Les jeux stochastiques : un modèle de coordination multi-agents

M.A. Hamila[1,2,3] E. Grislin-le Strugeon[1,2,3] R. Mandiau[1,2,3] A.I. Mouaddib[4]

[1] Univ Lille Nord de France, F-59000 Lille, France
[2] UVHC, LAMIH, F-59313 Valenciennes, France
[3] CNRS, UMR 8530, F-59313 Valenciennes, France
[4] GREYC, Université de Caen Basse-Normandie, France

{mohamed.hamila,rene.mandiau,emmanuelle.grislin}@univ-valenciennes.fr
mouaddib@info.unicaen.fr

Résumé

Le formalisme des jeux stochastiques offre un cadre intéressant pour modéliser les problèmes de planification et de coordination dans les systèmes multi-agents. Le but de ce travail, est de permettre à des agents simples de travailler conjointement, pour l'accomplissement de leurs buts individuels ou globaux. Nous proposons un algorithme basé sur la programmation dynamique, permettant de converger vers des équilibres qui induisent des comportements d'agents "coopératifs". Nous avons testé et comparé notre méthode de résolution avec celle d'une approche centralisée (MMDP, i.e. Multi-Agent Markov Decision Process). Les résultats préliminaires montrent l'intérêt du modèle, malgré une complexité plus importante.

Mots Clef

Coordination multi-agents, décision, jeux stochastiques, équilibre de Nash.

Abstract

The formalism of the stochastic games offers an interesting setting to model the problems of planning and coordination in the multi-agent systems. The aim of this work, is to allow simple agents to work jointly, for the achievment of their individual or global goals. We propose an algorithm based on the dynamic programming, permitting to converge towards equilibria which induce "cooperative" behaviors. We tested and compared our resolution method with a centralized approach (MMDP, i.e. Multi-Agent Markov Decision Process). Preliminary results show the interest of the model, despite a greater complexity.

Keywords

Multi-agent coordination, decision, stochastic games, Nash equilibrium.

1 Introduction

Le point clé des systèmes multi-agents réside dans la formalisation de la coordination entre les agents. Dans ce papier, nous nous focalisons sur l'un des formalismes existants : les jeux stochastiques. En effet, le cadre des jeux stochastiques fournit une base théorique pour de nombreux travaux récents [1, 18, 19, 28], traitant de la planification et de l'apprentissage multi-agents. Un jeu stochastique peut être considéré comme une extension des processus décisionnels de Markov (MDP) [17, 23, 27] et de la théorie des jeux [21, 22]. Désormais, il est possible d'y avoir plusieurs agents, pouvant être (non)-coopératifs (voire assortis d'objectifs contradictoires), dont les actions impactent directement les récompenses résultantes et le choix de l'état suivant.

La classe de problème traitée dans ce papier, décrit les tâches de décision séquentielle multi-agents [3, 4] dans l'incertain, c'est-à-dire pour lesquelles il est nécessaire, à chaque étape, de décider quelle action jointe choisir. Nous supposons qu'il n'y pas de communication directe entre agents et qu'un consensus global doit émerger à partir de l'interaction, d'une part entre les différents agents et d'autre part avec l'environnement qui les entoure. De ce fait, il nous semble naturel d'envisager que les jeux stochastiques peuvent modéliser typiquement un tel système. Ce papier tente de présenter une généralisation des travaux antérieurs initiés dans [16]. Nous nous attachons ainsi à déterminer des politiques jointes afin de faire se coordonner les agents pour l'accomplissement de leurs objectifs.

Le plan de l'article comprend quatre sections. Dans la section 2, nous rappelons les définitions et notations des jeux stochastiques et leurs méthodes de résolution. Dans la section 3, nous décrivons un algorithme générique pour la résolution et nous appliquons le modèle des jeux stochastiques à un exemple de coordination illustratif. Quelques résultats préliminaires sont présentés et discutés dans la section 4 et enfin nous concluons dans la section 5.

2 Les jeux stochastiques

Dans cette section, nous présenterons d'une part une introduction aux jeux stochastiques, et d'autre part deux aspects cruciaux du modèle, à savoir l'équilibre de Nash et les méthodes de résolution.

2.1 Notions

Un jeu stochastique est défini par le tuple [14, 25, 26] :

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

- Ag : Ensemble fini des joueurs.
- A_i : Ensemble fini des stratégies du joueur i ($i \in Ag$). Une stratégie pure correspond à une action choisie de façon déterministe, contrairement à une stratégie mixte, définie comme une distribution de probabilité sur l'ensemble des stratégies pures.
- R_i : Fonction de récompense du joueur i , $R_i(a) \rightarrow \mathbb{R}$.
- S : Ensemble fini des états du jeu.
- T : Fonction de transition du jeu, $T : S \times A \times S \rightarrow [0, 1]$, indiquant la probabilité de passer d'un état $s \in S$ à un état $s' \in S$ en exécutant l'action jointe $a =< a_1, a_2, \dots, a_{|Ag|} > (a \in A)$. Cette fonction vérifie la propriété de Markov (le futur ne dépend que du présent), c'est pourquoi les jeux stochastiques sont souvent appelés jeux de Markov.

