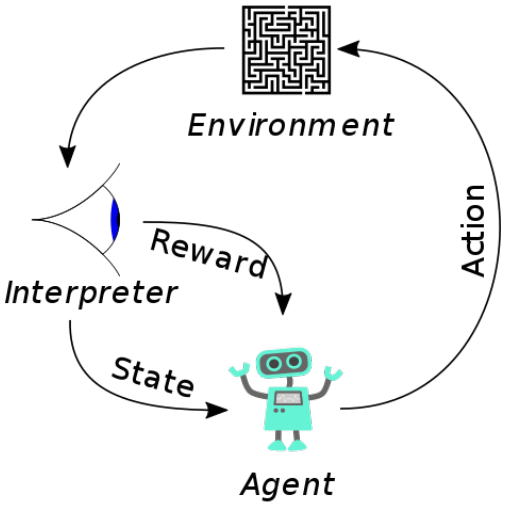
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE

ÉTUDE BIBLIOGRAPHIQUE

Apprentissage multi-agent par renforcement



Rand ASSWAD
Génie Mathématique

Table des matières

Chapitre 1

Introduction

1.1 Systèmes Multi-Agents

Un système multi-agents (SMA) est un système automatisé composé de plusieurs agents interagissant entre eux et avec leur environnement. Nous avons étudié en cours de *systèmes multi-agents* la conception d'un modèle de jeu *Capture The Flag* à l'aide de l'environnement *Jason*, nous avons aussi étudié la simulation d'un modèle de propagation d'épidémie dans un village à l'aide de la plateforme *GAMA*.

Néanmoins, les systèmes multi-agents peuvent répondre à des problèmes diverses qui sont mieux abordés dans un environnement de SMA qu'avec d'autres alternatifs. Les systèmes multi-agents apparaissent souvent dans le contexte du domaine d'apprentissage par renforcement. Ce qui est le domaine d'étude de l'article de **Littman** « Markov games as a framework for multi-agent reinforcement learning » publié en 1994.

Dans un premier temps, nous allons introduire l'apprentissage par renforcement et les systèmes multi-agents dans ce contexte. Nous allons ensuite étudier le papier de Littman et le comparer avec d'autres œuvres répondant au même problématique.

1.2 Apprentissage par renforcement

On définit d'abord le formalisme de l'apprentissage par renforcement et le procéssus de décision markovien.

Dans le domaine de l'apprentissage par renforcement (Reinforcement Learning, **RL**), un agent interagit séquentiellement avec son environnement.

L'environnement de l'agent est caractérisé par un ensemble d'états et d'actions possibles à prendre dans chaque état, avec une fonction de transition caractérisant la probabilité de passer d'un état à l'autre par une action donné, et d'une récompense pour chaque action prise à chaque état.

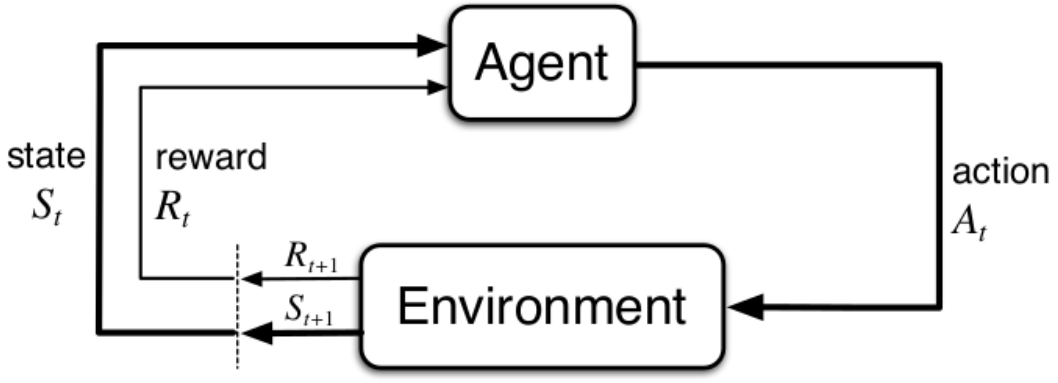


FIGURE 1.1 – Le modèle d'apprentissage d'un agent par renforcement

L'apprentissage par renforcement consiste à développer une stratégie optimale pour l'agent dans son environnement à partir d'expériences. La stratégie développée qu'on appelle « politique » (policy) permet donc de décider l'action « optimale » pour chaque état.

On définit formellement les éléments du modèle RL :

- S - l'espace d'états de l'agent
- A - l'espace d'actions de l'agent
- $T : S \times A \times S \to \mathbb{R}$ - la fonction de transition dans l'environnement, $T(s, a, s')$ représente la probabilité de passer de l'état s à s' par l'action a.

$$T(s, a, s') = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$$

- $R : S \times A$ - la fonction de récompense, $R(s, a)$ représente la récompense que l'agent obtient en faisant l'action a à l'état s.
- $\pi : S \to A$ (déterministe) ou $\pi : S \times A \to [0, 1]$ (stochastique) [1]- la politique d'action de l'agent.

On peut remarquer le fait que la fonction de transition ne dépend que du pair (s, a) à l'instant précédent, d'où le caractère markovien du modèle RL.

Il existe plusieurs méthodes pour déterminer la politique de l'agent. On s'intéressera au processus de décision markovien.

1.3 Processus de décision markovien

Un processus de décision markovien (Markov Decision Process, **MDP**) est un modèle stochastique permettant de définir une politique stochastique de l'agent.

Une fois la politique déterminée, la fonction de transition se réduit à

$$T(s, s') = \mathbb{P}(s_{t+1} = s' | s_t = s)$$

résultant d'un agent dont le comportement est une chaîne de Markov.

Le principe de MDP pour déterminer π consiste à maximiser les récompenses dans le futur. Néanmoins, la fonction de transition est de caractère stochastique, les futurs récompenses le sont également. [Foerster, 2018]

On introduit le facteur de dévaluation (amortissement) au cours du temps $\gamma \in [0, 1]$. Cette valeur permet d'accentuer la récompense dans une certaine période du temps ; si cette valeur est proche de 0 l'agent cherche son gain immédiat, si elle est proche de 1 l'agent cherche son gain dans le futur lointain.

On considère alors *l'espérance de la somme cumulative amortie*

$$\mathbb{E}\left\{\sum_{j \geq 0} \gamma^j r_{t+j}\right\}$$

où $r_t = R(s_t, \pi(s_t))$ est la récompense a obtenue après avoir effectué l'action déterminée par la politique à l'instant t.

Par conséquent, grâce à la propriété markovienne du modèle, la politique optimale pourra être définie en fonction de l'état seulement.

1.4 Jeux stochastiques

Le modèle décrit considère un seul agent interagissant avec son environnement [figure 1.1]. En revanche, un agent vis presque toujours avec d'autres agents. Il doit interagir avec ces agents afin de réaliser ses objectifs. [Littman, 1994]

Le théorie des jeux (von Neumann et Morgenstern) a été conçue afin de répondre aux problèmes de systèmes multi-agents. En 1953, Llyod Shapley a définit le modèle des jeux stochastiques (*Markov Games* ou *Stochastic*

1. on note aussi $\pi : S \to \mathcal{PD}(A)$ où $\mathcal{PD}(A)$ est l'espace de distributions de probabilités sur l'ensemble des actions A

Games), étant le premier modèle d'un environnement dynamique [Solan and Vieille, 2015]. Le jeu avance par étapes d'un état à l'autre suivant des probabilités de transition contrôlées par les joueurs. [Shapley, 1953]

En effet, les jeux stochastiques étendent la théorie des jeux aux environnements MDP-semblables. [Littman, 1994]

La prémisse du papier de Littman est d'étudier les conséquences d'implémenter un agent par un jeu stochastique *à la place* d'un processuss de décision markovien dans le cas d'un jeu à deux de somme nulle (zero-sum game).

1.5 Apprentissage multi-agent par renforcement

Un jeu de Markov peut être vue comme un cas général d'un MDP. En effet, on redéfinit l'espace d'actions $\mathbf{A} = A_1 \times \ldots \times A_n$ pour chaque agent de l'environnement où $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbf{A}$ avec $a_i \in A_i$ l'action de l'agent i. Chaque agent a également sa fonction de récompense associée

$$R_i : S \times \mathbf{A} \to \mathbb{R}$$

et cherche à maximiser l'espérance de sa somme cumulative amortie

$$\mathbb{E}\left\{\sum_{j\geq 0} \gamma^j r_{i,t+j}\right\}$$

où $r_{i,t} = R_i(s_t, \boldsymbol{\pi}(s_t))$ est la récompense que l'agent i a obtenue après avoir effectué l'action déterminée par la politique à l'instant t.

