Republique Algérienne Démocratique Et Populaire

Ministère De L'enseignement Supérieur Et De La Recherche Scientifique

Universite « Abbes Laghrour » de Khenchela

Faculté 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 : Cid Ryhan

Daghbouche Souad

Option : Génie Logiciel Des Systèmes Distribués (GLSD)

Vers Une Nouvelle Planification De La Trajectoire Pour Les Systèmes Multi-Agent

Soutenu le 26/06/2022 :

- **Dr. Souidi Mohammed El Habib (l'encadrant)**
- **Dr. Wahiba Chouhal (président)**
- **M. Lehis saida (Examinateur)**

2021/2022

Résumé

Un système multi-agent est un ensemble d'entités capables d'interagir afin d'effectuer une tâche complexe. Les Systèmes Multi-Agents (SMA) doivent souvent poursuivre un but commun et pour y parvenir ils doivent former des coalitions efficaces. Dans ce travail, nous proposerons un algorithme de la planification de la trajectoire d'un système multi-agent. Cet algorithme est basé sur la collaboration hiérarchique des agents assignés à la même tache lors de leurs déplacements dans l'environnement. Pour cela, nous nous baseront sur l'apprentissage par renforcement ainsi que le Processus Décisionnel de Markov, En addition. Cet algorithme sera comparé avec une approche basée sur le Q-learning concernant le traitement du problème Multi-Pursuer Multi-Evader. nous divisons notre travail en trois chapitres le premier chapitre concerne les systèmes multi-agents le deuxième chapitre traite du problème de poursuite-évasion et de notre algorithme proposé ,quant au troisième chapitre, nous parlons de la comparaison entre notre approche hybrid path planning et la première approche leader follower, en analysant les résultats et en choisissant la meilleure approche.

Mots clé : Planification de trajectoire collaborative, Processus décisionnels de Markov, Apprentissage par renforcement, le Q-learning, Les systèmes multi-agents, Problème de poursuite-évasio

Abstract

A multi-agent system is a set of entities capable of interacting to perform a complex task. Multi-Agent Systems (MAS) must often pursue a common goal and to achieve this they must form effective coalitions. In this work, we will propose an algorithm for planning the trajectory of a multi-agent system. This algorithm is based on the hierarchical collaboration of agents assigned to the same task during their movements in the environment. For this, we will rely on reinforcement learning as well as the Markov Decision Process, In addition. This algorithm will be compared with an approach based on Q-learning concerning the treatment of the Multi-Pursuer Multi-Evader problem. we divide our work into three chapters the first chapter concerns multi-agent systems the second chapter deals with the pursuit-evasion problem and our proposed algorithm, as for the third chapter, we talk about the comparison between our hybrid path planning approach and the first leader-follower approach, analyzing the results and choosing the best approach.

Keywords: Collaborative trajectory planning, Markov decision-making processes, Reinforcement learning, Q-learning, multi-agent system, Pursuit-Evasion Problem

Remerciement

*Tout d'abord nous remercions **ALLAH** tout-puissant Qui nous donnés*

La force, la volonté et le soutien Pour achever ce travail

*Nous remercions nous **parents** pour leurs sacrifices et leur soutien à nous, Nous tenons à remercier notre encadrant*

Dr. Mohammed El habib souidi pour sa patience,

Sa disponibilité et nous le remercions de nous avoir encadrés, guidés, aidés et conseillés.

Nous adressons nos remerciements à tous nos enseignants du Département d'informatique de l'Université de Khenchela pour leurs efforts et leur travail acharné avec nous.

Merci beaucoup.

Dédicace

Je dédie ce projet a la mémoire de mon père a ma chère mère a mes enfants qui mon soutenus tous au Long de cette thèse à mon cher binôme a toute ma famille frères et sœurs mes amis

Remerciement

La présentation de ce modeste travail m'offre l'occasion d'exprimer ma profonde gratitude à monsieur le docteur ***Mohammed El habib souidi*** *notre encadrant pour l'accompagnement la disponibilité et les conseils, ma sœur professeur* ***Yasmina*** *qui était tj à mes côtés malgré la distance, mon amie professeur* ***Yahiaoui Hadia*** *qui m'a facilité*

La continuité de mes études

Souad

Dédicace

*A mes **chers parents**, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études,*

*A mes chères sœurs **Safa,Selma** pour leurs encouragements permanents, et leur soutien moral,*

*A mes chers frères, **Dhia,Imran,Abd-Elrahman** pour leur appui et leur encouragement, Je vous aime tous*

*A ma copine **Rihab**, tous mes remerciements et mon amour Merci pour ta bonne amitié,*

*A mes cousines **wiam ,manel** et* à mon cher binôme *et toute ma famille pour leur soutien tout au long de mon parcours universitaire,*

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Ryhan

Table des matières

Liste des figures

Liste des tableaux

Chapitre 01 :

1. Agent et système multi-agent

Contenue

L'introduction :

L'émergence des systèmes multi-agents a été l'un des évènements marquants des années quatre-vingt-dix dans la recherche informatique. Apparus tout d'abord comme un sous domaine de recherche de l'Intelligence Artificielle (IA), les systèmes multi-agents (SMA) sont actuellement l'un des domaines les plus actifs de la recherche avec une assise applicative de plus en plus importante. Cette discipline, joignant plusieurs domaines, allant de l'IA, des systèmes informatiques distribués au génie logiciel, s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes appelées agents.

Elle permet d'offrir une alternative intéressante pour la conception, la mise en œuvre et la simulation de systèmes Complexes distribués et ouverts. D'autre part, la résolution des problèmes distribués se préoccupe de la façon dont un problème donné peut être résolu par plusieurs entités qui coopèrent en divisant et en Partageant la connaissance à propos du problème et des solutions développées.

Dans ce chapitre Les concepts et principes fondamentaux des SMA sont introduits afin de situer ce travail dans son contexte de recherche. Aussi, les différentes problématiques que soulèvent les SMA sont énoncées telles que l'interaction, la communication et la coopération. **(01)**

1.1 Agent :

Les agents représentent sans doute la brique de base utilisée pour la réalisation des SMA, En plus nous pouvons considérer l'agent comme la notion la plus polémique dans ce domaine.

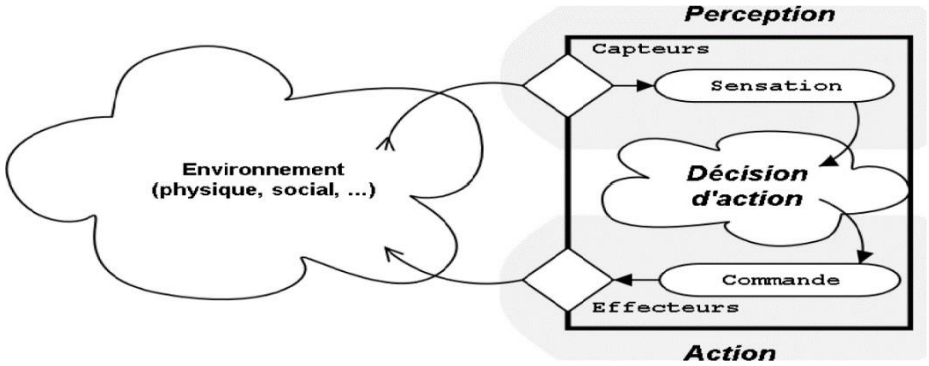

Figure 1: Structure classique d'un agent

1.1.1 Définitions :

Nous proposons d'aborder brièvement quelques définitions du concept d'agent :

- Selon **Ferber** : « Un SMA est un macro-système constitué d'agents autonomes qui interagissent dans un environnement commun pour réaliser une activité collective cohérente ». **(02)**
- Pour **Dema eau** : « Un agent est une entité réelle ou virtuelle, dont le comportement est autonome, évoluant dans un environnement, capable de le percevoir, d'agir dessus et d'interagir avec les autres agents ». **(03)**
- Pour **Wooldridge**: « Un agent est un système informatique capable d'agir de manière autonome et flexible dans un environnement **(04)**

1.1.2 Caractéristiques d'un Agent :

- ***La Réactivité :*** percevoir l'environnement et répondre, en temps réel, aux changements

- ***La Pro-activité :*** capacité de prendre l'initiative / comportement orienté but
- ***La Sociabilité :*** capacité d'interagir avec d'autres agents ou utilisateurs
- ***L'Autonomie :*** capacité d'agir sans intervention directe d'un humain ou d'un autre agent, et sous-entend que l'agent a le contrôle de ses propres actions et de son état interne.
- ***La situation :*** Un agent peut percevoir son environnement et agir sur les changements de cet environnement. **(01)**

1.1.3 Les types d'agents :

Les caractéristiques des agents citées dans la section précédente peuvent être considérées comme des critères de classification des agents. En effet, on peut classifier les agents aux agents mobiles et agents fixes selon le critère de la mobilité. En plus, le critère de la communication peut être servi pour classifier les agents en agents avec communication directe ou des agents avec communication indirecte (appelés des agents situés). On peut, aussi, choisir le niveau de coopération des agents comme un critère de classification. **(05)**

- **Les agents réactifs :** les agents réactifs sont basés sur l'intelligence artificielle réactive. Les agents réactifs sont des agents qui ne possèdent ni une représentation de leur environnement ni un vrai mécanisme de raisonnement. En plus, leur comportement est modélisé par des règles de type stimulus-réponse qui produisent des actions à partir de perceptions.

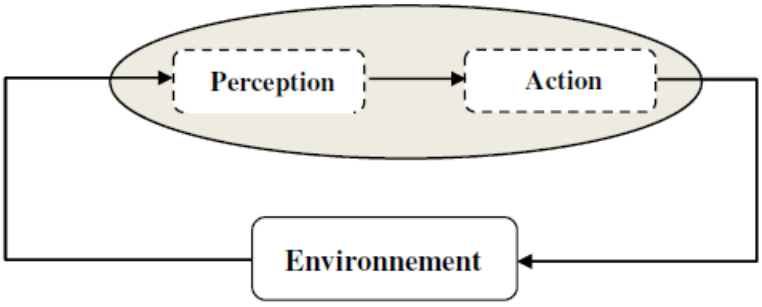

Figure 2 : Agent réactif

- **Les agents cognitifs :** l'agent possède une représentation symbolique de son environnement, des autres agents, des actions et des états internes de l'agent. Sachant que les buts de l'agent sont modélisés comme des états internes, le raisonnement de l'agent

consiste à trouver l'ensemble des actions possibles pour satisfaire ses buts.

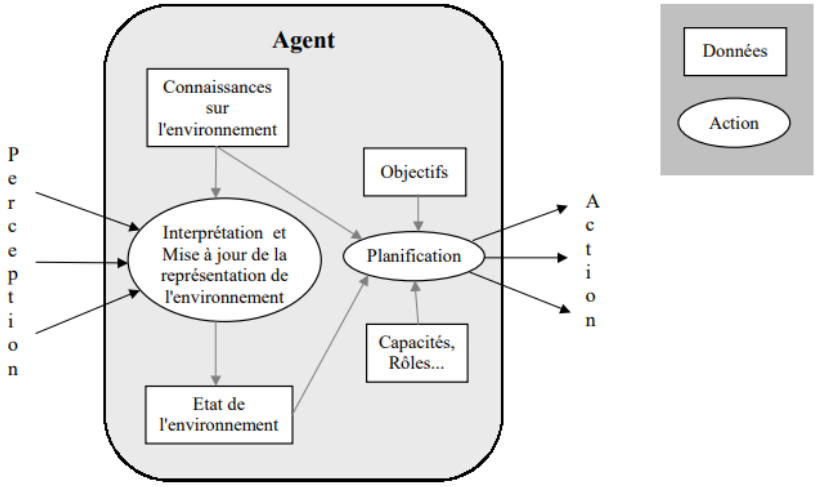

Figure 3 : Représentation de l'architecture interne d'un agent cognitif

- **Les agents hybrides :** d'autres types d'agents qualifiés d'hybrides, utilisant donc ces deux types de comportement, sont ensuite apparus. Les agents hybrides intègrent l'aspect cognitif et réactif. L'idée est de combiner les deux types d'approches qui peuvent être vues comme complémentaires. Dans une telle architecture un agent est composé de modules qui gèrent indépendamment la partie réactive et la partie cognitive.

  Le problème central reste de trouver le mécanisme idéal assurant cette combinaison.

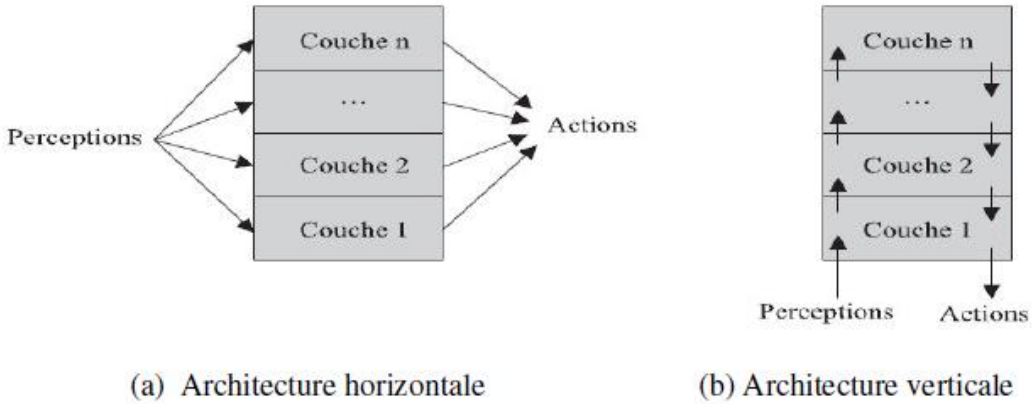

Figure 4 : Architecture d'agent hybride

Tableau 1 : Le tableau suivant résume les différences entre les agents cognitifs et les agents réactifs.

Les agents cognitifs	Les agents réactifs
Tiennent compte de leur passé	Tiennent compte de leur passé
Représentation explicite de l'environnement	Pas de représentation explicite
Agents complexes	Agent simple (fonctionnement stimulus/réaction)
Petit nombre d'agents	Grand nombre d'agents

Tableau 2: Comparaison entre les agents cognitifs et les agents réactifs

1.2 Les systèmes multi-agents :

Au premier abord, un système multi-agents peut d'une manière simpliste être considéré comme un ensemble d'agents partageant un environnement commun. Certes, la notion d'environnement est primordiale dans un système multi-agents mais elle reste, tout de même insuffisante pour qualifier un système multi-agents. Un système multi-agents réside essentiellement dans la capacité qu'ont les agents à interagir pour résoudre collectivement un problème.

1.2.1 Définitions

- Ferber a défini les **systèmes multi-agents** **(06)**

On appelle système multi-agents un système composé des éléments suivants :

- Un environnement E, disposant en général d'une métrique
- Un ensemble d'objets O, auxquels on peut associer une position dans E à un moment donné. Ces objets (hormis les agents) sont passifs : les agents peuvent les percevoir, les créer, les détruire et les modifier.
- Un ensemble d'agents A, lesquels représentent les entités actives du système,
- Un ensemble de relations R, qui unissent les objets (et agents) entre eux
- Un ensemble d'opérateurs Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.

- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

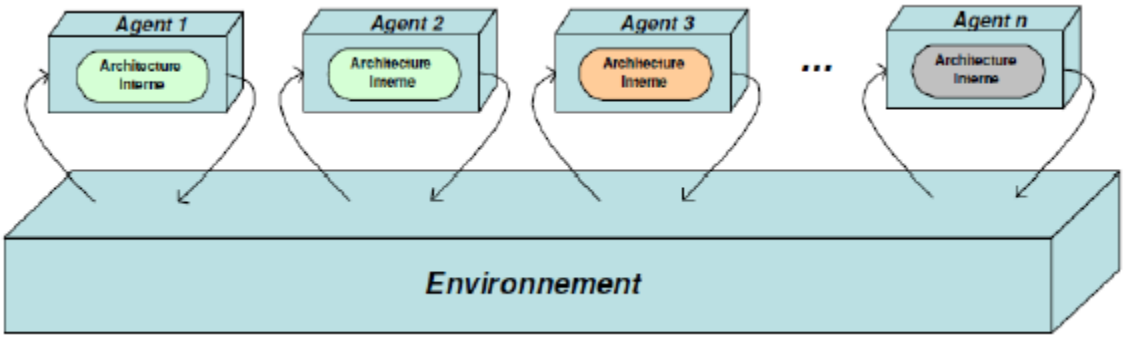

Figure 5 : Représentation schématique d'un système multi-agent

- Une autre définition, donnée par Wooldridge et Jennings, présente un SMA comme étant : « Un SMA est un ensemble d'agents ayant des buts ou des tâches, et qui interagissent pour les accomplir ». **(07)**

  Un SMA est donc un système compose d'entités informatiques, appelées des agents, qui évoluent et interagissent dans un environnement commun.

  La notion d'interaction entre agents est essentielle car chacun d'eux est impliqué dans une dynamique commune. Elle représente aussi ce qui permet de construire la réponse collective à partir des réponses individuelles.

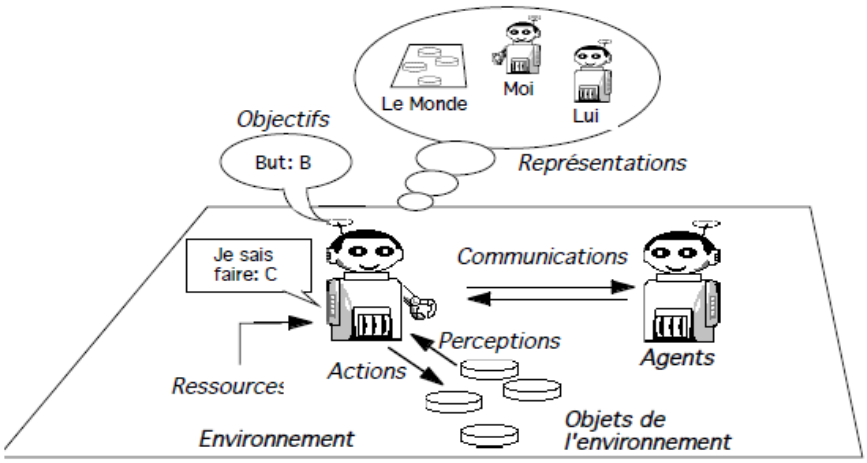

Figure 6 : Environnement d'un système multi agents

1.2.2 Caractéristiques d'un SMA :

Il existe des caractéristiques propres aux SMA, par rapport aux autres systèmes informatiques. Nous en fournissons une liste proposée par O.Boissier un SMA possède la plupart des caractéristiques suivantes : **(08)**

- **Distribution** : le système est modulaire, l'élément de base étant l'agent.
- **Autonomie** : un agent est en activité permanente et prend ses propres décisions en fonction de ses objectifs et de ses connaissances.
- **Décentralisation :** les agents sont indépendants, il n'y a pas de décisions centrales valables pour tout le système.
- **Échange de connaissances :** les agents sont capables de communiquer entre eux, selon des langages plus ou moins élabores.
- **Interaction :** les agents ont une influence localement sur le comportement des autres agents, généralement sur un pied d'égalité (il n'y a pas d'ordres, seulement des requêtes).
- ***Organisation*** : les interactions créent des relations entre les agents, et le réseau de ces relations forme une organisation qui peut évoluer au cours du temps.

1.2.3 L'Environnement :

L'environnement est un élément important dans le système multi-agents.

1.2.3.1 Définitions :

C'est tout ce qui entoure l'agent, et grâce à lui que les agents peuvent coexister et interagir. Avec celle des comportements individuels, la spécification de l'environnement permet de définir la dynamique d'un SMA. Lorsque les agents sont réactifs, l'environnement détient une importance capitale. En effet, comme ces agents ne peuvent communiquer directement entre eux, il est le médiateur de leurs interactions. Ils s'influencent mutuellement soit par leur position s'ils sont situés, soit par l'intermédiaire d'objets qu'ils perçoivent et modifient. **(01)**

1.2.3.2 Les Propriétés d'environnement :

L'environnement possède certaines propriétés sont décrites ci-dessous : **(09)**

- **Accessible ou inaccessible :** un agent a accès à l'état complet de l'environnement alors nous disons que l'environnement est accessible à cet agent ou non.
- **Déterministe ou indéterministe :** le changement de l'état de l'environnement est uniquement déterminé par l'état courant et les actions des agents ou non.
- **Statique ou dynamique :** l'environnement peut changer quand l'agent est en action (réflexion) ou non.
- **Discret ou continu :** le nombre de perceptions et d'actions est limité ou pas.

1.2.3.3 Les Topologies d'environnement :

Les agents existent dans un environnement dont la nature peut être très différente en fonction des modèles, il existe donc différentes façons de représenter cet environnement et de connecter les agents.

1.2.3.3.1 Topologie représentation par grilles :

Formées de cellules carrées en deux dimensions, On utilise la représentation des grilles lorsqu'on veut simuler des « agents statiques » (des états) et leur interaction avec leur environnement. Exemples : Modèle de ségrégation de Schelling.

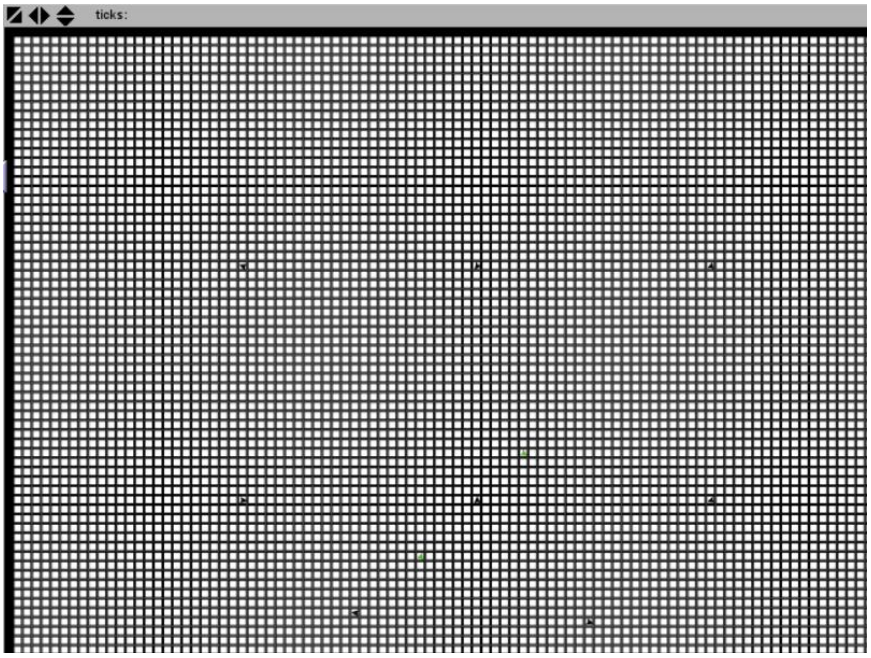

Figure 7 : Topologie représentation par grilles

1.2.3.3.2 Topologie : Représentation par réseaux

La représentation en utilisant des réseaux est beaucoup utilisée, avec les réseaux, on peut représenter des phénomènes comme la propagation et la communication entre autres.

Exemples: Réseaux sociaux, Chaîne d'approvisionnement. **(10)**

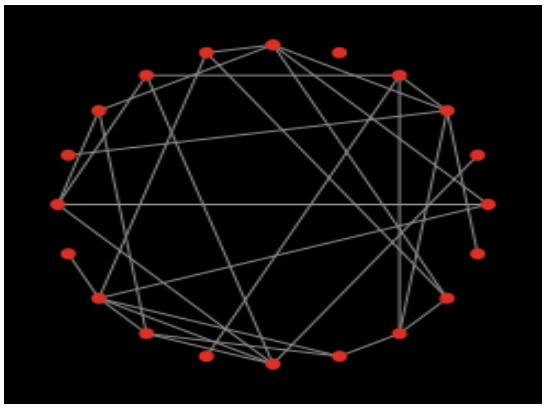

Figure 8 : Topologie Représentation par réseaux

1.3 Coordination dans les SMA :

1.3.1 Définition

La coordination des agents est la capacité fondamentale d'un agent de décider de ses propres actions dans le contexte des activités d'autres agents. **(11)**

Définition de Gelernter et Carriero : 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. **(12)**

1.3.2 L'importance de coordination :

Dans un système d'agents multiples, il est souhaitable de pouvoir considérer les agents collectivement, comme une société. Tout comme dans la société humaine, nous avons l'adage selon lequel "aucun homme n'est une île", indiquant le fait qu'aucune personne ne se suffit à elle-même, cela s'applique également dans le monde des agents. Avec la multiplication des réseaux informatiques, il devient de plus en plus rare de trouver des agents qui opèrent seuls. Le désir de faire communiquer les agents entre eux et de partager la charge des tâches a conduit à la création d'un nouveau domaine de recherche et d'application connu sous le nom d'intelligence artificielle distribuée (DAI).

DAI s'appuie sur des concepts issus d'une grande variété de domaines, notamment l'informatique, la sociologie, l'économie, la philosophie et les sciences de gestion et peut être défini comme « l'étude, la construction et l'application de systèmes multi-agents, C'est-à-dire des systèmes dans lesquels plusieurs agents intelligents en interaction poursuivre un ensemble d'objectifs et accomplir un ensemble de tâches. **(13)**

L'une des tâches de DAI est de développer des mécanismes et des méthodes permettant aux agents d'interagir de manière cohérente en tant qu'unité.

> La coordination n'implique pas de coopération : un concurrent efficace coordonnera les décisions pour maximiser son avantage contre un adversaire, comme une entreprise chronométrant une promotion de produit pour réduire un rival. Cela n'implique pas non plus la réciprocité : un agent peut coordonner ses actions avec un autre agent qui n'en est pas conscient, comme lorsqu'un conducteur d'automobile croise un deuxième conducteur dont l'esprit est entièrement ailleurs. **(11)**

Ainsi, la coordination peut être réalisée sans communication [14], à condition que les agents impliqués possèdent des modèles de comportement de chacun. Cependant, plus souvent, les agents peuvent devoir communiquer entre eux – ils doivent faire connaître leurs objectifs, leurs intentions, leurs résultats et les déclarer à d'autres agents. S'ils refusent de faire connaître leurs objectifs (en tout ou en partie) ou leurs intentions, la situation serait clairement celle d'une concurrence exigeant que d'autres agents participants postulent sur les objectifs des concurrents. La coordination a été utilisée pour aborder plusieurs Intelligence artificielle distribuée et informatique distribuée, y compris :

- Cohérence du réseau : maximiser le fonctionnement d'un système réparti d'agents ensemble ;
- Répartition des tâches et des ressources entre les agents ;
- Reconnaître et résoudre les disparités ou les conflits dans les buts, les faits, les croyances et comportements des agents.
- Déterminer la structure organisationnelle d'un agent, c'est-à-dire définir les rôles, Les responsabilités et les chaînes d'autorité entre les agents. [15]

Bien que la liste ci-dessus ne soit pas exhaustive, elle donne une idée des types de problèmes qui sont résolus à l'aide de techniques de coordination.

La coordination n'est pas seulement une préoccupation pour les chercheurs mandataires. Au fur et à mesure que les tendances évoluent vers les systèmes informatiques distribués, des parties de systèmes ouverts distribués, qui sont analogues aux agents, doivent être coordonnées. En termes simples, si les agents doivent devenir le nouveau paradigme des systèmes informatiques, nous ne pouvons pas négliger la nécessité de la recherche sur la coordination des agents techniques.

1.3.3 Techniques de coordination :

Nwana, et Jennings ont classé les techniques de coordination en quatre grandes catégories : [15]

1. Modèle organisationnelle
2. Coordination par protocoles
3. Planification multi-agents
4. Négociation

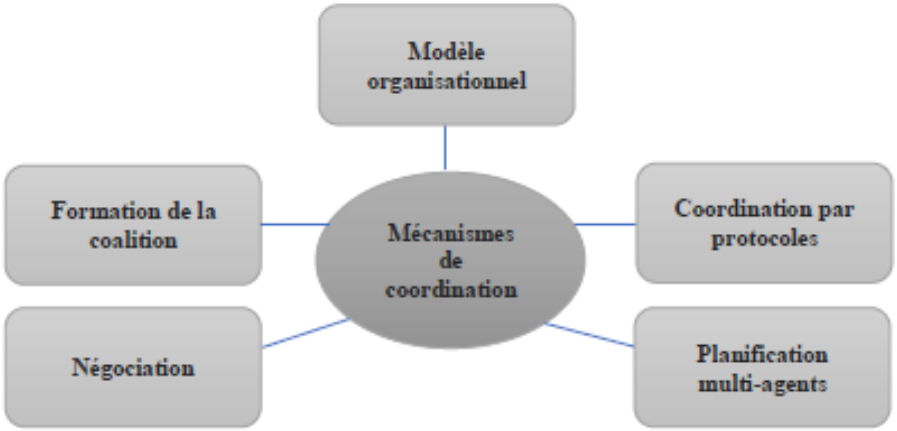

Figure 9 : Approches des mécanismes de coordination

La coordination est nécessaire :

- Lorsque les agents ont besoin d'informations et de résultats que d'autres agents peuvent fournir ;
- Lorsque les ressources communes sont limitées ;
- Pour éviter des actions inutiles ;
- Pour permettre aux agents de satisfaire des objectifs dépendants ;

De ces faits, le but de la coordination est de trouver parmi un ensemble de comportement d'agents qui interagissent, une collection de comportement qui réalise d'une façon satisfaisante les objectifs les plus importants des agents.

La coordination de multiples agents est essentielle pour la viabilité des systèmes dans lesquels ces agents partagent des ressources. La plupart des recherches en intelligence artificielle distribuée se concentrent sur le développement de stratégie de coordination en différé.

1.4 La coordination de la trajectoire

1.4.1 Définition :

La planification : Elle sert à donner les moyens à l'agent de trouver une suite d'actions à appliquer sur le monde, pour le faire passer de l'état initial à un état qui satisfait le but à atteindre.

Trajectoire ou plan : Est un ensemble structuré d'actions qui mène au but.

- 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. [16]

1.4.2 Planification de mouvements :

La planification de mouvements est la génération d'une série de mouvements continus, d'une configuration 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* 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 ». [17]

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

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

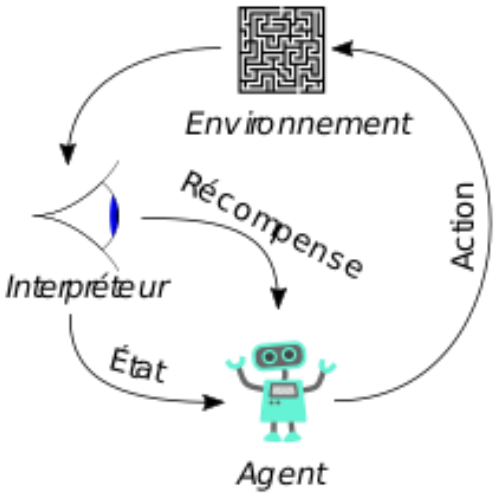

Figure 10 : Structure classique d'un agent

1.4.2.1 Processus décisionnel de Markov :

Le processus décisionnel de Markov ou MDP, est utilisé pour **formaliser les problèmes d'apprentissage par renforcement**.

Si l'environnement est complètement observable, alors sa dynamique peut être modélisée comme un **processus de Markov**. Dans MDP, l'agent interagit constamment avec l'environnement et effectue des actions ; à chaque action, l'environnement répond et génère un nouvel état.

MDP contient un tuple de quatre éléments (S, A, P **a**, R **a**)

- Un ensemble d'états finis **S**
- Un ensemble d'Actions finies **A**
- Récompenses reçues après la transition de l'état **S** à l'état **S'**, en raison de l'action **a**.
- Probabilité $P_{\mathbf{a}}$ **(18)**

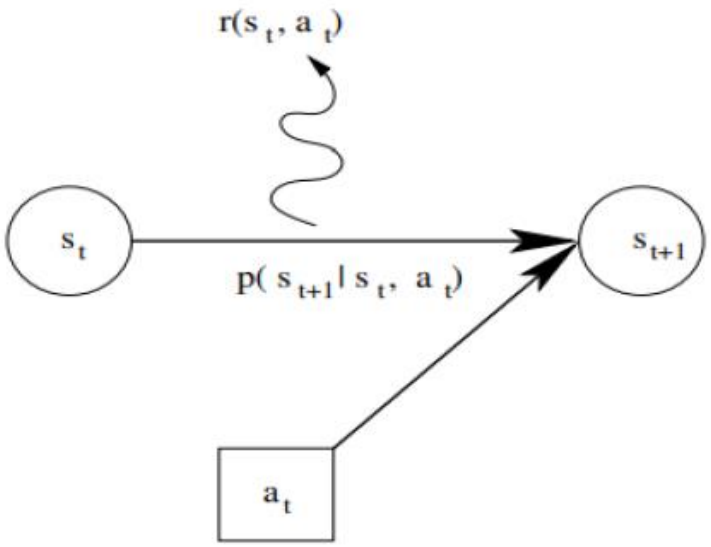

Figure 11 : Evaluation d'un MDP

MDP utilise **la propriété de Markov**

1.4.2.1.1 Propriété de Markov :

Il dit que ***"Si l'agent est présent dans l'état actuel S1, effectue une action a1 et passe à l'état s2, alors la transition d'état de s1 à s2 ne dépend que de l'état actuel et de l'action future et les états ne dépendent pas du passé. Actions, récompenses ou états."***

Ou, en d'autres termes, selon la propriété de Markov, la transition d'état actuelle ne dépend d'aucune action ou état passé. Par conséquent, MDP est un problème RL qui satisfait la propriété de Markov. Comme dans une **partie d'échecs, les joueurs se concentrent uniquement sur l'état actuel et n'ont pas besoin de se souvenir des actions ou des états passés**. **(18)**

MDP fini : Un MDP fini, c'est quand il y a des états finis, des récompenses finies et des actions finies. En RL, nous ne considérons que le MDP fini.

1.4.2.2 L'apprentissage par renforcement :

L'apprentissage par renforcement est une technique d'apprentissage automatique basée sur la rétroaction dans laquelle un agent apprend à se comporter dans un environnement en effectuant les actions et en voyant les résultats des actions. Pour chaque bonne action, l'agent reçoit un retour positif, et pour chaque mauvaise action, l'agent reçoit un retour négatif ou une pénalité

L'agent apprend avec le processus de coup et d'essai, et sur la base de l'expérience, il apprend à mieux exécuter la tâche. Par conséquent, nous pouvons dire que ***"l'apprentissage par renforcement est un type de méthode d'apprentissage automatique dans lequel un agent intelligent (programme informatique) interagit avec l'environnement et apprend à agir en son sein".*** La façon dont un chien robotique apprend le mouvement de ses bras est un exemple d'apprentissage par renforcement **(18)**

1.4.2.3 Les éléments de l'apprentissage par renforcement

Il y a quatre éléments principaux de l'apprentissage par renforcement, qui sont donnés ci-dessous : **(13)**

- **Politique :** une politique peut être définie comme la façon dont un agent se comporte à un moment donné. Il met en correspondance les états perçus de l'environnement avec les

actions entreprises sur ces états. Une politique est l'élément central de la RL car elle seule peut définir le comportement de l'agent. Dans certains cas, il peut s'agir d'une simple fonction ou d'une table de recherche, alors que, dans d'autres cas, il peut s'agir d'un calcul général en tant que processus de recherche. Il peut s'agir d'une politique déterministe ou stochastique : **Pour la politique déterministe : $a = \pi(s)$**
Pour la politique stochastique : $\pi (a \mid s) = P [At = a \mid St = s]$

- **Signal de récompense :** L'objectif de l'apprentissage par renforcement est défini par le signal de récompense. À chaque état, l'environnement envoie un signal immédiat à l'agent d'apprentissage, et ce signal est appelé **signal de récompense**.
  Ces récompenses sont attribuées en fonction des bonnes et des mauvaises actions prises par l'agent. L'objectif principal de l'agent est de maximiser le nombre total de récompenses pour les bonnes actions. Le signal de récompense peut changer la politique, par exemple si une action sélectionnée par l'agent conduit à une faible récompense, alors la politique peut changer pour sélectionner d'autres actions à l'avenir.
- **Fonction de valeur :** La fonction de valeur donne des informations sur la qualité de la situation et de l'action et sur la récompense à laquelle un agent peut s'attendre. Une récompense indique le **signal immédiat pour chaque bonne et mauvaise action**, tandis qu'une fonction de valeur spécifie **le bon état et l'action pour le futur**. La fonction de valeur dépend de la récompense car, sans récompense, il ne pourrait y avoir de valeur. L'objectif de l'estimation des valeurs est d'obtenir plus de récompenses.
- **Modèle :** Le modèle est utilisé pour la planification, ce qui signifie qu'il fournit un moyen de prendre une ligne de conduite en considérant toutes les situations futures avant de les expérimenter réellement. Les approches pour résoudre les problèmes RL **à l'aide du modèle** sont appelées **approche basée sur** un modèle. Comparativement, une approche **sans utiliser de modèle** est appelée une approche sans **modèle**. **(18)**

1.4.2.4 L'équation de Bellman

L'équation de Bellman a été introduite par le mathématicien **Richard Ernest Bellman en 1953** et est donc appelée équation de Bellman. Il est associé à la programmation dynamique et utilisé pour calculer les valeurs d'un problème de décision à un certain point en incluant les valeurs des états

précédents. C'est une façon de calculer les fonctions de valeur dans la programmation ou l'environnement dynamique qui conduit à l'apprentissage par renforcement moderne.

Les éléments clés utilisés dans les équations de Bellman sont : [18]

- L'action effectuée par l'agent est appelée "une"
- L'état qui s'est produit en exécutant l'action est "s".
- La récompense/rétroaction obtenue pour chaque bonne et mauvaise action est "R".
- Un facteur d'actualisation est Gamma "γ".

L'équation de Bellman peut s'écrire :

V(s) = max [R (s, a) + γV(s`)] ….. (1)

V(s)= valeur calculée en un point particulier.

R (s, a) = Récompense à un état particulier s en effectuant une action.

***γ* = Facteur d'actualisation**

V(s`) = La valeur à l'état précédent. [18]

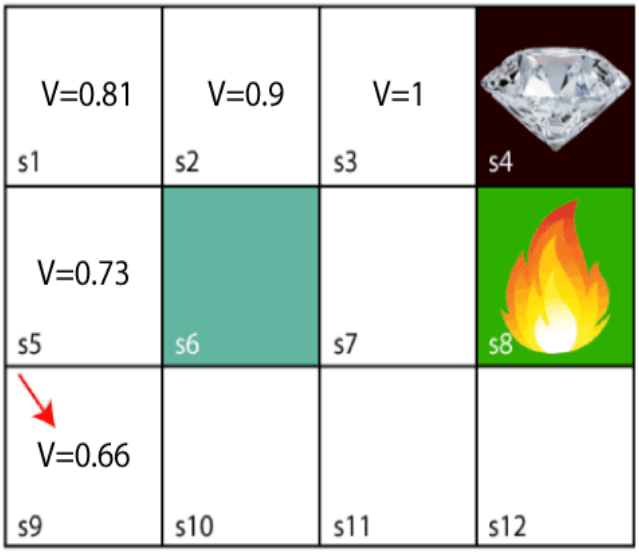

Figure 12: Les valeurs à l'états

1.4.2.5 Représenter l'état agent :

Nous pouvons représenter l'état de l'agent en utilisant l'état **de Markov** qui contient toutes les informations requises de l'historique. L'état St est un état de Markov s'il suit la condition donnée :

$$\mathbf{P\,[S_{t}+1 \mid S_{t}] = P\,[S_{t}+1 \mid S_{1}, ..., S_{t}] \ldots\ldots (2)}$$

L'état de Markov suit la **propriété de Markov**, qui dit que le futur est indépendant du passé et ne peut être défini qu'avec le présent. Le RL fonctionne sur des environnements entièrement observables, où l'agent peut observer l'environnement et agir pour le nouvel état. Le processus complet est connu sous le nom de processus de décision de Markov. **(18)**

1.4.2.6 Les Algorithmes d'apprentissage par renforcement

Les algorithmes d'apprentissage par renforcement sont principalement utilisés dans les applications d'IA et les applications de jeu. Les principaux algorithmes utilisés sont :

- **Q-Apprentissage**
- **Action d'État Récompense Action d'État (SARSA)**
- **Réseau de neurones Deep Q (DQN)**

1.4.2.6.1 Q-Apprentissage :

Q-learning est un algorithme populaire d'apprentissage par renforcement sans modèle basé sur l'équation de Bellman. « **L'objectif principal du Q-learning est d'apprendre la politique qui peut informer l'agent des actions à entreprendre pour maximiser la récompense dans quelles circonstances. »** **(16)**

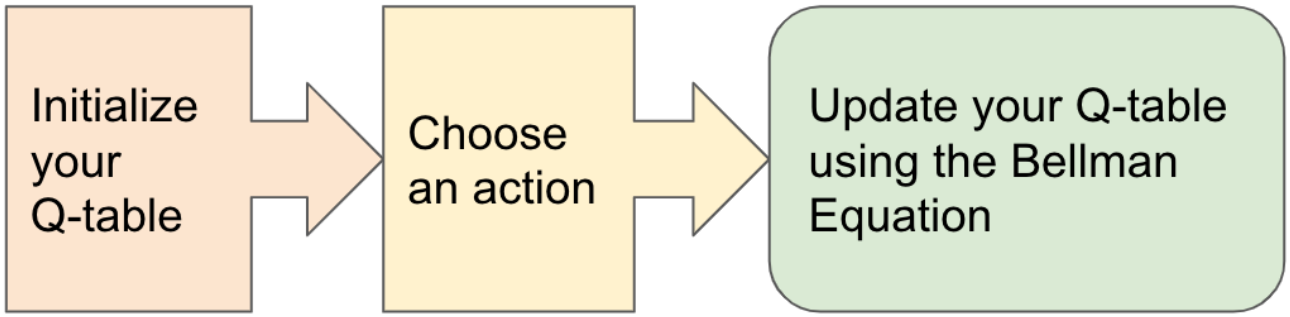

Figure 13 : L'algorithme Q-Learning

La valeur du Q-learning peut être dérivée de l'équation de Bellman.

Considérez l'équation de Bellman donnée ci-dessous :

$V(s)=\max [R(s,a)+\gamma \Sigma_{s'} P(s,a,s') V(s')]$ ….. (3)

Dans l'équation, nous avons divers composants, y compris la récompense, le facteur d'actualisation (γ), la probabilité et les états finaux s'. Mais aucune valeur Q n'est donnée.

Q- représente la qualité des actions à chaque état. Ainsi, au lieu d'utiliser une valeur à chaque état, nous utiliserons une paire d'état et d'action, c'est-à-dire Q (s, a). La valeur Q spécifie quelle action est plus lubrifiante que les autres, et selon la meilleure valeur Q, l'agent effectue son prochain mouvement. L'équation de Bellman peut être utilisée pour dériver la valeur Q. Pour effectuer n'importe quelle action, l'agent recevra une récompense R (s, a), et il se retrouvera également dans un certain état, donc l'équation de la valeur Q sera :

$Q(S,a)=R(s,a)+\gamma \Sigma_{s'} [P(s,a,s')\max Q(S',a')]$ ….. (4)

Le Q signifie **qualité** dans **Q-learning**, ce qui signifie qu'il spécifie la qualité d'une action entreprise par l'agent.

Q-table : Une table Q ou une matrice est créée lors de l'exécution du Q-learning. Le tableau suit la paire d'état et d'action, c'est-à-dire [s, a], et initialise les valeurs à zéro. Après chaque action, la table est mise à jour et les valeurs q sont stockées dans la table. L'agent RL utilise cette table Q comme table de référence pour sélectionner la meilleure action en fonction des valeurs q.

1.5 Conclusion

Dans ce chapitre nous avons présenté les concepts de base des systèmes multi-agents et nous avons aussi exposé l'un des puissants outils de l'IAs.

Les SMA proposent aujourd'hui une nouvelle technologie effective de mise en œuvre de systèmes complexes dès lors que ceux-ci requièrent distribution, ouverture, coopération et autonomie ajustable. Ces systèmes forment une communauté de recherche issue de plusieurs influences. Les principales sont :

- L'intelligence artificielle distribuée.
- La vie artificielle ou les systèmes inspirés d'organismes vivants.
- Les sciences sociales et les sciences cognitives.

Chapitre 02

2. Proposition d'un nouvel algorithme de planification de trajectoire

Contenue

Introduction :

Beaucoup d'efforts ont été consacrés aux jeux de poursuite-évasion dans des environnements bruyants pour un grand nombre d'applications telles que les engagements militaires sur le champ de bataille, la poursuite criminelle, les conceptions d'évitement de collision dans des environnements intelligents.

Les systèmes de transport, le suivi et la collecte des débris spatiaux et d'autres domaines connexes. Dans de tels jeux, les poursuivants souhaitent pour capturer le ou les évadés, tandis que le ou les évadés tentent d'éviter la capture. Pour un couple poursuivant-fuyant, si à un moment donné, le distance entre le poursuivant et l'esquive est inférieure à une distance unitaire prédéfinie, le poursuivant capture l'esquive. Si l'esquive peut rester à l'écart de cette plage pour toujours, l'esquive gagne.

Le point de départ des jeux de poursuite-évasion (PE) est le cas d'un poursuivant et d'un évadé, qui est un jeu à deux joueurs à somme nulle qui pourrait être résolu via les équations de Hamilton-Jacobi-Isaacs (HJI). Problème comme, le problème de l'homme-lion dans lequel un poursuivant essaie d'attraper un évadé à la même vitesse dans un espace délimité, poursuite-évasion dans des environnements polygonaux, poursuite-évasion dans des environnements polygonaux avec une porte ont été étudiée. Dans les applications du monde réel, des algorithmes tels que la recherche aléatoire et les ensembles de niveaux ont été utilisés pour calculer les stratégies de contrôle numérique applicables pour un ou les deux joueurs. **(19)**

3.1 Le problème de poursuite-évasion:

Les jeux de poursuite-évasion (PE) sont considérés comme l'un des domaines distinctifs de l'intelligence artificielle distribuée traitant le problème de décision coopérative dans les systèmes multi-agents (SMA).

Il se concentre principalement sur les approches coopératives utilisées pour connecter de nombreux agents autonomes jouant le rôle de poursuivants pour capturer les évadés en fait, le problème de la poursuite nécessite une coordination des actions des poursuivants.

Il est largement utilisé en relation avec les différents types de mécanismes de coordination et d'algorithmes de formation de coalition depuis sa création.

Dans le but de décrire ce problème, plusieurs types d'environnement ont été utilisés dans les activités de recherche récentes. Par exemple les auteurs ont utilisé un espace euclidien à m dimensions où les esquives et les poursuivants peuvent se déplacer à la même vitesse.

Notant que, pour chaque évadé, il existe un ensemble fini de poursuivants essayant de le capturer. Ils ont prouvé par une méthode constructive simple qu'une k-capture est toujours réalisable lorsque le poursuivant se trouve à l'intérieur de la k-coque des poursuivants (la k-coque est définie comme un ensemble de tous les points p tels que toute ligne passant par p divise le point donné en deux ensembles de k points chacun au moins), le problème a été étudié dans le cadre des réseaux.

De capteurs et d'acteurs sans fil (environnement planaire) où une nouvelle poursuite coopérative en plusieurs étapes basée sur l'algorithme de formation **Dai-sy-Chain**[20] et une méthode basée sur

Le mode glissant a été utilisées pour contrôler le processus de poursuite. De plus, le jeu a été introduit dans deux cas. Dans la première, l'évadé est toujours supposé être statique dans l'environnement. Dans l'autre, l'évadé se déplacera une fois qu'il aura détecté les poursuivants,

Le jeu de poursuite était présenté dans un environnement totalement basé sur la théorie des graphes, dans lequel l'objectif principal était de localiser le nœud contenant l'évadé. à cette fin, l'algorithme Iterative Greedy Node Search (IGNS) a été entreprispour activer la recherche garantie hors ligne.En ce qui concerne la planification du cheminement des poursuivants, nous avons appliqué la méthode d'apprentissage par renforcement (RL) connue sous le nom de Q-learning pour optimiser le mouvement des agents pendant la poursuite. L'algorithme RL est considéré

comme un simple cadrage du problème d'apprentissage basé sur les interactions environnementales afin d'atteindre un objectif spécifique. chaque agent caractérisé par un niveau de compétences permettant les interactions avec l'environnement est appelé le décideur.

Les informations générées dans l'environnement sont communiquées à l'agent par le biais d'interactions entre eux. En fait, l'agent sélectionne une action à exécuter dans l'environnement à partir des informations reçues. Les changements d'environnement provoqués de différentes manières par les actions sont transmis à l'agent via un signal de renforcement scalaire. Les actions de l'agent sont effectuées pour maximiser les récompenses générées par l'environnement sous forme de valeurs numériques spéciales. Les méthodes d'apprentissage par différence temporelle, la programmation dynamique. Et Monte-Carlo représente les trois principales méthodes utilisées en RL. En effet, les méthodes d'apprentissage par différence temporelle reposent sur différents concepts extraits à la fois des méthodes de programmation dynamique et des techniques de Monte Carlo. **(20)**

3.1.1 Définition :

La poursuite-évasion est un problème consistant à trouver des trajectoires pour un groupe d'agents dans un environnement connu de telle sorte que l'un des agents est capable d'approcher la distance critique d'une cible se déplaçant au hasard dans la zone gardée dans le temps le plus court possible Si la carte de la zone est disponibles et nous avons quelques autres informations préliminaires sur les lieux d'intrusion potentielle, le problème est de trouver la trajectoire de chaque agent qui conduira à attraper l'évadé dans les plus brefs délais. **(21)**

3.2 Le poursuivant :

Dans un jeu de poursuite-évasion, les poursuivants tentent de capturer les évadés en les assiégeant de toutes les directions dans un monde en grille.

L'ensemble des poursuivants est désigné par : P= $\{P1, ..., Pn\}$

3.2.1 Paramètres de capacité d'un poursuivant :

- ***Position de l'environnement*** **:** La position de l'agent dans l'environnement est un critère essentiel pour les séquences de poursuite, car la capture sera plus facile si le poursuivant est plus proche de l'évider. La position Pos est calculée comme suit : **(22)**

$$\mathbf{Pos} = \boldsymbol{Dist}\ (\boldsymbol{SP}, \boldsymbol{SE}) \ldots\ldots \mathbf{(5)}$$

SP : est l'état (cellule) du poursuivant,

***S*E :** est l'état (cellule) de l'évadé.

Dist : est la distance entre le poursuivant et l'évadé.

$$\mathbf{D}\boldsymbol{ist}\ (\boldsymbol{SP}, \boldsymbol{SE}) = \sqrt{(\mathbf{CoPx} - \mathbf{CoEx})^2 + (\mathbf{CoPy} - \mathbf{CoEy})^2} \ \ldots\ldots \mathbf{(6)}$$

(Cox, Coy): Coordonnées cartésiennes de la cellule contenant l'agent dans l'environnement.

La récompense : La récompense de chaque cellule est inversement proportionnelle à la distance entre la cellule concernée et la cellule contenant la cible poursuivie

$$\mathbf{R}(\boldsymbol{s}) = \mathbf{M} - \boldsymbol{Dist}\ (\boldsymbol{s}, \boldsymbol{s'}) \ldots\ldots \mathbf{(7)}$$

M :la récompense maximale ou la récompense de la cellule contenant la cible.
S: l'état contenant le poursuivant.
S′: l'état contenant l'évadé

- **La vitesse d'un poursuivant :** C'est Combien de cellules parcoure l'agent par un nombre d'itérations et est une condition nécessaire à la capture d'un évadé et on le calcule comme suit :

$$\mathbf{Vitesse} = \frac{\mathbf{Dist\ (SP,SE)}}{\mathbf{nombre\ d'itérations}} \ \ldots\ldots \mathbf{(8)}$$

3.3 L'évadé :

Chaque évadé est un agent qui tente toujours à éviter la capture des poursuivants. L'ensemble des évadés est désigné par : E = {$E1$, ..., Em}

Types d'évadé:

- **Statique** : sans mouvement.
- **Dynamique :** Il en existe deux types : l'évadé se déplace aléatoirement et librement sans aucune restriction,

et au deuxième type, l'évadé se déplace intelligemment, qui est réactif , tout en s'appuyant sur la méthode constructive

Chaque fugue Chaque évadé est caractérisé par un type **Re**, avec **Re ∈ {I, II, III, IV}** pour indiquer combien de poursuivants sont nécessaires pour le capturer ;

- Re = I : signifie que cet évadé a besoin d'un poursuivant pour être capturé.
- Re = II : signifie que cet évadé a besoin de deux poursuivants pour être capturé.
- Re = III : signifie que cet évadé a besoin de trois poursuivants pour être capturé.
- Ré = IV : signifie que cet évadé a besoin de quatre poursuivants pour être capturé. La planification de trajectoire en poursuite-évasion :

Le jeu de poursuite-évasion est un problème connu dans les systèmes multi-agents. Ce problème est abordé par l'élaboration de différents mécanismes de coordination des tâches et de planification des trajectoires.

Concernant les méthodes de planification de parcours utilisées, la majorité des recherches sont basées sur des méthodes d'apprentissage automatique telles que le Q-Learning.

Concernant la planification de parcours en environnement stochastique te, dans lequel le problème était formel -isé sous la forme d'une planification de chemin théorique de décision multi-objectifs et transformé en 2VMDP (Vector-Valued Markov Decision Process) pour déterminer comment calculer une politique d'équilibrage entre les différents critères pris en compte. une nouvelle méthode basée sur MAMDP a été appliquée au Gate AssignmentProblem (GAP) sachant que l'horaire des vols peut contenir certains événements stochastiques tels que des retards qui doivent être planifiés. Les portes dans cette approche sont représentées par des agents collaboratifs accomplissant un ensemble de tâches d'assignation de vols données par un contrôleur centralisé afin de constituer un mécanisme robuste absorbant les perturbations des vols.

En ce qui concerne Pursuit-Evasion, MDP est généralement utilisé pour fournir la planification de mouvement pour les poursuivants mobiles grâce à la maximisation des récompenses obtenues pendant la poursuite. **(23)**

3.4 Approche Leader Follower :

Leader Follower Approach (LFA), un agent agit comme un leader dont le mouvement définit le chemin pour l'ensemble du groupe. Tous les Followers utiliseront le chemin défini pour atteindre un certain objectif ou pour accomplir une tâche définie.

Les Followers doivent se positionner en fonction de la position et de l'orientation du leader. Le leader se déplace le long d'une trajectoire assignée et Les Followers maintiennent la distance et l'orientation souhaitées par rapport à l'agent leader. Pour chaque agent, une transformation de coordonnées est d'abord dérivée pour déterminer l'erreur dans le système. Sur la base de cette transformation, un algorithme de suivi est conçu avec un observateur pour minimiser l'erreur. [31]

3.5 L'algorithme Leader Follower :

```
Nbr_Eev ← nombre d'évader détectés
i← 0
k← 0
  While ( k <= Nbr_Eev) do
            Créer groupe pursuers
            K ←k+1
End
 Foreach groupe do
     Créer pursuers-List
    PrsL [i] ← pursuers-List
    Créer  rewardList
    rewardP [i]← rewardOf_Pursuers_List
     Foreach   pursuers in pursuers-List do
```

```
        If    rewardP [i] == max then
                PrsL [i] == leader
        Else
        PrsL [i] == followers
        Endif
        Move (leader)
    End
  End
    Repeat
     Move (evader) = true
     Move_to (leader , evader)
     Move_to (follower ,leader)
   If  mean of   rewardP [i] >= 96.5 then
        move (evader)  = false
        Capture = true
     Else
     Capture = false
    Endif
Until (capture=true)
  End
```

3.6 Proposition d'un nouvel algorithme de la planification de la trajectoire :

```
Notre algorithm "hybrid path planning"

Nbr_Eev ← nombre d'évader détectés

i← 0

k← 0

  While ( k <= Nbr_Eev) do

                Créer groupe pursuers

                K ←k+1

  End

 Foreach groupe do

    Créer pursuers-List

     PrsL [i] ← pursuers-List

     Créer  rewardList

    rewardP [i]← rewardOf_Pursuers_List

      Foreach   pursuers in pursuers-List do

            If   rewardP [i] == max then

                  PrsL [i] == leader

           Else

              PrsL [i] == followers

          Endif

          Move (leader)

      End
```

```
    r1← reward_to (follower ,leader)
    r2← reward_to (follower , evader)
  End
    Repeat
     Move (evader) =true
     If   r1 > r2  then
          Move_to (follower ,leader)
    Else
      Move_to (follower , evader)
    Endif
   If  mean of  rewardP [i] >= 96.5 then
        move (evader)  = false
        Capture = true
     Else   Capture = false
    Endif
Until (capture=true)
  End
```

3.6.1 explication de l'algorithme proposé (hybrid path planning):

La première chose que nous attribuons au nombre d'évadés, puis au groupes de poursuivants qui sont attribuer au nombre de l'évadés. Au sein de chaque groupe des poursuivants , il y a une chaîne d'agents ,Nous attribuons le type d'agents au sein de tous les groupes de poursuivants soit un leader ou un follower, en fonction de la récompense de chaque agent.

Celui qui a une plus grande récompense que le reste des agents est le leader où le leader suit toujours l'évadé. Notre nouvel algorithme se concentre sur la récompense alors que nous comparons les récompenses entre les followers et les leaders et entre les followers et les évadés Si la récompense entre les followers et le leader est supérieure à la récompense entre les followers et l' évadé alors les followers suivent le leader sinon les followers poursuivront l' évadé, pour capture l'évadé doit étre la moyenne de récompense pour les poursuivants supérieur ou egale 96.5 ,Cette poursuite est répétée jusqu'à ce que l'évad soit capturé.

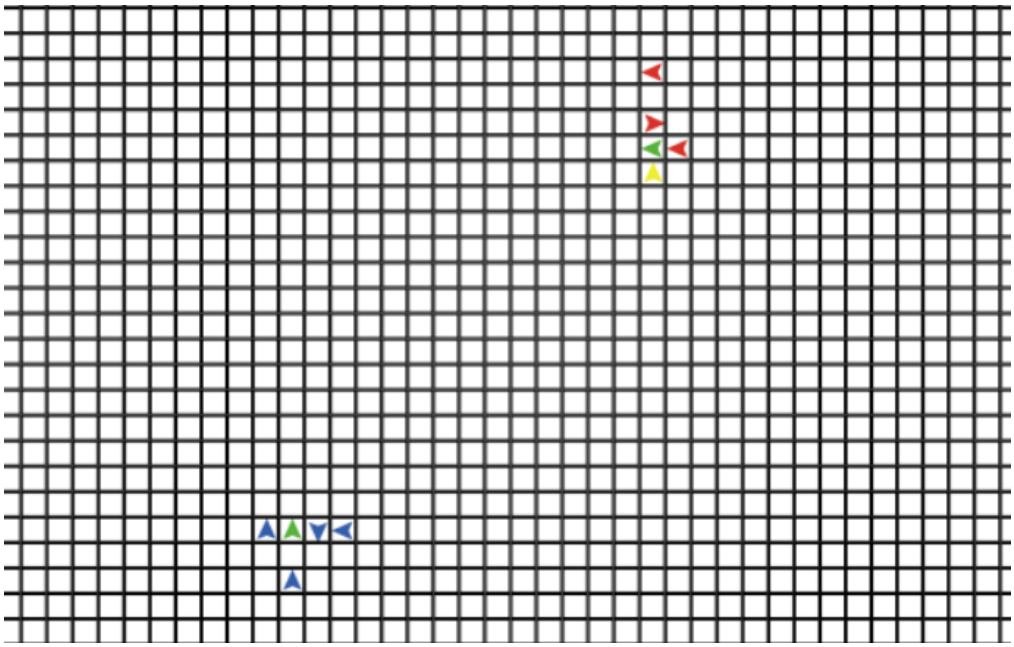

Figure 14 : les deux groupes d'agents

3.7 Modélisation de l'environnement :

Les valeurs de la cellule peuvent être représentées par le vecteur [**X, Y, i, Z, R**], le tableau 02 suivant détaille chaque variable :

La variable	Sa signification
X	Cette variable montre la valeur de la Récompense obtenue par les leaders appartenant au premier groupe 01 s'ils atteignent la cellule concernée.

	Elle augmente lorsque la distance entre l'agent et son but diminue. Dans notre environnement elle renvoi aux leaders du groupe rouge. Si l'un de ces derniers occupe une cellule, il reçoit une récompense de sa valeur de X
Y	Cette variable montre la valeur de la Récompense obtenue par les suiveurs (followers) appartenant au premier groupe 01 s'ils atteignent la cellule concernée. Elle augmente lorsque la distance entre l'agent et son but diminue. Si l'un de ces derniers occupe une cellule, il reçoit une récompense de sa valeur de Y
I	L'indice de cellule. Cette variable peut Contient deux valeurs 1 si la cellule contient un agent ou 0 si la cellule est vide (Ça facilite aux agents d'apprendre l'action suivante)
Z	Cette variable montre la valeur de la Récompense obtenue par les leaders appartenant au deuxième groupe 02 s'ils atteignent la cellule concernée. Elle augmente lorsque la distance entre l'agent et son but diminue. Dans notre environnement elle renvoi aux leaders du groupe bleu. Si l'un de ces derniers occupe une cellule, il reçoit une récompense de sa valeur de Z

R	Cette variable montre la valeur de la Récompense obtenue par les suiveurs (followers) appartenant au deuxième groupe 02 S'ils atteignent la cellule concernée. Elle augmente lorsque la distance entre l'agent et son but diminue. Si l'un de ces derniers occupe une cellule, il reçoit une récompense de sa valeur de R

Tableau 3 : Explication de valeurs du vecteur d'une cellule

3.7.1 Exemple :

Nous prenons par exemple la cellule contenant le vecteur **[76 38 1 69 15] dans la figure 14**, le nombre **76** signifie la récompense obtenue par le leader du premier groupe dans le cas où il est positionné dans la cellule concernée**,** aussi le nombre **38** signifie la position du follower du premier groupe si un suiveur du premier groupe l'occupe, il reçoit une récompense de **38**, la valeur 1 du vecteur, elle montre qu'un agent occupe cette cellule, Ou si la cellule est vide, on la représente avec la valeur **0.**

La valeur **69** signifie la position du leader du deuxième groupe si un leader du deuxième groupe l'occupe, il reçoit une récompense de **69**, et la même chose pour le follower du deuxième groupe, il reçoit une récompense **15.**

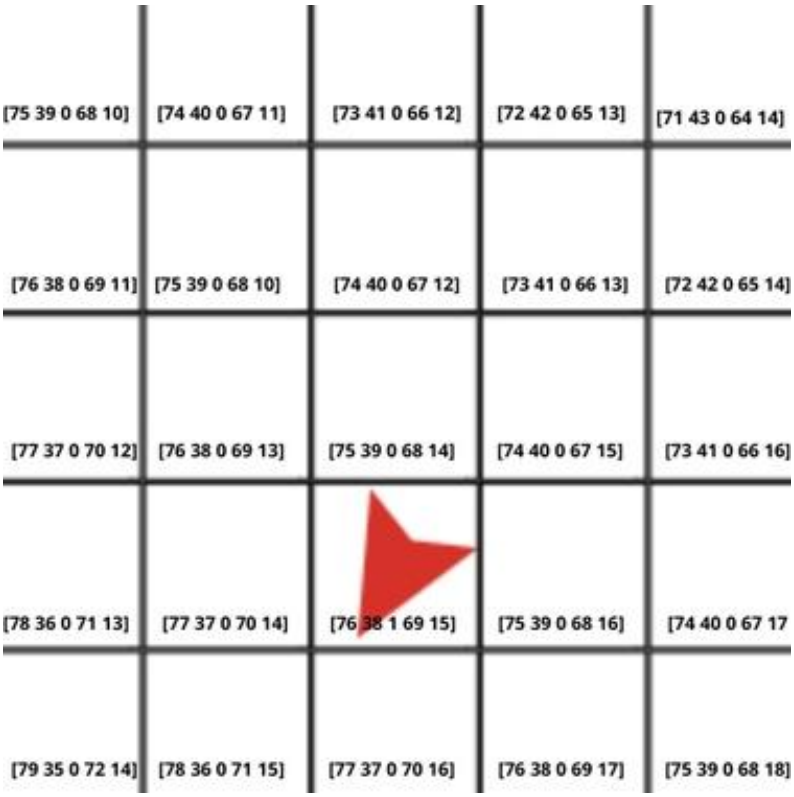

Figure 15 : Fonction de récompense appliquée à l'environnement de grille

3.8 Conclusion :

Dans ce chapitre, nous avons abordé le problème de poursuite-évasion et également la planification de trajectoire en poursuite-évasion, et nous avons expliqué les détails de notre approche (hybrid path planning), Après avoir bien détaillé notre approche proposée dans ce chapitre, nous allons pouvoir l'implémenter en comparaison avec une autre approche (Leader-Follower) dans le chapitre suivant via l'utilisation de la plateforme Netlogo.

Chapitre 03

4. Expériences de simulation

Contenue

Introduction :

La recherche sur les Systèmes Multi-Agents (SAM) a récemment conduit au développement de langages de programmation pratiques et des outils adaptés à la mise en œuvre de tels systèmes. Construire cette nouvelle programmation paradigme devient rapidement l'un des plus importants pour encoder directement une certaine théorie de l'agence, tandis que d'autres étendent les langages existants pour s'adapter aux particularités de ce nouveau paradigme. L'utilisation de ces langages, au lieu de langages plus conventionnels, s'avère utile lorsque le problème est modélisé comme un système multi-agents, et appréhendé en termes de et les concepts sociaux tels que les croyances, les objectifs, les plans, les rôles et normes. **(24)**

Le concept d'agent est aujourd'hui plus qu'une technologie efficace, il représente un nouveau paradigme pour le développement de logiciels dans lesquels l'agent est un logiciel autonome qui a un objectif, évolue dans un environnement et interagit avec d'autres agents au moyen de langages et de protocoles. Souvent, l'agent est considéré comme un objet « intelligent » ou comme un niveau d'abstraction au-dessus des objets et des composants. Les méthodes de développement orientées objet – au vu des différences entre les objets et les agents ne sont pas directement applicables au développement de SMA. Il est alors devenu nécessaire d'étendre ou de développer de nouveaux modèles, de nouvelles méthodologies et de nouveaux outils adaptés au concept d'agent.

Après la révolution de la conception et de la programmation orientée objet, nous sommes donc à l'aube d'une nouvelle révolution qui serait celle de la conception et de la programmation orientée agent/interaction/organisation. Le constat concernant la conception de SMA, dans les années 1998-1999, est que le développement de SMA est coûteux en temps à cause de leur complexité, mais aussi par le fait que pour chaque application, il faut créer tout le SMA adéquat. En effet, la majorité des applications existantes en SMA ont été développées de manière ad hoc. **(25)**

Ce foisonnement a conduit en parallèle à de multiples propositions de modèles d'agents, notamment par le rapprochement effectué avec l'approche objet. De là, plusieurs formalismes sont

apparus, chacun mettant en avant une représentation de l'agent et de son système. En 1999, se fait donc sentir le besoin de fournir des modèles, des méthodologies, des plates-formes pour faciliter la prise en compte de la complexité des systèmes à concevoir [(26)] et pour aider les concepteurs qui ne sont pas nécessairement spécialistes des SMA.

Dans ce chapitre, nous allons exposer notre choix concernant la sélection de la plateforme **Netlogo** pour effectuer nos simulations. Durant ces simulations, nous allons étudier le temps d'exécution ainsi que le nombre d'itération utilise pour capturer un évader.

4.1 La programmation orientée agents :

L'idée de la programmation orientée agent a été initiée par **Shoham** [(24)] comme un nouveau paradigme de programmation combinant l'utilisation de notions mentalistes (telles que la croyance) pour programmer des agents autonomes (individuels) et une vision sociétale du calcul.
La contribution importante de la programmation orientée agent en tant que nouveau paradigme a été de fournir des moyens d'aider programmeurs pour développer des systèmes autonomes. Par exemple, les langages de programmation d'agents ont généralement des constructions de programmation qui facilitent (par rapport aux langages de programmation traditionnels).
Le développement de systèmes qui fonctionnent en permanence et réagissent aux événements qui caractérisent les changements dans les environnements dynamiques où ces des systèmes autonomes fonctionnent généralement. [(24)]

Dans cette approche, les agents sont les éléments centraux du langage, de la même façon que les objets sont centraux pour les langages orientés objets. La perspective sur les agents est cognitive : les agents sont caractérisés par des notions mentales comme leurs croyances, leurs décisions et leurs obligations. De plus, chaque agent a associé un ensemble d'habiletés qui représentent ce que l'agent sait faire. En même temps, la programmation orientée agents suppose qu'on va développer des programmes dans lesquels plusieurs agents interagissent, ce qui met l'accent sur la dimension sociale des agents. Le langage de programmation proposé par **Shoham** comme démonstration de ce nouveau paradigme s'appelle AGENT0.
La différence principale entre un tel langage et un langage de programmation classique que l'on pourrait utiliser pour développer des agents, vient du fait que les notions mentales qui caractérisent les agents apparaissent dans le langage lui-même, et que la sémantique du langage est intimement

liée à la sémantique de ces notions mentales. La programmation orientée agents peut être vue comme une spécialisation de celle orientée objets parce que les modules du programme sont maintenant des agents, c'est-à-dire des objets avec un état qui définit les notions mentales associées, et que les messages entre objets sont remplacés par des messages entre agents. **(27)**

4.1.1 Plates-formes de développement des systèmes multi-agents :

Une plate-forme de développement des systèmes multi-agents est une infrastructure de logiciels utilisée comme environnement pour le déploiement et l'exécution d'un ensemble d'agents.

Ce qui importe, c'est qu'à un moment ou un autre, la plate-forme simplifie la tâche du développeur. Plus le développeur est assisté dans ses différentes tâches, plus la plateforme est performante. **(27)**

Parmi les plates-formes fournies comme logiciels libres, il y quelques plates-formes plus connues pour avoir été utilisées dans le développement de plusieurs applications : JADE, MACE, ZEUS, et MADKIT pour les agents cognitifs, et SWORM pour les agents réactifs. 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. **(28)**

4.1.1.1 La plateforme de simulation NetLogo :

NetLogo est un environnement de programmation au sein duquel il est possible de simuler des phénomènes sociaux et naturels. NetLogo a été créé en 1999 par Uri Wilensky et depuis lors, il est en développement continu, toujours sous sa direction, au Center for Connected Learning and Computer-Based Modeling (CCL) de l'Université Northwestern aux États-Unis.

Il est particulièrement adapté à la modélisation de systèmes complexes qui évoluent dans le temps. Les modélisateurs peuvent donner des instructions à des centaines, voire des milliers d'agents, qui fonctionnent indépendamment. Cela permet d'explorer le lien entre le comportement des individus au niveau micro et les configurations macro qui émergent des interactions entre lesdits agents. **(29)**

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 effectués dans le but de modéliser et de simuler des systèmes complexes évoluant dans le temps.

Il a été utilisé pour concevoir et mettre en œuvre 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.

Les caractéristiques qui distinguent NetLogo peuvent être résumées comme suit : **(30)**

- **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.

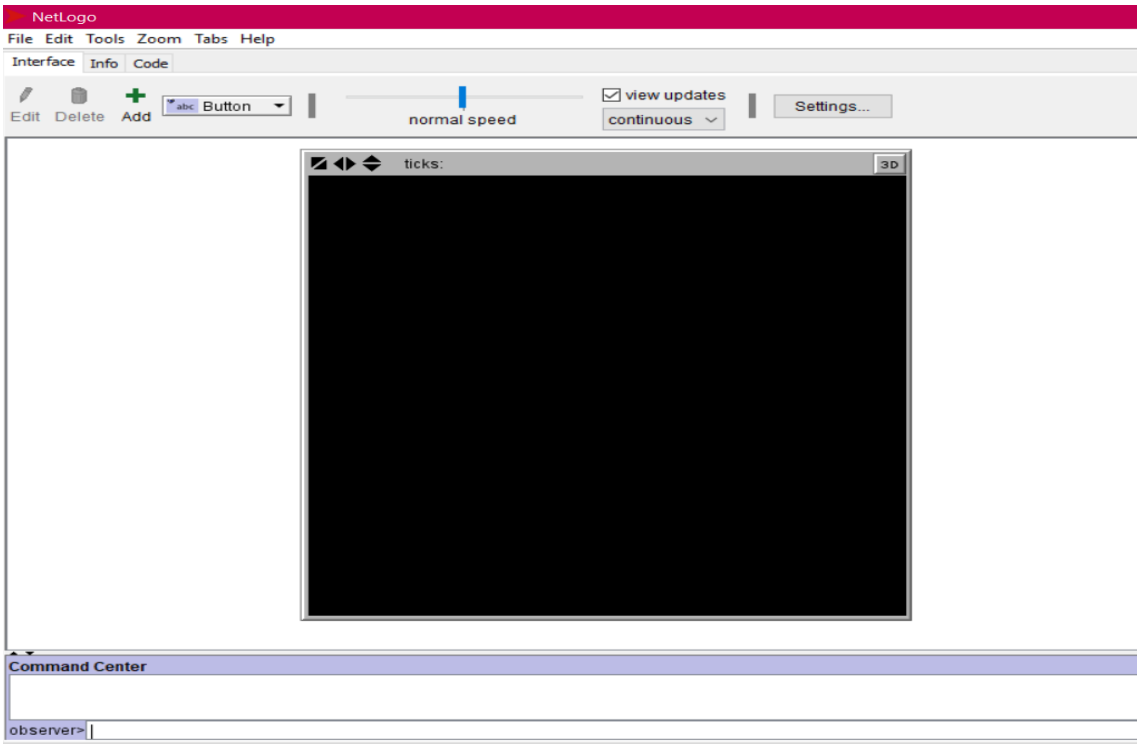

Figure 16 : Interface NetLogo

Au cours de notre travail, nous avons constaté l'utilité d'utiliser NetLogo dans le but de décrire l'environnement de poursuite-évasion ainsi que le processus de poursuite des coalitions d'agents jusqu'à la capture des évadés détectés.

NetLogo dispose d'un ensemble d'agents préinstallés dans le système, prêts à être utilisés pour écrire les programmes. Les agents NetLogo peuvent être regroupés en quatre catégories: **(28)**

- **Les tortues (Turtles)** : ce sont nos agents individuels, Ces agents peuvent se déplacer dans le monde, La forme initiale des tortues est celle de petits triangles. Cette forme peut être modifiée à l'aide d'une commande qui permet de donner une forme spécifique.

- **Les patches :** forment une grille qui couvre le monde, les agents se déplacent sur les patches et les patches peuvent stocker des variables. En d'autres termes, chaque patch

caractérisé par des coordonnées cartésiennes est considéré comme une cellule pouvant être occupée par un agent ou un obstacle.

- **Les liens (links) :** Ces agents servent à « relier » deux ou plusieurs tortues entre elles. Qui est représenté comme une ligne dessinée entre ces agents.

- **L'observateur :** Cet agent, qui n'est pas visible, est l'agent situé au sommet de la hiérarchie. L'observateur peut donner des ordres aux autres agents. On communique avec lui par le biais de la « fenêtre de l'observateur » située tout en bas de l'interface.

4.2 Résultats de la simulation :

Afin d'évaluer notre approche proposée, nous implémentons un jeu d'évasion sur une plate-forme de développement de modélisation NetLogo basée sur des agents open source. Le jeu de poursuite-évasion est illustré dans une grille rectangulaire 2D composée de 100 × 100 cellules, les poursuivants sont regroupés en plusieurs groupes pour capturer les différents évadés détectés. Notant que, les poursuivants et les évadés ont la même vitesse et nos simulations sont basées sur dix (10) poursuivants et deux (02) évadés, comme le montre la Figure 17, il est expliqué en détail comment capturer un évadé.

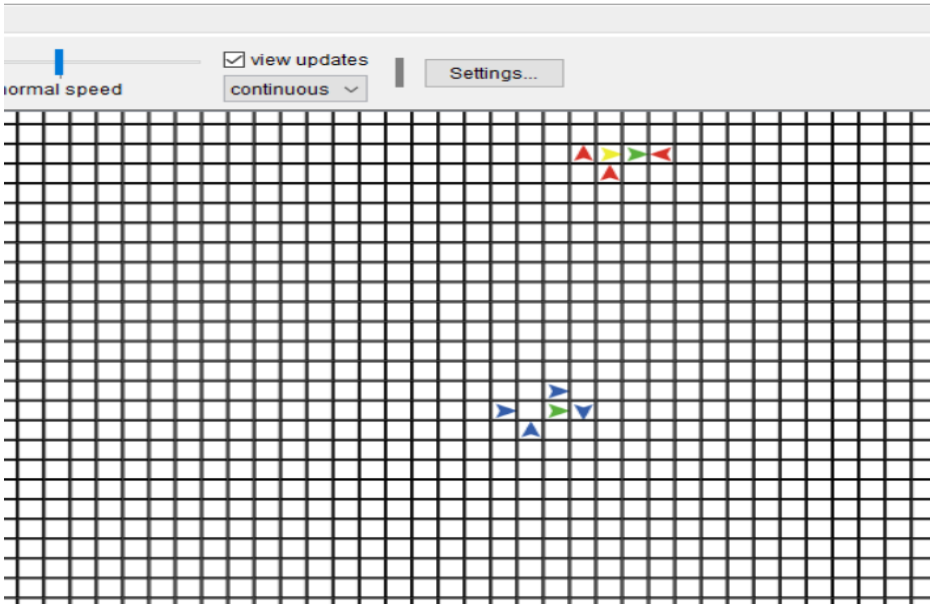

Figure 17: Capture des évadés

La **Figure 17** au-dessous montre l'écran d'interface Net Logo pour notre simulation, en cliquant sur le bouton setup on obtient l'état initial du programme, Le bouton **last-case** renvoi la simulation de notre algorithme « **hybrid path planning** ».

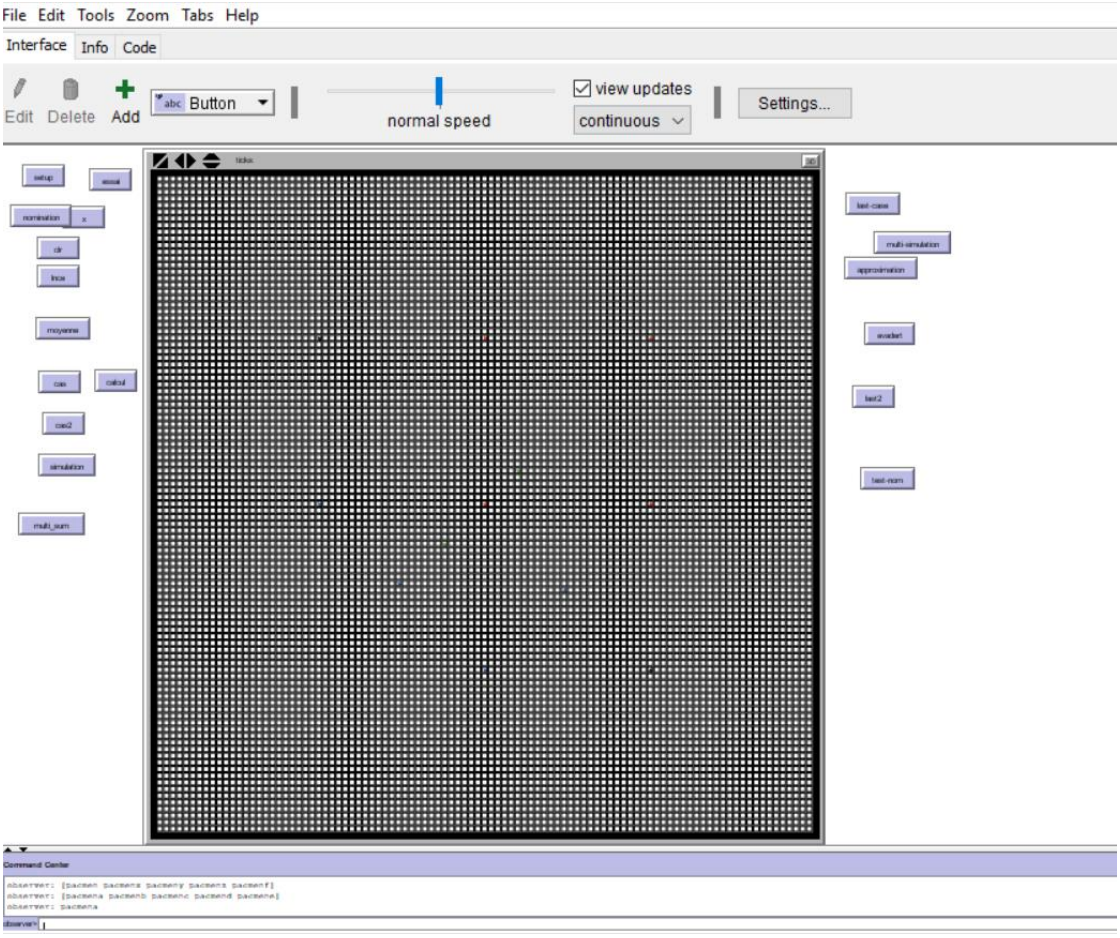

Figure 18: l'interface NetLogo pour notre simulation

Pour prouver l'efficacité de notre algorithme, nous avons vu l'utilité de les comparer avec l'autre algorithme « Leader Follower » :

Cas 01: Leader Follower algorithm

Cas 02: hybrid path planning algorithm

Les résultats de la simulation sont basés sur l'étude de capture des évadés

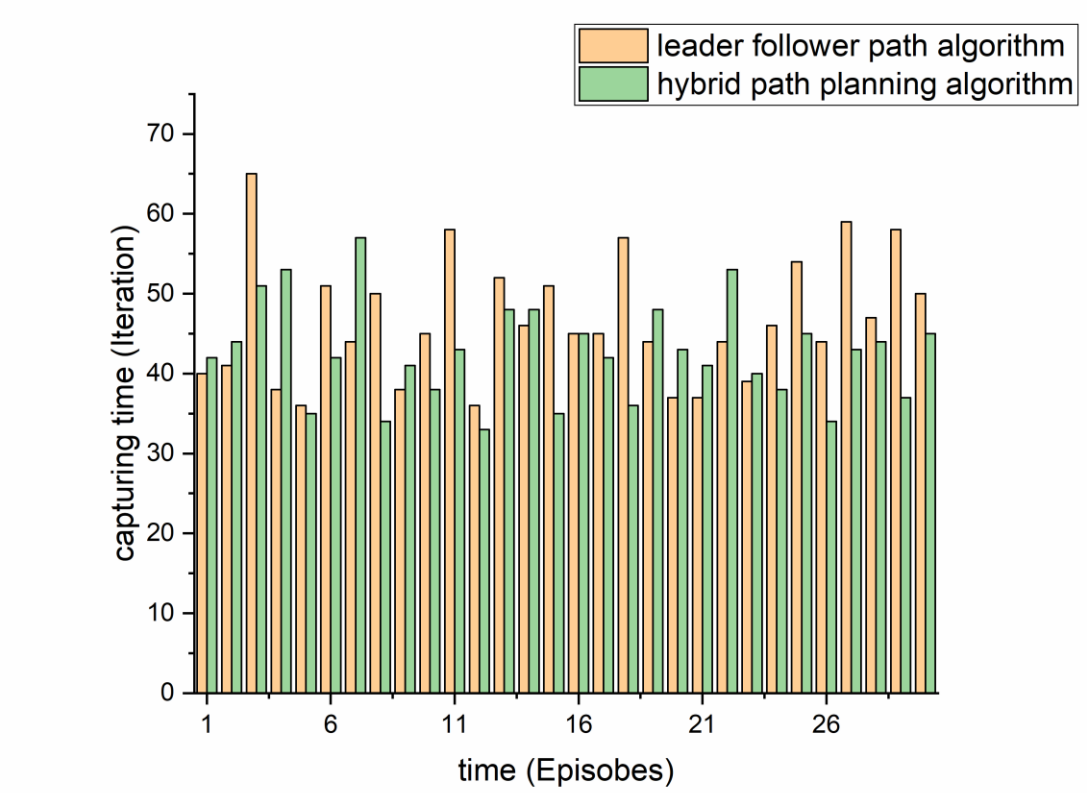

Figure 19 : Temps de capture (Itérations)

- **Résultat 01 :** Dans la premier **approche (leader follower path algorithm),** la durée de capture moyen obtenu est égal à 46.5 itérations.

  Mais en notre deuxième approche **(hybrid path planning algorithm),** nous constatons une diminution allant jusqu'à 42,**6** **(hybrid path planning algorithm).**

  On note que dans 12 sur 30 épisodes, Représente la durée minimale de poursuite est 33, dans l'approche **(hybrid path planning algorithm)** en comparaison avec l'autres approche.

  Cela montre la préférence de cette approche proposée par rapport à la première. Cette préférence est le résultat de changement les conditions des récompenses dans la deuxième approche.

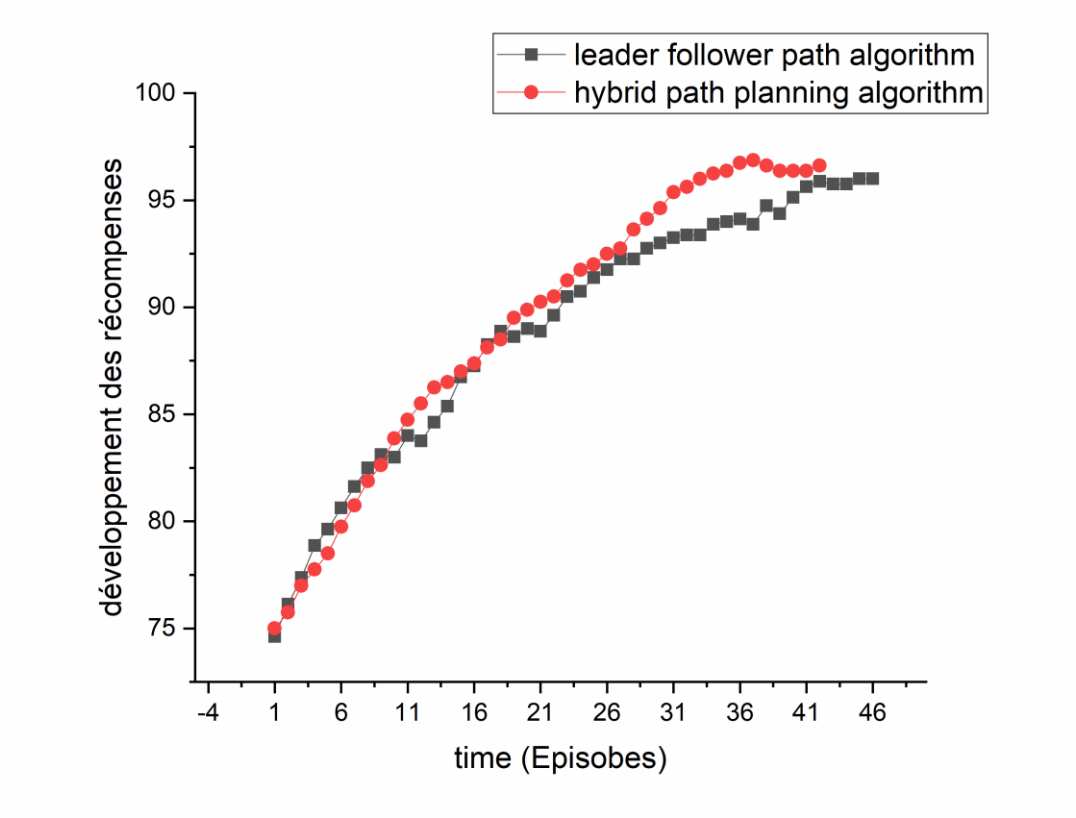

Figure 20: Le développement des récompenses des poursuivants

- **Résultat 02**: Le nombre d'itération **(leader follower path algorithm): 46**
  Le nombre d'itération (hybrid path planning algorithm) : 42
  En la premier approche **(leader follower path algorithm)**
  Nous remarquons que le maximum de récompenses **95,87** est acquis durant l'itération **42**
  Par rapport la deuxième approche **(hybrid path planning algorithm)** le maximum de récompense **96,87** acquis durant l'itération **37,** Dans notre approche proposée, nous obtenons plus des récompenses dans moins de temps que la première approche, ce qui indique la préférence de cette approche.

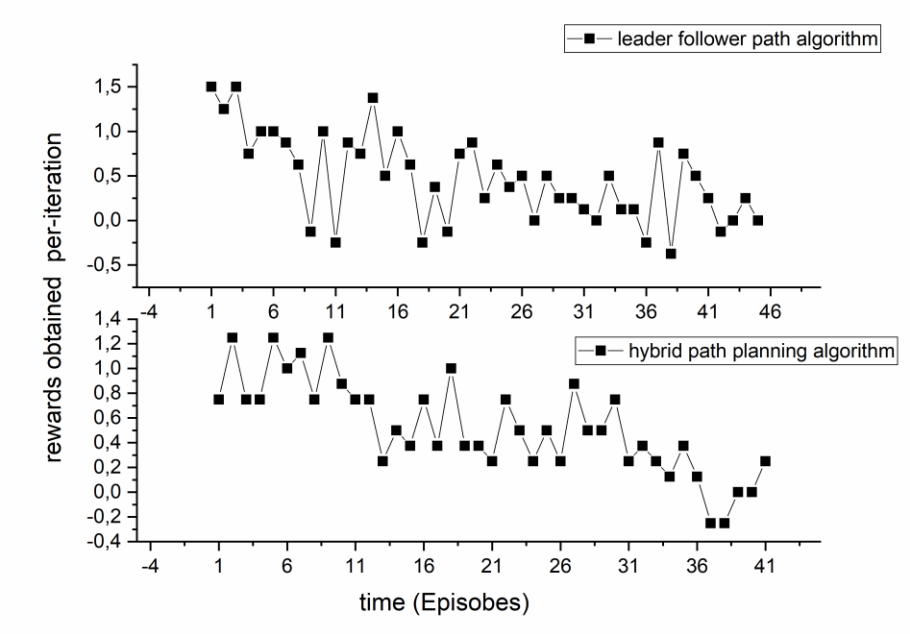

Figure 21 : Récompense moyenne obtenue par itération pendant un épisode complet de la poursuite

- **Résultat 03 :** Cette courbe montre les récompenses obtenues par itération au cours d'un épisode complet de la poursuite, Dans notre approche **(hybrid path planning algorithm)** on note uniquement deux diminutions négatives de la récompense, aux itérations 37 et 38 mais dans l'approche **(leader follower path algorithm) on** note 8 diminutions négatives de la récompense (comportement intelligent de l'évadé), ce qui indique la préférence de l'approche **(hybrid path planning algorithm)**.
  Moyenne **(hybrid path planning algorithm)** = 0,527439
  Moyenne **(leader follower path algorithm) =** 0,475

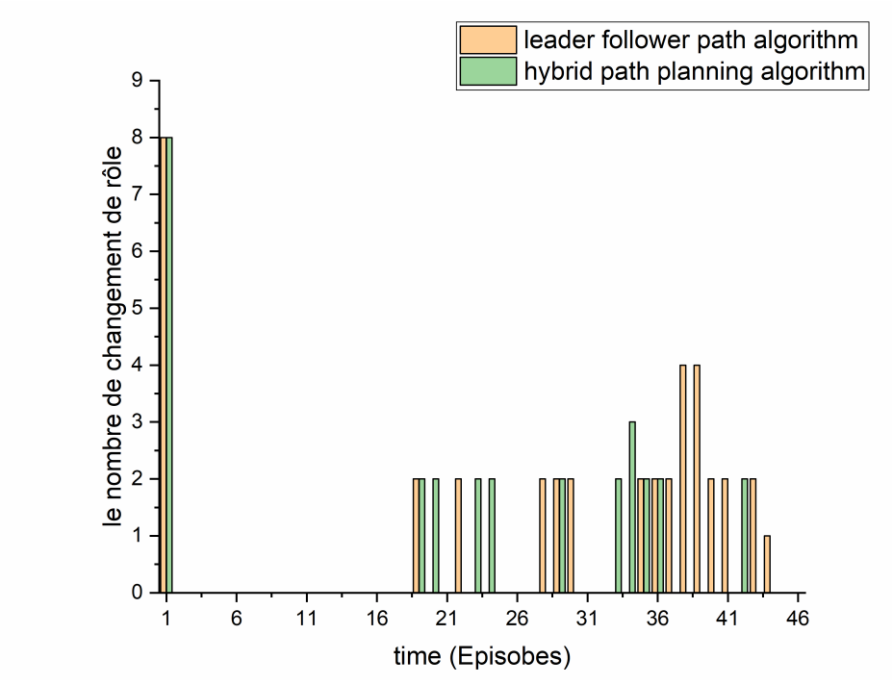

Figure 22: Nombre de changement de rôle

- **Résultat 04 :** Cette courbe Représente le nombre de changement de rôle des poursuivants Avec l'approche **(hybrid path planning algorithm)** on atteint une certaine stabilité (moyenne est égale à **0,69** changements) des poursuivants dans leurs groupe durant la poursuite meilleure que l'approche **(leader follower path algorithm)** (moyenne est égale à **0,84** changements).

  Ces moyennes prouvent que la nouvelle approche **(hybrid path planning algorithm)** proposée procure une certaine stabilité aux groupes de poursuite. Ce fait influence positivement le temps de capture ainsi que l'acquisition des récompenses durant l'exécution des taches.

Le tableau **3** résume tous les résultats obtenus lors de la partie simulation.

A partir de ces résultats, on peut facilement remarquer la supériorité de **(hybrid path planning algorithm)** proposée par rapport à **(leader follower path algorithm)**. 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).

	Le temps de capture (itération)	Récompense moyenne obtenue par itération pendant un épisode complet	Le développement des récompenses des poursuivants	La Moyenne de Changement de rôle (Stabilité)
Cas 01	46.5	0,475	95,875	0,847826
Cas 02	42,6	0,527439	96,875	0,690476

Tableau 4 : Résultats obtenus lors de la simulation

Ces résultats obtenus lors de la simulation, à cause le changement d'algorithme, Comme le changement d'algorithme a joué un grand rôle dans l'amélioration de tous les paramètres étudiés, dans le premier algorithme, les poursuivants suivent le leader uniquement pour capturer l'évadé, mais dans notre algorithme proposé, nous avons changé la méthode de poursuite, car les poursuivants comparent les récompenses. Entre le leader et l'évadé et choisir la meilleure (La plus grande) et cet algorithme nous a aidé à améliorer l'acquisition des récompenses en un temps de capture court, contrairement au premier algorithme, La **figure 23** ci-dessous montre la différence entre les deux approches.

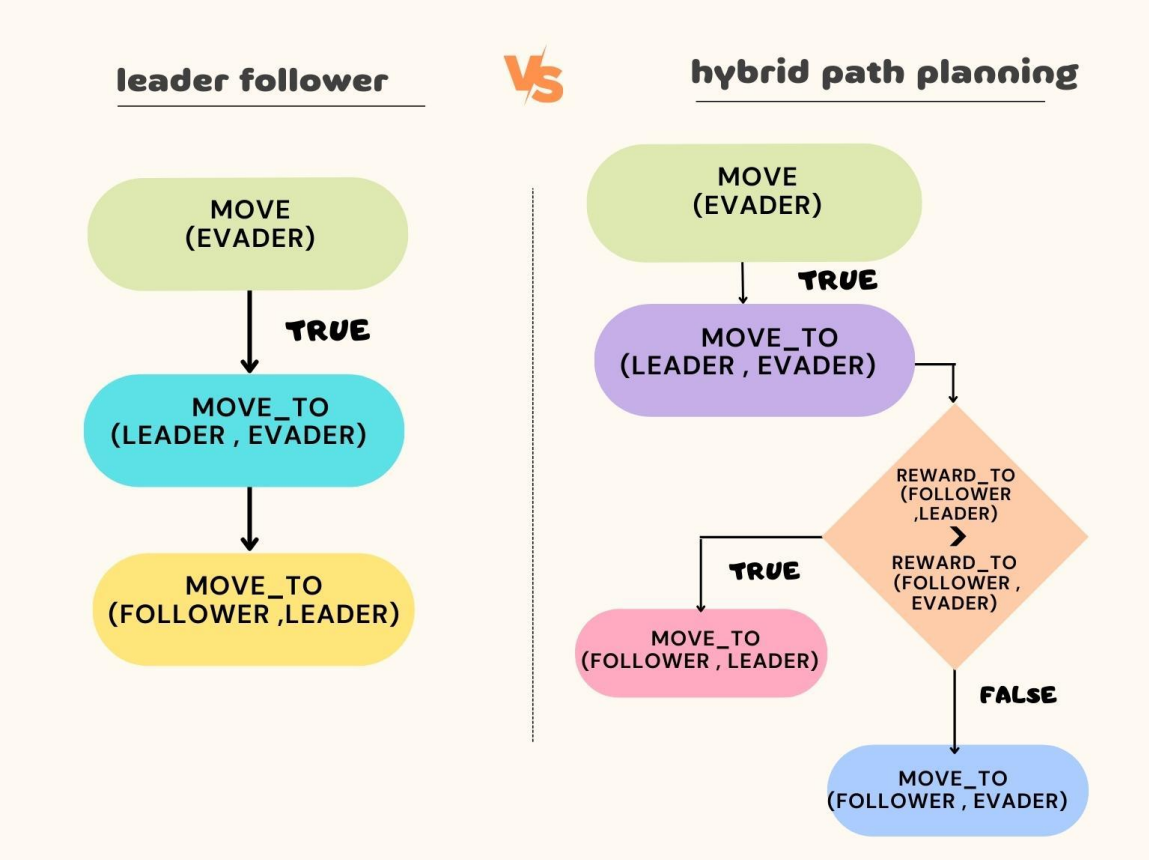

Figure 23: Comparaison des deux algorithmes

4.3 Conclusion :

Dans ce chapitre, nous nous sommes concentrés sur l'implémentation de l'algorithme de (hybrid path planning) afin de traiter le problème de poursuite-évasion. Pour cela, nous avons utilisé la plateforme Netlogo que nous avons jugé comme la plus apte à faciliter cette implémentation. Pour démontrer l'efficacité de cette nouvelle approche, nous l'avons comparé à un algorithme de (hybrid path planning) et leader follower path.

Durant les simulations, nous nous sommes basés sur le temps de capture, la stabilité des groupes, et ainsi que l'acquisition des récompenses durant la poursuite. Les résultats expliqués dans cette partie de ce chapitre prouvent que cette approche (hybrid path planning) améliore tous les paramètres étudiés, comme la durée de capture est courte et inférieure à la durée de capture de la

première approche (leader follower path), nous avons également obtenu des récompenses dans notre approche (hybrid path planning) plus que les récompenses de la première approche, et c'est ce qui distingue notre approche, qui est d'obtenir de nombreuses récompenses en peu de temps de capture et aussi notre approche plus stables que la première approche.

Conclusion générale

Ce mémoire avait pour ambition de réaliser un nouvel algorithme de la planification de la trajectoire d'un système multi-agent basé sur la collaboration hiérarchique des agents assignés à la même tâche lors de leurs déplacements dans l'environnement.

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

La deuxième étape consiste à introduire le sujet en précisant les grandes étapes pour pouvoir réaliser la troisième étape qui introduit le nouvel algorithme proposé basé sur l'apprentissage par renforcement ainsi que le Processus Décisionnel de Markov, Cette dernière vise à augmenter le degré de coopération entre les agents.et il améliore l'acquisition des récompenses en un temps de capture court. Cette application a été effectuée en comparaison avec d'autres algorithme (leader follower path planning).

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. Les simulations sont créées sous la plateforme **NetLogo 5.0.0.**

Enfin, les résultats obtenus reflètent la différence entre les deux algorithmes comparés, avantageusement pour la nouvelle planification de trajectoire proposée.

Pour conclure, notre perspective concernant le sujet étudié est l'application de la nouvelle approche proposée dans d'autres problèmes complexes tels que les systèmes distribués.

Références

Références

(01) KARIMA, B. (1945). *Conception d'un système multi-agents adaptatif pour la résolution de problème* (Doctoral dissertation, UNIVERSITE BADJI MOKHTAR-ANNABA).

(02) FERBER, Jacques et WEISS, Gerhard. *Multi-agent systems: an introduction to distributed artificial intelligence*. Reading: Addison-Wesley, 1999.

(03) J.P. Briot, Y. Demazeau, "Introduction aux agents: Principes et architecture des systèmes multi-agents", Collection IC2, Hermès, France, 2001J.P. Briot, Y. Demazeau, "Introduction aux agents: Principes et architecture des systèmes multi-agents", Collection IC2, Hermès, France, 2001

(04) WOOLDRIDGE, Michael et JENNINGS, Nicholas R. Pitfalls of agent-oriented development. In: *Proceedings of the second international conference on Autonomous agents*. 1998. p. 385-391.

(05) http://www.univ-oeb.dz/fsesnv/wp-content/uploads/2019/04/Cours-SMA-Complet.pdf

(06) Ferber J, Les systèmes multi-agents, vers une intelligence collective. Edition Inter-éditions, 1995.

(07) M. Wooldridge, N. R. Jennings., D.Kinny, The Gaia Methodologyfor Agent- Oriented Analysis and Design, Journal of Autonomous Agents and Multi-Agent Systems, vol. 3, n° 3, p. 285-312, Kluwer Academic Publishers, 2000.

(08) O., Boissier, S., Gitton, P.Glize, Caractéristiques des Systèmes et des Applications. Systèmes Multi-Agents, Observatoire Français des Techniques Avancées,

ARAGO 29, Diffusion Editions TEC & DOC, pp. 25-54, 2004.

(09) Russell S., Norvig P. Intelligence artificielle. Pearson Éducation 2édition.

2006.

(10) https://fr.slideshare.net/GilJGH/introduction-netlogo

(11) Durfee, E. H. (2001). Scaling up agent coordination strategies. *Computer*, *34*(7), 39-46.

(12) Gelernter D. et Carriero N., *Coordination Languages and Their Significance*. *Communications of the ACM*, n°35(2), pp. 96-107, 1992.

(13) Gerhard. A modern approach to distributed artificial intelligence. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev*, 1999, vol. 22, no 2.

(14) Huhns M and Singh M. P., "CKBS-94 Tutorial: Distributed Artificial Intelligence for Information Systems." Dake Centre, University of Keele. 1994

Références

(15) Nwana H. S., Lee L., and Jennings N. R., "Co-ordination in Multi-Agent Systems."Lecture Notes in Computer Science; 1198 (c) Springer-Verlag Berlin Heidelberg 1997.

(16) PLANNING, Spatial. A Configuration Space Approach. *IEEE Trans, on Computers (C*, 1983, vol. 32, p. 108-120.

(17) 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.

(18) https://www.javatpoint.com/reinforcement-learning#What

(19) WEI, Mo, CHEN, Genshin, CRUZ JR, Jose B., *et al.* Multi-pursuer multi-evader pursuit-evasion games with jamming confrontation. *Journal of Aerospace Computing, Information, and Communication*, 2007, vol. 4, no 3, p. 693-706.

(20) EL HABIB SOUIDI, Mohammed, SIAM, Abderrahim, PEI, Zhaoyi, *et al.* Jeu de poursuite-évasion multi-agents basé sur l'architecture organisationnelle. *Journal de l'informatique et des technologies de l'information,* 2019, vol. 27, n° 1, p. 1-11.

(21) Sincák, D. (2009). Multi–robot control system for pursuit–evasion problem. *J. Electr. Eng*, *60*(3), 143-148

(22) SOUIDI, Mohammed El Habib, SONGHAO, Piao, GUO, Li, *et al.* multi-agent cooperation pursuit based on an extension of AALAADIN organisational model. *Journal of Experimental & Theoretical Artificial Intelligence*, 2016, vol. 28, no 6, p. 1075-1088.

(23) SOUIDI, Mohammed El Habib, PIAO, Songhao, et LI, Guo. Mobile agents path planning based on an extension of Bug-Algorithms and applied to the pursuit-evasion game. In : *Web Intelligence*. IOS Press, 2017. p. 325-334.

(24) O., Bordini, R. H., Hübner, J. F., Ricci, A., & Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, *78*(6), 747-761.

(25) Treur J., Report of the First SIGMeeting, Contribution toMethodologies/software engineering SIG, Rapport, First SIG Meeting – AgentLink, 1998.

(26) Jennings N. R., « Agent-Oriented Software Engineering », Garijo F. J., Boman M. (dir.), Multi-Agent System Engineering, 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99, Valence, Espagne, juin-juillet 1999, vol. 1647 de LNAI, Springer, 1999.

(27) Benaouda, A. (2018). *Contribution à la conception et à l'implémentation d'un langage de spécification formelle dédié à la e-maintenance des systèmes de production par l'approche des Systèmes Multi-Agents* (Doctoral dissertation).

(28) TISUE, Seth et WILENSKY, Uri. Netlogo: A simple environment for modeling complexity. In : International

(29) CHAVERRI, Francisco Quesada. INTRODUCCIÓN AL LENGUAJE NETLOGO Y LA PROGRAMACIÓN BASADA EN AGENTES. 2018.

Références

(30) STIGBERG, David. An introduction to the Netlogo modeling environment. In : *Ecologist-Developed Spatially-Explicit Dynamic Landscape Models*. Springer, Boston, MA, 2012. p. 27-41.

(31) ZHU, Zhe-Yang et LIU, Cheng-Lin. A Novel Method Combining Leader-Following Control and Reinforcement Learning for Pursuit Evasion Games of Multi-Agent Systems. In : 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE, 2020. p. 166-171.