Les jeux stochastiques ressemblent au cadre des MDPs [23, 26], sauf que nous avons plusieurs agents sélectionnant simultanément différentes actions et que l'état suivant ainsi que les récompenses dépendent désormais de l'action jointe de tous les agents. Il est également important de noter que chaque agent possède une fonction de récompense distincte de celle des autres. L'objectif principal d'un agent est de sélectionner les actions qui maximisent au plus ses futurs gains.

Ces jeux, peuvent être vus aussi comme une extension de la théorie des jeux avec plusieurs états. L'ensemble de tous les gains possibles pouvant être obtenus est représenté par une matrice. Dans ces jeux il y a deux joueurs (i.e. agents), le premier joueur sélectionne les lignes et le deuxième sélectionne les colonnes de la matrice. Les choix conjointement faits détermineront les gains de chacun en fonction de la matrice jouée.

Considérons l'exemple montré en tableau 1. Il représente un jeu à somme générale [5], car il n'y a aucune restriction sur la nature des gains. Si le premier joueur choisit la première ligne et que le deuxième choisit la deuxième colonne, alors ils auront respectivement les paiements 0 et -15.

	Joueur 2	
Joueur 1	(-8,-8)	(0,-15)
	(-15,0)	(-1,-1)

TABLE 1 – Jeu à somme générale : le dilemme du prisonnier.

Dans les jeux stochastiques, chaque état s représente une matrice M^s. Par exemple, supposons que les deux joueurs commencent le jeu à l'étape s, ils jouent alors la matrice M^s (voir partie 2.2). Juste après, les joueurs obtiennent un certain paiement. Le jeu passe alors à $M^{s'}$ avec une probabilité $P(s'|s, (i_s, j_s))$ en fonction des actions (i_s, j_s) jouées à l'étape s et ainsi de suite.

2.2 Equilibre de Nash

Contrairement aux MDPs, il devient difficile d'employer le verbe "résoudre" pour les jeux stochastiques. Une action ne peut être évaluée que si les autres actions sont déjà connues. Par conséquent, il ne peut exister d'action optimale indépendamment de celles des autres joueurs. On cherche plutôt à déterminer des équilibres, exprimant la notion de "Best-Response" (BR).

Définition 1 *Etant données les meilleures actions des autres joueurs a^*_{-i}, la meilleure réponse (BR) du joueur i est :*

$$BR_i : a^*_{-i} \rightarrow argmax_{a_i \in A_i} \; R_i(< a_i, a^*_{-i} >) \;\; \forall a_i \in A_i$$

Le concept le plus utilisé comme solution dans les jeux stochastiques est l'équilibre de Nash :

Définition 2 *Un équilibre de Nash est un ensemble de stratégies (actions) a^*, tel que :*

$$R(< a^*_i, a^*_{-i} >) \;\geq\; R(< a_i, a^*_{-i} >) \;\; \forall a_i \in A_i$$

La notion d'équilibre de Nash peut alors s'écrire en utilisant la notion de meilleure réponse :

$$a^*_i \;\in\; BR_i(a^*_{-i}) \;\; \forall i$$

Autrement dit, *dans un équilibre de Nash aucun joueur n'a intérêt à changer de stratégie si les autres joueurs continuent à suivre leurs parts de l'équilibre*. Par exemple, dans le dilemme du prisonnier, l'issue (1,1) est un équilibre de Nash.

Notion de politique
Pratiquement, un équilibre de Nash est une action jointe. L'ensemble de tous les équilibres, un par état, constitue une politique :

$$\Pi : S \rightarrow A, \;\; A \;\in \{A_1 \times A_2 \times ... \times A_{|Ag|}\}$$

Plus simplement, une politique Π définit un chemin, une séquence d'actions à partir de la position de départ jusqu'à la destination finale. Dans cet article, nous limitons notre étude particulièrement aux politiques stationnaires [1].

1. Notons que, les politiques non-stationnaires, autorisant le choix des actions en fonction de l'histoire du jeu, sont plus complexes et relativement moins bien étudiées dans le cadre des jeux stochastiques [19].

2.3 Résolution des jeux stochastiques

Dans cette section, nous allons discuter de certaines solutions aux jeux stochastiques. En règle générale, nous pouvons classer les algorithmes de résolution en deux grandes catégories :

– L'apprentissage qui consiste à apprendre la politique en faisant agir les agents dans le monde (on-line), les données pouvant servir soit à calculer la politique directement, soit à estimer un modèle du monde à partir duquel on calculera ensuite la politique [5].
– La planification qui consiste à calculer une séquence d'actions conduisant à un but. Le calcul de la politique optimale se fait hors ligne (off-line) à partir d'un modèle du monde préconnu.

