DEPARTMENT DE SCIENCES ET INGÉNIERIE

PROJET DE FOSYMA

Dedale - Exploration coopérative de l'environnement et Capture des Golems

Groupe 11:
Jawher Jriby 3702809
Sama Satariyan 21318843

Sommaire

1 Introduction **1**

2 Exploration et Communication **2**
2.1 ExploreCoop Agent 2
2.2 Stratégie 2
2.2.1 Priorisation des neouds 3
2.2.2 Limites 3
2.2.3 Complexité 3

3 Chasse **4**
3.1 Hunter Agent 4
3.2 Stratégie 4
3.2.1 Complexité 5
3.2.2 Limites 5

4 Conclusion **6**

1 Introduction

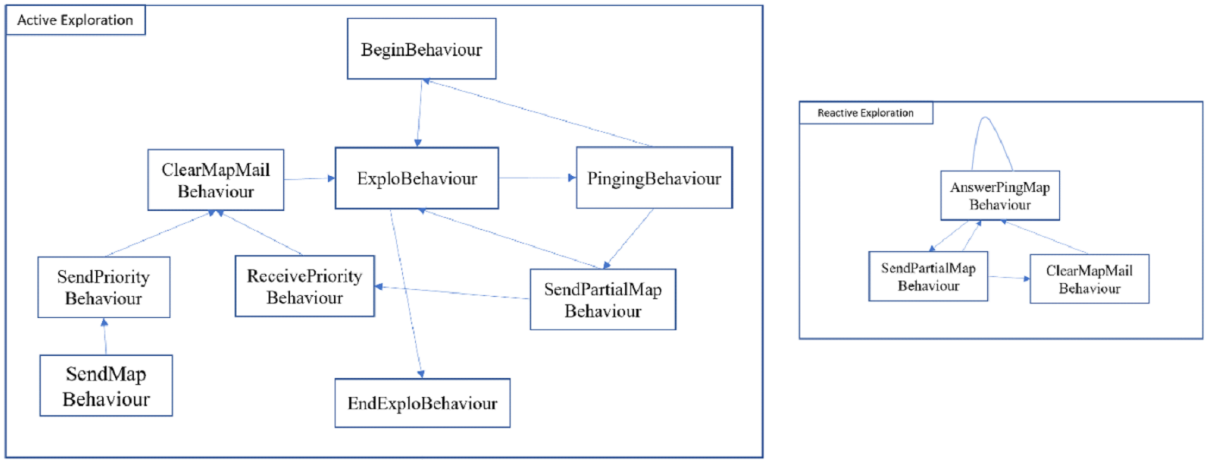
Le jeu "Hunt the Wumpus", conçu par Gregory Yob en 1972, est un des pionniers des jeux informatiques. Dans ce jeu, un joueur solitaire navigue à travers un labyrinthe souterrain peuplé de pièges mortels et de trésors cachés. L'objectif est de traquer et de tuer le Wumpus, une créature terrifiante, sans tomber dans les gouffres ou être tué par des chauves-souris superstitieuses. Ce jeu classique combine exploration, stratégie et prise de risque, posant les bases des mécanismes de jeux d'aventure et de survie.

Dans le cadre de notre projet FOSYMA, nous allons réinventer "Hunt the Wumpus" en introduisant une dimension multi-agent où la coopération et l'interaction complexe entre les agents seront au cœur du gameplay. Notre projet consistera à développer une variante où une équipe d'agents coopératifs devra explorer conjointement un environnement inconnu. La tâche centrale sera l'exploration efficace de la carte, nécessitant une communication et une coordination sophistiquée entre les agents pour partager des informations en temps réel et construire une représentation complète de l'environnement. De plus, les agents devront développer des stratégies pour chasser les Golems, en utilisant des tactiques de blocage et d'encerclement pour les immobiliser et les capturer.

2 Exploration et Communication

2.1 ExploreCoop Agent

Le type d'agent ExploreCoop est responsable de l'exploration de l'environnement. Il possède son propre ensemble de comportements. Les FSM qui contrôlent le flux du comportement de l'agent ExploreCoop sont présentés dans les diagrammes ci-dessous.



2.2 Stratégie

Les agents s'inscrivent d'abord dans l'environnement, puis commencent à observer et à visiter les nœuds du graphe. L'agent explore l'environnement, prend des décisions basées sur des observations et une stratégie de cartographie. L'agent marque son nœud actuel comme étant fermé dans sa représentation cartographique afin d'éviter d'y revenir. Il examine les nœuds environnants et les ajoute à la carte. Les nouveaux nœuds sont marqués et les arêtes entre les nœuds sont établies. La sélection du nœud suivant est basée sur la liste de priorités reçue par l'agent. S'il existe des nœuds prioritaires et qu'ils sont directement accessibles, ou qu'un chemin vers eux peut être calculé, l'agent suit ce chemin. Si aucun nœud prioritaire ne requiert une attention immédiate, l'agent calcule le chemin le plus court vers le nœud non visité (ouvert) le plus proche et s'y rend. Cela permet de couvrir efficacement tous les nœuds. L'agent continue à se déplacer vers de nouveaux nœuds jusqu'à ce qu'il n'y ait plus de nœuds non visités disponibles, la phase d'exploration étant alors considérée comme terminée.

