récompense et l'initialisation de la fonction de valeur ont aussi été préconisés. Notamment, une initialisation à l'aide d'une fonction d'influence générique efficace et non risquée a été proposée. Elle permet d'améliorer les performances de l'apprentissage pour les problèmes de plus court chemin stochastique.

Étude des enjeux de l'apprentissage d'agents indépendants

En second lieu, nous avons considéré le champ des systèmes multiagents et leurs liens avec l'apprentissage. Ceci nous a permis d'une part de préciser différents concepts mis en oeuvre dans ce mémoire, tels que la coopération ou la coordination, et d'autre part d'exposer les formalismes des jeux matriciels et des jeux de Markov d'équipe.

Les objectifs de l'apprentissage ont été précisés et nous avons établi que l'optimalité globale pouvait être atteinte à travers un point de vue local dans le cadre coopératif.

En effet, il est nécessaire que chaque agent trouve sa fonction de valeur locale optimale pour que les actions jointes soient optimales. Cette condition suppose qu'une hypothèse optimiste soit posée.

Nous avons ensuite mis en évidence un certain nombre de difficultés que doivent surmonter des agents indépendants dans des jeux de Markov d'équipe. Un contre-exemple a permis d'illustrer que la propriété fondamentale de Markov n'est plus nécessairement vérifiée au niveau local. Divers facteurs pouvant compliquer la coordination ont été isolés et interprétés sur des jeux matriciels : les équilibres cachés, la sélection d'équilibres et l'environnement bruité. Enfin, l'impact de l'exploration sur l'apprentissage d'agents indépendants a été mis en évidence à travers des phénomènes de destruction de la stratégie optimale. Trouver un algorithme robuste face à la stratégie d'exploration est un enjeu primordial concernant les agents indépendants.

A travers une partie bibliographique proposant une notation uniforme, les algorithmes existants fondés sur le Q-learning et ses variantes ont été analysés selon les critères précédents : la robustesse face à l'exploration, l'apprentissage de politiques individuelles optimales malgré les phénomènes d'équilibres cachés et de bruit, et la sélection d'équilibres. Au vu de cette analyse, nous nous sommes orientés vers *le développement d'algorithmes robustes face à l'exploration et capables de surmonter les facteurs de non-coordination dans les jeux de Markov d'équipe*. Pour cela, deux méthodes présentant des points communs et des caractéristiques intéressantes face aux enjeux de l'apprentissage sont mises en avant. Le Q-learning distribué d'une part, qui est robuste face à l'exploration et possède une preuve de convergence vers un équilibre de Nash Pareto optimal dans tous jeux de Markov d'équipe déterministes. Le *Frequency Maximum Q-value* (FMQ) d'autre part, qui surmonte la difficulté d'environnements partiellement bruités dans les jeux matriciels en découplant le bruit dû à l'exploration des agents du bruit dû à l'environnement.

Approches développées

Deux approches ont alors été proposées pour l'apprentissage par renforcement d'agents indépendants dans le cadre des jeux de Markov d'équipe. Pour cela, il est nécessaire de définir pour un agent indépendant une évaluation correcte de ses actions individuelles dans chaque état du système. Le premier algorithme est appelé le **Q-learning hystérétique**. Il repose sur une extension du Q-learning décentralisé utilisant des agents « à tendance optimiste réglable » grâce à deux vitesses d'apprentissage. Les agents hystérétiques sont alors « à forte tendance optimiste » car ils accordent peu d'importance aux pénalités reçues mais sans les ignorer totalement comme le ferait le Q-learning distribué. Malgré les améliorations qu'il reste à apporter à cet algorithme, les agents hystérétiques sont simples à mettre en oeuvre et les résultats obtenus sur divers benchmarks confirment la pertinence de cette évaluation hystérétique.

Le second algorithme est le ***Swing between Optimistic or Neutral*** (SOoN). Son principe est de permettre à des agents indépendants d'adapter automatiquement leurs évaluations à la stochasticité de l'environnement. Pour cela, les couples état-action sont

TABLE 9.1 – Caractéristiques des algorithmes décentralisés d'apprentissage par renforcement pour agents indépendants dans les jeux d'équipe. « NT » signifie que la caractéristique n'a pas été testée.

	Jeux matriciels	Jeux de Markov	Sélection d'équilibres	Équilibres cachés	Environnement stochastique	Stratégie d'exploration
Q-Learning décentralisé [WD92]	✓	✓	✓ avec GLIE		✓ partiellement	GLIE
Q-Learning distribué [LR00]	✓	✓	✓	✓		stationnaire
Agents indulgents [PSL06]	✓		NT	NT	NT	GLIE
WoLF PHC [BV02]	✓	✓	✓		NT	stationnaire
FMQ [KK02]	✓		✓ avec GLIE	✓	✓ partiellement	GLIE
Q-learning indicé [LR04]	✓		✓	✓	✓	stationnaire
Q-learning hystérétique	✓	✓	✓ avec GLIE	✓	✓ partiellement	GLIE
SOoN	✓	✓	✓	✓	✓ partiellement	stationnaire

évalués par une nouvelle heuristique qui effectue une interpolation linéaire entre les évaluations moyennes du Q-learning décentralisé et les évaluations optimistes du Q-learning distribué. Au départ, l'environnement est supposé déterministe et les agents sont donc optimistes. Puis, l'algorithme s'adapte automatiquement à la stochasticité de son environnement à travers l'estimation de la probabilité de réalisation du gain optimal.

Pour chacun de ces algorithmes, nous nous sommes efforcés de préciser les choix des différents paramètres de l'apprentissage de sorte à assurer la robustesse de ces méthodes face à l'exploration des autres. Pour cela, la notion d'**exploration globale** a été introduite. Elle permet de quantifier le bruit dans le système dû à l'exploration de tous les agents. Concernant l'algorithme SOoN, le réglage des paramètres est simple. Il a été précisé en fonction de l'exploration globale choisie de sorte à obtenir une fréquence robuste à l'exploration des autres.

Ces deux méthodes ont été testées avec succès sur divers jeux de Markov d'équipe. Les résultats obtenus ont notamment illustré que l'algorithme SOoN réussit efficacement la coordination à travers deux phases caractéristiques : une première phase d'adaptation

automatique des évaluations qui fluctuent entre des évaluations optimistes et moyennes, et une seconde phase de coordination où les évaluations se sont ajustées. Ces résultats confirment de plus que l'interpolation linéaire utilisée par le SOoN est une bonne heuristique d'évaluation des actions proche des valeurs réelles.

La table 9.1 synthétise les caractéristiques des algorithmes présentés et développés dans ce mémoire. Une stratégie d'exploration stationnaire permet d'éviter le choix parfois complexe des paramètres de décroissance d'une stratégie GLIE [1].

Commande décentralisée d'un système distribué de micromanipulation

Ces travaux ont été appliqués à la commande décentralisée d'un système distribué de micromanipulation, appelé *smart surface*, dans le cadre d'un projet de l'agence nationale de la recherche. Un système distribué de micromanipulation est une matrice de microactionneurs pouvant être considérée comme un système multiagent coopératif où l'ensemble des agents (ou microactionneurs) autonomes ont un objectif compatible qui est le positionnement ou le convoyage d'un objet sur la matrice.

L'approche décentralisée par apprentissage par renforcement multiagent a été validée sur une tâche de positionnement grâce au modèle de convoyage en deux dimensions que nous avons développé. Ensuite, nous avons proposé une méthode générique pour étendre les algorithmes développés dans ce mémoire à un cas d'observabilités partielles dans l'optique du contrôle de la *smart surface* totalement intégrée. Des macro-actions et des observabilités « non-contrôlées » ont été utilisées afin de pouvoir étendre les jeux de Markov d'équipe aux actions temporellement étendues. L'observabilité individuelle des agents est aussi complétée par le partage des informations des capteurs. Les résultats obtenus en simulation sur une tâche de convoyage, notamment avec le SOoN, sont exposés.

9.2 Perspectives

De nombreuses perspectives de recherches se dessinent à partir de ces travaux. Une première partie concerne les évolutions potentielles des méthodes développées. Une seconde partie de ces perspectives se rapporte à l'application de ces méthodes à la commande décentralisée de systèmes.

9.2.1 Évolutions potentielles des méthodes développées

Enrichissement de la méthode hystérétique

Le principe des agents hystérétiques pourrait être étendu à d'autres algorithmes décentralisés d'apprentissage par renforcement. Particulièrement, il pourrait être associé aux traces d'éligibilité dans une version décentralisée hystérétique des algorithmes TD(λ). Une étude plus approfondie du choix du coefficient β de décroissance des Q-valeurs dans

1. *Greedy in the Limit with Infinite Exploration.*

la méthode hystérétique pourrait être réalisée afin notamment de connaître l'influence de ce paramètre sur la performance de la méthode.

Jeux coopératifs

Les méthodes proposées pourraient être testées avec des situations de coopération où la récompense est distribuée localement de sorte à expérimenter l'applicabilité de nos algorithmes au cadre plus général des jeux de Markov d'équipe coopératifs. Dans ces jeux, la satisfaction individuelle n'est alors pas nécessairement celle du groupe, et un équilibre de Nash n'est pas nécessairement un optimum de Pareto. Le dilemme du prisonnier et sa version étendue à n agents et m actions, appelée « dilemme social multiagent » [SG03], pourraient par exemple être utilisés en premier lieu.

Adaptation aux systèmes continus

Une autre perspective concerne le problème d'adaptation de nos algorithmes à des espaces d'états de grandes tailles. Afin d'éviter l'emploi de tables pour représenter les fonctions de valeurs ou les fréquences, il serait intéressant de combiner les solutions utilisées dans le cadre mono-agent, telles que les méthodes basées sur les fonctions d'approximation, avec nos algorithmes dans l'optique notamment d'améliorer leurs performances lors de la commande de systèmes continus.

Partage de l'information d'exploration

Un enjeu important de l'apprentissage d'agents indépendants mis en évidence par ces travaux est la robustesse des méthodes face à l'exploration globale. En effet, l'exploration d'un agent peut détruire la politique apprise ou la fréquence d'occurrence d'un de ses congénères. Un mécanisme de partage de l'information d'exploration entre agents indépendants pourrait être mis en oeuvre. Ce mécanisme pourrait consister en un réseau de diffusion où chaque agent « explore et donc émet » ou « n'explore pas et donc écoute ». Ainsi, chaque agent pourrait savoir si les actions des autres sont des actions d'exploration ou d'exploitation. Il pourrait alors aisément découpler les diverses causes de bruit. Les phénomènes de « destruction » seraient évités grâce à une simple suspension de la mise à jour des agents qui exploitent lorsqu'un ou plusieurs des congénères explorent. Cette méthode pourrait facilement être appliquée à de nombreux algorithmes, et en particulier à nos méthodes. Elle permettrait de supprimer une des difficultés de l'apprentissage d'agents indépendants et offrirait des perspectives concernant l'amélioration des performances des agents.

9.2.2 Applications

Concernant la commande décentralisée d'un système distribué de micromanipulation, de nombreuses perspectives se dégagent des premiers résultats présentés dans ce mémoire.

Les méthodes développées seront appliquées à court terme sur le prototype de surface active dans le cadre du projet de l'agence nationale de la recherche. Afin d'avoir un point de vue local, les capteurs et le partage d'informations seront « émulés » à l'aide d'une caméra, d'un mécanisme d'allocation individuelle des actionneurs et d'une unité centrale de calcul. La perspective à long terme étant évidemment d'implanter nos méthodes sur une *smart surface* totalement intégrée.

Cependant, l'apprentissage en ligne d'un processus réel présente des résultats mitigés concernant la durée de l'apprentissage. Des mécanismes de généralisation ou d'apprentissage progressif (en utilisant par exemple une initialisation de l'apprentissage sur simulation grâce à notre modèle) seront nécessaires. Les résultats pourraient aussi être améliorés grâce à une meilleure prise en compte de la dynamique de l'objet et grâce à l'utilisation de méthodes par champ de vecteur programmable pour initialiser l'apprentissage de manière simple et non risquée.

Des tâches plus complexes devront aussi être réalisées, comme le contrôle d'une trajectoire donnée sur la surface ou la possibilité de gérer plusieurs objets présents sur la matrice.

Enfin, au-delà de la commande d'un système distribué de micromanipulation, nos méthodes décentralisées d'apprentissage par renforcement pour agents indépendants et coopératifs pourraient être utilisées comme un moyen pour commander des systèmes divisés en plusieurs entités afin de diminuer leur complexité. Cela pourrait être appliqué à divers systèmes, par exemple en robotique avec la coordination des doigts d'un préhenseur ou des pattes d'un robot hexapode, ou en microrobotique avec la commande d'une aile de libellule.

Annexe A

Choix des paramètres de l'apprentissage pour les expériences sur les jeux matriciels à $n > 2$ agents

Q-learning décentralisé

nombre d'agents	répétitions par épisode	paramètres de l'algorithme	méthode de décision	stratégie d'exploration
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0003 * t)$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0002 * t)$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0001 * t)$

TABLE A.1 – Choix des paramètres de l'apprentissage avec le Q-learning décentralisé.

FMQ récursif

nombre d'agents	répétitions par épisode	paramètres de l'algorithme	méthode de décision	stratégie d'exploration
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\alpha_f = 0.01$ $\gamma = 0$ $\mathsf{Q}_{max,ini} = -100$	ϵ-glouton	stationnaire $\psi = 0.1$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\alpha_f = 0.001$ $\gamma = 0$ $\mathsf{Q}_{max,ini} = -100$	ϵ-glouton	stationnaire $\psi = 0.15$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\alpha_f = 0.0001$ $\gamma = 0$ $\mathsf{Q}_{max,ini} = -100$	ϵ-glouton	stationnaire $\psi = 0.2$

TABLE A.2 – Choix des paramètres de l'apprentissage avec le FMQ récursif.

WoLF-PHC

nombre d'agents	répétitions par épisode	paramètres de l'algorithme	méthode de décision	stratégie d'exploration
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\delta_{perdre} = 0.006$ $\gamma = 0$ $\delta_{gagner} = 0.001$	ϵ-glouton	stationnaire $\psi = 0.1$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\delta_{perdre} = 0.0006$ $\gamma = 0$ $\delta_{gagner} = 0.0001$	ϵ-glouton	stationnaire $\psi = 0.15$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\delta_{perdre} = 0.0006$ $\gamma = 0$ $\delta_{gagner} = 0.0001$	ϵ-glouton	stationnaire $\psi = 0.2$

TABLE A.3 – Choix des paramètres de l'apprentissage avec le WoLF-PHC.

Annexe B

Descriptif des *benchmarks* utilisés

L'ensemble des *benchmarks* présentés dans cette annexe et utilisés dans ce mémoire ont été implantés dans une **bibliothèque d'outils pour Matlab-Simulink pour l'apprentissage par renforcement**, nommée **BOSAR**. Cette bibliothèque est sous licence GNU GPL et a été développée en C/C++ durant cette thèse. Elle est disponible en libre accès sur le site http ://www.lab.cnrs.fr/openblockslib/. C'est une bibliothèque de fonctions qui permet de développer et d'étudier les algorithmes d'apprentissage par renforcement. L'ensemble des fonctions et *benchmarks* qui y sont disponibles sont documentés dans une aide en ligne.

B.1 Jeu de coordination à 2 agents de Boutilier

Craig Boutilier a introduit dans [Bou99] un jeu de Markov d'équipe pour la coordination de 2 agents, donné à la figure B.1. Ce jeu a 7 états. Chaque agent a le choix parmi deux actions a et b. Le jeu débute avec les deux agents dans l'état s_1. Les transitions sur la figure sont indiquées par un couple d'actions où se retrouvent respectivement l'action de l'agent 1 et celle de l'agent 2. Le sigle $*$ est un joker représentant une action quelconque. Aucune récompense n'est reçue sauf lorsque les états absorbants s_4, s_5 et s_6 sont atteints. Les récompenses reçues sont alors respectivement 11, une pénalité de k et 7. Dans la version stochastique de ce jeu, la récompense dans l'état s_6 est bruitée : les récompenses 0 et 14 sont reçues de manière équiprobable.

Dans l'état s_2, les agents doivent se coordonner en choisissant tous deux la même action individuelle. Deux actions jointes sont donc optimales dans cet état : $\langle a, a\rangle$ et $\langle b, b\rangle$. Les agents doivent aussi se coordonner pour choisir la même action jointe optimale ; ils sont face à un problème de sélection d'équilibres. De plus, s'ils échouent à se coordonner dans cet état, ils sont alors pénalisés. Les actions jointes optimales sont donc cachées par ces pénalités de non-coordination.

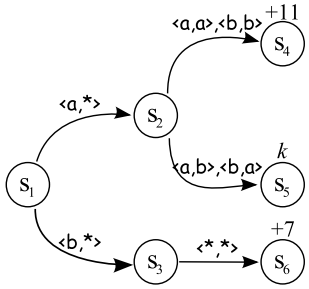


FIGURE B.1 – Jeu de coordination à 2 agents de Boutilier.

B.2 Jeu de poursuite à deux agents

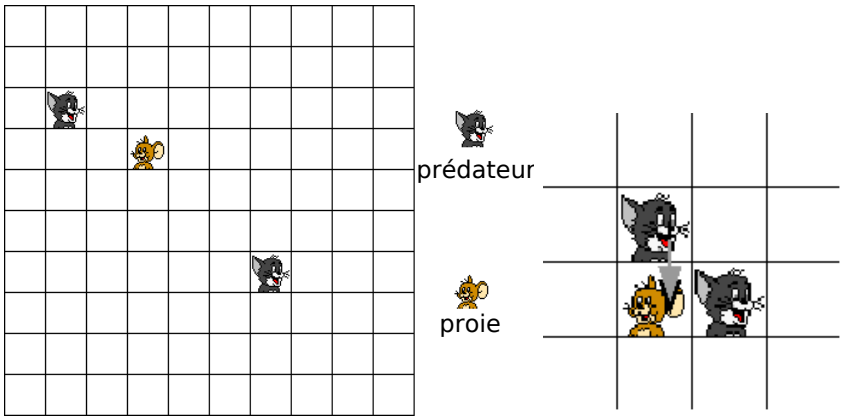
Description du problème

Le jeu de la poursuite est un problème multi-agent très répandu dans lequel un ensemble d'agents coopérant, appelés prédateurs, doivent se coordonner afin de capturer une ou plusieurs proies [BJD86]. Plusieurs instances de ce problème existent. Elles diffèrent à plusieurs niveaux : le nombre de prédateurs, le nombre de proies, la méthode de capture ou les hypothèses faites sur l'environnement telles que les perceptions des prédateurs.

Nous choisissons l'instance proposée par [KV04, JRK06] dans laquelle 2 prédateurs doivent coordonner leurs actions afin de capturer une unique proie dans un labyrinthe toroïdal de taille 10×10 (*cf.* figure B.2a). Au début de chaque épisode, les prédateurs et la proie sont positionnés aléatoirement dans des cellules distinctes. A chaque pas, tous les prédateurs exécutent simultanément une de leur 5 actions possibles : nord, sud, est, ouest ou rien. La proie suit une politique aléatoire : elle reste dans sa position courante avec une probabilité de $0,2$ et sinon, elle se déplace dans l'une des cellules libres adjacentes à la sienne (nord, sud, est, ouest) avec une probabilité uniforme. La proie est capturée lorsque les deux prédateurs sont positionnés dans deux cellules adjacentes à la proie, et que l'un d'entre eux se déplace dans la même cellule que celle de la proie pendant que l'autre reste sur place pour le supporter. Une situation possible de capture est donnée à la figure B.2b. Lorsque la proie est capturée, l'épisode se termine.

Description des agents

– **actions** : chaque prédateur dispose de cinq actions possibles, correspondant à des déplacements d'une case vers le nord, l'ouest, le sud, l'est ou aucun déplacement. On peut ajouter de l'incertitude quant au résultat de ces actions en ajoutant un bruit. Si ce bruit est effectivement ajouté, le déplacement choisi est 8 fois sur 10 obtenu, les déplacements « voisins » étant obtenus de manière équiprobable le reste du temps,



(a) Situation quelconque

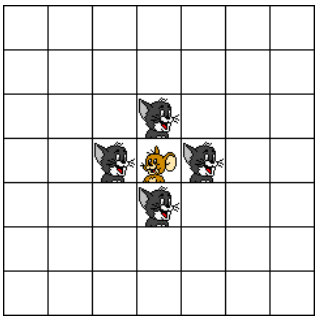
(b) Situation possible de capture

FIGURE B.2 – Jeu de poursuite à 2 prédateurs et une proie dans un labyrinthe 10×10.

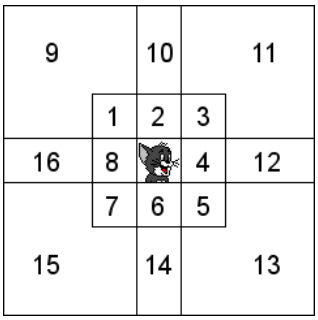
- **perceptions** : chaque prédateur peut percevoir sa position relative par rapport à la proie ainsi que celle de son congénère. L'espace d'état est donc l'ensemble des combinaisons des positions relatives des deux prédateurs, *i.e.* 99×98 états. On a donc dans le cas centralisé une table de Q valeurs de taille $99 \times 98 \times 5^2$, soit 242550 paires d'état-action jointe. Dans le cas décentralisé, on a deux tables Q_i chacune de taille $99 \times 98 \times 5$, soit 48510 paires d'état-action pour chaque prédateur,
- **récompense** : lorsque la proie est capturée, les deux prédateurs reçoivent une récompense $r = 10$. Si les prédateurs se retrouvent dans la même cellule, ou si un des prédateurs se déplace dans la même cellule que la proie sans être supporté par son congénère, alors ils sont tous deux pénalisés ($r = -50$) et replacés aléatoirement dans des cellules libres et distinctes du labyrinthe. Dans tous les autres cas, la récompense est $r = 0$.

Enjeux face à la coordination

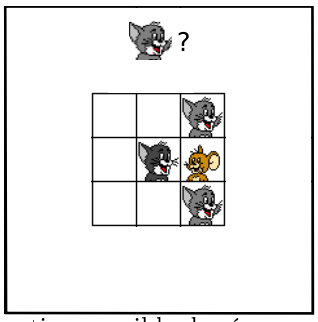
Dans ce jeu de poursuite, les agents doivent coordonner leurs actions à travers une combinaison spécifique de leurs actions individuelles afin de capturer la proie. Des pénalités sont de plus associées à une non-coordination des actions. Les équilibres optimaux sont donc risqués. L'environnement est donc stochastique car la proie présente un comportement aléatoire. De plus, on peut aussi ajouter de l'incertitude sur le résultat des actions des prédateurs.



(a) Situation de capture



(b)Perceptions d'un agent



c) Situation possible de récompense (autres situations obtenues par rotation de 90°.)

Un prédateur est récompensé lorsqu'il se situe dans une cellule adjacente à la proie et à l'ouest de celle-ci et qu'il perçoit deux autres prédateurs entourant la proie.

FIGURE B.3 – Jeu de poursuite à 4 prédateurs et une proie dans un labyrinthe 7×7.

B.3 Jeu de poursuite à quatre agents et perceptions partielles

Description du problème

Ce jeu de poursuite est inspiré du problème posé par [BJD86, Buf03] dans lequel quatre prédateurs doivent encercler une proie pour la capturer (*cf.* figure B.3a). L'environnement est un labyrinthe toroïdal de taille 7×7.

Au début de chaque épisode, les prédateurs et la proie sont positionnés aléatoirement dans des cellules distinctes. A chaque pas, tous les prédateurs exécutent simultanément une de leur 5 actions possibles : nord, sud, est, ouest ou rien. La proie suit une politique aléatoire : elle reste dans sa position courante avec une probabilité de $0,5$ et sinon, elle se déplace dans l'une des cellules libres adjacentes à la sienne (nord, sud, est, ouest) avec une probabilité uniforme. Lorsque la proie est capturée, l'épisode se termine. Les collisions des agents sont bloquées.

Description des agents

– **actions** : chaque prédateur dispose de cinq actions possibles, correspondant à des déplacements d'une case vers le nord, l'ouest, le sud, l'est ou aucun déplacement.

Les collisions des prédateurs entre eux et avec la proie sont bloquées, *i.e.* que si leurs actions entraînent une collision, ils restent sur place,
- **perceptions** : chaque prédateur perçoit ses 3 congénères et la proie avec une faible résolution. Ainsi, chacun des 4 autres est perçu selon une des 4 directions cardinales et un critère de proximité (« Est-il sur l'une des huit cases entourant l'agent ? »). On a ainsi 16 cas possibles pour chacun des quatre autres perçus (*cf.* figure B.3b), soit 16^4 perceptions possibles par agent.
- **récompense** : lorsqu'au moins un des prédateurs est dans la position de la figure B.3c, ils reçoivent tous une récompense $r = 25$. Sinon, ils reçoivent $r = 0$.

Enjeux face à la coordination

La faible résolution induite dans les perceptions entraîne du bruit dans l'environnement, ce qui est ici le facteur principal compliquant la coordination des agents.

B.4 Version discrète de la smart surface

Description du problème

Étant donnée notre application réelle qui est le contrôle décentralisé d'une surface active, nous avons développé une version discrète simplifiée de ce système.

Le système est composé de 270 agents (ou actionneurs) organisés en matrice (*cf.* figure B.4). Une pièce est positionnée sur cette surface. Elle recouvre plusieurs agents et est symbolisée par un rectangle. Au début d'un épisode, la pièce est dans sa position initiale. L'épisode se termine lorsque la pièce atteint sa position finale (état absorbant). La pièce se déplace case par case (une ou plusieurs) en translation sur la surface mais sa rotation est bloquée. De plus, elle ne peut pas sortir de la surface et reste au bord de celle-ci. Le déplacement de la pièce sur la surface est proportionnel à la somme des actions des agents situés en-dessous de la pièce et à sa périphérie. L'influence des agents situés à la périphérie de la pièce est trois fois supérieure à l'influence des agents situés en-dessous. En effet, sur le prototype réel, un jeu d'air orienté qui atteint la tranche de la pièce a beaucoup plus d'influence qu'un jet d'air orienté qui atteint la face inférieure de la pièce. Par exemple, dans le cas de la figure B.4, deux agents situés sous la pièce poussent vers le sud, un agent situé sous la pièce pousse vers le nord, et un agent situé à sa périphérie pousse vers le sud. La pièce va donc se déplacer de quatre cases vers le sud. De plus, afin de prendre en compte les facteurs de dispersion présents pendant le processus de fabrication des micro-actionneurs MEMS du prototype réel, de l'incertitude sur le résultat des actions des agents est intégrée dans ce *benchmark*.

Description des agents

- **actions** : chaque agent(ou actionneur) dispose de cinq actions possibles, correspondant à pousser vers le nord, l'ouest, le sud, l'est ou à ne rien faire. Ces actions

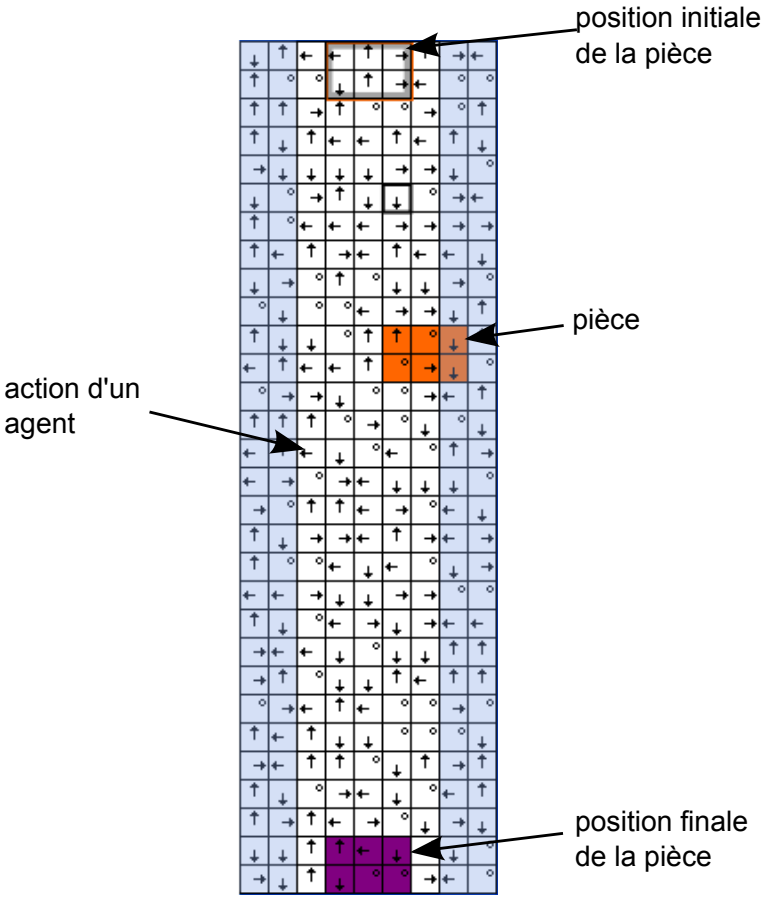


FIGURE B.4 – Version discrète de la smart surface avec 9×30 agents.

sont représentées sur la figure B.4 par des flèches indiquant le sens de poussée de l'agent, ou par un rond si l'agent n'effectue aucune poussée. A chaque pas de temps, tous les agents effectuent simultanément une action. De l'incertitude est ajoutée quant au résultat de ces actions : l'action de chaque actionneur est aléatoire parmi les 5 possibles (choix équiprobable) avec une probabilité de $0,001$,
- **perceptions** : chaque agent perçoit la position de la pièce selon l'indice de son coin supérieur gauche. L'espace d'état est donc l'ensemble des positions possibles du coin supérieur gauche de la pièce sur la surface,
- **récompense** : lorsque la pièce atteint la position désirée, tous les agents reçoivent une récompense $r = 10$. Si la pièce se rapproche des bords (colonnes grisées sur la figure B.4), tous les agents sont pénalisés et reçoivent une pénalité $r = -50$. Dans tous les autres cas, la récompense est $r = 0$.

Enjeux face à la coordination

Dans ce système, les agents doivent coordonner leurs actions pour déplacer la pièce à un endroit donné. L'intérêt majeur de ce *benchmark* est d'étudier la coordination d'un grand nombre d'agents. Le système se compose de 270 agents mais dans chaque état ne sont vraiment impliqués que les agents situés en-dessous et à la périphérie de la pièce, c'est-à-dire 16 agents dans notre cas. L'environnement est stochastique étant donnée l'incertitude sur le résultat des actions des agents.

Annexe C

Constantes du modèle dynamique du convoyage 2D

Vélocité latérale du jet d'air en mm/s en fonction de sa distance à l'orifice d en mm :

$$v_{air}(d) = \beta e^{-\frac{d^2}{2\sigma^2}} \tag{C.1}$$

avec $\beta = 5500$ et $\sigma = 1,3$ si $d \geq 0$ et $\beta = -1000$ et $\sigma = 0,2$ si $d < 0$.

Paramètre	Description	Valeur	Unité
m	masse de l'objet	$6,6.10^{-3}$	g
l	épaisseur de l'objet	$2,5.10^{-1}$	mm
C_D	coefficient de traînée	$1,11$	
ρ	densité de l'air	$1,3$	kg/m^3
J	moment d'inertie de l'objet	$0,05$	g/mm^2
η	viscosité de l'air	$1,81.10^{-5}$	kg/m.s
K	coefficient géométrique de la force de viscosité	$2,75$	mm

Bibliographie

[Ado05] Lounis Adouane : *Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un groupe de Robots Mobiles*. Thèse de doctorat, Université de Franche-Comté, 2005. 3, 4

[ALB+ss] M. Ataka, B. Legrand, L. Buchaillot, D. Collard et H. Fujita : Design, fabrication and operation of two dimensional conveyance system with ciliary actuator arrays. *IEEE Journal of Microelectromechanical Systems*, in press. 160

[AMF07] M. Ataka, M. Mita et H. Fujita : Multi-object conveyance by peripherally controlled micro actuator/sensor array. *Transducers*, 2007. 160

[AOTF94] M. Ataka, A. Omodaka, N. Takeshima et H. Fujita : Fabrication and operation of polyimide bimorph actuators for a ciliar motion system. *IEEE/ASME Journal of Micro-Electro-Mechanical Systems*, 2(4):146–150, 1994. 5, 160

[Ark89] Ronald C. Arkin : Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, 1989. 43

[BA94] Tucker Balch et Ronald C. Arkin : Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994. 62

[BAC06] Darrin C. Bentivegna, Christopher G. Atkeson et Gordon Cheng : Learning similar tasks from observation and practice. *In Proceedings of International Conference on Intelligent Robots and Systems (IROS06)*, 2006. 32

[BB05] S. Behnke et M. Bennewitz : Learning to play soccer using imitative reinforcement. *In Proc. of the ICRA Workshop on Social Aspects of Robot Programming through Demonstration*, Barcelona, April 2005. 34

[BBR+00] K.-F. Böhringer, Vivek Bhatt, Bruce R, All Donald et Ken Goldberg : Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26:389–429, 2000. 171

[BBS06a] Lucian Busoniu, Robert Babuska et Bart De Schutter : Decentralized reinforcement learning control of a robotic manipulator. *In Proceedings*

of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006), pages 1347–1352, Singapore, décembre 2006. 98, 99

[BBS06b] Lucian Busoniu, Robert Babuska et Bart De Schutter : Multi-agent reinforcement learning : A survey. *In Int. Conf. Control, Automation, Robotics and Vision*, pages 527–532, December 2006. 81

[BD95] Steven J. Bradtke et Michael O. Duff : Reinforcement learning methods for continuous-time markov decision problems. *In Advances in Neural Information Processing Systems 7*, pages 393–400. MIT Press, 1995. 183

[BDKL00] K.-F. Böhringer, Bruce R. Donald, Lydia Kavraki et Florent Lamiraux : Part orientation with one or two stable equilibria using programmable vector fields. *IEEE Transactions on Robotics and Automation*, 16:157–170, 2000. 171, 174

[BDM96] K.-F. Böhringer, B. R. Donald et N. C. MacDonald : Single-crystal silicon actuator arrays for micro manipulation tasks. *In Proc. of the 9th IEEE Annual International Workshop on Micro Electromechanical Systems*, pages 7–12, 1996. 160, 171

[BDMM94] K.-F. Böhringer, B. R. Donald, R. Mihailovich et N. C. MacDonald : Sensorless manipulation using massively parallel microfabricated actuator arrays. *In Proc. of IEEE ICRA*, pages 826–833, San Diego, CA, May 1994. 160, 171

[Bel57a] Richard Bellman : *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957. 12, 15, 22

[Bel57b] Richard Bellman : A markov decision process. *Journal of Mathematical Mechanics*, 6:679–684, 1957. 12

[Bey06] Aurélie Beynier : *Une contribution à la résolution des Processus Décisionnels de Markov Décentralisés avec contraintes temporelles.* Thèse de doctorat, Université de Caen Basse-Normandie, 2006. 78

[BG88] Alan H. Bond et Les Gasser : *Readings in Distributed Artificial Intelligence*. Morgan Kaufman, 1988. 52

[BGC+98] K.-F. Böhringer, Ken Goldberg, Michael Cohn, Roger Howe et Al Pisano : Parallele microassembly with electrostatic force field. *In Proceedings of IEEE ICRA*, 1998. 159, 160

[BGIZ02] Daniel S. Bernstein, Robert Givan, Neil Immerman et Shlomo Zilberstein : The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002. 64, 76, 78

[BJD86] M. Benda, V. Jagannathan et R. Dodhiawala : On optimal cooperation of knowledge sources - an experimental investigation. Rapport technique BCS-G2010-280, Boeing Advanced Technology Center, Boeing Computing Services, Seattle,Washington, 1986. 146, 150, 204, 206

[BK96] Paul Bakker et Yasuo Kuniyoshi : Robot see, robot do : An overview of robot imitation. *In Workshop on Learning in Robots and Animals (AISB96)*, pages 3–11, Brighton, UK, 1996. 32

[Bon94] Eric Bonabeau : *Intelligence collective*. 1994. 53

[Bou96] Craig Boutilier : Planning, learning and coordination in multiagent decision processes. *In Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996. 72, 93, 104

[Bou99] Craig Boutilier : Sequential optimality and coordination in multiagent systems. *In IJCAI*, pages 478–485, 1999. 72, 145, 203

[Bow03] Michael Bowling : *Multiagent learning in the presence of agents with limitations*. Thèse de doctorat, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 2003. 66, 109

[Bow05] M. Bowling : Convergence and no-regret in multiagent learning. *In* Y. Weiss L. K. Saul et L. Bottou, éditeurs : *Advances in Neural Information Processing Systems*, pages 209–216. MIT Press, 2005. 111

[BP03] Bikramjit Banerjee et Jing Peng : Adaptive policy gradient in multiagent learning. *In AAMAS '03 : Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 686–692, New York, NY, USA, 2003. ACM. 111

[Bro51] G. W. Brown : Iterative solution of games by fictitious play. *In* T. C. Koopmans, éditeur : *Activity Analysis of Production and Allocation*, chapitre XXIV. 1951. 61

[BSA88] Andrew G. Barto, Richard S. Sutton et Charles W. Anderson : Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):835–846, 1988. 14

[Buf03] O. Buffet : *Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs*. Thèse de doctorat, Université Henri Poincaré - Nancy 1, France, 2003. paper. 150, 206

[BV01] Michael Bowling et Manuela Veloso : Convergence of gradient dynamics with a variable learning rate. *In Proc. 18th International Conf. on Machine Learning*, pages 27–34. Morgan Kaufmann, San Francisco, CA, 2001. 111

[BV02] Michael Bowling et Manuela Veloso : Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002. 109, 110, 111, 112, 114, 197

[BZI00] Daniel S. Bernstein, Shlomo Zilberstein et Neil Immerman : The complexity of decentralized control of markov decision processes. *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*, Standford, California, 2000. 76

[CB98a] Caroline Claus et Craig Boutilier : The dynamics of reinforcement learning in cooperative multiagent systems. *In Proceedings of the Fifteenth*

National Conference on Artificial Intelligence, pages 746–752, 1998. viii, 61, 62, 92, 95, 100

[CB98b] Robert H. Crites et Andrew G. Barto : Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998. 98

[Cd75] Chaib-draa : A comparison of natural and artificial intelligence. *SIGART newsletter*, 53:11–13, 1975. 14

[CDB96] Marco Colombetti, Marco Dorigo et Giuseppe Borghi : Robot shaping : The HAMSTER experiment. *In* M.Jamshidi et al., éditeur : *Proceedings of ISRAM'96, Sixth International Symposium on Robotics and Manufacturing*, Montpellier, France, May 28–30 1996. 33

[CES02] Gilles Caprari, T. Estier et R. Siegwart : Fascination of down scaling - alice the sugar cube robot. *Journal of Micro-Mechatronics*, pages 177–189, 2002. 4

[CG05] Jacob W. Crandall et Michael A. Goodrich : Learning to compete, compromise, and cooperate in repeated general-sum games. *In ICML '05 : Proceedings of the 22nd international conference on Machine learning*, pages 161–168, New York, NY, USA, 2005. ACM. 75

[Con99] Frédéric de Coninck : La formation du salarié flexible et la crise des processus sociaux d'apprentissage. Document du LATTS - ENPC, avril 1999. 32

[CZF+07] Y.-A. Chapuis, L. Zhou, Y. Fukuta, Y. Mita et H. Fujita : Fpga-based decentralized control of arrayed mems for microrobotic application. *IEEE Transactions on Industrial Electronics*, 54(4):1926–1936, 2007. 160

[CZFH08] Y.-A. Chapuis, L. Zhou, H. Fujita et Y. Hervé : Multi-domains simulation using vhdl-ams for distributed mems in functionnal environment : Case of a 2-d air-jet micromanipulator. *Sensors and Actuators A : Physical*, 2008. In press, Available online. 165, 167, 172

[Day92] Peter Dayan : The convergence of TD(λ) for general λ. *Machine Learning*, 8:341–362, 1992. 28

[DC94] Marco Dorigo et Marco Colombetti : Robot shaping : developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994. 33

[dCLR06] Enrique Munoz de Cote, Alessandro Lazaric et Marcello Restelli : Learning to cooperate in multi-agent social dilemmas. *In AAMAS '06 : Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 783–785, New York, NY, USA, 2006. ACM. 96

[Die00] Thomas G. Dietterich : Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000. 180

[DMK00] Kevin R. Dixon, Richard J. Malak et Pradeep K. Khosla : Incorporating prior knowledge and previously learned information into reinforcement learning agents. Rapport technique, Institute for Complex Engineered Systems, CMU, 2000. 33

[DS94] Peter Dayan et Terrence J. Sejnowski : TD(λ) converges with probability 1. *Machine Learning*, 14(1):295–301, 1994. 28

[EFC08] Andres El-Fakdi et Marc Carreras : Policy gradient based reinforcement learning for real autonomous underwater cable tracking. *In Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS08)*, 2008. 33

[EHRLR80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser et D. Raj Reddy : The hearsay-ii speech-understanding system : Integrating knowledge to resolve uncertainty. *ACM Comput. Surv.*, 12(2):213–253, 1980. 52

[fBRNM96] Karl friedrich Böhringer, Bruce Randall, Donald Noel et C. Macdonald : What programmable vector fields can (and cannot) do : Force field algorithms for mems and vibratory parts feeders. *In In Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 822–829, 1996. 171, 174

[FCMF06] Y. Fukuta, Y.-A. Chapuis, Y. Mita et H. Fujita : Design, fabrication and control of mems-based actuator arrays for air-flow distributed micromanipulation. *Journal of Micro-Electro-Mechanical Systems*, 2006. vii, 8, 160, 169, 172

[Fer94] Jacques Ferber : La kénétique : des systèmes multi-agents à une science de l'interaction. *Revue internationale de systémique*, 8:13–27, 1994. 54

[Fer95] Jacques Ferber : *Les Syst èmes Multi-Agents : vers une Intelligence Collective*. InterEditions, 1995. 53, 54, 55, 56, 57, 58, 61

[FG96] S. Franklin et A. Graesser : Is it an agent, or just a program ? a taxonomy for autonomous agents. *In Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193, Berlin, Germany, 1996. Springer-Verlag. 54

[Fuj96] H. Fujita : Future of actuators and microsystems. *Sensors and Actuators*, A 56:105–111, 1996. 161

[FV07] Nancy Fulda et Dan Ventura : Predicting and preventing coordination problems in cooperative q-learning systems. *In Proceedings of the International Joint Conference on Artificial Intelligence*, 2007. 91

[Gar04] Pascal Garcia : *Exploration guidée et induction de comportements génériques en apprentissage par renforcement*. Thèse de doctorat, INSA Rennes, 2004. 11, 180

[GH03] Amy Greenwald et Keith Hall : Correlated-q learning. *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 242–249, 2003. 75

[GR06] Thomas Gabel et Martin Riedmiller : Multi-agent case-based reasoning for cooperative reinforcement learners. *In ECCBR*, pages 32–46, 2006. 113