Nous nous concentrons particulièrement sur la dernière catégorie, dans laquelle l'algorithme de Shapley constitue l'algorithme de base.

Algorithme de Shapley et jeux stochastiques à somme nulle.

Notre travail a été inspiré par celui de Shapley [24] traitant de la planification dans les jeux stochastiques. L'algorithme de Shapley [6] peut être vu comme une extension de l'algorithme "Value Iteration", il consiste à chercher initialement un équilibre de Nash de façon centralisée dans un jeu stochastique fini escompté à deux joueurs et à somme nulle [2]. Il est composé de deux parties :

– La première partie est la "différence temporelle", nécessaire pour résoudre la multiplicité d'états du jeu.
– La deuxième partie est "le jeu" proprement dit, c'est-à-dire la recherche d'équilibre pour le système multi-agents. Cette opération se déroule en deux étapes :

1. On construit (récursivement) la matrice M^s correspondant à s (l'état en cours) avec l'équation de Bellman [2] suivante :

$$M^s_a = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')\, V_{t-1}(s')$$

2. L'application de la fonction Value() à la matrice M^s, permettra d'obtenir un vecteur de valeurs correspondant à un équilibre de Nash. En effet, cette fonction n'est que l'exécution de l'algorithme Minimax [13], disposant d'un temps d'exécution polynomial en la taille de la matrice.

Cet algorithme trouve un équilibre de Nash en un temps polynomial, pour n'importe quel jeu stochastique à somme nulle et à horizon infini [24]. Il nous servira de base pour l'élaboration d'un algorithme dédié aux jeux stochastiques à somme générale (voir partie 3.1).

Les jeux stochastiques à somme générale.

Les jeux stochastiques à somme générale ont des fonctions de récompense qui peuvent être différentes pour chaque agent. Dans certains cas, il peut alors être difficile de trouver des politiques qui maximisent le critère de performance pour tous les agents. C'est pourquoi dans les jeux stochastiques à somme générale, un point d'équilibre est recherché. Cet équilibre est une situation dans laquelle aucun agent ne pourra améliorer son critère de performance s'il est le seul à changer de politique. On retrouve ici la définition d'un équilibre de Nash. L'équilibre de Nash permet notamment de répondre aux mieux aux objectifs individuels des agents. Cependant, la question de la convergence vers cet équilibre, n'est pas encore résolue. Récemment, Zinkevich, Greenwald et Littman [28] ont montré que :

1. L'algorithme Value-Iteration peut ne pas converger dans le cas des jeux stochastiques à somme générale avec horizon infini.
2. L'algorithme converge toujours vers des politiques non pas stationnaires mais cycliques.

En effet, la convergence est difficile (voir parfois impossible), du fait que dans un même jeu :

– Il peut y avoir plusieurs équilibres.
– Comme il ne peut y avoir aucun équilibre.

Dans ce contexte d'incertitude sur la convergence de l'algorithme, nous voulons dans un premier temps adopter une approche empirique, visant à évaluer le comportement de l'algorithme appliqué à un exemple d'interaction incluant plusieurs agents ($|Ag| \geq 2$).

Dans un deuxième temps, nous voulons comparer la politique jointe obtenue à celle issue d'un contrôle centralisé, en l'occurrence avec un MMDP [3] [4]. Dans ce formalisme, un processus central dispose de l'ensemble des informations et calcule la politique jointe afin de redistribuer cette politique parmi les agents. Les deux formalismes sont différents (architecture et méthode de résolution), mais leur utilisation est très proche (calcul centralisé avec distribution de la politique d'actions).

La comparaison avec un MMDP revient donc à comparer

Algorithme 1 Algorithme de Shapley

ENTRÉES: Un jeu stochastique JS, $\gamma \in [0,1]$: degré d'importance du futur et $\epsilon \geq 0$: condition d'arrêt

pour tout $s \in S$ **faire**
  $V_0(s) = 0$
fin pour
$t \leftarrow 0$
répéter
  $t \leftarrow t + 1$
  pour tout $s \in S$ **faire**
    // Remplir la matrice M^s
    $M^s = \{\forall a \in A, M^s_a = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_{t-1}(s')\}$
    // Chercher un équilibre
    $V_t(s) = Value[M^s]$
  fin pour
jusqu'à $max_{s \in S} |V_t(s) - V_{t-1}(s)| < \epsilon$

2. Jeu à somme nulle : si les gains de l'un représentent les pertes de l'autre.

3. MMDP : Multi-agent Markov Decision Process.

avec le cas optimal. Elle permettra ainsi de mesurer la qualité de la solution et par la suite de décider de la justesse d'utilisation de l'équilibre de Nash pour la coordination entre agents.

3 Proposition d'un algorithme et application sur un exemple d'interaction entre agents coopératifs

Dans cette section, nous présentons un algorithme de résolution pour les jeux stochastiques à somme générale, puis nous allons appliquer le modèle à un scénario d'interaction entre agents.