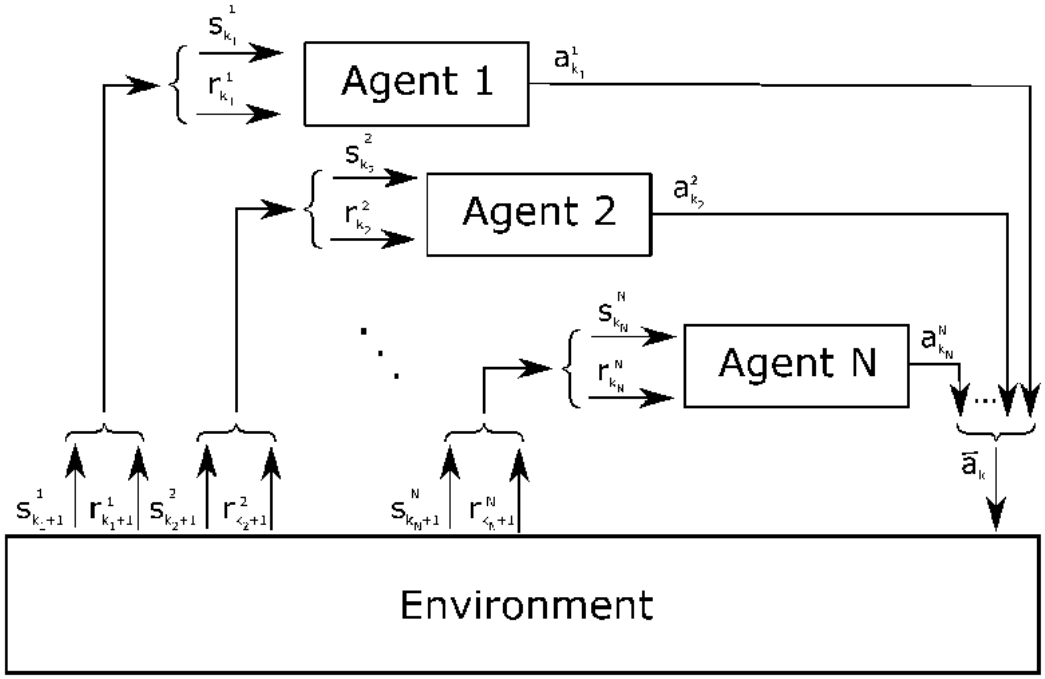


FIGURE 1.2 – Le modèle d'apprentissage multi-agent par renforcement

Chapitre 2

Contexte de l'article étudié

2.1 Le système étudié

Dans l'article étudié, on considère le cas de $n = 2$. On notera donc $\mathbf{A} = A \times O$ où A est l'espace d'actions de l'agent et O est l'espace d'action de l'adversaire.

Comme il s'agit d'un jeu à somme nulle avec seulement deux agents dont les objectifs sont opposés, nous pouvons unifier les fonctions de récompenses en définissant $R : S \times A \times O$ telle que

— $R(s, a, o) = R_a(s, (a, o))$
— $R(s, o, a) = R_o(s, (a, o))$

On l'interprète par la récompense de l'agent pour avoir effectué l'action $a \in A$ à l'état $s \in S$ quand son adversaire fait l'action $o \in O$.

Bien que cette simplification facilitera le calcul, elle élimine la possibilité de considérer un phénomène de coopération entre les agents. [Littman, 1994]

2.2 Politique Optimales

2.2.1 Définition des politiques optimales

Nous avons déjà établi que l'objectif de l'agent est de maximiser l'espérance de la somme des récompenses cumulative amortie. Avant de passer au calcul des politiques optimales il faut définir les types de politiques.

Non-dominée Une politique optimale est non-dominée si aucune autre politique a une espérance de sommes des récompenses supérieure à celle-ci pour tout état dans S.

Stationnaire Une politique optimale est stationnaire si elle ne dépend pas du temps, ce qui implique qu'elle ne dépend que de l'état d'avant (la propriété de Markov).

Déterministe Une politique est déterministe si elle associe à chaque état une seule action (les autres ont une probabilité nulle). Soit $\pi : S \to \mathbf{A}$.

Stochastique Une politique est stochastique si pour chaque état elle associe une distribution probabiliste d'actions. On note $\pi : S \to \mathcal{PD}(\mathbf{A})$ ou $\pi : S \times \mathbf{A} \to [0, 1]$.

Dans le cas d'un processus de décision markovien, il existe au moins une politique optimale stationnaire et déterministe.

Pour les jeux de Markov, ils existent toujours des politiques optimales dont une au moins est stationnaire. En revanche, il n'existe pas toujours une qui est déterministe. La politique optimale est potentiellement stochastique ;

elle indique la probabilité d'effectuer l'action **a** à l'état l'état s. On justifie ceci par le fait qu'une politique déterministe dans le jeu classique «chifoumi» pourra être battu systématiquement.

De même, pour un MDP il existe au moins une politique optimale non-dominée. Or, pour un jeu de Markov il n'y au *aucune* politique non dominée. Ceci peut être justifié par l'incertitude de l'action du joueur, contrairement à l'environnement qui est prévisible.

Une solution pour ce problème qui se trouve dans litterature de la théorie des jeux est d'évaluer chaque politique face à *la pire* action de son adversaire. La politique résultante est plutôt *conservatrice* ; telle stratégie finira souvent en égalité, contrairement à d'autres stratégies qui gagneront contre certains adversaires et perdront contre d'autres. Cette mesure est le principe essentiel de l'algorithme **minimax**.

2.2.2 Calcul des politiques optimales

Dans cette partie nous allons étudier les trois cas que Littman a étudié pour déterminer la politique optimale.

2.2.2.1 Jeu de matrice

Dans la théories des jeux, un *jeu de matrice* (matrix game) parfois appelé *jeu sous forme normale* (normal-form game) est un jeu pour deux joeurs défini par une matrice R de récompenses immédiates, où $R \in \mathbb{R}^{n \times n}$, avec n actions possible pour chaque agent. Le coefficient $R_{i,j}$ correspond à la récompense pour l'agent en choisissant l'action j lorsque son adversaire choisit l'action i.

		Agent		
		rock	paper	scissors
Opponent	rock	0	1	-1
	paper	-1	0	1
	scissors	1	-1	0

FIGURE 2.1 – Tableau du jeu chifoumi

La politique optimale de l'agent est **stochastique**, la politique est donc de la forme $\pi : S \to \mathcal{PD}(\mathbf{A})$. On cherche à obtenir π_a pour tout action a possible pour l'agent. En effet, on voudrais maximiser la récompense minimale de l'agent. C'est-à-dire, de trouver la politique permettant d'obtenir la meilleure récompense pour l'agent dans le cas où l'adversaire effectue *la pire* action possible (point de vu du premier agent).

Littman propose de trouver une politique avec une récompense attendue d'au moins V pour toute action possible de l'adversaire. Le problème se réécrit sous la forme d'un **problème d'optimisation linéaire**

$$V = \max_{\pi \in \mathcal{PD}(\mathbf{A})} \min_{o \in O} \sum_{a \in A} R_{o,a} \pi_a \tag{2.1}$$

où $R_{o,a}\pi_a$ est la récompense attendue pour la politique π contre l'action o de l'adversaire.

Cela se traduit simplement dans le cas du jeu Chifoumi à partir du tableau 2.1

$$\begin{array}{rcrcrcll} & & \pi_{\text{feuille}} & - & \pi_{\text{ciseaux}} & \geq & V & \text{(vs. pierre)} \\ - & \pi_{\text{pierre}} & & + & \pi_{\text{ciseaux}} & \geq & V & \text{(vs. feuille)} \\ & \pi_{\text{pierre}} & - \ \pi_{\text{feuille}} & & & \geq & V & \text{(vs. ciseaux)} \\ & \pi_{\text{pierre}} & + \ \pi_{\text{feuille}} & + & \pi_{\text{ciseaux}} & = & 1 & \end{array}$$

A l'aide d'un programme linéaire on peut trouver que $V = 0$ et $\pi = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

2.2.2.2 Procéssus de décision markovien

Dans le cas d'un MDP, Littman propose de trouver la politique optimale par la méthode de l'itération de valeurs **(value iteration)** [Bertsekas, 1987].

