MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE «Abbès LAGHROUR» DE KHENCHELA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département de Mathématique et Informatique

Mémoire

Présenté pour l'obtention du diplôme de Master en informatique (L.M.D)

Par :

LAASSAMI Fériel
BENAISSA Djamel-Eddine

Option : Sécurité et Technologies Web (STW)

Proposition d'un algorithme de planification de trajectoire pour les SMAs basé sur l'apprentissage par renforcement

Soutenu le : 00/00/2020 devant le jury :

- ***Dr. (présidant)***
- ***Dr. (examinateur)***
- ***Dr. SOUIDI Mohammed El Habib (encadreur)***

Année : *2019/2020*

Résumé

Le problème de planification de trajectoire devient un des plus intéressants défis dans différents systèmes multi-agents comme les jeux multi-agents, il est généralement résolu par l'utilisation de méthodes d'apprentissage par renforcement. Dans ce mémoire, nous proposons un nouvel algorithme de planification de trajectoires du jeu Ludo basé sur Q-learning et un ensemble de propriétés de coopération que les agents doivent suivre pour atteindre l'objectif, pour cela, nous introduisons une modélisation de l'environnement de jeu Ludo selon les principes du processus de décision de Markov (MDP). L'objectif principal de ce travail est de montrer l'avantage de cette proposition et à quel point va-t-elle être optimale. Pour réaliser ça, nous appliquons notre algorithme de planification de trajectoires (stratégie coopérative) par rapport à une stratégie égoïste également basée sur le Q-learning. Au cours de cette comparaison, nous avons étudié le temps d'exécution ainsi que l'acquisition de la récompense des agents dans différents cas (utilisation d'un dé ordinaire ou un dé truqué). Les résultats de la simulation reflètent les avantages apportés par ce nouvel algorithme et de la collaboration entre les agents dans un SMA.

Mots clés : Jeu multi-agent, Jeu Ludo, Planification de trajectoire collaborative, Processus décisionnels de Markov, Apprentissage par renforcement.

Abstract

The path planning problem becomes one of the most interesting challenges in different multi-agent systems like multi-agents games; it is usually solved by the use of reinforcement learning methods. In this thesis, we propose a new algorithm of path planning for **Ludo** game based on Q-learning and a set of cooperation properties that agents must follow to reach the goal. For this, we introduce a **Ludo** environment modeling according to the principles of the Markov decision process (MDP). The main objective of this work is to show the advantage of this proposal. To achieve this, we apply our path planning algorithm (cooperative strategy proposed) in comparison with a greedy strategy also based on Q-learning. During this comparison, we have studied the execution time as well as the allocation of the agents' reward in different cases (use of an ordinary dice or a faked dice). The results of the simulation reflect advantages of this new algorithm and collaboration between agents in MAS.

Keywords: Multi-agents games, Ludo Games, Collaborative path planning, Markov Decision Process, Reinforcement learning.

تلخيـــص

أصبحت مشكلة التخطيط المساري واحدة من أكثر التحديات المثيرة للاهتمام في الأنظمة المتعددة الوكلاء المختلفةمثل الألعاب المتعددة الوكلاء، ويتم حلها بشكل عام باستخدام طرق التعلم المعزز. في هذه المذكرة، نقترح خوارزمية جديدة لتخطيط مسارات لعبة الـ Ludoاستنادًا إلى مبدأ التعلم- Q ومجموعة من الخصائص التعاونية التي يجب على الوكلاء اتباعها للوصول إلى الهدف. لهذا، نقدم نموذجا لبيئة اللعب في Ludo معتمد على مبادئ عملية ماركوف .(MDP) هدفنا الرئيسي من العمل هو عرض أفضلية هذا الاقتراح و إلى أي درجة قد يعد الأمثل. لتحقيق ذلك، نطبق خوارزمية تخطيط المسار التعاونية خاصتنا مقارنة باستراتيجية أنانية تعتمد أيضًا على مبادئ التعلم-Q. خلال هذه المقارنة، درسنا وقت التنفيذ بالإضافة إلى تحصيل المكافآت من طرف الوكلاء في حالات مختلفة (استخدام نرد عادي و نرد مزيف). النتائج المتحصل عليها من خلال المحاكاة، تبرز فوائد هذه الخوارزمية المقترحة و فوائد التعاون بين الوكلاء داخل نظام متعدد الوكلاء.

Keywords: Multi-agents games, Ludo Games, Collaborative path planning, Markov Decision Process, Reinforcement learning.

Remerciement

La réalisation de ce mémoire a été faite à l'aide des plusieurs personnes qu'on ne pourra jamais leurs exprimer notre gratitude.

Nous tenons à remercier spécialement notre directeur de mémoire le Dr.Souidi Mohammed Elhabib pour sa patience, sa présence, sa disponibilité durant des mois de travail ainsi pour le temps qu'il a consacré à nous guider.

On souhaite aussi remercier les enseignants de l'informatique à l'université de Khenchela qu'ont fourni les outils nécessaires à la réussite durant les années universitaires.

Un grand merci à nos familles, nos amis ainsi nos collègues pour leur soutien.

Dédicace

Une spéciale dédicace à mes très chers parents, ma source de vie, de soutien et de motivation...

A mon frère et ma sœur ma source d'encouragement…

A toute ma famille du grand au petit…

A mes amis, mes collègues…

Sans oublier mes chers enseignants dès les années primaires jusqu'aux aujourd'hui…

Djamel-Eddine

Dédicace

Je dédie le fruit de mes efforts à tous ce que j'aime, à tous ce qu'ils m'aiment.

Fériel

Table des matières

Table des matières

Table des matières

Liste des figures

Figure **page**

Liste des figures

Liste des Tableaux

Tableau **Page**

La planification de trajectoire dans un environnement est un axe majeur dédié à l'intelligence artificielle en général, et l'intelligence artificielle distribuée en particulier. L'objectif principal de ce type de planification est de trouver ou de tracer dans un environnement multi-agents le chemin optimal possible (aux agents) pour leurs permettre de se déplacer automatiquement en évitant les obstacles afin d'atteindre l'objectif à partir d'un point de départ.

Les problèmes de planification de trajectoire sont généralement résolus à l'aide des méthodes de l'apprentissage par renforcement. En effet, cette planification est appliquée dans plusieurs domaines tels que la robotique et les jeux vidéo. Dans le cas de jeux vidéo, les joueurs sont mis dans un environnement virtuel sous forme d'entités autonome (agents) où chacun d'eux doit refléter son comportement en respectant un ensemble de règles. Chaque agent peut interagir avec les autres comme avec l'environnement de manière directe ou indirecte.

Le jeu **Ludo** est un problème multi-agents connu. Basé sur des conditions identiques entre les groupes d'agents concurrents, Le principal objectif de ce jeu de multi-agents est de savoir comment diriger les agents d'un même groupe à coopérer les uns avec les autres afin de bloquer la stratégie de mouvements des adversaires pour atteindre le but en premier.

L'objectif principal de ce mémoire est de proposer un nouvel algorithme de planification de trajectoire de jeu **Ludo** basé sur Q-learning et un ensemble de propriétés de coopération que les agents doivent suivre pour atteindre leurs objectifs et d'augmenter le taux de coopération entre eux dans le but de diminuer le temps d'exécution des tâches assignées. Pour mettre en valeur l'amélioration apportée par cette proposition, nous appliquons notre algorithme de planification de chemin en comparaison avec une stratégie égoïste également basée sur le Q-learning.

Alors, La problématique sera la suivante "Comment, dans le contexte du jeu **Ludo** possédant un environnement multi-agents, peut-on réaliser une planification de la trajectoire?" Pour effectuer cela, notre travail est subdivisé en trois chapitres. Le premier sera introductif, où nous allons détailler les principes de base concernant les concepts liés à notre travail. Le deuxième chapitre présente notre algorithme de planification de trajectoires proposé et introduit une modélisation de l'environnement de jeu **Ludo** selon les principes du processus de décision de Markov (MDP). Dans le dernier chapitre nous allons étudier le temps d'exécution ainsi que l'acquisition de la récompense des agents dans différents cas. Les

résultats de la simulation reflètent les avantages apportés par ce nouvel algorithme rapport à un travail connexe.

Chapitre 1

Introduction

Contenu

1.1 Introduction

L'intelligence artificielle distribuée, également appelée intelligence artificielle décentralisée, est un sous-domaine de la recherche en intelligence artificielle dédiée au développement de solutions distribuées pour les problèmes. **DAI** est étroitement lié au domaine des systèmes multi-agents.

Dans ce premier chapitre, nous allons détailler les principes de base concernant les concepts liés à notre travail. Parmi ces concepts, on retrouve les agents ainsi que les systèmes multi-agents. Aussi, nous allons expliquer comment que ces agents coordonnent leurs comportements afin de pouvoir résoudre des taches complexes (taches qui ne peuvent pas être résolues par un seul agent). Plus précisément, nous détaillerons les principes de quelques mécanismes de coordination de tache et de mouvements pour lever l'ambiguïté sur ce point.

1.2 Un agent

1.2.1 Définition

Qu'est-ce qu'un agent? C'est une question très vaste, pour cela on a choisi quelques définitions.

- Définition de l'agent dûe à Ferber [53]: « Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents et dont le comportement et la conséquence de ces observations, De ses connaissances et des interactions avec les autres agents ».

On peut souligner les mots : **'autonome', 'agir', 'communiquer', 'interaction'**. Comme propriétés clés.

- Définition de l'agent dûe à Russel [97] : « Un agent est une entité qui perçoit son environnement et agisse sur celui-ci ».

Pour plus de précision, l'agent fait percevoir à l'aide des ***capteurs***[1] (sensors en anglais), et agir à l'aide des ***effecteurs***[2] (actuators en anglais). **La Figure 1.1** nous montre ça:

[1]Liens entre environnement-agent, lui aident à percevoir. ''Entrées sensorielles ''.

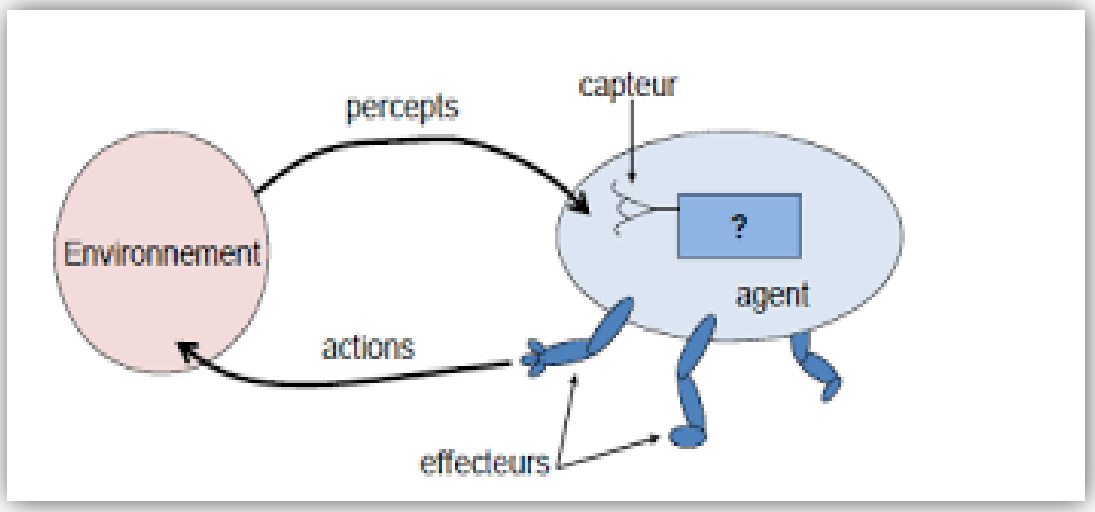

Figure 1.1 : Interaction d'un agent à l'aide des capteurs et des effecteurs avec son environnement.

On trouve d'autres définitions qui ne sont pas contradictoires avec les précédentes. Elles ne diffèrent que selon le type d'application ou domaine d'application de l'agent, et pas en propriétés clés.

1.2.2 Typologies

Il existe diverses classifications (comme plusieurs définitions), qui ont des points de vue complémentaires. **« …une typologie établie par (Frankline et Grasser 1996) selon la nature des agents… la typologie de (Nwana 1996) qui porte sur les propriétés des agents»** [1].

On s'intéresse à classer les agents d'une manière plus courante comme suit :

-**Agent réactif** : réagit aux évènements externes (reflexe), et construit sur un cycle de type : Perception/Réaction (peut intelligent).

-**Agent cognitif** : a un objectif à atteindre, un plan à réaliser. Capable de planifier ses tâches, deMémoriser, de raisonner de manièreapprofondie). Il est construit sur un cycle de type : Perception/Délibération/Action (plus intelligent).

Remarque :

[2]Liens entre agents-environnement, luis aident à agir et faire des actions.

On peut construire un agent réactif-cognitif qui appelé **‘’Agent hybride’’**.

1.3 Systèmes multi-agents

1.3.1 Définition

Comme la notion d’agent, un système multi-agents ou **SMA** n’a pas une définition bien précise. Il a plusieurs définitions comme :

- Définition d’un **SMA** selon Anne Nicole : « un système multi-agents (SMA) comporte plusieurs agents qui interagissent entre eux dans un environnement commun.Certains de ces agents peuvent être des personnes ou leurs représentants (avatars) ou même des machines mécaniques. S’il y’a moins de trois agents on parle plutôtd’interaction homme/machine ou machine/machine que de système multi-agents ».

Pour plus de précision, un **SMA** est un ensemble organisé d’agents qui échangent entre eux. Il est considéré aussi comme un ensemble de processus informatiques qui déroulent en même temps, vivant en même environnement. Ce dernier est un système distribué composé de plusieurs agentsqui interagissent selon certaines relations. Ces agents peuvent être des processus, des robots ou un être humain …etc.

1.3.2 Systèmes mono-agents et systèmes multi-agents

Lorsque un agent est en interaction avec son monde, on peut les considérer **-agent et environnement-** comme un système, dit ***mono-agent*** s’il est seul ou bien ne communique pas avec d’autres agents, ***ou multi-agents*** (SMA) s’il fait une communication avec d’autres agents. Chaque système est devisé en **« …deux sous-systèmes dynamiques couplés, le couplage s’effectuant aux travers des perceptions que l’agent a du monde, et des actions qui modifient ce monde »[2] .**

Selon l’auteur, les systèmes mono/multi-agents composés de deux parties différentes selon l’activité de l’agent (percevoir/agir).

1.3.3 Pourquoi doit-on utiliser un SMA ?

- Lorsqu’on parle d’un **SMA** on parle implicitement de l’agent. Un agent peut en plus contrôler son comportement au contraire des autres entités, donc un environnement de plusieurs agents va être aussi mieux géré, mieux contrôlé.

- On a dit qu'un **SMA** est un système distribué, ça donne la possibilité de coopérer des machines distantes. Alors les données et les ressources sont distribuées.
- Permettre l'interopérabilité entres les systèmes non homogènes.
- Un **SMA** est mieux **« pour un problème complexe et peut être décomposé »**[3]**.**
- Rendre le système adaptatif.
- Robustesse et performance.
- Gain en temps **« on peut paralléliser le problème »** [3].

1.3.4 La communication entre les agents dans un SMA

« La communication est un des éléments importants du système multi-agents. C'est l'élément de toute interaction. Elle permet l'échange entre deux agents… c'est grâce à la communication que les différents protocoles d'interaction sont exécutés »[4].

C'est-à-dire :

Les agents doivent communiquer entre eux pour coordonner, coopérer et collaborer leurs activités. Alors la communication est une transformation d'une information entre agents à l'aide d'un langage articulé.