3.1 Adaptation de l'algorithme de Shapley aux jeux stochastiques à somme générale

En tenant compte de la simplicité structurelle de l'algorithme de Shapley, nous proposons un algorithme dédié aux jeux stochastiques à somme générale et à horizon infini (voir algorithme 2). Il prend en entrée : un jeu stochastique JS, γ (degré d'importance du futur) et ϵ (condition d'arrêt) et renvoie une politique d'actions par joueur $\pi_k(s), k \in 1, 2, ..., |Ag|$.
L'algorithme fait appel à deux fonctions lors de la phase d'évaluation des états :
– La fonction Search_Nash(), qui prend en paramètres la matrice de gain des joueurs et renvoie l'équilibre de Nash associé.
– La fonction Value(), qui extrait la valeur d'une action à partir de l'équilibre choisi.

L'algorithme est similaire à celui de Shapley présenté dans la section 2. Cependant, il y a quelques différences :
– Il faut construire dorénavant une matrice multi-dimensionnelle (un hypercube $|Ag|$-dimensionnel, où chaque dimension représente les décisions possibles pour le joueur associé).
– Une matrice à somme générale implique la possibilité d'existence de plusieurs équilibres pour un même état.

1. Comment faire le choix ? Une première méthode consiste à utiliser la notion de Pareto optimalité. Le jeu est dans un état Pareto optimal s'il existe un état dans lequel aucun joueur ne peut augmenter son gain sans détériorer le gain d'au moins un autre. Cependant, le système peut avoir plusieurs situations Pareto optimales, avec des performances sociales différentes.

2. Comment peut-on comparer des équilibres multiples entre-eux ? Une mesure de qualité de performance dans le jeu doit être définie. Compte tenu de la description d'un tel système, l'évaluation des équilibres paraît pour l'instant difficile. L'algorithme se contente de choisir le premier équilibre de Nash qui maximise les gains de tous joueurs $(R_1 + R_2 + ... + R_{|Ag|})$.

– A ces difficultés, s'ajoute le fait qu'il est possible de ne pas y avoir du tout d'équilibre en stratégies pures. Notre solution consiste à chercher plutôt dans ce cas un équilibre approximatif. Dans les expériences, que nous avons menées, dans le pire des cas ces états ne représentaient pas plus de 0,01% de l'ensemble total de tous les états, donc l'approximation à ce niveau ne peut pas biaiser le résultat général des expérimentations.

Au niveau de la sortie, la politique obtenue est supposée être une politique stationnaire associant à chaque état un équilibre de Nash.

Algorithme 2 Algorithme pour JS à somme générale

ENTRÉES: Un jeu stochastique JS, $\gamma \in [0, 1]$, $\epsilon \geq 0$
pour tout $s \in S$ **faire**
  $V_0(s) = 0$
fin pour
$t \leftarrow 0$
répéter
  $t \leftarrow t + 1$
  pour tout $s \in S$,$a \in \{A_1 \times A_2 \times ... \times A_{|Ag|}\}$ **faire**
    pour tout $k \in 1..|Ag|$ **faire**
      $v^k = R_k(s, a) + \gamma \sum_{s' \in S} T(s, a, s')\, V_{t-1}(s')$
    fin pour
    // Remplir la matrice M^s
    (1) $M^s = \{M^s(a) = (v^1, v^2, ..., v^{|Ag|})\}$
    // Chercher un équilibre
    (2) $V_t(s) = Value(Search\_Nash(M^s))$
  fin pour
jusqu'à $max_{s \in S}|V_t(s) - V_{t-1}(s)| < \epsilon$

Complexité.
Déterminer la complexité des équilibres de Nash a suscité beaucoup d'intérêt ces dernières années. Il a été démontré que trouver un équilibre de Nash, en stratégies mixtes, est un problème PPAD-complet [10], y compris avec seulement deux agents [7, 8].
Alors que les stratégies pures sont conceptuellement plus simples que les stratégies mixtes, les problèmes de calcul associés ne semblent pas être pour autant plus faciles. En effet, 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 [9, 12, 15].
Par conséquent, le temps d'exécution de l'algorithme 2 n'est pas polynomial en la taille de la matrice du jeu d'état (en raison du fait que la fonction utilisée pour calculer un équilibre de Nash est elle-même non polynomiale). De plus, les tailles des espaces d'état et d'action jointe sont exponentielles en le nombre d'agents. Globalement, nous pensons que le temps d'exécution de notre algorithme est aussi exponentiel.

Quant à la complexité spatiale, elle dépend principalement de la matrice qui sera générée chaque fois que l'état correspondant est évalué :
– La taille de la matrice est statique, elle est toujours de $|A_1| \times |A_2| \times ... \times |A_{|Ag|}|$;
– Chaque entrée dans la matrice est composée de $|Ag|$ valeurs.