Dans cette méthode on introduit deux fonctions :
- $V(s)$: la **valeur** de chaque état, définie par la somme des récompenses amortis obtenue suivant la *politique optimale* à partir de l'état $s \in S$.
- $Q(s,a)$: la **qualité** de l'action a à l'état s, définie par la somme des récompenses amortis obtenus suivant la *politique optimale* après avoir effectué l'action a.

Ces deux fonctions sont récursivement liées, par les relations suivantes pour tout $a \in A$ et $s \in S$.

$$V(s) = \max_{a \in A} Q(s,a) \tag{2.2}$$

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s') \tag{2.3}$$

Ces équations sont aussi vu dans la littérature sous la forme implicite, connue par **l'équation d'optimalité de Bellman**.

$$V(s) = \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s') \right)$$

Dans la méthode *value iteration* on utilise les équations sous la forme récursive, ce qui permet de trouver itérativement valeurs de $V(s)$ et $Q(s,a)$ pour tout $s \in S$ et $a \in A$, en traitant l'opérateur d'*égalité* d'un opérateur d'*affectation*, à partir d'une estimation initiale. Les valeurs de V et Q convergent vers leurs *vraie valeurs* [Bertsekas, 1987].

Etant donné $Q(s,a)$, l'agent adopte une stratégie « gloutonne » (greedy) en choisissant toujours l'action qui maximise la valeur de Q. Comme la valeur de Q est une bonne approximation des futurs récompenses, la politique trouvée est en effet optimale.

2.2.2.3 Jeu markovien

Dans le cas d'un jeu markovien, il est possible de d'appliquer la même stratégie en adaptant la définition des fonctions V et Q pour un système multi-agents. La qualité $Q(s,a,o)$ représente alors la somme des récompenses amortis obtenus suivant la politique optimale quand l'agent effectue l'action a et son adversaire effectue l'action o. Similairement au cas d'un jeu de matrice, on voudrais maximiser la récompense future (estimée par Q) minimale, pour toutes les actions possibles de l'agent face à la pire action de son adversaire.

$$V(s) = \max_{\pi \in \mathcal{PD}(\mathbf{A})} \min_{o \in O} \sum_{a \in A} Q(s,a,o)\pi_a \tag{2.4}$$

$$Q(s,a,o) = R(s,a,o) + \gamma \sum_{s' \in S} T(s,a,o,s')V(s') \tag{2.5}$$

Par le même procédé vu dans le cas d'un MDP, les valeurs de V et Q se calculent itérativement convergeant vers leurs vraies valeurs. L'itération de calcul de $V(s)$ se fait par un programme linéaire comme dans le cas d'un jeu de matrice. [Littman, 1994]

Littman précise que pour les jeux à tours alternés l'expression de V se simplifie car il existe une politique déterministe, épargnant ainsi le calcul d'un programme linéaire.

$$V(s) = \max_{a \in A} \min_{o \in O} Q(s,a,o)$$

2.2.3 Apprentissage des politiques optimales

2.2.3.1 Value iteration

Cette algorithme est une méthode classique pour l'apprentissage des politiques optimales dans le cas des MDP. Comme nous l'avons vu [section 2.2.2.2], les équations de Bellman sont évaluées pour tout $s \in S$ jusqu'à ce que V et Q convergent vers leurs vraies valeurs.

Algorithme 1 : Value Iteration

Initialiser $V(s)$ arbitrairement
répéter
  pour chaque $s \in S$ **faire**
    $V(s) \leftarrow \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V(s') \right)$
  fin
jusqu'à *la différence entre les* $V(s)$ *est suffisamment petite* $\forall s \in S$

2.2.3.2 Q-learning

Watkins [Watkins, 1989] propose une modification à la méthode précédente qui consiste à mettre à jour les valeurs de manière asynchrone. Il s'agit d'*effectuer* l'action a passant de l'état s à s' recevant une récompense r. Cet évènement arrive avec une probabilité de $T(s,a,s')$, cette probabilité est donc implicitement prise en compte lorsque la valeur de Q est mise à jour suivant la formule

$$Q(s,a) \leftarrow r + \gamma V(s')$$

Cette méthode d'apprentissage converge, supposant que chaque action est essayée dans chaque état un nombre infini de fois, les nouvelles estimations sont intégrées avec les anciennes avec une moyenne pondérée exponentiellement.

Algorithme 2 : Q-learning

Initialiser $Q(s,a)$ arbitrairement
Initialiser s
boucle
  $a \leftarrow \underset{a \in A}{\operatorname{argmax}}\, Q(s,a)$
  Effectuer l'action a, recevoir une récompense r et arriver à l'état s'
  $V(s') \leftarrow \max_{a' \in A} Q(s',a')$
  $Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \left(r + \gamma V(s') \right)$
  $s \leftarrow s'$
fin
$\forall s \in S, \pi(s) \leftarrow \underset{a \in A}{\operatorname{argmax}}\, Q(s,a)$

2.2.3.3 Minimax-Q

Littman propose d'appliquer la méthode Q-learning dans le cas d'un jeu markovien. L'algorithme est essentiellement Q-learning en remplaçant le *max* dans l'évaluation de $V(s')$ par un *minimax*.

On appelle α le taux d'apprentissage, il détermine le poids de nouvelles informations apprise à chaque itération. Un paramètre de décroissance (decay) δ est introduit afin de diminuer la valeur de α au cours du temps après l'avoir initialisée à 1. Une probabilité d'exploration p_{explor} est aussi introduit, si la fonction aléatoire renvoie vraie une action est choisie de manière aléatoire uniformément. Ceci permet de s'assurer que l'agent explore l'espace d'état

entièrement. [Littman, 1994]

Algorithme 3 : minimax-Q

$V(s) \leftarrow 1, \forall s \in S$
$Q(s,a,o) \leftarrow 1, \forall s \in S, a \in A, o \in O$
$\pi(s,a) \leftarrow \frac{1}{|A|}, \forall s \in S, a \in A$
$\alpha \leftarrow 1$
Initialiser s
boucle
  si *avec une probabilité* p_{explor} **alors**
    $a \leftarrow$ tirée aléatoirement suivant une loi uniforme sur A
  sinon
    $a \leftarrow$ tirée aléatoirement avec une probabilité $\pi(s,a)$
  fin
  Effectuer l'action a, recevoir une récompense r, arriver à l'état s' et l'adversiare effectue l'action o
  $Q(s,a,o) \leftarrow (1-\alpha)Q(s,a,o) + \alpha\left(r + \gamma V(s')\right)$
  $\pi(s) \leftarrow \underset{\pi' \in \mathcal{PD}(A)}{\text{argmax}} \min_{o' \in O} \sum_{a' \in A} Q(s,a',o')\pi'(s,a')$ (résoudre à l'aide d'un programme linéaire)
  $V(s) \leftarrow \min_{o' \in O} \sum_{a' \in A} Q(s,a',o')\pi(s,a')$
  $\alpha \leftarrow \alpha \cdot \delta$
  $s \leftarrow s'$
fin
$\forall s \in S, \pi(s) \leftarrow \underset{a \in A}{\text{argmax}}\, Q(s,a)$

2.3 Exemple d'application : football

2.3.1 Définition de l'environnement

Littman étudie dans son papier un jeu de football sur une grille $2D$ de taille 4×5 [figure 2.2]. Deux joueurs opposés, A et B, choisisent à chaque tour de se déplacer vers une cellule adjacente (haut, bas, gauche, droit) ou de rester dans sa cellule.