Donc les agents doivent échanger leurs buts, résultats et leurs états, pour qu'ils puissent finir leur travail collaboratif.

Un agent qui peut coordonner sans communiquer a le modèle de comportement des autres agents.

La communication est plus efficace pour observer les changements de l'environnement.

On a deux types de communication. Elle peut être ***Directe*** ou ***Indirecte***selon certaines conditions.

1.3.4.1 La communication directe

Dans ce type de communication, l'information est échangée explicitement, via des messages (un agent envoi un ou des messages à un ou des autres agents).

« **Elle est propre aux agents cognitifs, car ces derniers relativement aux agents réactifs ont des connaissances sur eux-mêmes et sur autrui. Ils ont aussi des connaissances sur**

leurs accointances ainsi que des intentions et des compétences communicationnelles qui leurs permettent d'envoyer des messages… »[4] .

Ça veut dire que ce type de communication permet précisément aux agents réactifs d'effectuer ses rôles et laisser leurs traces. Ce dernier utilise des langages de bas niveau.

1.3.4.2 La communication indirecte

Dans ce type de communication, les agents agissent sur leur environnement qui représente le support de la communication. Alors l'information ici est échangée implicitement via l'environnement.

« **Elle est utilisée par les agents réactifs, elle se fait à travers l'environnement ou bien le biais d'un tableau noir… »[4]**.

Ça veut dire que les agents s'adaptent aux changements de l'environnement (reflexes). Et les résultats dans ce type de communication sont écrits dans un espace de recherche partagé obtenu par les connaissances des agents, dont les agents peuvent accéder à lui.

Le point clé des **SMAs** réside dans la formalisation de la coordination.

1.4 La coordination

1.4.1 Définition

Avant de formuler une définition sur la coordination, on a trouvé quelques définitions telles que :

- Définition due à Gelenter et Carriero [92] : « la coordination est le processus de construction de programmes en assemblant les parties actives ; un modèle de coordination est la colle qui lie des activités séparées en un ensemble ».

Selon Gelenter et Carriero la coordination est un processus lié directement aux liens ou bien relations entre les agents, donc on parle ici de '**SMA**'.

- Définition due à Malone et Growston [94] : « la coordination est la gestion des dépendances entre les activités »

Selon Malon et Growston la coordination est assez vaste pour nous : la coordination est une tâche essentielle dans les **SMAs**, elle détermine l'organisation des agents la plus convenable avec le système, dont la performance de ce dernier dépend d'elle. Ainsi que **« la coordination détermine en quelques sorte quelle sont les règles de bon fonctionnement du système… »**[5].

Remarque :

On dit ***une coordination statique***basée sur la convention à appliquer telle règle, comme (rester à droite, stopper…etc).

On dit une ***coordination dynamique*** lorsque les agents basent sur une analyse sur l'information, Comme (il y'a des barrières ici, je vais changer mon chemin…etc).

1.4.2 Pourquoi doit-on coordonner ?

- Les agents peuvent gérer leurs états.
- Les protocoles de coordination permettront de gérer les états au profit de l'évolution de son environnement.
- L'agent décide quelle action doit-il prendre ?
- Rendre le système distributif.

Mais avant de réaliser tout ça on a besoin d'une pré-organisation, d'une planification.Pourquoi ?

1.5 La planification

C'est la création d'un plan ou séquence d'actions permet de coordonner les agents dans un **SMA** (on utilise la planification pour réaliser la coordination). D'une autre façon :

Une bonne planification => Coordination de qualité => Système performant

On a un agent ou bien tout un système qui va distribuer les plans aux agents.

Alors elle aide les agents à accomplir une mission. La planification est donc un ***mécanisme de coordination***[3].Elle est devisée en deux sous problèmes, **planification de tâches** et **planification de mouvements**.

La Figure suivante montre l'agent central qui distribue les missions partielles aux agents.

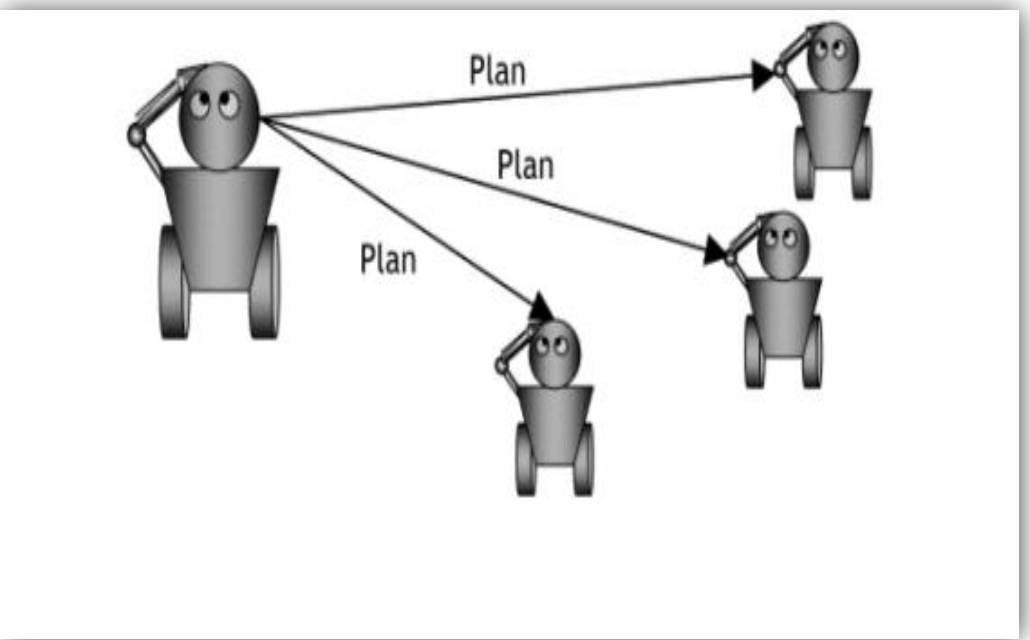

Figure 1.2 : Ladistribution des plans partiels dans un**SMA**

Pour terminer le travail de la planification c'est à dire pour coordonner, les plans partiels doivent être fusionnés.

La Figure suivante montre la coordination de la planification :

[3] Ferber a postulé qu'une approche de coordination s'appuie sur un mécanisme de Synchronisation, planification, réaction ou règlementation.

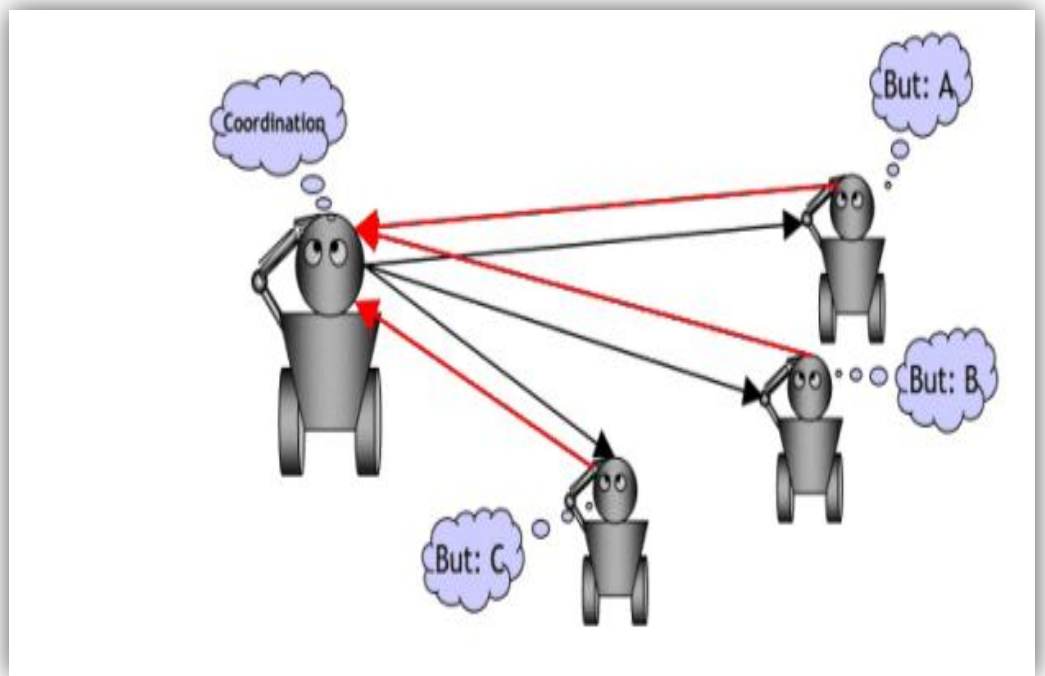

Figure 1.3 : Fusion de plans partiels

1.5.1 Planification de tâches

La décomposition de la tâche principale (mission) en sous-tâches (sous actions élémentaires). Ce qui fait cette décomposition est le planificateur de tâches, par exemple :***(HTN)***[4]**,** puis on les affecte –les tâches- au tel agent. Par exemple : (dans un système de conduite, la tâche principale est d'atteindre unemplacement bien défini, cette tâche peut-être devisée en : départ, arrêt et ralentissement…etc).Puis même ces tâches peuvent-être devisées, jusqu'à obtentiond'une tâche élémentaire.

Pour chaque tâche, le planificateur regroupe les sous tâches en méthodes, **« …chaque méthode décrit l'ensemble des sous tâches ainsi que les restrictions quand à son application, pour valider une tâche le planificateur choisit une méthode applicable et essaie de valider les sous tâches qui la composent, cette validation de sous tâches va elle aussi entrainer le choix d'une autre méthode… »**[6].

Alors la méthode applicable est la méthode dont toutes ses sous tâches peuvent-être réalisées. La validation d'elle lance une autre méthode c'est-à-dire des autres sous tâches, donc une autre activité.

[4] HierarchicalTask Network, c'est un planificateur des taches consiste à décomposer jusqu'obtention des actions élémentaires.

D'une part, les **SMAs** ont besoin d'une pré-organisation pour qu'ils soient adaptables, modulaires et décentralisés. Cette pré-organisation peut-être représentée sous forme d'un modèle pour élever l'hétérogénéité des systèmes multi-agents, pour que ces derniers puissent coopérer leurs tâches. Ce modèle est appelé un ***modèle organisationnel*** qui fait la décomposition des objectifs ou capacités en tâches et définition des rôles.

On a plusieurs modèles comme ***Opera***, ***OMNI*** et ***STEAM*** , mais le plus courant c'est l'**AGR**. Ces modèles décrivent les organisations des systèmes fortement dynamiques, auto-organisés et adaptatifs.

On a dit que le modèle le plus utilisé c'est l'**AGR,** parce qu'il décrit (à peu près) tous les types d'organisations ou systèmes multi-agents facile à décrire et à modéliser une planification de tâches.

D'une autre part, Pour modéliser les interactions, il faut tout d'abord savoir le formalisme qui étudie ces dernières entre les agents. Ce formalisme est la théorie des jeux (dite **TG** aussi). Généralement ce formalisme est fait pour les jeux dynamiques, afin defaire en sorte que la compétition entre processus évoluent dans le temps.

La théorie de jeux a pris plus l'importance au monde du **SMA** précisément pour la planification et l'application à fin de rendre les agents **autonomes**.

Un problème de coordination peut être vu comme un problème de théorie de jeux. La question qui se pose : Comment ?

A partir de cela, on va parler de l'**AGR**ainsi de la **TG**.

1.5.1.1 Le modèle AGR

Le modèle **AGR** (Agent-Groupe-Rôle) est un modèle proposé par Ferber et Gut Knecht [98], Gut Knecht et Michel [2003], qui permet une décomposition d'un système complexe en groupe d'agents, déterminant ainsi les interactions entre groupe/agent et les rôles joués. Pour plus de précisions :

- **Un agent :** on peut lui donner une petite définition dans ce modèle :

« Une entité agissante, autonome, aidant l'action s'exerce dans un contexte social »[7] .

Alors l'agent c'est l'entité la plus importante, sans elle, il n'y'aura pas d'un **SMA** ou modèle. Elle joue un rôle au sein d'un (des) groupe(s).

- **Un groupe** : le moyen de regroupement des agents « **le groupe est avant tout un terme générique pour qualifier la communauté d'agent en relation** »[7].
- **Un rôle** : on peut lui donner la définition : **« représentation abstraite d'une fonction du groupe pouvant contraindre le comportement de l'agent, et incarnée dans un ou des comportements spécifiques par l'entité** » [7] .

Ça veut dire que par ce rôle, l'agent pourra être identifié dans le groupe.

Pour la structure d'AGR :

«…1-Tout agent est membre d' (au moins) un groupe…

2-Deux agents communiquent seulement s'ils sont membres de même groupe…

3-Tout agent joue (au moins) un rôle dans le groupe…

4-Un agent est un membre du groupe dans lequel il joue un rôle…

5-Un rôle est défini dans la structure d'un groupe»[8] .

Alors le système a plusieurs groupes d'agents qui ne peuvent communiquer qu'avec les autres agents du même groupe.

La communication interne dans le groupe est ***direct***, par contre la communication externe est ***indirecte*** étant donné que chaque groupe réagit au travail des autres.

1.5.1.2 La théorie de jeux

La **TG** sert à étudier formellement ou mathématiquement des jeux où les joueurs sont des agents dont chacun d'eux peut prendre ses propres décisions, ses propres actions ou des décisions qui concernent les autres agents -joueurs- qui partagent le même jeu…etc.

Chaque agent peut-être un robot, un processus …etc.

Pour gagner plus de récompenses, l'agent ici réfléchit, décide et agit rationnellement.

La **TG** répond à la question suivante : Qui peut analyser les interactions ? Quelle sera l'utilité finale des agents ?

Pourquoi les agents ?

- Atteindre leur objectif dans les brefs délais.
- Calcul réparti de la solution.
- Résolution du problème.
- Travail collaboratif, décision collective.

Il vaut mieux prendre un simple jeu comme exemple : prenons le jeu (Pierre/Feuille/Ciseau), son principe est évident. Chacun choisit une pierre ou feuille ou un ciseau. Trois possibilités menées à chaque joueur (Gain/Nul/Défaite).

- Si les deux effectuent le même choix ça sera un nul.
- Si un des deux choisit une pierre, il va gagner si et seulement si l'autre choisi un ciseau.
- Si un des deux choisit une feuille, il va gagner si et seulement si l'autre choisiune pierre.
- Si un des deux choisit un ciseau, il va gagner si et seulement si l'autre choisi une feuille.
- Le gain est représenté par 1, le nul par 0 et la défaite par -1.

Chaque joueur va choisir la meilleure solution, les agents réfléchissent rationnellement, ils tentent d'arriver à la meilleure solution pour eux, donc à la meilleure situation.

Le tableau suivant montre les situations possibles :

			
	(0,0)	(-1,1)	(1,-1)
	(1,-1)	(0,0)	(-1,1)
	(-1,1)	(1,-1)	(0,0)

Tableau 1.1 : Situations possibles d'un agent dans le jeu (Pierre/feuille/ciseau)

1.5.2 Planification de mouvements

La planification de mouvements est la génération d'une série de mouvements continus, d'une ***configuration***[5] initiale à une autre finale, dans un espace de configurations, il n'y'a que les agents mobiles qui ont besoin d'elle. Cette dernière est faite par un planificateur de mouvements, qui les calcule au niveau géométrique. Ces mouvements dépendent totalement des objets, des agents et des positions dans l'espace du système. D'une façon brève, cette planification permet de trouver une ***trajectoire***[6] sans collision pour atteindre un but donné.

« On peut définir un algorithme de planification de mouvements par :

-Ses entrées : les descriptions géométriques d'un système articulé et d'un environnement.

-Sa sortie : une suite de mouvements qui résout certaine tâche »[9].