3.2 Exemple : Le rangement d'objets

Malgré leur simplicité, les jeux de grille possèdent tous les éléments clés des jeux dynamiques : états, actions, transitions et des récompenses immédiates et à long terme. Ainsi, l'exemple que nous avons choisi est similaire à des exemples issus de la littérature, tels que le "Two-Player Grid Soccer" de Littman [20] ou le "Two-Player Coordination Problem" de Hu et Wellman [18], il s'agit du cas de rangement d'objets. Il comprend : des agents, des objets et une boîte de rangement, voir figure 1. De toute évidence, l'objectif est de ranger les objets dans la boîte en un nombre minimal d'étapes. Les agents ne se partagent au préalable aucun plan et lorsqu'ils doivent prendre des décisions, ils doivent surtout éviter tout conflit (occuper la même place, porter le même objet, ...). L'exemple montre l'intérêt d'illustrer comment ce type de problème, où des entités se partagent un même environnement et doivent remplir une mission, peut être modélisé.

A la différence de nos travaux précédents [16], nous allons nous intéresser à des cas plus généraux, c'est-à-dire à un système incluant trois, voire quatre agents.

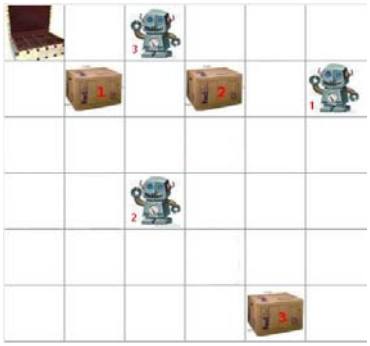

FIGURE 1 – Un exemple de scénario.

3.3 Modélisation de l'application

Rappelons que le modèle de base est :

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

Nous avons choisi de limiter le nombre d'agents à trois, essentiellement pour des problèmes de temps de calcul. L'application présentée précédemment peut se caractériser par :
– Ag : trois joueurs (ou agents).
– A_i : Chaque joueur possède un ensemble de cinq actions : (H)aut, (B)as, (G)auche, (D)roite, (N)e rien faire.
– R_i : La fonction de récompense a été définie de la même façon pour tous les joueurs ($R_1 = R_2 = R_3$), avec les valeurs arbitraires suivantes :
  – + 100 : Lorsque tous les objets seront rangés.
  – - 1000 : Si les joueurs occupent la même case ou si l'un porte un objet et essaie d'en porter un autre.
– S : Cet ensemble représente l'ensemble des "tuples-états", un tuple étant constitué à partir :
  1. Des positions géographiques dans la grille des agents et des objets.
  2. État interne de chaque agent (booléen : porte ou non un objet).
  3. État interne de chaque objet (booléen : déposé ou emporté).
– T : C'est la fonction classique de la robotique, représentant la réponse de l'environnement aux actions exécutées. Elle sera définie suivant les cas :
  – Si un agent entre sur une case contenant un objet alors il doit forcément transporter cet objet.
  – Un agent ne peut pas porter deux objets à la fois.
  – Si une action amène un objet dans la boîte, alors retirer cet objet du monde et marquer l'agent qui le transportait comme vide.

La section suivante se propose de valider notre algorithme sur l'exemple présenté.

4 Validation

Dans cette section, nous présentons les résultats obtenus en appliquant le modèle introduit de la section 2 à l'exemple de la section 3. Le simulateur a été implémenté en langage JAVA et les expériences ont été faites sur une machine, Quad core 2.8 GHZ et 4 Go de mémoire.
Notre validation porte successivement sur les temps de calcul-besoins en espace mémoire, sur le respect ou le non-respect des politiques jointes des agents.

4.1 Evaluation expérimentale en temps de calcul et besoin mémoire

Malgré la simplicité de l'exemple, nous pouvons constater que la taille de l'espace d'états n'est pas pour autant négligeable. La complexité est de : $[\ (2\ n^2)^A\ *\ 2^O\ ]$ où A représente le nombre d'agents et O le nombre d'objets. La figure 2, montre l'évolution de $|S|$ en fonction de la taille de l'environnement. Par exemple, pour une dimension de 9×9 et 3 objets à transporter, le nombre total d'états est de $34.012.224$;
Afin d'éviter toute évaluation subjective, nous comparons la solution obtenue avec celle issue d'un contrôle centralisé (un MMDP), considérée ici en tant que borne optimale pour le problème relaxé (environnement complètement observable). Ainsi, nous avons remarqué que la construction des matrices et la recherche d'équilibre augmentent considérablement le temps de calcul et la taille de mémoire vive nécessaire, voir tableau 2. Plus précisément, nous observons que le temps de calcul mesuré pour notre algorithme