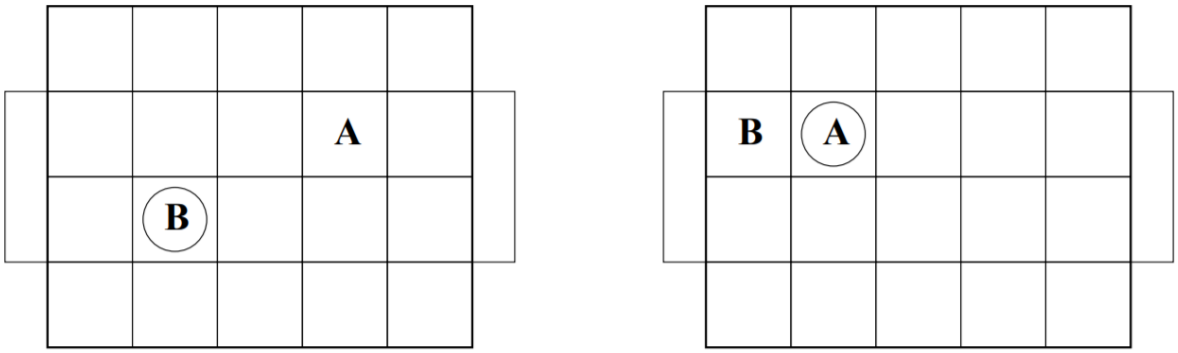


FIGURE 2.2 – (gauche) grille initiale (droit) grille intermédiaire

Le joueur possédant le ballon (représenté par un cercle) marque un point s'il se déplace vers le but de son opposant, il marque ainsi un point, ensuite la grille est remise à sa configuration initiale [figure 2.2 gauche], et la possession du ballon est tiré aléatoirement. Si un joueur essaye d'aller à une case occupé par son adversaire, la possession du ballon va au joueur stationnaire et le déplacement ne s'effectue pas.

Un but vaut un point et le facteur d'amortissement γ vaut 0.9, ce qui encourage l'agent à marquer des bût tôt.

2.3.2 L'apprentissage des agents

Quatres politiques différentes ont été apprises :
- **MR :** agent minimax-Q entraîné contre un agent aléatoire.
- **MM :** agent minimax-Q entraîné contre un agent minimax-Q.
- **QR :** agent Q-learning entraîné contre un agent aléatoire.
- **QQ :** agent Q-learning entraîné contre un agent Q-learning.

L'apprentissage des agents minimax-Q s'est fait suivant [Algorithme 3] avec une probabilité d'exploration $p_{\text{explor}} = 0.2$ et un *decay* $\delta = 10^{-2\times10^{-6}} \approx 0.9999954$ avec un million d'itérations. Comme $(\alpha_n)_n$ est une suite géométrique de raison δ avec $\alpha_0 = 1$, le taux d'apprentissage se donne par $\alpha_n = \delta^n$, à la fin de l'algorithme $\alpha_{10^6} \approx 0.01$.

L'algorithme Q-learning est expliqué pour le cas d'un MDP, Littman l'applique au modèle multi-agent en reprenant l'algorithme minimax-Q en remplaçant l'opérateur "minimax" par un "max", en gardant le même code et mêmes paramètres.

Quant à l'agent aléatoire, il s'agit d'un agent utilisant une politique fixée qui choisit les actions selon une loi uniforme sur A. Soit $\pi(s,a) = \frac{1}{A}, \forall s \in S, \forall a \in A$.

Pour les agents MM et QQ, l'adversaire d'entraînement est d'implémentation identique mais il diffère par les valeurs Q et V dû au fait qu'il est entraîné contre un adversaire différent.

2.3.3 Évaluation des agents

Littman a mis ses quatres agents sous trois tests :
- Contre un agent d'une politique aléatoire : le match est limité à cent mille tours, avec une probabilité de 0.1 à chaque tour d'arrêter le match et le déclarer une égalité.
- Contre un agent d'une politique déterministe implémentée à la main : le match suis les règles de base sans contrainte d'égalité. Cette adversaire a terminé 5600 matchs contre l'agent aléatoire et en a gagné 99.5%.
- Contre un agent *challenger* : l'adversaire est entraîné avec l'algorithme Q-learning contre chacun des quatre agents, développant une politique d'attaque spécifique à chaque agent testé.

2.3.4 Résultats

	MR		MM		QR		QQ	
	% won	games	% won	games	% won	games	% won	games
vs. random	99.3	6500	99.3	7200	99.4	11300	99.5	8600
vs. hand-built	48.1	4300	53.7	5300	26.1	14300	76.3	3300
vs. MR-challenger	35.0	4300						
vs. MM-challenger			37.5	4400				
vs. QR-challenger					0.0	5500		
vs. QQ-challenger							0.0	1200

FIGURE 2.3 – Tablaux des résulats des tests

Contre l'agent de politique aléatoire Les quatre agents ont bien testé contre cette politique. Néanmoins, l'agent QR a eu le succès le plus important parmi les quatre comme il a complété plus de matchs et en a gagné la majorité ; cela est attendu comme l'agent QR a été entraîné contre cette politique. En revanche, l'agent MR qui a été entraîné contre la même politique n'a pas eu le même succès, Littman explique ceci par le fait que l'apprentissage minimax-Q s'entraîne toujours contre les pires scénarios possibles.

Contre l'agent de politique déterministe Les agents minimax-Q ont eu des bon résultats, autour de 50% de réussite. De plus, la politique de MM a légèrement mieux réussi que MQ, ce que Littman interprète d'une preuve que la politique minimax-Q n'est pas encore arrivé à son optimum car elle devrais être insensible à son adversaire. Les résultats des agents Q-learning ont été très différents. D'une part, la politique QR a convergé vers une stratégie qui n'était pas du tout adapté à son adversaire. D'autre part, la politique QQ a eu un très grand succès, ce que Littman interprète par chance. Il explique que le succès de QQ comparé aux politiques minimax-Q est surprenant

et non-justifiable mathématiquement, et qu'en pratique l'apprentissage tend à "se bloquer" une fois un maximum local qui arrête l'apprentissage des deux agents prématurément. Malgré le fait que d'autre chercheurs ont eu un succès avec cette politique également.

Contre l'agent challenger Cette expérience permet de mesurer la pire performance de chaque politique. Les politiques apprises était bloqués alors que leurs *challenger* apprenais à leur battre. Ce qui représente les scénarios que minimax-Q a été conçu pour battre. Néanmoins, les politiques MR et MM n'ont pas donné les résultats espérés, ce qui veut dire que l'apprentissage avait besoin de plus d'itérations (qu'un million). Ceci dit, la politique MM a légèrement mieux réussi que MR. En revanche, les politiques QR et QQ n'ont pas réussi à marquer contre leurs adversaires respectifs, cela est dû au fait que Q-learning cherche des politiques déterministes, alors que chaque politique déterministe comme dans le cas du jeu Chifoumi a une défense parfaite adaptée. Alors que minimax-Q étend Q-learning afin de trouver des politiques probabilistes pour répondre à ce genre de problèmes.

2.4 Conclusion

Ce papier emploie les jeux de Markov comme framework pour les environnements multi-agent dans le contexte de l'apprentissage par renforcement. Littman propose une approche MARL pour un jeu à somme nulle entre deux joueurs, en s'appuyant sur le principe de minimax permettant de survivre les pires scénarios.

La complexité algorithmique de l'apprentissage est importante car il s'agit d'un programme linéaire dans la boucle imbriquée. Néanmoins, cela ne pose pas de problème dans le cas des jeux aux tours alternés (tic-tac-toe, dames, backgammon, etc) car l'opérateur minimax s'implémente efficacement sans programmation linéaire.

Chapitre 3

Autres Travaux

Dans ce chapitre, nous allons étudier des papiers qui permettent d'élargir sur le domaine d'apprentissage multi-agent par renforcement.

3.1 From Single-Agent to Multi-Agent Reinforcement Learning

L'article de Gonçalo Neto « From Single-Agent to Multi-Agent Reinforcement Learning : Foundational Concepts and Methods » [Neto, 2005] permet d'étudier au plus près l'apprentissage par renforcement dans le cas d'un seul agent et d'étudier les différents systèmes multi-agents dans le même cadre.

Cet article est une revue général dans le domaine de MARL, nous n'intéressons qu'aux parties non-vus et qui concernent les cas multi-agents.

3.1.1 Single-Agent Framework

Nous avons vu le modèle RL dans le cas d'un agent simple [section 1.5]. Ainsi que la définition formelle d'un procéssus de décision markovien et de politiques optimales. Néanmoins, nous allons visiter les notions d'optimalité car nous ne l'avons pas vu auparavant.

