Université de Valenciennes et du Hainaut-Cambrésis

Thèse de doctorat

Pour obtenir le grade de Docteur de l'Université de

VALENCIENNES ET DU HAINAUT-CAMBRESIS

Discipline, spécialité selon la liste des spécialités pour lesquelles l'Ecole Doctorale est accréditée :
INFORMATIQUE

Présentée et soutenue par Mohamed Amine HAMILA

Le 03/04/2012, à VALENCIENNES

Ecole doctorale :
Sciences Pour l'Ingénieur (SPI)

Equipe de recherche, Laboratoire :
Laboratoire d'Automatique, de Mécanique et d'Informatique, Industrielles et Humaines (LAMIH)

Planification multi-agents dans un cadre markovien : les jeux stochastiques à somme générale

JURY

Président du jury

- Pierre Marquis, Professeur des Universités, Université d'Artois.

Rapporteurs

- Amal El Fallah-Seghrouchni, Professeur des Universités, Université Paris VI.
- François Charpillet, Directeur de Recherche, LORIA-INRIA Nancy.

Examinateurs

- Pierre Marquis, Professeur des Universités, Université d'Artois.
- Rachid Alami, Directeur de Recherche CNRS, LAAS – Toulouse.
- Abderrafiâa Koukam, Professeur des Universités, Université de Belfort.

Directeur de thèse

- René Mandiau, Professeur des Universités, Université de Valenciennes et du Hainaut-Cambrésis.

Co-directeur de thèse

- Abdel-Illah Mouaddib, Professeur des Universités, Université de Caen Basse Normandie.

Co-encadrant

- Emmanuelle Grislin-Le Strugeon, Maître de conférences HDR, Université de Valenciennes et du Hainaut-Cambrésis.

Remerciements

La rédaction de ce manuscrit de thèse est l'aboutissement d'une réflexion scientifique, d'un long parcours, semé d'embuches, ponctué de moment de joie, de doutes et d'expériences enrichissantes. Bien évidemment, cette belle aventure n'a pas été vécue en solitaire et il m'est un grand plaisir de saisir cette occasion pour remercier tous ceux qui m'ont, d'une manière ou d'une autre, contribué à l'accomplissement de ce travail. Certains pour leurs connaissances scientifiques et d'autres pour leur soutien pendant les moments pénibles.

Comment commencer ces remerciements par une autre personne que mon directeur de thèse, donc je souhaite remercier en premier lieu M. René Mandiau, Professeur des Universités pour m'avoir accueilli au sein de son équipe. Je lui suis également reconnaissant pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma profonde gratitude pour tout cela.

Des remerciements tous particuliers reviennent également à mon co-directeur de thèse, M. Abdel-Illah Mouaddib, Professeur des Universités, pour son aide précieuse. Il m'a permis de m'orienter, de développer et de mettre au point mes idées et d'améliorer ainsi la qualité de mon travail.

J'adresse également mes remerciements les plus chaleureux à ma co-encadrante de thèse, Mme Emmanuelle Grislin-Le Strugeon, Maître de conférence, pour la confiance qu'elle m'a témoignée, pour tous les précieux conseils qu'elle m'a donnés et sans qui ce travail n'aurait pas vu le jour.

Je remercie également mes deux rapporteurs, Mme Amal El Fallah-Seghrouchni et M. François Charpillet pour la lecture qu'ils ont fait de ce manuscrit et les remarques positives et critiques qui m'ont permis de prendre du recul sur mon travail. Je tiens aussi à remercier mes examinateurs, M. Pierre Marquis, M. Rachid Alami (par skype) et M. Abderrafiâa Koukam, qui ont bien voulu examiner ce manuscrit.

Je remercie tous les chercheurs et personnels de l'université de Valenciennes pour leur grande gentillesse et leur contribution à l'élaboration de ce travail. En écrivant ces remerciements, je pense évidement à mes compagnons d'infortune (ceux que je ne citerais pas et qui se reconnaîtront) de la salle des thésards qui fut autant un lieu de vie qu'un lieu de travail durant ces quelques années passées au laboratoire.

Enfin, au terme de ces remerciements, je souhaiterai rendre hommage à tous mes proches et à ma famille, pour leur soutien, leurs encouragements et leur appui tout au long de mes études.

Le futur ne dépend que du présent.
A. Markov

Table des matières

Table des figures **9**

Liste des tableaux **13**

Liste des Algorithmes **15**

Introduction générale

Chapitre 1
Etat de l'art

1.1 De l'Agent au Système Multi-Agents 7
1.2 Les processus décisionnels de Markov 9
1.2.1 Les processus stochastiques et les chaînes de Markov 10
1.2.2 Formalisme 10
1.2.3 Algorithmes de résolution 15
1.2.4 Exemple d'utilisation d'un MDP 19
1.3 Les jeux matriciels 20
1.3.1 Formalisme 21
1.3.2 Classification des jeux 22
1.3.3 Concepts de solution 24
1.4 Les jeux stochastiques 28
1.4.1 Formalisme 28
1.4.2 Extensions multi-agents 29
1.4.3 Exemples 31
1.4.4 Concepts de solution 33
1.5 Conclusion 35

Chapitre 2
Algorithmes de résolution pour les jeux stochastiques

2.1 Algorithme de Shapley 40
2.2 Algorithme de Kearns et al. 42
2.3 Autres travaux 45
2.3.1 L'algorithme InfiniteVI de Kearns et al. 46
2.3.2 L'algorithme FiniteVICOM de Burkov et al. 48
2.3.3 Algorithme de Pollatschek et Avi-Itzhak 50
2.3.4 Algorithmes issus des techniques d'apprentissage 50
2.4 Implémentation de l'équilibre de Nash 54
2.5 Limitations 56
2.6 Conclusion 57

Chapitre 3
Proposition d'un algorithme de planification pour les jeux stochastiques

3.1 Proposition d'un algorithme de planification décentralisé pour les jeux stochastiques 61
3.1.1 Calcul des équilibres 62
3.1.2 Sélection d'un équilibre 68
3.1.3 Algorithme pour la résolution des jeux stochastiques 73
3.1.4 Application numérique 77
3.2 Amélioration de l'algorithme 78
3.2.1 Principe de la dominance stratégique 78
3.2.2 Extension de l'algorithme SGInfiniteVI 81
3.3 Conclusion 85

Chapitre 4
Résultats expérimentaux

4.1 Un jeu de grille 89
4.1.1 Description 90
4.1.2 La modélisation sous forme d'un jeu stochastique 91
4.2 Evaluation de la version initiale de l'algorithme 92
4.2.1 Principe de fonctionnement 92
4.2.2 Evaluation des fonctions de sélection d'équilibre 93
4.2.3 Expériences complémentaires 105

4.3 Evaluation de l'algorithme avec élimination des stratégies dominées 109
4.3.1 Evaluation quantitative 110
4.3.2 Evaluation qualitative 114
4.4 Conclusion 117

Conclusion et perspectives

Bibliographie **123**

Table des figures

1.1 Composition d'un agent : Perception, Décision, Action. 7
1.2 Un système multi-agents. 8
1.3 Représentation d'un exemple de matrice de transition. 12
1.4 Application des MDPs. 20
1.5 Un exemple de jeu matriciel. 22
1.6 Le jeu du pile ou face (*Matching Pennies*). 22
1.7 Le jeu du dilemme du prisonnier. 23
1.8 La guerre des sexes. 26
1.9 Un jeu stochastique à deux joueurs : exemple de transition possible entre deux étapes, avec deux actions jointes a_0 et a_1 ; chaque état peut être vu comme un jeu matriciel définissant les récompenses de chaque agent. 29
1.10 Le jeu *Grid Soccer* de Littman. 32
1.11 Le jeu *Grid World* de Hu et Wellman. 32
1.12 Equilibre de Nash trouvé pour le jeu *Grid World*. 34

2.1 La matrice de paiement $M(s)$ du premier agent, relative à l'état s et construite à partir des Q-valeurs individuelles. 42
2.2 l'ensemble des équilibres de Nash du jeu *Grid-World* de Hu et Wellman, calculés par l'algorithme FiniteVI. 46
2.3 Un jeu matriciel à somme générale. Le couple de valeurs $(1, 10)$ est trouvé en calculant un équilibre de Nash, tandis que le couple $(10, 1)$ est trouvé en appliquant la technique de *Maximin*. 47

3.1 Technique de calcul d'un équilibre de Nash par énumération des meilleures réponses. 63
3.2 Matrice ne comportant pas d'équilibre. 65
3.3 Recherche d'équilibre de Nash approximatifs avec la méthode de meilleures réponses. 66
3.4 Equilibres de Nash déterminés avec les différentes fonctions de sélection. . . 72

3.5 Equilibres de Nash déterminés avec les différentes fonctions de sélection. . . 73
3.6 Exemple de jeu stochastique. 77
3.7 Ordre d'élimination des stratégies dominées. 80
3.8 L'élimination des stratégies faiblement dominées peut entraîner l'omission d'autres équilibres présents dans le jeu. 80
3.9 Equilibre de Nash calculé par élimination des stratégies dominées. 82
3.10 Equilibre de Nash calculé par (1) élimination des stratégies dominées et (2) par énumération des meilleures réponses. 83

4.1 Le problème de déplacement d'objets par des agents. 89
4.2 Un exemple de scénario : deux agents, quatre objets et une boîte de rangement. 90
4.3 Evolution de l'espace d'états en fonction de la taille de la grille (scénario avec 3 agents et 3 objets). 92
4.4 Déroulement d'une mission de déplacement d'objets : 2 agents et 2 objets (grille toroïdale). 94
4.5 Situation conflictuelle induite par les déplacements des agents. 96
4.6 Situation de livelock induite par les déplacements des agents. 97
4.7 Situation de deadlock induite par les déplacements des agents. 97
4.8 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de conflits, avec des scénarios de deux agents et quatre objets. 98
4.9 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de livelocks, avec des scénarios de deux agents et quatre objets. 99
4.10 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de deadlocks, avec des scénarios de deux agents et quatre objets. 100
4.11 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de conflits, avec des scénarios de deux agents et quatre objets. 103
4.12 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de livelocks, avec des scénarios de deux agents et quatre objets. 104
4.13 Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de deadlocks, avec des scénarios de deux agents et quatre objets. 104

4.14 Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen d'étapes et avec des scénarios de deux agents et quatre objets. 107
4.15 Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen d'étapes et avec des scénarios de deux agents et quatre objets. 108
4.16 Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen de conflits et avec des scénarios de deux agents et quatre objets. 109
4.17 Comparaison entre l'élimination des stratégies strictement et faiblement dominées (avec des scénarios de deux agents et quatre objets). 110
4.18 Comparaison entre l'élimination des stratégies strictement/faiblement dominées et la version initiale de l'algorithme suivant le temps de calcul (avec des scénarios de deux agents et quatre objets). 111
4.19 Construction et recherche d'équilibre avec l'algorithme EISD. 113
4.20 Construction partielle de la matrice et recherche d'équilibre. 113
4.21 L'espace mémoire nécessaire pour calculer la politique d'actions (scénario avec 3 agents et 3 objets). 115
4.22 Evaluation du nombre moyen de conflits avec et sans élimination des stratégies dominées. 116
4.23 Evaluation du nombre moyen de deadlocks, avec ou sans élimination des stratégies dominées. 117

Liste des tableaux

4.1 Le pourcentage d'équilibres de Nash différents, suivant les trois fonctions de sélection d'équilibre, différentes tailles de la grille et quatre objets transportés. 95
4.2 Evaluation du non respect de l'équilibre de Nash en fonction du nombre d'étapes. 102
4.3 Evolution du nombre d'états et du temps de calcul de la solution en fonction du nombre d'agents (le nombre d'objets est fixé à 4). 105
4.4 Mémoire utilisée (Mo) et temps de calcul (sec) pour différents nombres d'objets, différentes tailles de la grille et 3 agents. 106
4.5 Gain en mémoire suivant le nombre d'agents. 114

Liste des Algorithmes

1 **Algorithme Value-Iteration** 16
2 **Policy-Iteration** 18

3 **Algorithme de Shapley pour la résolution des jeux stochastiques à somme nulle, escomptés et à deux joueurs.** 41
4 **Algorithme FiniteVI pour la résolution des jeux stochastiques à somme générale, non pondérés, à horizon fini et à deux joueurs.** . 44
5 **Algorithme FiniteVICOM pour la résolution des jeux stochastiques à somme générale, à horizon fini et avec communication.** . . 49
6 **Algorithme générique pour la résolution par apprentissage des jeux stochastiques.** 52
7 **Algorithme du jeu fictif pour les jeux stochastiques.** 53

8 **EMR (Enumération des meilleures réponses).** 64
9 **EMR-epsilon (à epsilon près).** 67
10 **EMR (adaptation au cas de n agents).** 69
11 **SGInfiniteVI pour la résolution des jeux stochastiques à somme générale et à plusieurs agents.** 74
12 **Elimination Itérative des Stratégies Dominées (EISD)** 81
13 **SGInfiniteVI pour la résolution des jeux stochastiques à somme générale et à plusieurs agents, avec élimination des stratégies dominées.** 84

Introduction générale

Un agent intelligent est la combinaison de trois fonctions : perception, raisonnement et action. Il reçoit des observations sur l'état de l'environnement et choisit une action parmi celles qui lui sont disponibles. La composante de raisonnement est définie à partir des observations, de l'état de l'agent et de son but. Un but peut être par exemple, un état de l'environnement à atteindre ou un gain à maximiser. Le comportement optimal d'un agent se définit comme une séquence d'actions, lui permettant d'optimiser ses gains en tenant compte de l'incertitude. Il s'agit donc d'un problème de planification sous incertitude. Les Processus Décisionnels de Markov proposent un formalisme mathématique particulièrement adapté à la résolution de tels problèmes décisionnels mono-agent sous incertitude.

Quand un environnement contient plus d'un agent, on peut déjà parler d'un système multi-agents. Prendre les "bonnes" décisions dans des environnements multi-agents est considéré comme un problème difficile. En effet, la présence de plusieurs agents peut impliquer une multiplicité de décisions, des conflits d'intérêts et un manque de coordination. De plus, si les décideurs interagissent en permanence à travers le temps, ils doivent alors décider non seulement ce qu'il faut faire dans le présent, mais aussi (et cela est souvent plus important) décider comment leurs décisions présentes peuvent affecter le comportement des autres à l'avenir. Certains modèles de la théorie des jeux, tels que les jeux matriciels, sont particulièrement adaptés pour étudier les aspects à long terme des interactions multi-agents.

Les actions de l'agent affectent, non seulement le comportement futur des autres agents, mais aussi l'environnement, qui peut évoluer d'un état à un autre. Le modèle des jeux stochastiques reflète cet aspect en supposant qu'un jeu peut avoir de multiples états et des transitions stochastiques, contrôlées conjointement par l'ensemble des agents.

Problématique

L'objectif de cette thèse est de trouver par des moyens algorithmiques la réponse à certaines questions qui sont habituellement soulevées, lorsque l'on traite un problème de décision présenté comme un système multi-agents. Par l'expression "moyens algorithmiques", nous entendons la façon avec laquelle nous répondons aux problèmes que nous traitons dans cette thèse. Par l'expression "certaines questions" nous voulons dire que nous n'avons pas l'intention de répondre à toutes les questions qui peuvent surgir d'un système multi-agents. Plus précisément, étant donné un jeu stochastique, nous nous sommes intéressés à deux types de questions :

- Comment les agents peuvent-ils interagir les uns avec les autres et coordonner leurs décisions ? Par exemple, lorsque plusieurs agents (ne pouvant pas communiquer) sont mis dans un environnement et abandonnés à leur sort, comment un agent devrait-il raisonner pour prévoir ses actions ainsi que celles des autres ?
- Comment les agents peuvent-ils calculer une politique jointe d'une manière distribuée ? Par exemple, lorsque toutes les propriétés de l'environnement sont connues, comment peuvent-ils déterminer, de manière distribuée, un ensemble de plans d'actions individuels, de manière à ce que leur comportement global soit cohérent ?

Ce travail s'intéresse donc au domaine de la planification en utilisant le modèle des jeux stochastiques. L'objectif est de permettre à plusieurs agents de coordonner leurs actions de manière décentralisée. Les approches existantes restent souvent impraticables à cause de plusieurs restrictions. Ainsi, dans cette thèse nous relaxons certaines hypothèses, cela ne conduit pas nécessairement à rechercher des comportements optimaux, souvent trop coûteux, mais plutôt à avoir un comportement global satisfaisant.

Plan synthétique de la thèse

Ce manuscrit se décompose en quatre chapitres. Le premier chapitre réalise un état de l'art des différents outils utilisés pour la suite de nos travaux. Le second chapitre introduit les travaux existants pour la résolution des jeux stochastiques. Le troisième chapitre présente nos contributions. Enfin, le quatrième chapitre présente les différentes expérimentations menées en vue de valider nos travaux.

Chapitre 1 : Etat de l'art

En nous appuyant sur une étude bibliographique, nous introduirons les différents concepts et notions utilisés dans nos travaux. Dans ce chapitre, nous présentons le cadre

des jeux stochastiques comme une description d'un système multi-agents. Cela nous amènera à présenter tout d'abord le concept d'agent, pour ensuite discuter de la mise en interaction de plusieurs de ces entités dans un même système. Mais avant d'arriver au cadre des jeux stochastiques, nous examinerons deux cadres plus simples : les processus décisionnels de Markov pour les problèmes mono-agent sous incertitude et les jeux matriciels pour les domaines multi-agents et mono-état. Nous passons en revue les bases de ces cadres, en explorant à la fois les résultats de la planification et de la théorie des jeux. Ce chapitre constitue les fondations sur lesquelles le reste de notre travail sera construit.

Chapitre 2 : Algorithmes de résolution pour les jeux stochastiques

Le deuxième chapitre sera consacré à une revue des travaux liés à la résolution des jeux stochastiques. Nous examinerons alors les approches par planification et par apprentissage et nous discuterons des difficultés liées à leur résolution.

Chapitre 3 : Proposition d'un algorithme de planification pour les jeux stochastiques

Ce chapitre présentera notre contribution au domaine de la planification multi-agents en utilisant le modèle des jeux stochastiques. Nous développerons un algorithme de planification permettant de tenir compte des différents types de difficultés émanant du modèle. L'algorithme devra permettre à un agent d'évaluer individuellement ses décisions en tenant compte de la présence des autres agents. Nous introduirons alors nos solutions à des problèmes comme le calcul ou la multiplicité des équilibres. La deuxième partie de ce chapitre présentera une amélioration de l'algorithme initial, permettant de simplifier le problème de décision et d'optimiser le processus de résolution.

Chapitre 4 : Résultats expérimentaux

Le quatrième chapitre de cette thèse s'attachera à démontrer la validité de nos travaux. Nous nous intéresserons à l'efficacité de l'algorithme développé ainsi qu'à la qualité des solutions obtenues et ce en présence de différents scénarii de coopération.

Finalement, nous conclurons en tirant le bilan des travaux exposés dans cette thèse et présenterons différentes perspectives ouvertes par ce travail.

Chapitre 1

Etat de l'art

Sommaire

1.1 De l'Agent au Système Multi-Agents 7
1.2 Les processus décisionnels de Markov 9
1.2.1 Les processus stochastiques et les chaînes de Markov 10
1.2.2 Formalisme 10
1.2.3 Algorithmes de résolution 15
1.2.4 Exemple d'utilisation d'un MDP 19
1.3 Les jeux matriciels 20
1.3.1 Formalisme 21
1.3.2 Classification des jeux 22
1.3.3 Concepts de solution 24
1.4 Les jeux stochastiques 28
1.4.1 Formalisme 28
1.4.2 Extensions multi-agents 29
1.4.3 Exemples 31
1.4.4 Concepts de solution 33
1.5 Conclusion 35

Dans ce chapitre, nous présentons le cadre formel de cette thèse. Ce cadre est caractérisé par un environnement contenant un ensemble d'agents, capables d'effectuer des actions, de percevoir l'état de leur environnement et d'obtenir une certaine récompense grâce à leurs actions. Il existe plusieurs modèles pour représenter de tels environnements. Chaque modèle est caractérisé par différentes propriétés comme le nombre d'agents, le nombre d'états de l'environnement et les préférences des agents. Pour chacun de ces modèles, nous donnons une définition, un exemple de problème de décision et nous décrivons certaines approches pour déterminer la solution.

Introduction

D'un point de vue épistémologique, un modèle est un système logique ou mathématique représentant les structures essentielles d'une réalité et capable d'en expliquer ou d'en reproduire dynamiquement le fonctionnement. Un modèle peut donc servir à représenter une réalité, il s'agit de son utilisation et de sa fonction la plus courante. Il permet de décrire les règles et mécanismes en jeu dans un phénomène précis, de fournir une base commune d'étude pour de nombreuses classes de problèmes, de suggérer des solutions et de permettre la comparaison avec d'autres modèles de la réalité. C'est en ayant toutes ces raisons à l'esprit, que nous commençons ce chapitre par l'introduction de quelques modèles pour la planification mono/multi-agents.

Plus précisément, nous considérons le cadre des jeux stochastiques. Les jeux stochastiques sont considérés comme la synthèse de deux cadres plus simples : les processus décisionnels de Markov et les jeux matriciels. Les processus décisionnels de Markov ont été largement explorés dans les domaines de la planification et de l'apprentissage par renforcement, alors que les jeux matriciels forment les concepts de base du champ de la théorie des jeux. Les processus décisionnels de Markov sont associés à un modèle mono-agent ayant de multiples états. Les jeux matriciels représentent un cadre multi-agents, mais avec un seul état.

Les jeux stochastiques peuvent être considérés comme un modèle unificateur des deux modèles précédents, définissant ainsi un cadre multi-agents et multi-états. Puisque les jeux stochastiques partagent les mêmes concepts que ces deux modèles, nous avons décidé de les considérer séparément. La présentation séparée permet de mettre en valeur les problèmes fondamentaux des jeux stochastiques, tels qu'ils ont été initialement définis avec les processus décisionnels de Markov et les jeux matriciels.

Ce chapitre est organisé comme suit. Nous commençons avec un très bref aperçu de la notion d'agent et de système multi-agents. Ensuite, nous présentons les processus

décisionnels de Markov. Nous verrons notamment, une vue d'ensemble du modèle, les fondements théoriques de base et les méthodes de résolution. Nous introduisons aussi, le domaine de la théorie des jeux, les différentes classifications ainsi que ses concepts de solution.

Finalement, nous présentons les jeux stochastiques, un modèle généralisant les deux modèles précédents et offrant ainsi un contexte à la fois multi-agents et multi-états. Ce modèle permet de représenter l'environnement et les interactions inter-agents et a été souvent utilisé par les algorithmes de planification multi-agents pour la résolution des problèmes de décisions séquentielles sous incertitude. Dans son intégralité, le présent chapitre introduit la majorité des définitions nécessaires à la compréhension de cette thèse.

1.1 De l'Agent au Système Multi-Agents

Il est difficile de trouver une définition commune [Brooks, 1991, Ferber, 1995, Franklin et Graesser, 1997, Bratman *et al.*, 1988] de ce qu'est un agent. Nous adoptons la définition de Russell et Norvig [Russell et Norvig, 2003] :

Définition 1.1 *Un agent est tout ce qui peut être considéré comme percevant son environnement grâce à des capteurs et agissant sur ce même environnement par des actionneurs.*

Un agent est donc défini, comme une entité qui reçoit des informations de son environnement et qui effectue des actions pouvant modifier l'état de celui-ci. Cependant, analyser les observations de l'environnement et procéder en conséquence à des actions, nécessite une troisième faculté, celle de raisonnement. Ce qui nous amène à élargir la définition d'un agent à un triplet : Perception, Décision, Action (figure 1.1).

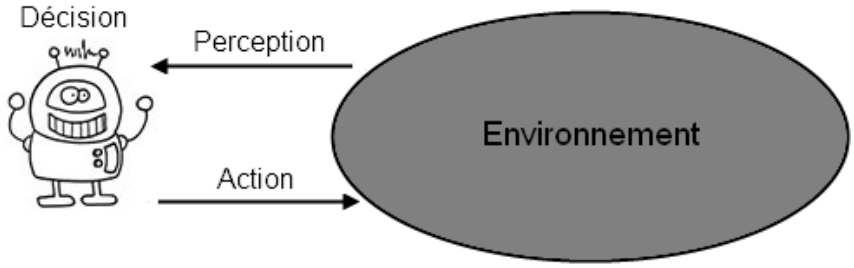


FIGURE 1.1 – Composition d'un agent : Perception, Décision, Action.

Considérons maintenant l'existence de plusieurs agents évoluant dans un seul et même environnement. Tel un agent, ils observent aussi leur environnement, raisonnent et exécutent des actions. Ce type de système est appelé : Système Multi-Agents (SMA) (figure 1.2). Le concept a été largement étudié [Durfee *et al.*, 1987, Ferber, 1995, Jennings *et al.*,

1998, Weiss, 1999, Horling et Lesser, 2005] et de même que la multitude des définitions qui existent pour un agent, plusieurs définitions de SMA subsistent aussi. Nous adoptons celle de Wooldridge [Wooldridge, 2009], définissant un système à plusieurs agents et mettant l'accent sur le concept d'interaction.

Définition 1.2 *Un système multi-agents est un système composé d'un certain nombre d'agents, interagissant entre eux.*

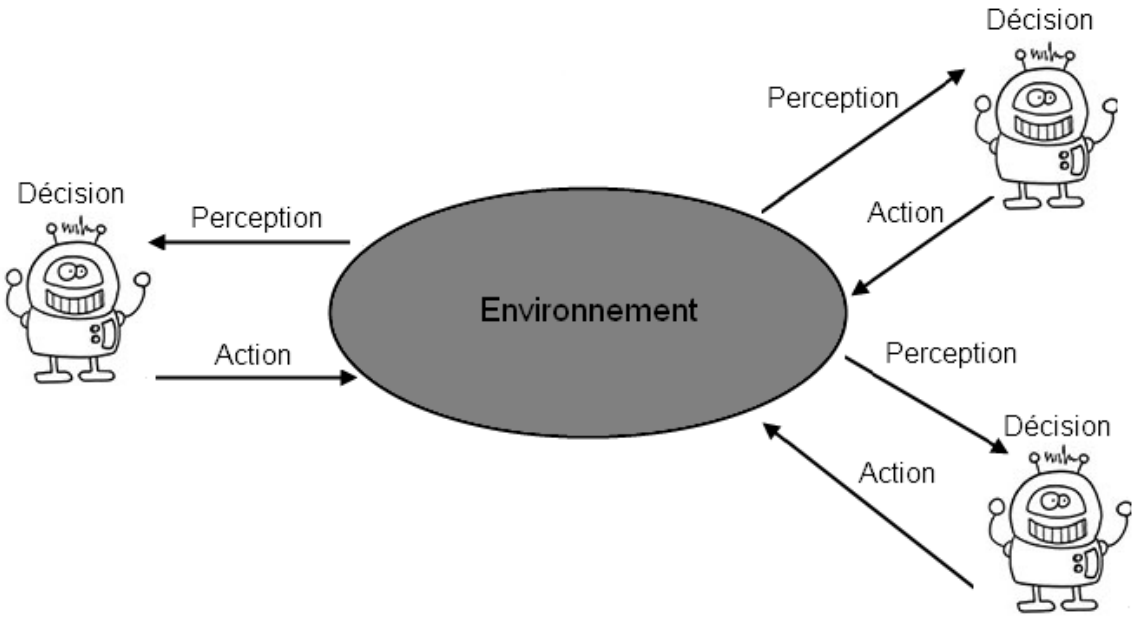


FIGURE 1.2 – Un système multi-agents.

Dans un cadre multi-agents, les agents peuvent être différents, ou avoir des objectifs distincts, interagir et faire évoluer leur environnement d'un état à un autre. Un état dans un cadre multi-agents, est défini comme une situation de l'environnement commun à un instant t. Le problème de la planification, qui est l'objet de ce travail, consiste alors à chercher les meilleures actions [1] des agents pour chaque état de l'environnement. Toutefois, il existe deux considérations à prendre en compte. La première suppose connu l'ensemble des états du système. En effet, évaluer et choisir les meilleures actions à faire, exige une connaissance préalable du monde dans lequel les agents effectuent leurs missions [2].

Une autre considération importante pour la planification multi-agents, repose sur le degré d'observabilité de l'environnement. Une hypothèse commune, que nous considérons

1. Meilleures actions : en terme de gain, d'utilité, de bien-être individuel/social, etc.
2. Une mission est un ensemble d'objectifs à atteindre par le ou les agents.

tout au long de ce travail, consiste à estimer comme complètement observable l'environnement. Cela signifie que les agents connaissent, à chaque instant, l'état du monde dans lequel ils évoluent.

Certains modèles relaxent cette hypothèse, en permettant aux agents de n'apercevoir que partiellement leur environnement. Bien que l'observabilité partielle de l'environnement représente un cadre plus réaliste [3] d'interaction, notre travail se restreint à une observabilité totale et a essentiellement pour objectif l'étude de l'effet du contexte multi-agents sur les méthodes de planification, ainsi que sur la manière dont les algorithmes de résolution devraient être conçus.

Le développement de ces algorithmes dans un contexte d'observabilité partielle les rendrait complexes, voire intraitables même pour des petits problèmes. Il est connu que la complexité de la prise de décision multi-agents dans un environnement partiellement observable est NEXP [Bernstein *et al.*, 2000, Seuken, 2007, Sigaud et Buffet, 2008]. Ainsi, nous nous concentrons sur le cas où les agents sont en mesure de percevoir sans ambiguïté les états de leur environnement.

Les sections suivantes, présenteront les formalismes des principaux modèles précédemment cités, à savoir les processus décisionnels de Markov correspondant à l'agent décrit par la figure 1.1, la théorie des jeux et les jeux stochastiques schématisés par le système multi-agents de la figure 1.2.

1.2 Les processus décisionnels de Markov

Un agent est souvent amené à prendre des décisions dans des environnements séquentiels et incertains. Le caractère séquentiel se retrouve en général dans les problèmes où l'action prise à l'instant t_0 a une incidence sur les décisions qui vont suivre à tout instant $t > t_0$. L'aspect incertain reflète quant à lui, l'incertitude sur les effets des actions, dans la mesure où ces effets ne sont pas déterministes mais stochastiques.

Nous considérons dans cette thèse, les problèmes de prise de décisions séquentielles sous incertitude ainsi que les processus décisionnels de Markov pour la modélisation de ces problèmes. Cette section introduit le modèle, ainsi que ses concepts les plus pertinents.

3. Cadre réaliste : par exemple le dysfonctionnement d'un capteur ou la défaillance d'un moteur de mouvement peuvent empêcher l'agent de connaître avec exactitude sa position.

1.2.1 Les processus stochastiques et les chaînes de Markov

Un processus stochastique $\{X_t; t \in T\}$ est un ensemble de variables aléatoires indexées par un paramètre t (temps discret) et définies sur un même espace de probabilités. La variable X_t représente l'état du processus au temps t et l'ensemble de toutes les valeurs possibles pour cette variable est appelé l'espace des états du processus et sera noté S.

Une chaîne de Markov à temps discret est un processus stochastique à temps discret $\{X_t, t = 0, 1...\}$, défini sur un espace d'états S fini et vérifiant la propriété de Markov suivante :

$$\forall i \in S, \forall t \geqslant 1 \quad [X_t = i \mid X_0, ..., X_{t-1}] = P[X_t = i \mid X_{t-1}] \tag{1.1}$$

Dans un tel processus, dit processus sans mémoire, la prédiction du futur à partir du présent ne nécessite pas la connaissance du passé.

1.2.2 Formalisme

Les chaînes de Markov permettent de modéliser les systèmes stochastiques mais n'autorisent pas un agent à intervenir et à agir sur l'évolution du système. Les processus décisionnels de Markov sont une extension des chaînes de Markov permettant de modéliser la dynamique d'un système soumis au contrôle d'un agent par le biais d'actions et en attribuant des récompenses aux transitions entre les états.

Définition 1.3 *Un processus décisionnel de Markov (MDP) [Bellman, 1957b, Puterman, 1994] est défini par le tuple $\langle S, A, P, R, T \rangle$ avec :*

- *S : ensemble fini d'états.*
- *A : ensemble fini d'actions.*
- *P : fonction de transition, $S \times A \times S \to [0, 1]$.*
- *R : fonction de récompense, $S \times A \to \mathbb{R}$.*
- *T : appelé horizon de planification, c'est le nombre d'étapes d'actions. L'horizon peut être fini ou infini.*

Etats

Un état est un ensemble de descripteurs (de l'environnement et de l'agent) permettant de décrire la configuration du monde à un instant donné. Suivant la propriété de Markov (la transition dépend uniquement de l'état courant), chaque état doit contenir l'information nécessaire à la prédiction de l'évolution du système et par conséquent à la

prise de décision. Par exemple : dans une grille bidimensionnelle, l'état peut prendre la forme (X, Y).

L'agent peut avoir une connaissance totale ou partielle de l'état dans lequel il se trouve : on parle alors d'observabilité totale ou d'observabilité partielle (POMDP). Dans un POMDP [Kaelbling *et al.*, 1998, Hansen, 1998, Pineau *et al.*, 2003], un agent ne connaît pas exactement l'état courant, mais il se base sur un état de croyances et possède une distribution de probabilité sur tous les états du système. Dans le cadre de cette thèse, nous allons nous intéresser à des problèmes décisionnels dans lesquels des agents doivent déterminer comment agir afin de maximiser leurs récompenses (individuelles). Par conséquent, nous étudierons plus particulièrement les problèmes avec une observabilité totale.

Actions

Les actions permettent la transition du système d'un état s à un état s' et sont stochastiques car elles peuvent conduire à différents états s' selon une distribution de probabilité. A noter que le modèle des MDP considère que la durée d'une transition est d'une unité de temps. Ceci étant dit, l'adaptation du modèle à des espaces d'actions continus est possible avec les MDP Semi-Markoviens [Sutton *et al.*, 1999, Barbu et Limnios, 2008].

Fonction de transition

La fonction de transition indique la probabilité de passer d'un état s à un état s' quand l'action a est exécutée. Cette fonction décrit l'évolution du système et vérifie la propriété de Markov, elle se résume alors par l'équation suivante :

$$P(s_{t+1} = s' | s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = P(s_{t+1} = s' | s_t, a_t) \tag{1.2}$$

La figure 1.3, montre un exemple simple de fonction de transition. Lorsque l'agent se trouve dans l'état s_0, il peut accomplir deux actions a_0 ou a_1. Lorsqu'il effectue l'action a_0, il a une probabilité $P(s_0, a_0, s_1)$ d'arriver dans l'état s_1 et une probabilité P(s_0, a_0, s_2) d'arriver dans l'état s_2. Lorsqu'il effectue l'action a_1, il n'a qu'une seule probabilité $P(s_0, a_1, s_3)$ d'arriver dans l'état s_3.

Fonction de récompense

La fonction de récompense indique le degré de satisfaction de l'agent lorsqu'il exécute l'action a à partir de l'état s. Cette fonction permet de traduire l'objectif à atteindre en valeurs numériques réelles et permet donc de définir des états désirables et d'autres non désirables. Il existe plusieurs façons de définir une fonction de récompense, il est

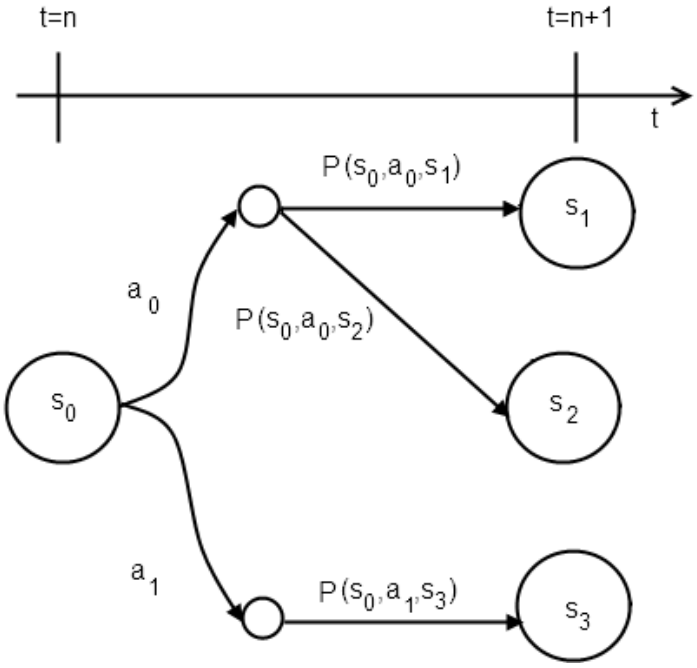


FIGURE 1.3 – Représentation d'un exemple de matrice de transition.

possible de récompenser le fait de passer d'un état s à un état s' ($R(s, a, s')$) ou bien tout simplement le fait d'être dans un état s' ($\forall s, \forall a\ R(s, a, s')$). A chaque instant $t \in T$ et en fonction de l'état s du système, une action a est choisie puis exécutée : le système évolue alors vers un état s' et une récompense $R(s, a, s')$ est observée.

Nous adoptons dans cette étude les hypothèses classiques suivantes :

– les espaces d'états et d'actions sont finis,

– les probabilités de transition ne dépendent pas du temps et sont donc stationnaires.

Politique

Le comportement de l'agent est guidé par une politique π. Une politique est définie comme une fonction de l'espace d'états vers l'espace d'actions :

$$\forall s \in S \quad \pi : S \to A$$

Cette équation signifie qu'une politique associe à chaque état s une action a. Résoudre un MDP, revient donc à calculer une politique, indiquant pour chaque état du système la meilleure action à exécuter, maximisant un certain critère d'optimalité. Cette solution est appelée politique optimale et sera notée π^*. Bellman [Bellman, 1957a] a démontré que tout MDP possédait au moins une politique optimale déterministe et stationnaire. De ce fait, la recherche de la politique peut être réalisée dans l'espace des politiques déterministes.

Critères de performance

Les processus décisionnels de Markov modélisent des systèmes où l'effet des actions de l'agent est propagé dans le temps. En effet, lorsqu'une action a est exécutée à partir d'un état s, cela a une conséquence non seulement sur la récompense immédiate mais aussi sur les transitions futures et par conséquent sur les récompenses des états successeurs. Ces récompenses à long terme, constituent l'élément commun aux différents critères de performances utilisés dans le cadre des MDPs.

Le critère de performance est mathématiquement exprimé par l'espérance des sommes de récompenses récoltées tout au long de l'horizon temporel T. Suivant les systèmes modélisés, il existe plusieurs types de critères de performance, nous en citons trois couramment utilisés :

- Horizon fini : ce critère n'est utilisé que lorsque T est considéré comme fini. Ainsi, l'espérance est calculée sur les T prochaines étapes :

$$E\Big[\sum_{t=0}^{T} r_t\Big] = E\Big[r_0 + r_1 + \ldots + r_T\Big] \tag{1.3}$$

- Horizon infini : il est possible de n'avoir *a priori* aucune information sur le nombre d'étapes de décision nécessaires pour atteindre un but. Dans ce cas, le processus évolue indéfiniment et il y a deux méthodes pour calculer l'espérance :

  1. Espérance moyenne : cette méthode consiste à calculer le gain moyen par étape de décision. Nous obtenons alors :

$$\lim_{n\to\infty} \frac{1}{n}\, E\Big[\sum_{t=0}^{n} r_t\Big] \tag{1.4}$$

     Dans ce cas, toutes les récompenses ont le même poids et cela a tendance à masquer les récompenses à court terme, au profit des récompenses à long terme.

  2. Espérance pondérée : afin d'assurer la convergence de l'espérance, un facteur d'atténuation nommé $\gamma \in [0, 1[$ est utilisé :

$$E\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big] = E\Big[r_0 + \gamma r_1 + \ldots + \gamma^t r_t + \ldots\Big] \tag{1.5}$$

     Ce facteur d'atténuation doit être choisi avec précaution car il influence le comportement de l'agent. En effet, la valeur de γ calibre le poids du futur lointain par rapport au futur proche : lorsque γ est égal à 0 alors le critère de performance correspond à la récompense immédiate. Plus γ est proche de 0, plus les récompenses à court terme sont privilégiées. Au contraire, plus γ est proche de 1, plus les récompenses à long terme sont privilégiées.