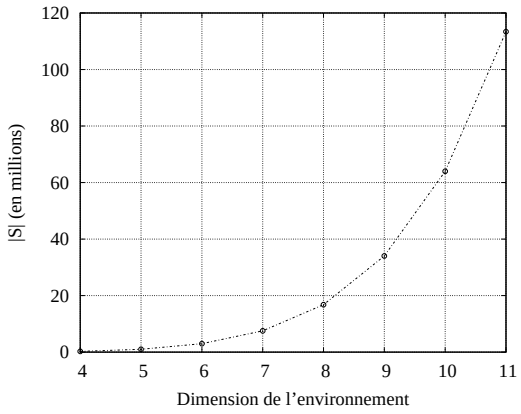

FIGURE 2 – Evolution de l'espace d'états en fonction de la dimension de l'environnement.

est en moyenne 1.5 fois supérieur à celui de l'approche centralisée. De même, le besoin spatial est de l'ordre de 4 à 5 fois supérieur à l'approche de référence.

TABLE 2 – Mémoire utilisée (Mo) et temps de calcul (sec) pour différents nombres d'objets, différentes tailles de l'environnement et 3 agents.

	Grille	Mémoire		Temps	
		MMDP	JS	MMDP	JS
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

4.2 Evaluation sur les politiques jointes des agents

Dans un premier temps, nous verrons ce que peut être une politique jointe (issue de l'algorithme 2). Quelques copies d'écrans, voir figure 3, montrent le comportement de deux agents (par souci de clarté) face aux différentes situations.

Les étapes sont les suivantes :

– Etape 1 : l'agent 1 doit aller à gauche (prendre ainsi l'objet 2) et l'agent 2 doit aller en haut (éviter une collision).

– Etape 2 : l'agent 1 n'a plus qu'à ranger l'objet transporté dans la boîte en allant vers le bas (monde torique).

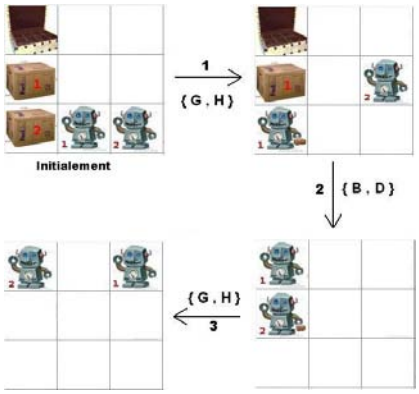

FIGURE 3 – Un exemple de politique d'actions.

L'agent 2, ne risquant plus une collision, décide d'aller vers l'objet restant.

– Etape 3 : afin de permettre à l'agent 2 de déposer l'objet qu'il transporte, l'agent 1 lui cède la voie.

Le jeu commence à partir d'un état initial choisi aléatoirement. Après avoir observé l'état actuel, les agents sélectionnent leurs actions simultanément. Ils observent ensuite le nouvel état, ainsi que l'action prise par chacun. Lorsque tous les objets seront rangés dans la boîte, le jeu s'arrête. Pour cette application, aucune hypothèse contraignante n'a été imposée. Notamment, les situations conflictuelles de gestion spatiale ne sont pas interdites. Par exemple, il est tout à fait possible qu'un agent portant un objet, décide de se déplacer vers une case adjacente contenant un autre objet. Nous avons constaté également que le fait de différencier les fonctions de récompenses n'eut pas vraiment d'impact sur la qualité de la solution.

Concernant les expériences nous avons choisi de les réaliser sur 150 politiques [4]. Pour chacune de ces politiques, 200.000 tests ont été effectués [5], nous obtenons ainsi 30 millions de tests par expérimentation. L'élément principal de comparaison est la moyenne des nombres d'étapes nécessaires pour déplacer tous les objets (fixés arbitrairement à trois). Malgré des temps de calcul et des besoins en espace mémoire supérieurs à l'approche de référence, nous observons des résultats qui sont proches de ceux de la solution MMDP - Cf. figure 4. Ceci nous confortent dans l'idée que les jeux stochastiques à somme générale permettent bien de faire se coordonner des agents.

Le passage à un nombre d'agents supérieur à trois, est possible mais seulement en utilisant des équilibre de Nash dits à ϵ près [11]. Cependant, nous sommes confrontés à une difficulté majeure, qui est la complexité tant au niveau algorithmique que spatiale.

4. Les positions des objets sont choisies aléatoirement suivant une distribution uniforme sur l'intervalle $[0,n^2[$.

5. Les positions des agents sont choisies aléatoirement suivant une distribution uniforme sur l'intervalle $[0,n^2[$.

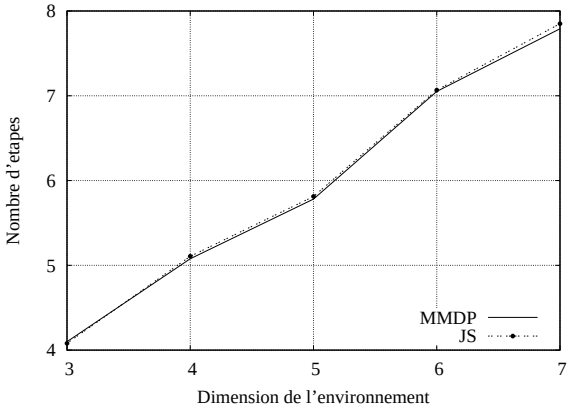

FIGURE 4 – Comparaison entre jeux stochastiques et MMDP suivant le nombre d'étapes nécessaires pour atteindre l'objectif.