3.1.1.1 Notions d'optimalité

Le but d'un agent dans le modèle MDP est maximiser ses récompenses au cours du temps, ce qui peut être formulé dans plusieurs manières différentes.

Modèle fini simple L'agent cherche à maximiser la somme de récompenses pour les prochaines M étapes.

$$\mathbb{E}\left\{\sum_{j=0}^{M} r_{t+j}\right\}$$

Ce modèle est adopté lorsque le jeu/tâche est limité à M étapes, ce qui n'est pas toujours le cas.

Modèle infini amorti L'agent cherche à maximiser la somme de récompenses *amortis* au long-terme.

$$\mathbb{E}\left\{\sum_{j=0}^{\infty} \gamma^j r_{t+j}\right\} \quad \gamma \in [0,1]$$

Dans ce modèle, l'objectif est de privilégier les actions qui donnent des récompenses court-terme (γ petit) ou long-terme (γ grand).

Modèle infini moyen L'agent cherche à maximiser la *moyenne* de récompenses au long-terme.

$$\lim_{M\to\infty} \mathbb{E}\left\{\frac{1}{M}\sum_{j=0}^{M} r_{t+j}\right\}$$

Ce modèle ne différencie pas les récompenses court-terme ou long-terme.

Le modèle infini amorti est préféré dans les applications de MDP dans l'apprentissage par renforcement, car il permet de privilégier une période d'action et de borner la somme.

3.1.1.2 Programmation dynamique

La résolution des MDP est souvent associé avec la notion *programmation dynamique* qui a été introduit par [Bellman, 1957]. Deux méthodes classiques de programmation dynamique pour les MDP sont *value iteration* et *policy iteration*. Ceux deux méthodes se basent sur les équations d'optimalité de Bellman.

$$V(s) = \max_{a\in A} Q(s,a) \tag{3.1}$$

$$Q(s,a) = R(s,a) + \gamma \sum_{s'\in S} T(s,a,s')V(s') \tag{3.2}$$

La méthode de *policy itération* cherche la politique optimale dans un MDP en manipulant directement la politique plutôt que de la trouver à travers les valeurs d'états $V(s)$.

Néanmoins, nous n'allons pas aller plus loin dans l'étude des méthodes des systèmes d'un seul agent. Ayant ces bases, nous pouvons étudier en détails le modèle multi-agent.

3.1.2 Multi-Agent Framework

Comme nous l'avons vu avec Littman, l'étude des modèles multi-agents nous dirige vers la théorie des jeux. On s'intéresse particulièrement aux *jeux stochastiques* (markovien) et aux *jeux de matrice* (matrix games).

Neto utilise dans ce contexte le mot *stratégie* à la place de *politique* pour éviter la confusion avec le cas d'un agent simple, en remplaçant la notation $\pi(s)$ par $\sigma(s)$. Néanmoins, nous allons garder la même notation avec conformément à l'article de Littman et la majorité de la littérature.

3.1.2.1 Matrix Games

La notion de *jeux de matrice* (Matrix Games, **MG**), parfois appelés *jeux stratégiques* est un framework simple pour le cas des jeux à plusieurs joueurs avec un seul espace d'état et ses récompenses associées, ce qui permet de les représenter sous forme de matrice.

Un exemple classique que nous avons vu avec Littman est le jeu « Chifoumi ». En effet, les deux agents partagent le même espace d'état S.

	Rock	Paper	Scissors
Rock	(0 , 0)	(-1 , 1)	(1 , -1)
Paper	(1 , -1)	(0 , 0)	(-1 , 1)
Scissors	(-1 , 1)	(1 , -1)	(0 , 0)

FIGURE 3.1 – La matrice du jeu chifoumi

Nous rappelons qu'une politique $\pi_i \in \mathcal{PD}(A_i)$ définit l'action que l'agent i effectue dans le jeu. Nous introduisons la notion de **joint policy** (*politique commune*) qui représente une collection de n politiques (une pour chaque agent), elle s'écrit $\pi = (\pi_i, \pi_{-i})$ où π_{-i} est la politique commune de tous les agents sauf l'agent i.

Nous appellerons aussi *pure policy* les politiques déterministes et *mixed policy* pour les politiques stochastiques.

Pour chaque politique commune, il existe une récompense associée pour chaque joueur $R_i(\pi)$ qui pourra être définie en terme des récompenses des actions.

$$R_i(\pi) = \sum_{a \in A} R_i(a) \prod_{j=1}^{n} \pi_i(a_i)$$

Notions d'optimalité

Dans un premier temps nous allons définir quelques notions importantes permettant d'étudier l'optimalité d'une politique plus rigoureusement.

Best-response policy Une politique individuelle est dîte *best-response policy* si pour une politique commune de tous les autres agents π_{-i} elle réalise la récompense la plus élevée possible. On note

$$\pi_i \in \mathcal{BR}_i(\pi_{-i})$$

où $\mathcal{BR}_i(\pi_{-i})$ représente l'ensemble des meilleurs politiques (best-response policies) pour l'agent i face à la politique commune π_{-i}. Soit mathématiquement

$$\pi_i^* \in \mathcal{BR}_i(\pi_{-i}) \Leftrightarrow \forall \pi_i \in \mathcal{PD}(A_i), R_i((\pi_i^*, \pi_{-i})) \geq R_i((\pi_i, \pi_{-i}))$$

Équilibre de Nash L'équilibre de Nash est une collection de politiques, une pour chaque agent, où chacune de ces politique est une best-response policy, ce qui signifie qu'aucun de ces agents à intérêt de changer de stratégie si les autres gardent les leurs.

$$\forall i \in \{1, \ldots, n\}, \pi_i \in \mathcal{BR}_i(\pi_{-i})$$

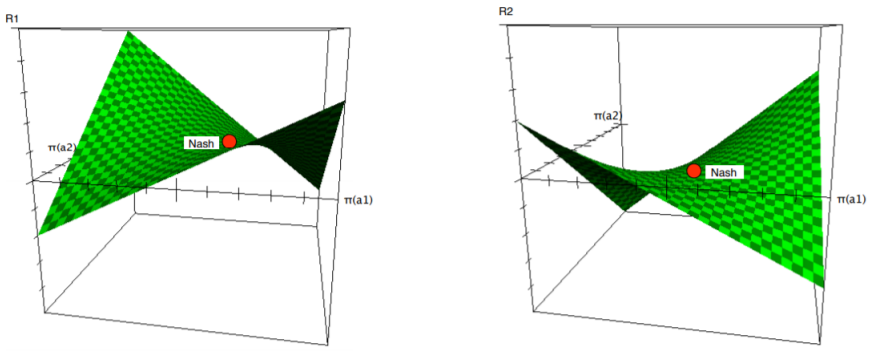
Une caractéristique importante des jeux de matrice est l'existence d'au moins un équilibre de Nash dans ces jeux.

Nous classifions les jeux de matrices dans les catégories suivantes :

Jeux à somme nulle *(zero-sum games)* sont des jeux pour deux joueurs ($n = 2$) où la récompense de chaque joeur est symétrique à celle de son adversaire. Le jeu Chifoumi est un exemple de tel jeu. L'article de Littman a étudié seulement cette catégorie.

Dans les jeux de ce type, il peut y avoir plusieurs points d'équilibres, qui ont donc les mêmes récompenses et sont interchangeables. De plus, l'équilibre de Nash correspond au pire scénario, car si le premier agent joue une politique d'équilibre π_1 l'autre agent ne pourra pas améliorer sa récompense en jouant une autre politique que π_2, de même, le premier joueur ne peut pas avoir une récompense plus petite que sa récompense actuelle.

Dans les jeux aux tours alternés, la solution est d'appliquer une approche minimax, ce qui retourne une politique déterministe. Néanmoins, dans les jeux de matrice où les joueurs jouent simultanément la politique optimale obtenue avec le minimax est une politique face à *l'espace des politiques* plutôt que l'espace d'actions, elle est donc optimale et probabiliste. La problème minimax se réécrit sous la forme d'un problème d'optimisation linéaire comme nous l'avons dans le papier de Littman [equation 2.1].



