Rapport de projet Agents

Etude de la coopération multi-agents en apprentissage par renforcement

- Encadré par :

  Mr. HUTZLER Guillaume

- Réalisé par :

  RAMOUL Samy Rayan : *raysamram@gmail.com*

04/05/2020

Table des matières

Table des figures

Introduction

Dans la société humaine, l'exploitation de stratégies de coopération par la communication entre individus permet la réalisation des tâches plus complexes ordinairement irréalisables ou encore de les executer plus rapidement. De tels mécanismes pourraient trouver leur intérét dans un contexte de systèmes multi-agents.

De nos jours le domaine de l'intelligence artificielle est en pleine essort, et les méthodes de coopération entre agents ont pourtant encore très peu été explorées et étudiées dans certaines branches comme l'apprentissage par renforcement.

Ce projet implémente le papier *Multi Agent Reinforcement Learning Independent vs Cooperative Agents*, qui, en se basant sur la technique d'apprentissage par renforcement et plus précisement l'algorithme de Q-Learning, étudie les méthodes de communication et surtout la coopération possibles dans un système multi-agents de par différents types de communication entre des agents intelligents, ainsi qu'en étudiant l'impact sur l'efficacité dans la réalisation de taches différentes et sur la performance du système.

Chapitre 1

Etat de l'art

1.1 Apprentissage par Renforcement

1.1.1 Description

L'apprentissage par renforcement est une méthode d'apprentissage automatique se concentrant sur le principe d'agents. Dans cette méthode contrairement au reste des techniques d'apprentissage, il n y'a pas besoin de connaissance humaine préalable, ainsi le but étant que l'agent apprenne de par ses propre expérimentations dans un modèle sous forme de processus Markovien, ou la tâche à accomplir pour chaque agent est de maximiser sa récompense à long terme.
Pour se faire il dispose d'un tableau d'association entre les paires d'États et Actions vers une espérance de récompense qu'il met à jour au fur et à mesure de son expérimentation, et depuis lequel il sélectionne soit la valeur maximale (meilleure action possible ou "greedy") soit une action aléatoire (pour effectuer de l'exploration et tester des possibilités).

1.1.2 Elements de l'Apprentissage par Renforcement

1.1.2.1 Environnement

Un environnement est l'espace dans lequel sont deployés les agents, on peut considérer que l'on a un modèle de l'environnement lorsqu'avec la possession d'un état *t*, et d'une action, on peut prédire l'état suivant.

1.1.2.2 Ensemble d'Actions

Représente toutes les actions que l'agent peut effectuer à un moment donné, par exemple pour un jeu d'echecs cela serait représenté par les déplacements autorisés pour chaque pion du joueur.

1.1.2.3 Perception ou Etat

Est un sous-ensemble extrait de l'environnement, c'est cela que voit l'agent à un moment *t* et l'information sur laquelle il se basera pour effectuer son prochain choix.

1.1.2.4 Récompense

Valeur pouvant être négative ou positive et servant à la convergence du modèle en le poussant à la maximiser sur le long terme.

1.1.2.5 Episode

Un épisode est un ensemble contenant : une série de perceptions de l'agent, ses actions choisies, et les récompenses obtenues le tout l'ayant mené vers une condition de terminaison (attraper une proie par exemple).

1.1.2.6 Fonction de valeur

Ou "Value Function" en Anglais, cette fonction représente la valeur attribuée à une paire Etat/Action, représentant ainsi l'espérance de récompense à obtenir en effectuant une action à un état donné elle est représentée ainsi $Q(état, action)$.

1.1.2.7 Q-Table

Afin de stocker l'ensemble des connaissances des associations Etats/Actions (ou valeurs de Q) on utilise une matrice que l'on appel "Q-Table" ou chaque ligne est un des états connus. Les colonnes représentent toutes les actions possibles, la valeur d'une de ces cases par exemple $(etat, action)$ représente la valeur associée à la paire état et action (ou encore l'espérance de récompense)

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

FIGURE 1.1: Q-Table

Dans la figure 1.1 on observe la configuration de cette matrice, ainsi que le fait qu'elle soit initalisée à des zéros puis est mise à jour de manière à converger vers une représentation plus précise de la fonction de valeur.

1.1.2.8 Fonction de Mise à jour

Afin de mettre à jour cette Q-Table on fait appel à l'équation de Bellman comme suit :

$$Q^{nouveau}(etat_t, action_t) := Q(etat_t, action_t) + \beta(récompense_t + \gamma * maxQ(etat_{t+1}, actions) - Q(etat_t, action_t)) \tag{1.1}$$

Dans l'équation 1.1 on peut observer qu'après chaque action l'agent met à jour la fonction de valeur en prenant en compte son ancienne valeur et en y ajoutant selon une proportion β la nouvelle récompense (variable à ajuster que l'on peut considérer comme un "taux d'apprentissage") ainsi qu'avec la proporition α ce qu'on appel la différence temporelle ou "temporal difference" en Anglais, cette méthode se basant sur le fait que l'on connaisse le modèle (que l'on puisse connaitre l'état suivant à partir d'un état et de l'action choisie) et permet ainsi d'apprendre la valeur associée en se basant sur cet état futur, on initialise toutes ces valeurs à 0.

A cette méthode on peut vouloir ajouter de l'aléatoire afin de permettre l'exploration c'est pour cela qu'on ajoute deux paramètres : ε et le *decay rate*, à chaque état un agent fait un jet aléatoire si celui-ci est inférieur à epsilon il effectue de l'exploration (en choisissant une action au hasard) sinon il choisit celle ayant la meilleure valeur. Le decay rate lui viendra se soustraire d'epsilon à la fin de chaque episode laissant de moins en moins place à l'exploration. Cette méthode est appelée "ε-Greedy".

1.1.2.9 Mémoire

Chaque agent garde en mémoire l'ensemble des épisodes par lesquels il est passé. lui permettant plus tard d'en apprendre ou de rejouer la mémoire d'un autre agent afin d'en tirer de l'information.

1.1.3 Algorithme

L'algorithme d'apprentissage par renforcement peut se résumer comme suit :

1. Initialiser les paramètres gamma, les récompenses et l'environnement.
2. Initialiser la Q-Table à des zéros.
3. Pour chaque épisode :
   (a) Choisir l'état initial.
   (b) Tant qu'un état final n'est pas atteint, faire :
      — Sélectionner une des actions possibles depuis l'état actuel.
      — En utilisant cette action considérer l'état d'après.
      — Récuperer la valeur maximale de Q pour ce prochain état sur l'ensemble des actions possibles.
      — Mettre à jour la Q-Table.
      — Définir le prochain état comme l'état actuel.
   (c) Fin du Faire

4. Fin

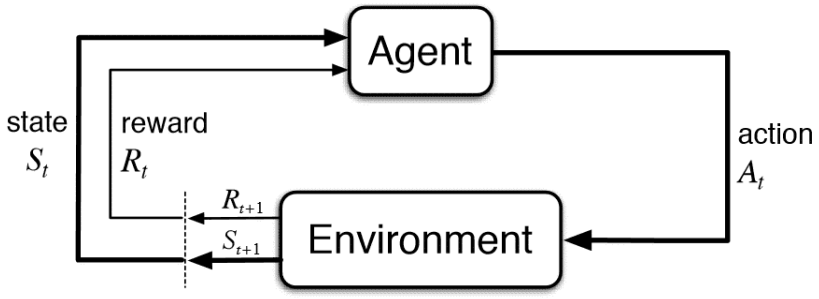


FIGURE 1.2: Apprentissage par Renforcement

La figure 1.2, représente ce processus d'apprentissage, ses boucles, et la dépendance existante entre les différents aspects de l'environnement et de l'agent.

Chapitre 2

Protocole Experimental

2.1 Apprentissage par renforcement et Communication

L'ensemble des fonctions nécessaires à l'apprentissage par renforcement ainsi que la communication a été représentée et codée sous une interface python permettant plus de flexibilité et de possibiltiés quant aux aspects abordés par le papier (principalement les calculs matriciels à faire) en ne faisant appel qu'à la librairie *numpy* pour les calculs matriciels. Le paramètrage et l'environnement lui est représenté sous Netlogo en utilisant l'extension "py".

La problématique abordée est celle de la chasse entre chat et souris, les chats étant les agents intelligents et devant apprendre à coopérer pour attraper des souris qui elles se déplacent de manière aléatoire.

2.2 Environnement d'Experimentation

L'environnement est un espace toroïdal sous forme d'une grille de 11 par 11, donc enveloppée verticalement et horizontalement.

FIGURE 2.1: Environnement

Tout comme on peut l'observer dans la figure 2.1 la perception d'un agent à un instant t est limité par un champ de vision qui est un paramètre ajustable (et important à la performance de l'agent), à partir de ce champ de vision on calcule une paire (x_{proie}, y_{proie}) représentant la distance du chasseur vers la proie la plus proche (il ne verra donc qu'une unique proie même si il y'en a plus dans son champ de vision) s'il n'y en a pas il a comme état 0. On remarque aussi la condition d'arrêt ou de fin d'épisode qui est le fait que le chat se retrouve dans la zone de capture (qui par exemple ici est définie comme étant le fait de se retrouver sur la même case que la proie).

L'ensemble de ses actions possibles sont des déplacements dans les 4 directions : Haut, Bas, Gauche et Droite, ainsi que ne pas bouger. La récompense est définie à -0.1 tant que la proie n'est pas attrapée et à 1 autrement.

2.3 Mesure d'évaluation

On définit comme mesure d'efficacité du modèle le nombre d'étapes (ou itérations) nécessaires en moyenne au chasseur pour atteindre sa proie et donc ainsi atteindre la fin de l'épisode. Celle-ci peut être analysée sous 2 angles différents : d'une part la vitesse (ou nombre d'épisodes nécessaires) que prendra l'agent avant de converger vers un nombre stable d'itérations représentant ainsi la vitesse ou qualité de l'apprentissage, d'une autre la moyenne finale aprés convergence définissant l'efficacité finale de l'agent.

Cinq exécutions différentes sont réalisées pour chacun des scénarios, les résultats sont recoltés au travers du module de plot de Netlogo, puis moyennés afin d'être affiché et comparé aux autres méthodes grâce au module matplotlib de python.

2.4 Scénarios

Plusieurs scénarios possibles de communication ou coopération entre agents ont été développées par l'auteur du papier afin d'en analyser les performances.

2.4.1 Scénario Zéro : Agents Indépendants

Avant de passer à l'étape de communication des tests sont effectués sur des agents indépendants afin de déterminer quel est le paramètrage optimal pour ces derniers que ce soit au niveau des paramètres de l'apprentissage par renforcement (Beta, Gamma..ect), mais aussi par rapport à leurs champs de vision.

2.4.2 Premier Scénario : Partage de sensation

Pour ce cas de figure un agent "Eclaireur" a été ajouté dont le seul but est d'envoyer ses observations ainsi que ses actions au chasseur et qui lui ne peut chasser la proie et se déplace aléatoirement.

Dans ce scénario l'état que verra le chasseur sera exprimé ainsi : si il a une proie dans son champ de vision, c'est cette information là qui est utilisée comme état pour lui (comme dans la représentation normale) , sinon il utilise la distance entre lui et la proie que pourrait voir l'éclaireur (et ce calculé à partir du fait que le chasseur connait sa distance avec l'éclaireur et que ce dernier lui communique sa vision).

Le paramètre important à ajuster ici étant le périmètre de vision de l'éclaireur.

2.4.3 Second Scénario : Partage de politique de choix ou d'épisodes

Pour ce scénario le type de simulation choisi est avec 2 chasseurs et 1 proie afin d'étudier des interactions possibles entre les politiques des agents intelligents que sont les chasseurs et ce au travers de plusieurs méthodes.

2.4.3.1 Premier cas : Partage de Q-Table

Les 2 chasseurs partagent la même Q-Table et mettent donc à jour conjointement ses valeurs.

2.4.3.2 Deuxième cas : Synchronisation

Les 2 chasseurs à une fréquence (étant le paramètre clé ici) d'épisodes régulière moyennent leurs tableaux, les différentes valeurs testées étant chaque 20, 50 et 100 épisodes.

2.4.3.3 Troisième cas : Partage d'expériences

Le principe étant ici qu'un agent réussissant à attraper une proie puisse envoyer sa mémoire d'épisodes à un autre agent afin qu'il puisse les rejouer et mettre à jour ainsi sa propre Q-Table.

D'abord cela sera testé sur deux agents chasseurs classiques, puis plus tard avec un agent normal et un agent "expert" dont les mouvements sont codés au préalable afin de se déplacer dans la direction de la proie.

2.4.4 Troisième Scénario : Tâches jointes

Dans ce dernier scénario, afin de capturer une proie il faut que les 2 chasseurs soient soi sur la même case que la proie ou dans une des cases adjacentes (comme illustré dans la figure 2.2) .

Dans ce cas de figure trois types d'interactions sont comparés :

— Les chasseurs sont indépendants (comme dans le cas initial donc ils ne voient que la proie la plus proche) .

— Les chasseurs observent passivement (ou chaque chasseur connaît la distance le séparant de son associé).

— Enfin un modèle ou les deux chasseurs partagent activement leurs perceptions (distance les séparant ainsi que la distance de chacun vers sa proie la plus proche).

Et ce sur 2 initialisations différentes :

— 2 Chasseurs et 1 proie.

— 2 Chasseurs et 2 proies.

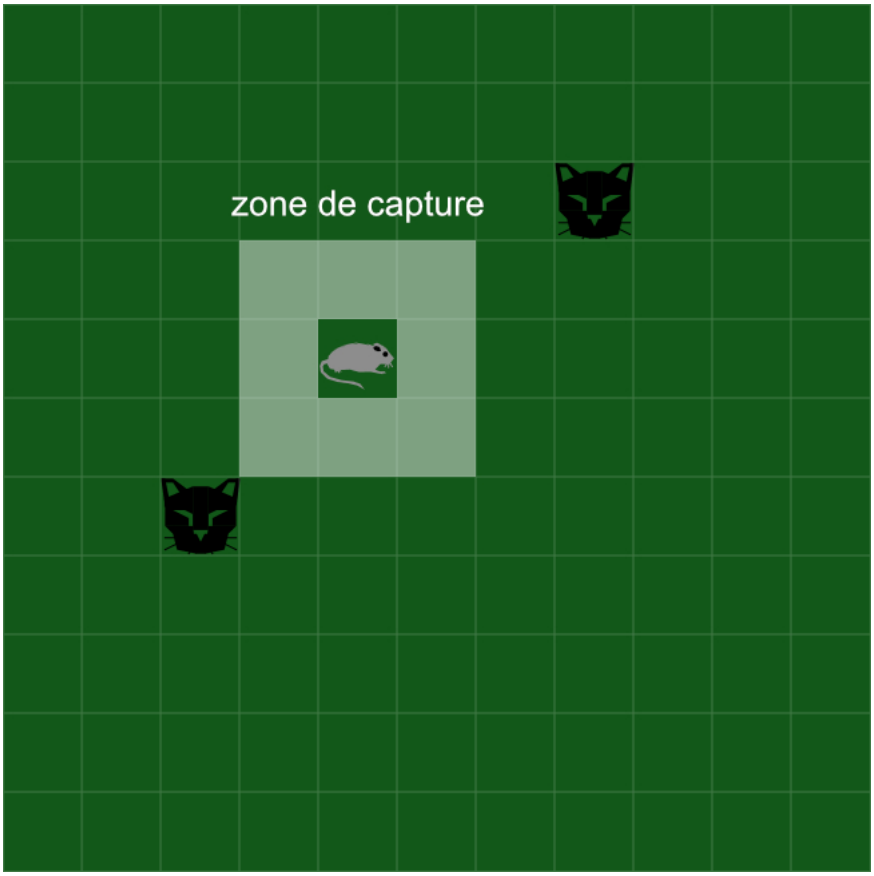


FIGURE 2.2: Perception et zone de capture

Ce modèle fût le plus complexe a implémenter de par le manque d'information (plus particuliérement de détails techniques) dans le papier car même si l'état que recevait à ce moment un agent était assez explcité par l'auteur, la manière dont cela été mise à jour durant l'execution ne l'était pas. L'agent à un instant donné recevait en état : sa percepetion, la perception de ses collègues et les distances les séparant. Cependant pour mettre à jour la Q-Table il faut avoir l'état *t*, l'action prise, puis l'état *t+1*, la question étant l'état *t+1* pour lui seulement ou faut-il aussi que ses coéquipiers refasse une analyse de leur entourage pour envoyer une nouvelle perception ? Plusieurs cas de figures ont été testés afin de trouver celui qui permettait un apprentissage correct ou l'état *t+1* ne met à jour que la perception de l'agent concerné (pour effectuer la mise à jour de la Q-Table) sans le faire pour les autres à ce moment.

Chapitre 3

Résultats

3.1 Résultats par Scénarios

3.1.1 Agents indépendants

Plusieurs valeurs de Beta et Gamma ont été testés durant mon expérimentation (allant de 0.5 à 1) mais les valeurs ayant donné les meilleurs résultats étaient $\beta = 0.8$ et $\gamma = 0.75$ (contrairement à $\beta = 0.8$ et $\gamma = 0.9$ precisés dans le papier, cependant les résultats au delà de 0.7 pour les deux paramètres étant très similaires cela n'a pas impacté grandement l'apprentissage.

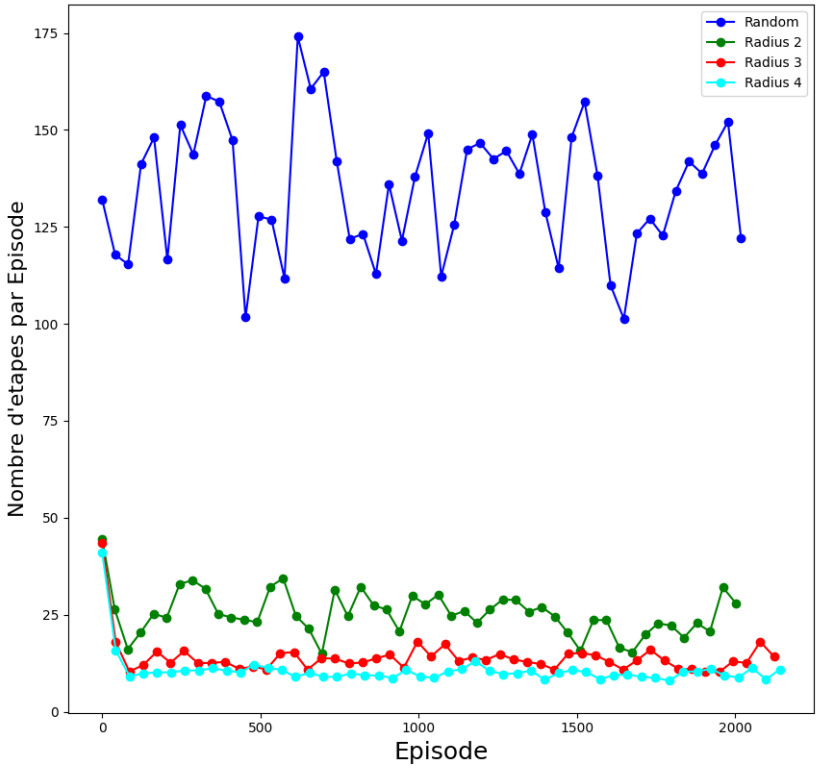
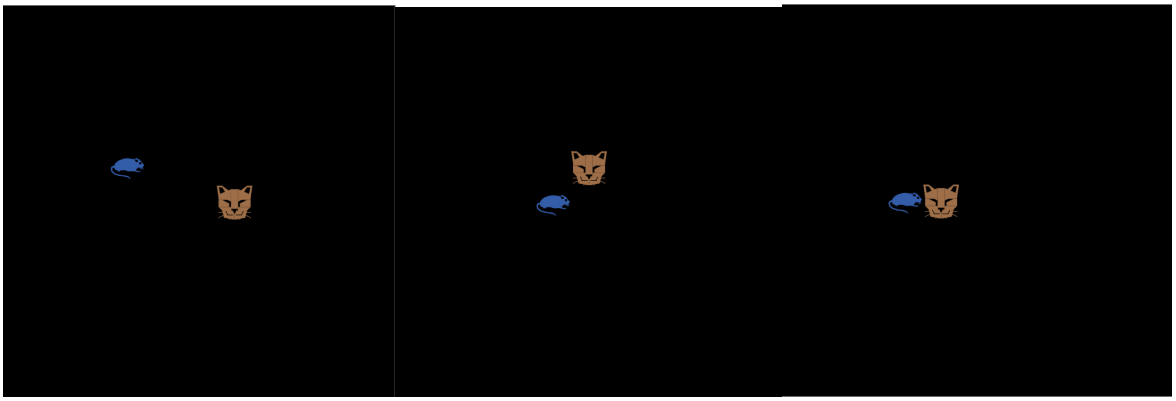


FIGURE 3.1: Chasse par rayon

On observe dans la figure 3.1 une très nette différence de performance entre un agent aléatoire et des agents intelligents prouvant ainsi d'une part que l'algorithme d'apprentissage converge vers une efficacité dans la réalisation de la tache de capture, d'une autre que plus le périmetre de vision d'un agent est grand plus petit est le nombre moyen d'étapes nécessaires afin d'achever un épisode. Ce paramètre est donc très important à l'effiacité d'un agent.



(a) Etape 1 (b) Etape 2 (c) Etape 3

FIGURE 3.2: Exemple d'exécution

Dans la figure 3.2 on peut observer la suite de comportement qu'aura un agent intelligent et qu'aura pu faire émerger le processus d'apprentissage. Ayant aperçu une proie dans son champ de vision il se dirige au fur et à mesure dans la direction de celle-ci. C'est un comportement qu'il a pu donc découvrir par lui même sans avoir été explicitement codé pour. Un autre comportement intéréssant à noter aussi est le cas ou il n'aperçoit aucune proie. Là, il a appris que la meilleure solution pour lui était tout simplement de parcourir l'ensemble de la grille de bas en haut (le sens pouvant changer d'une initialisation à l'autre) jusqu'à tomber sur celle-ci.

3.1.2 Premier Scénario : Partage de sensation

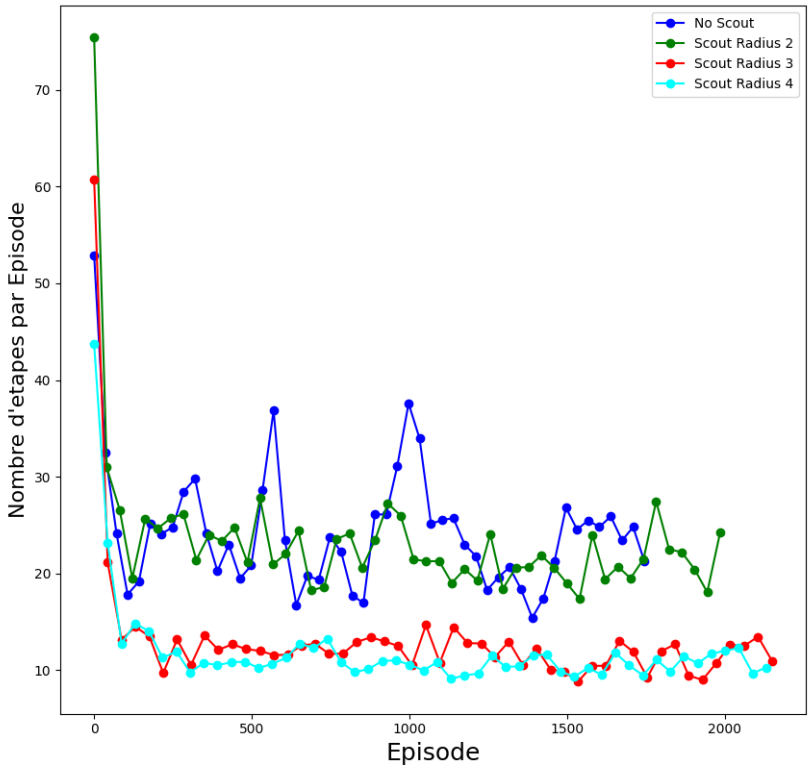


FIGURE 3.3: Partage de Sensations

Dans la figure 3.3 on observe un impact plutôt important de l'éclaireur dans la performance du chasseur, ainsi plus celui sans éclaireur est le moins éfficace que ceux disposant d'un éclaireur et plus le champ de vision de l'éclaireur est grand plus grande est l'efficacité du chasseur. Tout en ajustant le périmètre de vision de l'éclaireur celui du chasseur l'a été aussi, cependant les différences des performances observées dans la figure 3.3 sont restés sensiblement similaires tant que le champ de vision du chasseur était inférieur à 4, cela pouvant être expliqué par le fait que si le chasseur a déjà une information très importante sur l'environnement, l'ajout de la perception occasionnelle du chasseur n'améliorera pas grandement ses performances. Ces résultats sont cohérents avec ceux obtenus dans le papier.

Ce scénario voit émerger un nouveau comportement. Là où dans le cas d'agents indépendants, tant qu'ils n'avaient pas de proie en vue traversaient la grille sans cesse, ici ce comportement est altéré pas seulement si une proie entre dans le champ de vision du chasseur mais aussi si elle entre dans celui de l'éclaireur.

3.1.3 Second Scénario : Partage de politique de choix ou d'épisodes

Les cas explorés par ce scénario traitent de l'aspect de partage d'éxperience et de Q-Table et non pas d'ajout à la représentation de l'état par conséquent ils ne voient pas émerger de nouveaux

comportement, cependant ils permettent d'obtenir des résultats importants concernant la vitesse de convergence de la performance des agents.

3.1.3.1 Premier cas : Partage de Q-Table

Les 2 chasseurs partagent la même Q-Table et mettent donc à jour conjointement ses valeurs. Ce cas a démontré une convergence bien plus rapide.

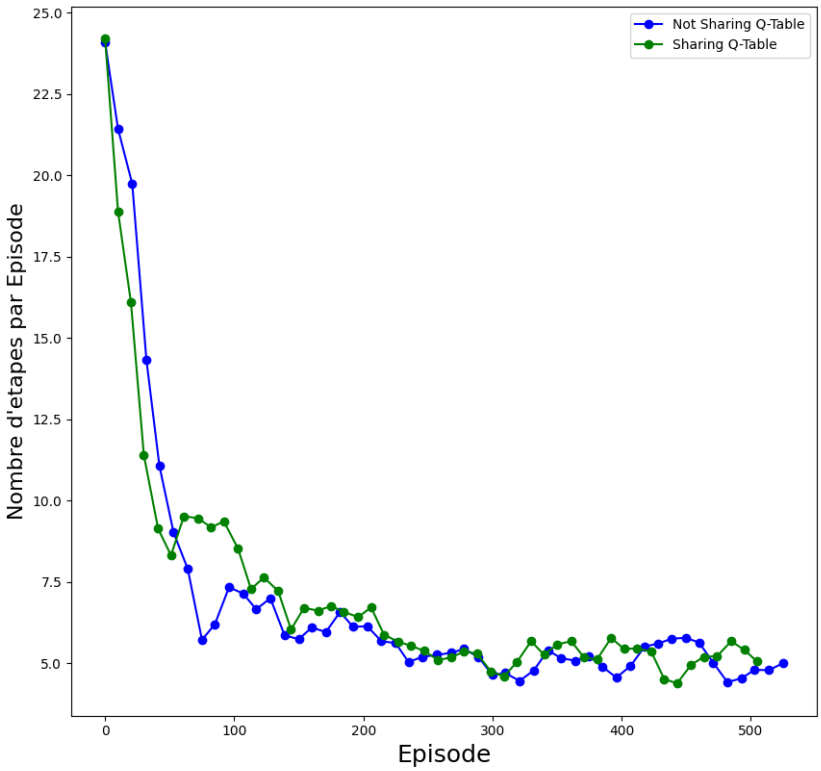


FIGURE 3.4: Partage de Q-Table

Dans la figure 3.4 on peut voir que le fait de partager la Q-Table n'affecte que légérement la vitesse d'apprentissage d'un chasseur avant que celle-ci ne rejoigne les même valeurs que celle d'un agent classique. Et dans ce cas on observe la première différence avec les résultats cités par le papier, car celui-ci obtenait des résultats significativement meilleurs par le partage de Q-Table

3.1.3.2 Deuxième cas : Synchronisation

Les 2 chasseurs à une fréquence régulière moyennent leurs tableaux.

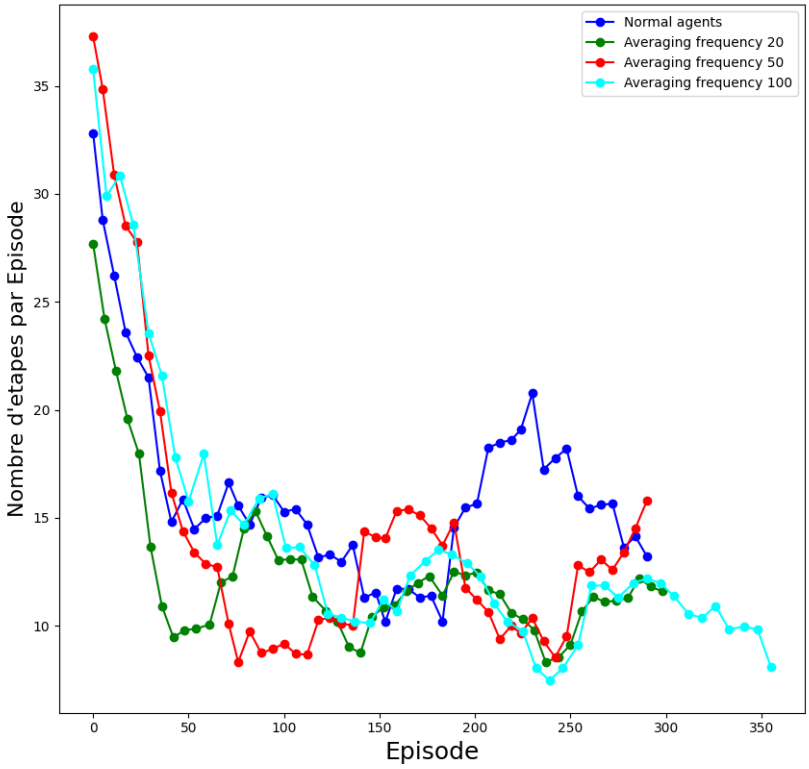


FIGURE 3.5: Synchronisation selon fréquences

Dans la figure 3.5 on peut observer que plus la fréquence de synchronisation est élévée, meilleure est la courbe initiale d'apprentissage, avant que chacun de ces cas tendent vers des valeurs similaires. Cette solution peut donc être envisagé comme une coopération afin d'accélerer la convergence de plusieurs agents. Le paramètre de la fréquence de synchronisation est donc très important dans la vitesse d'apprentissage d'un agent et confirme les résultats obtenus par le papier.

3.1.3.3 Troisième cas : Partage d'expériences

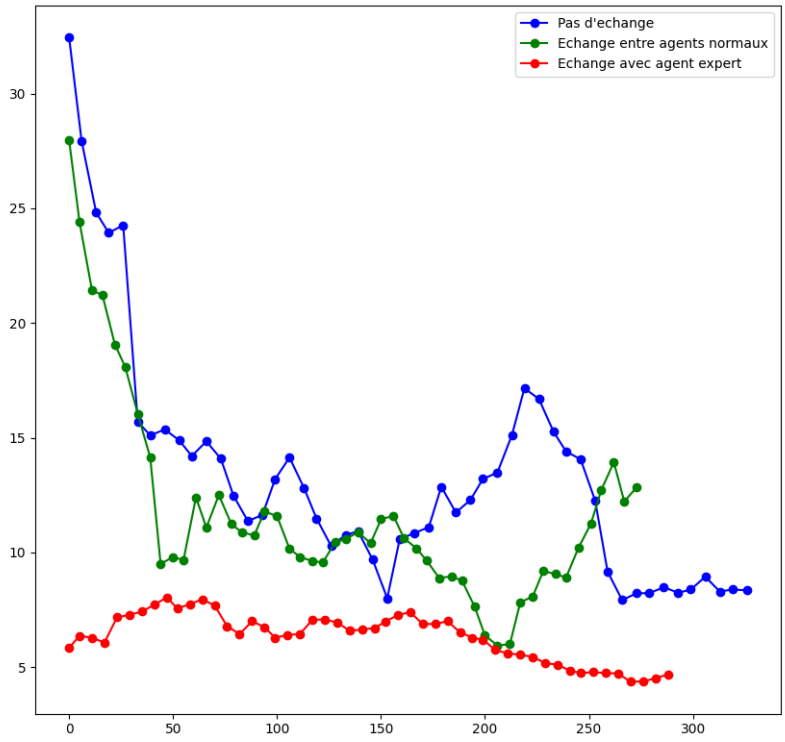


FIGURE 3.6: Echange d'Episodes

Dans ce cas les agents échangent activement leurs épisodes. Dans un premier temps on observe (Figure 3.6) qu'un échange entre agents normaux a un impact en moyenne assez clair sur l'apprentissage de ces derniers, mais surtout l'impact le plus clair est remarquable grâce à l'agent expert.

Ces résultats sont cohérents avec ceux obtenus par le papier sauf pour le cas du partage d'épisode avec l'agent expert, l'auteur ayant obtenu dans les premiers épisodes des résultats plus grands puis décroissants au fur et à mesure. Cette différence de résultat pourrait être expliqué par un manque de clarté de l'auteur concernant la manière qu'aura l'agent d'apprendre de l'expert car plusieurs cas de figures peuvent être imaginés :

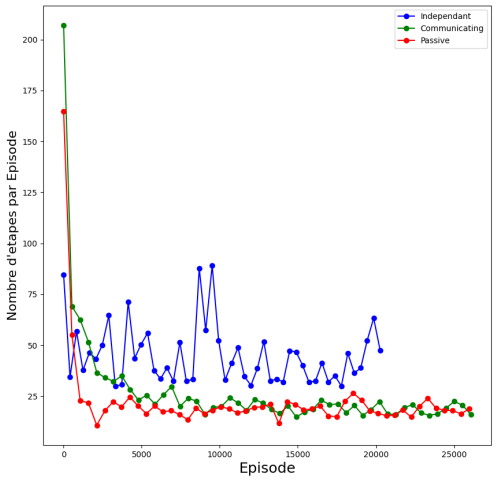
— Les deux agents existent au même moment et dès que l'expert réussit un épisode il envoie sa mémoire à l'agent normal.

— L'agent expert effectue un certain nombre d'expériences avant que l'agent normal soit crée et qu'il apprenne de celles-ci avant de commencer ses épisodes.

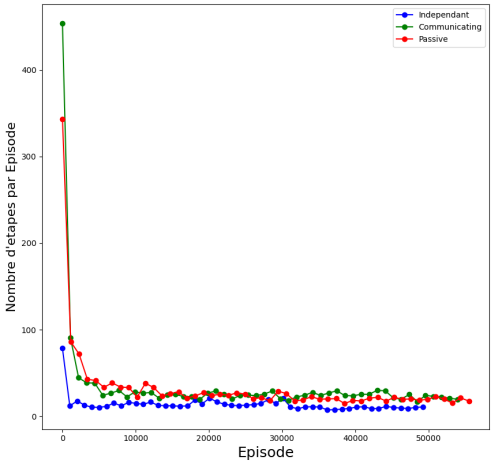
— L'agent normal commence ses épisodes et à un épisode donné l'agent expert est introduit.

Ces différents scénarios ont tous été implementés cependant aucun n'a obtenu des résultats similaires à ceux du cas de l'agent expert du papier.