La sortie est les mouvements obtenus par le planificateur à l'aide des entrées qui sont-elles même des positions.

[5]Positions accessibles par les effecteurs d'un agent.
[6]Trajectoire sans collision c'est éviter les barrières, soient des agents ou objets

Parmi les algorithmes on a ***PRM***[7] et ***RRT***[8].

La planification en général nommée aussi une branche de la théorie de la décision,est utilisée pour forcer un agent à atteindre son but dans un temps optimal (forcer le à réfléchir).

La planification dans l'incertain, est appelée :***l'apprentissage par renforcement.***

1.5.2.1 L'apprentissage par renforcement

Cet apprentissage (**RL** pour Reinforcement Learning) fait référence à une classe de problèmes d'***apprentissage automatique***, dont le but est d'apprendre les actions à apprendre à partir des expériencespour trouver la meilleure solution.
Dans un tel problème, on dit qu'un ***agent*** plonge, interagit avec son ***environnement*** et prend ses décisions en fonction de son état courant, pour trouver la solution optimale.
L'apprentissage par renforcement diffère des problèmes supervisés et non supervisés par : l'exploration que l'agent lui fournit à plusieurs solutions, l'observation de la réaction de l'environnement et l'adaptation de son comportement (comportement décisionnel ou politique ou stratégie).

L'apprentissage par renforcement est aussi appelé : ***un apprentissage par essai et erreur,***dont l'environnement va décider ou indiquer la valeur de l'action (récompense ou punition). Son principe est représenté sous la forme de ***« l'exécution à chaque instant n d'une action $a_n$depuis l'état courant $s_n$qui conduit à un nouvel état s'_n et qui fournit la récompense r_n »***[10].

Prenons un exemple présenté dans la **Figure 1.4,** l'apprentissage par renforcement peut être utilisé pour apprendre à jouerle jeu SuperMario, dontl'agent Mario va trouver une stratégie pour atteindre son but sans mourir.

[7]PRM : pour Probabilistic Roadmaps, algorithme de planification de mouvements [96], utilisant les arbres aléatoires.
[8]RRT : Rapidly-exploringTrees, algorithme de planification de mouvements [96], utilisant les arbres aléatoires.

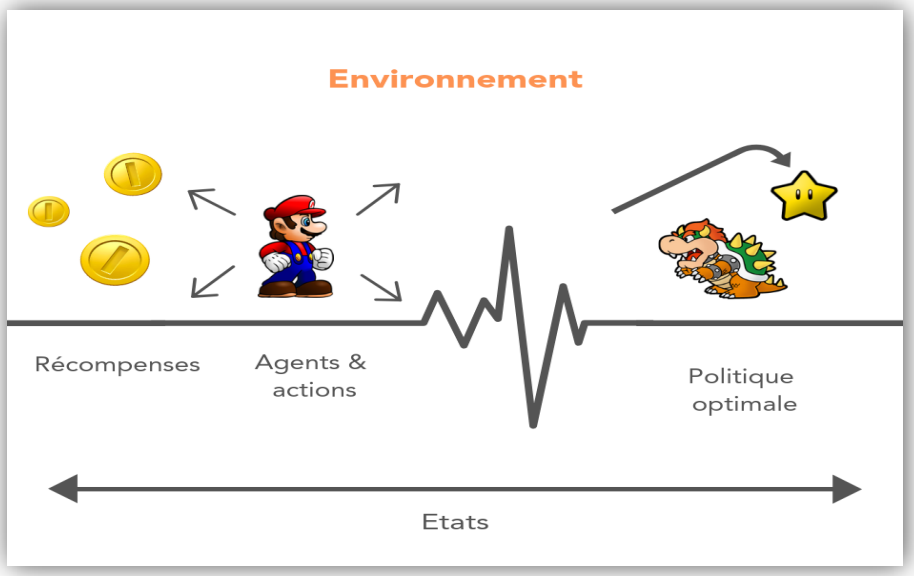

Figure 1.4 : Exemple d'application d'apprentissage par renforcement**.**

Passons maintenant aux domaines de la planification et de l'apprentissage.

1.5.2.2 Processus décisionnels de Markov

Unprocessus décisionnel de Markov ou***MDP*****« constituent un cadre devenu standard pour représenter les problèmes de décision séquentielle dans l'incertain… »[11].**

Alors un **MDP**est la modélisation de l'incertitude sur les résultats d'actions de l'agent, dans laquelle il peut prendre une décision, est reçoit une récompense.

Un **MDP** est un modèle ***stochastique***[9]. Il est utilisé pour résoudre des problèmes d'optimisation à l'aide de l'apprentissage par renforcement.

1.4.2.2.1 Les composants d'un MDP

-Un espace d'états *S* (incluant un état initial s_0).

-Un espace d'actions *A(s)* lorsque je me trouve à l'état *s*.

-Une fonction de transition, indiquant la probabilité P(s'|s,a) (ou a appartient à *A(s)*) d'atteindre l'état *s'* après avoir entrepris l'action *a* dans l'état *s*.

-Une fonction de récompense *r()* (utilité d'être à l'état *s*).

[9] Cadre formel permet de modéliser un environnement multi-agent non coopératif.

-Un espace temporel *T*.

Alors là le but c'est décrire un objectif à atteindre à partir d'une fonction de récompense basée seulement sur l'état courant.

« **La résolution d'un problème de contrôle d'un MDP se fait généralement au travers de la définition d'une politique π qui associe à chaque état d'action…** »[11].

c-à-d, Le **MDP** base sur la stratégie ou la politique ou le comportement de l'agent dans chaque état.

La Figure 1.5**« représente un MDP sous la forme d'un diagramme d'influence. A chaque instant *t* de *T*, l'action a_t est appliquée dans l'état courant s_t, influençant le processus dans sa transition vers l'état s_t+1. La récompense r_t est émise au cours de cette transition »**[10].

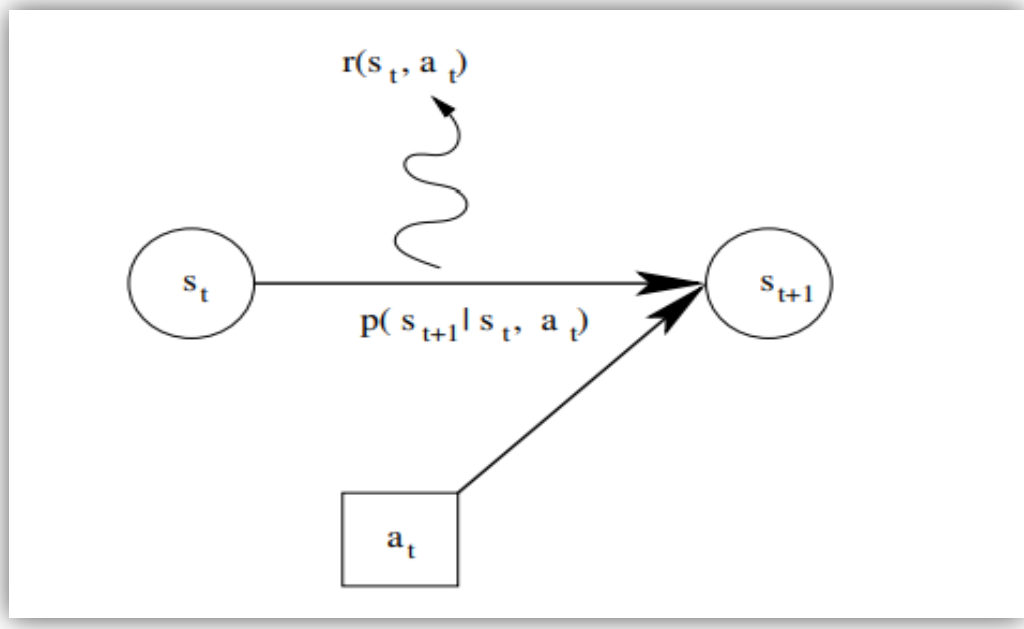

Figure 1.5 : Evolution d'un MDP

1.5.2.2.2 Les politiques d'un MDP

Pour contrôler un **MDP**, on peut définir des politiques, qui vont indiquer comment choisir les actions à chaque pas de temps *t* en fonction de l'historique h^t que l'on a observé.

Politique 1 (stochastique) : Une politique stochastique est une famille de distributions de probabilité selon laquelle une action *a* doit être sélectionnée pour chaque historique observé h^t.

Politique 2 (déterministe) : Une politique déterministe définit précisément l'action qui doit être sélectionnée en se basant sur l'historique $h^{t.}$.

Politique 3 (markovienne) : On parle de politique (stochastique ou déterministe) markovienne lorsque la politique ne dépend que de l'état courant s^t et non de l'ensemble de l'historique $h^{t.:}$ Dans le cas contraire, on parle de politique histoire-dépendante.

Politique 4 (markovienne stationnaire). On appelle politique (stochastique ou déterministe) stationnaire une politique markovienne qui ne dépend pas du temps.

1.5.2.2.3 Les domaines d'application des MDPs

« Les MDPs ont été premièrement utilisés pour la gestion de route de l'état d'Arizona en 1978 [Golabi et al., 1982; Puterman, 1994; Mundhenk et al., 2000]. Depuis ces formalismes puissants ont attiré l'attention de plusieurs chercheurs dans différents domaines»[12].

Ça présente la diversité des applications d'un **MDP**. D'une façon brève, les domaines sont :

- La robotique.
- Résoudre des problèmes de navigation.
- Les chercheurs de La NASA utilisent les **MDPs** pour modéliser les problèmes de planification des robots envoyés sur Mars [Bresina et al., 2002].
- Problèmes de planification des opérations militaires [Aberdeen et al., 2004].

Et finalement, dans ce modèle, ou dans l'apprentissage par renforcement en générale, on calcule les récompenses à l'aide du ***Q-learning***.

1.5.2.3 Q-learning

Le Q-learning est une technique d'apprentissage par renforcement, **« Le Q-learning consiste à évaluer cette Q-fonction dynamiquement par l'expérience de l'agent dans l'environnement. Étant donnés l'état précédent s, l'action a effectuée en s et l'état courant… »[13].**

Ça veut dire l'agent fait une action dans son environnement à chaque état, et reçoit une récompense (selon l'état). Cette récompense est le résultat de l'action (gain/perte).

On l'appelle Q-learning avec la lettre Q qui signifie et référencie la fonction Q ou Q-Fonction qui doit être calculée, et mise à jour à chaque état.

Le but de l'agent ici s c'est maximiser le gain.

Prenons un exemple d'application:

Dans cet exemple on va s'intéresser à un problème simple, c'est une petite voiture dans un environnement, son but est d'aller à la maison sans écraser les piétons. Cette **Figure** représente l'environnement:

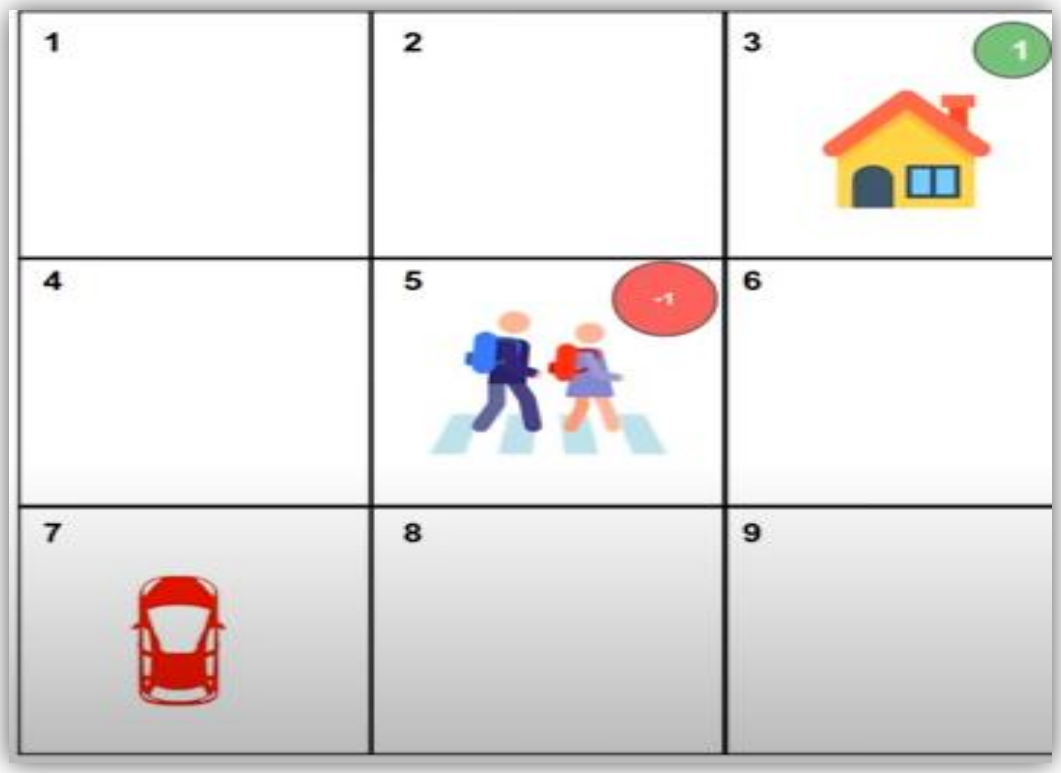

Figure 1.6: exemple d'application le Q-learning

- Chaque grille contient une récompense et chaque chiffre de 1 à 9 représente un état.
- Lorsque la voiture arrive à l'état 3, elle reçoit une récompense de 1.
- Lorsque la voiture arrive à l'état 5, elle reçoit une récompense de -1.
- Lorsque la voiture arrive à n'importe autre case, elle reçoit une récompense de 0.
- La voiture peut bouger seulement à gauche, à droite, en haut et en bas.

On va définir un nouveau concept qui s'appelle Q-fonction (ou Q-Function en anglais), il est défini comme suit:

La formule $Q(s_t,a_t)$, dont s_t c'est l'état de l'agent (la voiture ici à un moment). Et a_t c'est l'action à prendre (ici c'est gauche, droite, haut et bas).

Le but de la fonction est de répondre à la question suivante :Si je prends telle action, Quelle serait la récompense que je pourrais avoir ?

La formule mathématique de notre fonction deviendra alors:

$$Q(s_t,a_t)^{\pi} = E[R_{t+1} + \Upsilon R_{t+2} + \Upsilon^2 R_{t+3} + \Upsilon^3 R_{t+4} + ... \mid s_t, a_t]$$

Où le facteur Υ est donné.

En appliquant cette formule à l'exemple précédent pour mieux comprendre, où Υ égale à 1, on obtiendra :

-prenons les actions (droite haut droite bas), la récompense obtenue par la formule ou la fonction sera 0 parce que: 0 + (-1) + 0 + 1= 0.

-prenons les actions (droite haut droite bas), la récompense obtenue par la formule ou la fonction sera 1 parce que: 0 + (-1) + 0 + 1= 1.

Plus on est proche de 1, plus on est proche de but (1 c'est le gain obtenu au but final).

1.6 Conclusion

Après avoir introduit les principes de base des systèmes multi-agent, l'objectif de notre travail consistera à décrire le problème du jeu **Ludo** ainsi que notre solution proposée

Dans le chapitre suivant, nous allons détailler notre algorithme de planification du mouvement via son application au jeu **Ludo**.

Chapitre 2

Proposition d'un nouvel algorithme de planification de trajectoire

Contenu

2.1 Introduction

La planification de la trajectoire d'un système multi-agent peut être définie comme un mécanisme ou une stratégie qui va guider le déplacement de chaque agent de la position initiale jusqu'à la position finale.