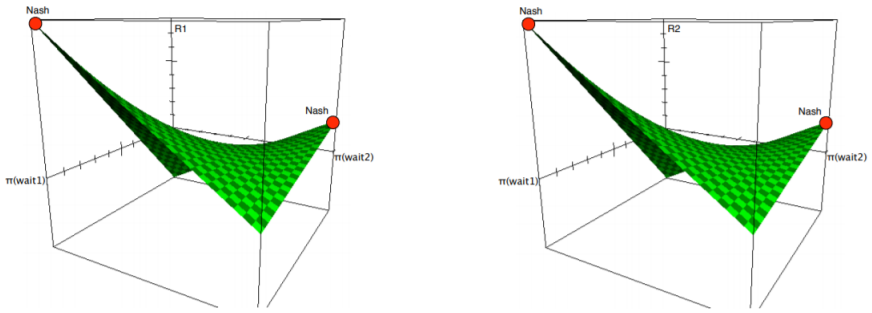
(a) Reward function for player 1 (b) Reward function for player 2

FIGURE 3.2 – Récompense et équilibres de Nash pour un jeu à somme nulle

La représentation de la fonction de récompense en figure 3.2 permet de voir que le point d'équilibre est un point-selle. On voit aussi la symétrie par rapport au plan horizontal nul.

Jeux d'équipe *(team games)* peuvent avoir un n nombre de joeurs. En revanche, la récompense est la même pour chaque action commune.

Dans ce type de jeux il existe aussi des solutions triviales comme les récompenses sont les mêmes pour tout les agents, l'équilibre de Nash peut être trouver simplement en cherchant l'action commune qui maximise la récompense. Similairement au cas de MDP, la politique est gloutonne. En effet, toutes les distributions de probabilités sur les actions commune gloutonnes sont des équilibres de Nash. On voit sur la figure 3.3 les courbes de récompenses identiques.



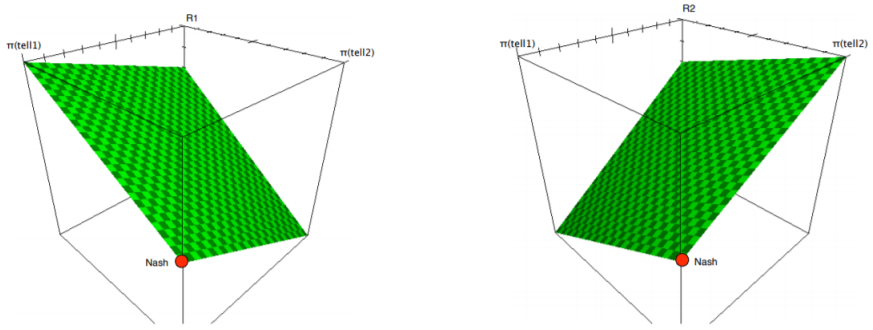
(a) Reward function for player 1 (b) Reward function for player 2

FIGURE 3.3 – Récompense et équilibres de Nash pour un jeu d'équipe

Jeux à somme générale *(general-sum games)* sont touts les autres types de jeux. Il s'agit souvent des jeux où les joeurs ne sont pas tous dans la même équipe et ne sont pas complètement opposés. Un exemple fameux est la Dilemme du Prisonnier.

En revanche, il n'est pas aussi facile de trouver l'équilibre de Nash pour ces jeux. Néanmoins, le problème s'écrit sous la forme d'un problème d'optimisation quadratique quand il s'agit de deux joueurs seulement.

Certains problèmes comme la dilemme du prisonnier n'ont qu'un seul point d'équilibre, qui est dans ce cas là déterministe pour les deux joueurs.



(a) Reward function for player 1 (b) Reward function for player 1

FIGURE 3.4 – Récompense et équilibres de Nash pour un jeu à somme générale

3.1.2.2 Jeux stochastiques

Les jeux stochastiques (jeux markoviens), comme nous avons vu avec Littman, peuvent être vus comme une extension des jeux de matrice et/ou MDP car elles traîtent les cas *multi-agents* dans une situation *multi-état*.

Les mêmes notions d'optimalité vues précédament s'appliquent pour les jeux stochastiques en adaptant simplement au modèle multi-agent mult-état. La valeur d'état et la Q-value se donnent ainsi par :

$$V_i^{\pi}(s) = \sum_{a \in A} Q(s,a)\pi(s,a) \tag{3.3}$$

$$Q(s,a) = R_i(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V_i^{\pi}(s') \tag{3.4}$$

Jouer contre des politiques stationnaires Si toutes les politiques sont stationnaire sauf pour un agent, le jeu stochastique se réduit à un MDP. En effet, tout ce que les autres agent font est définir les probablités de transition et les récompenses pour le MDP équivalent.

La methode Q-learning suffira dans tel situation pour converger vers la politique *best-response* qui est optimale face à la politique commune des adversaires.

Les algorithmes de résolutions des jeux stochastiques peuvent être divisées en deux catégories :

— **best-response learners :** il s'agit d'agents qui essayent d'apprendre une politique $\mathcal{BR}$ face à la politique commune de ses adversaires.
— **equilibrium learners :** il s'agit d'agents qui se contentent d'apprendre des politiques d'équilibres, quelque soit leurs adversaires.

3.1.2.3 Best-response Learners

Les méthodes dans cette catégorie n'essayent pas d'apprendre une politique d'équilibre, elles essayent plutôt d'apprendre une politique qui est optimale par rapport à celles de ses adversaires. Il s'agit donc d'apprendre une politique d'attaque, ce qui permet ainsi dans le cas où les adversaires adoptent des politiques d'équilibres d'obtenir des meilleurs récompenses que celles que l'équilibre propose. En revanche, si ses adversaires n'ont pas une politique stationnaire ou ne convergent pas vers une, la méthode peut rencontrer des difficultés à s'adapter à ses adversaires.

Méthodes MDP Comme on l'a déjà constaté, un jeu stochastique est équivalent à un MDP du point de vu d'un agent si tout ses adversaires adoptent des politiques stationnaires. Dans ce cas là, tous les algorithmes d'apprentissage de politiques optimales pour un MDP peuvent être appliqués pour le jeu stochastique.

Ce principe repose sur une hypothèse qui n'est pas très réaliste : les autres agents n'essayent pas d'apprendre et d'adapter leurs politiques. Généralement, les autres agents essayent également de maximiser leurs récompenses et

ne peuvent pas être battus avec des politiques déterministes, alors que la plupart des méthodes MDP trouve des politiques déterministes.

Ce cas revient à l'application de Littman du Q-learning dans le jeu markovien, les actions des adversaires sont traîtés simplement d'états du jeu. Ce qui justifie l'échec total des politiques Q-learning face à un agent qui a appris à battre ces politiques.

Joint-action Learners L'apprentissage des *Joint-action Learners* (JALs) [Claus and Boutilier, 1998] répond au problèmes de jeux d'équipe, où la fonction de récompense est la même pour tous les agents. Ces méthodes diffèrent de celles des MDP par le fait qu'elles apprennent les valeurs de Q suivant les *actions communes* plutôt que les actions de chacun, supposant que chaque agent peut observer l'état et les actions des autres.

Néanmoins, un problème risque d'apparaître lorsque les agents essaye de coordonner une action commune : il peut y avoir un décalage dans l'apprentissage des agents ce qui empêche de trouver une action commune *cohérente*. D'où [Claus and Boutilier, 1998] propose une fonction de qualité pour remplacer la fonction Q.

$$EV(a_i) = \sum_{a_{-i} \in A_{-i}} Q((a_i, a_{-i})) \prod_{j \neq i} \widehat{\pi}_j(a_{-i}[j])$$

où $\widehat{\pi}_j$ est une estimateur de la politique de l'agent j, il s'obtient en divisant le nombre de fois que l'agent j choisit une action par le nombre de tentatives.

Opponent Modeling Il s'agit d'une méthode similaire à JALs mais qui s'applique dans le contexte des jeux à somme nulle. Dans cette méthode, on considère que les autres agents représente un seul *grand* agent avec la capacité d'effectuer des actions communes, et donc estime statistiquement ces actions et ces politiques.

$$\widehat{\pi_{-i}}(a_{-i}) = \frac{n(s, a_{-i})}{n(s)}$$

où $n(s, a_{-i})$ représente le nombre de fois l'action commune adversaire a_{-i} a été choisie par la politique π_{-i} à l'état s, et $n(s)$ représente le nombre visite de l'état s.

