
Mémoire de stage de Licence 3
JEUX STOCHASTIQUES PARTIELLEMENT OBSERVABLES

ABEL DOUZAL (ENS-PSL)

8 - août - 2023

ENCADREMENT : BRUNO GAUJAL (INRIA) ROMAIN CRAVIC (LIG-UGA)

Table des matières

1	Introduction	2
1.1	Jeux stochastiques	2
1.2	Fonction objectif et fonction de valeur	3
1.3	Équation de Bellmann	4
1.4	Typologie de jeux stochastiques	4
1.4.1	MDP	4
1.4.2	SG	5
1.4.3	POMDP	5
2	POSG	6
2.1	Définition	6
2.2	Limitations	6
2.3	OS-POSG	6
2.4	Bilan	7
3	Réduction de la dimension du jeu	7
4	Méthode de jeu pour les POSG par proximité des historiques	9
4.1	Motivation et principe général	9
4.2	Reconnaissance de séquences clefs	9
4.2.1	Distance sur l'espace des séquences	9
4.2.2	Sélection des séquences clefs	10
4.3	Optimisation de la stratégie de jeu sur l'espace des séquences clefs	10
4.4	Pseudocode et implémentation	11
4.5	Expérimentation	12
4.5.1	Description des jeux et des stratégies	12
4.5.2	Paramétrage du facteur d'exploration λ	13
4.5.3	Résultats	13
4.5.4	Analyse	14
4.6	Perspective et conclusion	14
	Bibliographie	15

1 Introduction

Les jeux stochastiques partiellement observables (POSG) sont une modélisation des jeux à deux joueurs possédant une information parcellaire de l'état du jeu. L'objectif de ce stage est d'étudier les POSG, et de déterminer des comportements artificiels efficaces pour les joueurs.

On présentera dans un premier temps certaines sous-classes de POSG et les stratégies de résolutions qui ont été utilisées pour les résoudre, puis on s'intéressera aux limites de résolutions envisageables pour les POSG dans le cas général. Enfin on proposera une méthode pour faciliter la résolution en réduisant la dimension de ces jeux, puis une méthode de jeu par proximité des historiques dont on évaluera les performances sur un jeu de poursuite.

Les exécutable mentionnés dans le présent rapport sont fournis sur moodle sous forme d'une archive **tar**.

1.1 Jeux stochastiques

Les POSG sont des jeux stochastiques. Présentés dans ce qui suit comme fonctionnant selon le même principe : le jeu se fait sur un espace d'états fini, avec un ou plusieurs joueurs. Les joueurs ont accès à leur propre historique d'actions et d'observations, les états de jeu ou cachés. La partie se déroule tour à tour, chaque tour se déroule comme suit : les joueurs jouent chacun simultanément une action. Un nouvel état du jeu et les nouvelles observations sont obtenus selon un noyau de probabilités de transition dépendant de l'état du jeu courant et des actions jouées, un gain est alors obtenu par chaque joueur. L'objectif global de chaque joueur est de maximiser la somme de ses gains. Pour cela on cherche à déterminer la meilleure stratégie pour chaque joueur.

On distingue -comme représenté dans le tableau suivant- quatre sous-classes de jeux stochastiques selon que l'état du jeu soit ou non visible par les joueurs et pour 1 ou 2 joueurs. Dans le cas où l'état du jeu est visible, on parlera de jeu à *information complète* ou *observable*, et dans le cas où il ne l'est pas de jeu à *information incomplète* ou *partiellement observable*. Pour rendre l'état du jeu visible par les joueurs, il suffit que l'observation obtenue à chaque tour soit l'état du jeu.

	1 joueur	2 joueurs
observable	MDP	SG
partiellement observable	POMDP	POSG

Pour tout jeu stochastique, une stratégie est une application de l'espace des historiques des informations accessibles à un joueur donné au cours d'une partie à l'espace des actions (cas MDP) ou dans l'espace des distributions de l'ensemble des actions (cas SG et jeux partiellement observables) jouables par ce même joueur. Les joueurs peuvent avoir différentes stratégies pour jouer, on en distingue différents types selon ce que l'on cherche à obtenir au final.

Définition 1.1. On distingue trois principaux types de stratégies :

- *Conservative* : vise à maximiser le pire gain
- *Optimiste* : vise à maximiser le meilleur gain
- *Coopérative* : vise à maximiser le gain commun (i.e. la somme des gains des joueurs)

Définition 1.2. Un jeu à deux joueurs à somme nulle est un jeu pour lequel la somme des gains des deux joueurs est de somme nulle (ou constante) à tout instant du jeu.

Dans la suite, les jeux à deux joueurs sont considérés à somme nulle. Le principal intérêt de cette contrainte est de garantir l'existence d'équilibres de Nash entre les stratégies, par application du théorème minimax de von Neumann.

De manière générale, l'approche de résolution la plus utilisée, découlant des travaux réalisés sur les MDP par Bellmann est la *value iteration*, consiste à approcher la *fonction de valeur* ou *value function* correspondant au meilleur gain réalisable face au meilleur adversaire, soit l'espérance du gain obtenu à l'équilibre de Nash. Dans un second temps, des méthodes sont utilisées pour établir des stratégies qui lorsque elles sont suivies fournissent une espérance de gain proche de cette valeur d'équilibre.

1.2 Fonction objectif et fonction de valeur

La méthode la plus courante pour résoudre un jeu stochastique est de le définir comme un problème d'optimisation d'une fonction objectif associé aux gains du jeu dépendant des stratégies choisies.

On distingue plusieurs types de fonctions objectif, pour une suite de gains obtenus $(r_t)_{t \in \mathbb{N}^*}$. On distingue le cas horizon fini $(r_t)_{1 \leq t \leq T}$ où $T \in \mathbb{N}^*$ des cas à horizon infini (Escomptée, Totale, Moyenne) définis comme suit :

Type de fonction objectif	Expression
Horizon fini	$\sum_{t=1}^T r_t$
Escomptée	$\sum_{t=1}^{+\infty} \gamma^{t-1} r_t$
Totale	$\sum_{t=1}^{+\infty} r_t$
Moyenne	$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T r_t$

Dans le cas horizon fini, la définition de la fonction objectif influe sur la stratégie à adopter. On s'intéressera dans la suite essentiellement au cas escompté.

On considère un jeu stochastique dont le gain à un instant t pour un joueur est r_t . On note s_1 l'état initial du jeu, et π le n -uplet de stratégies suivies par les n joueurs. Pour tout $t \in \mathbb{N}$, s_t est une variable aléatoire dépendant de π , du toyo de de probabilités de transition et des s_i pour $i < t$. De plus, pour tout $t \in \mathbb{N}$, r_t est une variable aléatoire dépendant des s_i pour $i \leq t$. La fonction de valeur V du jeu dépendant de stratégies π , est définie sur l'espace des états du jeu. Lorsque le jeu dure un temps au plus T (horizon fini), la fonction de valeur est alors définie par :

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=1}^T r_t \mid s_1 = s \right]$$

Dans le cas d'un jeu pouvant durer arbitrairement longtemps (horizon infini), on introduit un paramètre supplémentaire $\gamma \in]0, 1[$ (le facteur d'escompte) correspondant au poids donné aux gains futurs relativement au gain présent. On définit alors V par :

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_1 = s \right]$$

1.3 Équation de Bellmann

Dans la plupart des jeux présentés dans la suite, la fonction de valeur V vérifie une équation de Bellmann, c'est à dire une équation fonctionnelle (ici en version déterministe) du type :

$$V(s_0) = \max_{a_0} F(s_0, a_0) + \beta V(s_1)$$

Sous les contraintes suivantes :

$$a_0 \in \Gamma(s_0), s_1 = T(s_0, a_0)$$

Où les s_i sont des états de jeu, les a_i des actions, F une fonction réelle, $\Gamma(s)$ l'ensemble des actions jouables depuis l'état s et T une fonction de transition. Lorsque l'application qu'elle définit est contractante, le théorème du point fixe de Picard assure l'existence d'une unique solution à cette équation (c'est à dire un point fixe).

1.4 Typologie de jeux stochastiques

1.4.1 MDP

Un problème de décision de Markov ou MDP (Markov decision process) est un jeu stochastique à un joueur à information complète. C'est le cas le plus simple de tels jeux. Formellement, un MDP est un quadruplet (S, A, R, P) où :

- S est l'ensemble fini des états
- A est l'ensemble fini des actions jouables
- $R : S \times A \rightarrow \mathbb{R}$ est la fonction de gain
- $P : S \times A \rightarrow \Delta(S)$ est le noyau de probabilités de transition

On se donne aussi un facteur d'escompte $\gamma \in]0, 1[$.

On remarque que dans ce type de jeu, comme pour une chaîne de Markov, la stratégie optimale consiste à jouer une action de manière déterministe en fonction de l'état dans lequel on se trouve. Ce n'est pas le cas dans la plupart des SG ou dans les jeux à information partielle.

La fonction de Bellmann pour ce jeu définie sur l'espace des fonctions de valeur est :

$$H[V](s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \right] \quad (1)$$

Qui est bien contractante, on en déduit l'équation de Bellmann optimale, c'est à dire celle réalisée par l'unique point fixe ($H[V^*] = V^*$) :

$$V^*(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s') \right] \quad (2)$$

On peut approcher V^* grâce à l'algorithme 1, dont la correction et la convergence sont assurées par le théorème du point fixe de Picard.

Algorithm 1: Value iteration pour MDP

Data: MDP (S, A, R, P) , $\epsilon > 0$
Result: V tel que $\|H[V] - V\|_\infty < \epsilon$
 $V^{(0)}(s) \leftarrow 0 \quad \forall s \in S$
 $k \leftarrow 0$
while $\|V^{(k)} - V^{(k-1)}\|_\infty \geq \epsilon$ **or** $k = 0$ **do**
 $V^{(k+1)} \leftarrow H[V^{(k)}]$
 $k \leftarrow k + 1$

J'ai implémenté cet algorithme dans le cas d'un jeu simple sur une grille (voir `robot.py`). et observé son comportement selon ses paramètres. Le jeu consiste en un robot qui se promène sur une grille et obéit à la direction choisie (parmi haut, bas, droite, gauche) avec une proba $(1 - p)$, et bouge aléatoirement avec une proba p . Si le robot sort de la grille, il tombe et c'est fini, sinon il gagne le reward de la case sur laquelle il se trouve. En outre, à chaque instant, le robot a une proba $(1 - \gamma)$ de tomber en panne de batterie et de s'arrêter

de fonctionner.

Cet algorithme ne donne pas directement la stratégie à adopter, cependant elle s'obtient facilement, en

sélectionnant pour un $s \in S$ donné le $a \in A$ maximisant dans l'équation (2). C'est le principe de l'*algorithme d'itération de stratégie* ou *Policy iteration*.

1.4.2 SG

Les stochastic games ou SG sont des jeux à information complète à deux joueurs, leur cas a été traité dans [3], on ne s'y attardera pas. Leur résolution met en oeuvre des méthodes de résolution assez similaires à celles pour les MDP, à la différence près qu'il est nécessaire de randomiser les stratégies de jeu, c'est à dire sélectionner les actions à jouer selon une loi de probabilités à déterminer.

1.4.3 POMDP

Les problèmes de décision de markov à observation partielle ou POMDP (partially observable Markov decision process) sont une classe de jeux à un joueur, où le joueur a une connaissance partielle de son état du jeu. Plus formellement, un POMDP est un quintuplet (S, A, O, T, R) où :

- S est l'ensemble fini des états
- A est l'ensemble fini des actions jouables
- O est l'ensemble fini des observations que peut faire le joueur
- $T : S \times A \rightarrow \Delta(O \times S)$ est le noyau de probabilités de transition
- $R : S \times A \rightarrow \mathbb{R}$ est le gain du joueur obtenu à état donné et action effectuée

De plus, au départ le joueur connaît une distribution sur S , $b^{init} \in \Delta(S)$ correspondant à une hypothèse initiale de présence sur chaque état. Cette hypothèse est mise à jour au cours du jeu par :

$$\tau(b, a, o) = \frac{1}{|S|\mathbb{P}(o | a)} O(s', a, o) \sum_s T(s, a, s') b(s)$$

En notant $O(s', a, o)$ la probabilité d'observer $o \in O$ sachant que le jeu est dans l'état s' et que l'action a a été jouée. On utilise ici une distribution sur S plutôt que un état puisque il est inconnu. Cela est aussi employé pour la fonction de valeur.

On donne la trame de l'algorithme 2 : HSVI (heuristic search value iteration) pour POMDP (algorithme 3) proposé dans [4]. Le principe utilisé ici sera réutilisé dans le cas des OS-POSG. On cherche dans ce cas à approcher

$$V^*(b) = \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid b \right] \quad (3)$$

L'équation de Bellmann devenant :

$$H[V](b) = \max_a \left[\sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in O} \mathbb{P}(o \mid b, a) V(\tau(b, a, o)) \right] \quad (4)$$

Pour ce faire, on encadre V^* par deux fonctions $\bar{V}$ et $\underline{V}$, de façon à avoir $V^*(b) \in [\bar{V}(b), \underline{V}(b)] \quad \forall b \in \Delta(S)$. La fonction $\underline{V}$ est représentée comme un maximum d'applications linéaires, et $\bar{V}$ comme étant la fonction dont le graphe est l'enveloppe convexe de points déterminés. $\bar{V}$ et $\underline{V}$ sont représentées par les ensembles Γ et Υ respectivement :

- $\underline{V}(b) = \max_{\alpha \in \Gamma} (\alpha \cdot b)$ où Γ est un ensemble de vecteurs de $\mathbb{R}^{|S|}$
- Υ est un ensemble point/valeur $(b_i, \bar{v}_i) \in \mathbb{R}^{|S|} \times \mathbb{R}$ dont le graphe de $\bar{V}$ est l'enveloppe convexe (plus précisément la partie min de cette enveloppe).

Ces fonctions sont mises à jour dans l'algorithme 2 pour affiner l'approximation, de manière à obtenir $|\bar{V}(b^{init}) - \underline{V}(b^{init})| \leq \varepsilon$.

L'algorithme **explore** (4) termine, puisque pour un état b fixé, $\varepsilon \gamma^{-t} \xrightarrow[t \rightarrow +\infty]{} +\infty$ et l'écart entre $\underline{V}$ et $\bar{V}$ est borné. L'heuristique de choix donnée dans (4) se justifie, en cherchant à réduire la contribution participant le plus à l'incertitude globale, c'est à dire à l'écart entre $\underline{V}$ et $\bar{V}$.

Algorithm 2: HSVI pour POMDP

Data: POMDP (S, A, O, T, R) , $\epsilon > 0$

Result: $\pi = \text{HSVI}(\epsilon)$

On initialise en prenant les bornes max et min

while $|\bar{V}(b_0) - \underline{V}(b_0)| > \epsilon$ **do**

 | **explore**($b_0, \epsilon, 0$)

Renvoyer la stratégie réalisant la borne

inférieure : jouer l'action a réalisant la valeur

$\underline{V}(b)$

Algorithm 3: backup($\underline{V}, b$)

$$\beta_{a,o} \leftarrow \arg \max_{\alpha \in \Gamma} (\alpha \cdot \tau(b, a, o))$$

$$\beta_a(s) \leftarrow R(s, a) + \gamma \sum_{o, s'} \beta_{a,o}(s') O(s', a, o) T(s, a, s')$$

$$\beta \leftarrow \arg \max_{\beta_a} (\beta_a \cdot b)$$

Algorithm 4: explore(b, ε, t)

```

if  $|\bar{V}(b) - \underline{V}(b)| > \varepsilon \gamma^{-t}$  then
  Heuristique de mise à jour :
   $a^* \leftarrow \arg \max_a \sum_{s \in S} R(s, a) b(s) +$ 
     $\gamma \sum_{o \in O} \mathbb{P}(o | b, a) \bar{V}(\tau(b, a, o))$ 
   $o^* \leftarrow \arg \max_o [\mathbb{P}(o | b, a^*) \text{excess}(\tau(b, a^*, o), t + 1)]$ 
  explore( $\tau(b, a^*, o^*), \varepsilon, t + 1$ )
   $\Gamma \leftarrow \Gamma \cup \text{backup}(\underline{V}, b)$ 
   $\Upsilon \leftarrow \Upsilon \cup (b, H[\bar{V}](b))$ 

```

2 POSG

On s'intéresse dans cette partie aux POSG, à leurs limitations et à certaines réalisations qui ont pu être faites dans des sous-classes de tels jeux.

2.1 Définition

Un jeu stochastique à information partielle ou POSG (partially observable stochastic game) est un jeu à deux joueurs où chaque joueur n'a accès qu'à son historique d'actions et observations. On suppose de plus qu'il ne bénéficie pas au cours du jeu de la valeur du gain obtenu, mais qu'il en bénéficie en fin de partie. Formellement, un POSG est un septuplet $(S, A_1, A_2, O_1, O_2, T, R)$ où :

- S est un ensemble fini représentant les états du jeu
- A_1, A_2 sont des ensembles finis d'actions de chaque joueur
- O_1, O_2 sont des ensembles finis d'observations de chaque joueur
- $T : S \times A_1 \times A_2 \rightarrow \Delta(S \times O_1 \times O_2)$ est le noyau de transition
- $R : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ est la fonction de gain

2.2 Limitations

Cette classe de jeux souffre principalement de deux limitations : Cette classe est assez large, elle regroupe des cas allant du simple SG à des jeux où les observations sont inexistantes et où les joueurs jouent totalement à l'aveugle. Pour pallier ceci, on considérera que les observations reçues par les joueurs sont "pertinentes" en un certain sens, c'est à dire apportent un renseignement en rapport avec la situation de jeu. De cette façon, on évite les cas où les observations sont seulement une source de bruit indépendant du jeu.

Cette classe n'assure pas une capacité de gain comparable pour tous les joueurs. Pour éviter cela, on supposera que la "qualité" des observations est similaire pour tous les joueurs. Par exemple, en fournissant des informations sur le jeu de même nature aux différents joueurs (par exemple leur éloignement relatif). Cette hypothèse permet d'éviter un trop grand déséquilibre entre les joueurs, cependant elle peut être désirée dans certains cas, comme pour les OS-POSG.

2.3 OS-POSG

On commence par étudier une sous-classe de POSG : les OS-POSG qui ont pour principale caractéristique d'offrir à un des joueurs une information complète sur l'état du jeu. On va alors chercher une stratégie pour le joueur ayant une information partielle. Cette hypothèse permet d'appliquer une version modifiée de l'algorithme HSVI pour POMDP, présenté dans [2], que j'ai implémenté dans `hsvi.py`.

Un jeu stochastique partiellement observable latéralisé ou OS-POSG (One-sided partially observable stochastic game) est un septuplet $(S, A_1, A_2, O, T, R, \gamma)$ où :

- S est l'ensemble fini des états
- A_1 et A_2 sont les ensembles finis des actions de joueurs 1 et 2 respectivement
- O est l'ensemble fini des observations
- $T : S \times A_1 \times A_2 \rightarrow \Delta(O \times S)$ est le noyau de probabilités de transition
- $R : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ et la fonction de gain du joueur 1.
- $\gamma \in]0, 1[$ est le facteur d'escompte

De plus, au départ le joueur connaît une distribution sur S , $b^{init} \in \Delta(S)$ correspondant à l'hypothèse initiale de présence sur chaque état.

Une méthode de résolution de tels jeux est proposée dans [2]. Cet algorithme : HSVI pour OS-POSG est assez similaire à HSVI pour POMDP, où l'on réalise un encadrement de la fonction objectif par un maximum d'applications linéaires, et par une application convexe définie par un ensemble points-valeurs. On représente l'état des deux joueurs lorsque du point de vue du premier joueur par des distributions de probabilités sur S . Ce qui permet de travailler sur des espaces de dimension finie.

L'algorithme HSVI pour OS-POSG nécessite pour son implémentation l'emploi de nombreuses fois d'algorithmes de résolution de problèmes linéaires, malgré sa relative longueur, l'implémentation se limite à choisir les bonnes structures et les fonctions de manipulation des données afférentes. Après lecture et présentation des preuves de correction et de terminaison, j'ai réalisé une implémentation (`hsvi.py`) de cet algorithme en langage Python, en utilisant essentiellement la bibliothèque Numpy. Je ne mets pas de pseudocode de cet algorithme par souci de place.

2.4 Bilan

Les algorithmes présentés jusqu'à présent utilisent le noyau de transition comme étant une donnée acquise. Cependant, ce n'est pas forcément le cas en pratique si l'on cherche à réaliser une stratégie dans un environnement réel ou si la taille de ce même noyau est trop importante. Les algorithmes présentés précédemment, dont HSVI pour OS-POSG souffrent de ce problème : dans [2], pour le cas d'un jeu de poursuite-évasion sur une grille de taille 8×3 , le jeu n'a pas pu être résolu en 10 heures le jeu possédant de l'ordre de 3000 états et 100000 transitions. C'est pourquoi je présente dans la partie suivante une méthode pour réduire la dimension des jeux, tout en conservant une borne sur la valeur d'équilibre, suivi d'une méthode d'apprentissage permettant de déterminer directement des stratégies à jouer, en s'inspirant des méthodes d'apprentissage par renforcement.

3 Réduction de la dimension du jeu

Je propose ici une méthode pour diminuer le nombre d'états d'un jeu tout en ayant un certain contrôle sur les valeurs d'équilibre des jeux. Une approche générale est d'identifier les états de jeu similaires en un certain sens. Soit $G = (S, A_1, A_2, O_1, O_2, T, R)$ un POSG et $\gamma \in]0, 1]$. Un des principaux problèmes lors de la résolution de jeux stochastiques est la complexité des calculs liée à la taille de l'ensemble S des états. Une première approche pour réduire la dimension du problème est d'identifier les états de jeu dont les caractéristiques sont similaires, puis de jouer similairement lorsque l'on se trouve (ou que l'on suppose se trouver) dans ces états. On fixe $\varepsilon > 0$, et une partition de l'ensemble S des états en classes $(C_i)_{i \in I}$ (notons $\sim$ la relation d'équivalence associée), telle que $|I|$ soit minimal et que :

$$\forall s \sim s' \quad \|R(s, \cdot, \cdot) - R(s', \cdot, \cdot)\|_\infty < \varepsilon \quad (5)$$

$$\begin{cases} \forall s \in S \quad \exists (\sigma_1^s, \sigma_2^s) \in \mathfrak{S}_{A_1} \times \mathfrak{S}_{A_2} \quad \forall s' \sim s \quad \forall v \sim v', \\ T(s, \sigma_1^s(\cdot), \sigma_2^s(\cdot))(v, \cdot, \cdot) = T(s', \sigma_1^{s'}(\cdot), \sigma_2^{s'}(\cdot))(v', \cdot, \cdot) \end{cases} \quad (6)$$

Notons $\tilde{S} = \{C_i \mid i \in I\}$ l'ensemble ainsi formé. On définit aussi une nouvelle fonction de gain $\tilde{R}$ et de transition $\tilde{T}$ pour tout $\tilde{s}, \tilde{s}_1, \tilde{s}_2 \in \tilde{S}$:

$$\tilde{R}(\tilde{s}, \cdot, \cdot) = \frac{1}{|\tilde{s}|} \sum_{s \in \tilde{s}} R(s, \cdot, \cdot) \quad (7)$$

$$\tilde{T}(\tilde{s}_1, \cdot, \cdot)(\tilde{s}_2, \cdot, \cdot) = \sum_{s_2 \in \tilde{s}_2} T(s_1, \sigma_1^{s_1}(\cdot), \sigma_2^{s_1}(\cdot))(s_2, \cdot, \cdot) \quad \text{pour } s_1 \in \tilde{s}_1 \text{ quelconque} \quad (8)$$

On a donc défini un nouveau jeu $\tilde{G} = (\tilde{S}, A_1, A_2, O_1, O_2, \tilde{T}, \tilde{R})$, "proche" en un certain sens du jeu initial, $\tilde{T}$ étant bien un noyau de probabilités de transition correspondant aux espaces nouvellement définis.

La projection canonique $S \xrightarrow{\rho} \tilde{S}$ et les définitions de $\tilde{T}$ et $\tilde{R}$ permettent de définir une opération de transformation du jeu réduisant sa dimension, c'est à dire son nombre d'états.

$$G \xrightarrow{\rho} \tilde{G}$$

Comme chaque joueur n'a accès qu'à ses observations et actions au cours du temps, on définit une stratégie π_i du joueur i comme étant une application de l'espace des historiques dans l'espace des distributions des actions du joueur.

$$\pi_i : (O_i \times A_i)^* \times O_i \rightarrow \Delta(A_i) \quad \forall i \in \{1, 2\} \quad (9)$$

Comme une stratégie π_i sur un tel jeu ne dépend que des historiques sur les ensembles O_i, A_i pour $i \in \{1, 2\}$ et que les fonctions σ_1, σ_2 sont bijectives, il existe une correspondance biunivoque entre les stratégies dans G et dans $\tilde{G} = \rho(G)$. On notera cette transformation $\pi \longrightarrow \tilde{\pi}$.

Définition 3.1. On définit les fonctions de valeur des jeux G et $\tilde{G}$ respectivement par :

$$V^* = \max_{\pi_1} \min_{\pi_2} \mathbb{E} \left[\text{Esc}_{\gamma, \pi_1, \pi_2}^G \right] \quad \tilde{V}^* = \max_{\tilde{\pi}_1} \min_{\tilde{\pi}_2} \mathbb{E} \left[\text{Esc}_{\gamma, \tilde{\pi}_1, \tilde{\pi}_2}^{\tilde{G}} \right]$$

Théorème 3.1. Sous les hypothèses (5), les valeurs d'équilibres des jeux vérifient

$$|V^* - \tilde{V}^*| \leq \frac{\varepsilon}{1 - \gamma} \quad (10)$$

Démonstration. Tout d'abord, fixons une stratégie π_1, π_2 sur G , comme elles ne dépendent que de l'historique, on peut définir les mêmes stratégies appliquées sur $\tilde{G}$, notons les $\tilde{\pi}_1, \tilde{\pi}_2$. Montrons que l'écart entre les valeurs de jeux obtenues pour ces stratégies est faible. Nous utiliserons les notation suivantes :

$$A_N = (A_1 \times A_2)^{\{0, \dots, N\}} \quad S_N = S^{\{0, \dots, N\}} \quad O_N = (O_1 \times O_2)^{\{0, \dots, N\}}$$

Par définition,

$$\mathbb{E} \left[\text{Esc}_{\gamma, \pi_1, \pi_2}^G \right] = \lim_{N \rightarrow \infty} \sum_{\substack{(s_t)_{t \in S_N} \\ (o_t)_{t \in O_N} \\ (a_t)_{t \in A_N}}} \sum_{t=0}^N \gamma^t R(s_t, a_t) \times \left(\prod_{t=0}^N \pi_t^1 \pi_t^2 T(s_t, a_t)(s_{t+1}, o_t) \right) = \lim_{N \rightarrow \infty} H^N \quad (11)$$

$$\mathbb{E} \left[\text{Esc}_{\gamma, \tilde{\pi}_1, \tilde{\pi}_2}^{\tilde{G}} \right] = \lim_{N \rightarrow \infty} \sum_{\substack{(\tilde{s}_t)_{t \in \tilde{S}_N} \\ (o_t)_{t \in O_N} \\ (a_t)_{t \in A_N}}} \sum_{t=0}^N \gamma^t \tilde{R}(\tilde{s}_t, a_t) \times \left(\prod_{t=0}^N \tilde{\pi}_t^1 \tilde{\pi}_t^2 \tilde{T}(\tilde{s}_t, a_t)(\tilde{s}_{t+1}, o_t) \right) = \lim_{N \rightarrow \infty} \tilde{H}^N \quad (12)$$

$$\begin{aligned} |\tilde{V}^* - V^*| &= \left| \max_{\pi_1} \min_{\pi_2} \mathbb{E} \left[\text{Esc}_{\gamma, \pi_1, \pi_2}^G \right] - \max_{\tilde{\pi}_1} \min_{\tilde{\pi}_2} \mathbb{E} \left[\text{Esc}_{\gamma, \tilde{\pi}_1, \tilde{\pi}_2}^{\tilde{G}} \right] \right| \\ &\leq \left| \lim_{N \rightarrow \infty} H^N - \lim_{N \rightarrow \infty} \tilde{H}^N \right| \\ &\leq \lim_{N \rightarrow \infty} |H^N - \tilde{H}^N| \end{aligned}$$

Comme

$$\sum_{(a_t)_{t \in A_N}} \prod_{t=0}^N \tilde{\pi}_t^1 \tilde{\pi}_t^2 \tilde{T}(\tilde{s}_t, a_t)(\tilde{s}_{t+1}, o_t) = \sum_{(s_t) \in (\tilde{s}_t)} \sum_{(a_t)_{t \in A_N}} \prod_{t=0}^N \pi_t^1 \pi_t^2 T(s_t, a_t)(s_{t+1}, o_t)$$

Il vient :

$$\begin{aligned} &|H^N - \tilde{H}^N| \\ &= \left| \sum_{\substack{(s_t)_{t \in S_N} \\ (o_t)_{t \in O_N} \\ (a_t)_{t \in A_N}}} \sum_{t=0}^N \gamma^t R(s_t, a_t) \times \left(\prod_{t=0}^N \pi_t^1 \pi_t^2 T(s_t, a_t)(s_{t+1}, o_t) \right) - \sum_{\substack{(\tilde{s}_t)_{t \in \tilde{S}_N} \\ (o_t)_{t \in O_N} \\ (a_t)_{t \in A_N}}} \sum_{t=0}^N \gamma^t \tilde{R}(\tilde{s}_t, a_t) \times \left(\prod_{t=0}^N \tilde{\pi}_t^1 \tilde{\pi}_t^2 \tilde{T}(\tilde{s}_t, a_t)(\tilde{s}_{t+1}, o_t) \right) \right| \\ &= \sum_{\substack{(\tilde{s}_t)_{t \in \tilde{S}_N} \\ (o_t)_{t \in O_N} \\ 0 \leq t \leq N}} \left[\left(\sum_{\substack{(s_t) \in (\tilde{s}_t) \\ (a_t)_{t \in A_N}}} \gamma^t R(s_t, a_t) \prod_{t=0}^N \pi_t^1 \pi_t^2 T(s_t, a_t)(s_{t+1}, o_t) \right) - \left(\gamma^t \tilde{R}(\tilde{s}_t, \cdot) \sum_{\substack{(s_t) \in (\tilde{s}_t) \\ (a_t)_{t \in A_N}}} \prod_{t=0}^N \tilde{\pi}_t^1 \tilde{\pi}_t^2 \tilde{T}(s_t, a_t)(s_{t+1}, o_t) \right) \right] \\ &= \sum_{\substack{(\tilde{s}_t)_{t \in \tilde{S}_N} \\ (o_t)_{t \in O_N} \\ 0 \leq t \leq N}} \sum_{(a_t)_{t \in A_N}} \sum_{(s_t) \in (\tilde{s}_t)} \gamma^t [R(s_t, a_t) - \tilde{R}(\tilde{s}_t, \cdot)] \times \mathbb{P}_{\text{chemin se réalise}} \\ &\leq \sum_{0 \leq t \leq N} \gamma^t \varepsilon \leq \left\lceil \frac{\varepsilon}{1 - \gamma} \right\rceil \quad (13) \end{aligned}$$

D'où le résultat.

□

La méthode de réduction de jeu précédente est utilisable pour réduire la dimension de l'espace de jeu. L'exemple suivant pourra être illustratif.

Exemple 3.1. On considère un jeu de poursuite proie-prédateur. Une proie et un prédateur se déplacent sur une grille rectangulaire de taille $n \times m$ recollée en un tore comme représenté en figure 1. On suppose que le prédateur gagne 100 points si il est sur la même case que la proie, qui elle perdra 100 points. Sinon, la proie marque 1 point et le prédateur en perd 1. Les deux agents peuvent se réplacer dans les quatres directions des cloisons de la grille. Les observations réalisées par les deux agents les informent lorsque leur adversaire est présent dans une des 8 cases adjacentes à la leur. Ce jeu possède $(nm)^2$ états. L'information qui va permettre de créer les classes est la position relative d'un agent par rapport à l'autre, que l'on peut même prendre à isométrie de l'espace près. Ceci forme un espace de dimension nm .

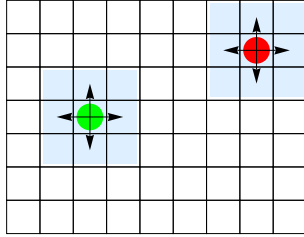


FIGURE 1 – Un exemple de grille de jeu, des zones d'observation et des actions jouables

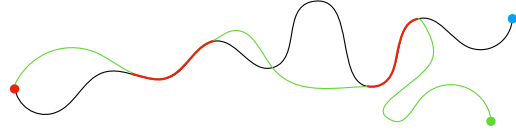


FIGURE 2 – Exemple de correspondance des chemins de l'espace $A_i \times O_i$ pour deux parties du joueur i

Bien que d'importantes diminutions de dimension de jeu puissent être obtenues, notamment lorsque le jeu possède des symétries, par exemple dans le cas des jeux de poursuite. Cette méthode ne peut pas s'appliquer lorsque des états ou actions similaires n'existent pas, auquel cas la partition sera un ensemble de singletons. On pourra s'intéresser à des méthodes pour déterminer de façon performante de telles partitions.

4 Méthode de jeu pour les POSG par proximité des historiques

4.1 Motivation et principe général

Dans le cas d'un POSG, et ne connaissant pas le noyau de probabilités de transition, on cherche à déterminer des stratégies efficaces contre certains types d'adversaires. Le principe est le suivant : les joueurs vont jouer une partie de durée déterminée, puis mettre à jour leurs stratégies avant de jouer la partie suivante. On se place dans un cas horizon fini.

La méthode que je propose est de reconnaître les séquences clefs dans les parties, puis de déterminer une stratégie efficace (c'est à dire jouer une action) à partir de ce moment. Pour ce faire, on va s'intéresser aux historiques d'actions-observations. On va aussi favoriser certaines actions peu jouées avec un paramètre d'exploration pour ne pas éliminer de bons coups qui n'ont pas été joués au départ.

On va supposer que les séquences pertinentes (que l'on appellera *séquences clefs* dans la suite) sont de longueur au plus K , c'est un paramètre que l'on fixe au départ. Et que toute partie se termine en temps fini T .

4.2 Reconnaissance de séquences clefs

Les moments décisifs du jeu sont les séquences de coups précédant un coup qui change de manière importante l'issue du jeu, ou le gain reçu. Considérons deux séquences $a_1 o_1, \dots, a_T o_T$ et $a'_1 o'_1, \dots, a'_T o'_T$ avec pour gains finaux R et R' comme représenté en figure 2, où les séquences en rouge sont considérées comme des séquences clefs.

Dans la suite, on notera E l'espace des séquences de taille K , $E = (O \times A)^K \times O$.

4.2.1 Distance sur l'espace des séquences

On a besoin d'évaluer la similitude entre les séquences, donc de définir une mesure de dissimilarité (voir [1]) sur l'espace des séquences de taille K . Fixons une telle mesure de dissimilarité et notons la d . Dès lors, en fixant $\varepsilon \geq 0$, on considère que deux séquences $s, s' \in E$ sont similaires si $d(s, s') \leq \varepsilon$.

On utilisera pour l'implémentation $\varepsilon = 0$ et la mesure de dissimilarité suivante : pour $x = (x_1, \dots, x_{2K+1}) \in A \cup O$, $y = (y_1, \dots, y_{2K+1}) \in A \cup O$,

$$d(x, y) = \text{Card}(\{i | x_i \neq y_i\})$$

4.2.2 Sélection des séquences clefs

Lorsque l'on compare les historiques de toutes les parties jouées, on veut pouvoir déterminer les séquences décisives. Et les mettre sous la forme de triplets (x, a, R) où $x \in E$ est la séquence clef, $a \in A$ est l'action ayant succédé à cette séquence et $R \in \mathbb{R}$ est le gain obtenu après avoir joué cette action. Pour faire cette sélection, plusieurs approches peuvent être choisies. On propose trois méthodes, mais de nombreuses autres approches peuvent être employées :

Séquence commune la plus en aval Si on suppose que le dernier coup joué est le plus déterminant, on peut employer cette méthode de sélection. On s'intéresse à tous les couples d'historiques de parties, étant donnés h_1 et h_2 deux tels historiques avec des suites de gains R_1 et R_2 ; on va chercher la séquence similaire d'indice de départ maximum dans h_1 , et de même dans h_2 . On a obtenu deux séquences clefs x et y avec des actions a_x^1, a_y^1 et a_x^2, a_y^2 et des gains respectifs $R_x^1, R_x^2, R_y^1, R_y^2$ comme représenté en figure 3. Cela forme quatre triplets séquence-action-gain : (x, a_x^1, R_x^1) , (x, a_x^2, R_x^2) , (y, a_y^1, R_y^1) , (y, a_y^2, R_y^2) .

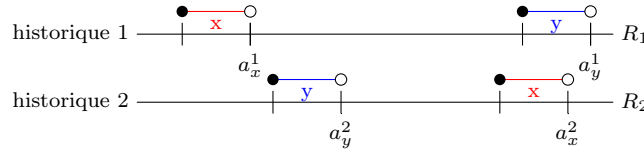


FIGURE 3 – Séquences similaires pour deux historiques

On peut utiliser une structure d'arbre suffixe pour stocker les historiques et réaliser les recherches de plus lointaines séquences communes. Pour ce faire, on peut utiliser une version arrangée de l'algorithme en-ligne de Ukkonen [5], garantissant une complexité temporelle totale linéaire à chaque nouvelle séquence ajoutée.

Séquences ayant engendrés les gains les plus extrêmes L'objectif ici est de sélectionner les séquences ayant donné les gains les plus significatifs. Une approche possible est de se fixer une proportion, disons 25% puis de choisir toutes les triplets (x, a, R) pour lesquels le gain R est parmi les 25% des gains possibles les plus élevés, et des plus faibles.

Toutes séquences On considère ici que tous les moments de l'historique sont importants et comportent une information pertinente. On sélectionne dans ce cas toutes les séquences. de tous les historiques passés pour constituer l'ensemble des séquences clefs. C'est la méthode employée dans l'implémentation, pour des raisons de simplicité, et d'absence d'hypothèse supplémentaire.

4.3 Optimisation de la stratégie de jeu sur l'espace des séquences clefs

On cherche à présent à établir une stratégie de jeu sur l'espace des séquences clefs E . La méthode proposée consiste étant donné une séquence $s \in E$ à jouer une action selon une distribution de probabilités. Cette distribution étant déterminée par l'ensemble des séquences clefs précédemment déterminé.

La principale idée ici est de dire que pour des séquences proches au sens de la mesure de dissimilarité, les bonnes actions jouées doivent être les mêmes. C'est le rôle de la contribution illustrée en figure 4.

L'approche employée est de se fixer un facteur d'exploration $\lambda \in [0, 1]$, de déterminer deux distributions sur A pour chaque $s \in E$: la première favorisant d'autant plus les actions qu'elles ont peu été jouées, et la seconde favorisant les actions ayant apporté les meilleurs gains. La stratégie jouée est la moyenne pondérée par λ de ces deux distributions.

Définition 4.1. Pour d une mesure de dissimilarité sur E , une contribution est une application $C : E \times E \rightarrow \mathbb{R}_+$, telle que $C(s, s') = f(d(s, s'))$ où f est monotone strictement décroissante et tendant vers 0 en $+\infty$.

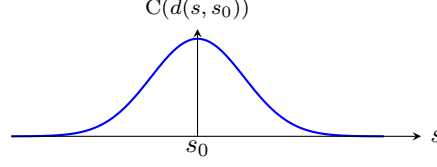


FIGURE 4 – représentation de la contribution selon l'éloignement

Pour l'implémentation, on utilisera la contribution suivante : $\forall s, s' \in E \quad C(s, s') = \delta_{s, s'}$. où δ est le symbole de Kronecker.

Définition 4.2. Une stratégie est une application $\pi : E \rightarrow \Delta(A)$

Supposons que l'on ait joué un certain nombre de parties, et que l'on aie sélectionné les séquences clefs $H = \{(s_i, a_i, r_i) \mid 1 \leq i \leq M\}$. On s'intéresse à la stratégie que l'on va accorder aux séquences de E .

On définit la distribution d'exploration ρ , et la distribution d'optimisation ν dépendant d'une séquence $s \in E$ et pour une action $a \in A$ à partir de $\tilde{\rho}$ et $\tilde{\nu}$ définis comme suit, en notant $s + a$ la séquence concaténée de s et a :

$$\tilde{\rho}[s + a] = \frac{\text{Card}(\{i \mid s_i = s \ a_i \neq a\})}{\text{Card}(\{i \mid s_i = s\})} \quad (14)$$

$$\tilde{\nu}[s + a] = \sum_{\substack{r_i \leq 0 \\ a_i = a}} C(s_i, s) \exp(r_i) + \sum_{\substack{r_i < 0 \\ a_i \neq a}} C(s_i, s) \exp(-r_i) \quad (15)$$

Dans le cas que l'on implémente, cela donne :

$$\tilde{\nu}[s + a] = \sum_{\substack{s_i = s \\ r_i \leq 0 \\ a_i = a}} \exp(r_i) + \sum_{\substack{s_i = s \\ r_i < 0 \\ a_i \neq a}} \exp(-r_i) \quad (16)$$

On définit alors ρ et ν comme étant les valeurs normalisées de $\tilde{\rho}$ et $\tilde{\nu}$, de manière à obtenir des noyaux de probabilités.

$$\rho[s + a] = \frac{\tilde{\rho}[s + a]}{\sum_{b \in A} \tilde{\rho}[s + b]} \quad (17)$$

$$\nu[s + a] = \frac{\tilde{\nu}[s + a]}{\sum_{b \in A} \tilde{\nu}[s + b]} \quad (18)$$

ρ est un noyau de probabilités de transition sur A servant à pondérer le degré d'exploration de la stratégie en chaque action, en favorisant les actions qui ont peu été jouées jusqu'à présent. ν est un noyau de probabilités de transition sur A correspondant à l'optimisation voulue. Les actions ayant apporté les plus grands gains sont jouées avec plus grande probabilité.

On définit à présent la stratégie $\pi[s] \in \Delta(A)$ pour $s \in E$:

$$\boxed{\pi[s](a) = \lambda \rho[s + a] + (1 - \lambda) \nu[s + a]} \quad (19)$$

Pour l'implémentation, on note une telle stratégie obtenue π_H . Le choix de λ se fait via une calibration préalable.

Remarque 4.1. On ne prend pas en compte le caractère récent de la séquence clef obtenue, c'est toutefois une information importante si l'on veut réaliser une méthode permettant d'adapter rapidement la stratégie à celle de l'adversaire.

On remarque que lorsque les observations révèlent exactement l'état du jeu, la méthode présentée dans le cas $K = 0$ correspond à un apprentissage par renforcement dans le cas d'un SG à deux joueurs.

4.4 Pseudocode et implémentation

L'algorithme 5 simule des parties de durée M un nombre $N = 2^P$ de fois face à un adversaire de stratégie π_2 . On décompose la suite des parties simulées en paliers de taille doublant à chaque fois, au bout desquels on réalise un changement de stratégie avec la méthode présentée précédemment. Ceci permet aux bonnes décisions prises

Algorithm 5: Methode(π_2, N, M)

```
 $H \leftarrow [];$ 
for  $k = 1$  to  $P$  do
     $S \leftarrow [];$ 
    for  $n = 2^{k-1}$  to  $2^k - 1$  do
         $h = \text{Simulation}(\pi_H, \pi_2, M)$  /* On simule une partie avec ces stratégies */;
         $S = S + h;$ 
     $H \leftarrow H + S;$ 
return  $\pi_H;$ 
```

par la dernière stratégie utilisée de dépasser en importance celles des anciennes stratégies, sans pour autant les négliger totalement.

La complexité en temps et en espace de cet algorithme est au pire en $O(NM)$.

J'ai implémenté cet algorithme en Python, dans `methode.py`. Les expériences réalisées ont été faites avec deux stratégies stationnaires pour le joueur 2 : π_2 (c'est à dire dépendant uniquement de la dernière observation faite) et une avec un adversaire suivant la même méthode d'apprentissage que le joueur 1.

4.5 Expérimentation

4.5.1 Description des jeux et des stratégies

On évalue la méthode sur deux jeux de poursuite analogues en dimension 1 et 2. On se place dans le cas d'un jeu de poursuite. Il y a une proie et un chasseur. Les observations indiquent à chaque joueur si l'adversaire est proche de lui. Lorsque les deux joueurs sont sur des cases différentes le gain du chasseur est -1. Lorsque ils sont sur la même case, le gain est 8 en dimension 1 et 40 en dimension 2 (ce choix de valeurs sert à visualiser confortablement les résultats). On cherche à optimiser la stratégie du chasseur. La proie suit une stratégie π_2 . On va réaliser des essais avec trois types de stratégies pour la proie : une stratégie "lapin", qui reste immobile lorsque il y a une observation à distance 0 du chasseur, une stratégie "gazelle" qui fuit lorsque il y a une observation du chasseur et enfin un jeu où on va apprendre chacune des deux stratégies avec la méthode présentée. Pour les deux premières stratégies de la proie, on joue une action selon l'unique dépendance de l'observation précédente. On évalue la performance de notre stratégie évolutionnaire π_1 face à ces différents adversaires.

1. Dans le cas de la dimension 2 (analogue à ce qui est représenté par la tableau 1), les deux joueurs se déplacent sur un carré de côté $n = 10$ recollé en un tore. ils peuvent se déplacer à droite ($\rightarrow$), à gauche ($\leftarrow$), en haut ($\uparrow$), en bas ($\downarrow$) ou rester sur place ($\cdot$). Ils ont une vision à distance 2 indiquant la distance entre les deux agents. Les stratégies adversaires "lapin" et "gazelle" dépendent des noyaux décrits en tableau 1 et tableau 2 fournissant une distribution de probabilités sur l'espace d'actions en fonction de la distance observée entre les adversaires.

	$\leftarrow$	$\cdot$	$\rightarrow$	$\downarrow$	$\uparrow$
0	1/25	21/25	1/25	1/25	1/25
1	3/25	13/25	3/25	3/25	3/25
2	5/25	5/25	5/25	5/25	5/25
3	5/25	5/25	5/25	5/25	5/25

TABLE 1 – Noyau "lapin" en dimension 2

	$\leftarrow$	$\cdot$	$\rightarrow$	$\downarrow$	$\uparrow$
0	6/25	1/25	6/25	6/25	6/25
1	3/25	13/25	3/25	3/25	3/25
2	5/25	5/25	5/25	5/25	5/25
3	5/25	5/25	5/25	5/25	5/25

TABLE 2 – Noyau "gazelle" en dimension 2

2. Dans le cas de la dimension 1, les deux joueurs se déplacent sur une bande refermée de longueur $n = 10$. ils peuvent se déplacer à droite, à gauche ou rester sur place. Ils ont une vision à distance 1 correspondant aux observations. Leurs noyaux de transitions sont les tableaux 3 et 4.

	$\leftarrow$	$\cdot$	$\rightarrow$
distance ≥ 1	3/9	3/9	3/9
distance ≤ 1	1/9	7/9	1/9

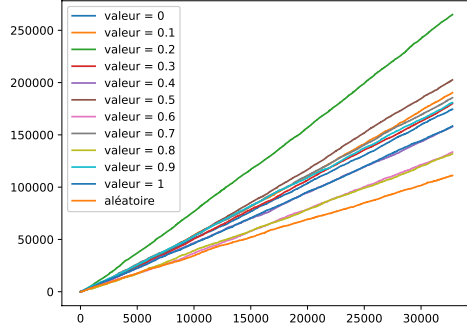
TABLE 3 – Noyau "lapin" en dimension 1

	$\leftarrow$	$\cdot$	$\rightarrow$
distance ≥ 1	3/9	3/9	3/9
distance ≤ 1	4/9	1/9	4/9

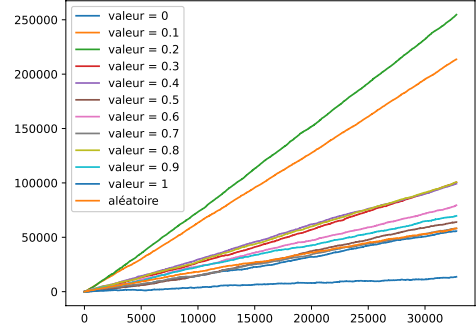
TABLE 4 – Noyau "gazelle" en dimension 1

4.5.2 Paramétrage du facteur d'exploration λ

On réalise plusieurs essais avec des valeurs de λ différentes et on sélectionne celle qui donne les meilleures performances de croissance.

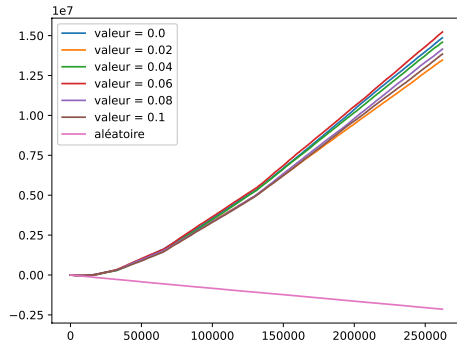


(a) Calibrage lapin

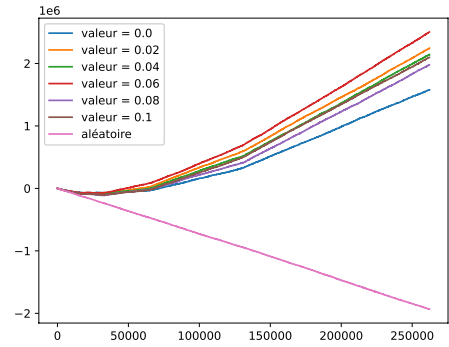


(b) Calibrage gazelle

FIGURE 5 – Le cas dimension 1



(a) Calibrage lapin



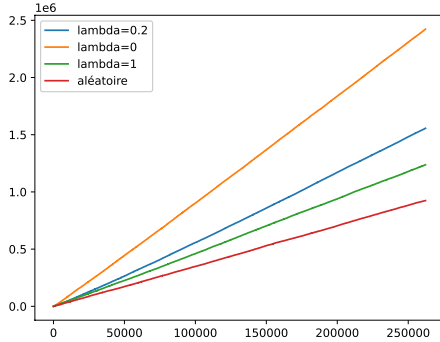
(b) Calibrage gazelle

FIGURE 6 – Le cas dimension 2

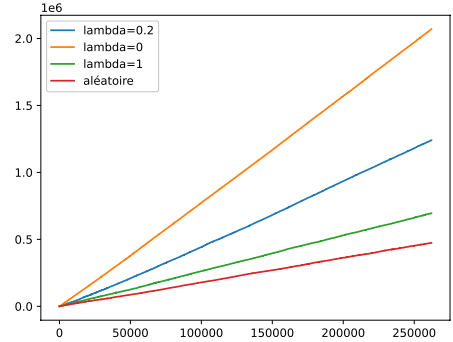
On choisit donc en dimension 1 $\lambda = 0.2$, et pour la dimension 2 $\lambda = 0.06$. Ce choix n'est pas idéal, mais permet d'avoir une vitesse d'apprentissage et des performances meilleures.

4.5.3 Résultats

On présente en figures 7 et 8 les sommes cumulées des gains du joueur chasseur au cours des parties face aux adversaires "lapin" et "gazelle", ceci pour différentes valeurs d'exploration.

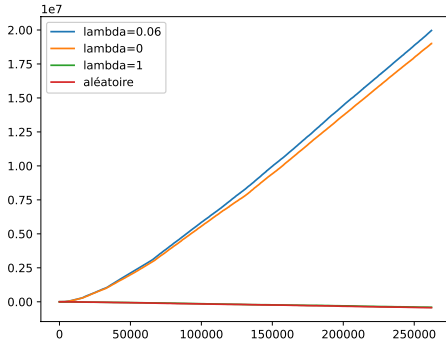


(a) Adversaire lapin

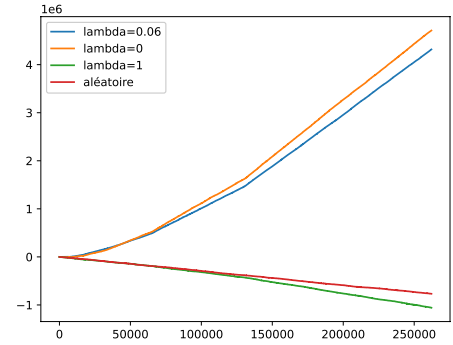


(b) Adversaire gazelle

FIGURE 7 – Somme cumulée de gains obtenus au cours des parties, en dimension 1



(a) Adversaire lapin



(b) Adversaire gazelle

FIGURE 8 – Somme cumulée de gains obtenus au cours des parties, en dimension 2

4.5.4 Analyse

On constate comme attendu un phénomène d'amélioration de la performance du joueur "chasseur" face à un adversaire stationnaire, ce qui se traduit par des pentes de plus en plus importantes à chaque nouveau palier (visible en figure 8). On constate qu'en dimension 1 l'apprentissage se fait très vite, lors des premières itérations de l'algorithme, pour ensuite maintenir une stratégie aux performances constantes (figure 7) se traduisant par des droites. On observe également que l'augmentation des pentes tend vers 0 à mesure que des plateaux d'amélioration sont passés.

4.6 Perspective et conclusion

Au bilan, la méthode présentée améliore les performances de jeu au cours des phases d'apprentissage, et ce d'autant mieux que le nombre de paliers. Donc le nombre de parties jouées est grand.

On remarque que dans le cas où les observations sont les états, on réalise pour $K=0$ une optimisation sur les états, ce qui ramène la méthode présentée à une méthode d'apprentissage par renforcement dans le cas SG, notamment, lorsque l'adversaire est stationnaire, il s'agit d'apprentissage par renforcement dans un MDP.

La méthode présentée nécessite un paramétrage qui peut se révéler long et fastidieux pour déterminer la bonne constante d'exploration λ ; de plus comme souvent pour un algorithme d'apprentissage par renforcement, il y a nécessité de phases d'apprentissage longues pour obtenir des résultats significatifs. Elle a toutefois l'intérêt de

ne pas nécessiter le stockage en mémoire du noyau de transition du jeu, et de s'adapter à une éventuelle nouvelle stratégie qui lui est opposée à tout instant.

En perspective, on pourra rechercher une preuve de l'augmentation de la pente à chaque palier (ce que l'on peut constater empiriquement), et une évaluation de la performance de méthodes s'adaptant à un adversaire, ou si cette méthode ou une méthode autre par apprentissage permet de s'approcher d'un équilibre de Nash du jeu. On pourra s'intéresser de manière plus générale à des méthodes travaillant uniquement sur les espaces d'historiques pour trouver des stratégies efficaces et pouvant s'adapter à un changement d'adversaire.

Références

- [1] Michel-Marie Deza Elena Deza. *Encyclopedia of distances*. Springer, 2009.
- [2] Karel Horák, Branislav Bošanský, Vojtěch Kovařík, and Christopher Kiekintveld. Solving zero-sum one-sided partially observable stochastic games. *Artificial Intelligence*, 316 :103838, 2023.
- [3] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10) :1095–1100, 1953.
- [4] Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. *arXiv preprint arXiv :1207.4166*, 2012.
- [5] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3) :249–260, 1995.