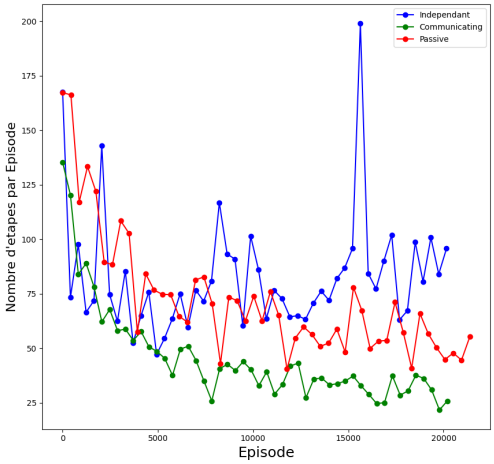
3.1.4 Troisième Scénario : Tâches jointes



(a) 2 Chasseurs 1 Proie Radius 2



(b) 2 Chasseurs 1 Proie Radius 4



(c) 2 Chasseurs 2 Proies

FIGURE 3.7: Tâche Jointe

Dans le cas de tâche jointe mais avec l'existence d'une unique proie, les résultats différaient de moins en moins plus les champs de vision des agents étaient grands, chaque agent ne nécessitant pas de réelles informations externes mais simplement de se diriger vers la proie dès qu'il l'aperçoit. La différence dûe à l'impact du champ de vision est expliquable par le fait que si à un instant un des agents ne voit pas de proie il exploitera l'information de l'autre agent.

Cependant pour le cas de l'existence de 2 proies la tâche s'avérait plus complexe. Ainsi un agent simple sans communication ou observation n'a pas pu réellement apprendre de concepts car les informations qu'il avait étaient incomplètes chacun d'eux pouvant se diriger vers l'agent la proie lui étant la plus proche sans avoir la donnée lui permettant de déduire que l'autre agent ne chassait pas la même proie que lui. Cependant, plus ces agents avaient d'informations concernant

leur coequipier, plus ils étaient efficaces, comme on peut l'observer dans la figure de droite dans la figure 3.7. L'incovénient de cette méthode étant que plus la quantité d'information est elevée plus la taille de l'information l'est et par conséquent le nombre d'états possibles. Cela implique une croissance exponentielle du temps de convergence, ainsi pour la cas avec le plus de données (celui ou chaque agent connaissait les perceptions de l'autre agent ainsi que sa distance vers celui-ci) il a fallu attendre les 20000 épisodes avant de converger vers un résultat satisfaisant.

Ici la majeure différence par rapport aux résultats du papier sont liés aux temps de convergence, l'auteur a obtenu une convergence à partir de 2000 épisodes (qu'il considérait déja comme conséquent) alors qu'ici on la convergece ne commence réllement qu'après les 20000 épisodes.

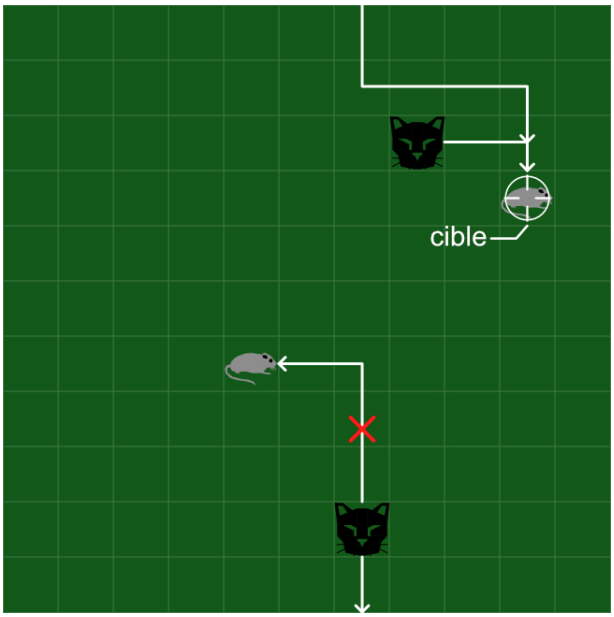


FIGURE 3.8: Exécution de Tâche Jointe

Dans la figure 3.8 on observe un nouveau comportement émergeant par coopération ou chacun des chasseurs est le plus proche de proies différentes, cependant un des chasseurs rejoint le deuxième afin de s'approcher d'une seule et même proie et réussir l'épisode.

Conclusion

De par ces expériences nous pouvons déduire plusieurs choses :

1. La majeure partie des résultats du papier ont pu être reproduits et ceux n'ayant pas pu l'être sont certainement dû à un manque de détails techniques quant à leurs implémentations.
2. Plusieurs types de coopération entre agents ont démontré leurs efficacités
   - Une permettant d'augmenter le temps de convergence et ce en moyennant les tables de savoir de tous les agents par exemple, cette fréquence qui plus étant petite moins le temps de convergence est grand, d'une autre part la méthode de partage de Q-Table ne permet pas de grande amélioration dans la performance ou temps de convergence des agents. Enfin la dernière méthode de ce type (et la plus efficace) consiste a un échange d'expériences d'épisodes réussis entre agents et plus particuliérement avec un agent expert.

   - Une deuxième permettant de réaliser des tâches impossibles à réaliser pour deux agents non coopérants et ce en élargissant la représentation d'état en y incluant des informations relatives à ses agents alliés.

Ces expérimentations ont prouvé ainsi que la coopération entre agents en apprentissage par renforcement et plus précisement avec l'algorithme de Q-Learning a un intérét réel et applicable.