Algorithme 4 : Opponent modeling

Initialiser $Q(s, a)$ arbitrairement
$n(s) \leftarrow 0, \forall s \in S$
$n(s, a_{-i}) \leftarrow 0, \forall s \in S, \forall a_{-i} \in A_{-i}$
Initialiser s
boucle
  $a \leftarrow$ tirée aléatoirement selon sa politique
  Effectuer l'action a_i, recevoir une récompense r, arriver à l'état s' et les adversaires effectuent l'action a_{-i}
  $Q(s, a_{i,-i}) \leftarrow (1 - \alpha) Q(s, a_i, a_{-i}) + \alpha \left(r + \gamma V(s')\right)$
  $V(s) \leftarrow \max_{a_i \in A_i} \sum_{a_{-i} \in A_{-i}} Q(s, a_i, a_{-i}) \widehat{\pi_{-i}}(a_{-i})$
  $n(s, a_{-i}) \leftarrow n(s, a_{-i}) + 1$
  $n(s) \leftarrow n(s) + 1$
  $s \leftarrow s'$
fin

3.1.2.4 Equilibrium Learners

Les méthodes de cette catégorie cherchent des politiques qui convergent vers des équilibres de Nash pour les jeux stochastiques. Ces équilibres sont difficiles à trouver dans les cas de plus que deux agents, donc ces méthodes s'intérèsse souvent au jeux à somme nulle ou aux jeux à somme générale pour deux personnes. L'avantage de trouver l'équilibre de Nash est que la politique est bornée inférieurement et converge vers cette borne, la politique résultante est presque indépendante de ses adversaires à la limite. L'idée générale est de trouver une politique d'équilibre pour chaque état qui permet de renvoyer le jeu à l'équilibre de Nash.

General Equilibrium Learner Neto formule dans son article [Neto, 2005] une stratégie générale pour apprendre des politiques optimales, il donne la formule de la fonction Q par :

$$Q_i^*(s,a) = R_i(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_i^{\pi^*}(s'), \forall i \in \{1,\ldots,n\}$$

où $V_i^{\pi^*}(s')$ représente la valeur d'équilibre pour l'agent i quand la politique commune jouée est l'équilibre de Nash π^* qui est calculée par rapport aux valeurs de Q.

Neto explique que cette approche fonctionne généralement pour les jeux avec un seul point d'équilibre ou bien s'il existe un protocol de coordination pour que tous les agents converges vers le même équilibre. Il est implicitement supposé que l'agent peut observer les états et les actions de tous les autres agents.

Algorithme 5 : General equilibrium learner

Initialiser $Q(s,a)$ arbitrairement
Initialiser s
boucle
  $a_i \leftarrow$ tirée aléatoirement selon la politique de Nash dérivée de Q(s,a) pour l'agent i avec une politique d'exploration
  Effectuer l'action a_i, recevoir une récompense r_i, arriver à l'état s' et les adversaires effectuent l'action commune a_{-i}
  pour $i = 1 \ldots n$ **faire**
    $Q_i(s, a_{i,-i}) \leftarrow (1-\alpha) Q_i(s,a_i,a_{-i}) + \alpha\,(r_i + \gamma V_i(s'))$
  fin
  $V(s) \leftarrow \text{Nash}([Q(s,a)])$
  $s \leftarrow s'$
fin

Minimax-Q Il s'agit de la méthode proposé [Littman, 1994] qui s'adresse au cas des jeux stochastique à somme nulle. Neto rajoute que la converge de cet algorithme vers une politique d'équilibre a été montré (par Littman en 1996). Cet algorithme sert de point de départ pour les méthodes d'apprentissage d'équilibre de Nash.

Nash-Q [Hu and Wellman, 2003] essaye de s'adresser au problème d'apprentissage de deux joueurs dans un jeu à somme générale, en s'inspirant du travail de Littman. La valeur d'équilibre est formulé comme un problème d'optimisation quadratique. Ce papier essaye d'étendre l'algorithme dans le contexte multi-agent *non-coopératifs*. Le principe de l'algorithme est de permettre à l'agent d'observer les récompenses des autres agents.

L'algorithme impose l'hypothèse que les jeux intermédiaires doivent avoir un seul point d'équilibre et que chaque équilibre devrais être un point-selle. Neto explique que cette condition est très stricte et contraignant car il n'est pas possible de savoir si l'hypothèse tiendra pendant l'apprentissage.

3.2 Mean Field Multi-Agent Reinforcement Learning

Cette article [Yang et al., 2018] s'adresse à la recherche d'une politiques d'équilibre pour une grande population d'agents. Yang propose de considérer les interactions dans la population d'agents celle entre un agent simple et l'effet moyen de la population total ou les agents voisins. L'apprentissage des politiques optimales des individus dépend des dynamiques de la population. L'article présente des algorithmes d'apprentissage mean field Q-learning et mean field Actor-Critic, nous ne n'intéressons qu'au premier car nous avons bien étudié l'algorithme Q-learning et ses variantes, contrairement à Actor-Critic.

3.2.1 Problématique

Les approches de recherche d'équilibres existants, ne peuvent répondre qu'au cas de quelques agents. La complexité importante des recherches directe d'équilibres de Nash ne permet pas d'appliquer ces approches aux cas des grand groupes d'agents. Pourtant, ils existent de nombreux cas pratiques qui requièrent des interactions stratégiques parmi des grand groupes d'agents. Par exemple, les robots de jeux enligne multi-player ou les agents traders dans les marchés de bourses.

3.2.2 Mean Field MARL

La dimension de l'action commune $\mathbf{a}$ est proportionnelle au nombre d'agents (on note $\mathbf{a}$ le vecteurs d'actions). Comme les agents suivent leurs politiques et évaluent simultanément leurs fonctions de valeurs selon les actions $\mathbf{a}$, l'apprentissage des de la fonction de valeur standard $Q_i(s, \mathbf{a})$ n'est plus faisable. Le principe de la méthode Mean Field est de factoriser la fonction Q par ses interactions locales deux-à-deux.

$$Q_i(s, \mathbf{a}) = \frac{1}{n_i} \sum_{k \in \mathcal{N}(i)} Q_i(s, a_i, a_k) \tag{3.5}$$

où $\mathcal{N}(i)$ est l'ensemble d'agents voisin de l'agent i de cardinalité $n_i = |\mathcal{N}(i)|$ détérminé par les paramètres spécifiques de l'application.

3.2.2.1 L'approximation Mean Field

Les interactions des $Q_i(s, a_i, a_k)$ dans [eq. 3.5] peuvent être approchées à l'aide de la théorie Mean Field (théorie du champ moléculaire).

L'action *moyenne* (mean) $\bar{a}_i$ se calcule à partir de du voisinage $\mathcal{N}(i)$ de l'agent i, l'action a_k de chaque voisin k s'exprime en fonction de la somme des $\bar{a}_i$ et d'une petite fluctuation $\delta a_{i,k}$

$$a_k = \bar{a}_i + \delta a_{i,k} \quad \text{où } \bar{a}_i = \frac{1}{n_i} \sum_{k \in \mathcal{N}(i)} a_k$$

Si la fonction Q est dérivable deux fois, elle peut être approché par un développement de Taylor, et [Yang et al., 2018] montre que

$$Q_i(s, \mathbf{a}) = \frac{1}{n_i} \sum_{k \in \mathcal{N}(i)} Q_i(s, a_i, a_k) \approx Q_i(s, a_i, \bar{a}_i)$$

L'apprentissage de la fonction Q est donc défini par :

$$Q_i(s, a_i, \bar{a}_i) \leftarrow (1 - \alpha) Q_i(s, a_i, \bar{a}_i) + \alpha \left(r_i + \gamma V_i(s')\right) \tag{3.6}$$

Grâce à cette approximation, le problème MARL se réduit à la recherche de politique optimale de l'agent π_i par rapport à l'action moyenne $\bar{a}_i$ et de ses voisins.

3.2.2.2 L'algorithme proposé

Une méthode itérative est proposé par [Yang et al., 2018] pour calculer la meilleur politique π_i pour chaque agent i.