4.3 Evaluation sur le non respect de la politique jointe

Rappelons que les résultats donnés ci-dessus, supposent une hypothèse forte, i.e. le respect obligatoire du plan d'actions durant toute la phase d'exécution par les agents. Des simulations complémentaires ont donc été effectuées sur une hypothèse moins contraignante, impliquant que l'un des agents adopte un comportement aléatoire (non respect de la politique jointe). Nous avons mesuré le comportement global dans des situations à deux agents en terme de nombre d'étapes et de conflits. Sur ces situations (voir les figures 5 et 6), nous constatons que le modèle des jeux stochastiques présente un nombre de conflits et un nombre d'étapes inférieurs à l'approche centralisée.

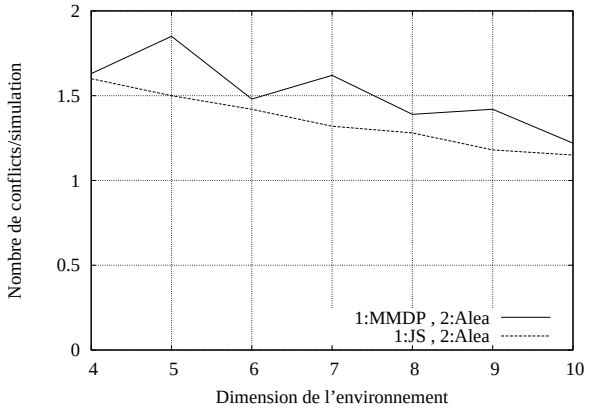

FIGURE 5 – Comparaison suivant le nombre de conflits entre jeux stochastiques et MMDP lorsque l'un des agents se comporte de façon aléatoire.

Les notations des légendes expriment que :
– Le premier agent exécute sa part de la politique jointe, calculée respectivement à partir des modèles MMDP (*1 :MMDP*) et jeux stochastiques (*1 :JS*).
– Le deuxième agent adopte un comportement aléatoire *(2 :Alea).*

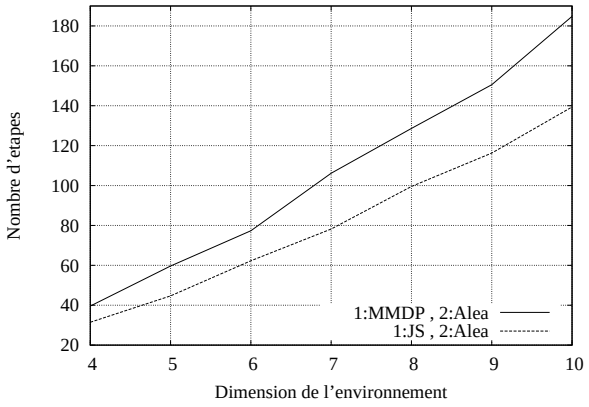

FIGURE 6 – Comparaison suivant le nombre d'étapes entre jeux stochastiques et MMDP lorsque l'un des agents se comporte de façon aléatoire.

Ces premiers résultats nous permettent de considérer les jeux stochastiques comme un mécanisme de coordination multi-agents. En effet, malgré les fortes hypothèses sur la rationalité des autres agents, le modèle donne des bons résultats, y compris avec des comportements non coopératifs. A l'opposé du modèle étudié, l'approche centralisée semble s'adapter plus difficilement à ces contraintes.

5 Conclusion

Le but de ce travail exploratoire était d'étudier le modèle des jeux stochastiques pour faire coordonner un système multi-agents. Nous avons mis en évidence que ce problème était considéré comme un problème difficile en terme de résolution. Pour appréhender cette difficulté, nous avons évalué et validé ce modèle sur un exemple illustratif (déplacement d'objets par trois agents). Nous avons remarqué que ce modèle, combiné avec la notion d'équilibre de Nash, donne des résultats équivalents (voire meilleurs) à une approche centralisée.

Nous envisageons de poursuivre ces travaux selon deux perspectives de recherche. A court-terme, ces résultats doivent être étendus aux cas où les agents possèdent différents objectifs. A moyen-terme et long-terme, nous nous proposons de vérifier la robustesse de l'approche dans le cadre d'un nombre d'agents plus important.

Remerciements

Ce travail de recherche a été financé par la communauté européenne, la Délégation Régionale à la Recherche et à la Technologie, le Ministère de l'Education Nationale, de la Recherche et de la Technologie, la région Nord Pas de Calais, le Centre National de la Recherche Scientifique, et l'Agence Nationale de la Recherche : projet AMORCES.

Références

[1] R. Aras, A. Dutech, and F. Charpillet, *Cooperation in stochastic games through communication*, AAMAS '05 : Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (New York, NY, USA), ACM, 2005, pp. 1197–1198.

[2] R. Bellman, *Dynamic programming*, Princeton University Press, 1957.

[3] C. Boutilier, *Planning, learning and coordination in multiagent decision processes*, In TARK, Morgan Kaufmann, 1996, pp. 195–210.

[4] C. Boutilier, *Sequential optimality and coordination in multiagent systems*, IJCAI '99 : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1999, pp. 478–485.

[5] M. Bowling, *Multiagent learning in the presence of agents with limitations*, Ph.D. thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, May 2003, Available as technical report CMU-CS-03-118.

[6] M. Bowling and M. Veloso, *An analysis of stochastic game theory for multiagent reinforcement learning*, Tech. report, Computer Science Department, Carnegie Mellon University, 2000.

[7] X. Chen and X. Deng, *Settling the complexity of two-player nash equilibrium*, FOCS '06 : Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2006, pp. 261–272.

[8] X. Chen, X. Deng, and S.Teng, *Computing nash equilibria : Approximation and smoothed complexity*, Electronic Colloquium on Computational Complexity (ECCC) **13** (2006), no. 023.

[9] V. Conitzer and T. Sandholm, *New complexity results about nash equilibria*, Games and Economic Behavior **63** (2008), no. 2, 621–641.

[10] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou, *The complexity of computing a nash equilibrium*, STOC '06 : Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 2006, pp. 71–78.

[11] C. Daskalakis and A. Mehtaand C. Papadimitriou, *A note on approximate nash equilibria*, Theor. Comput. Sci. **410** (2009), no. 17, 1581–1588.

[12] F. Fischer, M. Holzer, and S. Katzenbeisser, *The influence of neighbourhood and choice on the complexity of finding pure nash equilibria*, Inf. Process. Lett. **99** (2006), no. 6, 239–245.

[13] D. Fudenberg and J. Tirole, *Game theory*, MIT Press, Cambridge, MA, 1991.

[14] O. Gies and B. Chaib-draa, *Apprentissage de la coordination multiagent. une méthode basée sur le q-learning par jeu adaptatif*, Revue d'Intelligence Artificielle **20** (2006), no. 2-3, 383–410 (French).

[15] G. Gottlob, G. Greco, and F. Scarcello, *Pure nash equilibria : hard and easy games*, TARK '03 : Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge (New York, NY, USA), ACM, 2003, pp. 215–230.

[16] M.A. Hamila, E. Grislin le Strugeon, R. Mandiau, and A.I. Mouaddib, *Jeux stochastiques à somme générale pour la coordination multi-agents*, Les Journées Francophones sur les Systèmes Multi-Agents (JFSMA), 2009 (French).

[17] R.A. Howard, *Dynamic programming and markov processes*, Cambridge, Mass. : M.I.T. Press, 1960.

[18] J. Hu and M.P. Wellman, *Nash q–learning for general–sum stochastic games*, Journal of Machine Learning Research **4** (2003), 1039–1069.

[19] M. Kearns, Y. Mansour, and S. Singh, *Fast planning in stochastic games*, In Proceedings UAI-2000, Morgan Kaufmann, 2000, pp. 309–316.

[20] M.L. Littman, *Markov games as a framework for multi-agent reinforcement learning*, In Proceedings of the Eleventh International Conference on Machine Learning, Morgan Kaufmann, 1994, pp. 157–163.

[21] J. Von Neumann and O. Morgenstern, *Theory of games and economic behavior*, Princeton University Press, Princeton, 1944, Second edition in 1947, third in 1954.

[22] M. J. Osborne and A. Rubinstein, *A course in game theory*, The MIT Press, July 1994.

[23] M.L. Puterman, *Markov decision processes : Discrete stochastic dynamic programming*, Wiley-Interscience, 2005.

[24] L.S. Shapley, *Stochastic games*, Proceedings of the National Academy of Sciences USA (1953), 1095–1100.

[25] Y. Shoham, R. Powers, and T. Grenager, *Multi-agent reinforcement learning : a critical survey*, Tech. report, Stanford University, 2003.

[26] O. Sigaud and O. Buffet, *Processus décisionnels de Markov en intelligence artificielle*, IC2 - informatique et systèmes d'information, vol. 1 - principes généraux et applications, Lavoisier - Hermes Science Publications, 2008 (French).

[27] S. Zilberstein, R. Washington, D.S. Bernstein, and A.I. Mouaddib, *Decision-theoretic control of planetary rovers*, Revised Papers from the International Seminar on Advances in Plan-Based Control of Robotic Agents (London, UK), Springer-Verlag, 2002, pp. 270–289.

[28] M. Zinkevich, A.R. Greenwald, and M.L. Littman, *Cyclic equilibria in markov games*, NIPS, 2005.