Dans ce deuxième chapitre, nous allons introduire notre algorithme de planification de trajectoire via son application au jeu **Ludo**. Au début, nous allons discuter quelques travaux reliés a le nôtre concernent la planification de trajectoires, les jeux basés sur les SMA, ainsi que le jeu **Ludo**. Aussi, nous allons expliquer la modélisation de l'environnement du jeu à travers l'application des principes du Processeurs Décisionnel de Markov. Ce chapitre contient aussi l'application du Q-learning qui va permettre aux groupes d'agents de prendre une décision de mouvement d'une manière centralisée afin de générer un comportement collaboratif des agents.

2.2 Travaux reliés

Les jeux de société [14], sont des jeux multi-joueurs qui peuvent être considérés comme un type de jeux de table, dont L'objectif principal pour chaque joueur participant est de choisir la meilleure stratégie pour atteindre le but du jeu avant que les autres joueurs l'atteignent. En général, Les stratégies de jeu dépendent d'une séquence de nombres aléatoires obtenue via l'utilisation d'un dé ou d'un tirage de cartes contenant des instructions ambigües.

Parmi les jeux de société les plus populaires, on a le jeu Ludo[15], Ce jeu non déterministe peut être joué grâce à la participation de deux ou bien quatre joueurs. Dans ce dérivé de Pachisi [16], chaque joueur gère un groupe de quatre pions ou agents afin de leur fournir une collaboratif stratégie de planification de mouvements qui leur permet d'atteindre l'objectif avant les groupes concurrents.

La planification de trajectoires [17](dite path planning en anglais) multi-agents peut être définie comme la coordination de mouvements des agents mobiles pour y'arriver à l'emplacement de l'objectif. Cette coordination permet la coopération de l'agent de manière centralisée ou décentralisée. Dans la coopération centralisée[18], la trajectoire de chaque agent dépend de la trajectoire des autres agents comme dans le jeu **Ludo**. Par contre La

coopération[19]décentralisée concerne les cas où les agents se déplacent indépendamment sans prendre en considération les trajectoires des uns des autres.

Généralement, les mécanismes de planification de trajectoire sont liés aux jeux multi-joueurs pour proposer des stratégies de déplacement adéquates aux composants du jeu (agents). Le jeu Poursuite-Evasion[20] (dit Pursuit-Evasion en anglais ou **PEG**) a été l'un des cas les plus ciblés concernant l'application des méthodes de planification de trajectoire au cours des dernières années.

Dans [21], Les auteurs ont proposé une modélisation de l'environnement selon les principes **MDP**, pour guider les mouvements des agents. Précisément, les gains distribués dans l'environnement ont été inversés proportionnellement par rapport à la distance entre l'évasion et les cellules de l'environnement. En plus de ça, ils ont basé sur le Q-learning pour permettre aux poursuivants de choisir leurs directions lors des itérations de poursuite.

Dans [22], ils ont introduit une nouvelle extension des algorithmes de bug (parfois francisé bogue) [23] pour éviter les obstacles rencontrés. Même dans ce travail, ils ont basé sur la fonction de récompense pour détecter les points de départ de l'obstacle dans un temps minimal par rapport aux précédents algorithmes de bug proposés.

Dans les travaux importants concernant le jeu multi-agents **Ludo**[24], ils ont essayé d'améliorer les connaissances spécifiques au domaine pour **Ludo** et ses variantes de jeux de course. En fait, ils ont estimé la complexité de l'espace d'états du jeu **Ludo**, proposé et analysé des stratégies basées sur quatre mouvements de base. En plus de ça, ils ont comparé expérimentalement les versions pures et mixtes des stratégies étudiées. En relation avec l'apprentissage par renforcement[25]**.** Ils ont proposé un joueur expert basé sur l'amélioration de leurs stratégies de base proposées sur **Ludo**. Plus précisément, ils ont implémenté un joueur **Ludo** basé sur le **TD (λ)** et ils ont utilisé le joueur expert pour l'entraîner. De plus, ils ont implémenté un joueur **Ludo** basé sur le Q-learning en utilisant les connaissances acquises lors de la construction du joueur expert. Les résultats obtenus ont montré un avantage au profit des joueurs Q-learning et **TD (λ)** par rapport au joueur expert.

Dans [26], ils ont présenté une nouvelle stratégie pour évoluer les performances des algorithmes d'essaim (dits Swarm Algorithms en anglais) [27] basés sur les règles de **Ludo**. En basant sur ces règles, deux algorithmes d'essaim basés sur le jeu **Ludo** ont été introduits (**LGSI2, LGSI4**) pour simuler les cas de deux et quatre joueurs **Ludo**. Dans **LGSI2**, les deux

joueurs sont l'optimisation Moth-Flame (**MFO**pour Moth Flame Optimization) [28] et l'algorithme d'optimisation de Grasshopper (**GOA** pou Grasshopper Optimization Algorithm)[29]. Parallèlement, l'algorithme Sine Cosine (**SCA** pour Sine Cosine Algorithm)[30] et l'optimisation de *Loups Gris*(**GWO** pour Gray Wolf Optimization) [31] sont utilisés dans le cas de **LGSI4**.

2.3 Description du problème

Le problème **Ludo** peut être considéré comme un jeu multi-agents, dont chaque groupe d'agents essai d'atteindre ses cellules cibles en fonction des valeurs prévues par un dé. Sachant que les groupes n'utilisent qu'un seul dé à leur tour, l'objectif des agents appartenant au même groupe est d'atteindre leur cellule cible avant les agents appartenant aux autres groupes.

Dans ce type de jeu, on peut noter deux comportements d'agent différents : les comportements collaboratifs et concurrents. D'une part, un comportement collaboratif est relié aux agents appartenant au même groupe, qui coopèrent toujours selon une stratégie spécifique jusqu'à la fin de la partie. D'autre part, le comportement concurrent est relié aux agents appartenant aux différents groupes.

En d'autres termes, un agent peut retourner un autre agent d'un autre groupe à sa position initialesi le premier agent atteint une cellule occupée par le second (cas mortel). De plus, un groupe d'agents peut réutiliser les dés dans deux cas différents, si les dés fournissent le chiffre six ou en cas de tuer un autre joueur.

La **Figure 2.1**montre notre environnement de simulation. Elle représente l'état initial du jeu **Ludo**. On peut noter que les agents appartenant au même groupe ont la même couleur (en fait quatre joueurs donnent quatre couleurs : rouge, bleu, jaune et vert). L'environnement est composé de différents types de cellules. Les caractéristiques de chaque type de cellules peuvent être résumées comme suit :

- Les cellules initiales : les cellules contenant les agents jusqu'à ce que le joueur obtient le nombre de dé (numéro six) qui emmène l'un des agents à la cellule de départ. Chaque cellule initiale n'est accessible qu'à un seul agent.
- Les cellules de départ : sont des cellules contenant des flèches. Chacune d'eux reçoit des agents du groupe respectif de leurs cellules initiales. En plus, ces cellules sont

considérées comme neutres, parce qu'elles peuvent contenir plus d'un agent du même groupe ou de différents groupes en même temps.

- Les cellules Etoile : ces cellules sont également dites des cellules neutres (pas de mort).
- Les cellules noires : D'une part, ce type de cellules peut contenir plus d'un agent s'ils –les agents- appartiennent au même groupe. D'autre part, si la cellule est déjà occupée par un agent, et qu'un autre agent d'un autre groupe lui accède, le premier agent revient dans sa cellule initiale.
- Les cellules colorées : ces cellules ne sont accessibles que par des agents de même couleur des cellules concernées. Chaque cellule peut contenir plusieurs agents. On peut déduire que ces cellules sont également neutres.
- Le triangle coloré : chaque triangle coloré représente la cellule cible ou bien l'objectif principal des agents ayant la même couleur que ce triangle. Ce type de cellules a le même principe que les cellules colorées.

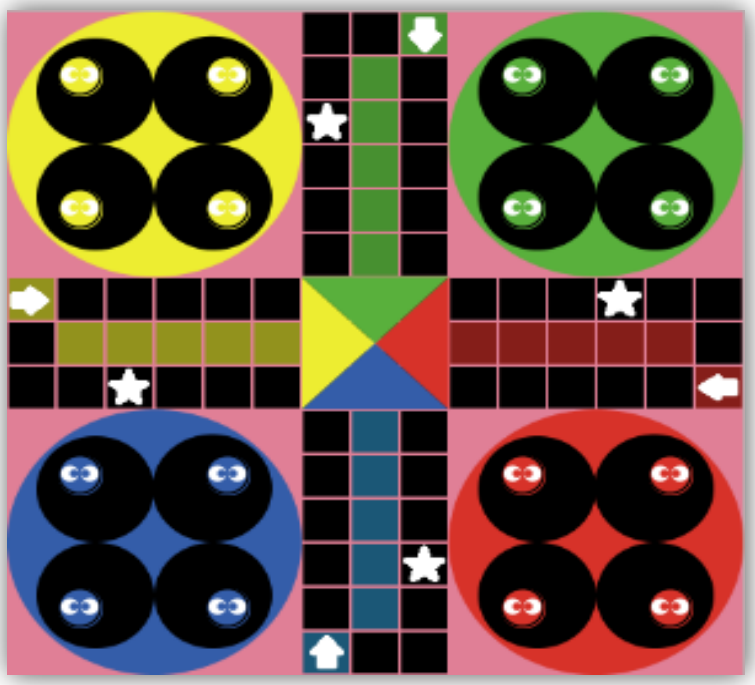

Figure 2.1 : L'environnement du jeu **Ludo**

Afin de mettre en œuvre notre méthodede planification de trajectoire, on a vu l'utilité de la modélisation du jeu d'environnement comme suit :

2.3.1 Proposition d'une planification de tâches pour le jeu Ludo

Dans un **SMA**, une méthode de planification de tâches peut-être définie comme un mécanisme automatique qui répond à la question «**quel agent ou ensemble d'agents doit être affecté pour résoudre la tâche ou l'ensemble de tâches concerné** ?». Dans le problème **Ludo**, les agents sont coordonnés selon le modèle organisationnel **AGR** (Agent-Groupe-Rôle). En fait, chaque agent appartient à un groupe spécifique et joue un rôle spécifique. Dans ce problème, il existe quatre groupes chacun est formé de quatre agents. Le rôle proposé par chaque groupe est la stratégie de planification de mouvement que les agents doivent suivre pour atteindre la cellule ciblée.

2.3.2 Proposition d'une Planification de mouvements pour le jeu Ludo

Afin de permettre les mouvements des agents, on a modélisé l'environnement selon les principes du**MDP**, Le chapitre précédent l'introduit, on a parlé de sa définition et de ses composants. A partir de cela, on peut modéliser l'environnement dans le tableau suivant :

Le composant MDP	**Sa signification reliée au problème Ludo**
L'espace d'états *S*	L'ensemble des cellules, chaque cellule occupée par un agent montre l'état de ce dernier. Le déplacement d'un agent d'une cellule à une autre détermine le changement de son état.
L'espace d'actions *A(s)* selon l'état *s*	Lorsque un agent se trouve à un état quelconque dans notre environnement, il peut uniquement faire les actions (passer à droite, à gauche, en haut et en bas) selon ce cas, parce qu'on a des cas admettent uniquement le passage à droite ou bien à gauche ou bien les deux

	uniquement etc…
La fonction de transition $P(s'\|s,a)$	Lorsqu'un agent est dans une cellule bien précise et veut passer à une autre cellule bien précise avec une action bien précise, le modèle va calculer la possibilité de ce passage ou cette transition. Plus précisément, elle montre si cette cellule admet l'accès de cet agent.
La fonction de récompense $r(s)$	Le gain obtenu par tel agent à raison d'être à une cellule. Chaque état s (cellule) lui donne un gain bien précis. Cette fonction exprime l'utilité d'un pion d'être à une position(une valeur numérique).
L'espace temporel T	Montre la durée de vie d'une partie, où chaque itération exprime un instant t de T.
La politique π	La stratégie à suivre pour choisir à quelle case le pion doit passer ?

Tableau 2.1 : Modélisation selon les principes **MDP**

En fait, les valeurs d'une fonction de récompense sont réparties dans chaque cellule de l'environnement en fonction du groupe de chaque agent. Cette fonction de récompense est proportionnellement inverse à la distance séparant la position de chaque agent de sa cellule cible en fonction du chemin à traverser. La distance entre l'agent et sa cible c'est la même que la distance entre la case occupant cet agent et la dernière case, sachant que les cases ou bien les cellules sont numérotées.Elle est calculée récursivement comme suit :

$$dist_i^k = \begin{cases} dist_{i-1}^k - 1 \\ dist_0^k = \alpha \end{cases}$$

α: Le nombre de cellules entre le début et la cellule cible concernant le groupe k

Supposons que l'agent est dans la position 2 et que la dernière position est 58, la distance entre cet agent et sa cible devient 56. Lorsqu'il passe à la position 3, sa distance va décrémenter par 1, donc elle devient 55. Ça veut dire qu'il s'approche de temps en temps au but. Si la distance devient 0, ça veut dire qu'il atteint son but.

La récompense dépend totalement de la distance, elle devient alors :

$$r_i^k = \alpha - dist_i^k$$

Prenons le même exemple : Si le même agent est dans sa même position (position 2), il reçoit une récompense de58 – 56 = 2.

De plus, chaque cellule contient un index montrant si la cellule est libre ou occupée par un autre agent (Ça facilite aux agents d'apprendre l'action suivante, par exemple il va décider de tuer l'agent qui occupe sa prochaine cellule). Les valeurs de la cellule peuvent être représentées par le vecteur [*x, y, z, w, n*], le tableau suivant détaille chaque variable :

La variable	Sa signification
X	cette variable montre la valeur de la récompense obtenue par les agents appartenant au premier groupe s'ils atteignent la cellule concernée. Elle augmente lorsque la distance entre l'agent et son but diminue. Dans notre environnement elle renvoi aux agents du groupe rouge. Si l'un de ces derniers occupe une cellule, il reçoit une récompense de sa valeur de *x*. Elle prend les valeurs de 1 jusqu'à 58, La valeur 1 pour la cellule de départ, la valeur 2 pour la deuxième cellule et ainsi de suite

	jusqu'à la valeur 58 affectée à la cellule cible. Les valeurs de cette variable sont statiques pour chaque cellule contrairement à la variable *n*.
y	Cette variable montre la valeur de la récompense obtenue par les agents appartenant au deuxième groupe s'ils atteignent la cellule concernée. Cette variable a le même principe que la variable *x*, sauf qu'elle renvoi aux agents du groupe bleu.
z	Cette variable montre la valeur de la récompense obtenue par les agents appartenant au troisième groupe s'ils atteignent la cellule concernée. Cette variable a le même principe que les variables *x* et *y*, sauf qu'elle renvoi aux agents du groupe jaune.
w	Cette variable montre la valeur de la récompense obtenue par les agents appartenant au quatrième groupe s'ils atteignent la cellule concernée. Cette variable a le même principe que les variable *x,y* et *z*, sauf qu'elle renvoi aux agents du groupe vert.
n	l'indice de cellule. Cette variable peut contient cinq valeurs différentes. Chaque cellule dans notre environnement peut prendre ces valeurs (changement dynamique), contrairement aux variables précédentes. Elles se changent comme suit :

	Si un agent du premier groupe (groupe rouge ici) l'occupe, elle prend automatiquement la valeur 1. Si un agent du deuxième groupe (groupe bleu ici) l'occupe, elle prend automatiquement la valeur 2. Si un agent du troisième groupe (groupe jaune ici) l'occupe, elle prend automatiquement la valeur 3. Si un agent du quatrième groupe (groupe vert ici) l'occupe, elle prend automatiquement la valeur 4. Aux autres cas, elle prend la valeur 0. L

Tableau 2.2 : Explication de valeurs du vecteur d'une cellule

Les instructions suivantes résument les possibilités de la variable *n :*

$$n = \begin{cases} 0\ \boldsymbol{if} empty(cell) = true \boldsymbol{or} neutral(cell) = true \\ 1\ \boldsymbol{if} agent \in \text{Group}_1 \textbf{and}\ \text{agent} \subset \text{cell} \boldsymbol{and} neutral(cell) = false \\ 2\ \boldsymbol{if} agent \in \text{Group}_2 \textbf{and}\ \text{agent} \subset \text{cell}\ \boldsymbol{and} neutral(cell) = false \\ 3\ \boldsymbol{if} agent \in \text{Group}_3 \textbf{and}\ \text{agent} \subset \text{cell}\ \boldsymbol{and} neutral(cell) = false \\ 4\ \boldsymbol{if} agent \in \text{Group}_4 \textbf{and}\ \text{agent} \subset \text{cell}\ \boldsymbol{and} neutral(cell) = false \end{cases}$$

Prenons par exemple le vecteur [25 12 51 38 0] d'une cellule quelconque, le chiffre 25 signifie la position d'un des agents du groupe rouge, si un agent du premier groupe l'occupe, il reçoit une récompense de 25. La même chose pour la valeur 12 concernant le deuxième groupe et ainsi de suite. En arrivant à la valeur 0 du vecteur, elle montre qu'aucun agent n'occupe cette cellule.

Si un agent du occupe cette cellule, son vecteur devient alors : [25 12 51 38 1] s'il est de premier groupe, [25 12 51 38 2] s'il appartient au deuxième groupe, [25 12 51 38 3] s'il appartient au troisième groupe et [25 12 51 38 4] s'il appartient au dernier groupe.

La Figure suivante montre la modélisation de l'environnement selon les principes **MDP** expliqués ci-dessus :

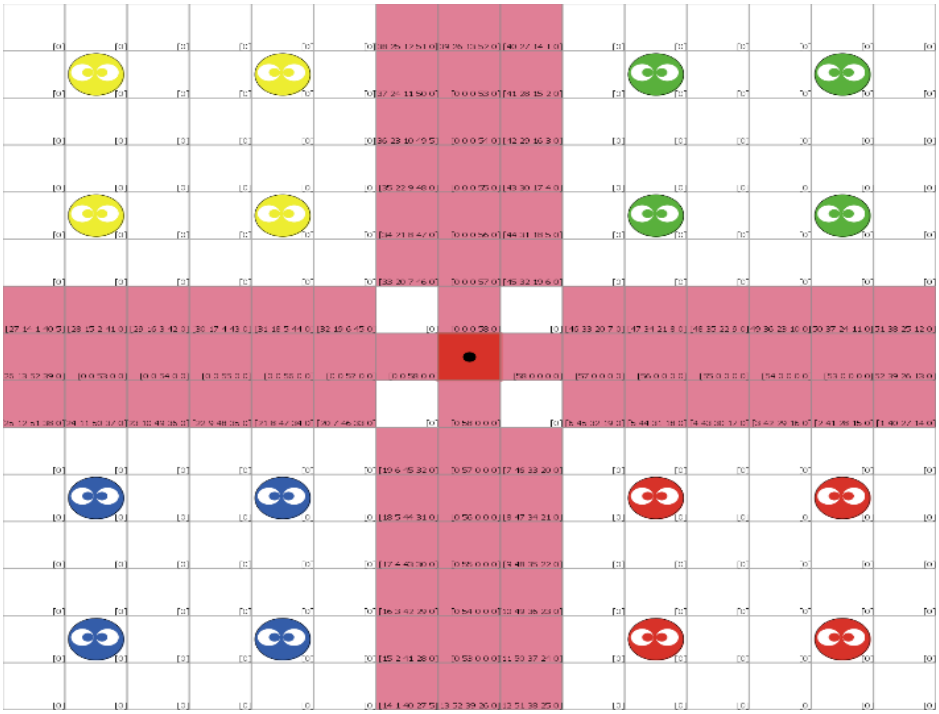

Figure 2.2 :L'environnement après la modélisation via les principes MDP

2.4 L'algorithme de planification de trajectoire

Dans cette partie, on présente le nouvel algorithme de planification de trajectoire du **SMA** proposé par son application au problème **Ludo**. Sachant que dans le jeu **Ludo**, l'agent est le seul qui peut se déplacer à chaque itération, la planification de trajectoire de l'agent va être ***centralisée***. En plus, on détaille les principaux objectifs de l'algorithme et explique également la nécessité de chaque étape de planification de mouvements :

Algorithm 1:

While end(game) = false **do**
For each group **do**
Use-dice($group_i$);
If Use-dice($group_i$) = 6 **then**
if killing($group_i$) = true **then**
Killing-best-choice($group_i$);
Ask-killed-agent(move-to(waiting-line));
Goto(line 3);
Else-if empty(waiting-line($group_i$)) = false **then**
Ask-one-of-$group_i$(move-to(initial-cell));

```
                    Goto(line 3);
        Else-if neutral(next-cell(group_i)) = true then
                    Move-best-neutral(group_i);
                    Goto(line 3);
        Else
                     Move-best-agent(group_i);
                     Goto(line 3);
        End-if;
        Else
        if killing(group_i) = true then
                    Killing-best-choice(group_i);
                    Ask-killed-agent(move-to(waiting-line));
                    Goto(line 3);
        Else-if neutral(next-cell(group_i)) = true then
                    Move-best-neutral(group_i);
        Else
                     Move-best-agent(group_i);
        End-if;
          End-if;
        Update(cells-index);
         counter ← 0;
For each agent do
        If  agent_i ⊂ Goal-cell then
        counter ← counter + 1;
        End-for;
        If counter = 16 then
          End(game) ← true;
       End-for;
End-while;
```

On a déjà dit que l'objectif de chaque groupe d'agents dans le problème **Ludo** est d'atteindre la cellule cible avant les autres groupes.

L'algorithme 1 montre la manière dont les agents du même groupe doivent coopérer pour réaliser cet objectif. Cet algorithme est basé sur la fonction de récompense générée par

l'application des principes **MDP**. Sachant que les groupes utilisent les dés tour à tour, chaque groupe doit prendre une décision pour répondre à la question : Quel agent va changer sa position ? Pour cela notre algorithme est basé sur quelques priorités on va les citer à partir de la plus importante :

- La première priorité : Renvoi au cas mortel. Précisément, Si les agents du même groupe ont la possibilité de tuer des agents concurrents l'algorithme teste s'ils y'a des agents victimes pour chaque prochaine cellule pour chaque agent de ce groupe. Et classe l'importance des cibles en fonction de leurs récompenses dans l'environnement.

  En d'autres termes, les agents doivent tuer l'agent avec la récompense la plus élevée (le plus proche de la cellule cible). Sachant que notre algorithme est basé sur la collaboration des agents. Pour retarder son adversaire autant que possible.

  Cette priorité se reflète dans l'algorithme en demandant aux agents de Tuer le meilleur choix.

  Killing-best-choice($group_i$);

  -Sinon s'il y'a une seule attaque il va l'effectuer.

  -Après l'algorithme demande à l'agent tué de retourner à sa case initiale en attendant un six.

  Cette action se reflète dans l'algorithme en demandant à l'agent tué de partir au départ.

  Ask-killed-agent(move-to(waiting-line));

  -si un agent tue un autre, le groupe du premier reçoit une chance de jeter le dé une autre fois de plus.

- la deuxième priorité : Est de Maximiser le nombre d'agents de même équipe pour maximiser les possibilités de jeu. Quatre pions mieux que trois, ça augmente la possibilité de tuer les autres, de se cacher et d'arriver à la cellule cibleavec un minimum de dégâts.

  Cette priorité se reflète dans l'algorithme en demandant aux agents de se déplacer vers la cellule de départ concernée jusqu'à ce que la ligne d'attente soit vide.

  Else-if *empty(waiting-line($group_i$)) = false* ***then***

  Ask-one-of-$group_i$(move-to(initial-cell));

- La troisième priorité : Est d'éviter la mort de ses agents (éviter d'être tué). Pour cela, les agents doivent se déplacer vers la meilleure cellule neutre possible pour se cacher.

La meilleure cellule neutre est la cellule neutre renvoyant le degré de récompense le plus élevé.

Cette priorité se reflète dans l'algorithme en demandant aux agents d'aller à la meilleure cellule neutre.

Else-if *neutral(next-cell(group$_i$)) = true* ***then***

Move-best-neutral(group$_i$);

Sinon s'il y'a une seule cellule neutre il doit l'occuper.

- Si les conditions des priorités précédentes ne sont pas remplies, l'agent situé dans la cellule renvoyant le plus haut degré de récompense doit déplacer.

Move-best-agent(group$_i$);

S'il y'a un seul agent dehors la ligne d'attente, il va se déplacer automatiquement.

La formule suivante montre comment calculer la récompense maximale :

$$Q_{i+1}(s, a) = (1 - \alpha)Q_i(s, a) + \alpha\left[r + \gamma \max_{a'}[Q_i(S_{ag1}, a'), Q_i(S_{ag2}, a''), Q_i(S_{ag3}, a'''), Q_i(S_{ag4}, a'''')]\right]$$

Sachant que l'environnement **Ludo** est déterministe alors α = 1, la formule devient :

$$Q_{i+1}(s, a) = r + \gamma \max_{a'}[Q_i(S_{ag1}, a'), Q_i(S_{ag2}, a''), Q_i(S_{ag3}, a'''), Q_i(S_{ag4}, a'''')]$$

Les deux **Figures** suivantes (**Figure 2.3 et 2.4**) montrent la priorité d'un cas mortel (avant et après la tuerie), où le jeu donne la priorité à l'agent bleu qu'est loin de l'objectif, pour tuer l'agent vert avec une valeur 4 de dé.

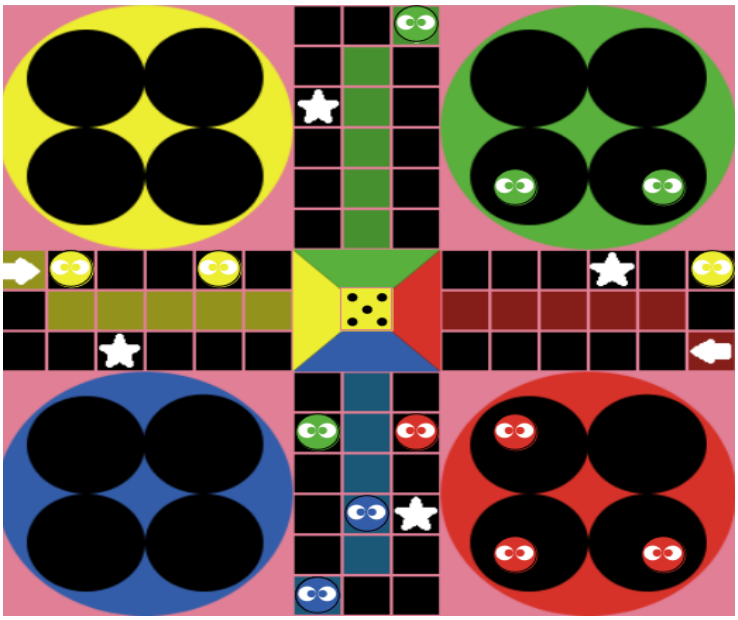

Figure 2.3: l'environnement avant la tuerie.

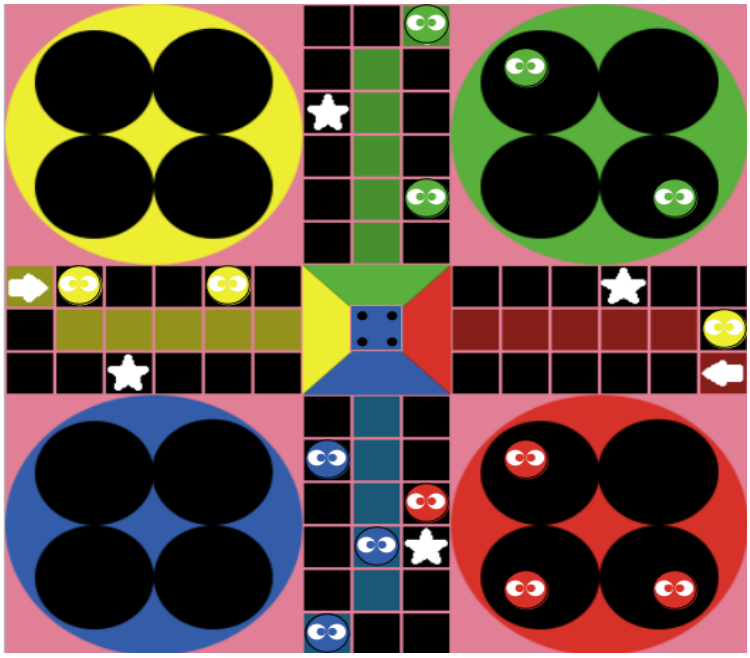

Figure 2.4: l'environnement après la tuerie.

En répondant à la question précédente Quel agent va changer sa position ? Passant aux autres principes du jeu :

- Avant de passer les dés d'un groupe à un autre, l'algorithme vérifie si tous les agents de ce groupe ont déjà atteint la cellule cible. Si cette condition est remplie, le jeu est considéré comme terminé pour ce même groupe. Après ça, le groupe transmet

automatiquement les dés au groupe suivant. Cette priorité est vérifiée selon l'équation suivante:

$$Max_r = \frac{1}{4} \times \sum_{i=1}^{4} r_s^{Ai}$$

On peut terminer la partie pour un groupe si et seulement si Max_r égale à 58. Parce qu'on a 58 cases au chemin de chaque groupe. C-à-d, l'agent doit traverser 58 cellule pour y'arriver à son but.

La **Figure 2.5** représente l'environnement **Ludo** lors de la fin du jeu pour l'équipe jaune.

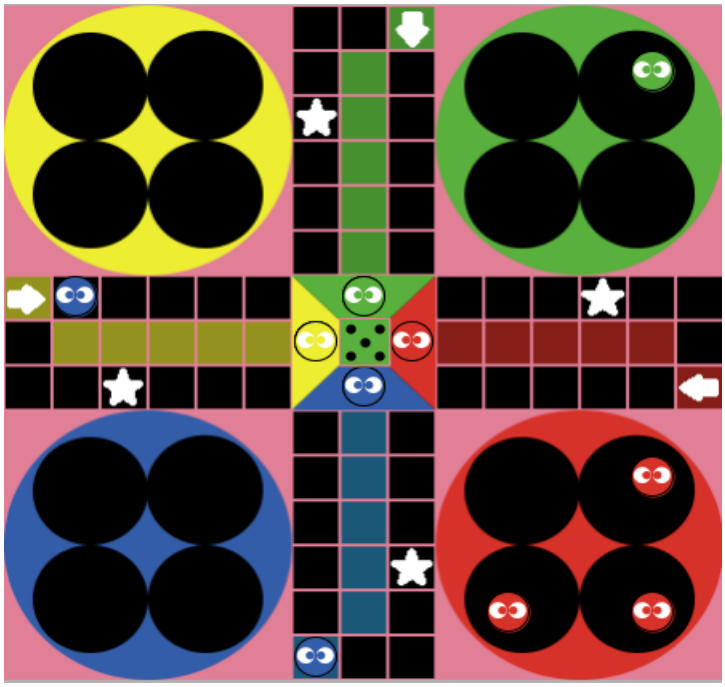

Figure 2.5 : Fin de la partie pour un groupe.

- Finalement, la partie de jeu sera terminée si tous les agents de tous les groupes atteignent les buts. L'équipe qui gagne en premier joue un nombre minimal d'itération, puis la deuxième et ainsi de suite jusqu'à l'équipe perdante qui un nombre d'itérations maximal. cela indique la nécessité de mettre un compteur pour chaque groupe, qui va incrémenter à chaque itération.**La Figure** suivante représente l'environnement après la fin d'une partie :

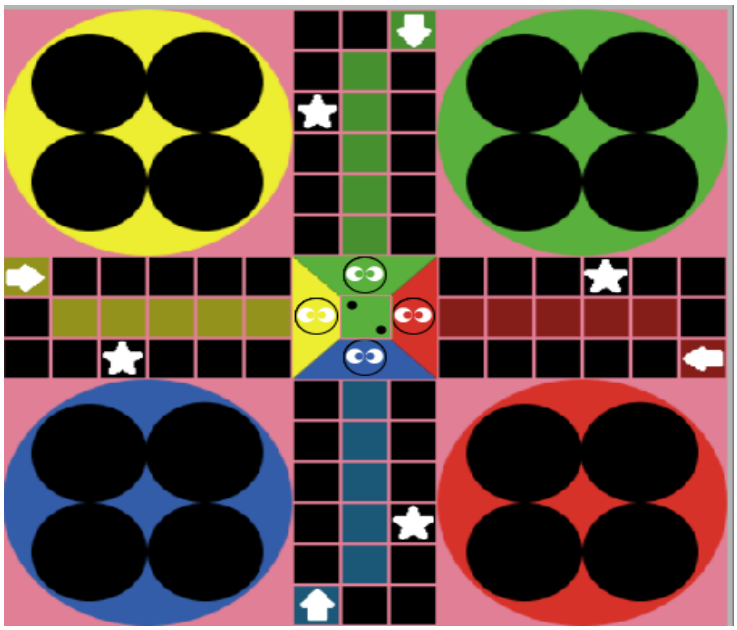

Figure 2.6 : Fin d'une partie **Ludo**

2.5 Conclusion

Après avoir bien détaillé le problème du jeu **Ludo** ainsi que notre solution proposée dans ce chapitre, nous allons pouvoir l'implémenter en comparaison avec un des travaux connexes dans le chapitre suivant via l'utilisation de la plateforme **Netlogo** [32].

Chapitre 3

Expérimental

Contenu

3.1 Introduction

Dans ce chapitre, nous allons lever l'ambiguïté sur quelques plateformes multi-agent telles que **Madkit** et **Jade**. Aussi nous allons argumenter notre choix concernant la sélection de la plateforme **Netlogo** pour effectuer nos simulations. Durant ces simulations, nous allons étudier le temps en nombre itérations concernant l'exécution des taches (atteinte des cellules finales pas les agents), et aussi le développement des agents (acquisition des récompenses) durant leurs déplacements. Tous ces paramètres seront étudiés en comparaison avec un travail connexe basé sur le Q-learning [33].

3.2 Plateformes multi-agents

Il existe plusieurs types des plateformes multi-agents :

« Une liste assez complète de ces plates-formes se trouve à l'adresse « 170H170H191H170H170Hhttp://www.agentlink.org/ ». Parmi les plates-formes fournies comme logiciels libres, il y a quelques-unes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, MACE, ZEUS, et MADKIT … SWORM ... Il faut noter que cette liste n'est pas unique, et qu'il y a aussi d'autres plates-formes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications »[34].

Ça montre que la diversité de plateformes dépend du problème même, de sa conception. Parmi les plateformes les plus utilisées, nous citons :

3-1-1-La plateformeJADE

JADE pour (Java Agent Development Framework).C'est une plateforme open source, écrite en java. Elle permet de construire des systèmes multi agents (**SMA**) créée par le laboratoire TILAB. Elle facilite le développement à l'aide de son architecture simple.

La **Figure 3.1** montre brièvement l'architecture **JADE.**Une plateforme JADE est constituée d'un conteneur spécial (main container), ce dernier gère plusieurs conteneurs. Chaque conteneur contient plusieurs agents où ce dernier est également une instance **JADE**. L'**AMS** (Agent Management System) et le **DF** (Directory Facilitator) sont des agents spéciaux, le premier nomme est supervise les autres agents, et le deuxième fournit un système de pages jaunes qui référence le fournisseur de service aux agents.

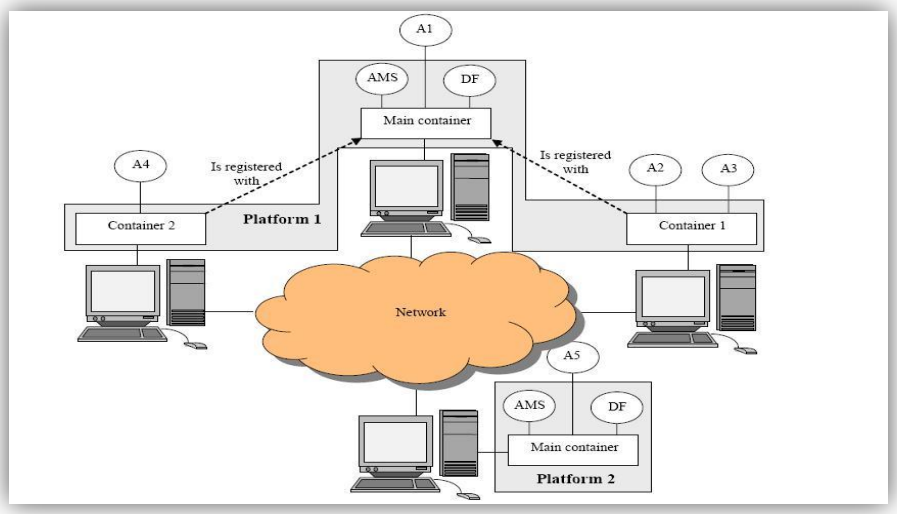

Figure 3.1 : l'architecture JADE

3.2.2 La plateforme MADKIT

MadKitpour (Multi-Agent Development Kit)est une plate-forme agent flexibleimplémentée en java, elle utilise le modèle organisationnel **AGR (**défini dans le chapitre 1**)** pour ses différentes taches de développement des agents, plutôt elle utilise une architecture d'agents ou modèle d'interaction spécifique.

La plateforme **MadKit**a le même principe que le modèle. Elle supporte plusieurs modèles de communication simultanément. On peut la considérer comme un ensemble de packages JAVA.

La **Figure 3.2** résume l'architecture **MadKit.** Cette plateforme est constituée d'un composant principal appelé Le ***micro-noyau agent*** c'est l'infrastructure ou l'environnement qui affecte et gère les rôles, responsable à la communication (échange de messages entre agents d'un même modèle ou de plusieurs modèles différents). Les agents sont des ***Threads JAVA***, qui évoluent d'une manière asynchrone. Ils sont devisés en ***Agents applicatifs*** (les agents artificiels qui subissent la gestion de leurs cycles de vie). Et en ***Agents système*** (des composants graphiques).

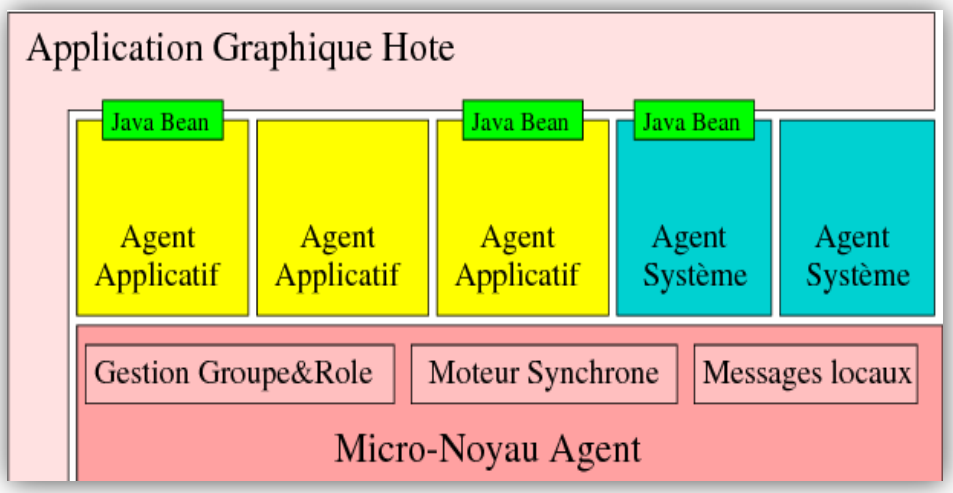

Figure 3.2 : l'architecture MadKit

3.2.3 La plateforme ZEUS

Est une plate-forme multi-agents, « **Zeus est un environnement intégré pour la construction rapide d'applications à base d'agents collaboratifs. Il est développé par l'Agent Research Program du British Telecom Intelligent System Research Laboratory** »[35].

Cette plateforme est écrite sur le langage **JAVA**, elle est basée sur la notion du rôle et utilisée dans les domaines qui nécessitent la coordination des agents,Dont chaque agent joue son rôle de contrôler le système externe à l'aide des autres agents en utilisant une ontologie du langage pour qu'elles soient réutilisées. On peut noter qu'elle est composé de :

- **Une boîte aux lettres et un gestionnaire de messages :** Comme indiquant les noms, ils gèrent la transmission des messages entre les agents ou les composants.
- **Un moteur de coordination :** C'est lui qui prend les décisions en charge**.**
- **Un planificateur :** C'est lui qui planifie les taches en basant sur les décisions du moteur de coordination**.**
- **Bases de données :** représentant les déroulements connus par les agents. Et les ontologies créées.
- **Gestionnaire d'horloge :** appelé également un contrôleur, c'est lui qui gère les activités et les taches fournies par les agents.

3.3 Plateforme de simulation NetLogo

Les plates-formes de simulation qui permettent un prototypage et des tests rapides des idées de conception initiales et des alternatives, peuvent améliorer le processus d'implémentation de systèmes complexes composés de milliers d'agents agissant en parallèle et multi-agents (**SMA**). Elles permettent aussi au développeur de concevoir son application sans avoir à une étude approfondie des **SMAs**.

Une telle plateforme idéale doit être caractérisée par la facilité d'apprentissage, une implémentation et affichage simples du **SMA**, Prendre en compte les fonctionnalités de programmation orientées agent qui permet de réaliser les alternatives de conception en un environnement de développement d'agent facilement. Cependant, ces exigences spécifiques font d'une telle plate-forme de simulation un outil d'apprentissage parfait.

NetLogoa été créé par le Dr Uri Wilensky au Centre d'apprentissage connecté et de modélisation informatique (**CCL**), situé à l'université de Tufts, Medford, MA. En 2000, le **CCL** a été transféré à l'université de Northwestern, Evanston, IL, où le traitement de **NetLogo** est poursuivi jusqu'à présent.

NetLogo est un langage de programmation multi-agents simple et primitif basé sur le langage de programmation ***Logo***. Cette plateforme est distribuée en open source dans sa version actuelle.

La conception et le développement de **NetLogo** ont été réalisés dans le but de modéliser et simuler des systèmes complexes évoluant dans le temps. Il a été utilisé pour concevoir plusieurs systèmes complexes dans différents domaines tels que l'économie, la biologie, la physique, la chimie, la psychologie, la dynamique des systèmes et de nombreux autres phénomènes naturels et sociaux.

Le **NetLogo** peut modéliser Les agents mobiles et immobiles. Il traite plusieurs agents appartenant au même environnement physique et peut également inclure des milliers d'agents dans la même simulation. En plus, il existe d'autres capacités pour lier les agents et de les mettre en réseau, des options à la fois bidimensionnelles et tridimensionnelles (2D et 3D) concernant les modes d'affichage, et une commutation facile entre l'exécution en une seule étape et les simulations continues. Les caractéristiques qui distinguent **NetLogo** peuvent être résumées comme suit :

- **Documentation claire et complète** : contenant un ensemble large et riche d'exemples (exemples de modèles) concernant différents domaines pour guider les objectifs des utilisateurs.
- **Une interface utilisateur graphique (GUI)** : facile à comprendre, elle compose d'un ensemble d'outils simples pour produire un modèle, le superviser et contrôler son comportement.
- **Langage de modélisation** : Cette plateforme offre un langage de modélisation facile à apprendre, puissant et flexible pour la création de modèles.
- **Un outil de simulation participative «HubNet»** : permettant à plusieurs utilisateurs exécutant des programmes clients distincts d'interagir via un modèle **NetLogo**. En plus, il contient plusieurs outils qui facilitent l'exécution de plusieurs programmes en arrière-plan.
- **Espace de comportement** : un outil qui facilite l'exécution d'un modèle plusieurs fois grâce à l'utilisation de différentes entrées pour étudier les impacts de scénarios alternatifs.
- **Éditeurs d'icônes et capacités d'importation pour les liens et les agents**.
- **Un environnement distinct :** équipé d'outils appropriés pour développer graphiquement les modèles de systèmes dynamiques au lieu de modèles basés sur des agents.

Au cours de notre travail, nous avons vu l'utilité d'utiliser **NetLogo** dans le but de décrire l'environnement du **jeu Ludo**ainsi que le processus de coopération des d'agentspour atteindre l'objectif.

En effet, cette plateforme offre la possibilité de créer différents types d'agents qui se distinguent par plusieurs caractéristiques telles que le degré de mobilité définissant la vitesse du mouvement et les chemins disponibles à suivre, la forme et la couleur définissant les groupes lors de processus du jeu.

En plus, l'environnement des agents proposé dans **NetLogo** plus connu sous le nom de (**Patches**) correspond parfaitement à notre environnement de grille de cellules du jeu **Ludo**. En d'autres termes, chaque patch caractérisé par des coordonnées cartésiennes est considéré comme une cellule pouvant être occupée par un agent. Les couleurs et les formes des Patches sont modifiables nous permettant de définir la ligne de départ, le circuit, les cellules neutres, le but ainsi que les bords d'environnement.

La **Figure 3.3** montre l'écran d'interface **NetLogo** pour notre simulation, en cliquant sur le bouton débuter (renvoi à la procédure débuter) on obtient l'état initial du jeu. Le bouton jouer renvoi au changement des états durant la partie. La partie (command center) montre les résultats de la simulation (résultats des fonctions) et les valeurs des variables (à chaque états si demandées). Le code de procédures est écrit dans l'onglet code.