1. Calculer l'action moyenne $\bar{a}_i = \frac{1}{n_i} \sum_{k \in \mathcal{N}(i)} a_k$, où les actions a_k sont déterminées par les politiques π_k et leurs actions moyennes précédentes $\bar{a}_k$.
2. Calculer la politique *Boltzmann* par la formule

$$\pi_i(a_i|s, \bar{a}_i) = \frac{\exp\left(-\beta Q_i(s, a_i, \bar{a}_i)\right)}{\sum_{a'_i \in A_i} \exp\left(-\beta Q_i(s, a'_i, \bar{a}'_i)\right)}$$

Cette itération permet d'améliorer les actions moyennes $\bar{a}_i$ et les politiques π_i correspendantes pour tous les agents i.

Il est montré par [Yang et al., 2018] que l'action moyenne $\bar{a}_i$ converge vers un seul point d'équilibres, et donc la politique π_i converge également.

3.2.2.3 Implémentation

Yang propose de reformuler le calcul de la fonction Q [eq. 3.6] à l'aide des réseaux de neurones artificielles, en paramétrisant la fonction Q par des points ϕ, permettant ainsi d'entraîner l'agent i en minimisant la fonction de perte suivante :

$$\mathcal{L}(\phi_i) = \left(y_i - Q_{\phi_i}(s, a_i, \bar{a}_i)\right)^2 \quad \text{avec } y_i = r_i + \gamma V_{\phi_i}(s')$$

En dérivant la fonction de perte on obtient

$$\nabla \mathcal{L}(\phi_i) = \left(y_i - Q_{\phi_i}(s, a_i, \bar{a}_i)\right) \nabla Q_{\phi_i}(s, a_i, \bar{a}_i)$$

ce qui permet d'appliquer des méthodes d'optimisation de gradient.

Cette méthode est appelé **MF-Q** [Yang et al., 2018]. Yang propose également de modéliser la politique par des réseaux de neurones explicitement avec des poids θ, emmenant ainsi à une méthode *actor-critic* (MF-AC) que nous n'allons pas approfondir.

3.2.3 Résultats et conclusions

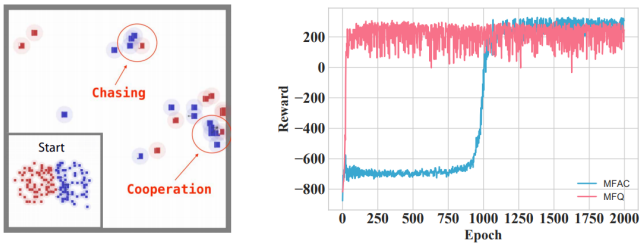
Le papier analyse ses algorithmes dans trois scénarios différents, nous nous intéressons au troisième : un jeu de bataille.

Environnement Le jeu de bataille est testé sur la plateforme open-source **MAgent** [Zheng et al., 2017]. Il s'agit d'un cas multi-agent *cooperatif-compétitif*, où il y a deux armées qui se battent dans une grille $2D$. Dans cette expérience, chaque armée est entraînée avec un algorithme RL différent, l'armée consiste de 64 agents dont le but est d'obtenir plus de récompenses en coopérant avec leurs coéquipiers afin de détruire les ennemies.

Les récompenses du jeu sont :
- un mouvement : -0.005
- attaque contre un ennemie : 0.2
- tuer un ennemie : 5
- attaque contre une case vide : -0.1
- se faire attaquer/tuer : -0.1

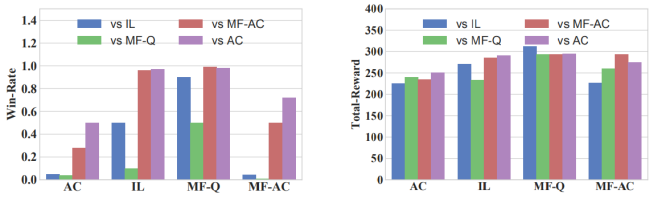
Résultats Les modèles **MF-Q** et **MF-AC** ont été comparés avec les modèles de bases de la plateforme MAgent, et ensuite contre un modèle $Q-learning$ indépendant (comme celui vu avec Littman et Neto) que Yang appelle **IL**, et enfin un modèle Actor-Critic AC (non Mean-Field).

L'apprentissage des agents MF-Q et MF-AC s'est fait contre eux-même sur 2000 jeux. Les deux agents apprennent très vite [fig. 3.5], et ont un grand succès comparé avec les modèles de bases et contre IL et AC également 3.6.



(a) Battle game scene. (b) Learning curve.

FIGURE 3.5 – Bataille et courbe d'apprentissage



(a) Average wining rate. (b) Average total reward.

FIGURE 3.6 – Comparaisons des algorithmes dans le jeu de bataille

Conclusion Ce papier a proposé deux méthodes de recherche de politiques d'équilibres MF-Q et MF-AC, en partant de minimax-Q [Littman, 1994] et Nash-Q [Hu and Wellman, 2003], et en l'adaptant à l'aide de la théorie de Mean Field, pour répondre à un contexte complexe qui est le cas *coopératif-compétitif* à grand échelle.

Le papier présente des expériences qui ont permis de confirmer l'efficacité des algorithmes proposées. De plus, une analyse théorie de convergence des algorithme est présentée dans le papier que nous avons fait le choix d'omettre à cause des contraintes de temps.

Chapitre 4

Conclusion

Cette étude m'a permis de découvrir le domaine d'apprentissage multi-agent par renforcement et des classes différentes de problèmes que le domaine permet de traîter.

En partant du papier de Littman, j'ai pu apprendre les premières notions de MDP et de MARL, et d'étudier le cas spécifique d'un jeu à somme nulle à deux joueurs qui a jeté les bases pour les futurs recherches d'équilibres dans les cas plus complexes et à plus grandes échelles comme nous avons vu dans [Yang et al., 2018].

D'une part, le papier de revu [Neto, 2005] a expliqué en détails les notions importantes d'optimalité et d'équilibre d'un point de vu mathématique et l'état de l'art des algorithmes de recherches de politiques *optimales*, en classifiant les méthodes différentes permettant de mieux comprendre le papier de Littman et d'apprécier l'importance de sa contribution dans le domaien de MARL.

D'autre part, le papier [Yang et al., 2018] qui est plus récent que ces deux premiers a présenté une nouvelle approche de MARL en s'appuyant sur les principes de Minimax-Q et Nash-Q, permettant d'adapter ces algorithmes à très grand échelle en appliquant la théorie de Mean Field.

Pour conclure, le domaine de MARL est très riche et vaste, et les papiers étudiés permettent d'élargir beaucoup plus sur le sujet. J'aurais sans doute aimé d'élargir la recherches sur d'autres approches récentes. Néanmoins, j'ai dû me contenter d'étudier les parties de ces articles que j'ai trouvé pertinentes avec la prémisse du papier de Littman.

Bibliographie

[Bellman, 1957] Bellman, R. (1957). *Dynamic programming*. Princeton University Press, Princeton, NJ, USA, 1 edition. tex.bib2html_rescat : General RL.

[Bertsekas, 1987] Bertsekas, D. P. (1987). *Dynamic programming : deterministic and stochastic models*. Prentice-Hall, Inc., USA.

[Claus and Boutilier, 1998] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 746–752, Madison, Wisconsin, USA. American Association for Artificial Intelligence.

[Foerster, 2018] Foerster, J. N. (2018). *Deep multi-agent reinforcement learning*. http ://purl.org/dc/dcmitype/Text, University of Oxford.

[Hu and Wellman, 2003] Hu, J. and Wellman, M. P. (2003). Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4(Nov) :1039–1069.

[Littman, 1994] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, pages 157–163, New Brunswick, NJ, USA. Morgan Kaufmann Publishers Inc.

[Neto, 2005] Neto, G. (2005). From Single-Agent to Multi-Agent Reinforcement Learning : Foundational Concepts and Methods. *Instituto de Sistemas e Robótica, Instituto Superior Técnico*.

[Shapley, 1953] Shapley, L. (1953). Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America*.

[Solan and Vieille, 2015] Solan, E. and Vieille, N. (2015). Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 112(45) :13743–13746.

[Watkins, 1989] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, UK. tex.bib2html_rescat : Parameter.

[Yang et al., 2018] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. (2018). Mean Field Multi-Agent Reinforcement Learning. *arXiv :1802.05438 [cs]*. arXiv : 1802.05438.

[Zheng et al., 2017] Zheng, L., Yang, J., Cai, H., Zhang, W., Wang, J., and Yu, Y. (2017). MAgent : A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence.