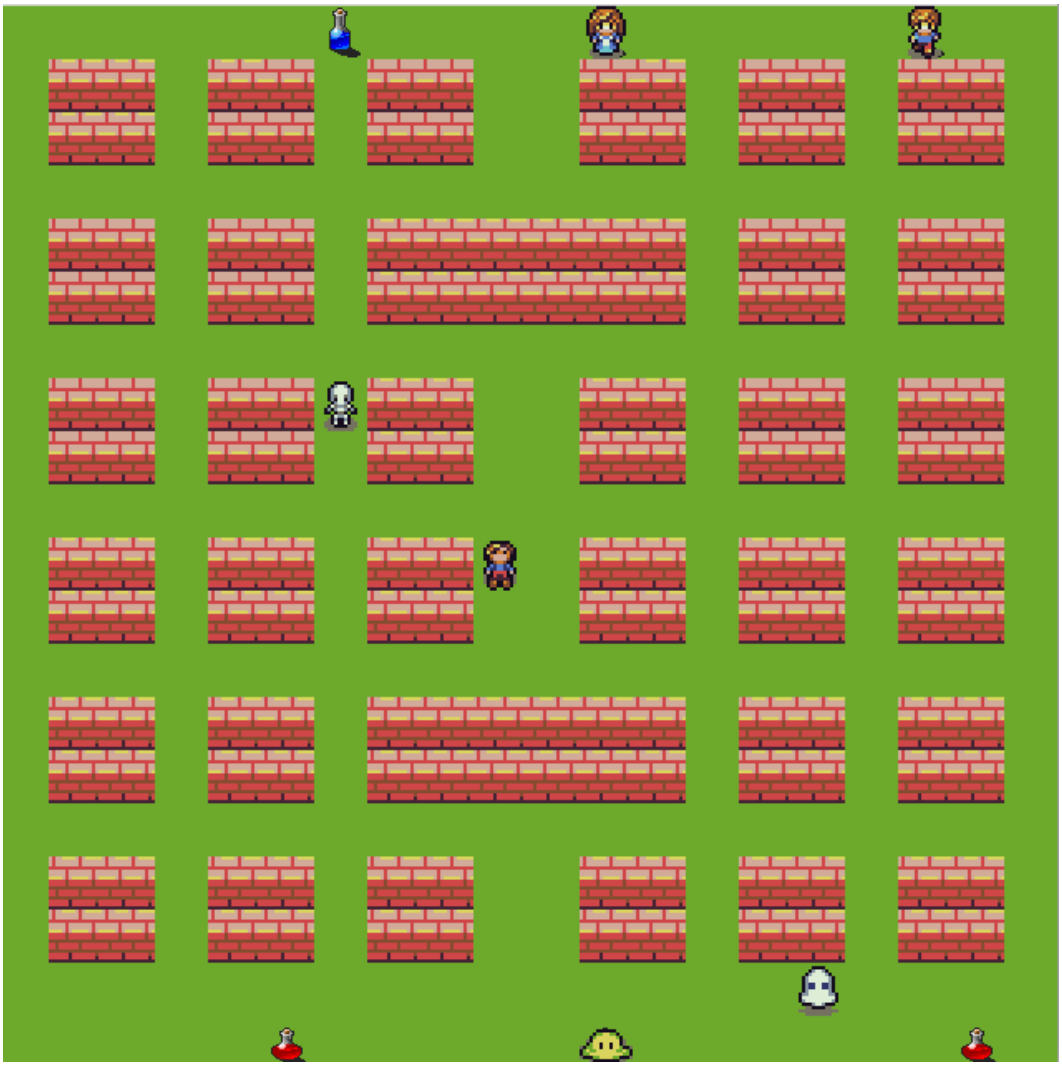
SORBONNE UNIVERSITÉ

Multi-Agent PathFinding



AZAR Jane - 3803112

SAICHI Lina - 3803151

LU3IN025 - Intelligence Artificielle et Jeux (Groupe 2)

Enseignante: BOURDACHE Nadjet

Introduction

Ce projet consiste à étudier une version adversariale du cooperative path-finding. Deux équipes d'agents sont en compétition et cherchent à atteindre leurs destinations avant leurs adversaires.

L'objectif du projet est de développer et de comparer plusieurs stratégies pour les équipes d'agents. A chaque agent est attribué un objectif unique, ce dernier est différent pour chaque agent.

Nous avons eut recours à implémenter trois stratégies différentes:

- Coopérative A * pour une équipe.
- A * indépendants au sein d'une équipe.
- Path-slicing.

Les déplacements des équipes suivent quelques règles:

- Les tours alternent entre les deux équipes
- Les agents se déplacent sur une case adjacente ou restent au même endroit
- Les agents peuvent se déplacer vers une case si celle-ci est libre ou libérée dans le même tour par un joueur de la même équipe.

Nous détaillerons les stratégies citées ci-dessus dans ce qui suit.

Stratégie 1: Coopérative A*

Explication de la stratégie utilisée

Cette stratégie consiste à calculer le path d'un joueur en prenant en considération les paths des autres joueurs de son équipe. Le but est d'éviter une collision avec eux. Autrement dit, les agents au sein d'une même équipe sont coopératifs entre eux. En connaissant les futurs mouvements de ces agents, il doit calculer son path sans les déranger *(ni dans le présent ni dans le futur).*

Explication de l'algorithme utilisé

Initialisation : calcul du path de tous les joueurs en utilisant A star

Pour chaque joueur :

Si j'ai ete immobile a l'itération d'avant :

Calcul du path avec l'algorithme de Astar en prenant en compte le path de mes équipiers : je ne dois pas passer par une case au même moment que l'un d'entre

eux. Je vais aussi regarder mes ennemis qui sont trop proches de moi, afin d'anticiper mon prochain coup.

Mise à jour du path : concaténation de celui d'avant et celui nouvellement calculé et l'ajouter dans la réservation pour que mes collègues en prennent conscience.

Faire bouger le joueur dans la case qui suit.

Mettre à jour ma position dans la liste des positions afin de faire connaître ma position actuelle.

Structures utilisées : pour gérer la coordination au sein d'une équipe, nous avons créé deux tables de réservations (une pour chaque équipe). Cette table contient les cases réservées par les joueurs de l'équipe pour un instant t. C'est un dictionnaire qui a comme clés les cases (x,y) et pour chaque case, le temps ou elle a été réservée et par quel agent.

Afin d'éviter les croisements entre deux agents qui veulent s'échanger leur positions, pour chaque case dans le path d'un joueur, ce dernier réserve la case pour l'instant t et l'instant suivant t+1.

Points forts et points faibles

Points forts	Points faibles
- Éviter les croisements entre les agents d'une même équipe. - La table de réservation peut être efficacement implémentée comme une table de hachage. - Inclut une troisième dimension qui est le temps pour connaître les intentions et les futurs mouvements des agents.	- En général, de tels algorithmes sont sensibles à l'ordre des agents, ce qui nécessite que des priorités raisonnables soient sélectionnées pour obtenir de bonnes performances. - Le cas de base est d'utiliser la distance de Manhattan. Cependant, cela peut donner de mauvaises performances dans des environnements plus difficiles.

Stratégie 2: A* indépendant

Explication de la stratégie utilisée

Cette stratégie consiste à calculer mon path en utilisant A*, le joueur n'a aucune connaissance sur les autres joueurs *(les membres de son équipe ainsi que les membres de l'équipe adversaire).* Lorsque le joueur essaye de bouger, il vérifie que sa nouvelle case n'est pas occupée par un autre joueur afin d'éviter de faire des collisions, dans le cas où la case est occupée le joueur re-calcule son path.

Explication de l'algorithme utilisé

Initialisation : calcul du path de tous les joueurs en utilisant A star

Pour chaque joueur :

Si je n'ai toujours pas atteint mon but:

Si la case qui suit est occupée par un agent (adversaire ou ami) :

Je recalcule un nouveau path commençant par ma position actuelle jusqu'au but.

Si la case du nouveau path est vide :

Je bouge vers la nouvelle case de ce path.

Sinon :

Je reste immobile jusqu'à la nouvelle itération en espérant que d'ici là l'agent qui l'occupe aura bougé, et reprendre mon chemin.

Points forts et points faibles

Lien

Points forts	Points faibles
-Il est intéressant de noter qu'au fil du jeu, la carte du jeu est modifiée continuellement. Le chemin initial n'est donc peut être plus le plus optimal.	-Ce n'est pas une bonne idée de recalculer le path si nous sommes confrontés à des chemins très longs.