Pour la suite de l'exposé, nous considérons uniquement le critère γ-pondéré, car il reste le critère le plus utilisé mais aussi le plus général.

Fonction de Valeur

Dans le but de maximiser ses gains, l'agent doit faire son choix parmi plusieurs politiques selon la valeur de chacune. La fonction de valeur d'une politique π, associe à chaque état s la valeur du critère de performance (considéré), en suivant π depuis s :

$$\forall \pi \in \Pi \ , \ V^{\pi} : S \rightarrow \mathbb{R} \tag{1.6}$$

En tenant compte du critère γ-pondéré, la fonction de valeur d'une politique π s'écrit alors :

$$V^{\pi}(s) = E^{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \Big] \tag{1.7}$$

où E^{π} désigne l'espérance mathématique lorsque la politique π est suivie.

Une politique π^* est dite optimale lorsque :

$$\forall \pi \in \Pi \ \forall s \in S, \ V^{\pi^*}(s) > V^{\pi}(s) \tag{1.8}$$

Le principe d'optimalité de Bellman L'équation de Bellman [Bellman, 1957b] est une équation récursive qui établit une propriété fondamentale de la fonction de valeur. Elle s'obtient en développant la définition de la fonction de valeur d'une politique π (équation 1.7) pour tout état s :

$$V^{\pi}(s) = R(s, \pi(s)) + \sum_{s' \in S} P(s, \pi(s), s') \ V^{\pi}(s') \tag{1.9}$$

Cette équation signifie que lorsque une action est exécutée (avec une certaine distribution de probabilité sur les états futurs) à partir de l'état s, la valeur de celle-ci sera constituée de :

– la récompense instantanée,

– l'espérance des récompenses à venir dans le futur.

A partir de l'équation de Bellman, on peut exprimer la fonction de valeur de la politique optimale, on la note V^* et elle est appelée équation d'optimalité de Bellman :

$$\forall s \in S, \ V^*(s) = max_{a \in A}\Big[\ R(s,a) + \sum_{s' \in S} P(s,a,s') \ V^*(s') \ \Big] \tag{1.10}$$

Optimalité Cette équation peut se réécrire de manière condensée : $V^* = FV^*$, avec F l'opérateur contractant de facteur γ pour la norme infinie [Puterman, 1994] et son unique point fixe est la fonction de valeur optimale V^*. L'opérateur F permet notamment d'exprimer le fait qu'une politique soit gloutonne par rapport à une fonction de valeur. Pour toute fonction de valeur V, on appelle politique gloutonne par rapport à V une politique π définie par :

$$\pi(s) = argmax_{a \in A}[\ R(s,a) + \gamma \sum_{s' \in S} P(s,a,s')\ V(s')\ ] \tag{1.11}$$

Ainsi, si on connaît la fonction de valeur optimale, alors on en déduit une politique optimale π^* en sélectionnant une politique gloutonne par rapport à V^*. A noter que pour un MDP, il peut exister plusieurs politiques optimales, toutes sont associées à la même fonction valeur V^*.

L'équation de Bellman est la source de plusieurs algorithmes de résolution des MDP, en particulier *Value-Iteration* [Sutton et Barto, 1998] et *Policy-Iteration* [Howard, 1960] que nous présenterons par la suite.

1.2.3 Algorithmes de résolution

Il existe essentiellement deux approches pour résoudre un MDP en fonction des connaissances dont dispose l'agent :

- Si l'agent connaît le modèle du monde, il peut s'en servir et planifier [Puterman, 1994] ses actions pour construire une politique optimale π^*.
- Si l'agent ne connaît pas le modèle du monde, il peut interagir avec son environnement pour acquérir de l'expérience et en inférer une politique optimale directement. Pour plus de détails, le lecteur peut se référer à Watkins [Watkins et Dayan, 1992] qui fournit une introduction aux mécanismes d'apprentissage.

Nous allons nous intéresser à la première approche, étant donné que nos travaux se placent dans le cadre de la planification. Dans ce cadre, Papadimitriou et Tsitsiklis [Papadimitriou et Tsitsiklis, 1987] ont étudié la complexité de résolution d'un MDP et ont prouvé que c'était un problème P-complet.

Il existe plusieurs familles d'algorithmes de résolution des MDP en planification, nous nous intéressons particulièrement aux algorithmes issus de la technique de programmation dynamique, couramment utilisée dans ce domaine. Ce concept a été inventé par Bellman et désigne un ensemble d'algorithmes permettant de calculer la politique optimale π^* d'un MDP fini.

Value-Iteration

L'algorithme d'itération sur les valeurs (*Value-Iteration*) [Sutton et Barto, 1998], est l'algorithme de programmation dynamique le plus couramment utilisé en raison de sa simplicité. Il consiste à améliorer graduellement la fonction de valeur initialisée arbitrairement. Selon le principe d'optimalité de Bellman, chaque itération de l'algorithme procède à une mise à jour de la fonction de valeur de l'ensemble des états.

L'équation de Bellman étant une contraction, il a été démontré [Puterman, 1994] que V_t converge asymptotiquement vers la fonction de valeur optimale V^* (quelle que soit l'initialisation de V_0).

Différentes méthodes peuvent intervenir pour l'arrêt de l'algorithme. La plus commune consiste à stopper l'algorithme lorsque la distance entre deux valeurs successives devient inférieure à un certain seuil ϵ :

$$||V_t - V_{t-1}|| \ < \epsilon \tag{1.12}$$

Par conséquent, une garantie sur la distance restante par rapport à la fonction de valeur optimale V^* est établie :

$$||V_t - V^*|| < \frac{2\gamma}{1-\gamma}\epsilon \tag{1.13}$$

De ce calcul itératif et du critère d'arrêt qui vient d'être vu, nous déduisons l'écriture de *Value-Iteration* présentée par l'algorithme 1.

Algorithme 1: Algorithme Value-Iteration

Entrées : MDP(S,A,P,R,T), $\gamma \in [0,1[$, $\epsilon \geq 0$
Sorties : π^*

pour $s \in S$ **faire**
  $V_0(s) = 0$
$t \leftarrow 0$
répéter
  pour $s \in S$ **faire**
    $V_{t+1}(s) = max_{a \in A}[\ R(s,a) + \gamma \sum_{s' \in S} P(s,a,s')\ V_t(s')\ ]$
  $t \leftarrow t+1$
jusqu'à $max_{s \in S}|V_{t+1}(s) - V_t(s)| < \epsilon$;
pour $s \in S$ **faire**
  $\pi^*(s) = argmax_{a \in A}[\ R(s,a) + \gamma \sum_{s' \in S} P(s,a,s')\ V^*(s')\ ]$

Complexité Afin de converger vers une politique optimale π^*, l'algorithme nécessite l'utilisation de différentes ressources. Ainsi, nous nous attachons à étudier les complexités temporelle et spatiale de l'algorithme.

La complexité temporelle de *Value-Iteration* dépend essentiellement de deux facteurs [Littman *et al.*, 1995] : la complexité d'une itération et le nombre nécessaire d'itérations pour converger vers une solution optimale :

- Complexité d'une itération : l'opération la plus coûteuse reste le calcul d'une transition entre les états, d'où une complexité $O(|S|^2|A|)$.
- Nombre d'itérations : ce nombre varie selon la taille du problème traité, mais il reste polynomial en fonction de S, A, γ et B, où B est le nombre de bits nécessaire à la représentation des fonctions P et R.

La complexité spatiale est linéaire en fonction de $|S||A|$ (il faut stocker en mémoire les espaces d'états et d'actions ainsi que $V_t(s)$ durant toute la phase de planification).

A noter que cet algorithme appartient à la classe des algorithmes *Anytime* [Hendler, 1992, Zilberstein, 1996]. Contrairement à un algorithme standard, où aucun résultat n'est accessible avant la fin de son exécution, un algorithme *Anytime* peut être stoppé à tout instant, et doit fournir des solutions de qualité croissante au cours du temps.

Policy-Iteration

L'algorithme *Policy-Iteration* [Howard, 1960] constitue une autre classe d'algorithmes de résolution basé sur le principe d'optimalité de Bellman. L'algorithme a besoin d'une politique π_0 par défaut qu'il va améliorer au fur et à mesure afin de la faire converger vers une politique optimale π^*. Chaque itération de l'algorithme (algorithme 2) se déroule en deux phases :

1. phase d'évaluation : calculer V_t^π en utilisant l'équation de Bellman, revient à résoudre un système d'équations linéaires de taille $|S|$.
2. phase d'amélioration : consiste simplement à choisir pour chaque état $s \in S$, l'action qui maximise l'espérance calculée à partir de la politique courante π_t.

La condition d'arrêt adoptée $\pi_t = \pi_{t+1}$ assure la convergence [4] vers une politique optimale.

4. Cela tient au fait que le nombre de politiques est considéré comme fini.

Complexité Comme pour l'algorithme *Value-Iteration*, la complexité en temps de l'algorithme dépend essentiellement de la complexité d'une itération et du nombre d'itérations :

- Complexité d'une itération : chaque itération consiste en deux opérations, la phase d'évaluation est effectuée en $O(|S|^3)$ et la phase d'amélioration est en $O(|A| \times |S|^2)$.
- Nombre d'itérations : ce nombre reste toujours dépendant du problème étudié. Puterman [Puterman, 1994] avance les mêmes résultats concernant le caractère polynomial de l'algorithme.

Concernant la complexité en espace, *Policy-Iteration* est linéaire en $|S||A|$. En théorie, le nombre maximum d'itérations nécessaires pour atteindre π^* est le même que celui de l'itération sur les valeurs [Littman *et al.*, 1995]. En pratique, le nombre d'itérations requis est plus faible que celui de l'algorithme *Value-Iteration*, mais au prix d'une complexité supérieure pour chaque itération. Le choix d'utiliser *Value-Iteration* ou *Policy-Iteration* demeure difficile et Littman [Littman, 1996] avance que ce choix dépendrait de la structure du problème.

Après une brève présentation des aspects formels des MDP, nous illustrons leur application par un exemple concret.

Algorithme 2: Policy-Iteration

Entrées : MDP(S,A,P,R,T), $\gamma \in [0, 1[$
Sorties : $\pi^* = \pi_{t+1}$

$\pi_{t+1} = \pi_0$
$t \leftarrow 0$
répéter
  $\pi_t = \pi_{t+1}$
  $t \leftarrow t + 1$
  pour $s \in S$ **faire**
    Calculer $V^\pi(s)$
  pour $s \in S$ **faire**
    si $\exists a \in A,\ [R(s,a) + \gamma \sum_{s' \in S} P(s,a,s')\ V_t^\pi(s')] > V_t^\pi(s)$ **alors**
      $\pi_{t+1}(s) = a$
    sinon
      $\pi_{t+1}(s) = \pi_t$
jusqu'à $\forall s \in S,\ \pi_t = \pi_{t+1}$;

1.2.4 Exemple d'utilisation d'un MDP

Initialisation du problème

Un agent se trouve en un point A sur une grille (figure 1.4 (a)), muni de quatre actions de déplacement (Haut, Bas, Gauche et Droite) il doit pouvoir aller au point B tout en évitant "les marécages" qu'il peut rencontrer.

Création du MDP

Rappelons qu'un MDP est défini par le tuple $\langle S, A, P, R, T \rangle$:

- S : ensemble des positions de l'agent dans la grille de dimension $n \times n$ (nous avons alors n^2 états différents).
- A : ensemble des actions permettant d'aller vers les quatre points cardinaux entourant l'agent.
- P : matrice de transition habituelle entre les différents états, incluant quelques règles pour ne pas traverser les murs.
- R : fonction de récompense, elle est définie arbitrairement comme suit :
  - $R(s) = +1$: lorsque l'agent atteint le point B.
  - $R(s) = -10$: lorsque l'agent passe sur un marécage.
- T : horizon, considéré comme infini.

Pour faciliter la mise en place de notre MDP, nous avons considéré que les actions de l'agent sont déterministes, c'est-à-dire que pour chaque état il n'y a qu'une seule action possible.

Résultats

La résolution de ce MDP a été réalisée à l'aide de l'algorithme *Value-Iteration*, avec $\gamma = 0.9$ et $\epsilon = 0.01$. Il a fallu plus d'une vingtaine d'itérations pour que l'algorithme converge. Par ailleurs, nous avons constaté que la politique optimale est souvent obtenue bien avant que l'algorithme ne s'arrête. En effet, les dernières itérations réalisent des améliorations de la fonction de valeur qui n'ont pour effet que d'augmenter la précision de l'évaluation (figure 1.4 (b)) mais ne modifient pas la politique (figure 1.4 (c)).

Etant donné sa position initiale sur le point A, l'agent n'a qu'à suivre le chemin suivant (figure 1.4 (d)) : D, H, H, H, H, G, G, G pour arriver à son objectif au point B.

Tout au long de cette section, nous nous sommes préoccupés des MDP, un modèle conçu initialement pour des problèmes mono-agent/multi-états. Nous avons relié les choix qui pouvaient être faits par un agent avec les gains qu'il pouvait en dériver. La section

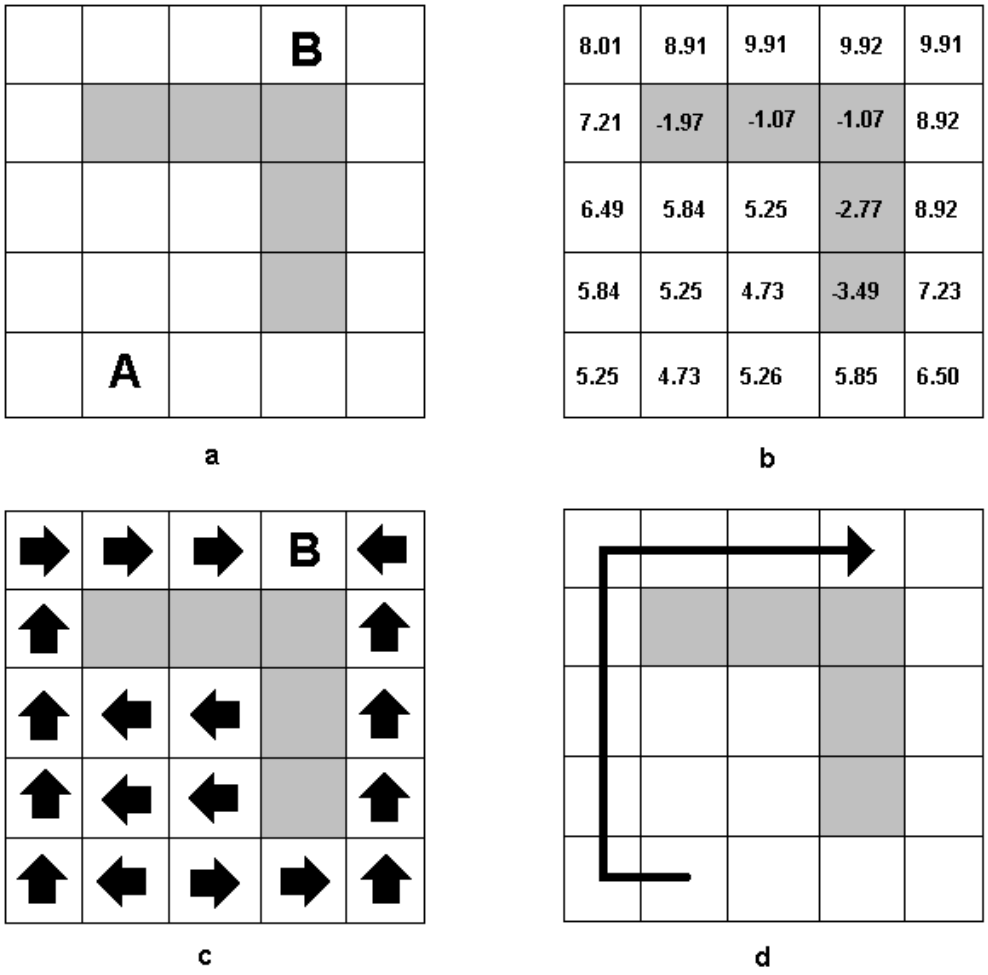


FIGURE 1.4 – Application des MDPs.

suivante, introduira le domaine de la théorie des jeux, permettant la modélisation des contextes dans lesquels plusieurs agents sont en situation de choix, les choix des uns affectant le gain des autres.

1.3 Les jeux matriciels

Les premiers travaux sur la théorie des jeux apparaissent au début du XXe siècle avec Zermelo (1912), Borel (1921), Von Neumann (1928). La théorie des jeux naît réellement comme discipline sous l'impulsion du mathématicien John Von Neumann et de l'économiste Oskar Morgenstern, qui écrivent en 1944 un livre fondateur : Games and Economic Behavior [Neumann *et al.*, 2007]. Depuis, la théorie des jeux a connu un développement mathématique important et de nombreuses applications dans diverses disciplines : économie, biologie, informatique, etc.

Qu'est-ce qu'un jeu ?

Un jeu est une situation où des individus (les joueurs) sont conduits à faire des choix parmi un certain nombre d'actions possibles, et dans un cadre défini à l'avance (les règles du jeu), permettant de déterminer qui peut faire quoi et quand. Les résultats de ces choix constituent une issue du jeu à laquelle est associée un gain pour chacun des participants. Ces résultats ne dépendent pas de la décision d'un seul joueur et ne dépendent pas non plus uniquement du hasard, bien que celui-ci puisse intervenir.

L'intérêt de la théorie des jeux est de proposer un outil permettant d'analyser l'aspect stratégique des interactions. On ne soucie pas uniquement du résultat, mais de la façon de l'atteindre. L'aspect stratégique signifie que chaque joueur sait que son choix aura une influence sur son résultat mais aussi sur celui des autres et inversement, il sait que son résultat va dépendre de ce que feront les autres.

1.3.1 Formalisme

Il existe deux formes de représentation des jeux : la forme extensive et la forme stratégique (forme normale). La forme extensive [Guerrien, 2002] utilise un arbre de décision décrivant les actions possibles des joueurs à chaque étape du jeu, ainsi que l'information dont ils disposent pour prendre leur décision. Lorsque le jeu est à coups simultanés, la représentation par la forme extensive devient particulièrement lourde et plus compliquée. Avec la forme stratégique, les joueurs peuvent choisir leurs actions de manière simultanée sans connaître le choix des autres et le jeu est décrit sous la forme d'une matrice. De ce fait, nous nous intéressons uniquement à cette dernière forme de représentation.

Définition 1.4 *Un jeu matriciel ou jeu sous forme stratégique est défini par le tuple suivant [Fudenberg et Tirole, 1991, Gibbons, 1992, Leyton-Brown et Shoham, 2008] :*

$$\langle Ag, A_i : i = 1 \dots |Ag|, R_i : i = 1 \dots |Ag| \rangle$$

- *Ag : ensemble fini de joueurs*[5].
- *A_i : ensemble fini des stratégies a_i de l'agent i. Conformément aux notations, on désigne $a_{-i} \in A_{-i}$ l'ensemble de toutes les stratégies sauf celle de l'agent i. Nous utiliserons également la notation $a = \langle a_i, a_{-i} \rangle$ pour désigner la stratégie jointe $a \in A$, où l'agent i choisit la stratégie a_i tandis que les autres choisissent les stratégies a_{-i}. L'ensemble des stratégies jointes A est défini par $A = A_1 \times \dots A_{|Ag|}$.*
- *R_i : fonction de récompense de l'agent i, $R_i(A) \rightarrow \mathbb{R}$, où le gain d'un joueur dépend non seulement de sa stratégie mais aussi de celles des autres joueurs.*

5. joueur : nous utilisons également le terme agent pour désigner un joueur.

Dans de tels jeux, l'ensemble des gains possibles qui peuvent être obtenus par les joueurs, est représenté par une matrice. Pour un jeu à deux joueurs, le principe est relativement simple : le premier joueur choisit ses actions à partir des lignes et le second joueur choisit ses actions à partir des colonnes. Le choix conjointement effectué, détermine les gains de chaque joueur en fonction de la matrice jouée.

La figure 1.5 représente sous forme matricielle un exemple de jeu, avec deux joueurs (*Joueur1* et *Joueur2*), chacun ayant deux actions a et b. Par convention les gains sont représentés sous la forme *(x,y)*, où x est le gain du joueur ligne (*Joueur1*) et y est le gain du joueur colonne (*Joueur2*). Si le premier joueur choisit la première ligne et que le second choisit la deuxième colonne, alors ils auront respectivement les paiements [6] 10 et 0.

		Joueur2	
		a	b
Joueur1	a	(-5,-1)	(10,0)
	b	(0,5)	(2,1)

FIGURE 1.5 – Un exemple de jeu matriciel.

1.3.2 Classification des jeux

De manière générale, les jeux matriciels sont classés suivant la nature de la fonction de récompense. De ce fait, nous pouvons citer les classes suivantes :

– Les jeux à somme nulle : il n'y a que deux joueurs et les gains de l'un représentent les pertes de l'autre ($R_1 = -R_2$), voir figure 1.6. Ce jeu comporte deux joueurs, chacun possède une pièce et doit choisir secrètement l'un des deux côtés. Le *joueur1* gagne si les deux pièces sont sur le même côté, sinon c'est le *joueur2* qui gagne.

		Joueur2	
		Face	*Pile*
Joueur1	*Face*	(1,-1)	(-1,1)
	Pile	(-1,1)	(1,-1)

FIGURE 1.6 – Le jeu du pile ou face (*Matching Pennies*).

– Les jeux à somme non nulle (générale) : les gains peuvent être de nature quelconque. Ces jeux permettent de modéliser toutes les situations, où par exemple certaines

6. Paiement : nous utilisons également les termes gain et récompense.

issues sont profitables pour tous. La figure 1.7 présente le dilemme du prisonnier, un problème classique en théorie des jeux. Deux suspects sont arrêtés par la police. Mais les agents n'ont pas assez de preuves pour les inculper, donc ils les interrogent séparément en leur faisant la même offre. Si un des deux prisonniers dénonce l'autre, il est remis en liberté alors que le second obtient la peine maximale (10 ans). Si les deux se dénoncent entre eux, ils seront condamnés à une peine plus légère (5 ans). Si les deux refusent de dénoncer, la peine sera minimale (6 mois), faute d'éléments au dossier..

		Prisonnier2	
		se taire	*dénoncer*
Prisonnier1	*se taire*	(-1/2,-1/2)	(-10,0)
	dénoncer	(0,-10)	(-5,-5)

FIGURE 1.7 – Le jeu du dilemme du prisonnier.

A ces deux classifications principales, se rajoutent d'autres distinctions :

– jeu à information complète/incomplète : si chaque joueur connaît, en plus des siens, les ensembles de stratégies et les fonctions de récompense de tous les autres joueurs, alors le jeu est considéré comme à information complète. A l'inverse, un jeu est dit à information incomplète si les joueurs ne connaissent que certains éléments du jeu.

– jeu statique/dynamique : un jeu statique est un jeu se jouant en un seul tour, où tous les joueurs choisissent leur unique coup simultanément. Contrairement aux jeux en forme statique, qui sont joués une fois, les jeux dynamiques décrivent des processus dépendant du temps. Ainsi, les joueurs peuvent adapter leur comportement présent sur les décisions observées des autres joueurs dans le passé.

– jeu à information parfaite/imparfaite : on parle de jeu à information parfaite dans le cas de jeu à mécanisme séquentiel, où chaque joueur a connaissance en détail de toutes les actions effectuées avant son choix.

Finalement, nous introduisons un dernier type de jeu, considéré à la fois comme dynamique et à information complète : les jeux répétés. La répétition d'un jeu, avec connaissance des résultats intermédiaires, change souvent son déroulement (les meilleurs coups et la conclusion). Par exemple, il peut être utile de prendre ponctuellement le risque de perdre "pour voir", de tester les autres joueurs, et de mettre en place des stratégies de communication par les coups joués (à défaut d'autre moyen de communication). Il se développe également des phénomènes de réputation qui vont influencer les choix stratégiques des autres joueurs. Enfin, le fait que le nombre total de parties soit connu à l'avance ou

non peut avoir des effets importants sur le résultat, l'ignorance du nombre de coups rapproche ainsi des jeux avec un nombre infini d'étapes, alors que sa connaissance rapproche au contraire du jeu à coup unique. Pour plus d'approfondissement, le lecteur est encouragé à consulter les ouvrages de référence [Fudenberg et Tirole, 1991, Sorin, 2002, Mailath et Samuelson, 2006].

1.3.3 Concepts de solution

Pour un joueur i, une stratégie est dite pure, si elle est choisie de manière déterministe et elle est considérée comme *mixte* lorsqu'elle est définie avec une certaine distribution de probabilité sur les stratégies pures, soit :

$$\sum_{a_i \in A_i} \Delta_i(a_i) = 1 \tag{1.14}$$

Conformément à cette notation, nous considérons Δ_{-i} comme l'ensemble des stratégies mixtes des autres joueurs (exceptée celle du joueur i). Comme pour les processus décisionnels de Markov, le joueur tente de maximiser un certain critère de performance, il s'agit de maximiser ses gains espérés. Le gain espéré (ou l'utilité espérée) d'un joueur, est la récompense qu'il peut obtenir étant donné sa politique, mais aussi les politiques des autres agents.

Définition 1.5 *Le gain espéré du joueur i selon la stratégie jointe $\pi = \langle \pi_i, \pi_{-i} \rangle$, est noté U_i^π et s'écrit :*

$$\begin{aligned} U_i^\pi &= E_{a \in A}\ [R_i(a)] \\ &= \sum_{a \in A} \Delta_a\ R_i(a) \\ &= \sum_{a_i \in A_i} \Delta_i(a_i) \sum_{a_{-i} \in A_{-i}} \Delta_{-i}(a_{-i})\ R_i(\langle a_i, a_{-i} \rangle) \end{aligned} \tag{1.15}$$

Prenons comme exemple, le dilemme du prisonnier et supposons les probabilités suivantes : pour le premier prisonnier, ses deux stratégies *se taire* et *dénoncer* ont respectivement les probabilités 0.75 (p) et 0.25 ($1-p$). Pour le deuxième prisonnier, *se taire* se fait avec une probabilité de 0.4 (q) et donc *dénoncer* se fait avec une probabilité de 0.6 ($1-q$). Le gain espéré du premier prisonnier est alors -5.4. Le même raisonnement pour le deuxième prisonnier donne (-1.9). Si le premier prisonnier veut maximiser son gain espéré, alors il doit trouver une probabilité p telle que $\frac{d(U_1)}{dq}(p) = 0$ (idem pour le deuxième avec $\frac{d(U_2)}{dp}(q) = 0$).

$$
\begin{aligned}
U_1^{\pi} &= \Delta_1(\textit{se taire}) \times \Delta_2(\textit{se taire}) \times R_1(\langle \textit{se taire}, \textit{se taire} \rangle) \\
&\quad + \Delta_1(\textit{se taire}) \times \Delta_2(\textit{dénoncer}) \times R_1(\langle \textit{se taire}, \textit{dénoncer} \rangle) \\
&\quad + \Delta_1(\textit{dénoncer}) \times \Delta_2(\textit{se taire}) \times R_1(\langle \textit{dénoncer}, \textit{se taire} \rangle) \\
&\quad + \Delta_1(\textit{dénoncer}) \times \Delta_2(\textit{dénoncer}) \times R_1(\langle \textit{dénoncer}, \textit{dénoncer} \rangle) \\
&= 0.75 \times 0.4 \times (-1/2) + 0.75 \times 0.6 \times (-10) + 0.25 \times 0.4 \times (0) + 0.25 \times 0.6 \times (-5) \\
&= -5.4
\end{aligned}
$$

Equilibre de Nash

Pour les MDP, la solution est définie comme la politique ayant la valeur la plus élevée selon une certaine formulation des récompenses. Dans les jeux matriciels, il ne peut y avoir de stratégie optimale évaluée indépendamment des stratégies des autres agents.

Ceci peut être illustré avec le jeu du *Matching Pennies* (évoqué précédemment). Dans ce jeu, si le $joueur2$ joue $Face$, alors la stratégie optimale du $joueur1$ est de jouer $Face$, mais si le $joueur2$ va jouer $Pile$, alors la stratégie optimale du $joueur1$ est de jouer $Pile$. Dans ce jeu, il n'y a aucune stratégie (pure ou mixte) qui peut être qualifiée d'optimale en faisant abstraction de l'adversaire. Lorsque toutes les solutions du jeu sont dépendantes des choix des autres, cela nous amène à introduire le concept de Meilleure réponse (en anglais *Best-Response - BR*).

Définition 1.6 *Pour un jeu matriciel, la fonction Meilleure réponse du joueur i fournit l'ensemble de toutes les stratégies optimales, étant donné que les autres joueurs jouent la stratégie jointe π_{-i}. Formellement, $\pi_i^* \in BR(\pi_{-i})$ si et seulement si :*

$$\forall \pi_i \in \Delta(A_i) \quad R_i(\langle \pi_i^*, \pi_{-i} \rangle) \geqslant R_i(\langle \pi_i, \pi_{-i} \rangle) \tag{1.16}$$

avec $\Delta(A_i)$ l'espace de toutes les stratégies mixtes de l'agent i.

Lorsque tous les joueurs cherchent à maximiser leur gain, l'intersection de toutes les fonctions "Meilleure réponse" peut alors former un équilibre. L'avancée majeure qui a conduit, en grande partie, au développement de la théorie des jeux est la notion d'équilibre de Nash [Nash, 1950].

Définition 1.7 *Un équilibre de Nash est une combinaison stratégique où chaque joueur i joue sa meilleure réponse face aux stratégies de tous les autres.*

$$\forall \pi_i \in \Delta(A_i) \quad \pi_i^* \in BR(\pi_{-i}^*) \tag{1.17}$$

L'équilibre de Nash peut s'écrire aussi avec :

$$\forall i \in Ag,\ \forall \pi_i \in \Delta(A_i) \quad R_i(\langle \pi_i^*, \pi_{-i}^* \rangle) \geqslant R_i(\langle \pi_i, \pi_{-i}^* \rangle) \tag{1.18}$$

Autrement dit, aucun agent ne peut faire mieux en changeant de stratégie, étant donné que les autres agents continuent à suivre leur part de l'équilibre. L'équilibre de Nash possède donc une propriété de stabilité qui est satisfaite pour chaque agent, c'est pourquoi on parle d'équilibre. Il a été montré par Nash [Nash, 1951], que chaque jeu possède au moins un équilibre de Nash en stratégies mixtes, néanmoins il peut ne pas y avoir d'équilibre en stratégies pures, ou au contraire y en avoir plusieurs.

Par exemple, dans le jeu du dilemme du prisonnier, l'équilibre de Nash (en stratégies pures) correspond au cas où les deux agents choisissent l'action *dénoncer*. En effet, si l'un d'eux tente de changer de stratégie et décide donc de *se taire*, il ne fera que rallonger sa peine de prison. Dans le jeu du *Matching Pennies*, il n'y a pas d'équilibre de Nash en stratégies pures, l'équilibre en stratégies mixtes correspond au fait que l'un ou l'autre des joueurs choisisse avec une probabilité de 1/2 de jouer *pile* ou $face$.

		Joueur2	
		Documentaire	*Comédie*
Joueur1	*Documentaire*	(2,3)	(1,1)
	Comédie	(1,1)	(3,2)

FIGURE 1.8 – La guerre des sexes.

Pour voir un exemple de la multiplicité d'équilibres, nous allons considérer le jeu de la guerre des sexes (figure 1.8). Un homme et une femme vont au cinéma, une fois sur place, ils doivent choisir entre aller voir un documentaire ou une comédie. L'un des deux préfère les documentaires et l'autre les comédies, mais tous deux préfèrent voir un film ensemble que séparément. Les stratégies disponibles pour chacun des deux sont alors : aller voir un Documentaire ou aller voir une Comédie. Dans ce jeu, les deux équilibres de Nash en stratégies pures sont les cas où les deux agents choisissent ensemble de regarder un documentaire $(2, 3)$ ou de voir une comédie $(3, 2)$.

Ces deux issues sont intéressantes, car en les considérant on s'aperçoit rapidement qu'il est impossible en supposant qu'une autre issue soit atteinte, d'améliorer le gain de l'un des deux joueurs sans diminuer le gain de l'autre. De cette remarque, nous pouvons introduire les concepts de Pareto-dominance et d'optimum de Pareto [Fudenberg et Tirole, 1991, Ng, 2004].

Définition 1.8 *Une action jointe $\hat{\pi}$ domine au sens de Pareto une autre action jointe π, si et seulement si :*

- *chaque agent i suivant $\hat{\pi}_i$ reçoit au moins le même gain qu'en suivant π_i :*

$$\forall i \in Ag \quad R_i(\langle \hat{\pi}_i, \hat{\pi}_{-i} \rangle) \geqslant R_i(\langle \pi_i, \pi_{-i} \rangle) \tag{1.19}$$

- *au moins un agent j suivant $\hat{\pi}_j$ reçoit un gain strictement supérieur à ce qu'il recevrait s'il suivait π_j :*

$$\exists j \in Ag \quad R_j(\langle \hat{\pi}_j, \hat{\pi}_{-j} \rangle) > R_i(\langle \pi_j, \pi_{-j} \rangle) \tag{1.20}$$

Une stratégie jointe est Pareto optimale, si aucun agent ne peut améliorer son gain sans qu'au moins un autre agent ne constate alors la diminution de son gain.

Définition 1.9 *Si une stratégie jointe $\hat{\pi}^*$ n'est dominée au sens de Pareto par aucune autre stratégie, alors $\hat{\pi}^*$ est Pareto optimale.*

Par rapport à l'exemple de la guerre des sexes, les deux équilibres de Nash trouvés sont considérés comme des issues Pareto-optimales. Par contre, dans le dilemme du prisonnier, la combinaison $(\textit{dénoncer}, \textit{dénoncer})$ est un équilibre de Nash mais la combinaison $(\textit{se taire}, \textit{se taire})$ la domine au sens de Pareto. Il faut donc retenir qu'un équilibre de Nash n'est pas forcément un optimum de Pareto. A noter également, que dans le cas où un jeu matriciel comporte de multiples équilibres, un équilibre peut en dominer un autre au sens de Pareto.

Nous estimons à ce stade que les concepts énoncés sont suffisants pour comprendre les jeux que nous étudierons par la suite. Les propriétés et autres théorèmes utilisés seront introduits au moment de leur utilisation. Pour un approfondissement sur la théorie des jeux, on pourra se référer notamment aux références suivantes [Gibbons, 1992, Leyton-Brown et Shoham, 2008, Shoham et Leyton-Brown, 2009].

Dans cette section nous avons présenté brièvement le domaine de la théorie des jeux et plus particulièrement les jeux matriciels. Nous avons introduit les concepts les plus importants, tels que les notions de gain espéré, de meilleure réponse et d'équilibre de Nash. Nous avons également montré, à travers plusieurs exemples, leur intérêt dans la modélisation des interactions stratégiques entre plusieurs agents. Ces jeux sont notamment intéressants, d'une part car ils constituent une partie intégrante des jeux stochastiques et d'autre part car il est avantageux de réutiliser toute leur ingénierie liée à la résolution et à la recherche d'équilibre.

1.4 Les jeux stochastiques

Les jeux stochastiques représentent un cadre permettant de modéliser des problèmes de décision dans l'incertain, de plusieurs agents interagissant dans un système multi-états. Ils étendent les MDP au cas de plusieurs agents dans un environnement commun. Les agents exécutent une action jointe qui définit la récompense obtenue par chaque agent et le nouvel état du système. Les jeux stochastiques peuvent être également perçus comme des jeux répétés à plusieurs états. Cela signifie que chaque état est un jeu matriciel et que les transitions se feront entre différents jeux matriciels. Ce cadre représente donc une généralisation des MDP et des jeux matriciels.

1.4.1 Formalisme

Les jeux stochastiques représentent un cadre formel permettant de modéliser un environnement multi-agents non-coopératif. Le fait d'être non-coopératif signifie que les agents ne possèdent pas *a priori* d'objectif commun, mais des objectifs individuels. Cependant, ils peuvent être amenés à coopérer, pour atteindre leurs buts individuels.

Shapley [Shapley, 1953] a été le premier à définir le modèle des jeux stochastiques. Un espace d'états S et une probabilité de transition P ont été ainsi ajoutés à la définition des jeux matriciels.

Définition 1.10 *Un jeu stochastique [Shapley, 1953, Shoham* et al.*, 2003] est défini par le tuple :*

$$\langle Ag, A_i : i = 1 \dots |Ag|,\ R_i : i = 1 \dots |Ag|,\ S,\ P \rangle$$

- Ag *: ensemble fini d'agents.*
- A_i *: ensemble fini des actions de l'agent* i *(*$i \in Ag$*).*
- R_i *: fonction de récompense de l'agent* i*,* $R_i(a) \rightarrow \mathbb{R}$*.*
- S *: ensemble fini d'états, où chaque état* $s \in S$ *est un jeu matriciel.*
- P *: fonction de transition*[7] *du jeu,* $P : S \times A \times S \rightarrow [0, 1]$*, indiquant la probabilité de passer d'un état* $s \in S$ *à un état* $s' \in S$ *en exécutant l'action jointe* a*.*

Remarque 1 : la notion d'horizon dans les jeux stochastiques est la même que celle introduite précédemment pour les MDP.

Remarque 2 : la même classification que pour les jeux matriciels est utilisée avec les jeux stochastiques. Les jeux à somme nulle sont des jeux où la somme des fonctions de

7. La fonction de transition d'un jeu stochastique vérifie la propriété de Markov, c'est pourquoi les jeux stochastiques sont également appelés jeux de Markov.

récompense de tous les agents est nulle. Les jeux à somme générale se réfèrent à tous les types de structures de récompense.

Les caractéristiques principales des jeux stochastiques sont (1) l'observabilité (individuelle) totale et (2) la fonction de récompense propre à chaque agent. A chaque étape de jeu, les agents choisissent leurs actions simultanément. Chaque agent i obtient alors une récompense $R_i(s, a)$ selon l'action jointe jouée. Le système passe à un nouvel état s' suivant la transition $P(s, a, s')$. Ainsi, dans chaque état on retrouve un jeu matriciel définissant les récompenses de chaque agent selon l'action jointe qui peut être jouée (figure 1.9).

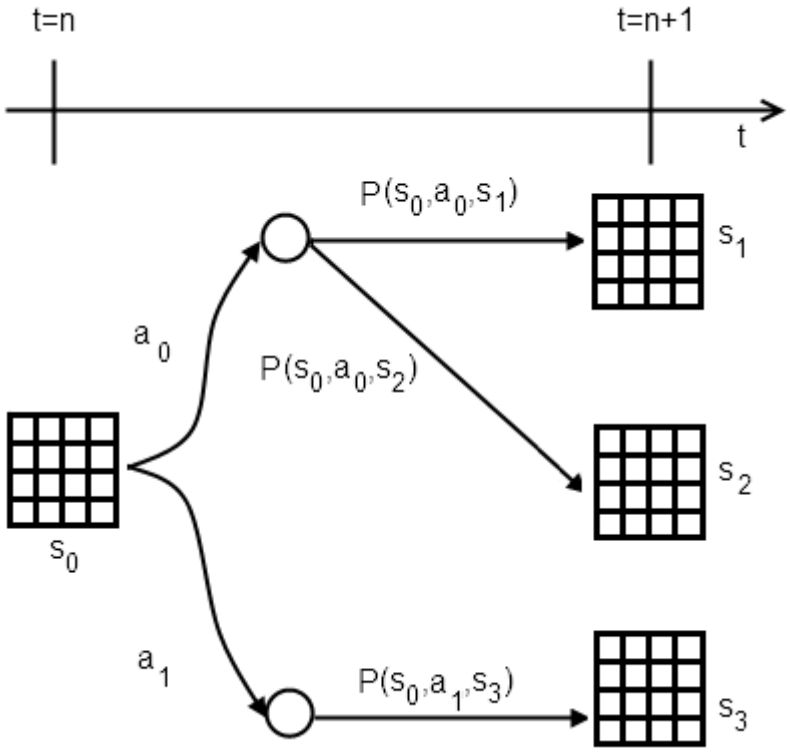


FIGURE 1.9 – Un jeu stochastique à deux joueurs : exemple de transition possible entre deux étapes, avec deux actions jointes a_0 et a_1 ; chaque état peut être vu comme un jeu matriciel définissant les récompenses de chaque agent.

1.4.2 Extensions multi-agents

Le formalisme des MDP a été conçu afin de modéliser le comportement d'un agent unique. Pour adapter ce modèle à un contexte multi-agents, plusieurs extensions ont été proposées.

Processus décisionnel de Markov multi-agents

Le modèle des MMDP (Multi-agent MDP) représente l'extension naturelle des MDP aux systèmes multi-agents. Ce modèle a été introduit par Boutilier [Boutilier, 1996, Boutilier, 1999] et aborde le problème de la coordination d'agents dans une situation de coopération. De ce fait, les agents ont tous la même récompense et perçoivent tous l'état complet du système.

Définition 1.11 *Un MMDP est défini par le tuple* $\langle \alpha, S, A, P, R, T \rangle$

- $\alpha = \{1, \ldots, n\}$ *: ensemble fini d'agents.*
- S *: ensemble fini d'états.*
- A *: ensemble fini des actions jointes des agents, avec* $A = \times_{i \in \alpha} A_i$*, où* A_i *est l'ensemble des actions individuelles de l'agent* i*.*
- P *: fonction de transition,* $S \times A \times S \rightarrow [0, 1]$*.*
- R *: fonction de récompense,* $S \times A \rightarrow \mathbb{R}$*.*
- T *: horizon de planification, c'est le nombre d'étapes d'actions.*

Un MMDP peut être vu comme un MDP ayant des espaces d'états et d'actions conséquents. L'ensemble des agents est alors considéré comme un seul agent, dont le but est de calculer une politique optimale pour le MDP joint. Résoudre un MMDP consiste à calculer une politique jointe $\pi = \langle \pi_1, \ldots, \pi_i, \ldots, \pi_n \rangle$ où π_i correspond à la politique individuelle de l'agent i. Pour ce faire, il suffit d'appliquer un algorithme classique tel que *Value-Iteration* pour obtenir π.

Les problèmes de coordination représentent la principale difficulté liée à l'utilisation d'un MMDP. Ce phénomène peut émerger lorsque chaque agent tente de résoudre le MMDP de manière indépendante, aboutissant à plusieurs politiques optimales distinctes pour l'ensemble des agents. Il existe cependant plusieurs approches pour résoudre ces problèmes de coordination. On peut, par exemple, considérer un ordre global sur les actions, ou utiliser un moyen de communication en vue de choisir une même politique jointe.

Processus décisionnels de Markov décentralisés

Les processus décisionnels de Markov décentralisés constituent une extension des MDP (sous classe des jeux stochastiques) pour les systèmes multi-agents où le contrôle est décentralisé. Ces formalismes permettent la modélisation de problèmes de décision multi-agents dans lesquels des agents doivent réaliser un ensemble de tâches en maximisant leur récompense globale.

Dans le cas où l'environnement est partiellement observable, c'est-à-dire que chaque agent doit décider de manière autonome quelle action exécuter tout en n'ayant qu'une vue partielle du système, Bernstein et al. [Bernstein *et al.*, 2000] ont proposé le formalisme des DEC-POMDP pour le contrôle décentralisé de plusieurs agents. Dans le cas où l'environnement est parfaitement observable, par l'union de toutes les observations des agents, on parle alors de DEC-MDP [Hansen, 2004]. Cependant, la complexité de la résolution des DEC-POMDP ainsi que des DEC-MDP, dans le cas général avec plus de deux agents, a été prouvée NEXP-complet. Du fait du caractère intraitable des DEC-POMDP, plusieurs sous-classes de complexité (souvent) inférieure, ont été proposées. Ainsi, de nombreux sous problèmes ont été identifiés permettant une résolution efficace et optimale de ceux-ci, voir la classification faite par Goldman et al. [Goldman et Zilberstein, 2004].

La complexité liée à la résolution de ces problèmes, a amené les chercheurs à développer des algorithmes approximatifs dont le but est de fournir une solution approchée, tout en garantissant si possible une certaine qualité de la solution. Pour une revue complète de la littérature sur ce type d'algorithmes, le lecteur intéressé est invité à lire [Seuken et Zilberstein, 2008].

1.4.3 Exemples

Les MDP et les jeux matriciels sont tous les deux inclus dans le cadre des jeux stochastiques, de ce fait nous avons déjà donné quelques exemples du modèle. Nous présentons également deux applications des jeux stochastiques, souvent rencontrés en littérature.

Grid Soccer

Littman [Littman, 1994] a proposé le jeu du *Grid Soccer* comme un exemple de jeu stochastique à somme nulle et à deux joueurs (figure 1.10). Le jeu se joue sur une grille de dimensions 4×5. Les joueurs A et B, occupent des cellules distinctes dans la grille et peuvent choisir entre cinq actions disponibles : $\{nord, sud, ouest, est, ne - rien - faire\}$. Les actions sont considérées comme stochastiques, c'est à dire que lorsque les agents choisissent de se déplacer, ils peuvent se trouver dans une direction non-prévue, avec une certaine distribution de probabilité. Lorsque l'agent possédant le ballon entre dans la zone *but* (à gauche pour l'agent A et à droite pour l'agent B), ce dernier obtient une récompense de $+1$ tandis que l'autre agent obtient une pénalité de -1 et le jeu est remis à l'état initial.

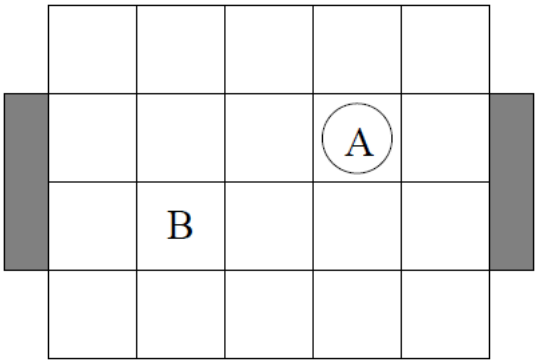


FIGURE 1.10 – Le jeu *Grid Soccer* de Littman.

Grid World

Un exemple d'un jeu stochastique à somme générale est le jeu *Grid World* de Hu et Wellman [Hu et Wellman, 2003] (figure 1.11). Les agents commencent dans deux coins différents de la grille et essayent d'atteindre leur place sur le côté opposé. Ils reçoivent une récompense de +1 pour une action qui amène à l'objectif et une récompense de 0 à défaut. Les actions possibles pour chaque agent sont : *nord*, *sud*, *ouest*, *est* et sont quasiment (avec une probabilité de 0.99) déterministes. Si les deux joueurs tentent d'occuper la même position, aucune transition ne sera faite.

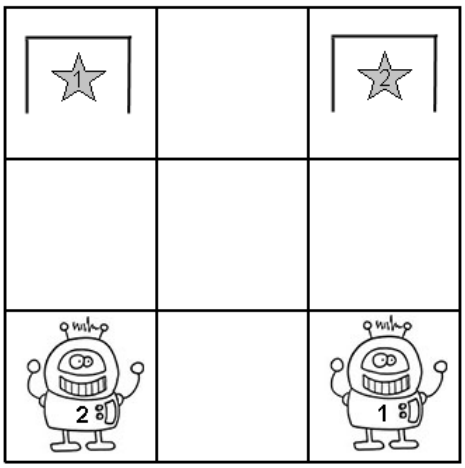


FIGURE 1.11 – Le jeu *Grid World* de Hu et Wellman.

Pour rendre le jeu intéressant et forcer les agents à interagir, l'action *nord* devient incertaine lorsqu'elle est exécutée à partir de la position initiale (avec une probabilité de 0.5). Ainsi, le chemin optimal pour chaque agent est de se déplacer latéralement à la

première étape du jeu, mais si les deux agents se déplacent latéralement en même temps, leurs actions échouent. Les agents doivent donc se coordonner entre eux afin d'atteindre leur but respectif sans se gêner.

1.4.4 Concepts de solution

Le but de chaque agent dans les jeux stochastiques est de calculer une politique qui maximise son gain à long terme.

Définition 1.12 *Une politique pour un agent i est une fonction qui définit, à partir d'un certain état s, une distribution de probabilité sur les différentes actions. Formellement,* $\pi : S \times A \to [0, 1]$*, avec :*

$$\forall s \in S \qquad \sum_{a \in A} \pi(s, a) = 1 \tag{1.21}$$

Nous adoptons les mêmes notations que celles des jeux matriciels, ainsi on note π_{-i} la politique jointe des autres agents exceptée celle de l'agent i. $\pi = \langle \pi_i, \pi_{-i} \rangle$ désigne la politique jointe de tous les agents, où l'agent i suit π_i et les autres suivent π_{-i}. Nous noterons aussi $\Delta(S, A_i)$ l'ensemble de toutes les distributions de probabilités qui recouvrent l'ensemble $S \times A_i$.

Afin de quantifier la valeur d'une politique, le même critère de performance que celui des MDP peut être utilisé, à savoir l'espérance sur les récompenses de l'agent à long terme. Ainsi, avec le critère γ-pondéré, la valeur d'une politique jointe pour l'agent i à l'état s, est :

$$V^{\pi}(s) \;=\; E^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{i,t} | s_0 = s\right] \tag{1.22}$$

avec $E^{\pi}[r_{i,t}|s_0 = s]$ l'espérance de gain que l'agent reçoit en étant à l'état s et à l'instant t, lorsque tous les agents suivent la politique jointe π. A noter que cette équation, est presque identique à l'équation 1.7 des MDP, excepté que chaque agent possède sa propre fonction de récompense et que toutes ces fonctions dépendent de la politique jointe π.

Equilibre de Nash

Les jeux stochastiques possèdent des fonctions de récompense qui peuvent différer d'un agent à un autre. Dans certains cas, il est difficile de trouver des politiques qui maximisent simultanément les critères de performance de tous les agents. C'est pourquoi, le concept d'équilibre est souvent recherché pour ce modèle. Cet équilibre est une situation, dans laquelle aucun agent ne peut améliorer son critère de performance, s'il est le seul à changer de politique : on retrouve ainsi la définition de l'équilibre de Nash.

Définition 1.13 *Une politique $\pi^* = \langle \pi_i^*, \pi_{-i}^* \rangle$ est un équilibre de Nash pour un jeu stochastique si :*

$$\forall s \in S, \forall i \in Ag, \forall \pi_i \in \Delta(S, A_i) \quad V_i^{\langle \pi_i^*, \pi_{-i}^* \rangle}(s) \geqslant V_i^{\langle \pi_i, \pi_{-i}^* \rangle}(s) \tag{1.23}$$

La figure 1.12 montre un exemple d'équilibre de Nash trouvé pour le jeu *Grid World*. La politique suivie est donc : $\langle nord, nord \rangle, \langle ouest, nord \rangle, \langle ouest, est \rangle, \langle nord, est \rangle$.

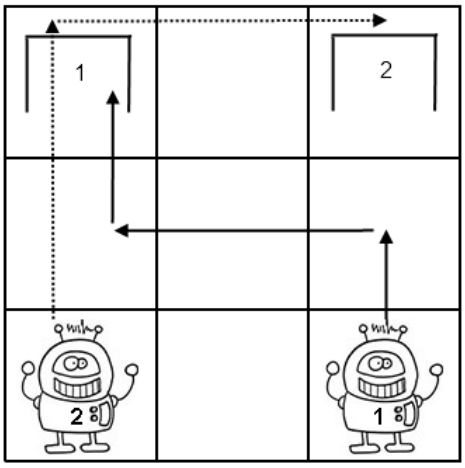


FIGURE 1.12 – Equilibre de Nash trouvé pour le jeu *Grid World*.

Ainsi que dans le cas des jeux matriciels, l'existence d'au moins un équilibre de Nash a été prouvée pour les jeux stochastiques [Shapley, 1953] à deux joueurs. Fink et Takahashi [Fink, 1964, Takahashi, 1964] ont généralisé ce résultat au cas d'un jeu stochastique escompté (avec le critère γ-pondéré) à N joueurs et en stratégies stationnaires [8].

Résoudre un jeu stochastique consiste à trouver des algorithmes permettant d'atteindre, pour chaque état, une position d'équilibre dans lequel aucun agent n'a intérêt à modifier son comportement en supposant qu'un tel équilibre existe. Cependant, calculer une politique pour le modèle des jeux stochastiques est considéré comme un problème difficile. La plupart des travaux menés sur la résolution des jeux stochastiques sont essentiellement axés sur les jeux à somme nulle. Déterminer une politique d'actions pour les jeux stochastiques à somme générale demeure complexe (calcul coûteux de l'équilibre de Nash, centralisation, explosion combinatoire, etc.). Peu de travaux de recherches ont été consacrés au domaine de la planification dans les jeux stochastiques, le reste des travaux se trouve dans le domaine de l'apprentissage par renforcement. Nous consacrons la totalité du chapitre suivant à l'analyse des algorithmes de résolution des jeux stochastiques et notamment aux travaux de Shapley [Shapley, 1953] et de Kearns [Kearns *et al.*, 2000].

8. Une stratégie est dite stationnaire, si et seulement si la règle de décision qui associe une action à un état dépend seulement de l'état courant.

1.5 Conclusion

Dans ce chapitre, nous avons présenté le cadre général dans lequel s'inscrivent nos travaux. Il est caractérisé par un ensemble d'agents pouvant (1) exécuter un ensemble fini d'actions à différents intervalles de temps, (2) percevoir l'état de leur environnement, et (3) obtenir un gain numérique à la suite de leurs actions. Ceci nous a amené à décrire la notion d'agent et de systèmes multi-agents. Nous avons présenté plusieurs modèles d'interaction entre agents, notamment les processus décisionnels de Markov et les jeux matriciels, qui ont été respectivement étudiés dans les domaines de la planification mono-agent et de la théorie des jeux.

Ainsi, nous avons commencé par introduire les concepts clés des processus décisionnels de Markov, comme la formulation des récompenses, la fonction de valeur, la politique optimale et les algorithmes de résolution. Un exemple d'application a été présenté dans le but de montrer l'adéquation du modèle à la résolution de problèmes décisionnels mono-agent.

Nous avons présenté ensuite, un aperçu des concepts clés des jeux matriciels, comme les notions de gain espéré, de Meilleure Réponse et d'équilibre de Nash. Différents jeux matriciels ont été exposés dont le but était d'expliquer les notions précédentes et de mettre en valeur le problème de la prise de décision collective.

Enfin, nous avons montré comment le modèle des jeux stochastiques englobe les deux cadres précédents, en combinant le domaine de la planification mono-agent et le domaine de l'interaction stratégique entre plusieurs agents. Dans le prochain chapitre, nous continuons l'exploration du contexte de cette thèse, en examinant les principaux algorithmes existants pour la résolution des jeux stochastiques.

Chapitre 2

Algorithmes de résolution pour les jeux stochastiques

Sommaire

2.1 Algorithme de Shapley 40
2.2 Algorithme de Kearns et al. 42
2.3 Autres travaux 45
2.3.1 L'algorithme InfiniteVI de Kearns et al. 46
2.3.2 L'algorithme FiniteVICOM de Burkov et al. 48
2.3.3 Algorithme de Pollatschek et Avi-Itzhak 50
2.3.4 Algorithmes issus des techniques d'apprentissage 50
2.4 Implémentation de l'équilibre de Nash 54
2.5 Limitations 56
2.6 Conclusion 57

Dans ce chapitre nous présentons une vue d'ensemble des algorithmes existants pour la résolution des jeux stochastiques. Ces algorithmes sont peu nombreux dans le domaine de la planification et la plupart des travaux sont réalisés avec les techniques d'apprentissage. Les algorithmes de résolution sont souvent composés de deux parties, (1) une partie de résolution MDP et (2) une partie de calcul d'équilibre. Ces algorithmes font différentes hypothèses sur l'environnement et les capacités des joueurs, et ils sont capables de résoudre différents types de jeux stochastiques. La caractéristique commune de ces approches, se résume à leur capacité à résoudre les problèmes de décision multi-agents d'une manière algorithmique.

Introduction

Les algorithmes de résolution pour les jeux stochastiques remontent aux travaux fondateurs de Shapley [Shapley, 1953], présentant une approche efficace pour résoudre les jeux stochastiques à somme nulle. Comme nous l'avons vu dans le chapitre précédent, il existe deux philosophies différentes pour résoudre les problèmes de décision dynamiques : soit en planifiant ou en apprenant.

Pour avoir une vision claire de ce que représente la planification, nous adoptons les deux définitions suivantes :

Définition 2.1 *Selon Russell et Norvig [Russell et Norvig, 2003], la planification est la tâche qui consiste à fournir une séquence d'actions pour résoudre un problème.*

Définition 2.2 *Pour Ghallab et al. [Ghallab* et al.*, 2004], planifier c'est raisonner avant d'agir. C'est un processus de délibération explicite et abstrait qui choisit et organise les actions en anticipant leurs résultats escomptés.*

La planification dite classique [Blum et Furst, 1995, Fikes et Nilsson, 1971] permet de répondre à un certain nombre de problèmes mono-agent. Cependant, il existe des classes de problèmes qui ne peuvent être résolues à l'aide de ces planificateurs. En effet, supposons qu'il s'agit de gérer plusieurs agents, possédant leurs propres buts. Si chaque agent planifie sans prendre en compte les interactions possibles avec les autres agents, il peut s'en suivre des conflits lors de l'exécution des plans calculés. Ainsi dans une planification multi-agents [Georgeff *et al.*, 1987a, Georgeff *et al.*, 1987b, Durfee, 2001, Weerdt *et al.*, 2005], les dépendances entre les agents doivent être résolues et les agents doivent se coordonner.

Lorsque nous parlons de planification dans les jeux stochastiques, nous supposons que les agents ont toutes les informations nécessaires pour prendre une décision. Le seul

problème est de savoir comment exploiter ces informations afin de trouver la *meilleure* stratégie qui constituera une partie de la solution.

L'approche par apprentissage suppose souvent que les agents ne possèdent qu'une connaissance partielle de leur environnement. Par conséquent, l'agent a besoin d'explorer l'environnement d'une certaine manière. L'exploration est censée aider l'agent à remplir la partie manquante de l'information, de manière à lui permettre de trouver une solution (voir apprentissage par renforcement [Watkins, 1989, Sutton, 1988]).

Rappelons, que les jeux stochastiques sont des jeux multi-états qui peuvent être vus comme une généralisation des MDP pour les systèmes multi-agents. Ces jeux, ne possèdent pas une solution optimale au sens des MDP, c'est à dire une solution calculée indépendamment des autres agents. Au lieu de cela, en résolvant ce modèle par des agents rationnels, les chercheurs se penchent généralement sur deux questions principales :

- Pour un agent donné, quelle est la meilleure stratégie de réponse face aux stratégies des autres agents (en supposant une distribution de probabilité sur leurs actions) ?
- Quelles sont les situations d'équilibre dans un jeu donné ?

Le concept le plus largement accepté, comme solution d'un jeu stochastique, est l'équilibre de Nash. Contrairement aux MDP, où la solution optimale est unique (ou alors il y a plusieurs solutions qui donnent à l'agent la même espérance de gain à long terme), l'équilibre de Nash n'est pas toujours unique. Le fait d'avoir des équilibres multiples peut conduire à des problèmes de coordination. En effet si les agents jouent différents équilibres, la stratégie jointe jouée peut ne pas constituer un équilibre. Selon les jeux, cela peut être risqué, surtout si les valeurs des utilités varient considérablement d'une stratégie à une autre.

Les algorithmes que nous allons étudier en planification, sont issus des techniques de programmation dynamique et plus particulièrement de l'algorithme Value-Iteration. Deux algorithmes s'imposent :

- L'algorithme de Shapley,
- L'algorithme de Kearns et al.

Ces algorithmes ont été conçus pour résoudre en planification des jeux avec certaines spécificités (la fonction de transition P et la fonction de récompense R sont connues). Chaque algorithme possède des avantages et inconvénients, dont nous allons discuter dans la suite.

Ce chapitre se divise en cinq sections. Les deux premières introduiront les deux travaux fondateurs dans le domaine de la résolution des jeux stochastiques en planification, à savoir les algorithmes de Shapley et de Kearns et al. La troisième section, présentera également

d'autres travaux pour la résolution. Ces travaux viennent compléter ceux déjà réalisés et apportent un éclairage nouveau sur le domaine. Notre étude portera également sur des méthodes de résolution en apprentissage, susceptibles d'aider à la compréhension du modèle des jeux stochastiques et les interactions inter-agents pouvant avoir lieu durant les simulations. La quatrième section, étudiera de manière approfondie, les méthodes de calcul de l'équilibre de Nash ainsi que la complexité correspondante. La cinquième et dernière section, présentera d'une manière générale les limitations et restrictions dont souffrent les algorithmes de résolution, les empêchant ainsi de s'appliquer à plus grande échelle.

2.1 Algorithme de Shapley

L'algorithme de Shapley [Shapley, 1953] calcule un équilibre de Nash pour les jeux stochastiques à somme nulle, pondérés et à deux agents. Cet algorithme est une extension de l'algorithme Value-Iteration mono-agent (vu dans le chapitre précédent) pour le modèle multi-agents des jeux stochastiques. Il cherche un équilibre de Nash pour chaque matrice $M(s)$ correspondant à la paire état-itération. Chaque jeu matriciel est construit en utilisant une fonction de valeur reflétant :

– l'utilité immédiate de choisir une action,

– l'utilité à long terme en suivant un équilibre à partir de l'état suivant.

Dans un premier temps, l'algorithme de Shapley (algorithme 3) commence par assigner des valeurs initiales (arbitraires) à $V(s)$. Ensuite, à chaque itération t une mise à jour est appliquée à ces valeurs, pour cela la matrice du jeu $M(s)$ doit être construite à partir des Q-valeurs correspondant à l'état s.

Définition 2.3 *Une fonction $Q : S \times A$, est une fonction qui attribue à chaque couple état-action (s, a) une Q-valeur $Q(s, a)$, correspondant à la récompense espérée en effectuant l'action a partant de l'état s et en suivant une politique π :*

$$Q^{\pi}(s, a) = R(s, a) + \sum_{s' \in S} P(s, a, s') \; V^{\pi}(s') \tag{2.1}$$

Rappelons que S est l'espace d'états, R est la fonction de récompense et V est la fonction de valeur (cf. chapitre 1.2). Par exemple, supposons qu'il y a deux agents et que chacun possède deux actions : A et B. Dans ce cas, la matrice de paiement $M(s)$ du premier agent, relative à l'état s et à l'itération t, ressemblerait à la matrice de la figure 2.1.

Notons que l'algorithme de Shapley fonctionne de façon similaire à l'algorithme d'itération sur les valeurs pour les MDP, avec l'opérateur *max* remplacé par l'opérateur d'Utilité

Algorithme 3: Algorithme de Shapley pour la résolution des jeux stochastiques à somme nulle, escomptés et à deux joueurs.

Entrées : Un jeu stochastique à somme nulle, $\gamma \in [0,1]$, $\epsilon \geq 0$

Sorties : Une politique d'actions π

1 **Initialisation**

2 t $\leftarrow 0$

3 V(s) $\leftarrow$ valeur arbitraire

4 **répéter**

5 **pour** $s \in S$ **faire**

6 **pour** $a \in A$ **faire**

7 $M(s,t) \leftarrow \{Q(s,a) : Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a,s')\, V(M(s', t-1))\}$

8 $V(M(s,t)) = Utilité\ (M(s,t))$

9 t $\leftarrow$ t+1

10 **jusqu'à** $|V(s,t) - V(s,t-1)| < \epsilon,\ \forall s \in S$;

11 **pour** $s \in S$ **faire**

12 **pour** $a \in A$ **faire**

13 Construire la matrice $M(s)$

14 $\pi(s,t) = Stratégie\ (M(s))$

(ligne 8 de l'algorithme 3). L'opérateur d'utilité résout un jeu matriciel (calcule un équilibre de Nash) à somme nulle et retourne l'unique utilité correspondant à la solution de ce jeu. Pour trouver un équilibre dans un jeu matriciel à somme nulle, l'opérateur d'utilité peut par exemple exploiter l'algorithme *Minimax*[9].

L'opérateur *Stratégie* (ligne 14 de l'algorithme 3) fonctionne de manière similaire à l'opérateur d'*Utilité* (ligne 8 de l'algorithme 3) avec la seule différence qu'il retourne un ensemble d'actions mixtes correspondant à la solution du jeu matriciel $M(s)$, et non pas un profil de gain. La condition d'arrêt de l'algorithme est le seuil ϵ fixé *a priori*.

La définition complète de l'algorithme de Shapley est donnée par l'algorithme 3. Dans cet algorithme, la variable $V(s)$ reflète le gain espéré à long terme des deux agents, à l'état s, calculé à l'itération t et correspondant à la solution du jeu matriciel $M(s)$.

9. L'algorithme *Minimax* [Harsanyi, 1973] est un algorithme qui s'applique à des jeux à deux joueurs et à somme nulle. Pour une vaste famille de jeux, le théorème du minimax de Von Neumann [Osborne et Rubinstein, 1994] assure l'existence d'un tel algorithme. Le meilleur choix est celui qui minimise les pertes du joueur tout en supposant que l'adversaire cherche au contraire à les maximiser.

		$Agent_2$	
		A	B
$Agent_1$	A	$Q(s, \langle A, A \rangle)$	$Q(s, \langle A, B \rangle)$
	B	$Q(s, \langle B, A \rangle)$	$Q(s, \langle B, B \rangle)$

FIGURE 2.1 – La matrice de paiement $M(s)$ du premier agent, relative à l'état s et construite à partir des Q-valeurs individuelles.

Shapley a prouvé [Shapley, 1953] que son algorithme est assuré de trouver un équilibre dans n'importe quel jeu stochastique à deux agents et à somme nulle (cf. théorème 2.1).

Théorème 2.1 *Tout jeu stochastique, à somme nulle, escompté et à horizon infini possède une valeur unique. Cette valeur est donnée par la séquence des valeurs uniques des jeux matriciels $M(s)$. Chaque joueur possède une stratégie optimale qui utilise les stratégies Minimax mixtes de chaque jeu $M(s)$.*

Concernant la complexité de l'algorithme, nous savons que l'algorithme de Shapley s'appuie sur la technique d'itération sur les valeurs et utilise l'algorithme *Minimax* pour trouver un équilibre de Nash pour chaque paire état-itération. Puisque le temps d'exécution de l'algorithme Value-Iteration est polynomial en la taille de l'espace d'états, alors le temps d'exécution de l'algorithme de Shapley l'est aussi mais en fonction du nombre d'agents et du nombre d'états du jeu stochastique. La principale limitation de cet algorithme est qu'il ne convient qu'à des jeux à somme nulle et par conséquent à deux agents, une sous-classe restreinte des jeux stochastiques. Par exemple, un système incluant plusieurs agents coopératifs ne peut pas être résolu avec l'algorithme de Shapley.

2.2 Algorithme de Kearns et al.

Kearns et al. [Kearns *et al.*, 2000] ont décidé d'aller plus loin en proposant un algorithme d'itération sur les valeurs pour résoudre n'importe quel jeu stochastique à somme générale. Cependant, leur algorithme nommé FiniteVI, a été conçu pour résoudre seulement les jeux stochastiques avec un horizon fini. L'algorithme reprend les mêmes principes que l'algorithme précédent et prend en compte un contexte multi-agents plus général.

Du fait que l'horizon est fini, FiniteVI considère seulement les T prochaines étapes pour le calcul du gain à long terme. Chaque paire état-itération (s, a) a une Q-valeur qui lui est associée. Nous désignons par $Q_i(s, a)$ le gain à long terme de l'agent i en jouant l'action a (conjointement avec les autres agents) à partir de l'état s.

A chaque itération t de l'algorithme, toutes les Q-valeurs seront mises-à-jour en utilisant l'équation suivante :

$$Q_i(s,a,t) = R_i(s,a) + \sum_{s' \in S} P(s,a,s')\ V_i^f(s', t-1) \tag{2.2}$$

où $V_i^f(s',t)$ désigne le gain de l'agent i, cumulé à l'horizon t et extrait d'un certain équilibre de Nash, lui-même calculé par une fonction de sélection d'équilibre, nommée f.

Pour calculer le gain à long terme V_i^f, la fonction de sélection d'équilibre f construit un jeu matriciel $M(s)$ à partir des Q-valeurs calculées à l'état s et à l'itération t, c'est-à-dire à partir des $Q_i(s,a,t)\ \forall a \in A,\ \forall i \in Ag$ (un jeu similaire est construit à la ligne 7 de l'algorithme 3).

La fonction de sélection d'équilibre n'a pas été clairement définie, mais elle peut résoudre les jeux matriciels en utilisant n'importe quelle méthode de calcul d'équilibre [Lemke et Howson, 1964, McKelvey et McLennan, 1996, Stengel, 2002, Porter *et al.*, 2004, Chen *et al.*, 2006].

L'algorithme prend donc comme entrée un jeu stochastique et une fonction de sélection d'équilibre f. La condition d'arrêt est désormais l'horizon T (contrairement à l'algorithme précédent qui s'arrête en fonction du paramètre ϵ). La définition précise de FiniteVI est donnée par l'algorithme 4.

Notons que la fonction de sélection d'équilibre f de l'algorithme FiniteVI est analogue à l'opérateur *Stratégie* de l'algorithme de Shapley. Parallèlement, V_i^f dans FiniteVI est analogue à l'opérateur *Utilité* de l'algorithme de Shapley.

Nous supposons que la fonction f construit le jeu matriciel $M(s)$ et le résout. Résoudre $M(s)$ consiste à trouver un équilibre pour le jeu. Or, cet équilibre n'est pas forcément unique car le jeu est à somme générale. Si le jeu présente plusieurs équilibres, la fonction f doit en choisir un et le communiquer à tous les agents.

Théorème 2.2 *Soit un jeu stochastique et soit f une fonction de sélection d'équilibre de Nash. La paire de politiques $\langle \pi_1, \pi_2 \rangle$ de sortie est un équilibre de Nash pour les T-étapes du jeu stochastique et ce à partir de n'importe quel état de départ.*

La fonction f peut être considérée comme un "oracle" [10] :

– dans un premier lieu, en ne définissant pas explicitement la manière dont l'équilibre de Nash doit être calculé, ou choisi en cas de multiplicité.

10. Oracle : fonction capable de résoudre un problème de décision en une seule opération. Ce problème peut être de n'importe quelle complexité. Les oracles sont des outils théoriques, puisque ce modèle évite soigneusement de soulever la question de leur applicabilité.

– dans un deuxième lieu, en n'indiquant pas la façon dont l'unique équilibre choisi, doit être distribué sur les agents.

Algorithme 4: Algorithme FiniteVI pour la résolution des jeux stochastiques à somme générale, non pondérés, à horizon fini et à deux joueurs.

Entrées : Un jeu stochastique, une fonction de sélection f, un horizon T
Sorties : le couple de politiques (π_1, π_2)

Initialisation
pour $s \in S$ **faire**
  pour $k \in \{1, 2\}$ **faire**
    $Q_k[s, 0] \leftarrow M_k[s]$
    $\pi_k(s, 0) \leftarrow f_k(M_1[s], M_2[s])$
Itération sur l'horizon T
$t \leftarrow 0$
tant que $t \leqslant T$ **faire**
  pour $s \in S$ **faire**
    pour $k \in \{1, 2\}$ **faire**
      pour $a \in A$ **faire**
        $Q_k[s,t](a) \leftarrow M_k[s](a) + \sum_{s' \in S} P(s'|s,a)\ V_k^f(Q_1[s', t-1], Q_2[s', t-1])$
    $\pi_k(s,t) = f_k(Q_1(s,t), Q_2(s,t))$
  $t \leftarrow t + 1$

Selon notre point de vue, l'exigence d'un tel oracle représente un sérieux obstacle [Chang *et al.*, 1997], limitant l'applicabilité de l'algorithme FiniteVI dans les systèmes distribués.

Concernant la complexité, selon Kearns et al. le temps d'exécution de l'algorithme FiniteVI est quadratique en la taille de l'espace d'états, en supposant que le temps d'exécution de la fonction f est unitaire. Cependant, comme on ne connaît pas d'algorithme polynomial [11] trouvant un équilibre de Nash dans un jeu matriciel à somme générale, nous admettons alors que le temps d'exécution de l'algorithme FiniteVI, utilisant un des algorithmes connus permettant de calculer un équilibre de Nash, ne peut pas être polynomial.

11. Nous détaillons ce propos à la section 5 de ce chapitre.

Tests expérimentaux

Nous avons implémenté l'algorithme FiniteVI sur le jeu de Hu et Wellman, présenté dans le chapitre 1. Nous rappelons que le jeu est constitué de deux agents, qui se déplacent dans une grille de dimension 3×3. Chaque robot doit rejoindre la case "but" dans le coin opposé de la grille. Les actions disponibles pour chaque agent sont :

$$A_i = \{Gauche, Droite, Haut, Bas, Ne - rien - faire\}$$

La fonction de récompense est définie par les valeurs suivantes :

- $+100$: ce gain est obtenu par l'agent pour toute action menant à la cellule but,
- -1 : cette pénalité est obtenue en cas de collision avec un autre robot,
- 0 : est obtenue dans tous les autres cas.

La fonction de transition dans ce jeu est considérée comme déterministe. Cela signifie que lorsque le robot exécute une action, supposons *Droite*, alors il se déplace à la cellule de droite (si la transition est possible) sinon il reste sur place (si la transition est impossible) avec une probabilité de 1.

Il est facile de vérifier que lorsque la fonction de transition est déterministe, ce jeu stochastique possède dix équilibres (ou trajectoires) pour l'horizon $T = 5$. Pratiquement, il n'y a que cinq équilibres, les cinq autres équilibres sont obtenus par symétrie (figure 2.2).

Nous avons remarqué que lorsque l'horizon de planification est supérieur à 5, le nombre et la configuration des trajectoires du robot (de la position de départ jusqu'au but) ne changent pas. Cependant, la politique d'action contiendra dans ces cas, pour chaque robot, des actions supplémentaires $Ne-rien-faire$ à exécuter à partir de la cellule but. D'autre part, lorsque l'horizon est inférieur à 5, la politique d'actions de chaque robot pointera toujours vers des actions de type $Ne - rien - faire$.

2.3 Autres travaux

Dans la section précédente, nous avons présenté les deux travaux fondateurs dans le domaine de la résolution des jeux stochastiques par planification. Néanmoins, il existe d'autres travaux connexes, que nous introduisons dans cette section. Ces algorithmes sont en quelque sorte des extensions des algorithmes de Shapley et de Kearns et al.

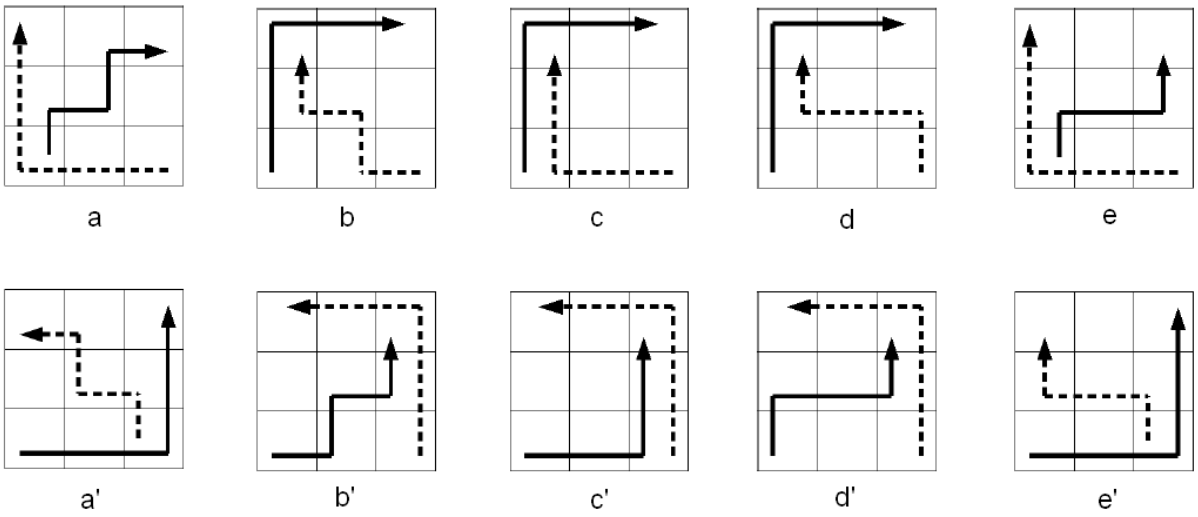


FIGURE 2.2 – l'ensemble des équilibres de Nash du jeu *Grid-World* de Hu et Wellman, calculés par l'algorithme FiniteVI.

2.3.1 L'algorithme InfiniteVI de Kearns et al.

Avec l'algorithme InfiniteVI, les auteurs [Kearns *et al.*, 2000] voulaient aller plus loin et généraliser la résolution au cas des jeux stochastiques à somme générale et à horizon infini. InfiniteVI est très similaire à l'algorithme FiniteVI, à l'exception de la politique qui n'est plus dépendante de l'horizon T et du facteur de pondération qui est désormais inclus dans le calcul de la matrice. Mais la différence majeure de cet algorithme, réside dans le fait que la fonction de sélection d'équilibre f, ne sélectionne plus un équilibre de Nash mais un équilibre basé sur les niveaux de sécurité.

Définition 2.4 *Le niveau de sécurité pour un agent donné, est un paiement (gain) qu'il peut se garantir indépendamment de ce que jouent les autres agents.*

Supposons l'existence de deux agents $agent_1$ et $agent_2$. Le premier agent $agent_1$ voulant maximiser ses gains, il commence alors à chercher *le pire état* qui peut lui arriver quand il joue chacune de ses actions (c'est-à-dire, le gain minimum possible pour chaque stratégie, appelé niveau de sécurité). Ensuite, il choisit l'action qui lui rapporte le maximum de ces gains minima. C'est ce qu'on appelle un *Maximin*, $agent_1$ peut donc se garantir un certain gain, indépendamment de ce que $agent_2$ peut jouer.

Définition 2.5 *Pour un agent i, l'action Maximin est définie par : $argmax_{s_i} min_{s_{-i}}\ V_i(s_i, s_{-i})$, tandis que la valeur Maximin est donnée par : $max_{s_i} min_{s_{-i}}\ V_i(s_i, s_{-i})$.*

Pour les jeux à somme nulle, l'équilibre de Nash coïncide avec celui basé sur les niveaux de sécurité, et ainsi l'utilité du jeu pour un agent donné correspond au *Maximin* calculé. Cependant, pour les jeux à somme générale, les couples de valeurs calculés en utilisant les niveaux de sécurité peuvent ne pas être des équilibres de Nash et ainsi fournir des gains moins importants.

Considérons le jeu matriciel de la figure 2.3. Si les deux agents appliquent le concept de *Maximin*, alors ils choisiront respectivement les actions A et B, ce qui correspond au paiement $(10, 1)$ et non pas au paiement $(1, 10)$ lié à l'équilibre de Nash du jeu.

		$Agent_2$	
		A	B
$Agent_1$	A	(1,10)	(10,1)
	B	(0,-10)	(0,-9)

FIGURE 2.3 – Un jeu matriciel à somme générale. Le couple de valeurs $(1, 10)$ est trouvé en calculant un équilibre de Nash, tandis que le couple $(10, 1)$ est trouvé en appliquant la technique de *Maximin*.

Las auteurs ont prouvé, grâce au théorème suivant, que l'algorithme InfiniteVI peut toujours converger avec un horizon infini.

Théorème 2.3 *Soit un jeu stochastique quelconque et soit f une fonction quelconque de calcul de niveaux de sécurité. Alors, lorsque $T \to \infty$, la politique $\pi(s)$ calculée par InfiniteVI, converge vers des équilibres basés sur les niveaux de sécurité, pour le jeu stochastique pondéré et à partir de n'importe quel état initial.*

Par la même occasion, Kearns et al. se sont posés la question de l'existence d'un algorithme de planification pour les jeux stochastiques à somme générale, à horizon infini et basé sur les équilibres de Nash. Ils ont prouvé, par un contre exemple, que même si ce genre d'algorithme existe, il peut néanmoins ne pas converger dans tous les cas et ce indépendamment de la fonction de sélection d'équilibre appliquée.