[GZ03] C. Goldman et S. Zilberstein : Optimizing information exchange in cooperative multiagent systems. *In International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 137–144, 2003. 78

[Har68] Garrett Hardin : The tragedy of the commons. *Science*, pages 1243–1248, 1968. 62

[HBZ04] E. Hansen, D. Bernstein et S. Zilberstein : Dynamic programming for partially observable stochastic games. *In In Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 709–715, 2004. 76

[HD94] Gillian Hayes et John Demiris : A robot controller using learning by imitation. *In Proceedings of the 2nd International Symposium on Intelligent Robotic Systems*, Grenoble, France, 1994. 32

[HHK+02] Y. Hirata, T. Hatsukari, K. Kosuge, H. Asama, H. Kaetsu et K. Kawabata : Transportation of an object by multiple distributed robot helpers in cooperation with a human. *Transactions of the Japan Society of Mechanical Engineers*, 68:181–188, 2002. vii, 6

[How60] Ronald A. Howard : *Dynamic Programming and Markov Processes*. The MIT Press, 1960. 12, 21

[HS99] G. Hailu et G. Sommer : On amount and quality of bias in reinforcement learning. *In Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1491–1495, Tokyo, Oct. 1999. 34, 40

[HW03] Junling Hu et Michael P. Wellman : Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003. 75

[JHS08] Nicholas K. Jong, Todd Hester et Peter Stone : The utility of temporal abstraction in reinforcement learning. *In AAMAS '08 : Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 299–306, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems. 180

[JJS94] Tommi Jaakkola, Michael I. Jordan et Satinder P. Singh : On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994. 24

[JRK06] Jelle R. Jelle R. Kok : *Coordination and Learning in Cooperative Multiagent Systems*. Thèse de doctorat, Faculty of Science, University of Amsterdam, 2006. 146, 204

[JSW98] Nicholas R. Jennings, Katia Sycara et Michael Wooldridge : A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998. 52

[KF94] S. Konishi et H. Fujita : A conveyance system using air flow based on the concept of distributed micro motion systems. *Journal of Micro-Electro-Mechanical Systems*, 3(2):54–58, 1994. 160

[KII94] Yasuo Kuniyoshi, Masayuki Inaba et Hirochika Inoue : Learning by watching : Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10: 799–822, 1994. 32

[KK02] Spiros Kapetanakis et Daniel Kudenko : Reinforcement learning of coordination in cooperative multi-agent systems. *In Proceedings of the Nineteenth NCAI*, 2002. 92, 106, 114, 124, 129, 197

[KK04] Spiros Kapetanakis et Daniel Kudenko : Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. *In AAMAS '04 : Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1258–1259, Washington, DC, USA, 2004. IEEE Computer Society. 106, 124

[Kla03] Eric Klavins : Communication complexity of multi-robot systems. *Springer Tracts in Advanced Robotics*, 7:275–292, 2003. 62

[KLC98] Leslie Pack Kaelbling, Michael L. Littman et Anthony R. Cassandra : Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998. 76

[KLM96] Leslie Pack Kaelbling, Michael L. Littman et Andrew W. Moore : Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 11

[Klo72] Harry A. Klopf : Brain function and adaptative systems - a heterostatic theory. Rapport technique AFCRL-72-0164, Air Force Cambridge Research Laboratories, 1972. 14

[Kön03] Ville Könönen : Asymmetric multiagent reinforcement learning. *In Intelligent Agent Technology, IAT2003. IEEE/WIC International Conference on*, pages 336–342, Jan 2003. 75

[KS96] Sven Koenig et Reid G. Simmons : The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22(1-3):227–250, 1996. 34, 35, 38

[KV04] J. Kok et N. Vlassis : Sparse cooperative Q-learning. *In Proc. of 21st International Conference of Machine Learning, Banff, Canada*, 2004. 146, 204

[KWSS01] Peng-Jui Ku, K.Tobias Winther, Harry E. Stephanou et Riko Safaric : Distributed control system for an active surface device. *In Proceedings of ICRA*, May 21-26 2001. 160, 172

[Lau02] Guillaume Laurent : *Synthèse de comportements par apprentissage par renforcement parallèles.* Thèse de doctorat, Université de Franche-Comté, Besançon, 2002. 11, 28, 33

[LC88] Victor R. Lesser et Daniel D. Corkill : The distributed vehicle monitoring testbed : a tool for investigating distributed problem solving networks. pages 69–85, 1988. 52

[LC05] Julien Laumônier et Brahim Chaibdraa : Multiagent q-learning : Preliminary study on dominance between the nash and stackelberg equilibriums. *In AAAI 2005 Workshop on Multiagent Learning*, Pittsburgh, USA, Jul 2005. 75

[Les99] Victor R. Lesser : Cooperative multiagent systems : A personal view of the state of the art. *Knowledge and Data Engineering*, 11(1):133–142, 1999. 51

[Lit01a] Michael Littman : Value-function reinforcement learning in markov games. *Journal of Cognitive Systems Research*, 2:55–66, May 2001. 75

[Lit01b] Michael L. Littman : *Friend-or-Foe Q-learning in General-Sum Games*. Morgan Kaufmann, San Francisco, CA, 2001. 75

[LMC01] Jonathan E. Luntz, William Messner et Howie Choset : Distributed manipulation using discrete actuator arrays. *The International Journal of Robotics Research*, 20:553–583, 2001. 159

[LR00] Martin Lauer et Martin Riedmiller : An algorithm for distributed reinforcement learning in cooperative multi-agent systems. *In Proc. 17th International Conf. on Machine Learning*, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000. 101, 102, 104, 114, 120, 129, 132, 197

[LR04] Martin Lauer et Martin Riedmiller : Reinforcement learning for stochastic cooperative multi-agent systems. *In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS04)*, volume 03, pages 1516–1517, Los Alamitos, CA, USA, 2004. IEEE Computer Society. 112, 113, 114, 197

[Mat94] M. J. Mataric : Reward functions for accelerated learning. *In Proc. of the 11th ICML*, pages 181–189, 1994. 43

[MB99] Nicolas Meuleau et Paul Bourgine : Exploration of multi-state environments : Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999. 26, 142

[MC68] D. Michie et R. Chambers : Boxes : An experiment in adaptative control. *Machine Intelligence 2*, pages 137–152, 1968. 14

[MC92] Sridhar Mahadevan et Jonathan Connell : Automatic programming of behavior-based robots using reinforcement learning. *Artif. Intell.*, 55(2-3):311–365, 1992. viii, 57, 59

[MDP03] Fabien Montagne, Samuel Delepoulle et Philippe Preux : A critic-critic architecture to combine reinforcement and supervised learnings. *In European workshop on reinforcement learning (EWRL03)*, Nancy, 2003. 33

[Mic61] Donald Michie : Trial and error. *Science Survey, Part 2*, pages 129–145, 1961. 14

[ML04] Hyungpil Moon et Jonathan E. Luntz : Toward sensorless manipulation using airflow. *In proceedings of IEEE ICRA*, pages 1574–1579. IEEE, 2004. 172

[MLFP06] Laetitia Matignon, Guillaume J. Laurent et Nadine Le Fort-Piat : Improving reinforcement learning speed for robot control. *In Proceedings of IROS*, pages 3172–3177, 2006. 49

[MLR06] Jose Enrique Munoz, Alessandro Lazaric et Marcello Restelli : Learning to cooperate in multi-agent social dilemmas. *In AAMAS*, pages 783–785, 2006. 111

[MRIB04] David L. Moreno, Carlos V. Regueiro, Roberto Iglesias et Senén Barro : Using prior knowledge to improve reinforcement learning in mobile robotics. *In Towards Autonomous Robotic Systems (TAROS'04)*, Colchester (UK), 2004. 33

[MS98] Amy McGovern et Richard S. Sutton : Macro-actions in reinforcement learning : an empirical analysis. Rapport technique 98-70, University of Massachusetts, 1998. 180

[MSF97] Amy McGovern, Richard S. Sutton et Andrew H. Fagg : Roles of macro-actions in accelerating reinforcement learning. *In Proceedings of the 1997 Grace Hopper Celebration of Women in Computing*, 1997. 180

[Nas50] John F. Nash : Equilibrium points in n-person games. *In Proceedings of the National Academy of Sciences of the United States of America*, volume 36, pages 48–49, 1950. 70

[Nas51] John F. Nash : Non-cooperative games. *Annals of Mathematics*, 54:286–295, 1951. 70

[NCD+04] Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger et Eric Liang : Autonomous inverted helicopter flight via reinforcement learning. *In International Symposium on Experimental Robotics*, 2004. 33

[NHR99] Andrew Y. Ng, Daishi Harada et Stuart Russell : Policy invariance under reward transformations : theory and application to reward shaping. *In Proceedings of the 16th ICML*, pages 278–287, 1999. 43, 47