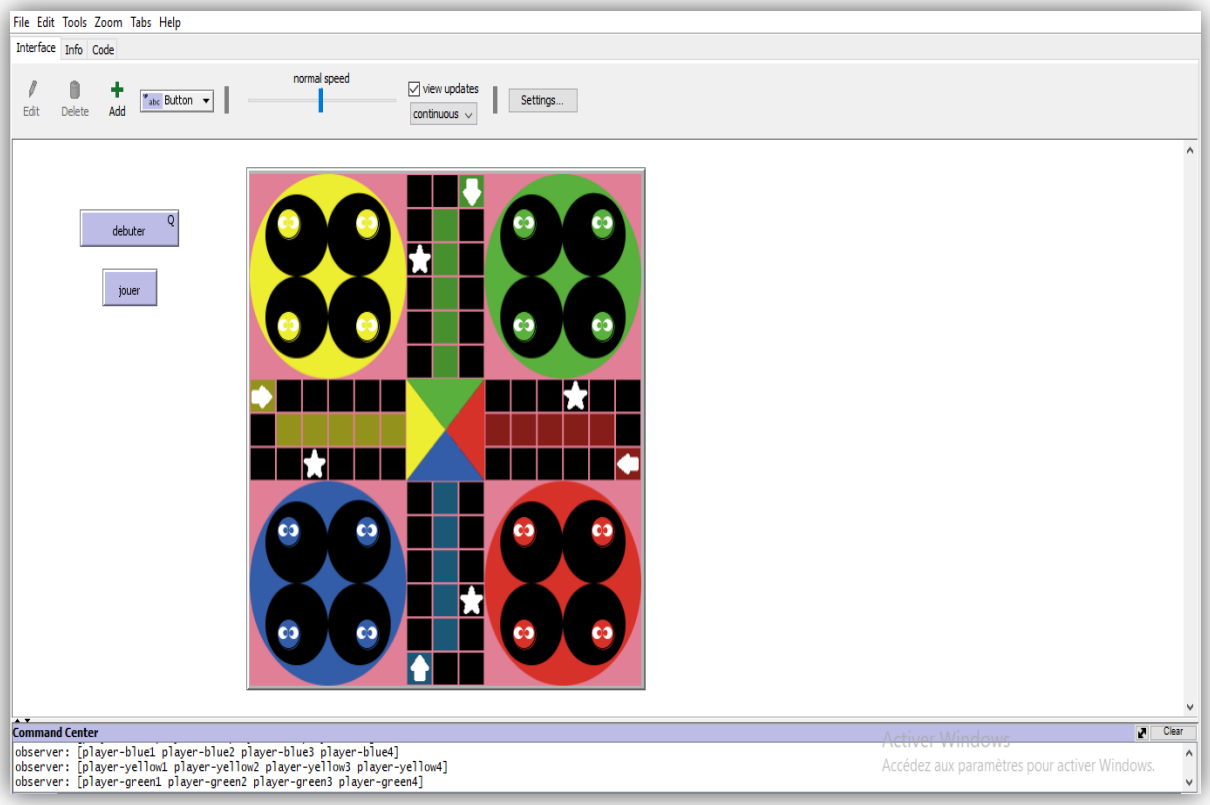

Figure 3.3 : L'écran d'interface **NetLogo** pour notre simulation

3.4 Résultats de la simulation

Afin de résoudre le problème Ludo, on a vu l'utilité d'utiliser la plate-forme **NetLogo**. Cette plateforme orientée Agent propose des méthodes prédéfinies qui facilitent notre implémentation. A partir de la **Figure 3.3**, on peut noter qu'on réalise notre jeu Ludo sur un exemple se déroulant dans une grille rectangulaire bidimensionnelle de 15 ×15 cellules.

Chaque agent doit parcourir un circuit de 58 cellules pour y'arriver à sa cellule cible. Ce circuit d'agents est composé de 14 cellules neutres et 44 cellules ordinaires (cellules noires).

En ce qui concerne les agents environnementaux, nos simulations sont basées sur 16 Agents, quatre groupes d'agents colorés avec une même couleur. Ces derniers sont totalement

capables de déterminer leurs victimes, et les victimes ont renvoyésau début. Au cours de cette simulation, nous avons comparé deux cas différents présentés comme suit :

- **Cas A ou (Case A) :** Ce cas reflète le groupe d'agents utilisant la planification coopérative de trajectoire proposée dans le chapitre précédent. Brièvement, les agents ici s'unissent les uns aux autres pour accomplir leur mission.
- **Cas B ou (Case B) :** Ce cas reflète le groupe d'agents utilisant la stratégie **greedy**(dite égoïste) ou essayant de maximiser les récompenses attendues afin d'atteindre l'objectif (chacun essai de maximiser son propre gain). Cette stratégie est basée sur des agents égoïstes. On peut résumer cette stratégie comme suit :

Chaque agent utilise les dés de manière répétitive sans les remettre aux autres agents appartenant au même groupe jusqu'à ce que l'agent atteigne son but.

Donc le deuxième agent (d'un groupe) ne peut pas sortir au départ tant que le premier (de ce même groupe) est en train de jouer.

Après, on a utilisé deux types de dés différents. C-à-d, chaque stratégie est simulée en utilisant les deux différents dés. Le tableau suivant montre la différence entre le type 1 et le type 2.

Type 1	**Type 2**
Le premier dé (**dés ordinaires ou aléatoires**) renvoie un nombre aléatoire compris entre 1 et 6 lors de chaque itération de jeu. Les dés aléatoires sont utiliser à fin d'augmenter les chances des (gains/défaites).	le deuxième dé (**dés truqués**) renvoie la même séquence de nombres pour les quatre groupes participants. Les dés truqués sont utilisés afin d'éviter le problème lié au tirage au sort, et également pour assurer une certaine équité entre les groupes.

Tableau 3.1 : Les deux différents types de dés

Donc on a 4 simulations :

- Simulation de la stratégie ***coopérative*** en utilisant des dés ***ordinaires***.
- Simulation de la stratégie ***égoïste*** en utilisant des dés ***ordinaires***.
- Simulation de la stratégie ***coopérative*** en utilisant des dés ***truqués***.
- Simulation de la stratégie ***égoïste*** en utilisant des dés ***truqués***.

Les résultats de la simulation sont basés sur l'étude de deux groupes où chacun utilise l'une des stratégies de trajectoire comparées **(cas A, cas B)** comme suit **:**

- **Résultat 1 : La Figure 3.4** représente la durée moyenne d'une partie à l'aide d'un dé aléatoire pendant 50 épisodes de partie. La durée est calculée par rapport au nombre d'itérations des groupes. Une itération de groupe peut être définie comme le nombre d'utilisation des dés jusqu'à ce que tous les agents de ce groupe atteignent la cellule cible.

La durée moyenne du jeu dans le **Cas A**(Case A) diminue jusqu'à 29,23% par rapport au **Cas B** (Case B). Ce fait est dû au comportement intelligent des agents fourni par la planification collaborative de trajectoire proposée. En d'autres termes, la priorité visant à vider la file d'attente permet l'augmentation du taux de collaboration entre les agents (l'augmentation de la possibilité de décisions), contrairement aux agents égoïstes qui perdent les chances à finir en durée minimale.

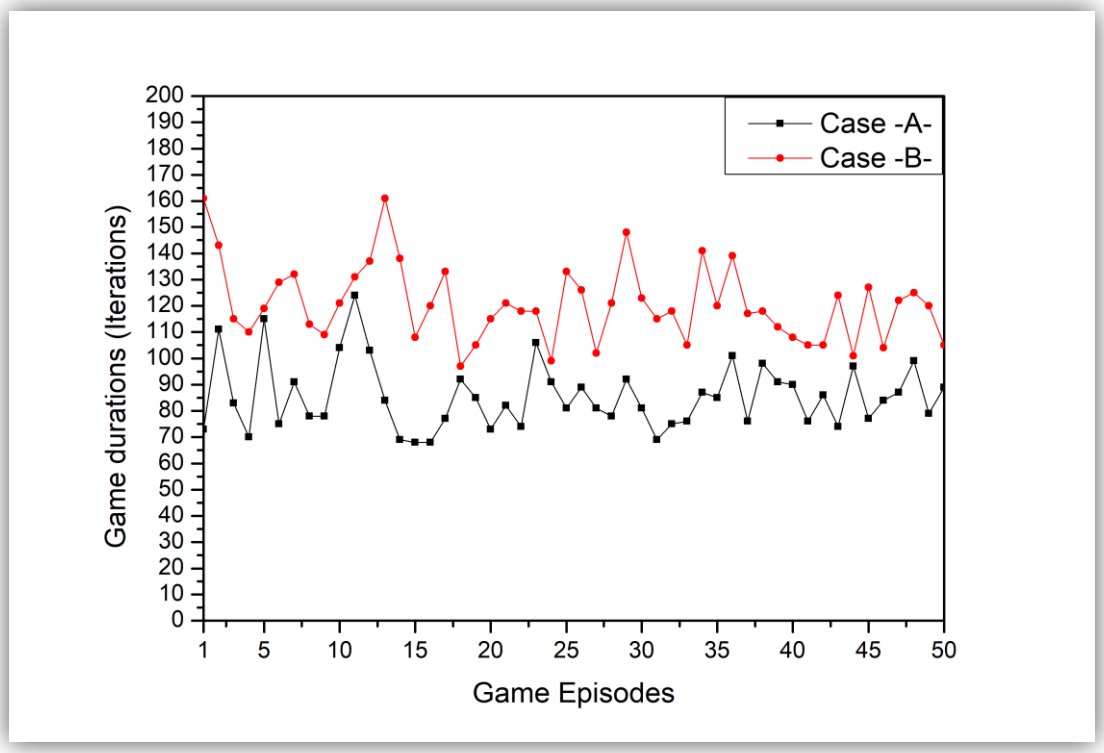

Figure 3.4: durée moyenne du jeu à l'aide d'un dé aléatoire

- **Résultat 2** : **La figure 3.5**représente le développement de la récompense moyenne des agents au cours d'un épisode du jeu complet à l'aide d'un dé aléatoire (par rapport aux itérations). On peut noter que les agents du **Cas A**(Case A) atteignent la récompense

moyenne maximale après 75 itérations du jeu.(Sachant que l'itération du jeu est obtenue à chaque utilisation de dés).

Cependant, dans le **Cas B**(Case B) : les agents obtiennent la récompense maximale après 110 itérations.

De plus, nous pouvons noter deux diminutions importantes concernant le **Cas B** (itération 11 et itération 46). Ces diminutions signifient que les agents utilisant la planification collaborative de trajectoire ont effectué deux tueries au cours de cet épisode de jeu. Ce fait présente les avantages de la priorité de destruction de l'algorithme proposé. Cette priorité provoque un certain déséquilibre dans le groupe adversaire entraînant un retard important dans l'exécution de la tâche.

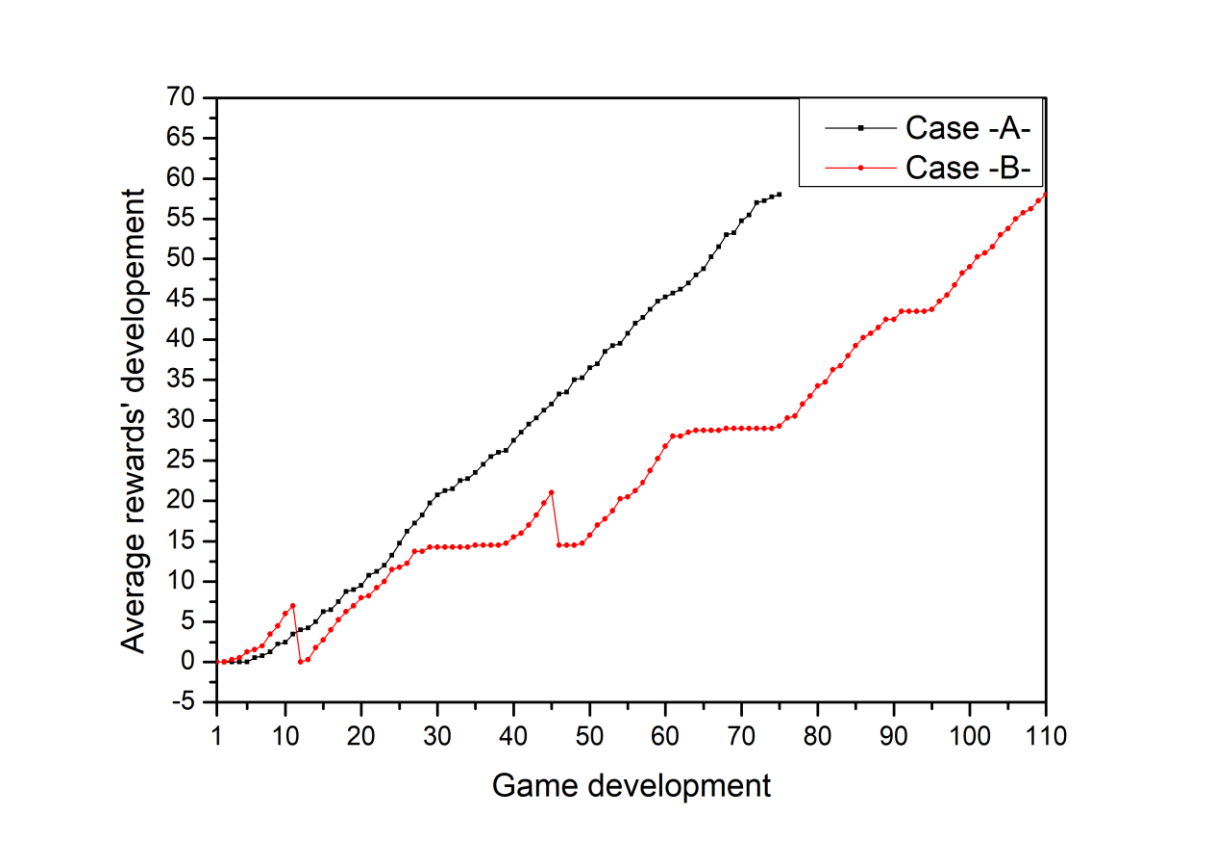

Figure 3.5 : Evolution moyenne des récompenses au cours d'un épisode de jeu

- **Résultat 3** : **La Figure 3.6** reflète la durée moyenne du jeu dans cinq épisodes de jeu à l'aide de dés truqués. Dans chaque épisode, les dés truqués génèrent une séquence répétitive de nombres (comme montrant **le tableau 3.2**), la durée moyenne diminue jusqu'à 24,34% dans le **Cas A** Par rapport à la durée moyenne de jeu obtenue dans le

Cas B. Ce résultat confirme les résultats de la**Figure 3.4**. on a déjà dit que la priorité visant à vider la file d'attente permet l'augmentation du taux de collaboration entre les agents (l'augmentation de la possibilité de décisions) c'est-à-dire, grâce à l'intelligence de agents du **Cas A**.

Episodes de jeu	Séquence des nombres
1	[1, 2, 3, 4, 5, 6]
2	[6, 3, 5, 2, 1, 4]
3	[5, 6, 2, 1, 4, 3]
4	[1, 3, 6, 4, 5, 2]
5	[3, 4, 6, 5, 1, 2]

Tableau 3.2 : les séquences des nombres obtenues à l'aide d'un dé truqué

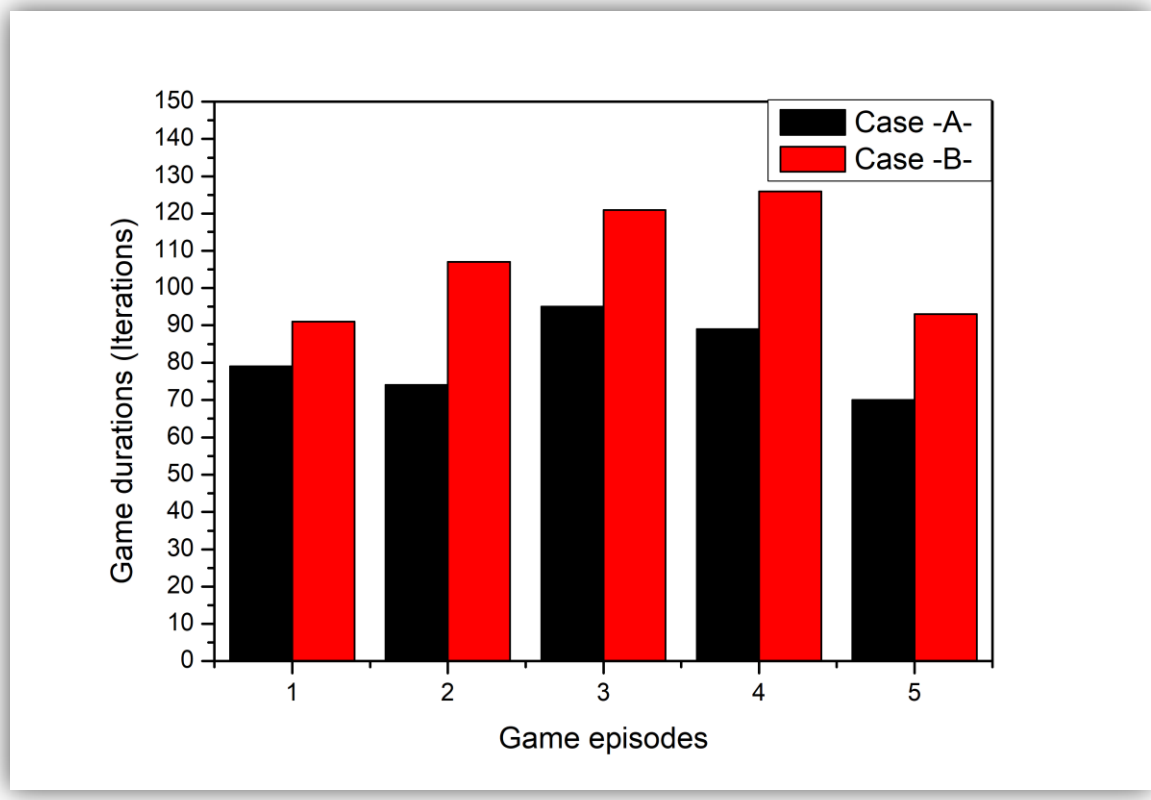

Figure 3.6 : durée moyenne du jeu à l'aide d'un dé truqué

- **Résultat 4** : La **Figure 3.7** montre les récompenses moyennes obtenues par les agents par itération au cours d'un épisode de jeu complet.