Tests expérimentaux

Nous avons testé l'algorithme InfiniteVI, sur le "toy problem" *Grid-World* de Hu et Wellman. Les expériences confirment que l'algorithme converge toujours en un nombre fini d'itérations vers une politique basée sur les niveaux de sécurité.

Cependant, les simulations montrent que les agents ont tendance à ne pas se déplacer de leurs positions, indépendamment de l'état dans lequel ils sont. Une analyse approfondie

des matrices de paiement, a révélé qu'elles ne contiennent que des valeurs nulles, ce qui explique parfaitement le comportement des agents. En effet, le comportement pessimiste adopté pas les agents, éclipse les récompenses à long terme et les empêchent ainsi d'avoir un plan d'actions vers l'état but.

2.3.2 L'algorithme FiniteVICOM de Burkov et al.

Dans la continuité des travaux précédents, Burkov et al. [Burkov et Chaib-Draa, 2008] ont proposé un algorithme de planification pour la résolution des jeux stochastiques à somme générale et à horizon fini. Afin d'éviter la centralisation impliquée par la fonction f et d'ajouter les propriétés d'un système distribué, les auteurs ont ajouté une composante de communication, basée sur la sélection d'équilibre.

Au lieu d'utiliser une fonction f centralisée, sélectionnant à chaque état un équilibre unique pour tous les agents, les auteurs ont proposé un processus de sélection d'équilibre en deux phases.

La première phase est une phase de "calcul d'équilibre", où chaque agent calcule un ensemble d'équilibres (pour chaque paire état-itération) en utilisant n'importe quel algorithme connu de recherche d'équilibre (pas nécessairement le même pour tous les agents). L'ensemble de ces algorithmes est noté $C = (C_1, \ldots, C_{|Ag|})$.

La deuxième phase est une phase de "communication". Durant cette phase, les agents communiquent entre eux afin d'éventuellement partager leurs équilibres calculés et de sélectionner un équilibre unique parmi ceux calculés. Selon leur approche, la phase de communication a lieu sous la forme d'un jeu matriciel. Ce nouveau jeu, qu'ils ont appelé "jeu de communication", est construit dynamiquement à partir des équilibres calculés par les agents lors de la phase de calcul d'équilibre. L'ensemble des algorithmes permettant la communication et le choix de l'équilibre global est noté $P = (P_1, \ldots, P_{|Ag|})$.

Le principe de l'algorithme FiniteVICOM est relativement simple (algorithme 5) :

- A chaque état s et à chaque itération t, l'algorithme doit sélectionner un équilibre unique. Tout d'abord, chaque agent i exploite son algorithme C_i pour calculer l'ensemble des équilibres du jeu matriciel $M(s)$, qui a été construit à partir des Q-valeurs de tous les agents.
- Puis, durant la phase de communication, les agents utilisent leur algorithme respectif P_i pour jouer au jeu de communication $CM(S)$. Dans ce jeu de communication, les actions disponibles correspondent à l'ensemble des équilibres calculés ε_i pour la matrice $M(s)$.

– Finalement, chaque agent communique aux autres un équilibre ε_i et observe leurs actions de communication. Si tous les agents jouent le même équilibre que celui joué par l'agent i, alors ce dernier obtient une certaine récompense (c'est l'utilité respective à l'action de communication jouée). Dans tous les autres cas, les agents obtiennent une récompense nulle.

Burkov et al. ont montré par des expériences que leur approche, une fois adoptée par tous les agents, facilite le calcul distribué et conduit à la sélection d'un équilibre unique pour les jeux stochastiques à somme générale avec communication.

Les auteurs supposent que la communication entre les agents est toujours disponible, fiable et gratuite. Ces trois conditions représentent des fortes hypothèses sur le déploiement de la solution, d'autant plus qu'aucun protocole et/ou support de communication n'ont été définis.

Algorithme 5: Algorithme FiniteVICOM pour la résolution des jeux stochastiques à somme générale, à horizon fini et avec communication.

Entrées : Un jeu stochastique, un horizon T, un ensemble d'algorithmes de calcul d'équilibre C, un ensemble d'algorithmes pour le choix d'équilibre

Sorties : $\pi = (\pi_1, \ldots, \pi_{|Ag|})$

$t \leftarrow 0$
tant que $t \leqslant T$ **faire**
  pour $s \in S$ **faire**
    pour $i \in Ag$ **faire**
      pour $a \in A$ **faire**
        Initialisation
        si *t=0* **alors**
          $Q_i(s,a,t) \leftarrow R_i(s,a)$
        sinon
          $Q_i(s,a,t) \leftarrow R_i(s,a) + \sum_{s' \in S} P(s,a,s')\, V_i^{\varepsilon}(s')$
      $\varepsilon_i \leftarrow C_i(s,t)$
    $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{|Ag|})$
    $\pi_i(s,t) \leftarrow Play(P,s,t,\varepsilon)$
  t=t+1

2.3.3 Algorithme de Pollatschek et Avi-Itzhak

Tout comme l'algorithme de Shapley, qui est une extension de l'itération sur les valeurs, Pollatschek et Avi-Itzhak [Pollatschek et Avi-Itzhak, 1969] ont proposé une extension de l'algorithme Policy-Iteration (Howard, 1960) aux jeux stochastiques à somme nulle et à horizon infini. Cet algorithme calcule la fonction de valeur de chaque état, à partir de laquelle une politique est dérivée. Cependant, l'algorithme n'est garanti de converger que lorsque la fonction de transition et le facteur de pondération γ satisfont certaines propriétés. Particulièrement, il faut que $\gamma < 1/3$ ou :

$$max_{s \in S} \sum_{s' \in S} [max_{a \in A}\ P(s,a,s') - min_{a \in A}\ P(s,a,s')] < (1-\gamma)/\gamma \tag{2.3}$$

Cela équivaut à limiter l'effet des actions jointes des agents sur la fonction de transition. Breton [Breton, 1986] dans sa thèse, a effectué une analyse empirique de cet algorithme et d'autres algorithmes issus de l'itération sur les valeurs. Avec 15 états et 15 actions par état, Breton a observé que l'algorithme de Pollatschek et Avi-Itzhak, convergeait plus vite que celui de Shapley. Un algorithme similaire a été proposé par Hoffman et Karp [Hoffman et Karp, 1966], en utilisant une technique d'apprentissage de politique, permettant ainsi d'éviter le problème de convergence.

2.3.4 Algorithmes issus des techniques d'apprentissage

Bien que notre étude ait porté jusque-là seulement sur les techniques de planification, nous portons aussi un intérêt aux méthodes d'apprentissage. En effet, les deux techniques de résolution partagent plusieurs points communs, comme la recherche d'équilibre et la coordination entre les agents. Nous pouvons classer les techniques d'apprentissage en trois classes :

– **Classe 1 : associant l'apprentissage par renforcement et des méthodes issues de la théorie des jeux**

  Les agents effectuent un apprentissage par renforcement (Q-learning [Sutton et Barto, 1998]), en exécutant simultanément des actions et en recevant des récompenses. Après chaque étape du jeu, les agents mettent à jour leurs fonctions Q qui associent des valeurs à chaque couple état-action :

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha\Big(R_i(s,a)\ +\ \gamma\ Valeur_i(s') - Q_i(s,a)\Big) \tag{2.4}$$

  où $\alpha \in [0,1]$ est un coefficient dénommé le taux d'apprentissage. La signification de la fonction $Valeur_i$ est la suivante :

- Pour l'algorithme Minimax-Q de Littman [Littman, 1994], conçu pour des jeux à deux agents et à somme nulle, la fonction $Valeur_i$ retourne à l'agent i la valeur $Minimax$ de chaque jeu (construit à partir des Q-valeurs des agents). La politique suivie par les agents est alors la politique *Minimax* en stratégies mixtes.
- Pour l'algorithme Nash-Q de Hu et Wellman [Hu et Wellman, 2003], conçu pour des jeux à somme générale et à plusieurs agents, la fonction $Valeur_i$ retourne la valeur de l'équilibre pour l'agent i. Les auteurs ont également proposé une politique de coordination permettant de choisir un équilibre unique pour tous les agents, par exemple sélectionner toujours le deuxième équilibre du jeu (l'algorithme Lemke-Howson [Lemke et Howson, 1964] retourne une liste d'équilibres triée). La politique suivie par les agents dans ce cas est alors la politique correspondant à l'équilibre de Nash choisi à chaque étape du jeu.

A noter que, que dans les deux cas, la fonction $Valeur_i$ requiert que l'agent i puisse observer toutes les actions et les récompenses des autres agents.
Vu que Minimax-Q et Nash-Q possèdent la même structure algorithmique, nous présentons alors un seul algorithme générique (algorithme 6) pour les deux, dans lequel il faut prendre en considération la spécificité de chaque algorithme (au niveau des fonctions $Equilibre_i$ et $Valeur_i$).

- **Classe 2 : associant l'apprentissage et des méthodes de modélisation d'autrui**

La modélisation de l'adversaire est une technique largement utilisée dans la théorie des jeux pour rendre chaque joueur capable de s'adapter à ses adversaires et à leurs éventuels changements de comportement. Les premiers travaux dans ce domaine sont dus à Brown [Robinson, 1951, Brown, 1951] et se sont réalisés dans le cadre des jeux répétés.

La technique utilisée sous le nom de *Fictitious play* ou de jeu fictif, permet à un agent d'estimer la stratégie jouée par son adversaire afin de jouer la meilleure réponse à cette estimation. De ce fait, les agents doivent maintenir des croyances sur les stratégies suivies par les autres agents π_{-i}. Le modèle du jeu fictif suppose que chaque agent i choisit ses actions à chaque période pour maximiser son utilité espérée $V_i^{\pi_{-i}}$, étant donnée l'estimation des politiques de ses adversaires.

Afin d'exprimer cette estimation, on suppose que l'agent i possède, pour chaque agent j, une fonction de poids $c_{ij} : A_j \rightarrow \mathbb{R}_+$. Ce poids est mis à jour en y ajoutant $+1$ à chaque fois qu'une stratégie adverse a_j est jouée.

Algorithme 6: Algorithme générique pour la résolution par apprentissage des jeux stochastiques.

Entrées : Un jeu stochastique
Sorties : π

Initialisation
pour $s \in S$ **faire**
  pour $i \in Ag$ **faire**
    pour $a \in A$ **faire**
      $Q_i(s,a) \leftarrow$ valeur arbitraire
$s \leftarrow$ état-courant
Construire $M(s)$ (à partir de $Q_i(s,a)$)
Choisir une politique $\pi(s) = Equilibre_i(M(s))$
répéter
  Jouer $\pi(s)$
  $a \leftarrow$ l'action jointe
  $r \leftarrow$ les récompenses de chaque agent
  $s' \leftarrow$ nouvel état
  Construire $M(s')$
  $\pi(s') \leftarrow Equilibre_i(M(s'))$
  pour $i \in Ag$ **faire**
    Mettre à jour $Q_i(s,a)$
  $s' \leftarrow s$
  $t \leftarrow t+1$
jusqu'à $t \leqslant T$;
pour $s \in S$ **faire**
  Construire $M(s)$
  $\pi_i(s) \leftarrow Equilibre_i(M(s))$

Il a été démontré par Fudenberg [Fudenberg et Levine, 1998] que le jeu fictif converge vers un équilibre de Nash pour certains types de jeux, appelés jeux solvables par dominance itérative, c'est-à-dire les jeux dans lesquels il est possible d'éliminer les actions strictement dominées de façon itérative dans le but d'obtenir une seule action ou un ensemble d'actions réduit. L'adaptation du jeu fictif aux cas des jeux stochastiques à somme générale, a été faite par Vrieze [Vrieze, 1987], algorithme 7. Pour une lecture plus approfondie sur ce mode d'apprentissage, nous conseillons les travaux suivants : [Young, 1993, Claus et Boutilier, 1998, Uther et Veloso, 1997].

Algorithme 7: Algorithme du jeu fictif pour les jeux stochastiques.

Entrées : Un jeu stochastique, N : nombre d'étapes
Sorties : π_i

Initialisation
pour $s \in S$ **faire**
  pour $a_i \in A_i$ **faire**
    $Q_i(s, a_i) \leftarrow \frac{1}{|A_{-i}|} \sum_{a_{-i}}^{A_{-i}} R_i(s, (a_i, a_{-i}))$
    $n \leftarrow 0$
répéter
  pour $s \in S$ **faire**
    $a_i^n = argmax_{a_i \in A_i} \frac{Q_i(s,a_i)}{n}$
    Jouer a_i^n
    $a = (a_i, a_{-i})$ Observer l'action jointe
    pour $a_i \in A_i$ **faire**
      Mettre à jour $Q_i(s, a)$
  $n \leftarrow n + 1$
jusqu'à $n > N$;
pour $s \in S$ **faire**

$$\pi_i^{a_i}(s) \leftarrow \begin{cases} 1 & \text{si } a_i = a_i^N \\ 0 & \text{sinon} \end{cases}$$

– **Classe 3 : associant de l'apprentissage et la méthode de descente du gradient.**

Singh et al. [Singh *et al.*, 2000] ont été les premiers à employer la technique de la descente de gradient dans l'apprentissage multi-agents. Leur algorithme, IGA (Infinitesimal Gradient Ascent), suppose que chaque joueur effectue une descente

de gradient dans l'espace de ses politiques pour trouver la politique qui maximise son utilité escomptée.

Dans le même esprit, l'algorithme PHC (Policy Hill Climbing) proposé par Bowling et al. [Bowling et Veloso, 2002], est le premier algorithme proposé dans le cadre des jeux stochastiques, héritant des propriétés de convergence de l'algorithme IGA. Cet algorithme, réalise une ascension locale (hill-climbing) dans l'espace des stratégies mixtes. De ce fait, il représente une simple modification du Q-learning mono-agent. Il comporte deux parties : la première est basée sur le Q-learning en vue de maintenir les valeurs des actions simples (et non des actions conjointes) dans les états ; la deuxième partie est issue de la théorie des jeux et maintient la stratégie mixte courante dans chacun des états du système. Bowling et al. [Bowling et Veloso, 2002] sont allés plus loin et ont proposé des extensions des algorithmes IGA et PHC, appelées respectivement WoLF-IGA et WoLF-PHC. Nous conseillons la lecture de la thèse de Bowling [Bowling, 2003], pour une meilleure compréhension des ces algorithmes.

Dans la majorité des travaux cités précédemment, la préoccupation principale était la recherche de situations d'équilibre en dépit de s'intéresser à comment calculer un équilibre de Nash. En effet, mis à part certains cas, le calcul d'un équilibre de Nash est considéré encore comme un problème difficile. Ainsi, la prochaine section fera état de la complexité algorithmique de calcul des différents types d'équilibres de Nash.

2.4 Implémentation de l'équilibre de Nash

Dans cette section, nous allons nous intéresser aux procédures utilisées en vue d'implémenter les fonctions de calcul de l'équilibre de Nash. Pour chacune des techniques, nous proposons la façon de formuler le problème, l'algorithme généralement utilisé pour le résoudre et la complexité correspondante.

Equilibre de Nash à deux joueurs et jeux à somme nulle

La classe de jeux à somme nulle et à deux joueurs est la classe la plus simple à résoudre, puisque l'équilibre de Nash pour ces jeux peut être exprimé par un programme linéaire (PL), signifiant que l'équilibre de Nash peut être calculé en un temps polynomial.

Si on considère V_1^* comme l'utilité espérée de l'agent i dans l'équilibre, alors $V_1^* = -V_2^*$. Dès lors, un programme linéaire consiste pour le deuxième agent à minimiser V_2^* et pour le premier agent à maximiser V_1^* (sous certaines contraintes). Pour plus de détails le lecteur pourrait se référer à Shoham [Shoham et Leyton-Brown, 2009].

Equilibre de Nash à deux joueurs et jeux à somme générale

Caractériser la complexité de calcul d'un équilibre de Nash n'est pas une tâche aisée. En effet, aucune réduction qui mènerait à problème de décision NP-complet, n'est connue jusqu'à présent.

Partant de ce constat, les recherches sur la complexité de calcul d'un équilibre de Nash se sont portées sur une autre classe de complexité, qui décrit le problème de trouver une solution déjà existante. Cette classe porte le nom de PPAD (Polynomial Parity Argument, Direct version). Pour plus de détails sur cette classe de complexité le lecteur pourrait se référer à la page web de Paul Goldberg [12]. Le principal résultat mis en évidence relativement à cette classe de complexité est :

Théorème 2.4 *Le problème de trouver un équilibre de Nash d'un jeu à deux ou plusieurs joueurs, à somme générale, est PPAD-complet [Daskalakis* et al.*, 2006].*

Un tel théorème stipule qu'un jeu à deux joueurs a de fortes chances d'avoir la même complexité qu'un jeu à n-joueurs. Pour le moment, on ne sait pas si P = PPAD ou non ; cependant, il est généralement admis que les deux classes ne sont pas équivalentes. Dès lors, il y a de fortes chances pour que dans le pire des cas, calculer un équilibre de Nash prendra un temps exponentiel en fonction de la taille du jeu.

Contrairement au problème de deux joueurs à somme nulle, le problème de trouver un équilibre de Nash dans le cas où la somme est générale, ne peut pas être formulé comme un programme linéaire PL. Toutefois, il peut être formulé comme un problème linéaire complémentaire (PLC) [Cottle *et al.*, 2009], une formulation très proche d'un PL.

Equilibre de Nash à N-joueurs et à somme générale

Pour des jeux à N joueurs ($N > 2$), le problème de trouver un équilibre de Nash ne peut plus être représenté comme un PLC. Toutefois, il pourrait être représenté par un problème non-linéaire complémentaire (PNLC) [Cottle, 1966] ; ainsi on ne peut le résoudre exactement et il faudra lui trouver des méthodes approximatives.

Stratégies Maximin et Minimax pour deux joueurs et jeux à somme nulle

Les stratégies *Minimax* et *Maximin* peuvent être calculées en un temps polynomial. En effet, on peut montrer aisément que de telles stratégies pour un jeu donné, correspondent en fait à des stratégies d'équilibre de Nash dans les jeux J' et J'' à somme nulle où J' est obtenu à partir de J en remplaçant V_2 par $-V_1$ et où J'' est obtenu similairement en remplaçant V_1 par $-V_2$.

12. http ://www.csc.liv.ac.uk/˜pwg/PPADintro/PPADintro.html, accessible en 2012.

Ainsi, on pourrait donc appliquer les résultats qu'on a précédemment vus pour le calcul de l'équilibre de Nash à des jeux à somme nulle et à deux joueurs.

2.5 Limitations

Dans cette section, nous rappelons les différentes limitations des algorithmes précédemment présentés. En effet, qu'ils soient en planification ou en apprentissage, la plupart des algorithmes de résolution pour les jeux stochastiques, présentent deux inconvénients majeurs, empêchant ainsi leur application à grande échelle, nous pourrons citer (1) l'explosion combinatoire et (2) les fortes restrictions sur la structure des jeux.

Prenons par exemple, l'algorithme de Shapley pour les jeux stochastiques à deux agents et à somme nulle. Cet algorithme, s'appuie sur la technique d'itération sur les valeurs et utilise l'algorithme *Minimax* pour trouver l'équilibre de Nash dans chaque jeu. De ce fait, nous pourrons dire que l'algorithme possède un temps d'exécution polynomial. Cependant, cette affirmation est contredite, si nous prenons en considération que l'espace d'actions et l'espace d'états sont exponentiels en fonction du nombre d'agents.

Concernant l'algorithme de Kearns et al., le même raisonnement est appliqué pour décider de son temps d'exécution. Dans cet algorithme, la fonction de sélection d'équilibre est supposée être unitaire, ce qui n'est pas le cas en pratique, car nous ne connaissons pas d'algorithme ayant un temps d'exécution polynomial pour le calcul d'un équilibre de Nash dans un jeu à somme générale. Ceci laisse supposer que l'algorithme n'est pas polynomial.

La complexité de calcul des différents algorithmes d'apprentissage n'est pas meilleure que celle des algorithmes de planification. Le résultat principal, sur lequel se basent les estimations de complexité, est celui de Koenig et Simmons (1996). Dans leur article sur la complexité du Q-learning, les auteurs ont prouvé que le temps nécessaire pour la convergence vers une solution optimale est exponentiel en fonction du nombre d'états du système.

A ces difficultés, se rajoute d'autres problèmes, en effet beaucoup de ces algorithmes imposent des restrictions sur la structure ou le déroulement du jeu. Nous citons quelques unes de ces limitations :

- Algorithme de Shapley : considère seulement deux agents.
- Algorithme de Kearns et al. : considère l'unicité de l'équilibre de Nash.
- Algorithme de Burkov et al. : ne définit pas le moyen communication et considère le coût correspondant comme nul.
- Algorithme Minimax-Q : considère seulement deux agents et suppose que l'autre agent utilise la même version de l'algorithme.

– Algorithme IGA : suppose connue la politique de l'adversaire ainsi que la matrice du jeu.

Il convient de noter que ce chapitre n'a pas abordé une autre branche de recherche dans le domaine des jeux stochastiques, cette branche s'occupe des jeux partiellement observables (POSG, pour Partially Observable Stochastic Games). Dans ce modèle, les jeux sont combinés avec le modèle des POMDP (Partially Observable Markov Decision Process [Kaelbling *et al.*, 1998, Sondik, 1978, Hansen, 1998]), ce qui permet de résoudre des problèmes où les agents n'ont qu'une vue partielle de l'environnement. Quelques approches ont montré leur efficacité dans la résolution des POSG, celles-ci se trouvent dans [Hansen, 2004, Emery-Montemerlo *et al.*, 2004, Szer et Charpillet, 2006].

2.6 Conclusion

Cette thèse s'inscrit dans la problématique de la planification, qui consiste à trouver pour un agent donné une séquence d'actions permettant d'atteindre un but donné. Plus spécifiquement, nous cherchons à résoudre des problèmes comportant plusieurs agents interagissant les uns avec les autres. Les jeux stochastiques permettent de modéliser ce type de système (jeux matriciels avec des transitions stochastiques) et représentent actuellement une théorie mathématique riche.

Le but de ce chapitre était de présenter un aperçu des algorithmes de planification et d'apprentissage utilisant le formalisme des jeux stochastiques. Pour l'approche par planification, nous avons détaillé quelques algorithmes dont les deux plus influents du domaine, à savoir l'algorithme de Shapley et celui de Kearns et al. Pour l'approche de l'apprentissage, nous avons introduit les méthodes de résolution les plus connues, en détaillant des algorithmes comme le *Minimax-Q* ou le *Fictitious play*. En partant du fait que la plupart des algorithmes présentés, possèdent une composante de calcul d'équilibre, nous avons dédié toute une section à la manière dont cet équilibre est calculé, les algorithmes et les complexités correspondants.

Nous avons pu constater que la majorité de ces algorithmes possédaient une solide base théorique. En revanche, leur application laisse à désirer et les études expérimentales ne sont pas nombreuses. Hormis les assises expérimentales manquantes, il existe d'autres problèmes qui doivent être résolus afin de permettre à ces algorithmes d'être appliqués dans la résolution de problèmes à grande échelle. Tous d'abord, le problème de la complexité qui reste assez élevée. En effet, une partie de ce problème est due à l'explosion combinatoire et oblige le nombre d'états à croître exponentiellement en fonction du nombre d'agents.

Ensuite, s'ajoute les problèmes du calcul de l'équilibre de Nash et la multiplicité des équilibres dans un jeu. Pour ce dernier problème, la coordination sur le choix d'un seul équilibre n'est pas évidente pour le moment et les solutions qui ont été proposées rendent le problème encore plus complexe (oracle, communication inter-agents gratuite, etc.).

Dans le chapitre suivant nous allons présenter une nouvelle approche, développée dans le cadre de ce travail, afin de surmonter certaines de ces lacunes. Il s'agit d'un algorithme de planification multi-agents, axé sur la résolution des problèmes liés aux équilibres de Nash ainsi que sur la réduction de la complexité de l'interaction.

Chapitre 3

Proposition d'un algorithme de planification pour les jeux stochastiques

Sommaire

3.1 Proposition d'un algorithme de planification décentralisé pour les jeux stochastiques 61

3.1.1 Calcul des équilibres 62

3.1.2 Sélection d'un équilibre 68

3.1.3 Algorithme pour la résolution des jeux stochastiques 73

3.1.4 Application numérique 77

3.2 Amélioration de l'algorithme 78

3.2.1 Principe de la dominance stratégique 78

3.2.2 Extension de l'algorithme SGInfiniteVI 81

3.3 Conclusion 85

Dans ce chapitre, nous nous focalisons sur le problème de la planification distribuée dans les jeux stochastiques à horizon infini. Le problème est caractérisé par l'hypothèse que l'environnement est complètement observable, et que les agents ne peuvent pas communiquer pendant et après le processus de planification. Le problème de la planification dans le contexte des jeux stochastiques consiste à calculer un équilibre pour chaque jeu/état. Cependant, certains types de jeux stochastiques, peuvent avoir de multiples équilibres. La multiplicité des équilibres est un sérieux obstacle dans toute démarche de planification pour ces jeux, parce que les agents peuvent avoir des préférences pour des équilibres différents, et leur action jointe peut ne pas constituer un équilibre. Dans ce chapitre, nous proposons une approche qui permet de surmonter cet obstacle, en proposant différentes fonctions de sélection d'équilibre, censées orienter les agents vers le même choix. La complexité constitue également un grand défi dans l'implémentation et l'application des algorithmes de planification. Cette complexité est due, d'une part à l'explosion combinatoire et d'autre part au coût occasionné par les calculs d'équilibres. Dans ce chapitre, nous proposons une procédure issue de la théorie des jeux et permettant de réduire les ressources utilisées pendant le processus de planification.

Introduction

"Planifier" dans les jeux stochastiques consiste à décider pour chaque agent ce qu'il faut faire à chaque état. Comme nous l'avons mis en évidence dans le chapitre 2, la planification pour résoudre un jeu stochastique suppose que les agents aient connaissance du modèle de l'environnement (les fonctions de récompense des autres agents et la fonction de transition du modèle).

Contrairement aux approches par apprentissage, il y a relativement moins d'algorithmes de planification pour les jeux stochastiques. Nos travaux sont dans la continuité des travaux de Kearns et al., et s'inspirent de leur algorithme FiniteVI. Cet algorithme d'itération sur les valeurs, peut être appliqué pour la résolution des jeux stochastiques à somme générale et à horizon fini. D'autres algorithmes, d'itération sur les valeurs pour les jeux stochastiques, peuvent être également appliqués, mais seulement à des sous-classes du modèle (l'algorithme de Shapley [Shapley, 1953]), ou pour chercher un autre type de solution (l'algorithme InfiniteVI de Kearns et al. [Kearns *et al.*, 2000]).

Cet algorithme nécessite cependant un niveau élevé de centralisation. Plus précisément, il nécessite la présence d'un "oracle" indispensable au processus de planification. Dans le cas de la multiplicité d'équilibres, la fonction f de sélection d'équilibre est censée

sélectionner un équilibre pour chaque couple état-action de l'algorithme et le communiquer à tous les agents.

L'objectif initial de ce travail est d'éviter la centralisation, impliquée par la fonction de sélection f. Notre approche vise à rendre les agents plus autonomes, et donc à leur laisser le choix du calcul d'équilibre. Le problème se pose réellement lorsque plusieurs équilibres de Nash coexistent dans le même jeu. Nous allons essayer de proposer (dans de telles circonstances) la fonction permettant aux agents de s'orienter vers la sélection d'un même équilibre.

Le second objectif de ce travail est de réduire la complexité de calcul de l'algorithme proposé. En effet, comme nous l'avons déjà évoqué, les temps d'exécution des algorithmes de type itération sur les valeurs ne sont pas polynomiaux. La première cause, vient du fait que les ensembles d'actions et d'états sont exponentiels en fonction du nombre d'agents. La deuxième cause, est la méthode de calcul de l'équilibre de Nash dans un jeu à somme générale, qui est non polynomiale. Notre approche se base sur la technique de la dominance itérative (issue de la théorie des jeux) afin de réduire les matrices de gain et accélérer ainsi le processus de planification.

Ce chapitre se divise en deux sections. La première section, présentera les différentes étapes qui ont été nécessaires à l'élaboration d'un algorithme de planification pour les jeux stochastiques à somme générale, comme le calcul et la sélection des équilibres. Nous aborderons également le problème de la convergence de l'algorithme ainsi que sa complexité temporelle et spatiale. Dans la deuxième section, nous proposons une amélioration de l'algorithme, basée sur la technique d'élimination des stratégies dominées et nous montrons que cette méthode pourrait réduire les ressources (temps de calcul et espace mémoire) nécessaires à l'exécution de l'algorithme.

3.1 Proposition d'un algorithme de planification décentralisé pour les jeux stochastiques

Comme mentionné ci-dessus, le principal inconvénient de l'algorithme FiniteVI est sa forte centralisation, remettant en question l'autonomie des agents. La centralisation provient de la fonction de sélection f, qui doit calculer un équilibre unique et le communiquer à tous les agents. Une telle centralisation est souvent indésirable, car elle suppose l'existence d'une entité qui officieusement parlant "sait mieux ce qu'il faut faire" pour chaque agent. Cette propriété est souvent difficile à appliquer, parce que les agents peuvent avoir différentes visions de "ce qu'il faut faire" et de "ce qu'il ne faut pas faire".

On peut imaginer la situation où tous les agents possèdent la même fonction de sélection d'équilibre ; un tel cadre permettrait d'éviter la nécessité d'un oracle. Plus précisément, il serait nécessaire que :

$$\forall s \in S, \quad f_1(s,t) = f_2(s,t) = \ldots = f_{|Ag|}(s,t) \tag{3.1}$$

Généralement, l'égalité des fonctions de sélection d'équilibre ne peut être assurée que rarement dans les systèmes distribués. En effet, le problème existera toujours, car pour calculer un équilibre unique d'une manière distribuée, tous les agents doivent avoir les mêmes matrices de paiement. Il est à noter qu'un problème similaire a été observé dans d'autres algorithmes pour les jeux stochastiques. Par exemple, dans l'algorithme de Nash-Q de Hu et Wellman [Hu et Wellman, 2003], les agents ont été programmés pour choisir le premier équilibre calculé, ou le second et ainsi de suite. En contre partie, cela implique, que les agents doivent calculer des ensembles d'équilibres identiques (ayant le même ordre).

Pour éviter cette centralisation et pour ajouter des propriétés souhaitables au système, comme le calcul distribué des équilibres, nous proposons une nouvelle approche pour la sélection d'équilibre. Notre approche est basée sur des fonctions de sélection mesurant l'utilité des équilibres, indépendamment de la manière dont ils sont calculés ou ordonnés.

3.1.1 Calcul des équilibres

Pour calculer un équilibre de Nash, l'étude de l'état de l'art nous a amené dans un premier temps à définir trois paramètres :

- Le nombre d'agents : conformément au formalisme des jeux stochastiques, nous avons considéré que le nombre d'agents peut être supérieur ou égal à deux.
- Le type du jeu : dans la continuité des travaux de Kearns et al. [Kearns *et al.*, 2000], nous choisissons de résoudre des jeux à somme générale.
- Le type de stratégies : concernant ce choix, nous avons décidé de considérer seulement les stratégies pures, car (1) représentant pour les agents la manière la plus simple de jouer et (2) ayant une complexité de calcul moins importante.

Actuellement, l'algorithme de Lemke-Howson [Lemke et Howson, 1964] est l'algorithme le plus efficace pour calculer un équilibre de Nash. L'idée de l'algorithme est de parcourir l'espace des actions en se basant sur la notion de meilleures réponses. Cependant, cet algorithme ne peut calculer que des équilibres de Nash à deux joueurs et avec des stratégies mixtes.

Dans le but de considérer un nombre d'agents plus important et des stratégies pures pour ces agents, nous proposons un nouvel algorithme pour le calcul des équilibres de Nash

dans un jeu matriciel à somme générale. L'algorithme se base également sur le concept de meilleure réponse et opère en deux étapes :

1. étape 1 : consiste à trouver la meilleure réponse de chaque agent i par rapport aux actions a_{-i} des autres agents, rappelons l'équation :

$$\forall a_i \in A_i \quad R_i(\langle a_i^*, a_{-i}\rangle) \geqslant R_i(\langle a_i, a_{-i}\rangle) \tag{3.2}$$

   Afin de marquer les meilleures réponses, une deuxième matrice a été nécessaire $BM(s)$. Cette matrice possède la même taille que la matrice de paiement traitée et ne contiendra que des valeurs booléennes. A chaque fois, qu'une action a_i de l'agent i est évaluée et est considérée comme la meilleure réponse face aux actions a_{-i} des autres agents, l'entrée correspondante $BM(s, \langle a_i, a_j\rangle)_i$ prend la valeur booléenne *Vrai*.

2. étape 2 : consiste à satisfaire l'équation suivante garantissant l'équilibre de Nash :

$$\forall i \in Ag,\ \forall a_i \in A_i \quad R_i(\langle a_i^*, a_{-i}^*\rangle) \geqslant R_i(\langle a_i, a_{-i}^*\rangle) \tag{3.3}$$

   Pratiquement, cela correspond à parcourir la matrice $BM(s)$ et à repérer tous les tuples $\langle Vrai, \ldots, Vrai\rangle$ représentant des équilibres de Nash.

$M(s)$

	$agent_2$	
$agent_1$	(2,10)	(0,10)
	(1,3)	(5,9)

	$agent_2$	
$agent_1$	**(2,10)**	(0,10)
	(1,3)	**(5,9)**

$BM(s)$

	$agent_2$	
$agent_1$	(Vrai,Vrai)	(Faux,Vrai)
	(Faux,Faux)	(Vrai,Vrai)

	$agent_2$	
$agent_1$	**(Vrai,Vrai)**	(Faux,Vrai)
	(Faux,Faux)	**(Vrai,Vrai)**

FIGURE 3.1 – Technique de calcul d'un équilibre de Nash par énumération des meilleures réponses.

L'exemple proposé en figure 3.1, montre les étapes nécessaires au calcul d'un équilibre de Nash par énumération des meilleures réponses. Ainsi, la meilleure action de l'$agent_1$ face à la première action de l'$agent_2$ est celle qui apporte le gain +2. Respectivement, c'est la deuxième action de l'$agent_1$ qui apporte le meilleur gain face à la deuxième action de l'$agent_2$. Si les deux agents suivent le même raisonnement, nous obtenons alors les équilibres $(2, 10)$ et $(5, 9)$. Nous présentons également la définition formelle de cette méthode (l'algorithme 8). L'algorithme prend en entrée une matrice de paiement bidimensionnelle à somme générale et renvoie tous les équilibres de Nash en stratégies pures pouvant exister.

Algorithme 8: EMR (Enumération des meilleures réponses).

Entrées : Une matrice de paiement $M(s)$

$ListeNash \leftarrow \emptyset$, $BM(s) \leftarrow False$
Première étape : énumérer les meilleures réponses
pour $a_j \in A_j$ **faire**
    $Max \leftarrow M(s, (\langle 0, a_j \rangle))$
    pour $a_i \in A_i$ **faire**
        si $M(s, (\langle a_i, a_j \rangle))_i > Max$ **alors**
            $Max \leftarrow M(s, (\langle a_i, a_j \rangle))_i$
    pour $a_i \in A_i$ **faire**
        si $M(s, (\langle a_i, a_j \rangle))_i = Max$ **alors**
            $BM(s, (\langle a_i, a_j \rangle))_i \leftarrow$ **Vrai**
pour $a_i \in A_i$ **faire**
    $Max \leftarrow M(s, (\langle a_i, 0 \rangle))_j$
    pour $a_j \in A_j$ **faire**
        si $M(s, (\langle a_i, a_j \rangle))_j > Max$ **alors**
            $Max \leftarrow M(s, (\langle a_i, a_j \rangle))_j$
    pour $a_j \in A_j$ **faire**
        si $M(s, (\langle a_i, a_j \rangle))_j = Max$ **alors**
            $BM(s, (\langle a_i, a_j \rangle))_j \leftarrow$ **Vrai**
Deuxième étape : extraire tous les équilibres de Nash existants.
pour $a_j \in A_j$ **faire**
    pour $a_i \in A_i$ **faire**
        si $BM(s, (\langle a_i, a_j \rangle)) = (\langle$ **Vrai**, **Vrai** $\rangle)$ **alors**
            $ListeNash \leftarrow M(s, (\langle a_i, a_j \rangle))$
Troisième étape : calculer des équilibres ε-Nash
si $ListeNash = \emptyset$ **alors**
    EMR-epsilon(M(s),ε)

Sorties : Tous les équilibres de Nash existants

Cependant, il existe des jeux où il n'y a pas d'équilibre de Nash, c'est à dire que quoi qu'il en soit un des agents regrettera son choix. Soit le jeu en stratégies pures (deux stratégies par agent s_1 et s_2) dont la matrice des gains est la suivante (figure 3.2) :

Si l'$agent_1$ joue s_1 :

– Et $agent_2$ joue s_1 : $agent_2$ regrettera son choix, il aurait dû jouer s_2 (gain $(0, 1)$).

– Et $agent_2$ joue s_2 : $agent_1$ regrettera son choix, il aurait dû jouer s_2 (gain $(1, 0)$).

Si l'$agent_1$ joue s2 :

– Et $agent_2$ joue s_1 : $agent_1$ regrettera son choix, il aurait dû jouer s_1 (gain $(1, 0)$).

– Et $agent_2$ joue s_2 : $agent_2$ regrettera son choix, il aurait dû jouer s_1 (gain $(0, 1)$).

		$agent_2$	
		s_1	s_2
$agent_1$	s_1	(1,0)	(0,1)
	s_2	(0,1)	(1,0)

FIGURE 3.2 – Matrice ne comportant pas d'équilibre.

Ainsi ce jeu n'a pas d'équilibre de Nash en stratégies pures. Des chercheurs comme Debreu [Debreu, 1952] et Glicksberg [Glicksberg, 1952] ont donné les conditions d'existence d'un équilibre de Nash en stratégies pures. Ce théorème d'existence a été étendu aux jeux ayant des fonctions de gain discontinues [Gatti, 2005]. L'existence d'un équilibre de Nash dans un jeu est assurée suivant les hypothèses suivantes :

– la continuité et la quasi-concavité des fonctions de récompense,

– la convexité et la compacité des ensembles d'actions.

L'existence d'un équilibre de Nash dans un jeu est liée à l'existence d'un point fixe dans la correspondance de meilleure réponse, pour plus de détails voir [Fudenberg et Tirole, 1991]. Néanmoins, les états d'un jeu stochastique sont générés automatiquement à partir des états individuels de chaque agent. Par conséquent, il est difficile d'assurer l'ensemble des quatre hypothèses et ainsi de garantir l'existence d'un équilibre de Nash en stratégies pures. A ce stade, l'algorithme d'énumération des meilleures réponses peut ne pas aboutir à un équilibre de Nash.

Pour contourner ce problème, il existe une solution calculant un équilibre de Nash approximatif [Radner, 1980]. Dans la théorie des jeux, un ε-équilibre de Nash (ou ε-Nash), est une action jointe qui satisfait à peu près la condition d'équilibre de Nash. La notion d'ε-équilibre est importante dans la théorie des jeux, car il existe des jeux sans équilibre de Nash, mais qui ont un ε-équilibre pour un epsilon donné strictement supérieur à 0.

Définition 3.1 *Etant donné un jeu et un paramètre $\varepsilon > 0$, une action jointe est un ε-équilibre de Nash, si aucun joueur i ne peut améliorer son gain de plus de ε en changeant unilatéralement de stratégie. Formellement, cela donne :*

$$\forall a_i \in A_i, \qquad R_i(a_i^*, a_{-i}^*) \geqslant R_i(a_i, a_{-i}^*) - \varepsilon \tag{3.4}$$

Notons que tout équilibre de Nash (ordinaire) est équivalent à un ε-équilibre de Nash où $\varepsilon = 0$. Notons également que pour tout $\varepsilon_1 < \varepsilon_2$, un ε_1-équilibre de Nash est également un ε_2-équilibre de Nash. Par conséquent, il s'ensuit que tout jeu fini possède un ε-équilibre de Nash, pour tout $\varepsilon \geqslant 0$. A notre connaissance, il n'existe pas de travaux permettant de déterminer la valeur de ε. Nous pensons que ce paramètre peut être fixé empiriquement, en fonction de la définition des fonctions de récompense des agents.

Ainsi, dans le but de trouver une action jointe, assurant le meilleur gain pour chaque agent dans les situations où il n'y a pas d'équilibre, nous autorisons le calcul des équilibres ε-Nash. En conséquence, l'algorithme 8 est modifié, de sorte qu'il inclura une troisième étape résolvant le cas de non existence d'équilibre. Nous présentons un deuxième algorithme, nommé EMR-epsilon (algorithme 9), qui prend en entrée une matrice $M(s)$ et le paramètre ε et permettant de calculer les meilleures réponses à ε près. De ce fait, nous estimons qu'au pire cas (toutes les matrices traitées ne possèdent pas d'équilibre), l'algorithme converge vers une solution approchée de l'équilibre de Nash.

La figure 3.3 montre un exemple de situation, où il est impossible de trouver un équilibre de Nash en stratégies pures. En fixant ε à une valeur donnée (± 0.2), l'algorithme trouve un équilibre approximatif avec le couple $(4.9, 10)$.

$M(s)$

	$agent_2$	
$agent_1$	(4.9,10)	(1,1)
	(5,3)	(0,9)

	$agent_2$ ± 0.2	
$agent_1$ ± 0.2	(4.9,10)	(1,1)
	(5,3)	(0,9)

$BM(s)$

	$agent_2$	
$agent_1$	(Faux,Vrai)	(Vrai,Faux)
	(Vrai,Faux)	(Faux,Vrai)

	$agent_2$	
$agent_1$	(Vrai,Vrai)	(Vrai,Faux)
	(Vrai,Faux)	(Faux,Vrai)

FIGURE 3.3 – Recherche d'équilibre de Nash approximatifs avec la méthode de meilleures réponses.

Adapter l'algorithme aux cas de n agents, consiste à construire dorénavant une matrice multidimensionnelle, au lieu d'une matrice à deux dimensions. Il s'agit d'un hypercube ayant $|Ag|$ dimensions, où chaque dimension représente l'ensemble des actions de chaque

Algorithme 9: EMR-epsilon (à epsilon près).

Entrées : Une matrice de paiement $M(s)$, $\varepsilon > 0$

$ListeNash \leftarrow \emptyset$, $BM(s) \leftarrow False$

Première étape : énumérer les meilleures réponses

pour $a_j \in A_j$ **faire**

  $Max \leftarrow M(s, (\langle 0, a_j \rangle))_i$

  pour $a_i \in A_i$ **faire**

    si $M(s, (\langle a_i, a_j \rangle))_i > Max$ **alors**

      $Max \leftarrow M(s, (\langle a_i, a_j \rangle))_i$

  pour $a_i \in A_i$ **faire**

    si $M(s, (\langle a_i, a_j \rangle))_i = (Max - \varepsilon)$ **alors**

      $BM(s, (\langle a_i, a_j \rangle))_i \leftarrow$ **Vrai**

pour $a_i \in A_i$ **faire**

  ⋮

Deuxième étape : extraire tous les équilibres de Nash existants.

pour $a_j \in A_j$ **faire**

  pour $a_i \in A_i$ **faire**

    si $BM(s, (\langle a_i, a_j \rangle)) = (\langle$ **Vrai**, **Vrai** $\rangle)$ **alors**

      $ListeNash \leftarrow M(s, (\langle a_i, a_j \rangle))$

Sorties : Tous les équilibres de Nash existants

agent. Le processus de recherche d'équilibre demeure le même, à savoir une phase d'énumération des meilleures réponses et une phase de repérage d'équilibre, voir algorithme 10.

3.1.2 Sélection d'un équilibre

Dans [Stengel, 1999] Stengel affirme que le nombre maximum d'équilibres de Nash d'une matrice à deux joueurs est borné par ϕ_{max}, avec :

$$\phi_{max} \simeq 0.921 \frac{\sqrt{\frac{27}{4}}^{|A|}}{\sqrt{|A|}}$$

où $|A| = |A_1| = |A_2|$ en supposant que les espaces d'actions des agents sont identiques. Cette équation montre que le nombre d'équilibres de Nash peut être conséquent même avec un nombre d'actions limité. Le problème posé par la multiplicité des équilibres de Nash est considéré en théorie des jeux comme un problème de coordination [Chatterjee, 2002]. Dans une telle situation, les questions qui se posent sont de savoir vers quels équilibres les agents vont se coordonner et sur quelle base. Les réponses dépendent des croyances de chaque agent sur ce que les autres feront.

Pour certains jeux, il y a des équilibres qui semblent être plus évidents pour les agents que d'autres. Ainsi, les individus se coordonnent spontanément vers un équilibre donné du fait de références communes, en s'appuyant sur certains mécanismes sociaux (règles, normes, conventions) permettant de rendre cohérentes les stratégies d'un ensemble d'acteurs.

Les références communes peuvent être définies comme des mesures de performance (ou de coût social) dans les jeux. Suivant les systèmes, cela peut se quantifier par exemple en fonction des ressources utilisées par les agents ou de la durée de leurs missions.

Les travaux existants, comme ceux de Hu et Wellman, considèrent le problème de la multiplicité des équilibres et proposent des fonctions de sélection choisissant entre le premier ou le deuxième équilibre. Ce raisonnement, suppose que les matrices de paiement sont les mêmes pour les deux agents $\forall s \in S,\ \forall t \in T \quad M_1(s,t) = M_2(s,t)$ et que l'algorithme calculant les équilibres renvoie une même liste ordonnée des équilibres du jeu. Il s'agit d'une résolution approximative, car certaines propriétés sont difficiles à réaliser en pratique. Par exemple, les agents peuvent avoir des fonctions de récompense différentes ou utiliser des algorithmes de calcul d'équilibre distincts.

Pour récapituler, le problème de la multiplicité d'équilibre n'étant pas le seul à engendrer un défaut de coordination entre les agents, se rajoute le problème des agents qui ne calculent pas les mêmes ensembles d'équilibres ou lorsque ceux-ci ne sont pas ordonnés

Algorithme 10: EMR (adaptation au cas de n agents).

Entrées : Une matrice de paiement $M(s)$

$ListeNash \leftarrow \emptyset$, $BM(s) \leftarrow False$

Première étape : énumérer les meilleures réponses

pour $i \in |Ag|$ **faire**

  pour $a_{|Ag|} \in A_{|Ag|}$ **faire**

    pour $a_{|Ag|-1} \in A_{|Ag|-1}$ **faire**

      $\vdots$

      pour $a_0 \in A_0$ **faire**

        $Max \leftarrow M(s, (\langle 0, \ldots, a_{|Ag|} \rangle))_i$

        pour $a_i \in A_i$ **faire**

          si $M(s, (\langle a_i, a_{-i} \rangle))_i > Max$ **alors**

            $Max \leftarrow M(s, (\langle a_i, a_{-i} \rangle))_i$

        pour $a_i \in A_i$ **faire**

          si $M(s, (\langle a_i, a_{-i} \rangle))_i = Max$ **alors**

            $BM(s, (\langle a_i, a_{-i} \rangle))_i \leftarrow$ **Vrai**

Deuxième étape : extraire tous les équilibres de Nash existants.

pour $a_{|Ag|} \in A_{|Ag|}$ **faire**

  pour $a_{|Ag|-1} \in A_{|Ag|-1}$ **faire**

    $\vdots$

    pour $a_0 \in A_0$ **faire**

      si $BM(s, (\langle a_0, a_1, \ldots, a_{|Ag|} \rangle)) = (\langle$ **Vrai**, **Vrai**, $\ldots$, **Vrai** $\rangle)$ **alors**

        $ListeNash \leftarrow M(s, (\langle a_0, a_1, \ldots, a_{|Ag|} \rangle))$

Troisième étape : calculer des équilibres ε-Nash

si $ListeNash = \emptyset$ **alors**

  EMR-epsilon(M(s),ε)

Sorties : Tous les équilibres de Nash existants

de la même manière. L'idée de vouloir obtenir un équilibre unique et commun à tous les agents, tout en distribuant le calcul, semble *a priori* être difficile à réaliser.

Dans le cas des jeux stochastiques, l'approche centralisée consiste donc à avoir des matrices de paiements identiques, des algorithmes calculant les mêmes équilibres et sélectionner un équilibre unique pour tous les agents. L'approche centralisée, représente en quelque sorte, une borne optimale ou supérieure de ce que les agents peuvent faire de mieux pour l'accomplissement de leurs missions. En termes d'utilité numérique, l'approche centralisée devra apporter pour un agent donné la meilleure utilité en fonction de ce que font les autres agents. Nous définissons la borne optimale comme la somme des utilités (gains) perçues par l'agent le long des T étapes qui ont été nécessaires à la réalisation de son objectif $\sum_{t=0}^{T} R_i(s, \langle a_i^*, a_{-i}^* \rangle)$.

L'approche décentralisée, relaxe quelques unes des hypothèses précédentes au profit d'une meilleure autonomie de décision. De ce fait, chaque agent agira au "mieux" et en fonction de ce qu'il possède comme information, sans qu'il ait besoin d'une entité centrale décidant à sa place en situation difficile (par exemple en cas de multiplicité des équilibres). Dans ce contexte, il ne peut prétendre qu'à une conception partielle des comportements que peuvent avoir les autres agents. En terme d'utilité numérique, l'approche décentralisée tente de s'approcher de la borne optimale définie plus haut, de sorte que les gains perçus par l'agent i ne peuvent dans le meilleur cas dépasser ceux pouvant être perçus avec l'approche centralisée.

$$\sum_{t=0}^{T} R_i(s, \langle a_i, a_{-i} \rangle) \leqslant \sum_{t=0}^{T} R_i(s, \langle a_i^*, a_{-i}^* \rangle)$$

En effet, même en considérant que le plan d'actions d'un agent i est optimal (au niveau individuel), la mise en oeuvre des plans de tous les agents peut ne pas constituer un optimum (un équilibre) à chaque étape du jeu. La question principale est d'être capable de mesurer la distance séparant ce comportement de sa borne optimale. En d'autres termes, l'approche décentralisée n'offre aucune garantie sur l'unicité des équilibres choisis par les agents mais admet tout de même que chaque agent planifie en fonction des autres et choisit ses actions parmi les situations qui sont susceptibles d'être bénéfiques pour tous. A ces difficultés, s'ajoute le problème de la convergence de l'algorithme vers la solution optimale.

Si nous admettons que l'objectif d'un agent est de maximiser son gain, nous pouvons alors définir les situations bénéfiques comme des équilibres offrant les meilleurs gains. De ce fait, une situation pouvant captiver tous les agents, se définit alors comme un équilibre garantissant le meilleur gain pour chaque agent et se démarque ainsi des autres équilibres présents dans le jeu.

Si nous supposons que la plupart des états du jeu stochastique présentent une multiplicité d'équilibres, et que l'agent préférera toujours certains équilibres à d'autres (toujours en fonction du même critère) alors il sera judicieux de définir un comportement par critère de choix. Suivant la définition du critère de sélection, le comportement de l'agent sera fixé à l'avance (avant la phase de planification). Le comportement de l'agent déterminera la manière dont il se comportera avec les autres agents jusqu'à l'accomplissement de son objectif.

Ainsi, en cas de multiplicité d'équilibres, nous avons mis en place plusieurs critères permettant d'affiner la sélection des équilibres. Il est possible de choisir parmi les fonctions suivantes :

1. $f^{NashPareto}$: cette fonction choisit un équilibre de Nash non-dominé au sens de Pareto.

   Rappelons qu'un jeu est dans un état Pareto-optimal si aucun joueur ne peut augmenter son profit sans diminuer le gain de l'un des autres joueurs. La notion d'optimum de Pareto permet de diviser en deux sous ensembles les équilibres possibles du jeu. On peut ainsi distinguer :

   – Ceux qui sont uniformément améliorables : il est possible d'augmenter le bien-être de certains agents sans réduire celui des autres. On parle ici d'une amélioration au sens de Pareto.
   – Ceux qui ne sont pas uniformément améliorables : l'augmentation du bien-être de certains agents implique la réduction du bien-être d'au moins un autre agent.

   Ce sont ces derniers équilibres que l'on désigne comme optimaux au sens de Pareto.

$$f^{NashPareto} : S \rightarrow \hat{A}, \text{ tel que pour l'agent i : } R_i(\langle \hat{a}_i, \hat{a}_{-i} \rangle) > R_i(\langle a_i, a_{-i} \rangle)$$
$$\text{et pour les autres agents : } \forall j \in Ag \; R_j(\langle \hat{a_j}, \hat{a}_{-j} \rangle) \geqslant R_j(\langle a_j, a_{-j} \rangle)$$

2. $f^{NashMaxTot}$: choisit l'équilibre maximisant le gain de tous les agents.

   La maxime utilitariste [Schmidt, 1988, Leroux et Livet, 2006] généralement attribuée à Bentham [Sen *et al.*, 1982], appelée également *le plus grand bonheur du plus grand nombre*, s'est assez rapidement simplifiée, en la maximisation sur l'ensemble des options (des équilibres) de la somme des utilités individuelles. Cette maximisation suppose évidemment que les valeurs prises par les fonctions d'utilité, sont des nombres réels formant ce qu'on appelle (en algèbre) un corps. Pour ce qui nous concerne, cela signifie qu'on puisse comparer les utilités pour un même individu ou pour des individus différents. Par exemple, si l'utilité de x pour un agent i est 4 et l'utilité de y pour l'agent j est 2, qu'on puisse affirmer que i avec x a une utilité deux fois plus grande que l'utilité de j avec y, et de là, si l'utilité est une espèce de

mesure de la satisfaction ou du bonheur, que i est deux fois plus heureux avec x que j avec y.

Ce qui est recherché avec le concept du *plus grand bonheur du plus grand nombre*, c'est la possibilité de dépasser l'optimalité au sens de Pareto, qui se révèle malheureusement peu discriminante (dans de nombreux cas, les situations optimales sont multiples).

$$f^{NashMaxTot} : S \rightarrow \hat{A}, \sum_{i}^{Ag} R_i(\langle \hat{a}_i, \hat{a}_{-i} \rangle) > \sum_{i}^{Ag} R_i(\langle a_i, a_{-i} \rangle) \tag{3.5}$$

3. $f^{NashMaxSub}$: cette fonction choisit l'équilibre maximisant le gain individuel de l'agent exécutant l'algorithme.

   De ce fait, le comportement de l'agent peut paraître individualiste, car son choix est uniquement déterminé en fonction de son utilité. Néanmoins, ce choix demeure un choix social, quelque soit le critère de sélection, parce qu'il se fait toujours parmi des équilibres, qui ne sont que des meilleures réponses aux actions des autres. A noter que cette fonction s'apparente à l'opérateur *max* employé dans les algorithmes d'itération sur les valeurs.

$$f^{NashMaxSub} : S \rightarrow \hat{A}, R_i(\langle \hat{a}_i, \hat{a}_{-i} \rangle) > R_i(\langle a_i, a_{-i} \rangle) \tag{3.6}$$

	$agent_2$	
$agent_1$	(2,10)	(0,10)
	(1,3)	(5,9)

	$agent_1$	$agent_2$
$f^{NashMaxTot}$	(5,9)	(5,9)
$f^{NashMaxSub}$	(5,9)	(2,10)
$f^{NashPareto}$	(2,10) ou (5,9)	

FIGURE 3.4 – Equilibres de Nash déterminés avec les différentes fonctions de sélection.

La figure 3.4 nous montre une matrice de paiement ainsi que les différents équilibres de Nash qui peuvent être obtenus avec les trois critères de sélection proposés. Cette matrice possède deux équilibres de Nash, apportant respectivement les gains suivant $(5, 9)$ et $(2, 10)$. L'équilibre $(5, 9)$ est celui auquel adhèrent les deux agents, en voulant maximiser leur gain global. Dans les autres cas, les agents choisissent des équilibres différents. En cherchant à maximiser le gain individuel, le premier agent choisit de ce fait l'équilibre $(5, 9)$ et le deuxième agent choisit l'équilibre $(2, 10)$. Par contre, les deux équilibres sont des équilibres non Pareto dominés, offrant la possibilité de choisir l'un ou l'autre.

Nous présentons, un deuxième exemple (figure 3.5), où les agents s'accordent à sélectionner le même équilibre quelque soit le critère de sélection choisi. Le jeu possède deux équilibres de Nash $(1,1)$ et $(2,2)$. Cependant, l'équilibre $(2,2)$ domine suivant tous les critères l'équilibre $(1,1)$, en effet il maximise le gain individuel, le gain global et il est également un équilibre non Pareto dominé.

	$agent_2$	
$agent_1$	(1,1)	(1,0)
	(0,1)	(2,2)

	$agent_1$	$agent_2$
$f^{NashMaxTot}$	(2,2)	
$f^{NashMaxSub}$	(2,2)	
$f^{NashPareto}$	(2,2)	

FIGURE 3.5 – Equilibres de Nash déterminés avec les différentes fonctions de sélection.

3.1.3 Algorithme pour la résolution des jeux stochastiques

Il nous a paru nécessaire de modifier la fonction f, fournissant aux agents le même équilibre de Nash du jeu matriciel courant. Cette fonction impose une centralisation de l'algorithme, tout en supprimant les problèmes de coordination. En fait, cette fonction donne aux deux agents la politique à jouer, à savoir un équilibre de Nash par état. L'objectif de notre travail est de modifier cette fonction f afin qu'elle puisse trouver tous les équilibres de Nash présents dans les jeux, et en sélectionner un.

Description

L'algorithme SGInfiniteVI [Hamila *et al.*, 2010a] combine deux domaines de recherche distincts : la planification mono-agent et la théorie des jeux. Sa structure est semblable à celle des algorithmes de Shapley et de Kearns, elle comprend une première composante "Différence temporelle" nécessaire à la gestion de la multiplicité d'états et une deuxième composante "Calcul d'équilibre" indispensable à la recherche de la meilleure action dans la matrice jouée. L'algorithme SGInfiniteVI, considère dans la deuxième composante un calcul décentralisé, où chaque agent transforme les états en des matrices de jeu et calcule ses propres équilibres. En tenant compte des améliorations et des solutions proposées, l'algorithme tentera au mieux d'éviter les problèmes de coordination liés à la possibilité de sélection d'équilibres différents.

Algorithme 11: SGInfiniteVI pour la résolution des jeux stochastiques à somme générale et à plusieurs agents.

Entrées :

- Un jeu stochastique
- $\gamma \in [0, 1]$
- $\epsilon \geq 0$
- $\varepsilon > 0$
- $f \begin{cases} f^{NashMaxTot} & \text{maximiser le gain global.} \\ f^{NashMaxSub} & \text{maximiser le gain individuel.} \\ f^{NashPareto} & \text{sélectionner un équilibre non Pareto dominé.} \end{cases}$

Sorties : Une politique d'actions π

Initialisation
$t \leftarrow 0$
$V^f \leftarrow 0$
ListeNash $\leftarrow \emptyset$
répéter
  pour $s \in S$ **faire**
    pour $a \in A$ **faire**
      pour $k \in 1 \ldots |Ag|$ **faire**
        $M(s,t) \; = \; R_k(s,a) \; + \; \gamma \; \sum_{s' \in S} \; P(s,a,s') \; V_k^f(M(s',a,t-1))$
    ListeNash $\leftarrow$ EMR$(M(s,t))$
    $V_k^f \leftarrow$ Utilité^f(ListeNash))
    $\pi(s,t) \; = \; f(M(s,t))$
  $t \leftarrow t \; + \; 1$
jusqu'à $max_{s \in S} |V^f(s,t) - V^f(s,t-1)| < \epsilon$;

L'algorithme (algorithme 11) cherche un équilibre de Nash pour chaque matrice $M(s)$ correspondant à la paire état-itération. Chaque jeu matriciel est construit en utilisant une fonction de valeur, qui reflète :

- l'utilité immédiate de choisir une action,
- l'utilité à long terme en suivant un équilibre à partir de l'état suivant.

Dans un premier temps, l'algorithme commence par assigner des valeurs initiales à $V(s)$. Ensuite, à chaque itération t une mise à jour est appliquée à ces valeurs, pour cela la matrice du jeu $M(s)$ doit être construite à partir des Q-valeurs correspondant à

l'état s. A la différence de l'algorithme de Kearns et al., les Q-valeurs sont construites instantanément au moment du remplissage de la matrice $M(s)$; et par conséquent elles ne sont pas sauvegardées, la fonction de valeur V est suffisante pour caractériser les différents états du jeu stochastique. A partir de la matrice $M(s)$, l'algorithme établit une liste de tous les équilibres de Nash existants. S'il n'y a pas d'équilibre, l'algorithme se tourne alors vers les équilibres de Nash approximatifs.

Notons que l'algorithme fonctionne de façon similaire à l'algorithme d'itération sur les valeurs pour les MDP, avec l'opérateur *max* remplacé par l'opérateur d'Utilité^f (ligne 11 de l'algorithme 11). L'opérateur d'utilité, muni d'un critère de sélection, choisit un équilibre parmi la liste des équilibres de Nash calculés à l'étape précédente et retourne l'unique utilité correspondant à la solution de ce jeu. Cette utilité est le gain que perçoit l'agent avec l'équilibre sélectionné.

L'algorithme prend comme paramètres : un jeu stochastique, γ (le degré d'importance du future), ϵ (la condition d'arrêt), un critère de sélection en cas de multiplicité d'équilibres et retournera une politique d'actions pour l'agent. Cette politique assure au moins un ε-équilibre de Nash pour chaque état du jeu. L'algorithme s'arrête lorsque la différence entre $V^f(s,t)$ et $V^f(s,t-1)$ est inférieure à ϵ.

Convergence

La convergence constitue un réel obstacle pour ce genre d'algorithme. En effet, Zinkevich et al. [Zinkevich *et al.*, 2005] ont montré que les algorithmes de type itération sur les valeurs peuvent, dans certains cas, ne pas converger. Hu et Wellman [Hu et Wellman, 2003] ont prouvé que la convergence dépend fortement de la structure de la matrice et que l'équilibre de Nash en tant qu'opérateur ne remplace pas systématiquement l'opérateur de contraction *max*. Les mêmes auteurs ont défini deux hypothèses sous lesquelles, un algorithme d'itération sur les valeurs converge :

1. L'équilibre de Nash sélectionné doit être un optimum global.

   Définition 3.2 *Une stratégie jointe est un optimum global pour le jeu matriciel $M(s)$, si chaque agent reçoit son gain le plus élevé dans ce jeu.*

2. Si l'équilibre de Nash n'est pas un optimum global, alors il doit être une situation où chaque agent reçoit le gain le plus élevé lorsque l'autre agent s'écarte de l'équilibre de Nash.

En s'appuyant sur ces deux hypothèses et sur les travaux de [Szepesvári et Littman, 1998], Hu et Wellman [Hu et Wellman, 2003] ont prouvé que l'équilibre de Nash n'est pas un opérateur d'expansion et par conséquent il peut être considéré comme un opérateur

de pseudo-contraction, garantissant ainsi la convergence de l'algorithme. Comme notre algorithme est basé sur des fonctions de sélection d'équilibre de Nash, nous pouvons alors réutiliser les hypothèses adoptées par Hu et Wellman pour montrer sa convergence vers une politique d'actions.

Complexité de l'algorithme SGInfiniteVI

Une première évaluation théorique de notre algorithme, nous amène à étudier sa complexité aux niveaux temporel et spatial.

Complexité temporelle Pour la recherche des équilibres, l'algorithme ne traite que les équilibres de Nash en stratégies pures. Il a été prouvé que, même avec des conditions très restrictives sur les stratégies des joueurs, déterminer si un jeu possédait un équilibre de Nash pur est NP-complet [Conitzer et Sandholm, 2008, Fischer *et al.*, 2006]. Par conséquent, nous pourrons affirmer que le temps d'exécution de l'algorithme SGInfiniteVI ne peut pas être polynomial (la fonction utilisée pour calculer un équilibre de Nash est elle-même non polynomiale). Comme tout algorithme de type itération sur les valeurs, la complexité en temps de l'algorithme dépend de la complexité de chaque itération et du nombre d'itérations.

Une itération nécessite : $|S|^2 \times |A|^{|Ag|} \times$ coût de calcul d'un équilibre de Nash. Si on se réfère à l'algorithme EMR défini plus haut, calculer un équilibre consiste en deux étapes. La première étape parcourt l'ensemble des stratégies des agents en $|Ag| \times |A|^{|Ag|}$ et la deuxième étape nécessite $|A|^{|Ag|}$. De ce fait, nous pourrons estimer le coût de calcul d'un équilibre de Nash en stratégies pures à : $|Ag| \times |A|^{|Ag|} + |A|^{|Ag|} \simeq |Ag| \times |A|^{|Ag|}$. Par conséquent, la complexité d'une itération de l'algorithme est $O(|S|^2|A|^{2|Ag|})$. De plus, les tailles des espaces d'états et d'actions sont exponentielles en fonction du nombre d'agents. Globalement, nous pensons que le temps d'exécution de notre algorithme est aussi exponentiel.

Il a été démontré que le nombre d'itérations d'un algorithme d'itération sur les valeurs est polynomial [Papadimitriou et Tsitsiklis, 1987]. Si on considère l'équilibre de Nash sous les conditions de convergence citées précédemment, il peut remplacer dans ce cas l'opérateur de maximisation et en conséquence nous pourrons affirmer que le nombre d'itérations de SGInfiniteVI est fini.

Complexité spatiale Contrairement à l'algorithme de Kearns et al., nous n'avons pas sauvegardé les matrices de jeu (ni les matrices booléennes correspondantes) en mémoire et nous avons jugé suffisant de garder seulement un couple de valeurs par état. En effet, la taille de chaque matrice est : $|A_1| \times |A_2| \times ... \times |A_{|Ag|}| = |A|$. Si chaque agent doit

conserver une matrice pour chaque état s alors le nombre total des valeurs requises serait : $|Ag| \times |S| \times |A|$. Or, si nous gardons en mémoire seulement $|Ag|$ valeurs par état, alors le nombre total de valeurs sera : $|Ag| \times |S|$. L'algorithme SGInfiniteVI est linéaire en fonction de S et Ag.

3.1.4 Application numérique

Afin d'illustrer le fonctionnement de l'algorithme, nous proposons une illustration sous forme d'application numérique. Soit un jeu stochastique illustré par la figure 3.6. Afin de simplifier l'exemple, les transitions seront déterministes, ce qui n'est pas le cas en général. Ce jeu est défini par :

- $S = \{s_0, s_1, s_2\}$
- $A_1 = \{a_{11}, a_{12}\}$ et $A_2 = \{a_{21}, a_{22}\}$
- $R_1(s_0) = 0$, $R_1(s_1) = 10$, $R_1(s_2) = 15$; $R_2(s_0) = 0$, $R_2(s_1) = 10$, $R_2(s_2) = 12$
- La fonction de transition a été définie suivant les règles suivantes :
  $P(s_0, \langle a_{12}, a_{21}\rangle, s_1) = 1$, $P(s_0, \langle a_{12}, a_{22}\rangle, s_2) = 1$, $\forall a \in A_2$ $P\langle s_0, (a_{12}, a\rangle, s_0) = 1$
  $P(s_1, \langle a_{11}, a_{21}\rangle, s_0) = 1$, $P(s_1, \langle a_{11}, a_{22}\rangle, s_2) = 1$, $\forall a \in A_2$ $P\langle s_1, (a_{11}, a\rangle, s_1) = 1$
  $P(s_2, \langle a_{11}, a_{21}\rangle, s_0) = 1$, $P(s_2, \langle a_{12}, a_{21}\rangle, s_1) = 1$, $\forall a \in A_1$ $P\langle s_2, (a, a_{22}\rangle, s_2) = 1$
- Nous avons utilisé ici le critère pondéré pour les récompenses à long terme, avec $\gamma = 0.9$.
- En cas de multiplicité d'équilibres, le critère de sélection est celui qui maximise le gain de tous les agents.

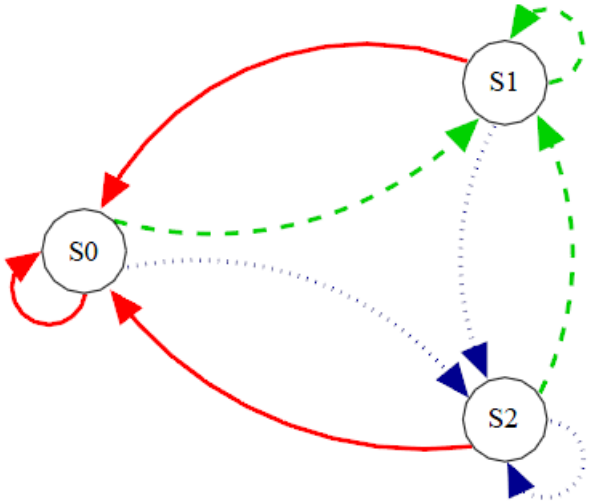


FIGURE 3.6 – Exemple de jeu stochastique.

Nous présentons les deux premières itérations de l'algorithme, incluant à chaque fois la construction des matrices, le choix de l'équilibre de Nash et la mise à jour des fonctions de valeurs.

Ainsi, à $t = 0$:

Etats	s_0			s_1			s_2		
Matrice		a_{21}	a_{22}		a_{21}	a_{22}		a_{21}	a_{22}
	a_{11}	(0,13.5)	(0,13.5)	a_{11}	(0,13.5)	(13.5,10.8)	a_{11}	(0,13.5)	(13.5,10.8)
	a_{12}	(9,9)	(13.5,10.8)	a_{12}	(9,9)	(9,9)	a_{12}	(9,9)	(13.5,10.8)
Actions	$\langle a_{12}, a_{21} \rangle$			$\langle a_{12}, a_{21} \rangle$			$\langle a_{12}, a_{22} \rangle$		
V(s)	$V_1 = 9$, $V_2 = 9$			$V_1 = 9$, $V_2 = 9$			$V_1 = 13.5$, $V_2 = 10.8$		

et à $t = 1$:

Etats	s_0			s_1			s_2		
Matrices		a_{21}	a_{22}		a_{21}	a_{22}		a_{21}	a_{22}
	a_{11}	(8.1,23.1)	(8.1,23.1)	a_{11}	(24.7,34.5)	(27.15,21.7)	a_{11}	(24.7,34.5)	(24.7,21.7)
	a_{12}	(18.1,18.1)	(27.5,21.7)	a_{12}	(18.1,18.1)	(18.1,18.1)	a_{12}	(32.2,41)	(24.7,21.7)
Actions	$\langle a_{12}, a_{22} \rangle$			$\langle a_{11}, a_{21} \rangle$			$\langle a_{12}, a_{21} \rangle$		
V(s)	$V_1 = 27.5$, $V_2 = 21.72$			$V_1 = 24.7$, $V_2 = 34.5$			$V_1 = 32.2$, $V_2 = 41$		

3.2 Amélioration de l'algorithme

Dans cette partie nous nous intéressons à l'amélioration [Hamila *et al.*, 2012] de l'algorithme SGInfiniteVI, dans le but de réduire les ressources systèmes utilisées. L'énumération des meilleures réponses, la recherche des équilibres de Nash et la multiplicité de ceux-ci, constituent un défi majeur ralentissant le déploiement des algorithmes de résolution. Ainsi, nous focalisons notre attention sur la procédure de calcul d'équilibre.

Lorsque le nombre d'agents est conséquent, il devient difficile pour chacun de considérer l'ensemble des actions jointes. Cela pouvant impliquer un coût de construction et de résolution très élevé. Pour réduire l'ensemble des actions jointes par état, la plupart des travaux (en théorie des jeux), se sont intéressés à l'étude des concepts de solutions plausibles.

3.2.1 Principe de la dominance stratégique

La dominance stratégique [Fudenberg et Tirole, 1991, Leyton-Brown et Shoham, 2008] représente un des concepts les plus utilisés, cherchant à réduire l'ensemble des actions qu'auront les agents lorsqu'ils sont confrontés à un jeu.

Définition 3.3 *Une stratégie $a_i \in A_i$ est dite strictement dominée s'il existe une autre stratégie $a'_i \in A_i$ telle que :*

$$R_i(a'_i, a_{-i}) > R_i(a_i, a_{-i}) \quad \forall a_{-i} \in A_{-i} \tag{3.7}$$

Il n'y a aucune raison pour qu'un agent rationnel décide de choisir une stratégie strictement dominée a_i si, en dépit de ce que les autres agents choisiront, il existe une autre stratégie a'_i lui apportant un meilleur gain.

La logique qu'une stratégie strictement dominée ne doit pas être jouée, peut être étendue de la manière suivante : après chaque itération du processus d'élimination, des stratégies non dominées initialement peuvent apparaître. Une seconde itération est nécessaire pour éliminer celles-ci. Ce processus d'élimination successive, appelé *Élimination Itérative des Stratégies Strictement Dominées*, peut être effectué par chaque joueur. Il garantit de trouver au moins un équilibre de Nash, si celui-ci existe.

Théorème 3.1 *S'il n'existe qu'une seule et unique solution provenant du processus d'élimination itérative des stratégies strictement dominées, alors cette solution est un équilibre de Nash (voir preuve [Apt et Erich, 2011]).*

Afin de réduire davantage l'espace d'actions, on peut relaxer le principe de dominance : jusqu'à présent nous avons seulement éliminé les stratégies strictement dominées, nous proposons d'éliminer les stratégies n'apportant pas plus de gain qu'une autre.

Définition 3.4 *Une stratégie $a_i \in A_i$ est dite faiblement dominée s'il existe une autre stratégie $a'_i \in A_i$ telle que :*

$$R_i(a'_i, a_{-i}) \geqslant R_i(a_i, a_{-i}) \quad \forall a_{-i} \in A_{-i} \tag{3.8}$$

L'idée d'éliminer les stratégies faiblement dominées semble être presque aussi attrayante que celle éliminant les stratégies strictement dominées. En effet, il semble tout à fait plausible qu'un joueur rationnel préférera toujours choisir une stratégie non dominée plutôt qu'une stratégie faiblement dominée. En plus, en éliminant les stratégies faiblement dominées, on élimine aussi toutes les stratégies strictement dominées et cela devra permettre d'obtenir des matrices plus compactes.

Cependant, cette procédure présente deux inconvénients majeurs :

– Contrairement aux cas des stratégies strictement dominées, le jeu résultant de l'élimination des stratégies, seulement faiblement dominées, peut dépendre de l'ordre suivant lequel les éliminations se feront. Etudions les stratégies dominées du jeu matriciel de la figure 3.7 :

		$agent_2$	
		a_{21}	a_{22}
$agent_1$	a_{11}	(0,0)	(0,0)
	a_{12}	(0,1)	(1,0)

FIGURE 3.7 – Ordre d'élimination des stratégies dominées.

– Pour l'$agent_1$, la stratégie a_{12} domine faiblement la stratégie a_{11}.

– Pour l'$agent_2$, la stratégie a_{21} domine faiblement la stratégie a_{22}.

Eliminons dans un premier temps la stratégie dominée de l'$agent_1$, on élimine donc a_{11}. Après cette élimination, la stratégie a_{21} de l'$agent_2$ domine faiblement la stratégie a_{22}. On obtient une stratégie jointe unique en $\langle a_{12}, a_{21}\rangle$. Eliminons à présent la stratégie dominée de l'$agent_2$ en premier. On élimine donc la stratégie a_{22}. Après cette élimination, l'$agent_1$ n'a plus de stratégie dominée. On obtient donc deux stratégies jointes en : $\langle a_{12}, a_{21}\rangle$ et $\langle a_{11}, a_{21}\rangle$. On voit que nous n'avons pas obtenu le même résultat suivant l'ordre d'élimination des stratégies faiblement dominées. Cela rend difficile la prédiction qui résulte d'une élimination des stratégies faiblement dominantes. Si nous considérons toujours l'hypothèse que les agents ne communiquent pas, il n'y aura donc aucune raison (sans communiquer) que les joueurs s'accordent sur la stratégie à éliminer en premier.

– En éliminant les stratégies faiblement dominées, cela peut écarter d'autres équilibres de Nash présents dans le jeu. En effet, même si le résultat d'une élimination de stratégies faiblement dominées est unique, il se peut qu'un équilibre de Nash soit éliminé, comme l'illustre l'exemple de la figure 3.8. Dans cet exemple, le couple de valeurs $(2, 2)$ est éliminé alors qu'il représente un équilibre de Nash.

$M(s)$

	$agent_2$	
$agent_1$	(1,1)	(2,0)
	(0,2)	(2,2)

	$agent_2$	
$agent_1$	(1,1)	(2,0)
	(0,2)	(2,2)

$BM(s)$

	$agent_2$	
$agent_1$	(Vrai,Vrai)	
		(Vrai,Vrai)

	$agent_2$	
$agent_1$	(Vrai,Vrai)	
		(Vrai,Vrai)

FIGURE 3.8 – L'élimination des stratégies faiblement dominées peut entraîner l'omission d'autres équilibres présents dans le jeu.

Une question importante dans ce contexte est la suivante : quelle est la complexité de déterminer si une stratégie donnée peut être éliminée ? Dans ce contexte, on pourrait se référer au théorème de [Porter *et al.*, 2008] affirmant que :

– l'élimination des stratégies strictement dominées est de complexité P,
– l'élimination des stratégies faiblement dominées est NP-complet.

Malgré les inconvénients cités, nous accordons un réel intérêt à l'élimination des stratégies faiblement dominées en terme d'élagage de l'espace des actions jointes. Ce concept est généralement accepté comme une procédure valable de simplification de l'analyse d'un jeu [Varian *et al.*, 2008]. Le chapitre suivant sera consacré en partie, à l'étude de l'impact que pourrait avoir l'élimination des deux types de stratégies autant sur le niveau des ressources utilisées que sur le comportement des agents.

3.2.2 Extension de l'algorithme SGInfiniteVI

Comme mentionné ci-dessus, nous proposons une extension de l'algorithme précédent, en tenant compte des différents types d'élimination des stratégies dominées. En effet, l'algorithme connaît ses limites dès l'augmentation de l'instance du problème. Notre objectif consiste à améliorer l'algorithme SGInfiniteVI, afin de parcourir efficacement l'ensemble des solutions et de permettre ainsi de réduire le temps de calcul, de réduire l'espace mémoire utilisé, tout en préservant la qualité de la solution et un comportement global cohérent.

Algorithme 12: Elimination Itérative des Stratégies Dominées (EISD)

Entrées : M(s), type de stratégies à éliminer : faiblement ou strictement dominées.

```
1 pour k ∈ 1...|Ag| faire
2     pour stratCandidat ∈ A_k faire
3         Dominée ← Vrai
4         pour stratAlternat ∈ A_k faire
5             si stratCandidat non-dominée par stratAlternat alors
6                 Dominée ← Faux
7         si Dominée alors
8             Eliminer stratCandidat de A_k
```

Sorties : $|M'(s)| \leqslant |M(s)|$

Nous présentons tout d'abord l'algorithme EISD (algorithme 12), qui effectue l'élimination itérative des stratégies dominées. L'algorithme prend en paramètre une matrice

$M(s)$ de dimension quelconque et renvoie une matrice $M'(s)$ de taille inférieure ou égale. Le processus d'élimination est appliqué itérativement jusqu'à ce que chaque joueur ait une seule stratégie restante, ou un ensemble de stratégies dont aucune n'est faiblement dominée. A noter que, chaque joueur cherchera à réduire ses matrices, en éliminant non seulement ses stratégies dominées mais aussi celles des autres. Ce raisonnement repose sur une forte hypothèse, celle de la rationalité des autres joueurs, une hypothèse précédemment rencontrée avec le concept d'équilibre de Nash.

A ce stade, nous ne précisons pas s'il s'agit d'éliminer des stratégies faiblement dominées ou des stratégies strictement dominées. En effet, la difficulté d'estimer l'issue de cette technique sur le comportement des agents ainsi que sur le déroulement du processus de résolution, nous oblige à considérer à ce stade d'étude les deux types de stratégies. Les expérimentations devront nous fournir les éléments de réponse nécessaires à ce choix.

Nous avons vu que le processus d'élimination des stratégies dominées ne permet pas forcément de trouver une solution pour tous les jeux matriciels, l'intégrer à l'algorithme SGInfiniteVI requiert donc un raisonnement supplémentaire pour aboutir à un équilibre de Nash. En effet, suivant la matrice $M'(s)$ renvoyée par l'algorithme EISD, il existe deux types de post traitements :

– Si la matrice est réduite à un couple unique de valeurs ($|M'(s)| = 1$) : alors cette solution constitue un équilibre de Nash (d'après le théorème **3.1**) et dans ce cas l'algorithme principal passe directement à la mise à jour des fonctions de valeurs (inutile de chercher d'autres équilibres), voir l'exemple de la figure 3.9 où l'élimination des stratégies dominées trouve une solution unique au jeu.

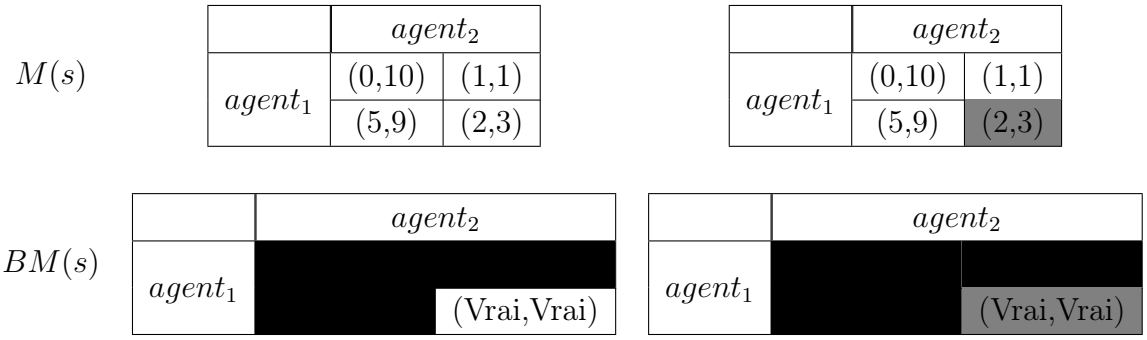


FIGURE 3.9 – Equilibre de Nash calculé par élimination des stratégies dominées.

– S'il reste encore plusieurs couples de valeurs dans la matrice ($|M'(s)| > 1$) : dans ce cas il est possible d'affiner la sélection d'équilibre afin de privilégier une stratégie plutôt qu'une autre. Ainsi, nous reprenons l'algorithme d'énumération des meilleures réponses et la fonction *Utilité* pour trouver un équilibre de Nash dans la nouvelle

matrice $M'(s)$. En cas de multiplicité d'équilibres, nous utiliserons une des fonctions de sélection d'équilibre définies dans la section précédente, à savoir : $f^{NashMaxTot}$, $f^{NashMaxSub}$ ou $f^{NashPareto}$ (voir figure 3.10).

$M(s)$

	$agent_2$	
$agent_1$	(0,10)	(1,10)
	(5,9)	(2,3)

	$agent_2$	
$agent_1$	(0,10)	(1,10)
	(5,9)	(2,3)

$BM(s)$

	$agent_2$	
$agent_1$		
	(Vrai,Vrai)	(Vrai,Faux)

	$agent_2$	
$agent_1$		
	(Vrai,Vrai)	(Vrai,Faux)

FIGURE 3.10 – Equilibre de Nash calculé par (1) élimination des stratégies dominées et (2) par énumération des meilleures réponses.

L'ajout d'une procédure basée sur la dominance stratégique à l'algorithme SGInfiniteVI a pour objectif de réduire les dimensions des matrices traitées et de générer ainsi un gain en temps et mémoire. En effet, pour chercher un équilibre de Nash, la première version de l'algorithme consiste à parcourir exhaustivement chaque matrice et à comparer les équilibres de Nash entre eux pour n'en sélectionner finalement qu'un seul. Le gain que peut apporter l'élimination des stratégies dominées n'est pas systématique, il dépend de la taille de la matrice $M'(s)$ et de la nécessité du post traitement. Nous distinguons deux cas où il est possible d'observer une amélioration du temps de calcul des équilibres, si après élimination des stratégies dominées :

- la matrice ne comporte plus qu'un seul couple de valeurs : alors il y aura diminution considérable du temps de calcul car l'algorithme SGInfiniteVI ne fera pas appel aux fonctions EMR et $Utilité$.
- la matrice comporte encore plusieurs couples de valeurs : alors il y aura seulement un gain en temps d'exploration lors de l'exécution de la fonction EMR car le nombre de stratégies à évaluer sera moins important.

Description de l'algorithme Le principe de fonctionnement de l'algorithme SGInfiniteVI présenté à l'algorithme 13 est fondamentalement identique à celui présenté avec l'algorithme 11 et consiste à parcourir itérativement l'ensemble des états et à chercher pour chaque état un équilibre de Nash avec un critère de sélection défini. La nouveauté réside dans l'intégration de la procédure éliminant les stratégies strictement/faiblement dominées. Ainsi, après avoir construit la matrice $M(s)$, on fait appel à la fonction EISD en précisant le type de stratégies qu'on veut éliminer (ligne 10).

Algorithme 13: SGInfiniteVI pour la résolution des jeux stochastiques à somme générale et à plusieurs agents, avec élimination des stratégies dominées.

Entrées :

- Un jeu stochastique
- $\gamma \in [0, 1]$
- $\epsilon \geq 0$
- $\varepsilon > 0$
- $f \begin{cases} f^{NashMaxTot} & \text{maximiser le gain global.} \\ f^{NashMaxSub} & \text{maximiser le gain individuel.} \\ f^{NashPareto} & \text{sélectionner un équilibre non-Pareto-dominé.} \end{cases}$
- $EISD \begin{cases} \text{stratégie strictement dominée} \\ \text{stratégie faiblement dominée} \end{cases}$

Sorties : Une politique d'actions π

Initialisation
$t \leftarrow 0$
$V^f \leftarrow 0$
ListeNash $\leftarrow \emptyset$
répéter
  pour $s \in S$ **faire**
    pour $a \in A$ **faire**
      pour $k \in 1 \ldots |Ag|$ **faire**
        $M(s,t) = R_k(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \, V_k^f(M(s',a,t-1))$
    $M'(s,t) = EISD(M(s,t))$
    si $|M'| = 1$ *//Matrice réduite au maximum* **alors**
      $\pi(s,t) = f^{extract}(M'(s,t))$
      $V_k^f \leftarrow$ Utilité^f(ListeNash))
    sinon
      ListeNash $\leftarrow$ EMR($M'(s,t)$)
      $V_k^f \leftarrow$ Utilité^f(ListeNash))
      $\pi(s,t) = f(M'(s,t))$
  $t \leftarrow t + 1$
jusqu'à $max_{s \in S} |V^f(s,t) - V^f(s,t-1)| < \epsilon$;

Deux issues sont possibles, si le résultat est une solution unique (ligne 10) alors l'algorithme passe directement aux étapes d'extraction de l'équilibre ($f^{extract}$) et de mise à jour des fonctions de valeurs V_k. Par contre, si $M'(s)$ comporte encore plusieurs entrées (ligne 14), alors dans ce cas l'algorithme fait appel à la procédure ordinaire consistant à énumérer les meilleures réponses, à lister tous les équilibres de Nash et à appliquer le critère de sélection.

L'évolution de l'algorithme permettra donc un gain en temps de calcul et facilitera également le calcul d'équilibre, en diminuant le nombre de stratégies à chaque étape du jeu stochastique. Si nous considérons la complexité de la version initiale de l'algorithme ($O(|S|^2|A|^{2|Ag|})$) nous pouvons dire que l'amélioration a porté sur la réduction de la partie ($|A|^{2|Ag|}$). Il reste à savoir, si l'élimination des stratégies faiblement dominées dégrade la qualité de la solution vu qu'il y a perte de certains équilibres. La partie expérimentale de cette thèse, fournira les éléments de réponse à la question posée.

3.3 Conclusion

Ce chapitre a été l'occasion de présenter notre contribution dans le domaine de la planification multi-agents en utilisant le modèle des jeux stochastiques à somme générale. Nos travaux s'inspirent des algorithmes de Shapley et de Kearns et al. et proposent une nouvelle approche de résolution. Ce chapitre a été organisé suivant deux parties, la première partie présentait l'algorithme SGInfiniteVI et la deuxième proposait une amélioration de celui-ci.

Dans un premier temps, il était primordial d'apporter des solutions aux différentes questions qui ont été posées par les travaux précédents. Parmi ces questions, nous citons : le calcul d'équilibre et la centralisation impliquée par la fonction de sélection f. Nous avons montré qu'il s'agissait d'un problème difficile en terme de coordination et que vouloir sélectionner un équilibre unique pour tous les agents impliquait plusieurs contraintes difficilement résolvables. Notre approche assure une meilleure autonomie pour chaque agent et propose différents critères de sélection pouvant focaliser les agents vers le choix d'un même équilibre. Nous avons discuté de la complexité de l'algorithme et avons montré que le calcul des équilibres de Nash rendait la complexité de l'algorithme SGInfiniteVI non polynomiale. A la fin de cette première partie, nous avons abordé l'aspect de la convergence de l'algorithme et nous avons évoqué les conditions nécessaires à cette opération.

Dans un deuxième temps, il était question d'améliorer l'algorithme proposé. En effet, le calcul et la sélection des équilibres de Nash, induisaient des problèmes autant sur la

complexité de l'algorithme que sur la coordination entre les agents. Nous avons montré, que le nombre d'actions croît exponentiellement en fonction du nombre d'agents et que l'évaluation de toutes les actions à chaque étape du jeu stochastique peut se révéler fastidieuse. Notre approche, a consisté à réduire l'ensemble des actions possibles en se basant sur le concept de dominance stratégique. Ce procédé, bien que connu dans la théorie des jeux, n'a pas été réellement étudié avec le modèle des jeux stochastiques. L'élimination des stratégies dominées peut se révéler bénéfique, (1) en réduisant le temps de calcul et l'espace mémoire utilisé durant la phase de planification et (2) en limitant la multiplicité des équilibres de chaque jeu matriciel.

Le chapitre suivant, se propose de valider l'algorithme SGInfiniteVI sur un scénario d'interaction entre des agents.

Chapitre 4

Résultats expérimentaux

Sommaire

4.1 Un jeu de grille 89

4.1.1 Description 90

4.1.2 La modélisation sous forme d'un jeu stochastique 91

4.2 Evaluation de la version initiale de l'algorithme 92

4.2.1 Principe de fonctionnement 92

4.2.2 Evaluation des fonctions de sélection d'équilibre 93

4.2.3 Expériences complémentaires 105

4.3 Evaluation de l'algorithme avec élimination des stratégies dominées 109

4.3.1 Evaluation quantitative 110

4.3.2 Evaluation qualitative 114

4.4 Conclusion 117

Nous allons dans ce chapitre, développer différentes expérimentations permettant d'étudier et de tester l'efficacité de notre approche ainsi que la qualité des solutions obtenues. Des expérimentations seront menées sur les deux versions de l'algorithme précédemment présenté. Le cadre applicatif, nécessaire aux expérimentations, est réalisé par simulation d'un scénario de coordination incluant plusieurs agents. Nous nous intéresserons plus particulièrement à étudier successivement des scénarios de coopération et de non-coopération entre les agents. L'évaluation empirique de l'algorithme portera sur l'analyse des ressources système utilisées durant la phase de planification, ainsi qu'à la qualité des politiques calculées. Ces expérimentations nous permettront alors de conclure sur l'adéquation de notre approche aux attentes initialement formulées.

Introduction

Ce chapitre a pour but d'illustrer l'efficacité de l'approche que nous avons proposée précédemment. Rappelons que notre travail a eu pour objectif la mise en place d'un processus de planification multi-agents, pouvant être utilisé afin de résoudre des problèmes décisionnels tels que ceux rencontrés avec le modèle des jeux stochastiques.

Pour ce faire, nous avons choisi pour la validation, un problème de coordination entre agents dans un environnement de type grille, que nous avons modélisé sous forme d'un jeu stochastique. Ainsi, dans la première partie de ce chapitre, nous allons accorder de l'importance à montrer le mode de fonctionnement de l'algorithme sur le problème de déplacement d'objets (relation entre les matrices de paiement, l'équilibre de Nash et les actions concrètes).

La deuxième partie, sera consacrée à l'évaluation de la version initiale de l'algorithme. Cette évaluation, tentera de déterminer la meilleure fonction de sélection d'équilibre en tenant compte de la qualité de la solution obtenue. Pour ce faire, différents critères d'évaluation seront définis et les expérimentations seront faites suivant des scénarios avec et sans respect de l'équilibre de Nash. Des expériences complémentaires viseront à comparer l'approche utilisée avec une approche centralisée basée sur le modèle MMDP. Globalement, nous allons chercher à vérifier que les solutions qui ont été précédemment proposées répondent bien aux exigences initialement recensées au chapitre 3.

Les problèmes envisagés pouvant être de taille importante, il a été nécessaire de mettre en place un algorithme de résolution efficace. Ce sujet sera traité dans la troisième partie de ce chapitre. Nous nous intéressons à l'évaluation de la seconde version de l'algorithme, qui consiste à réduire les ressources de calcul utilisées en utilisant le concept de dominance stratégique. En premier lieu, l'efficacité des deux procédures, éliminant respectivement les

stratégies strictement dominées et faiblement dominées, sera testée. En deuxième lieu, une technique sera rajoutée visant une réduction de l'espace mémoire. L'évaluation de cette deuxième version de l'algorithme SGInfiniteVI sera assurée par une combinaison de tests à la fois qualitatifs et quantitatifs.

4.1 Un jeu de grille

Le transport collectif est une des tâches les plus classiques de la robotique collective. Le problème de déplacement d'objets [13] dans une grille, a été initialement proposé par Seuken et Zilberstein [Seuken et Zilberstein, 2007] et décrit le comportement de deux agents déménageurs qui doivent ranger des objets en se déplaçant dans un environnement de type grille. L'environnement contient différents types d'objets, de petite et grande taille. Les grands objets nécessitent que les deux agents se coordonnent pour pouvoir les pousser. Le but est d'amener chaque objet dans la zone but qui lui est associée (en haut de la grille, figure 4.1). Les agents ont comme possibilité pour se déplacer : d'avancer (*Fwrd*), de tourner à droite (*Right*), à gauche (*Left*), ou de rester sur place (*Stay*). A chaque fois qu'un agent exécute une action, il reçoit une observation parmi les cinq suivantes : case vide, mur, autre agent, petit objet ou grand objet.

FIGURE 4.1 – Le problème de déplacement d'objets par des agents.

13. en anglais : Cooperative multi-agent box-pushing

4.1.1 Description

Afin d'adapter le problème à un contexte d'observabilité complète, nous supprimons l'ensemble des observations et nous supposons que les agents savent avec certitude l'état dans lequel ils sont. Avec cette extension du problème, nous admettons que :

- tous les objets ont la même taille et chaque agent est capable de transporter tout objet sans l'aide d'un autre agent (figure 4.2),
- les agents peuvent avoir des intentions de transporter le même objet.

Cette dernière hypothèse, laisse supposer que les agents peuvent se concurrencer pour le transport d'un objet donné et par conséquent avoir plus de liberté de décision, tout en ayant un objectif commun (le rangement de tous les objets dans la boîte). L'objectif du jeu [Hamila *et al.*, 2009], consiste à déplacer les objets en un nombre minimal d'étapes. Les agents ne partagent au préalable aucun plan, ne communiquent pas entre eux et lorsqu'ils doivent prendre des décisions, ils doivent éviter tout conflit (occuper la même place, porter le même objet, etc.).

Malgré la simplicité apparente du problème, ce jeu de grille possède tous les éléments clés d'un jeu dynamique, en ayant des actions, des transitions et des récompenses immédiates et à long terme. Ce problème est particulièrement intéressant, car il est très flexible dans le sens où il peut facilement être modifié en variant la taille de la grille, le nombre d'objets et le nombre d'agents.

FIGURE 4.2 – Un exemple de scénario : deux agents, quatre objets et une boîte de rangement.

4.1.2 La modélisation sous forme d'un jeu stochastique

Le jeu présenté ci-dessus peut être modélisé sous forme d'un jeu stochastique. Rappelons que le modèle des jeux stochastiques est défini par le tuple (cf. chapitre 1.4.1) :

$$< Ag, \{A_i : i = 1 \dots |Ag|\}, \{R_i : i = 1 \dots |Ag|\}, S, P >$$

- Ag : nombre d'agents.
- A_i : chaque agent i possède un ensemble de cinq actions : {(H)aut, (B)as, (G)auche, (D)roite, (N)e-rien-faire}. Les actions *porter* et *déposer* sont considérées comme implicites (voir fonction de transition).
- R_i : la fonction de récompense a été définie avec les valeurs arbitraires suivantes :

$$\begin{cases} +100 & \text{lorsque tous les objets sont transportés dans la boîte.} \\ -200 & \text{lorsque l'agent } i \text{ se déplace vers une case occupée par un autre agent.} \\ -200 & \text{lorsque l'agent } i \text{ transporte un objet et tente d'en transporter un autre.} \end{cases}$$

- S : les états du problème sont des tuples composés de :
  - la position de chaque agent dans la grille.
  - l'état interne de chaque agent (transporte un objet ou non).
  - la position de chaque objet dans la grille.
  - l'état interne d'un objet (déposé dans la boîte ou non).
- P : cette fonction représente la réponse de l'environnement face aux actions prises par les agents. Elle est définie en fonction des cas suivants :
  - un agent ne peut transporter plus d'un objet à la fois.
  - si un agent se déplace sur une case contenant un objet, il doit alors impérativement transporter cet objet (action *porter*).
  - si une action amène un objet dans la boîte alors cet objet est supprimé de la grille et l'agent (qui l'a transporté) est marqué comme étant vide (action *déposer*).
  - les agents se déplacent dans une grille toroïdale de dimensions $n \times n$, de manière à ce que les agents passent d'un côté à un autre de la grille en un seul pas.

En dépit de la simplicité du jeu, nous notons que l'espace d'états n'est pas pour autant petit (figure 4.3). La complexité est la suivante : $(2\ n^2)^{|Ag|}\ *\ 2^O$ où $|Ag|$ représente le nombre d'agents, O est le nombre d'objets et n^2 est le nombre de cases dans la grille. Par exemple, pour une mission incluant 3 agents, 3 objets et avec une taille de grille de 8×8, le nombre total d'états dépasse les 16 millions.

Durant les prochaines sections, nous présentons les résultats obtenus en appliquant l'algorithme SGInfiniteVI au jeu de grille présenté. Un simulateur a été implémenté en

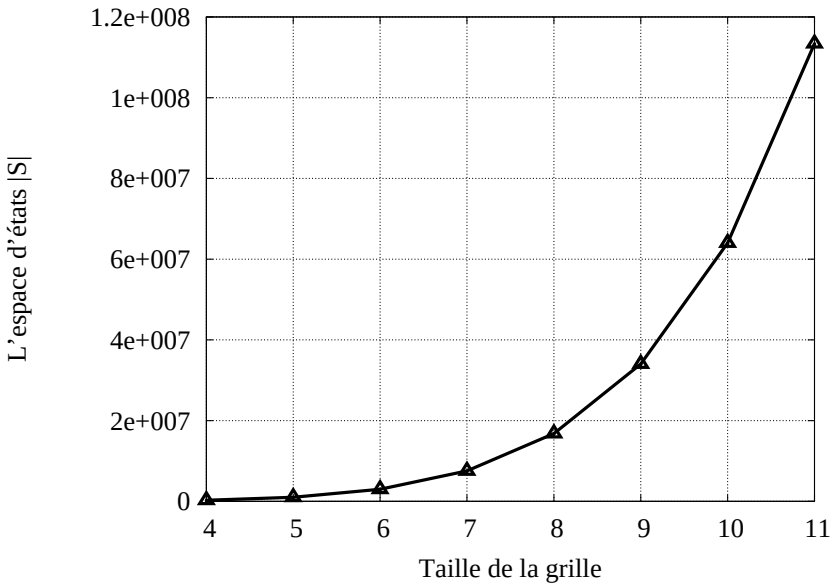


FIGURE 4.3 – Evolution de l'espace d'états en fonction de la taille de la grille (scénario avec 3 agents et 3 objets).

langage JAVA et les expériences ont été faites sur une machine Quad-core 2,8 GHz ayant 4 Go de mémoire vive. La validation comportera deux parties : la première s'attachera à évaluer la version initiale de l'algorithme avec les différentes fonctions de sélection d'équilibre. La deuxième partie, s'intéressera aux techniques utilisées pour l'amélioration de l'algorithme ainsi qu'à leurs effets sur la qualité de la solution.

4.2 Evaluation de la version initiale de l'algorithme

Dans cette première phase d'évaluation, nous verrons d'abord un exemple simple expliquant le principe de fonctionnement de l'algorithme avec le problème de déplacement d'objets. Ensuite, nous nous intéresserons à l'évaluation des fonctions de sélection d'équilibre, avec différents critères d'évaluation et deux scénarios de coopération.

4.2.1 Principe de fonctionnement

L'algorithme SGInfiniteVI fournit une politique d'actions pour l'agent qui l'exécute, en se basant sur sa propre vision des matrices de gain. Mais, pour des raisons de clarté et de lisibilité (car il faut deux matrices par état, une pour chaque agent), nous présentons dans cette section une illustration de la version centralisée de l'algorithme SGInfiniteVI avec une seule matrice pour tous les agents.

La figure 4.4 montre comment les deux agents ($agent_1$ et $agent_2$) se coordonnent, pour éviter les collisions et atteindre leur objectif. Le jeu commence par un état initial s_0 choisi au hasard. Après avoir observé l'état courant, les agents choisissent simultanément leurs actions et découvrent ainsi un nouvel état. Le jeu s'arrête quand tous les objets sont placés dans la boîte de rangement. Sur la figure, chaque état du monde a été représenté avec une "capture d'écran", la matrice de gain correspondante [14] et l'équilibre de Nash qui lui a été sélectionné. En cas de multiplicité d'équilibres, nous supposons que c'est le premier équilibre maximisant le gain global qui sera choisi. Finalement, le couple de valeurs (dans la matrice) représentant l'équilibre de Nash sélectionné, est marqué en gris.

L'exemple illustre les trois étapes qui ont été nécessaires aux agents pour le rangement de deux objets. Ces étapes sont les suivantes :

- étape s_0 : l'$agent_2$ se déplace vers la droite (pour prendre un objet) et l'$agent_1$ se déplace à l'approche de l'objet restant. L'équilibre choisi dans cette étape est important, car il évite une concurrence sur les objets à transporter et définit une mission pour chaque agent.
- étape s_1 : l'équilibre choisi pour cette situation, confirme un comportement cohérent des agents. En effet, si l'$agent_2$ essaie de maximiser ses gains alors il n'ira jamais (en haut) vers une case déjà occupée, mais plutôt vers le bas où il pourra déposer l'objet dans la boîte de rangement (grille toroïdale). Par conséquent, l'$agent_1$ peut transporter en toute sécurité l'objet restant, en ayant la certitude que l'autre agent n'a pas l'intention de le prendre.
- étape s_2 : l'$agent_2$ libère la case de rangement pour que l'$agent_1$ puisse placer son objet. Ainsi, tous les objets sont rangés et le but de la mission est atteint.

A noter qu'à partir de la quatrième étape, les agents se retrouvent sans but et dans ce cas l'algorithme fournit simplement des actions évitant les conflits.

4.2.2 Evaluation des fonctions de sélection d'équilibre

Dans cette partie, nous discutons des résultats expérimentaux de l'algorithme SGInfiniteVI et nous déduisons empiriquement la meilleure fonction de sélection d'équilibre.

Les tests ont été effectués sur 200 politiques d'actions [15]. Pour chacune de ces politiques, 100.000 tests [16] ont été effectués. Nous obtenons ainsi 20 millions de tests par expérimentation.

14. Les matrices proviennent de la dernière itération de l'algorithme SGInfiniteVI.
15. Les positions des objets sont choisies de manière aléatoire suivant une distribution uniforme sur l'intervalle $[0, n^2[$, où n^2 est le nombre cases dans la grille.
16. Les positions des agents sont choisies suivant une distribution uniforme sur l'intervalle $[0, n^2[$.

Etat

s_0

Matrice

$agent_2$

$agent_1$	G	B	H	D	N
L	3.8 ;3.9	1.1 ;1.1	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9
D	3.8 ;3.9	3.8 ;3.9	-196 ;-191	12.8 ;13.1	3.8 ;3.9
U	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	-196 ;-191
R	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	12.8 ;13.1	12.8 ;13.1
N	3.8 ;3.9	-198 ;-193	3.8 ;3.9	3.8 ;3.9	12.8 ;13.1

Actions $\langle Bas, Droite \rangle$

Etat

s_1

Matrice

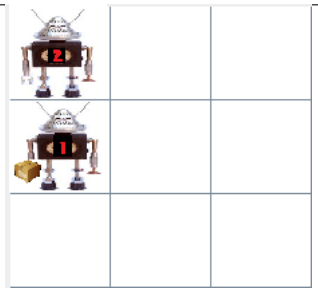
$agent_2$

$agent_1$	G	B	H	D	N
L	3.8 ;3.9	12.8 ;13.1	12.8 ;-181	3.8 ;3.9	12.8 ;13.1
D	-196 ;-191	12.8 ;13.1	3.8 ;-191	3.8 ;3.9	3.8 ;3.9
U	3.8 ;3.9	12.8 ;13.1	3.8 ;-191	3.8 ;3.9	3.8 ;3.9
R	12.8 ;13.1	42.8 ;43.7	-187 ;-181	12.8 ;13.1	12.8 ;13.1
N	3.8 ;3.9	12.8 ;13.1	12.8 ;-181	3.8 ;3.9	12.8 ;13.1

Actions $\langle Droite, Bas \rangle$

Etat

s_2



Matrice

$agent_2$

$agent_1$	G	B	H	D	N
L	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1
D	42.8 ;43.7	42.8 ;43.7	-157 ;-151	42.8 ;43.7	42.8 ;43.7
U	142 ;145	142 ;145	142 ;145	142 ;145	-157 ;-151
R	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1	12.8 ;13.1
N	42.8 ;43.7	-157 ;-151	42.8 ;43.7	42.8 ;43.7	42.8 ;43.7

Actions $\langle Haut, Gauche \rangle$

s_3

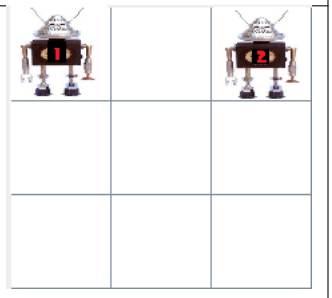


FIGURE 4.4 – Déroulement d'une mission de déplacement d'objets : 2 agents et 2 objets (grille toroïdale).

Comparaison suivant le nombre d'équilibres distincts

Avant de simuler les comportements des agents avec les politiques d'actions calculées, nous allons dans un premier temps compter et comparer le nombre d'équilibres de Nash distincts d'un agent à un autre. En effet, des agents qui basent leurs actions sur des résultats d'équilibre différents, risquent d'entrer en conflit. Cette première évaluation, peut nous fournir une première vue de ce qu'implique la décentralisation sur la solution et l'utilité des fonctions de sélection précédemment proposées.

Le tableau 4.1 présente une comparaison entre les trois fonctions de sélection, suivant le pourcentage d'équilibres de Nash distincts entre les deux agents. Nous constatons que dans la majorité des cas, le pourcentage d'équilibres de Nash différents reste faible et ne dépasse pas le seuil de 5%. A ce niveau, nous déduisons que la décentralisation n'implique qu'une faible discorde sur les choix des équilibres. Nous remarquons également que les fonctions $f^{NashMaxTot}$ et $f^{NashMaxSub}$ se montrent meilleures que la fonction $f^{NashPareto}$.

TABLE 4.1 – Le pourcentage d'équilibres de Nash différents, suivant les trois fonctions de sélection d'équilibre, différentes tailles de la grille et quatre objets transportés.

	$f^{NashMaxTot}$	$f^{NashMaxSub}$	$f^{NashPareto}$
4 x 4	2.36	2.46	4.59
5 x 5	0.64	0.61	1.05
6 x 6	0.61	0.59	1.17
7 x 7	1.27	1.38	2.07
8 x 8	0.86	0.83	2
9 x 9	1.19	1.13	1.96
10 x 10	1.55	1.45	1.93
11 x 11	1.08	1.76	2.99

Critères d'évaluation

Toutefois, le pourcentage d'équilibres différents ne nous renseigne que partiellement sur la qualité de la solution obtenue. En effet, jouer deux équilibres distincts ne signifie pas nécessairement des actions conflictuelles, nous devons donc envisager d'autres critères d'évaluation. Nous définissons trois critères qui nous semblent pertinents :

– **Nombre moyen de conflits par simulation** : pour cette application, aucune hypothèse contraignante n'a été imposée et les conflits d'espace n'ont pas été interdits. Par exemple, si les agents jouent des actions les amenant à occuper la même case, alors dans ce cas la situation est simplement considérée comme conflictuelle.

Ce critère de comparaison peut être considéré comme une mesure du degré de coordination entre les agents. La figure 4.5 montre une situation où les agents peuvent avoir la même position à l'instant $t + 1$ s'ils désirent transporter le même objet. Il est à noter qu'une situation conflictuelle est un état ordinaire, pour lequel les agents ont calculé un équilibre. En conséquence, aucune intervention n'est faite durant la simulation pour la gestion de ce type de situation.

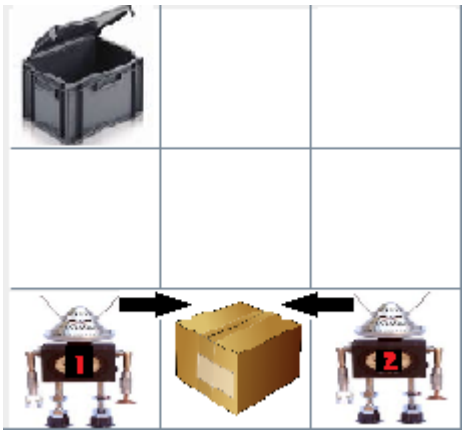


FIGURE 4.5 – Situation conflictuelle induite par les déplacements des agents.

- **Nombre moyen de livelocks par simulation** : un livelock est un cycle sans fin, empêchant les agents d'atteindre un état but. C'est une situation où un agent change continuellement d'état en réponse au changement d'état des autres agents sans pour autant être productif (figure 4.6). Si les agents tombent dans un tel cycle, le jeu ne pourra plus progresser. Un exemple humain, serait le cas de deux personnes se rencontrant dans un couloir étroit et se déplaçant de côté pour s'éviter mais n'arrivant pas à se synchroniser comme il faut et se gênant toujours l'une l'autre. Si le nombre d'étapes d'une mission dépasse un certain seuil, alors un livelock est signalé et la simulation est réinitialisée.
- **Nombre moyen de deadlocks** : un tel état est caractérisé par l'impossibilité de la poursuite de l'exécution de la mission, sans intervention d'un mécanisme extérieur. En d'autres termes, un deadlock est une situation où deux (ou plusieurs) agents sont bloqués mutuellement et indéfiniment. Un deadlock peut être atteint, lorsque l'une des deux conditions suivantes est rencontrée :
  - la présence d'une exclusion mutuelle lors de l'accès à une ressource,
  - les agents ayant simultanément besoin des mêmes ressources et chacun est en attente de la ressource que l'autre détient.

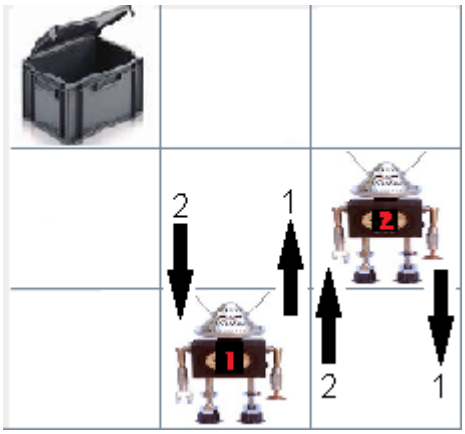


FIGURE 4.6 – Situation de livelock induite par les déplacements des agents.

Le phénomène est détecté automatiquement lorsque les agents ne changent pas de position au bout d'un certain nombre d'étapes. L'interblocage est alors répertorié et la simulation réinitialisée. Un exemple simple (figure 4.7) de deadlock, est le cas où les agents veulent interchanger leur position, alors que cette action jointe est interdite par la fonction de transition.

FIGURE 4.7 – Situation de deadlock induite par les déplacements des agents.

Pour l'ensemble des résultats (figures 4.8 à 4.13), nous remarquons que :

– les fonctions *NashMaxTot* et *NashMaxSub* ont des comportements similaires et sont très proches en valeurs (les valeurs diffèrent de l'ordre de 0.001). Nous avons alors choisi de tracer une seule courbe pour les deux fonctions. Pour l'application traitée, les agents doivent coopérer pour l'accomplissement d'un objectif commun ; i.e. maximiser leur gain global coïncide avec le fait de maximiser leur gain individuel (et *vice versa*).

– il existe une influence de la parité (arithmétique) de la taille de la grille sur les résultats. Si nous considérons par exemple que des valeurs paires, nous obtenons alors des courbes plus lisses.

Nous allons maintenant discuter des résultats obtenus pour les différentes expérimentations. Les scénarios testés comprennent deux agents, quatre objets et différentes dimensions de l'environnement.

Respect de l'équilibre de Nash

La figure 4.8 illustre une comparaison des fonctions de sélection d'équilibre, suivant le nombre de conflits. Le nombre moyen de conflits pour les fonctions maximisant le gain est très proche de zéro. En revanche, la fonction *NashPareto* décroît en fonction de la taille de l'environnement avec une variation de 0.4 (grille 5×5) à 0.02 (grille 12×12), en prenant en compte la remarque sur les dimensions paires/impaires de l'environnement. Si le nombre de conflits augmente, cela signifie que la coordination entre les agents est réduite. Par conséquent, les équilibres recherchés par la fonction *NashPareto* semble favoriser des comportements individualistes (i.e., le choix de l'équilibre est différent, cf. remarques sur le tableau 4.1).

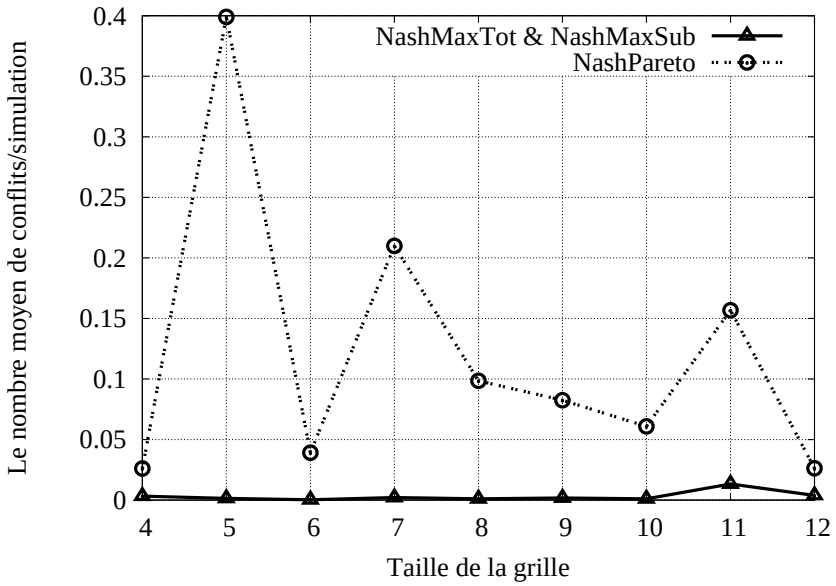


FIGURE 4.8 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de conflits, avec des scénarios de deux agents et quatre objets.

La figure 4.9 traite le nombre de livelocks pour les mêmes fonctions de sélection d'équilibre. Nous constatons également que le nombre moyen de livelocks par simulation est une

fonction quasi nulle pour les fonctions *NashMaxTot* et *NashMaxSub* ; alors que la fonction *NashPareto* fournit des variations plus importantes allant de 0.01 (grille 4×4) à 0.002 (grille 12×12), avec une quasi-stabilité (i.e. variation faible) sur des tailles allant de 7 à 10. Plus globalement, les valeurs obtenues pour le nombre de livelocks, montrent que ce type de situations ne se produit que rarement et ce pour plusieurs millions de simulations.

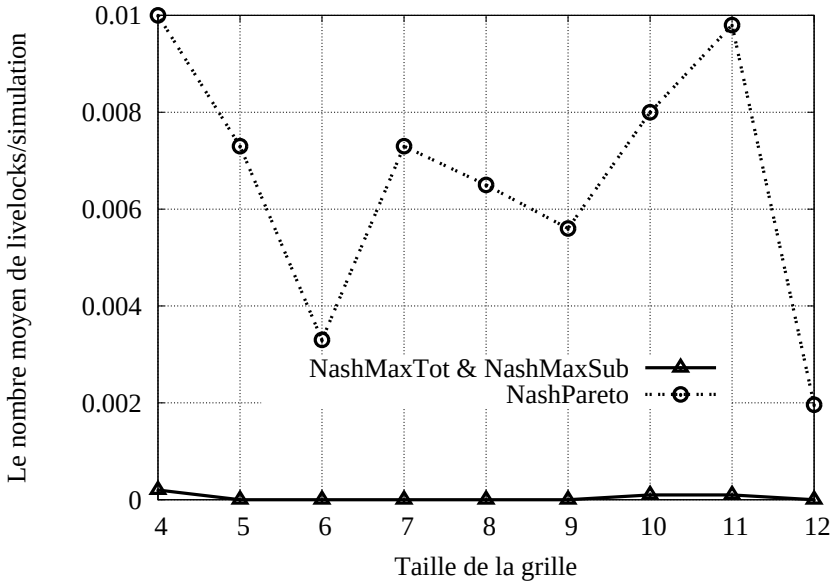


FIGURE 4.9 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de livelocks, avec des scénarios de deux agents et quatre objets.

La figure 4.10 vise à évaluer le nombre de deadlocks. Les résultats montrent cette fois-ci un comportement meilleur pour la fonction *NashPareto* au dépend des deux autres. Globalement, la variation pour la fonction *NashPareto* est entre $0.23 - 0.27$; tandis que pour les deux autres fonctions, elle est de $0.37 - 0.4$. Pour la fonction *NashMaxTot*, la situation de deadlocks est interprétée de la manière suivante : l'agent prédit que l'action peut être conflictuelle avec un autre agent ; et il décide donc de ne pas se déplacer (l'interprétation peut être similaire pour l'autre agent). Ceci explique un nombre important de situations de deadlocks.

En revanche pour la fonction *NashPareto* et dans les mêmes circonstances, chaque agent aura plus tendance à se déplacer et à occuper ainsi la case source d'interblocage. La justification d'un tel comportement est relativement simple, car du point de vue de chaque agent, se déplacer ne représente pas une action conflictuelle (cela permet d'augmenter son propre gain et ne diminue pas celui de l'autre). Ce raisonnement a pour effet le déblocage des situations de deadlocks (au contraire des autres fonctions) mais au risque d'augmenter

le nombre de conflits. D'ailleurs, cette interprétation est confortée par les résultats obtenus en figure 4.8.

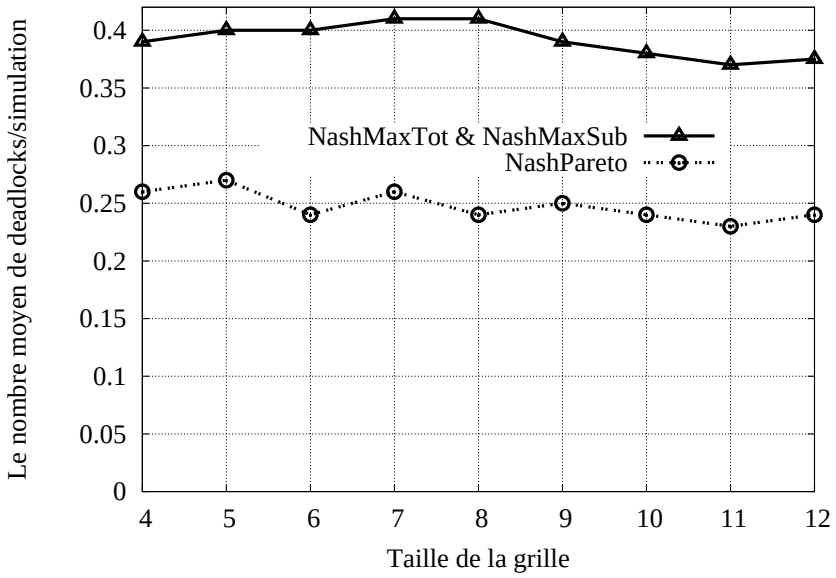


FIGURE 4.10 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de deadlocks, avec des scénarios de deux agents et quatre objets.

Nous pouvons ainsi affirmer qu'une situation d'optimalité au sens de Pareto n'est pas nécessairement une situation socialement "juste". Pour prendre un exemple, nous supposons l'existence d'un état s ayant de multiples équilibres de Nash. Si dans l'un de ces équilibres, un agent i détient toute la récompense liée à l'état s alors c'est un optimum de Pareto, car transférer une partie de ses gains à d'autres agents réduirait directement son bien-être. Par ailleurs, dans cette même situation, s'il devient possible de faire des changements qui augmenteraient le gain des autres agents sans retirer du gain à l'agent i, alors la situation n'est plus Pareto-optimale.

Non respect de l'équilibre de Nash

Le concept de l'équilibre de Nash est basé sur le fait que tous les agents jouent de façon optimale. Cependant, différents types de limitations peuvent affecter le comportement d'un agent. Nous commençons donc par examiner les sources de ces limitations ainsi que leur effet sur la qualité du comportement global des agents.

Nous définissons la limitation comme tout ce qui peut restreindre l'agent de jouer sa politique d'actions (obtenue après une phase de planification). Les limites peuvent être classées en deux catégories : les limites physiques et les limites rationnelles.

1. Les limites physiques sont celles causées par l'interaction de l'agent avec son environnement et leur occurrence est souvent inévitable. Une limitation physique évidente est que l'agent tombe tout simplement en panne. Un robot (agent mobile) peut cesser de se déplacer ou au moins peut perdre l'usage de l'un de ses actionneurs empêchant certains mouvements. L'observabilité partielle, est considérée également comme une limite physique, cela se produit quand un agent ne peut pas distinguer deux états différents du monde et risque de jouer une action inadéquate.

2. Les limites rationnelles sont des décisions explicitement choisies par l'agent, dans la mesure où il décide délibérément de ne pas suivre la politique calculée. En effet, on peut considérer le cas d'un agent doté d'une capacité de réflexion durant la phase d'exécution, lui permettant d'estimer que dans certains cas l'action fournie par l'équilibre de Nash n'est pas la meilleure action. Nous pourrons également envisager un système hétérogène composé d'agents et d'humains. L'humain n'ayant pas la même manière de raisonner qu'un agent, les actions qu'il adoptera peuvent constituer une politique sous-optimale.

Rappelons que les résultats donnés dans la section précédente supposaient l'hypothèse suivante : les agents doivent respecter impérativement leur politique au cours de la phase d'exécution. Nous relaxons cette hypothèse et nous considérons que l'un des agents est affecté par une des limitations que nous avons citées. Nous nous intéressons dans ce cas de figure, à étudier le comportement de l'autre agent, qui continue individuellement à respecter la politique qu'il s'est calculée.

Ce type d'expériences, permet dans une certaine mesure de vérifier la robustesse d'un système multi-agents (basé sur le concept de l'équilibre de Nash) face à différentes perturbations. La question est de savoir, si l'objectif global des agents sera atteint lorsque l'un des agents ne coopère plus. Notre étude s'est portée sur un cas particulier de non-coopération et considère que l'un des deux agents adopte un comportement aléatoire [17], aboutissant ainsi à des scénarios imprévisibles. A chaque étape du jeu, l'agent se déplace soit sur l'une des quatre cases voisines choisie au hasard, soit il reste sur place. Ces expériences ont pour but de vérifier la capacité des autres agents à s'adapter à de telles situations, rapprochant de la sorte la simulation aux contraintes réelles de la robotique.

Si nous considérons le nombre d'étapes nécessaires pour l'accomplissement d'une mission, les expériences (tableau 4.2) montrent une forte dégradation de la qualité de la solution. Par exemple, pour une grille de 7×7, le nombre d'étapes passent de 13 ± 1.33 à

17. Comportement aléatoire étudié : une même distribution de probabilité sur l'ensemble des actions à jouer.

93.57 ± 79.95. Cependant, tous les objets sont ramassés à chaque fois et ce malgré le fait qu'un agent donné ne joue plus son rôle dans la réalisation de la tâche collective.

TABLE 4.2 – Evaluation du non respect de l'équilibre de Nash en fonction du nombre d'étapes.

Taille grille	Nombre d'états	Respect de Nash	Nombre de pas	Ecart type
4×4	16384	$\surd$	8.17	1.33
		$\times$	26.44	17.55
5×5	40000	$\surd$	9.92	1.47
		$\times$	38.97	29.35
6×6	82944	$\surd$	11.72	2.11
		$\times$	54.78	48.57
7×7	153664	$\surd$	13.09	2.27
		$\times$	93.57	79.95
8×8	262144	$\surd$	15.51	2.80
		$\times$	138.24	114.19
9×9	419904	$\surd$	17.71	2.97
		$\times$	184.23	151.85
10×10	640000	$\surd$	18.84	3.39
		$\times$	227.91	196.47

Les expériences (figure 4.11) montrent globalement une diminution du nombre moyen de conflits parallèlement à la taille de la grille. Nous pouvons justifier ce résultat par le nombre d'agents et le nombre d'objets (relativement faible par rapport à l'espace disponible dans la grille). Nous pouvons également voir que la variation est moins accentuée (fonctions décroissantes) pour des tailles d'environnement plus élevées ($n > 8$). Les valeurs maximales correspondent à des environnements de taille 5 et 7 : 1.5 pour les fonctions *NashMaxTot* et *NashMaxSub* et 2.2 pour la fonction *NashPareto*. Notons que par rapport au scénario de respect de l'équilibre de Nash (figure 4.8), nous retrouvons le même agencement des courbes, mais avec un nombre de conflits plus important. L'explication nous semble évidente : le comportement aléatoire de l'un des deux agents amplifie le phénomène.

Les résultats proposés en figure 4.12, concernent le nombre moyen de livelocks et présentent des courbes croissantes en fonction de la taille de la grille. Nous nous apercevons que la valeur maximale est de 0.075 (resp. 0.035) pour la fonction *NashPareto* (resp. pour les deux autres fonctions). Ces valeurs maximales sont obtenues pour une grille de taille 12×12. En revanche, le nombre de livelocks est quasi-nul pour des grilles de dimension

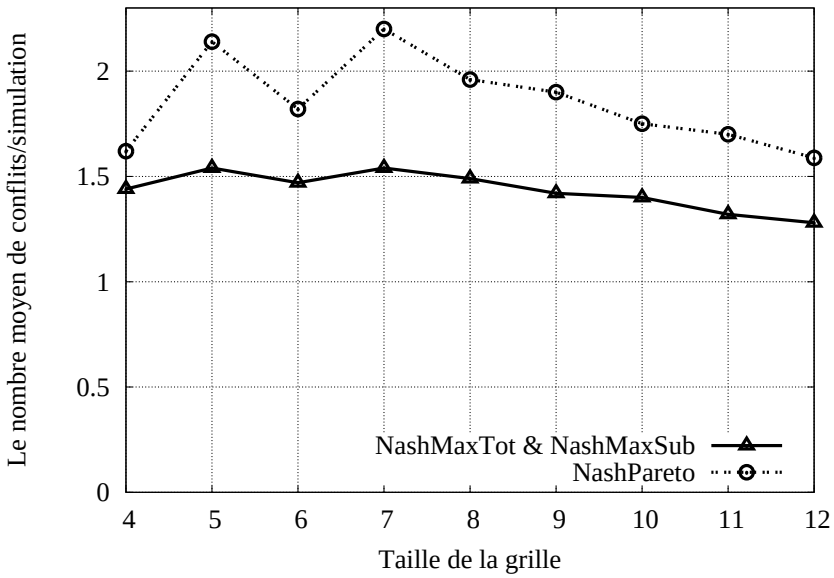


FIGURE 4.11 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de conflits, avec des scénarios de deux agents et quatre objets.

inférieure à 8×8. Globalement, nous constatons que le non respect de la décision planifiée, produit un nombre de livelocks plus important que dans des situations coopératives. A première vue, ce phénomène peut paraître anormal, car il y a peu de cycles infinis avec un comportement aléatoire. Mais les résultats s'expliquent par la définition du livelock, telle qu'elle a été implémentée : l'application considère la situation comme un livelock dès que la tâche commune n'est pas accomplie après un certain nombre d'étapes (comportement "non-productif"). Nous pouvons ainsi justifier les résultats obtenus.

La figure 4.13 vise à évaluer le nombre de deadlocks. Nous remarquons que les résultats semblent être très sensibles à la parité arithmétique de la dimension de la grille. La variation pour la fonction *NashPareto* est de $0.55 - 0.60$; tandis que pour les deux autres fonctions, elle est de $0.4 - 0.52$. Les résultats montrent cette fois-ci un comportement meilleur pour les fonctions *NashMaxTot* et *NashMaxSub* au dépend de la fonction *NashPareto*.

De manière générale, si nous admettons que les limitations (que peut subir un agent) sont plus faibles que celles provoquées par un comportement aléatoire, nous pouvons alors suggérer qu'une solution basée sur l'équilibre de Nash est résistante à des situations d'interaction réelles. En effet, les résultats obtenus montrent une dégradation de la qualité de la solution. Toutefois, cette dégradation est d'une part attendue en raison de perturbations introduites et d'autre part peut être qualifiée "d'acceptable", compte

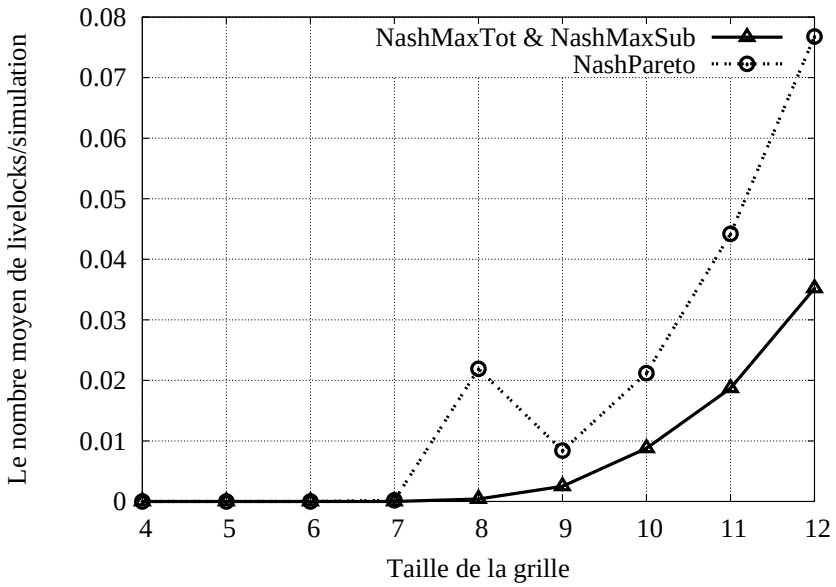


FIGURE 4.12 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de livelocks, avec des scénarios de deux agents et quatre objets.

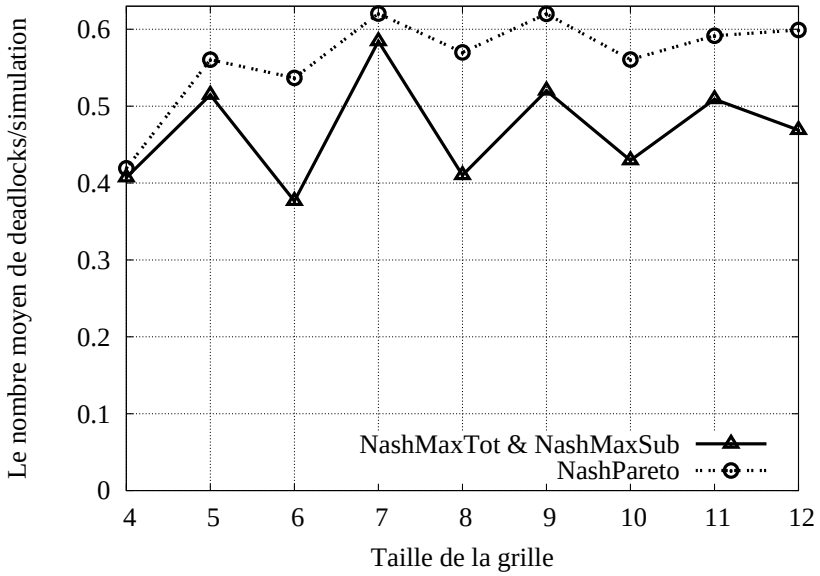


FIGURE 4.13 – Comparaison entre les différentes fonctions de sélection d'équilibre, suivant le nombre moyen de deadlocks, avec des scénarios de deux agents et quatre objets.

tenu des faibles valeurs constatées (1.5 conflit/simulation ; 0.01 livelock/simulation ; 0.4 deadlock/simulation).

4.2.3 Expériences complémentaires

Des expériences complémentaires ont été réalisées dans le but d'enrichir l'évaluation de l'algorithme SGInfiniteVI. Nous avons effectué des simulations avec un nombre d'agents supérieur à deux, dans le but d'explorer les limites de l'algorithme. Nous avons également, comparé les résultats obtenus (précédemment) avec ceux d'une approche centralisée.

Augmentation du nombre d'agents

Nous avons pu effectuer quelques expériences avec des scénarios allant jusqu'à cinq agents, mais au-delà de ce nombre, la résolution devient difficile à cause des débordements de mémoire ou des temps de calcul excessifs ; par exemple pour un scénario de 4 agents, 4 objets et une grille de 4×4, le temps nécessaire au calcul d'une seule politique dépasse les 16 heures (tableau 4.3). Nous avons également constaté que même avec une grande batterie de tests, les résultats n'ont montré aucun conflit ou livelock. Nous pensons que ce phénomène est dû au fait que le nombre d'étapes n'était pas suffisant pour avoir une interaction riche. Pour ces raisons, nous avons introduit seulement les résultats correspondant au scénario de deux agents.

TABLE 4.3 – Evolution du nombre d'états et du temps de calcul de la solution en fonction du nombre d'agents (le nombre d'objets est fixé à 4).

	Espace d'états	Temps de calcul (en minutes)
2 agents	16384	<1
3 agents	524.288	58
4 agents	16.777.216	963
5 agents	536.870.912	-

Notons également que la transition vers un plus grand nombre d'agents a été possible, mais seulement en utilisant des ε-équilibres de Nash. De ce fait, se pose alors le problème de comment définir ε ? A notre connaissance, il n'existe pas de travaux traitant ce problème avec le modèle des jeux stochastiques. Pour contourner cet obstacle, nous avons décidé de déterminer ε de manière empirique. Notre méthode, consiste d'abord à fixer arbitrairement ce paramètre et ensuite à déterminer le pourcentage d'états qui possèdent au moins un équilibre. Si le pourcentage constaté est faible, nous augmentons alors la valeur de ε et nous effectuons de nouveau les tests.

Comparaison avec une approche centralisée

Si nous rappelons l'objectif principal de ce travail, nous pourrons dire qu'il s'agissait de développer un algorithme de planification multi-agents pour le modèle des jeux stochastiques. Face au problème de la multiplicité des équilibres, nous avons proposé trois fonctions distinctes pour la sélection d'équilibre de Nash. Les expériences sur cette application, ont montré que les fonctions maximisant le gain global/individuel offraient les meilleurs comportements.

TABLE 4.4 – Mémoire utilisée (Mo) et temps de calcul (sec) pour différents nombres d'objets, différentes tailles de la grille et 3 agents.

	Grille	Mémoire (Mo)		Temps (sec)	
		MMDP	SGInfiniteVI	MMDP	SGInfiniteVI
2 Objets	4×4	5.2	13.5	33	74
	5×5	12	47.6	164	238
	6×6	33.2	140	443	678
	7×7	82.1	351.8	1187	1826
	8×8	181.5	782.5	2397	3761
	9×9	366.9	1585.4	-	-
3 Objets	4×4	7.22	25.38	90	166
	5×5	22.7	94.1	380	507
	6×6	65.4	279.2	960	1491
	7×7	163	702.5	2521	4233
	8×8	362	1564	-	-
4 Objets	4×4	12.4	49.8	255	337
	5×5	44.1	187.3	802	1188
	6×6	129.5	557.3	2034	3531

Afin d'éviter toute évaluation subjective, nous avons complété l'évaluation par d'autres expériences [Hamila *et al.*, 2010b]. Nous comparons la solution obtenue (contrôle distribué) avec celle issue d'un contrôle centralisé (un MMDP), considéré ici en tant que borne optimale pour le problème relaxé (environnement complètement observable). Nous utilisons cette borne dans le but d'évaluer la quantité de ressources (calcul/mémoire) nécessaire à l'algorithme de résolution durant la phase de planification.

Le tableau 4.4, présente une comparaison entre le modèle MMDP et le modèle des jeux stochastiques, suivant les besoins en espace mémoire et en temps de calcul [18]. L'agrandissement de la taille du problème, est assuré par la variation du nombre d'objets transportés

18. C'est la durée moyenne de calcul d'une politique d'actions.

et de la taille de la grille. Le nombre d'agents a été fixé à 3. Par exemple, pour une grille de 4×4 et 4 objets transportés, l'algorithme correspondant au modèle MMDP consomme 12.4 Mo de mémoire et nécessite 255 secondes pour calculer une politique, tandis que l'algorithme SGInfiniteVI a besoin de 4 fois plus de mémoire et de 30% en plus en temps de calcul.

Les deux algorithmes ont été implémentés de la même manière, c'est-à-dire sur la base d'un algorithme d'itération sur les valeurs. Ainsi, nous pointons la construction des matrices, la recherche et la sélection d'équilibre comme les causes de cet excès de consommation des ressources mémoire et temps. Cette conclusion nous amène inévitablement à repenser la manière dont les équilibres sont calculés et choisis (cf. section 4.3).

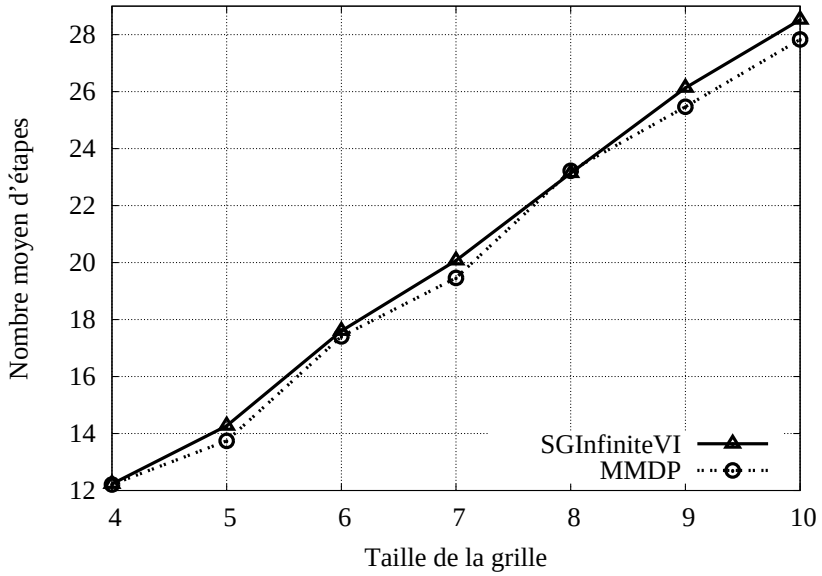


FIGURE 4.14 – Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen d'étapes et avec des scénarios de deux agents et quatre objets.

La figure 4.14 illustre une comparaison entre l'approche centralisée et notre approche, en fonction du nombre d'étapes nécessaires pour atteindre l'objectif. Les résultats montrent que les deux courbes sont assez proches l'une de l'autre. La différence est quasi-nulle avec un environnement réduit, mais s'affirme progressivement avec des tailles plus grandes. Nous en déduisons, que la décentralisation implique une légère dégradation de la qualité de la solution.

La figure 4.15 compare les deux approches suivant le nombre d'étapes lorsque le deuxième agent adopte un comportement aléatoire. Nous constatons alors un nombre

d'étapes élevé pour les deux approches, reflétant la perturbation qui a affecté le système. Cependant, l'approche décentralisée avec l'algorithme SGInfiniteVI fournit un meilleur comportement global en nécessitant moins de pas pour atteindre l'objectif commun. Par exemple, il faut environ 106 étapes aux agents utilisant l'approche centralisée pour transporter tous les objets, tandis qu'avec l'approche décentralisée les agents nécessitent seulement 78 étapes.

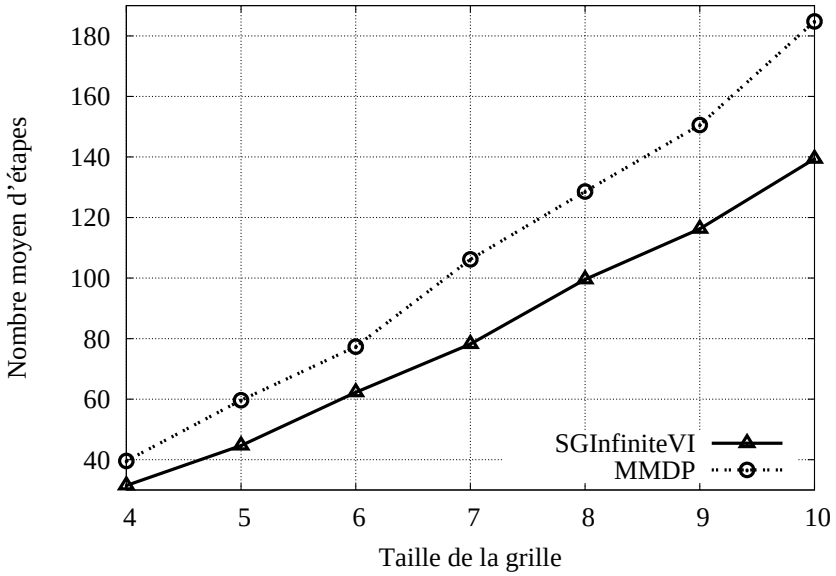


FIGURE 4.15 – Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen d'étapes et avec des scénarios de deux agents et quatre objets.

Toujours avec la même condition de comportement aléatoire pour le deuxième agent, nous comparons cette fois-ci les deux approches suivant le nombre de conflits. La courbe correspondant à l'approche centralisée est une courbe décroissante et varie entre les valeurs 1.9 et 1.25. La courbe de l'algorithme SGInfiniteVI, est une courbe également décroissante et varie entre 1.6 et 1.2. Le nombre de conflits est moins important pour l'approche décentralisée. Globalement, nous pouvons affirmer que l'approche décentralisée présente une dégradation de la qualité de la solution lorsque les agents respectent les politiques calculées. A l'inverse, en cas de perturbation affectant le comportement d'un agent donné, l'approche centralisée présente un comportement moins flexible. En effet, le fait d'avoir une seule fonction de récompense pour tous les agents, accroît la dépendance inter-agents et justifie le manque d'adaptation en cas de non coopération.

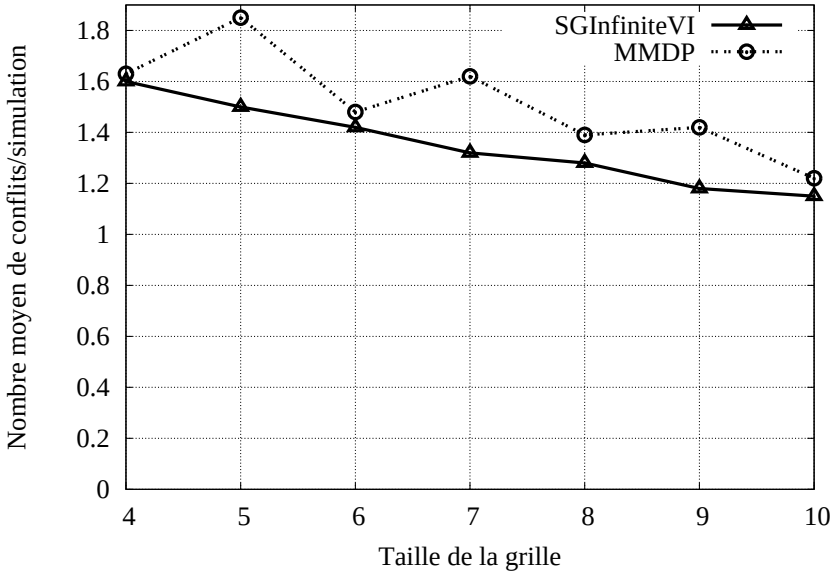


FIGURE 4.16 – Comparaison entre l'approche centralisée (MMDP) et l'approche décentralisée (SGInfiniteVI), suivant le nombre moyen de conflits et avec des scénarios de deux agents et quatre objets.

4.3 Evaluation de l'algorithme avec élimination des stratégies dominées

Dans cette section, nous nous intéressons à analyser la deuxième version de l'algorithme, qui consiste à améliorer le processus de résolution. Dans ce contexte, l'objectif visé est de réduire la complexité de calcul et de limiter la multiplicité des équilibres de Nash à chaque étape du jeu. Nous avons structuré notre évaluation en deux parties :

- Une évaluation quantitative qui analyse l'effet de l'élimination des stratégies strictement et faiblement dominées sur les ressources utilisées.
- Une évaluation qualitative qui analyse l'impact de l'élimination des stratégies dominées sur la qualité de la solution. En effet, le concept de dominance stratégique pourrait éliminer des stratégies qui sont inutiles à court terme mais qui peuvent être intéressantes à long terme. Nous allons donc mener des expériences pour étudier ce phénomène. Les expériences portent sur les mêmes critères d'évaluation (décrits plus haut) et permettent ainsi de comparer les résultats obtenus avec ceux de la version initiale de l'algorithme. Si la procédure ne dégrade pas le comportement des agents, cela permettra de considérer des instances de problèmes plus grands (par exemple en

augmentant le nombre d'agents ou la taille de l'environnement, etc.) et d'envisager des applications plus réalistes.

Le même nombre de simulations a été retenu, on calcule 200 politiques et ensuite on effectue 100.000 tests sur chacune de ces politiques. Nous obtenons ainsi 20 millions de tests par expérimentation.

4.3.1 Evaluation quantitative

Comme déjà évoqué, les stratégies peuvent être de deux types : strictement ou faiblement dominées. Ainsi, nous testons l'algorithme avec deux procédures différentes, effectuant respectivement : l'élimination itérative des stratégies strictement dominées et faiblement dominées. La question principale est de savoir laquelle des deux procédures apporterait un bénéfice sans pour autant dégrader la qualité de la solution. Le cas optimal de réduction correspond à une stratégie par joueur (dont la jointure forme un équilibre de Nash). On nommera le pourcentage de réduction τ.

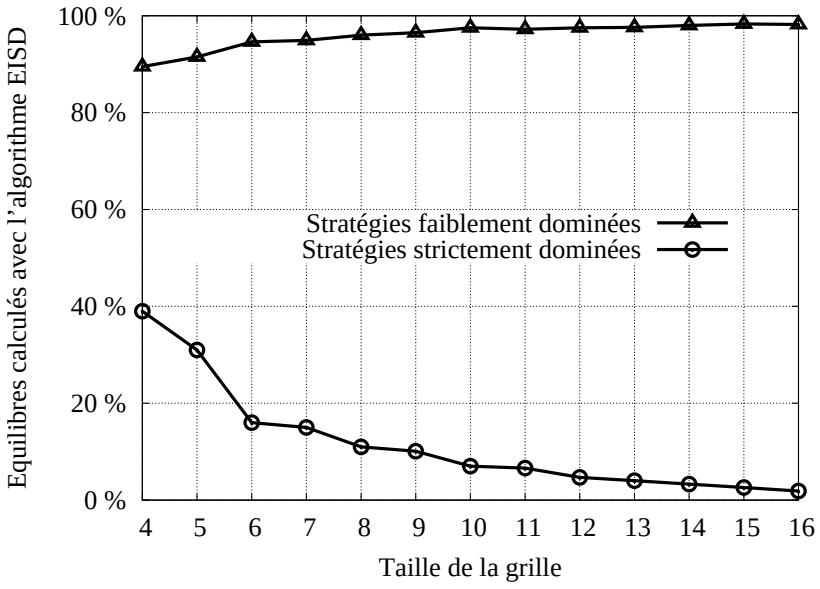


FIGURE 4.17 – Comparaison entre l'élimination des stratégies strictement et faiblement dominées (avec des scénarios de deux agents et quatre objets).

Nous avons vu (cf. chapitre 3.2.1), que l'élimination des stratégies dominées simplifie le jeu mais ne le résout pas toujours. La figure 4.17 illustre une comparaison entre la méthode éliminant les stratégies strictement dominées et la méthode éliminant les stratégies

faiblement dominées. La comparaison porte sur le pourcentage d'équilibres de Nash calculés directement avec l'algorithme EISD, c'est-à-dire sans avoir recours à l'énumération des meilleures réponses de chaque agent.

Les résultats montrent que la courbe correspondant à l'élimination des stratégies faiblement dominées est une courbe croissante, convergeant vers le cas optimal (tous les équilibres de Nash sont trouvés par élimination). En revanche, la courbe des stratégies strictement dominées est une courbe qui décroît en fonction de la taille du problème et tend vers 0 (grille 16×16). Les résultats obtenus sont certes dépendants de l'application choisie, mais ils reflètent d'une manière générale la prépondérance des stratégies faiblement dominées (en matière de compactage des matrices). Bien que cette conclusion soit d'avance prévisible, la divergence des courbes semble être un phénomène surprenant, car l'écart entre les deux se creuse au fur et à mesure que l'instance du problème s'agrandit. Nous pourrons justifier ces courbes par le fait que, le nombre d'états résolvables par élimination des stratégies strictement dominées, est une constante (ou varie peu) en fonction de la taille du problème. Ainsi, le pourcentage correspondant diminue à chaque fois que l'espace d'états augmente.

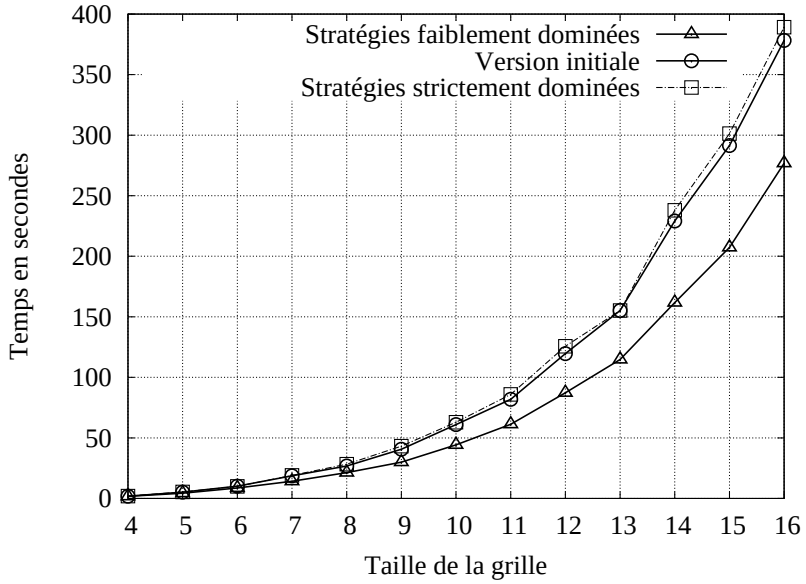


FIGURE 4.18 – Comparaison entre l'élimination des stratégies strictement/faiblement dominées et la version initiale de l'algorithme suivant le temps de calcul (avec des scénarios de deux agents et quatre objets).

Compte tenu des résultats obtenus ci-dessus, nous avons entrepris de comparer le temps de calcul utilisé par l'algorithme SGInfiniteVI avec et sans élimination des straté-

gies dominées (figure 4.18). Premièrement, nous constatons qu'il n'y a quasiment pas de différence entre les trois versions de l'algorithme ((1) version de base, (2) avec élimination des stratégies strictement dominées et (3) avec élimination des stratégies faiblement dominées) et ce pour des problèmes de petites tailles. Deuxièmement, nous remarquons que par rapport à la version initiale de l'algorithme, la méthode avec élimination des stratégies strictement dominées, présente une courbe qui est légèrement au dessus. Toujours par rapport à la version initiale, la méthode traitant avec les stratégies faiblement dominées présente une courbe légèrement au dessous, mais avec un écart qui devient plus important en fonction de la taille de la grille. Ces résultats, prouvent que si l'instance du problème est assez grande, alors l'élimination des stratégies faiblement dominées peut apporter un gain en temps de résolution (de l'ordre de 25%) ; nous appellerons ce gain G_{tr}.

En dehors du cadre de l'application considérée, nous pouvons affirmer que la réduction de la taille de la matrice n'est pas forcément synonyme de diminution du temps de calcul. En effet, le processus d'élimination des stratégies dominées apporte un gain en temps, appelé g_{te}. Ce gain porte sur :

- le temps d'exploration des matrices,
- le temps de recherche et de sélection d'équilibre.

Ce processus entraîne également un coût de calcul c_{tc}, correspondant au temps d'exécution de l'algorithme EISD. Pour qu'un gain G_{tr} soit perceptible au niveau général, il faut que le degré de réduction τ (de la matrice) soit assez grand pour générer un gain g_{te} couvrant c_{tc}. De ce fait, nous justifions l'augmentation du temps de résolution de l'algorithme SGInfiniteVI lorsqu'il est combiné seulement avec l'élimination des stratégies strictement dominées. Nous envisageons, d'intégrer le paramètre τ à l'algorithme SGInfiniteVI et permettre ainsi automatiquement à celui-ci de lancer si nécessaire (c'est-à-dire à partir d'un certain seuil) la procédure d'élimination des stratégies dominées.

Technique de réduction de l'espace mémoire

A ce niveau, la procédure d'élimination des stratégies dominées permet d'optimiser uniquement le temps de calcul. Afin d'avoir un gain en espace, nous présentons une technique réduisant la quantité de mémoire nécessaire à la construction d'un jeu matriciel $M(s)$, en le réduisant à un sous jeu plus petit $M''(s)$. Nous allons voir que cette technique est avantageuse, car elle permet d'obtenir les mêmes ensembles d'équilibres de Nash (que ceux de la technique classique d'élimination des stratégies dominées), tout en construisant partiellement les matrices.

On a vu précédemment, qu'avant de calculer les équilibres de Nash d'un jeu, il est préférable de simplifier ce jeu en supprimant les stratégies dominées. Dans un premier

temps, l'algorithme consistait à construire entièrement la matrice de paiement et ensuite faire appel à la procédure EISD, afin d'élaguer les stratégies non souhaitées. En réalité, durant la procédure d'élimination, un agent n'a besoin pour raisonner que de connaître ses propres valeurs pour supprimer telle ou telle stratégie. Par conséquent, il peut ainsi ignorer temporairement les gains des autres agents.

Pour mieux comprendre, considérons la matrice de la figure 4.19. Cette matrice a été entièrement calculée et après élimination des stratégies dominées, on obtient deux équilibres de Nash.

Considérons maintenant, la même matrice mais calculée partiellement (figure 4.20), où le premier agent ne calcule que ses propres valeurs. S'il fait appel à la fonction EISD, alors les trois premières stratégies et ainsi que la cinquième stratégie seront éliminées. En conséquence, les seuls paiements calculés pour le deuxième agent seront ceux qui correspondent à la ligne 4 du premier agent.

3.8 ;3.9	1.1 ;1.1	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9
3.8 ;3.9	3.8 ;3.9	-19 ;-19	12 ;13	3.8 ;3.9
3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	-19 ;-19
3.8 ;3.9	3.8 ;3.9	3.8 ;3.9	12 ;13	12 ;13
3.8 ;3.9	-19 ;-19	3.8 ;3.9	3.8 ;3.9	12 ;13

			12 ;13	12 ;13

FIGURE 4.19 – Construction et recherche d'équilibre avec l'algorithme EISD.

3.8 ;	1.1 ;	3.8 ;	3.8 ;	3.8 ;
3.8 ;	3.8 ;	-19 ;	12 ;	3.8 ;
3.8 ;	3.8 ;	3.8 ;	3.8 ;	-19 ;
3.8 ;	3.8 ;	3.8 ;	12 ;	12 ;
3.8 ;	-19 ;	3.8 ;	3.8 ;	12 ;

3.8 ; 3.9	3.8 ; 3.9	3.8 ; 3.9	12 ; 13	12 ; 13

FIGURE 4.20 – Construction partielle de la matrice et recherche d'équilibre.

Ainsi, l'algorithme SGInfiniteVI a été modifié de manière à prendre en compte la nouvelle technique de construction des matrices. Les différentes étapes nécessaires au calcul de l'équilibre de Nash seront les suivantes :

1. Chaque agent calcule uniquement ses paiements. Le nombre de valeurs calculées est donc : $|A|$.
2. Chaque agent applique ensuite le processus d'élimination des stratégies dominées à sa matrice de paiement. ρ caractérise le nombre d'états où l'équilibre de Nash est déterminé uniquement par ce processus d'élimination.

3. Ce dernier calcule enfin les paiements des autres joueurs correspondant aux stratégies restant après la procédure d'élimination. Le nombre de nouvelles valeurs remplies dans la matrice est donc : $|A_{-i}|$ et cela fait un total de $|A| + |A_{-i}|$ valeurs par matrice.
4. Poursuite de l'algorithme.

Le gain par matrice est estimé alors à :

$$\begin{aligned} g_m &= (nbrValMat - nbrValCalc) \ / \ nbrValMat \\ &= (\ (|A| * |Ag|) - (|A| + |A_{-i}|) \ ) \ / \ (|A| * |Ag|) \end{aligned}$$

Le gain total est :

$$G_m = g_m * \tau$$

TABLE 4.5 – Gain en mémoire suivant le nombre d'agents.

	$\|A_g\|$	$\|A\|$	$\|M\|$	g_m	τ	G_m
2 agents	2	25	50	40%	$\simeq 90\%$	35%
3 agents	3	125	375	60%	$\simeq 75\%$	45%
4 agents	4	625	2500	70%	$\simeq 53\%$	37%

Le tableau 4.5 et la figure 4.21 montrent l'évolution du gain en mémoire en fonction (respectivement) du nombre d'agents et de la taille du problème. Si nous considérons des instances de grands problèmes, alors nous pourrons constater un gain réel en espace mémoire. Par exemple, pour une grille de 7×7 le gain avoisine les 40%. Ce gain devrait permettre de considérer des tailles de problèmes plus grands, d'augmenter le nombre d'agents, etc. A noter qu'un léger gain en temps de calcul a été constaté, vu qu'il y a moins de valeurs (paiements) à calculer.

A la fin de cette évaluation numérique, nous pouvons affirmer que les résultats obtenus confirment l'effet escompté de l'usage des stratégies dominées. En effet, cette technique a permis d'améliorer considérablement l'algorithme SGInfiniteVI, en allouant un gain considérable en temps de résolution ainsi qu'en espace mémoire. Maintenant, il reste à savoir l'effet que peut avoir cette technique sur le comportement des agents, car en éliminant certaines stratégies on élimine aussi certains équilibres de Nash.

4.3.2 Evaluation qualitative

Nous avons vu que l'élimination des stratégies dominées simplifiait considérablement les jeux, au point d'avoir un gain en ressources de calcul. Un autre effet, que nous essayons d'explorer, est celui qui affecte la qualité des politiques calculées. En effet, les résultats

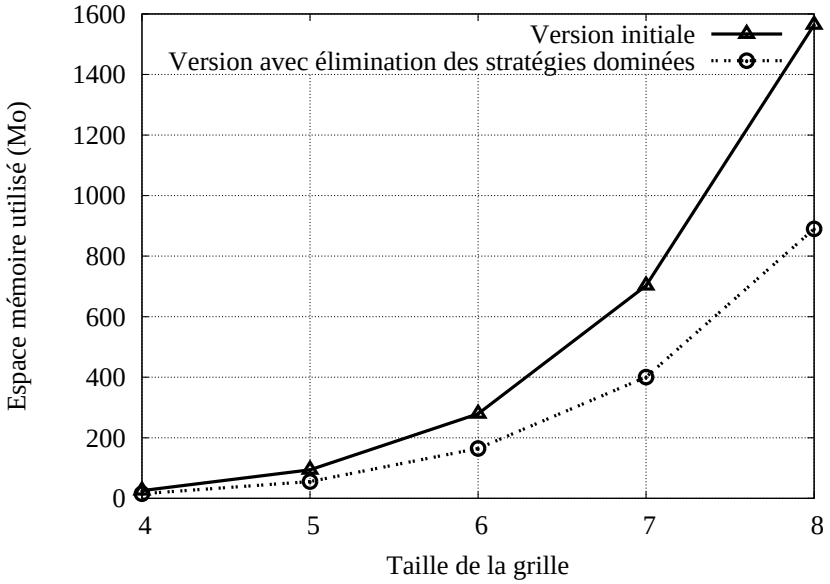


FIGURE 4.21 – L'espace mémoire nécessaire pour calculer la politique d'actions (scénario avec 3 agents et 3 objets).

précédents (figure 4.17) ont montré que dans la majorité des cas, le processus trouvait un équilibre de Nash sans faire appel à la procédure classique de recherche d'équilibre. Cela peut se voir comme un avantage, car on facilite le processus de sélection d'équilibre. Néanmoins, dans ce cas l'élimination de certains équilibres et la sélection d'un unique équilibre ne tiennent compte d'aucun choix social, la technique dépend seulement du poids que possède chaque stratégie par rapport aux autres. En plus, certaines stratégies peuvent ne pas être utiles à court terme et s'avèrent être intéressantes à long terme.

C'est pour ces raisons, que nous nous attachons à évaluer l'effet de l'élimination des stratégies dominées [19] sur le comportement des agents. Afin d'assurer une évaluation objective de la qualité de la solution, nous considérons toujours les mêmes critères d'évaluation : le nombre moyen de conflits, de livelocks et de deadlocks.

La figure 4.22 illustre une comparaison entre les deux versions de l'algorithme suivant le nombre de conflits. Nous nous apercevons que les valeurs correspondant à la version améliorée de l'algorithme, augmentent parallèlement à la taille de la grille et varient entre 0.15 et 0.3. Ainsi, nous concluons que l'élimination des stratégies dominées provoque une augmentation du nombre de conflits par rapport à la version initiale de l'algorithme. Néanmoins, les valeurs observées demeurent relativement minimes et peuvent ainsi être

19. Désormais nous considérons seulement les stratégies faiblement dominées, vue que l'élimination des stratégies strictement dominées ne peut pas apporter des gains significatifs.

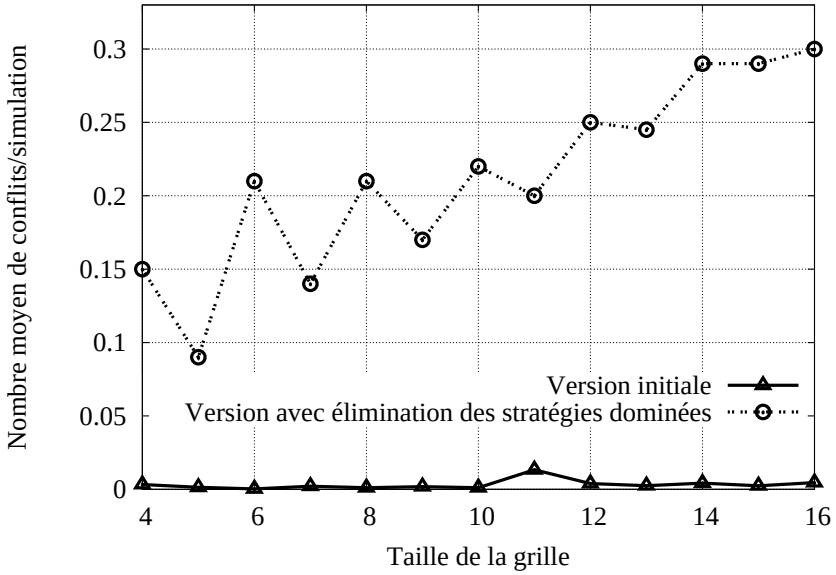


FIGURE 4.22 – Evaluation du nombre moyen de conflits avec et sans élimination des stratégies dominées.

considérées comme acceptables dans le cadre des simulations. Par exemple, pour un scénario de mission avec une taille de grille de 12×12, deux agents et quatre objets, la moyenne de conflits est de 0.25 seulement par simulation.

La figure 4.23 traite le nombre de deadlocks pour les deux versions de l'algorithme. Nous constatons une augmentation du nombre moyen de deadlocks par simulation, par rapport à la version initiale de l'algorithme. Le nombre de deadlocks en hausse, pourra témoigner d'une perte en termes de coordination entre les agents. En effet, ce genre de situation se produit lorsque les actions exécutées proviennent d'équilibres de Nash différents. Toutefois, étant donné que la version initiale de SGInfiniteVI présente un nombre moyen de conflits supérieur à zéro, l'augmentation semble être relativement faible et par conséquent tolérable sur le comportement observé des agents. Quant au nombre moyen de livelocks, il tend vers zéro dans les deux cas.

En résumé sur les différents critères de comparaison, les résultats obtenus illustrent en premier lieu un gain important en temps de calcul ainsi qu'une possibilité de gain en espace mémoire. Intuitivement, ce gain est non sans conséquence sur les politiques d'actions des agents. Cependant, il nous semble légitime de considérer ce gain en dépit d'une légère dégradation de la qualité de la solution.

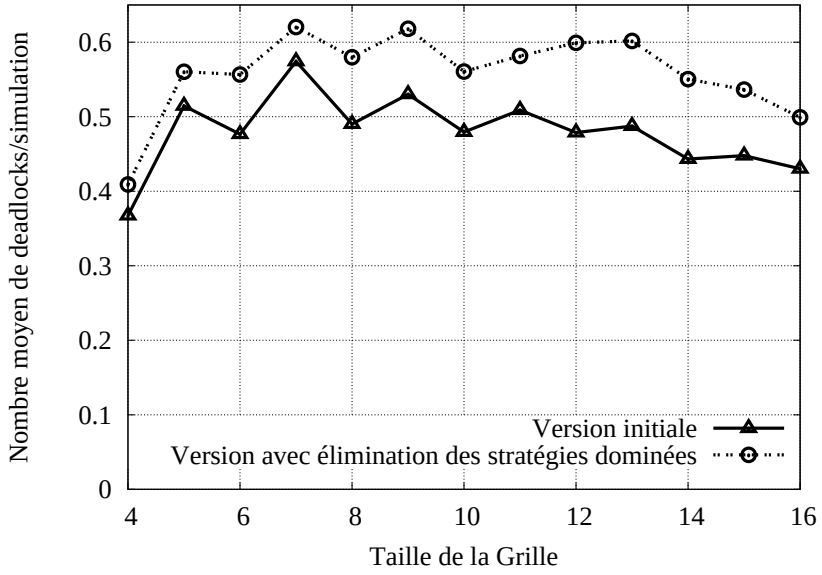


FIGURE 4.23 – Evaluation du nombre moyen de deadlocks, avec ou sans élimination des stratégies dominées.

4.4 Conclusion

Ce quatrième chapitre a porté sur la validation de notre approche pour la planification multi-agents. Nous avons ainsi cherché à démontrer que les solutions proposées dans cette thèse permettent de répondre à la problématique initialement formulée. En effet, l'étude des approches existantes réalisée au chapitre 2, a tout d'abord mis en évidence que le problème de planification des actions avec les jeux stochastiques est considéré comme difficile en terme de résolution. Les travaux dans ce domaine sont peu nombreux, on y trouve les principaux fondements théoriques, mais sans s'adresser aux différents problèmes d'implémentation des algorithmes et de leur applicabilité. Parmi ces problèmes, on peut citer l'utilisation de la fonction de sélection d'équilibre et la centralisation de la prise de décision.

Dans un premier temps, nous avons commencé par tester l'efficacité de l'algorithme que nous avons proposé et nous nous sommes intéressés à la qualité de la solution retournée. Globalement, les expériences qui ont été réalisées ont montré l'applicabilité de nos solutions pour la réalisation de tâches communes par des agents. Plus précisément, il a été montré que seules les fonctions maximisant le gain, permettent d'obtenir les meilleurs comportements. L'approche se basant sur la notion de Pareto-optimalité, semble augmenter les situations de non coordination entre les agents. En comparant notre approche avec une approche centralisée, nous avons montré une légère dégradation de la solution, mais

au profit d'une meilleure autonomie de décision. Toutefois, nous avons pu constater que les techniques utilisées pour la recherche d'équilibre ont conduit à une augmentation des besoins en calcul de l'algorithme.

Dans un deuxième temps, nous nous sommes penchés sur le concept de la dominance stratégique, visant à réduire les ressources utilisées par l'algorithme durant la phase de planification. Bien que ce concept ait été largement étudié en théorie des jeux, il demeure inconnu dans le domaine des jeux stochastiques. Nous avons ainsi synthétisé le processus correspondant et l'avons intégré dans l'algorithme SGInfiniteVI. Les résultats sont prometteurs et montrent un réel intérêt pour cette méthode.

Conclusion et perspectives

L'objectif de cette thèse était de trouver par des moyens algorithmiques la réponse à certaines questions, habituellement soulevées, lorsque l'on traite des problèmes de décision modélisés par des jeux stochastiques. Etant donné un jeu multi-agents d'une certaine forme et en supposant quelques critères de comparaison, nous nous sommes intéressés aux trois questions suivantes :

1. Comment un agent doit-il planifier ses actions afin de mieux s'organiser avec les autres agents pour l'accomplissement d'une tâche donnée ?

2. Comment les agents peuvent-ils calculer des plans d'actions conjoints d'une manière distribuée ?

3. Comment un agent se comporte-t-il dans le cas où les autres ne coopèrent plus ?

Ainsi, ce chapitre présente un bilan de cette thèse, en revenant sur les motivations des choix effectués ainsi que sur l'algorithme que nous avons proposé. Nous résumons la contribution que nous avons apportée avec cette thèse dans l'intention de répondre aux trois questions ci-dessus. Finalement, nous décrivons certaines perspectives pour des travaux futurs.

Contribution

Un premier chapitre bibliographique nous a permis de définir les notions d'agent et de systèmes multi-agents ainsi que le problème de la prise de décision sous incertitude. Dans ce contexte, nous avons introduit deux formalismes, (1) le premier est dédié à la modélisation des problèmes multi-états et mono-agent : les Processus Décisionnels de Markov et (2) le deuxième est consacré à la modélisation des problèmes multi-agents et mono-état : les jeux matriciels. L'étude des approches dédiées à la résolution de ces formalismes, nous a permis de mettre en évidence un certain nombre de difficultés pour lesquelles chaque formalisme peut proposer des solutions à l'autre.

Cette thèse a été l'occasion d'employer de manière simultanée ces deux formalismes, nous amenant ainsi à introduire un tiers formalisme soudant les deux précédents et permettant de concevoir des problèmes qui sont à la fois multi-agents et multi-états : les jeux stochastiques.

Intuitivement, nous nous sommes ensuite intéressés aux méthodes de résolution des jeux stochastiques, principalement à l'approche par planification. En y regardant de plus près (à travers des travaux antérieurs sur le sujet), nous avons mis en évidence un certain nombre de difficultés. En effet, sélectionner un équilibre unique implique une forte centralisation au niveau des algorithmes de résolution et constitue par conséquent un inconvénient, limitant l'applicabilité de ces derniers dans des systèmes distribués. D'autant plus, que la complexité des algorithmes permettant de trouver un équilibre de Nash est importante, réduisant ainsi l'efficacité de ce genre d'approches. Partant de ce constat, nos travaux ont été motivés par deux principaux objectifs :

1. Proposer un algorithme de planification pour les jeux stochastiques à somme générale, en garantissant la décentralisation des calculs sans détériorer la qualité de la solution.
2. Augmenter l'efficacité de résolution des problèmes envisagés.

La suite de cette thèse a été alors organisée autour de ces deux axes. Nous avons tout d'abord procédé à une analyse des problèmes que nous souhaitions résoudre afin d'en proposer une solution adaptée. Nous nous sommes ensuite tournés vers la résolution de ces problèmes. Enfin, nous avons procédé à la validation de notre approche dans le but de vérifier que les solutions proposées répondaient aux objectifs précédemment cités.

Planification distribuée

Lorsque les agents s'apprêtent à jouer dans un jeu stochastique, ils ont besoin d'une politique, ou plus précisément d'une action jointe par état. Cette politique peut être considérée comme une fonction, indiquant à chaque instant de temps, ce que chaque agent doit faire. Cette politique possède également une propriété de stabilité, dans le sens où aucun agent ne préférera agir différemment. Si nous considérons une politique comme une suite d'équilibres, le problème qui se pose est : comment calculer de tels équilibres et comment limiter la multiplicité pour ne choisir qu'un seul équilibre.

FiniteVI [Kearns *et al.*, 2000] est un algorithme basé sur la technique d'itération sur les valeurs pour les jeux stochastiques à somme générale et à horizon fini. Pour choisir un équilibre unique à chaque état, l'algorithme utilise une certaine fonction de sélection f qui retourne toujours un équilibre unique. Les auteurs, ne précisent pas cette fonction et la considèrent comme un "oracle" auquel tous les agents doivent faire confiance.

Notre approche assure une meilleure autonomie pour chaque agent et propose différents critères de sélection pouvant focaliser les agents vers le choix d'un même équilibre. En effet, nous avons montré que vouloir sélectionner toujours le même équilibre représente en quelque sorte une borne optimale, mais est considéré aussi comme un problème difficile. Notre approche, a relaxé cette hypothèse en autorisant des politiques jointes sous-optimales. De ce fait, chaque agent calcule indépendamment son plan d'actions et est aiguillé tout simplement durant la phase de planification vers les équilibres pouvant rassembler tous les autres. Nous avons, illustré le fonctionnement de notre algorithme en utilisant une application classique de SMA : le déplacement d'objets par des agents. Cette illustration nous a permis, de montrer une légère dégradation de la qualité de la solution au profit d'une meilleure adaptabilité pour les problèmes liés à la robotique autonome.

Amélioration de l'algorithme

Déterminer un équilibre de Nash dans un jeu matriciel reste, jusqu'à nos jours, un problème difficile impliquant un coût du calcul important. Rajouter à cette difficulté, un espace d'états exponentiel en fonction du nombre d'agents et les algorithmes de résolution pour le modèle des jeux stochastiques deviennent alors rapidement inexploitables. Dans ce contexte, nous avons adapté une technique classique issue du domaine de la théorie des jeux pour diminuer la complexité du modèle des jeux stochastiques. Cette technique, a eu pour effet de (1) réduire les ressources utilisées pendant la phase de planification et également faciliter la prise de décision en limitant la multiplicité des équilibres. Les travaux que nous avons présentés contribuent par ailleurs à augmenter l'applicabilité des jeux stochastiques. En effet, nous avons pu tester l'algorithme avec des scénarios incluant jusqu'à cinq agents et ayant un espace d'états de quelques dizaines de millions d'états.

Perspectives de recherche

Dans un premier temps, une des perspectives de ce travail a pour objectif d'augmenter l'espace des problèmes pouvant être résolus. Pour ce faire, nous envisageons de nous intéresser à :

– Simplifier l'espace d'états : comme nous l'avons déjà montré, l'espace d'états est exponentiel en fonction du nombre d'agents. Une analyse préliminaire de cet espace, montre qu'un bon nombre d'états semble être inutile, car il représente des situations inatteignables. Il est par exemple envisageable, de construire l'espace d'états intelligemment en fonction de certaines règles pouvant indiquer si un état est jouable ou non. Une autre alternative, consiste à détecter durant la phase de planification,

que l'évaluation de certains états peut être ignorée, soit parce qu'ils sont considérés comme irréalisables ou soit car ils sont équivalents à d'autres états déjà évalués [Givan *et al.*, 2003].

- Augmenter le nombre d'agents : considérer un nombre d'agents de plus de deux agents, constitue un défi majeur pour le modèle des jeux stochastiques. En effet, le nombre d'agents impacte directement l'ensemble des actions et ainsi la taille des matrices de paiements. Or, calculer un équilibre de Nash même pour deux agents, est considéré comme un problème difficile, ce qui peut constituer un obstacle. En revanche, certains chercheurs se posent la question sur la nécessité de la convergence des algorithmes de résolution vers un équilibre de Nash. Il se peut qu'au lieu de chercher à calculer un équilibre de Nash, on s'intéresse plutôt à trouver un équilibre ayant certaines propriétés spécifiques. Parmi ces propriétés, nous citons :
  - La garantie d'un certain gain : étant donné un jeu matriciel, existe t-il un équilibre pouvant assurer à l'agent i d'obtenir un certain gain ?
  - La garantie d'un "bien-être" social : étant donné un jeu matriciel, existe t-il un équilibre où l'union des utilités des agents dépasse une certaine valeur ?

La mise en oeuvre de ces perspectives nécessitera de réviser le modèle. En effet, la description des états, des actions et la définition de la fonction de transition devront très probablement être adaptées afin de modéliser au mieux ces idées.

Le travail présenté dans cette thèse s'est essentiellement appuyé sur une application liée à des agents "déménageurs". Nous pensons toutefois que notre travail ne se restreint pas à ce cadre. Nous souhaitons donc également étendre l'applicabilité de notre approche à des projets plus divers, comme la simulation d'un trafic routier [Doniec *et al.*, 2007]. Le trafic routier peut se définir comme l'ensemble des phénomènes complexes résultant du déplacement d'usagers sur un réseau routier de capacité limitée. La difficulté réside dans la modélisation d'un carrefour donné en un jeu stochastique. Les usagers de la route sont facilement assimilables à des agents, ayant des actions et des buts. En revanche, certaines de leurs caractéristiques (comme la durée de leurs actions, l'accélération ou la capacité de freinage) sont difficilement représentable en environnement discret. Nous pensons qu'une adaptation est *a priori* possible, de la même manière qu'avec un FMDP [Boutilier *et al.*, 1999, Guestrin *et al.*, 2003].

Bibliographie

[Apt et Erich, 2011] Apt, K. R. et Erich, G. (2011). *Lectures in game theory for computer scientists*. Cambridge : Cambridge University Press. Formerly CIP.

[Barbu et Limnios, 2008] Barbu, V. et Limnios, N. (2008). *Semi-Markov chains and hidden semi-Markov models toward applications : their use in reliability and DNA analysis*. Lecture notes in statistics. Springer.

[Bellman, 1957a] Bellman, R. (1957a). *Dynamic programming*. Rand Corporation research study. Princeton University Press.

[Bellman, 1957b] Bellman, R. (1957b). A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684.

[Bernstein *et al.*, 2000] Bernstein, D., Zilberstein, S. et Immerman, N. (2000). The complexity of decentralized control of Markov decision processes. *In UAI '00 : Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, pages 32–37, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Blum et Furst, 1995] Blum, A. L. et Furst, M. L. (1995). Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1):1636–1642.

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. *In Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, TARK '96, pages 195–210, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Boutilier, 1999] Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. *In IJCAI '99 : Proc. of the Sixteenth Int. Joint Conf. on Artificial Intelligence*, pages 478–485, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Boutilier *et al.*, 1999] Boutilier, C., Dean, T. et Hanks, S. (1999). Decision-Theoretic Planning : Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research*, 11:1–94.

[Bowling, 2003] Bowling, M. (2003). *Multiagent Learning in the Presence of Agents with Limitations*. Thèse de doctorat, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-03-118.

[Bowling et Veloso, 2002] Bowling, M. et Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250.

[Bratman *et al.*, 1988] Bratman, M., Israel, D. et Pollack, M. (1988). Plans and resource bounded practical reasoning. *Computational Intelligence*, 4:349–355.

[Breton, 1986] Breton, M. (1986). *Équilibres pour des jeux séquentiels*. Thèse de doctorat, Université de Montréal.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artif. Intell.*, 47:139–159.

[Brown, 1951] Brown, G. (1951). Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13:374–376.

[Burkov et Chaib-Draa, 2008] Burkov, A. et Chaib-Draa, B. (2008). Distributed planning in stochastic games with communication. *In Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, pages 396–401, Washington, DC, USA. IEEE Computer Society.

[Chang *et al.*, 1997] Chang, R., Chor, B., Goldreich, O., Hartmanis, J., Hastad, J., Ranjan, D. et Rohatgi, P. (1997). The random oracle hypothesis is false. Rapport technique, Journal of Computer and System Sciences.

[Chatterjee, 2002] Chatterjee, K. (2002). Complexity of strategies and multiplicity of Nash equilibria. *Group Decision and Negotiation*, 11:223–230. 10.1023/A :1015236512713.

[Chen *et al.*, 2006] Chen, X., Deng, X. et Teng, S. (2006). Computing Nash equilibria : Approximation and smoothed complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(023).

[Claus et Boutilier, 1998] Claus, C. et Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *In In Proceedings of National Conference on Artificial Intelligence (AAAI-98*, pages 746–752.

[Conitzer et Sandholm, 2008] Conitzer, V. et Sandholm, T. (2008). New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621–641.

[Cottle *et al.*, 2009] Cottle, R., Pang, J. et Stone, R. (2009). *The Linear Complementarity Problem*. Classics in applied mathematics. Society for Industrial and Applied Mathematics.

[Cottle, 1966] Cottle, R. W. (1966). Nonlinear programs with positively bounded jacobians. *SIAM Journal on Applied Mathematics*, 14(1):147–158.

[Daskalakis *et al.*, 2006] Daskalakis, C., Goldberg, P. et Papadimitriou, C. (2006). The complexity of computing a Nash equilibrium. *In STOC '06 : Proc. of the thirty-eighth annual ACM symposium on Theory of computing*, pages 71–78, New York, NY, USA. ACM.

[Debreu, 1952] Debreu, G. (1952). A social equilibrium existence theorem. *Proceedings of the National Academy of Sciences of the United States of America*, 38(10):886–893.

[Doniec *et al.*, 2007] Doniec, A., Mandiau, R., Espié, S. et Piechowiak, S. (2007). Comportements anticipatifs dans les systèmes multi-agents. application à la simulation de trafic routier. *Revue d'Intelligence Artificielle*, pages 183–221.

[Durfee *et al.*, 1987] Durfee, E., Lesser, V. et Corkill, D. (1987). Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transactions on Computers*, C36(11):1275–1291.

[Durfee, 2001] Durfee, E. H. (2001). *Multi-Agents Systems and Applications*, chapitre Distributed problem solving and planning, pages 118–149. Springer-Verlag, Berlin, Germany.

[Emery-Montemerlo *et al.*, 2004] Emery-Montemerlo, R., Gordon, G., Schneider, J. et Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *In In Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 136–143.

[Ferber, 1995] Ferber, J. (1995). *Les systèmes multi-agents : vers une intelligence collective*. Informatique, Intelligence Artificielle. InterÉditions.

[Fikes et Nilsson, 1971] Fikes, R. E. et Nilsson, N. J. (1971). Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208.

[Fink, 1964] Fink, A. M. (1964). Equilibrium in a stochastic n-person game. *Journal of Science in Hiroshima University, Series A-I*, 28:89–93.

[Fischer *et al.*, 2006] Fischer, F. A., Holzer, M. et Katzenbeisser, S. (2006). The influence of neighbourhood and choice on the complexity of finding pure Nash equilibria. *Inf. Process. Lett.*, pages 239–245.

[Franklin et Graesser, 1997] Franklin, S. et Graesser, A. (1997). Is it an agent, or just a program ? : A taxonomy for autonomous agents. *In Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35, London, UK. Springer-Verlag.

[Fudenberg et Levine, 1998] Fudenberg, D. et Levine, D. (1998). *The theory of learning in games*. MIT Press series on economic learning and social evolution. MIT Press.

[Fudenberg et Tirole, 1991] Fudenberg, D. et Tirole, J. (1991). *Game theory*. MIT Press, Cambridge, MA.

[Gatti, 2005] Gatti, J. (2005). A note on the existence of Nash equilibrium in games with discontinuous payoffs. Cambridge Working Papers in Economics 0510, Faculty of Economics, University of Cambridge.

[Georgeff *et al.*, 1987a] Georgeff, M., Lansky, A., for Artificial Intelligence, A. A., for the Study of Language, C. et (U.S.), I. (1987a). *Reasoning about actions & plans : proceedings of the 1986 workshop : June 30-July 2, 1986, Timberline, Oregon*. M. Kaufmann Publishers.

[Georgeff *et al.*, 1987b] Georgeff, M., Lansky, A. et Schoppers, M. (1987b). Reasoning and planning in dynamic domains : an experiment with a mobile robot. Rapport technique, Artificial Intelligence Center, SRI International,.

[Ghallab *et al.*, 2004] Ghallab, M., Nau, D. et Traverso, P. (2004). *Automated planning : theory and practice*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier/Morgan Kaufmann.

[Gibbons, 1992] Gibbons, R. (1992). *Game theory for applied economists*. Princeton paperbacks. Princeton University Press.

[Givan *et al.*, 2003] Givan, R., Dean, T. et Greig, M. (2003). Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.*, 147:163–223.

[Glicksberg, 1952] Glicksberg, I. L. (1952). A further generalization of the Kakutani fixed point theorem, with application to Nash equilibrium points. *Proceedings of the American Mathematical Society*, 3(1):170–174.

[Goldman et Zilberstein, 2004] Goldman, C. V. et Zilberstein, S. (2004). Decentralized control of cooperative systems : categorization and complexity analysis. *J. Artif. Int. Res.*, 22:143–174.

[Guerrien, 2002] Guerrien, B. (2002). *La théorie des jeux*. Economica.

[Guestrin *et al.*, 2003] Guestrin, C., Koller, D., Parr, R. et Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *J. Artif. Int. Res.*, 19:399–468.

[Hamila *et al.*, 2009] Hamila, M. A., Grislin, E., Mandiau, R. et Mouaddib, A.-I. (2009). Jeux stochastiques à somme générale pour la coordination multi-agents. *In Journées Francophones sur les Systèmes Multi-Agents*, Lyon, France.

[Hamila *et al.*, 2010a] Hamila, M. A., Grislin, E., Mandiau, R. et Mouaddib, A.-I. (2010a). An Algorithm for Multi-robot Planning : SGInfiniteVI. *In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '10, pages 141–148, Washington, DC, USA. IEEE Computer Society.

[Hamila *et al.*, 2010b] Hamila, M. A., Grislin, E., Mandiau, R. et Mouaddib, A.-I. (2010b). Les jeux stochastiques : un modèle de coordination multi-agents. *In Proceedings of RFIA 2010*, Caen. AFRIF.

[Hamila *et al.*, 2012] Hamila, M. A., Grislin, E., Mandiau, R. et Mouaddib, A.-I. (2012). Strategic dominance and dynamic programming for multi-agent planning - application to the multi-robot box-pushing problem. *In ICAART 2012 - Proceedings of the 4th International Conference on Agents and Artificial Intelligence*, Vilamoura - Algarve, Portugal. SciTePress.

[Hansen, 1998] Hansen, E. (1998). Solving POMDPs by searching in policy space. *In Proceedings of the Proceedings of the Fourteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 211–219, San Francisco, CA. Morgan Kaufmann.

[Hansen, 2004] Hansen, E. A. (2004). Dynamic programming for partially observable stochastic games. *In In Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 709–715.

[Harsanyi, 1973] Harsanyi, J. (1973). *Can the maximin principle serve as a basis for morality ? : a critique of John Rawls's theory.* Working papers in management science. Center for Research in Management Science, University of California, Berkeley.

[Hendler, 1992] Hendler, J. (1992). *Artificial intelligence planning systems : proceedings of the first international conference, June 15-17, 1992, College Park, Maryland.* M. Kaufmann.

[Hoffman et Karp, 1966] Hoffman, A. J. et Karp, R. M. (1966). On nonterminating stochastic games. *Management Science*, 12(5):359–370.

[Horling et Lesser, 2005] Horling, B. et Lesser, V. (2005). A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review*, 19(4):281–316.

[Howard, 1960] Howard, R. (1960). *Dynamic Programming and Markov Processes.* Cambridge, Mass. : M.I.T. Press.

[Hu et Wellman, 2003] Hu, J. et Wellman, M. (2003). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039–1069.

[Jennings *et al.*, 1998] Jennings, N. R., Sycara, K. et Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38.

[Kaelbling *et al.*, 1998] Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

[Kearns *et al.*, 2000] Kearns, M., Mansour, Y. et Singh, S. (2000). Fast planning in stochastic games. *In In Proc. UAI-2000*, pages 309–316. Morgan Kaufmann.

[Lemke et Howson, 1964] Lemke, C. E. et Howson, J. T. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

[Leroux et Livet, 2006] Leroux, A. et Livet, P. (2006). *Leçons de philosophie économique : Économie normative et philosophie morale*. Leçons de philosophie économique. Economica.

[Leyton-Brown et Shoham, 2008] Leyton-Brown, K. et Shoham, Y. (2008). Essentials of game theory : A concise multidisciplinary introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2(1):1–88.

[Littman, 1994] Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. *In In Proc. of the Eleventh Int. Conf. on Machine Learning*, pages 157–163. Morgan Kaufmann.

[Littman, 1996] Littman, M. L. (1996). *Algorithms for sequential decision-making*. Thèse de doctorat, Brown University, Providence, RI, USA. AAI9709069.

[Littman *et al.*, 1995] Littman, M. L., Dean, T. L. et Kaelbling, L. P. (1995). On the Complexity of Solving Markov Decision Problems. *In In Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence*, pages 394–402.

[Mailath et Samuelson, 2006] Mailath, G. et Samuelson, L. (2006). *Repeated games and reputations : long-run relationships*. Oxford University Press.

[McKelvey et McLennan, 1996] McKelvey, R. D. et McLennan, A. (1996). Computation of equilibria in finite games. *In Handbook of Computational Economics*, pages 87–142. Elsevier.

[Nash, 1951] Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2): 286–295.

[Nash, 1950] Nash, J. F. (1950). Equilibrium points in n-person games. *Proc. of the National Academy of Sciences of the United States of America*, 36(1):48–49.

[Neumann *et al.*, 2007] Neumann, J., Morgenstern, O., Rubinstein, A. et Kuhn, H. (2007). *Theory of games and economic behavior*. Princeton Classic Editions. Princeton University Press.

[Ng, 2004] Ng, Y. (2004). *Welfare economics : towards a more complete analysis*. Palgrave Macmillan.

[Osborne et Rubinstein, 1994] Osborne, M. et Rubinstein, A. (1994). *A Course in Game Theory*. The MIT Press.

[Papadimitriou et Tsitsiklis, 1987] Papadimitriou, C. et Tsitsiklis, J. (1987). The complexity of Markov chain decision processes. *Journal of Complexity*, 12(3):441–450.

[Pineau *et al.*, 2003] Pineau, J., Gordon, G. et Thrun, S. (2003). Point-based value iteration : An anytime algorithm for POMDPs. *In International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025 – 1032.

[Pollatschek et Avi-Itzhak, 1969] Pollatschek, M. A. et Avi-Itzhak, B. (1969). Algorithms for stochastic games with geometrical interpretation. *Management Science*, 15(7):399–415.

[Porter *et al.*, 2004] Porter, R., Nudelman, E. et Shoham, Y. (2004). Simple search methods for finding a Nash equilibrium. *In Games and Economic Behavior*, pages 664–669.

[Porter *et al.*, 2008] Porter, R., Nudelman, E. et Shoham, Y. (2008). Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2):642–662.

[Puterman, 1994] Puterman, M. (1994). *Markov decision processes : discrete stochastic dynamic programming*. Wiley series in probability and statistics. Wiley-Interscience.

[Radner, 1980] Radner, R. (1980). Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. *Journal of Economic Theory*, 22(2):136–154.

[Robinson, 1951] Robinson, J. (1951). An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301.

[Russell et Norvig, 2003] Russell, S. et Norvig, P. (2003). *Artificial Intelligence : A Modern Approach*. Prentice Hall, second édition.

[Schmidt, 1988] Schmidt, C. (1988). Programme de recherche benthamien et économie politique britannique. Deux rendez-vous manqués. *Revue Économique*, 39(4):809–840.

[Sen *et al.*, 1982] Sen, A., Williams, B. et Williams, B. (1982). *Utilitarianism and beyond*. Cambridge University Press.

[Seuken, 2007] Seuken, S. (2007). Memory-bounded dynamic programming for DEC-POMDPs. *In In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI*, pages 2009–2015.

[Seuken et Zilberstein, 2007] Seuken, S. et Zilberstein, S. (2007). Improved Memory-Bounded Dynamic Programming for Dec-POMDPs. *In Proc. of Uncertainty in Artificial Intelligence*, pages 2009–2015.

[Seuken et Zilberstein, 2008] Seuken, S. et Zilberstein, S. (2008). Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250.

[Shapley, 1953] Shapley, L. (1953). Stochastic games. *Proc. of the National Academy of Sciences USA*, pages 1095–1100.

[Shoham et Leyton-Brown, 2009] Shoham, Y. et Leyton-Brown, K. (2009). *Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, New York.

[Shoham *et al.*, 2003] Shoham, Y., Powers, R. et Grenager, T. (2003). Multi-agent reinforcement learning : a critical survey. Rapport technique, Stanford University.

[Sigaud et Buffet, 2008] Sigaud, O. et Buffet, O. (2008). *Processus décisionnels de Markov en intelligence artificielle*, volume 1 - principes généraux et applications de *IC2 - informatique et systèmes d'information*. Lavoisier - Hermes Science Publications.

[Singh *et al.*, 2000] Singh, S., Kearns, M. et Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. *In In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 541–548. Morgan.

[Sondik, 1978] Sondik, E. J. (1978). The optimal control of partially observable Markov processes over the infinite horizon : Discounted costs. *Operations Research*, 26:282–304.

[Sorin, 2002] Sorin, S. (2002). *A first course on zero-sum repeated games*. Mathématiques & applications. Springer-Verlag.

[Stengel, 1999] Stengel, B. V. (1999). New maximal numbers of equilibria in bimatrix games. *Discrete and Computational Geometry*, 21:557–568.

[Stengel, 2002] Stengel, B. v. (2002). Computing equilibria for two-person games. *In* Aumann, R. et Hart, S., éditeurs : *Handbook of Game Theory with Economic Applications*, volume 3, chapitre 45, pages 1723–1759. Elsevier, first édition.

[Sutton et Barto, 1998] Sutton, R. et Barto, A. (1998). *Reinforcement learning : an introduction*. Adaptive computation and machine learning. MIT Press.

[Sutton, 1988] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *In Machine Learning*, pages 9–44. Kluwer Academic Publishers.

[Sutton *et al.*, 1999] Sutton, R. S., Precup, D. et Singh, S. (1999). Between MDPs and Semi-MDPs : A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181–211.

[Szepesvári et Littman, 1998] Szepesvári, C. et Littman, M. L. (1998). A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation*, 11:11–8.

[Szer et Charpillet, 2006] Szer, D. et Charpillet, F. (2006). Point-based dynamic programming for DEC-POMDPs. *In proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1233–1238. AAAI Press.

[Takahashi, 1964] Takahashi, M. (1964). Equilibrium points of stochastic noncooperative n-person games. *Journal of Science of the Hiroshima University.*

[Uther et Veloso, 1997] Uther, W. T. B. et Veloso, M. M. (1997). Generalizing adversarial reinforcement learning. Rapport technique, In Proceedings of the AAAI Fall Symposium on Model Directed Autonomous Systems.

[Varian *et al.*, 2008] Varian, H., Hommet, J. et Thiry, B. (2008). *Analyse microéconomique.* Ouvertures économiques. Série Balises. De Boeck.

[Vrieze, 1987] Vrieze, O. (1987). *Stochastic games with finite state and action spaces.* Amsterdam : Centrum voor wiskunde en informatica.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards.* Thèse de doctorat, Cambridge University, England.

[Watkins et Dayan, 1992] Watkins, C. J. C. H. et Dayan, P. (1992). Technical note Q-Learning. *Machine Learning*, 8:279–292.

[Weerdt *et al.*, 2005] Weerdt, M. D., Mors, A. T. et Witteveen, C. (2005). Multi-agent planning : An introduction to planning and coordination. Rapport technique, In : Handouts of the European Agent Summer.

[Weiss, 1999] Weiss, G. (1999). *Multiagent systems : a modern approach to distributed artificial intelligence.* Intelligent Robotics and Autonomous Agents. MIT Press.

[Wooldridge, 2009] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* John Wiley & Sons.

[Young, 1993] Young, H. P. (1993). The evolution of conventions. *Econometrica*, 61(1): 57–84.

[Zilberstein, 1996] Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83.

[Zinkevich *et al.*, 2005] Zinkevich, M., Greenwald, A. et Littman, M. (2005). Cyclic equilibria in Markov games. *In Advances in Neural Information Processing Systems.* MIT.

Résumé

Planifier les actions d'un agent dans un environnement dynamique et incertain, a été largement étudié et le cadre des processus décisionnels de Markov offre les outils permettant de modéliser et de résoudre de tels problèmes. Le domaine de la théorie des jeux, a permis l'étude des interactions stratégiques entre plusieurs agents pour un jeu donné. La cadre des jeux stochastiques, est considéré comme une généralisation du domaine des processus décisionnels de Markov et du champ de la théorie des jeux et permet de modéliser des systèmes ayant plusieurs agents et plusieurs états. Cependant, planifier dans un système multi-agents est considéré comme difficile, car la politique d'actions de l'agent dépend non seulement de ses choix mais aussi des politiques des autres agents.

Le travail que nous présentons dans cette thèse porte sur la prise de décision distribuée dans les systèmes multi-agents. Les travaux existants dans le domaine, permettent la résolution théorique des jeux stochastiques mais imposent de fortes restrictions et font abstraction de certains problèmes cruciaux du modèle. Nous proposons un algorithme de planification décentralisée pour le modèle des jeux stochastiques, d'une part basé sur l'algorithme Value-Iteration et d'autre part basé sur la notion d'équilibre issue de la résolution des jeux matriciels. Afin d'améliorer le processus de résolution et de traiter des problèmes de taille importante, nous recherchons à faciliter la prise de décision et à limiter les possibilités d'actions à chaque étape d'interaction. L'algorithme que nous avons proposé, a été validé sur un exemple d'interaction incluant plusieurs agents et différentes expérimentations ont été menées afin d'évaluer la qualité de la solution obtenue.

Mots-clés: Jeux stochastiques, planification, équilibre de Nash, jeux matriciels, processus décisionnels de Markov

Abstract

Planning agent's actions in a dynamic and uncertain environment has been extensively studied. The framework of Markov decision process provides tools to model and solve such problems. The field of game theory has allowed the study of strategic interactions between multiple agents for a given game. The framework of stochastic games is considered as a generalization of the fields of Markov decision process and game theory. It allows to model systems with multiple agents and multiple states.

However, planning in a multi-agent system is considered difficult : agent's decisions depend not only on its actions but also on actions of the other agents. The work presented in this thesis focuses on decision making in distributed multi-agent systems. Existing works in this field allow the theoretical resolution of stochastic games but place severe restrictions and ignore some crucial problems of the model. We propose a decentralized planning algorithm for the model of stochastic games. Our proposal is based on the Value-Iteration algorithm and on the concept of Nash equilibrium. To improve the resolution process and to deal with large problems, we sought to ease decision making and limit the set of joint actions at each stage. The proposed algorithm was validated on a coordination problem including several agents and various experiments were conducted to assess the quality of the resulting solution.

Keywords: Stochastic games ; Planning ; Nash equilibrium ; Matrix game ; MDP