[NM44] John Von Neumann et Oskar Morgenstern : *Theory of Games and Economic Behavior*. Princeton University Press, 1944. 66

[NWM99] H. Nakazawa, Y. Watamasa et O. Morita : Electromagnetic micro-parts conveyer with coil-diode modules. *In Proc. of the IEEE International Conference of Solid-State Sensors and Actuators (Transducers '99)*, pages 1192–1195, 1999. 160

[OR94] M. J. Osborne et A. Rubinstein : *A course in game theory*. MIT Press, 1994. 66, 70

[Par98] Ronald Edward Parr : *Hierarchical control and learning for markov decision processes*. Thèse de doctorat, University of California Berkeley, 1998. Chair-Stuart Russell. 180, 182, 183, 184

[Pav27] Ivan Pavlov : *Conditionning reflexes*. Oxford University Press, Oxford, UK, 1927. 13

[PDM08] PDMIA : *PDMIA*. PDMIA, 2008. 66

[PFH90] K. S. J. Pister, R. Fearing et R. Howe : A planar air levitated electrostatic actuator system. *In Proc. of the IEEE Workshop on Micro Electro Mechanical Systems (MEMS)*, pages 67–71, Napa Valley, California, Feb. 1990. 160

[PH96] Simon Perkins et Gillian Hayes : Robot shaping - principles, methods and architectures. *In Workshop on Learning in Robots and Animals (AISB96)*, 1996. 33

[PHLR08] Mickaël Paris, Yassine Haddab, Philippe Lutz et Patrick Rougeot : Practical characterisation of the friction force for the positioning and orientation of micro-components. *In Proceedings of IROS*, 2008. 159

[PKMK00] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau et Leslie P. Kaelbling : Learning to cooperate via policy search. *In Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314, San Francisco, CA, 2000. Morgan Kaufmann. 72, 77

[PL05] Liviu Panait et Sean Luke : Cooperative multi-agent learning : The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005. 51

[Pre00] Doina Precup : *Temporal abstraction in reinforcement learning*. Thèse de doctorat, University of Massachusetts, 2000. Director-Richard S. Sutton. 180, 181

[PS97] Doina Precup et Richard S. Sutton : Multi-time models for reinforcement learning. *In Proceedings of the ICML'97 Workshop on Modelling in Reinforcement Learning*, 1997. 183

[PSL06] Liviu Panait, Keith Sullivan et Sean Luke : Lenient learners in cooperative multiagent systems. *In AAMAS '06 : Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 801–803, New York, NY, USA, 2006. ACM Press. 108, 109, 114, 132, 197

[PT87] C.H. Papadimitriou et J.N. Tsitsiklis : The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. 21, 78

[PT02] David V. Pynadath et Milind Tambe : The communicative multiagent team decision problem : Analyzing teamwork theories and models. *Journal of AI research*, 16:389–423, 2002. 60

[Put94] M. L. Puterman : *Markov Decision Problems*. 1994. 180

[Qui06] Joel Quinqueton : Emergence in problem solving, classification and machine learning. *In SYNASC '06 : Proceedings of the Eighth International*

Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pages 5–9, Washington, DC, USA, 2006. IEEE Computer Society. 55

[RA98] J. Randlov et P. Alstrom : Learning to drive a bicycle using reinforcement learning and shaping. *In Proceedings of the 16th ICML*, pages 463–471, 1998. 49

[Ren02] Jean-Philippe Rennard : *Vie artificielle : Où la biologie rencontre l'informatique*. 2002. 55

[RN94] G. A. Rummery et M. Niranjan : On-line Q-learning using connectionist systems. Rapport technique CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994. 26, 28

[RN03] Stuart Russell et Peter Norvig : *Artificial Intelligence : A Modern Approach*. Prentice Hall, second édition, 2003. 54

[RP03] Bohdana Ratitch et Doina Precup : Using mdp characteristics to guide exploration in reinforcement learning. *In ECML*, pages 313–324, 2003. 26

[Sam59] Arthur Samuel : Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:221–229, 1959. 14

[SB98] Richard S. Sutton et Andrew G. Barto : *Reinforcement Learning : An Introduction*. The MIT Press, Cambridge, 1998. 11, 15, 26, 34, 81, 143

[Sch97] Stefan Schaal : Learning from demonstration. *In* Michael C. Mozer, Michael I. Jordan et Thomas Petsche, éditeurs : *Advances in Neural Information Processing Systems*, volume 9, page 1040. The MIT Press, 1997. 32

[SG03] J. L. Stimpson et M. A. Goodrich : Learning to cooperate in a social dilemma : A satisficing approach to bargaining. *In ICML*, 2003. 199

[Sha53] Lloyd S. Shapley : Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–1100, 1953. 72

[SJLS00] Satinder P. Singh, Tommi Jaakkola, Michael L. Littman et Csaba Szepesvari : Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000. 26, 95, 99

[SK02] W. Smart et L. Kaelbling : Effective reinforcement learning for mobile robots. *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA02)*, pages 3404–3410, Piscataway, NJ., 2002. 32

[Ski38] Buhrrus F. Skinner : *The Behavior of Organisms*. Appleton-Century Crofts, Inc., New-York, 1938. 13

[SKM00] Satinder P. Singh, Michael J. Kearns et Yishay Mansour : Nash convergence of gradient dynamics in general-sum games. *In UAI '00 : Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 109

[SPBL06] Keith Sullivan, Liviu Panait, Gabriel Balan et Sean Luke : Can good learners always compensate for poor learners ? *In AAMAS '06 : Proceedings of the fifth international joint conference on Autonomous agents and*

multiagent systems, pages 804–806, New York, NY, USA, 2006. ACM Press. 108, 109, 132

[SPG04] Yoav Shoham, Rob Powers et Trond Grenager : Multi-agent reinforcement learning : a critical survey. *In Proceedings of the AAAI Fall Symposium on Artificial Multi-Agent Learning*, 2004. 75, 81

[SPS99] Richard S. Sutton, Doina Precup et Satinder Singh : Between mdps and semi-mdps : Learning, planning, and representing knowledge at multiple temporal scales. *Artificial Intelligence*, 112:181–211, 1999. 180, 181

[SR02] Ralf Schoknecht et Martin Riedmiller : Speeding-up reinforcement learning with multi-step actions. *In Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2002. 180

[SR03] Ralf Schoknecht et Martin Riedmiller : Reinforcement learning on explicitly specified time scales. *Neural Computing & Applications*, 23:61–80, 2003. 180

[SRI03] Ralf Schoknecht, Martin Riedmiller et Lehrstuhl Informatik : Learning to control at multiple time scales. *In Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Joint International Conference ICANN/ICONIP 2003*. Springer, 2003. 180

[SS73] R.D. Smallwood et E.J. Sondik : The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973. 76

[SS96] Richard S. Sutton et Satinder P. Singh : Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:213–158, 1996. 27

[SS98] Sandip Sen et Mahendra Sekaran : Individual learning of coordination knowledge. *JETAI*, 10(3):333–356, 1998. 99

[SSH94] Sandip Sen, Mahendra Sekaran et John Hale : Learning to coordinate without sharing information. *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994. 97, 99

[SST95] Andrea Schaerf, Yoav Shoham et Moshe Tennenholtz : Adaptive load balancing : A study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2:475–500, 1995. 62

[ST92] Yoav Shoham et Moshe Tennenholtz : On the synthesis of useful social laws for artificial agent societies. *In In Proceedings of AAAI-92*, pages 276–281, 1992. 104

[STM05] S. Syafiie, F. Tadeo et E. Martinez : Model-free intelligent control using reinforcement learning and temporal abstraction-applied to ph control. *In Proceedings of IFAC World Congress*, 2005. 180

[Sut88] Richard S. Sutton : Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988. 14, 23, 27

[SV00] Peter Stone et Manuela M. Veloso : Multiagent systems : A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000. 51

[TA07] Kagan Tumer et Adrian Agogino : Distributed agent-based air traffic flow management. *In AAMAS '07 : Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM. 98

[Tan93] Ming Tan : Multiagent reinforcement learning : Independent vs. cooperative agents. *In 10th International Conference on Machine Learning*, pages 330–337, 1993. 62, 97, 98

[Tho11] Edward L. Thorndike : *Animal Intelligence*. Hafner, Darien, CT, 1911. 13

[Thr92] Sebastian B. Thrun : Efficient exploration in reinforcement learning. Rapport technique CMU-CS-92-102, School of Computer Science, Carnegie Mellon University, 1992. 26

[Tur50] Alan M. Turing : Computing machinery and intelligence. *Mind*, 59:433–460, 1950. 14, 52

[TW00] Kagan Tumer et David Wolpert : Collective intelligence and braess' paradox. *In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 104–109. AAAI Press / The MIT Press, 2000. 63