Dans ce cas, la séquence des nombres de dé truqué utilisée dans l'épisode 2 (du **tableau 3.2**) est utilisée. La récompense moyenne par itération est calculée comme suit:

$$\Omega_s = \frac{(r_s^{A1} + r_s^{A2} + r_s^{A3} + r_s^{A4}) - (r_{s-1}^{A1} + r_{s-1}^{A2} + r_{s-1}^{A3} + r_{s-1}^{A4})}{4}$$

$$\Omega_s = \frac{1}{4} \times \sum_{i=1}^{4} (r_s^{Ai} - r_{s-1}^{Ai})$$

Dans le **Cas B** de cette même figure, on peut noter trois diminutions négatives des itérations 12, 21 et 26 signifiant que les agents du **Cas A** ont effectué trois attaques lors de cet épisode de jeu.

De plus, la récompense moyenne obtenue dans le **Cas A** est de 0,84. Cependant, cette moyenne diminue jusqu'à 0,29 dans le **Cas B**.

Alors, les résultats montrés dans cette figure valident les résultats montrés sur la **Figure 3.5**, dans laquelle les récompenses sont basées sur un dé aléatoire.

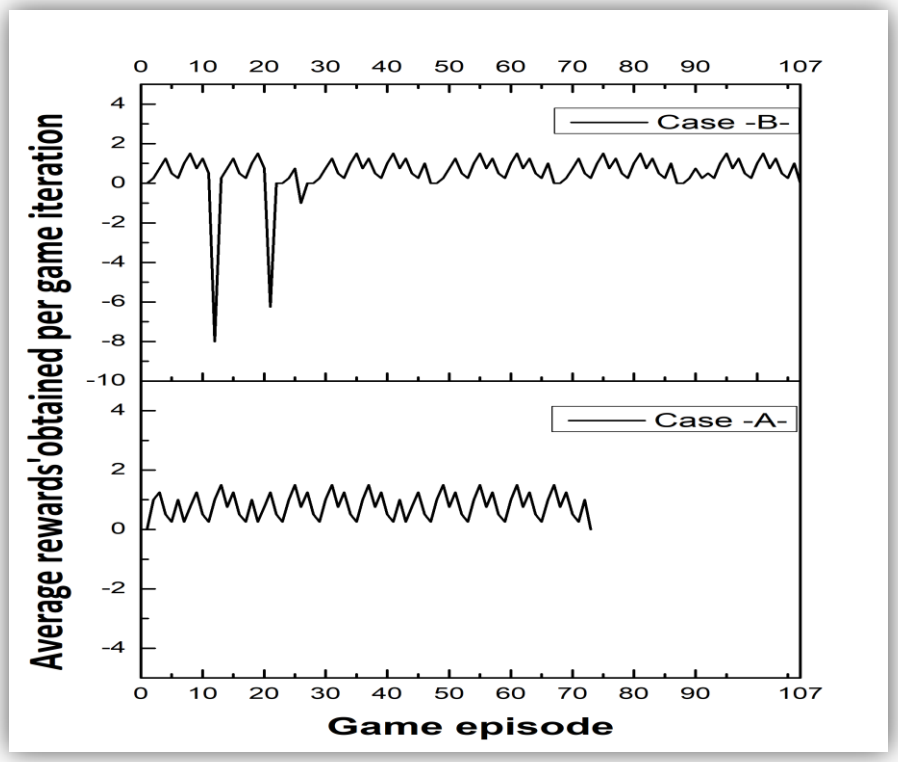

Figure 3.7 :récompense moyenne obtenue par itération pendant un épisode de jeu complet

Le **tableau 3.3** résume tous les résultats obtenus lors de la partie simulation. A partir de ces résultats, on peut facilement remarquer la supériorité de la planification collaborative de trajectoire proposée par rapport à la stratégie égoïste. Cette supériorité concerne le développement des agents au cours du jeu (acquisition de récompenses) ainsi que la durée du jeu (temps d'exécution des tâches).

	Durée du jeu en utilisant des dés ordinaires (itérations)	**Durée du jeu à l'aide de dés truqués (itérations)**	**Récompense moyenne obtenue par itération en utilisant des dés ordinaires**	**Récompense moyenne obtenue par itération à l'aide de dés truqués**
Cas A	85.44	81.4	0.75	0.84
Cas B	120.74	107.6	0.093	0.29

Tableau 3.3 : Résultats obtenus lors de la simulation

3.5 Conclusion

Grâce à notre implémentation, nous pouvons dire qu'il est très important de faire le bon choix concernant bonne plateforme à utiliser pour obtenir des résultats fiables. A travers les résultats reflétés dans ce chapitre, nous pouvons dire que notre nouvel algorithme collaboratif a permis l'obtention d'un temps d'exécution adéquat ainsi qu'un très bon degré de développement des agents durant l'exécution des taches en comparaison un travail récent et connexe.

CONCLUSION

Ce mémoire avait pour ambition de réaliser une planification de la trajectoire basée sur les principes **MDP** dans le contexte du jeu **Ludo** possédant un environnement multi-agents permettant de mettre en valeur l'aspect coopération entre les agents appartenant au même groupe et aussi l'aspect concurrence entre les agents adverses. Les solutions ne doivent pas forcément être optimales mais proposer un résultat convenable par rapport aux principes de jeu.

Pour réaliser ce mémoire, nous avons d'abord rassemblé le plus de documents concernant le sujet. La planification de trajectoires étant un domaine relativement récent, malheureusement, il existe peu de livres liés à ce sujet, on a aussi le problème de confinement et heureusement, on a trouvé des thèses, articles et autres papiers sont disponibles à ce sujet.

La deuxième étape consiste à introduire le sujet à l'aide de documentations en précisant les grandes lignes et les notions basiques pour pouvoir réaliser la troisième étape qui introduit la nouvelle planification collaborative proposée et basée sur les gains générés par l'application des principes **MDP**. Cette dernière vise à augmenter le degré de coopération entre les agents en définissant un ensemble de priorités que les agents doivent respecter lors de l'exécution des tâches. En proposant un algorithme vise à refléter l'acquisition de récompenses ainsi le nombre d'itérations.

Cette application a été effectuée en comparaison avec la stratégie égoïste suivie par un autre groupe d'agents. Au cours des simulations, nous avons étudié le développement de l'acquisition des récompenses ainsi que la durée d'exécution des tâches via le nombre d'itérations. De plus, nous avons utilisé deux types de dés (ordinaires et truqués) pour éviter le problème lié au tirage. Les simulations sont créées sous la plateforme **NetLogo 6.0.4**.

Enfin, on représente les résultats. Ces derniers reflètent la différence entre les deux stratégies comparées, avantageusement pour la nouvelle planification de trajectoire proposée.

Pour conclure, nos perspectives concernant le sujet étudié sont que les pions (agents) **Ludo** pourront jouer avec l'être humain même et proposer ou changer leurs propres stratégies selon le comportement de l'adversaire, et qu'ils pourront aussi apprendre en cours de jeu afin de gagner une partie avec un humain facilement. De plus, notre planification pourra être appliquée dans le domaine de la robotique où les robots feront une course et tendront des pièges pour entraver les autres et gagner en premier.

Références bibliographique

[1] Romaric Charton, « Des agents intelligents dans un environnement de communication multimédia : vers la conception des services adaptatifs », Thèse de doctorat en informatique, sous la direction de Jean-Paul Haton, Nancy, HenréPoincarré, 2003, 200p.

[2] Ferber j, « les systèmes multi-agents : un aperçu général », technique et science informatique, Vol.16, n°8,1997, pp.979-1012.

[3] Voir le site: http://beefchunk.com/documentation/ia/cours_IA/node123.html , consulté le : 21/04/2020 à 15 h 13.

[4] Nachet Bakhta, « modèle multi-agents pour la conception de systèmes d'aide à la décision collective », Thèse de doctorat en informatique, sous la direction d'Adla A, Oran, université d'Oran, 2013, 155p.

[5] ReguiegSeddik, « Elaboration d'un protocole de coordination dans un SMA pour la gestion de production dynamique : vers une approche décentralisée », Mémoire du magister en informatique, sous la direction de [M.Senouci] [N.Taghezout], Oran, université d'Oran, 2010, 117p.

[6] Fabien,G., &Masayuki,I. La planification de taches et de mouvements pour un robot humanoïde dans une cuisine, Université de Tokyo, 7(1), 3-1.

[7] Olivier Gutknecht, « proposition d'un modèle organisationnel générique de système multi-agents : Examen de ses conséquences formelles, implémentation et méthodologiques », Thèse de doctorat en informatique, sous la direction de [J.Briot] [L.Grasser], Montpellier, université Montpellier 2, 2001, 220p.

[8]Voir le site:
http://theses.univlyon2.fr/documents/getpart.php?id=lyon2.2008.lanadecarvalho_l&part=149928 , consulté le : 21/12/2019 à 21 h 11.

[9] Sébastien Dalibard, ‘’planification de mouvements pour systèmes anthropomorphes’’, thèse de doctorat en informatique, sous la direction de Jean-Paul Laumond, Toulouse, université de Toulouse, 2011, 107p.

[10] Rémi Munos. Programmation dynamique avec approximation de la fonction valeur. Processus décisionnels de Markov et intelligence artificielle, Hermes, pp.19-50, 2008. ffhal-00830192f.

[11] Emmanuel Rachelson, ‘’problèmes décisionnels de Markov Temporels Formalisation et Résolution’’, thèse de doctorat en Systèmes embarqués, sous la direction de [F.Garcia] [R.Munos] , Toulouse, université de Toulouse, 2009, 83p.

[12] Abdelmoumène Hiba, « Contribution à la Résolution des Processus Décisionnels de Markov Décentralisés», Thèse de Doctorat en informatique, sous la direction de Blleili-Souici Habiba, Annaba, Badji Mokhtar, 2016, 158p.

[13] Voir le site: https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.researchgate.net%2Fpublication%2F242487064_Une_methode_basee_sur_le_Q-learning_par_jeu_adaptatif%3Ffbclid%3DIwAR1pu9zNQZJFkk8yk_rSR75dUwfIuST7pabATkO2zYvKjJjM9AANEunkgd0&h=AT0li7uReNdBcgpgvbcvM2vnr1I80Qcj8uG_P-RkIsQeWYHlsHgDeOEcs9B5kn9M8jj1W7M1sPQ3E1PYJdqMsHBrtjLdtE4Z4zWoZG4yw5SE5phfZ11tGkRWYUwcTU-0i-TA , consulté le : 08/05/2020 à 16 h 10.

[14] Parlett, David Sidney. *The Oxford history of board games*. Oxford University Press, USA, 1999.

[15] Kroll, Moritz, and Rubino Geiß. "A ludo board game for the agtive 2007 tool contest." *URL: http://gtcases. cs. utwente. nl/wiki/uploads/ludo karlsruhe. pdf* (2007).

[16] Brown, W. Norman. "The Indian games of pachisi, chaupar, and chausar." *Expedition* 6.3 (1964): 32.

[17] Gorbenko, Anna, and Vladimir Popov. "Multi-agent path planning." *Applied Mathematical Sciences* 6.135 (2012): 6733-6737.

[18] Luna, Ryan, and Kostas E. Bekris. "Efficient and complete centralized multi-robot path planning." *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011.

[19] Desaraju, Vishnu R., and Jonathan P. How. "Decentralized path planning for multi-agent teams with complex constraints." *Autonomous Robots* 32.4 (2012): 385-403.

[20] Souidi, Mohammed El Habib, Abderrahim Siam, and Zhaoyi Pei. "Multi-agent pursuit coalition formation based on a limited overlapping of the dynamic groups." *Journal of Intelligent & Fuzzy Systems* Preprint (2019): 1-13.

[21] Souidi, Mohammed El Habib, et al. "Multi-Agent Pursuit-Evasion Game Based on Organizational Architecture." *Journal of computing and information technology* 27.1 (2019): 1-11.

[22] Souidi, Mohammed El Habib, Songhao Piao, and Guo Li. "Mobile agents path planning based on an extension of Bug-Algorithms and applied to the pursuit-evasion game." *Web Intelligence*. Vol. 15. No. 4. IOS Press, 2017.

[23] Rajko, Stjepan, and Steven M. LaValle. "A pursuit-evasion bug algorithm." *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 2. IEEE, 2001.

[24] Alvi, Faisal, and Moataz Ahmed. "Complexity analysis and playing strategies for Ludo and its variant race games." *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 2011.

[25] Alhajry, Majed, Faisal Alvi, and Moataz Ahmed. "TD (λ) and Q-learning based Ludo players." *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2012.

[26] Singh, Prabhat R., Mohamed Abd Elaziz, and Shengwu Xiong. "Ludo game-based metaheuristics for global and engineering optimization." *Applied Soft Computing* 84 (2019): 105723.

[27] Kennedy, James, and Russell C. Eberhart. "A discrete binary version of the particle swarm algorithm." *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. Vol. 5. IEEE, 1997.

[28] Mirjalili, Seyedali. "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm." *Knowledge-based systems* 89 (2015): 228-249.

[29] Saremi, Shahrzad, Seyedali Mirjalili, and Andrew Lewis. "Grasshopper optimisation algorithm: theory and application." *Advances in Engineering Software* 105 (2017): 30-47.

[30] Mirjalili, Seyedali. "SCA: a sine cosine algorithm for solving optimization problems." *Knowledge-based systems* 96 (2016): 120-133.

[31] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." *Advances in engineering software* 69 (2014): 46-61.

[32] Wilensky, Uri. "Center for connected learning and computer-based modeling." *NetLogo*. Northwestern University, 1999.

[33] Alhajry, Majed, Faisal Alvi, and Moataz Ahmed. "TD (λ) and Q-learning based Ludo players." 2012 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, 2012.

[34] Voir le site :

http://theses.univ-lyon2.fr/documents/getpart.php?id=lyon2.2008.nfaoui_eh&part=152187, consulté le : 18/05/2020 à 02 h 19.

[35] Bendahmane Tawfik, « Conception d'une plateforme multi agent pour la collecte de données dans une base de données distribuée», Mémoire de Magister en informatique, sous la direction de Kazar O, Biskra, Université Mohamed Khider, 2014, 67 p.