Stratégie 3: Path slicing

Explication de la stratégie utilisée

Cette stratégie est assez similaire à la deuxième stratégie *(A* indépendant)*. En cas de rencontre d'un autre joueur, le joueur recalcule son path entre sa position actuelle et les cases suivantes (au lieu de recalculer tout le path, le joueur recalculer un morceau de son path) en prenant en considération les positions actuelles des autres joueurs.

Explication de l'algorithme utilisé

Initialisation : calcul du path de tous les joueurs en utilisant A star

Pour chaque joueur :

Si je n'ai toujours pas atteint mon but:

Si la case qui suit est occupée par un agent (adversaire ou ami) :

Je recalcule un nouveau path commençant par ma position actuelle jusqu'à la case après la collision.

Si la case du nouveau path est vide :

Je bouge vers la nouvelle case de ce path.

Sinon :

Je reste immobile jusqu'à la nouvelle itération en espérant que d'ici là l'agent qui l'occupe aura bougé, et reprendre mon chemin.

Points forts et points faibles

Points forts	Points faibles
- Il résout plus d'instances sur des grilles plus petites et s'adapte de manière fiable à 100 agents ou plus sur des cartes de jeu plus grandes. Lien - Permet de ne pas recalculer le chemin entier, mais un bout de chemin seulement. - Les chemins épissés peuvent être trop courts pour permettre au chemin de remplacement de contourner l'obstacle proprement, davantage de chemins sous-optimaux seront trouvés. - L'épissage du chemin est beaucoup plus rapide que le recalcul du chemin - Les mauvais chemins peuvent souvent être détectés en regardant la longueur du nouveau chemin	- Le nouveau chemin calculé peut être plus long que le chemin initial. Ce qui pourrait dire que la meilleure solution serait peut-être parfois d'attendre que la case se libère. - Il ne répond pas bien aux changements majeurs dans le chemin. Il est possible de détecter plusieurs de ces situations et d'utiliser le recalcul du chemin à la place. - Le path slicing ne gère pas non plus les situations où les unités doivent se coordonner pour se croiser. - Des situations peuvent se présenter dans lesquelles le nouveau chemin est long et pas très bon.

Tests et comparaisons entre les différentes stratégies

A* indépendant Vs Path-Slicing

Nous avons comparé les deux stratégies citées ci-dessus, en comparant l'équipe gagnante pendant 10 tours.

L'équipe 1 utilise la stratégie A* indépendante et l'équipe 2 utilise la stratégie de Path-slicing.

Tour	1	2	3	4	5	6	7	8	9	10
Equipe gagnante	2	2	1	1	2	1	1	1	1	1

Résultats: L'équipe 1 a gagné 7 fois et l'équipe 2 a gagné 3 fois. D'où on peut dire que la stratégie A* indépendante est plus performante que la stratégie du Path-Slicing.

Analyse des résultats: La stratégie de A* indépendante s'avère être plus performante. On peut expliquer cela par le fait qu'elle recalcule un chemin vers le but après chaque collision en prenant compte la position des agents. Néanmoins, la carte utilisée dans nos essais est petite et le nombre d'agents est moindre. Dans une carte plus grande, le recalcul d'un nouveau path vers le but peut s'avérer long et fastidieux.

A* Coopérative Vs A* indépendant

Nous avons comparé les deux stratégies citées ci-dessus, en comparant l'équipe gagnante pendant 10 tours.

L'équipe 1 utilise la stratégie A* coopérative et l'équipe 2 utilise la stratégie de A* indépendant.

Tour	1	2	3	4	5	6	7	8	9	10
Equipe gagnante	1	2	2	2	2	1	2	1	2	2

Résultats: L'équipe 1 a gagné 3 fois et l'équipe 2 a gagné 7 fois. D'où on peut dire que la stratégie A* indépendante est plus performante que la stratégie du Cooperative Path finding.

Analyse des résultats: La stratégie de A* indépendant s'avère être plus performante que la stratégie Coopérative path-finding. Cela n'est pas ce qu'on s'attendait à avoir, car le cooperative path-finding gère les collisions entre les agents au sein d'une même équipe. Néanmoins, il reste à gérer les collisions avec les ennemis ce qui n'est pas possible vu que nous n'avons pas connaissance de leurs intentions.

Conclusion

Dans ce projet, nous nous sommes familiarisés avec les différentes stratégies de path-finding. Nous avons implémenté et testé trois stratégies, ce qui nous a permis de les comparer entre elles.