Rapport de projet de LY Jean-Baptiste et RICHAUME Anthéa
Fichiers : Opportuniste.py Cooperative2D.py Cooperative2Dbis.py Cooperative3D.py projet.py

Cooperative Pathfinding

I. Introduction

Tout d'abord, le projet a pour objectif d'étudier des stratégies de coopérations entre agents afin de maximiser le score commun.
Pour cela, nous avons utilisé pygame afin de manipuler un système simple de jeux composés de personnages/joueurs qui seront nos agents, ayant pour but de ramasser leur fiole respective, tout en évitant les collisions avec les murs, mais aussi surtout entre eux.

II. Description des algorithmes

Ainsi, nous avons établi plusieurs stratégies. Toutes ces stratégies reposent sur l'algorithme A* utilisant la distance de Manhattan comme heuristique.

1) La stratégie opportuniste de portionnement de chemins (pathslicing) : les joueurs qui suivent leur chemin optimal. Il s'agit, lorsqu'un conflit est constaté, de recalculer une petite partie du chemin actuel en prenant en compte l'obstacle détecté (comme si c'était un mur par exemple). (Implémentée)

1bis) La stratégie dite « semi-opportuniste » : Une variante de la stratégie opportuniste. Lorsqu'un conflit est constaté, le joueur fait une pause au lieu de recalculer un autre chemin. (Non implémentée)

2) La stratégie coopérative de base : il s'agit d'identifier des chemins qui ne partagent aucune case. De manière évidente, ces chemins peuvent être exécutés en parallèle par les personnages en étant certain de ne pas entrer en collision.
Dès qu'un joueur détecte que personne ne sera sur son trajet, il fait son chemin optimal, qu'il a préalablement réservé.
Le problème réside dans le fait qu'un joueur devra attendre que tout le monde ait parcouru une bonne partie de leur chemin, avant de commencer à se déplacer. Ainsi, nous avons dans ce cas une perte de temps (dans notre cadre, une perte d'itérations). (Implémentée)

2bis) C'est une variante de la stratégie coopérative de base : Dans la stratégie précédente, si un joueur attend, il continuera à attendre jusqu'à que tous les autres joueurs aient fini de se déplacer sur une/des case(s) du chemin optimal du joueur qui attend. Dans cette variante, le joueur attend et calcule à chaque itération, si un nouveau chemin est disponible ou non, en sachant que lorsqu'un joueur se déplace, il annule la réservation spatiale de la case qu'il vient de quitter. Ainsi, à chaque itération, des cases se libèrent afin que l'autre joueur puisse partir plus tôt.
Le problème de cette stratégie, selon les cartes, il se peut qu'il ne prenne pas du tout un chemin optimal comme dans la stratégie précédente, voire qu'il prenne un chemin demandant beaucoup plus de temps/d'itérations. (Implémentée)

3) Stratégie coopérative avancée : Cette stratégie permet de combler le défaut de la stratégie précédente. En effet, l'idée générale est d'utiliser une structure partagée, une table de réservation spatio-temporelle, où les cases sont désormais un triplet (x,y,t), où x et y sont les coordonnées spatiales, et t la coordonnée temporelle.
Chaque joueur calculera tout d'abord son chemin optimal et réservera les cases de son chemin, au temps t où il bougera sur la dite case. Ainsi, cela permet à tous les joueurs de se déplacer en même

temps, au lieu d'attendre sur sa case initiale et de perdre davantage de temps. Il se peut donc que dans son chemin optimal obtenu, il puisse faire une pause sur une case à un temps t, afin de ne pas entrer en collision avec un autre joueur. Comme dans la stratégie précédente, les premiers joueurs qui réservent, auront leur chemin le plus optimal possible. (Implémentée)

4) Stratégie de coopération hiérarchique avec une nouvelle heuristique : « True Distance heuristic » : (Bonus)

Pour des cartes sous forme de grilles, la distance de Manhattan est souvent utilisée pour l'heuristique, car il est très simple de faire la somme des coordonnées (x,y), et c'est une bonne estimation de temps pour arriver à la destination. Cependant, si le plus court vrai chemin est plus long que la distance de Manhattan, alors cette dernière devient une estimation médiocre.

Cette nouvelle heuristique « True Distance Heuristic » représente la vraie distance, en prenant en compte tous les obstacles (mais pas les autres agents) du départ jusqu'à la destination. C'est donc une heuristique plus précise.
De plus, cette nouvelle heuristique est admissible puisqu'elle ne surestime jamais la distance à l'état but., et elle est consistante. En effet, dans l'algorithme A*, la somme du coût du chemin jusqu'au nœud n et le coût estimé de n jusqu'à un état but, reste non-décroissante sur les chemins depuis la racine.

En revanche, il y a plusieurs façons de l'implémenter. La première serait d'appliquer A* d'une façon totalement naïve à chaque nouvelle case explorée par A* utilisant la nouvelle heuristique.
En essayant cette implémentation, nous nous sommes aperçus que l'algorithme était beaucoup trop lent, pour chaque coup de chaque joueur. Ainsi dans ce cas, il vaudrait mieux utiliser l'heuristique de Manhattan.

Nous devons donc utilisé la méthode du « Backwards Search » afin de remédier à ce problème.

III. Expérimentations

Ce tableau représente la résolution de la carte en nombre d'itérations en fonction de la stratégie utilisée. Les cases grisées représentent les cas impossibles.