2.2.1 Priorisation des neouds

À la réception d'un message qui correspond à des critères spécifiques il tente de désérialiser le contenu dans un objet de Knowledge, qui comprend un graphe et une liste de nœuds avec leurs distances respectives par rapport au partenaire. Le comportement compare ensuite ces distances avec ses propres distances par rapport à ces nœuds, calculées à partir de la position actuelle. Les nœuds plus proches de l'agent que du partenaire sont ajoutés à la liste des priorités de l'agent. Il prépare ensuite un nouveau message à envoyer à l'agent partenaire. Ce message contient les nœuds considérés comme plus proches du partenaire que de l'expéditeur.

Priority Sending behavior: En d'autres termes, il s'attend à recevoir un objet Knowledge, qui comprend une carte et des détails sur les nœuds avec les distances. Le message filtre le protocole (SHARE-PARTIAL-MAP) et l'expéditeur. Il analyse ensuite les données reçues pour déterminer les zones prioritaires sur la base des calculs de distance - en comparant ses propres distances par rapport aux nœuds à celles indiquées par le partenaire. Après avoir déterminé les nœuds prioritaires pour le partenaire sur la base de la proximité, il rassemble ces informations dans un message et l'envoie au partenaire. Ce message utilise un protocole différent (SHARE-PRIORITY).

Priority Reception and Update behavior: Lorsqu'il reçoit un message (SHARE-PRIORITY) d'un partenaire spécifique, il met à jour sa liste de priorités interne en fonction du contenu, ce qui influe sur les actions ou les stratégies de navigation ultérieures.

2.2.2 Limites

Le problème de cette idée est que lorsque le nombre de nœuds augmente, le calcul peut devenir un surcoût. Un autre problème est que la hiérarchisation est basée uniquement sur des comparaisons de distance, ce qui ne conduit pas toujours à des décisions optimales, car cela ne tient pas compte d'autres facteurs tels que l'importance stratégique des nœuds. Dans les travaux futurs, au lieu de se concentrer uniquement sur la proximité, nous pourrions hiérarchiser les nœuds en fonction de l'intensité de l'odeur que les golems ont laissée derrière eux.

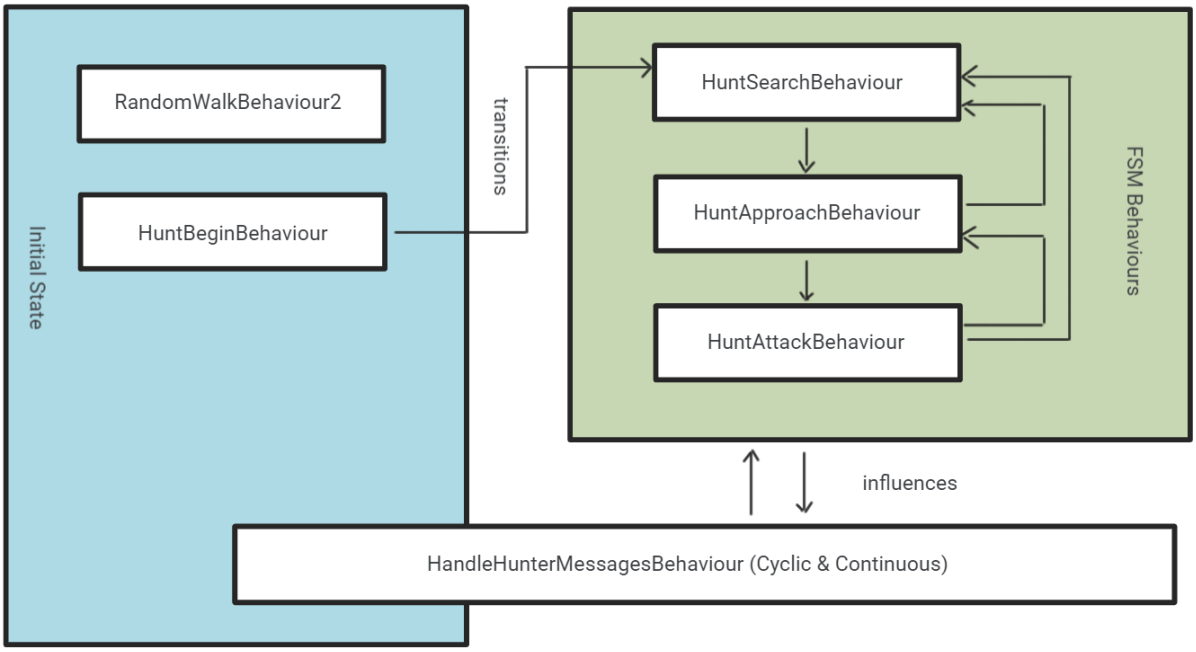
2.2.3 Complexité

La stratégie d'exploration et de priorisation des nœuds par les agents dans un graphe présente une complexité considérable en termes de temps, de mémoire et de communication. Les agents marquent et connectent systématiquement les nœuds visités, augmentant ainsi le temps de calcul, particulièrement dans des graphes denses. Ils maintiennent aussi une carte interne, ce qui élève la consommation de mémoire, surtout dans des environnements complexes. En outre, la nécessité de communiquer fréquemment pour synchroniser les informations sur les positions et les priorités peut surcharger le réseau et augmenter la latence. Cette approche peut devenir ressource-intensive dans des scénarios à grande échelle avec de nombreux nœuds ou agents.

3 Chasse

3.1 Hunter Agent

Le type d'agent Hunter est responsable de chasse de golems. Il possède son propre ensemble de comportements. Les comportements de l'agent Hunter sont présentés dans les diagrammes ci-dessous.



3.2 Stratégie

Dans la mise en œuvre de la stratégie d'encerclement des golems, chaque agent détecte des signaux uniques émis par les golems, souvent conceptualisés comme une "odeur" dont la valeur du rayon varie d'un golem à un autre. Cette détection signale la proximité immédiate d'un golem. Lorsqu'un tel signal est détecté, l'agent responsable de la découverte communique rapidement position exacte où l'odeur a été detectée, à l'ensemble du réseau d'agents. Cette transmission d'informations permet une mobilisation rapide et coordonnée vers la source de l'odeur. En progressant, si un agent rencontre un obstacle qui empêche tout avancement ultérieur qui ne s'agit pas d'un agent (après communication), il conclut logiquement que le golem se trouve très probablement dans un noeud adjacent. Sur cette supposition, les agents se déploient de manière tactique autour de ce point central, occupant les noeuds voisins pour encercler le golem et ainsi bloquer toutes ses voies de sortie potentielles. Ce positionnement ciblé et ces mouvements coordonnés sont essentiels pour contenir efficacement le golem dans un périmètre restreint.

3.2.1 Complexité

La complexité temporelle de la stratégie d'encerclement des golems dépend de la rapidité avec laquelle les agents détectent le golem, communiquent sa position et se positionnent stratégiquement. Cela est influencé par l'efficacité des capteurs et la rapidité de la réponse des agents dans leur environnement. Sur le plan de la mémoire, chaque agent doit gérer des informations substantielles, incluant les détails de l'environnement, la position des autres agents, et les données sur le golem, ce qui peut nécessiter une mémoire importante, surtout si des techniques d'apprentissage automatique sont impliquées. En termes de complexité de communication, la stratégie exige une communication rapide et fiable entre tous les agents, qui doivent échanger des informations volumineuses et synchronisées, rendant la gestion du trafic de communication et la cohérence des données des enjeux critiques.

3.2.2 Limites

La stratégie d'encerclement des golems, tout en étant efficace, est limitée par plusieurs facteurs. Elle repose fortement sur une communication fluide et synchronisée entre les agents; tout retard ou interruption peut désynchroniser l'effort et permettre au golem de s'échapper. De plus, la mobilité et l'adaptabilité élevées du golem peuvent le rendre capable d'éviter l'encerclement, surtout si les agents ne peuvent pas accéder rapidement à toutes les positions requises en raison de la complexité de l'environnement. Cette stratégie demande également un nombre conséquent d'agents, ce qui peut s'avérer coûteux en ressources et limiter leur disponibilité pour d'autres golems.

4 Conclusion

Bien que le projet ait atteint de nombreux objectifs, il présente certaines limites et domaines d'amélioration. Premièrement, la stratégie d'exploration et de priorisation des nœuds se heurte à des défis de scalabilité lorsque le nombre de nœuds augmente, ce qui peut entraîner un surcoût en calcul et une consommation excessive de ressources.

En outre, la stratégie d'encerclement des golems, bien qu'efficace, requiert une synchronisation et une coordination impeccable entre les agents. Tout retard dans la communication ou toute défaillance dans le positionnement peut compromettre l'efficacité de l'encerclement, permettant potentiellement aux golems d'échapper.

Pour améliorer l'adaptabilité et la robustesse du système dans notre projet, on peut intégrer des capacités d'adaptation dynamique permettant aux agents de modifier leurs stratégies en temps réel en réponse aux changements de l'environnement, tels que l'apparition de nouveaux obstacles ou les modifications des comportements adverses. Par ailleurs, renforcer la gestion des erreurs et des exceptions est crucial pour accroître la fiabilité du système, en développant des routines avancées pour la récupération après des pannes de communication ou des erreurs d'exécution. En complément, l'intégration d'algorithmes d'apprentissage automatique permettrait aux agents d'apprendre de leurs expériences antérieures, optimisant ainsi leurs décisions futures grâce à des techniques telles que l'apprentissage par renforcement, ce qui améliorerait significativement leurs performances au fil du temps