[VD98] José M. Vidal et Edmund H. Durfee : Learning nested models in an information economy. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):291–308, 1998. 61

[Vla03] N. Vlassis : A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, 2003. 51, 90

[VNPT07] Katja Verbeeck, Ann Nowé, Johan Parent et Karl Tuyls : Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, 2007. 98

[Wat89] Christopher J.C.H. Watkins : *Learning from Delayed Rewards*. Thèse de doctorat, Cambridge University, Cambridge, England, 1989. 14, 23, 28, 178

[WD92] Christopher J.C.H. Watkins et Peter Dayan : Technical note : Q-learning. *Machine Learning*, 8:279–292, 1992. 114, 197

[WdS06] Ying Wang et Clarence W. de Silva : Multi-robot box-pushing : Single-agent q-learning vs. team q-learning. *In Proc. of IROS*, pages 3694–3699, 2006. 99

[Wei99] Gerhard Weiss : *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999. 53

[Wie03] E. Wiewiora : Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003. 49

[WJ95] Michael Wooldridge et Nicholas R. Jennings : Intelligent agents : Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995. 54

[Wol02] Stephen Wolfram : *A New Kind of Science*. Wolfram Media, 2002. vii, 3

[Woo02] Michael Wooldridge : *An Introduction to MultiAgent Systems*. John Wiley and Sons, Mars 2002. 54

[WST01] D.H. Wolpert, J. Sill et K. Tumer : Reinforcement learning in distributed domains : Beyond team games. *In Proceedings of the Seventeenth IJCAI*, 2001. 98

[YFF06] Y. Mita Y. Fukuta, Y.-A. Chapuis et H. Fujita : Design, fabrication and control of mems-based actuators arrays for air-flow distributed micromanipulation. *Journal of Microelectromechanical Systems*, 2006. 162, 163

[YG04] Erfu Yang et Dongbing Gu : Multiagent reinforcement learning for multi-robot systems : A survey. Rapport technique, Department of Computer Science, University of Essex, 2004. 81

[Zho07] Lingfei Zhou : *Modélisation VHDL-AMS multi-domaines de structures intelligentes, autonomes et distribuées à base de MEMS*. Thèse de doctorat, Université Louis Pasteur, 2007. 165, 167, 172

Publications personnelles

Revues internationales à comité de lecture

L. Matignon, G. J. Laurent et N. Le Fort-Piat, "SOoN : an algorithm for the coordination of independent learners in cooperative Markov games", *Autonomous Agents and Multi-Agent Systems*, en révision.

L. Matignon, G. J. Laurent, N. Le Fort-Piat, Y.-A. Chapuis et H. Fujita, "Learning decentralized control of contact-free MEMS-arrayed micromanipulator", *IEEE/ASME Transactions on Mechatronics, Focused Section on Mechatronics for MEMS and NEMS*, soumis.

Communications internationales à comité de lecture

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Toward distributed control of arrayed MEMS air-jet micromanipulator : a reinforcement learning approach", *Proceedings of International Conference on Robotics and Automation (ICRA09)*, soumis.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "A Study of FMQ Heuristic in Cooperative Multi-Agent Games", *Proceedings of International Workshop Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domains (MSDM08)*, en association avec *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, p. 77-91, 2008, Portugal.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Hysteretic Q-Learning : an algorithm for Decentralized Reinforcement Learning in Cooperative Multi-Agent Teams", *Proceedings of the International Conference on Intelligent Robots and Systems (IROS07)*, p. 64-69, 2007, San Diego, CA, USA.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Improving Reinforcement Learning Speed for Robot Control", *Proceedings of the International Conference on In-*

telligent Robots and Systems (IROS06), p. 3172-3177, 2006, Chine.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Reward Function and Initial Values : Better Choices for Accelerated Goal-Directed Reinforcement Learning", *Lecture Notes in Computer Science*, vol. 4131, p. 840-849, 2006.

Communications nationales avec actes

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Un algorithme décentralisé d'apprentissage par renforcement multi-agents coopératifs : le Q-Learning Hystérétique", *2ème Journées Francophones Planification, Décision, Apprentissage pour la conduite de Système (JFPDA)*, p. 115-121, 2007.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Choix de la fonction de renforcement et des valeurs initiales pour accélérer l'Apprentissage par Renforcement dans le cadre de problèmes de plus court chemin stochastique", *1ère Journées Francophones Planification, Décision, Apprentissage pour la conduite de Système (JFPDA)*, 2006.

Communications nationales sans actes

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Contrôle distribué d'une Smart Surface par Apprentissage par Renforcement", *Journées Nationales de la Recherche en Robotique (JNRR)*, 9-12 Octobre 2007, Obernai, France.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Contrôle distribué d'une Smart Surface par Apprentissage par Renforcement", *Ateliers du Laboratoire Européen Associé en Microtechnique*, 10-11 septembre 2007, Arc-et-Senans, France.

L. Matignon, G. J. Laurent et N. Le Fort - Piat, "Injection de connaissances en apprentissage par renforcement", *Ateliers du Laboratoire Européen Associé en Microtechnique*, 20-21 septembre 2005, Arc-et-Senans, France.

Index

Q

S

T

V

W

Résumé

De nombreuses applications peuvent être formulées en termes de systèmes distribués que ce soit une nécessité face à une distribution physique des entités (réseaux, robotique mobile) ou un moyen adopté face à la complexité d'appréhender un problème de manière globale. A travers l'utilisation conjointe de méthodes dites d'apprentissage par renforcement et des systèmes multi-agents, des agents autonomes coopératifs peuvent apprendre à résoudre de manière décentralisée des problèmes complexes en s'adaptant à ceux-ci afin de réaliser un objectif commun. Les méthodes d'apprentissage par renforcement ne nécessitent aucune connaissance *a priori* sur la dynamique du système, celui-ci pouvant être stochastique et non-linéaire. Cependant, afin d'améliorer la vitesse d'apprentissage, des méthodes d'injection de connaissances pour les problèmes de plus court chemin stochastique sont étudiées et une fonction d'influence générique est proposée. Nous nous intéressons ensuite au cas d'agents indépendants situés dans des jeux de Markov d'équipe. Dans ce cadre, les agents apprenant par renforcement doivent surmonter plusieurs enjeux tels que la coordination ou l'impact de l'exploration. L'étude de ces enjeux permet tout d'abord de synthétiser les caractéristiques des méthodes décentralisées d'apprentissage par renforcement existantes. Ensuite, au vu des difficultés rencontrées par ces approches, deux algorithmes sont proposés. Le premier est le Q-learning hystérétique qui repose sur des agents « à tendance optimiste réglable ». Le second est le *Swing between Optimistic or Neutral* (SOoN) qui permet à des agents indépendants de s'adapter automatiquement à la stochasticité de l'environnement. Les expérimentations sur divers jeux de Markov d'équipe montrent notamment que le SOoN surmonte les principaux facteurs de non-coordination et est robuste face à l'exploration des autres agents. Une extension de ces travaux à la commande décentralisée d'un système distribué de micromanipulation (*smart surface*) dans un cas partiellement observable est enfin exposée.

Mots-clés : systèmes multiagents, agents indépendants, agents adaptatifs, agents coopératifs, jeux de Markov d'équipe, apprentissage par renforcement, commande décentralisée, système distribué de micromanipulation, smart surface.

Abstract

Numerous applications can be formulated in terms of distributed systems, be it a necessity face to a physical distribution of entities (networks, mobile robotics) or a means of confronting the complexity to solve globally a problem. The objective is to use together reinforcement learning methods and multi-agent systems. Thus, cooperative and autonomous agents can learn to resolve in a decentralized way complex problems by adapting to them so as to realize a joint objective. Reinforcement learning methods do not need any *a priori* knowledge about the dynamics of the system, which can be stochastic and nonlinear. In order to improve the learning speed, knowledge incorporation methods are studied within the context of goal-directed tasks. A generic goal bias function is also proposed. Then we took an interest in independent learners in team Markov games. In this framework, agents learning by reinforcement must overcome several difficulties as the coordination or the impact of the exploration. The study of these issues allows first to synthesize the characteristics of existing reinforcement learning decentralized methods. Then, given the difficulties encountered by this approach, two algorithms are proposed. The first one, called hysteretic Q-learning, is based on agents with "adjustable optimistic tendency". The second one is the Swing between Optimistic or Neutral (SOoN) in which independent agents can adapt automatically to the environment stochasticity. Experimentations on various team Markov games notably show that SOoN overcomes the main factors of non-coordination and is robust face to the exploration of the other agents. An extension of these works to the decentralized control of a distributed micromanipulation system (smart surface) in a partially observable case is finally proposed.

Keywords : multi-agent systems, independent learners, adaptative agents, cooperative agents, team Markov games, reinforcement learning, decentralized control, distributed micromanipulation systems, smart surface.