Cartes	Opportuniste	Coopérative 2D	Coopérative 3D	Coopérative 2D bis
pathfindingWorld_MultiPlayer1	29	49	32	66
pathfindingWorld_MultiPlayer2	21		23	43
pathfindingWorld_MultiPlayer3	21	28	21	21
10x10	20	23	20	26
Lab_15x15	22		25	32
map				
map2				
pathfindingWorld_MultiPlayer_6x6			10	
pathfindingWorld_MultiPlayer_bomberman	13	18	12	18
pathfindingWorld_MultiPlayer_impossible				
pathfindingWorld_MultiPlayer_incroyable	35	48	30	38
pathfindingWorld_MultiPlayer_labyrinthe2	13	13	13	13
pathfindingWorld_MultiPlayer_labyrinthe3	19		29	19
thirst	8		8	8

Ces résultats nous montrent que la stratégie Opportuniste a pour la plupart du temps, une meilleure solution que les autres stratégies. Son nombre d'itérations est l'un des plus faibles parmi toutes les stratégies présentées. En revanche, nous remarquons qu'il peut exister des cas où cette stratégie pourrait ne pas aboutir à une terminaison. Cela s'explique par le fait que les joueurs ne coopérent pas, ainsi il pourrait avoir des cas où les joueurs se bloquent entre eux même.

C'est donc pour cela qu'il faut créer une stratégie où les joueurs pourraient coopérer afin de privilégier certains autres joueurs et essayer d'éviter des cas bloquants.
Cependant, malgré l'esprit coopératif, la stratégie de coopération de base aboutit davantage à des configurations bloquantes. En effet, cela s'explique par le fait que l'algorithme ne permet le passage des joueurs qui attendent, uniquement dans le cas où personne n'est pas ou est déjà passé sur son passage. Or cela signifie donc que si un joueur J1 a par exemple sa fiole sur le chemin d'un autre joueur J2, et si J1 commence avant J2, alors J2 ne bougera jamais de sa position initiale car le joueur J1 attendra sur sa position une fois qu'il a ramassé sa fiole, et l'algorithme ne se terminera pas.

C'est donc pour cela que nous avons eu comme première idée naïve, d'implémenter une seconde stratégie de coopération. Cette stratégie de coopération bis, consiste à calculer à chaque itération, un nouveau chemin pour tous les joueurs en pause, sachant que les joueurs qui se déplacent, libèrent leur case qu'ils viennent de quitter. Ainsi cela permet de limiter les bouchons et les interblocages entre joueurs.
Cependant, certes cette stratégie bis permet d'être jouée sur plus de cas que ceux de la stratégie de base, mais la solution qu'elle retourne n'est pas forcément la solution optimale, pire, elle peut conduire à une solution très mauvaise, comme le montre le tableau des résultats, où le nombre d'itérations de cette stratégie bis peut être deux fois supérieur à celui de la stratégie opportuniste.

Il faut donc prendre en compte une troisième dimension qu'est le temps, afin de pouvoir optimiser au maximum le chemin de chaque joueur, d'où la stratégie de coopération avancée reposant sur 3 dimensions.
Cet algorithme a donc deux gros avantages : un nombre d'itérations nécessaire correct, proche de celui de la stratégie opportuniste qui privilégie le chemin optimal de chaque joueur, et le fait de pouvoir être jouée sans blocages sur des cartes plus complexes, comme celle de la carte 6x6, où l'algorithme de cette stratégie de coopération avancée est le seul à pouvoir se dérouler sans blocage. Tous les autres algorithmes conduisent à un échec.
Cela est dû au fait que les joueurs peuvent réserver leurs cases seulement à un temps t, ainsi cela leur permet de pouvoir se déplacer jusqu'à rencontrer un obstacle (mur ou autre agent) et aboutir à une coopération de la part du second joueur, qui va faire une pause sur une autre case qui est réservée à un temps t par le premier joueur. Une fois le premier joueur passé, le second joueur va pouvoir continuer son chemin jusqu'à sa fiole.

IV. Conclusion et ouvertures :

La stratégie opportuniste donne l'une des meilleures solutions parmi les stratégies proposées, voire la meilleure. Cependant elle n'est pas adaptée à toutes les cartes, surtout celles où la carte présente des passages uniques. Pour remédier à ce problème, il faut impérativement faire coopérer les agents afin qu'un agent ou des agents soit/ent privilégier par rapport à d'autres, mais aussi utiliser la dimension du temps afin de réserver une case seulement à un instant t, ce qui permet aux autres joueurs d'aller sur cette case aux autres temps t. La stratégie de coopération avancée est donc la meilleure car elle propose un chemin proche de celui du chemin optimal, tout en gérant une variété de cas où les agents pourraient se bloquer avec d'autres stratégies.

En revanche, il existe des limites à cette stratégie de coopération avancée. En effet, nous pouvons imaginer qu'une fois la fiole récupérée, le joueur coince le chemin d'un autre joueur. Dans ce cas là, il faudrait faire en sorte que le joueur se déplace même après sa fiole récupérée, afin de ne pas gêner les autres agents.
Nous pouvons aussi imaginer une stratégie qui donnerait à chaque joueur, une certaine priorité. Des agents pourraient être privilégiés sur d'autres agents. Par exemple, si un joueur ayant une plus haute priorité qu'un autre joueur, pourrait annuler la réservation de l'autre joueur pour sa propre réservation. L'autre joueur ayant perdu sa réservation, devra recalculer un autre chemin optimal en prenant en compte sa réservation annulée.

De plus, dans notre projet, les agents ne se déplacent que d'une case. Nous pourrions imaginer que certains agents pourraient se déplacer de plusieurs cases en même temps. Dans ce cas-là, il faudrait donc modifier notre table de réservation.