Cooperative Path-Finding

Hakim CHEKIROU & Rym KACI

25 mars 2019

Introduction

Les jeux de stratégies représentent un problème ardu pour les algorithmes de cheminements, du fait de la présence de plusieurs agents sur une carte dynamique et congestionnée.
L'algorithme A* peut amener un unique agent vers sa destination, mais la présence de plusieurs agents peut induire des collisions. Nous allons donc nous intéresser à des stratégies de cheminement coopératif entre les agents.
Ce projet explore trois stratégies pour résoudre efficacement ce problème. La première Stratégie appelée ici, path Splicing, consiste a utiliser A* pour contourner un obstacle. La deuxième stratégie fait passer en parallèle les agents dont les chemins ne se croisent pas. Quant à la troisième, elle consiste à rajouter une troisième dimension, le temps, à l'algorithme A*. Nous explorerons aussi différentes améliorations aux stratégies proposées. Nous évaluerons le comportement de ces stratégies face à des cartes illustrant des cas de figures significatif par rapport à notre problématique.

Stratégie Opportuniste : Path Splicing

Dans cette stratégie, tout les agents calculent leurs chemins vers leurs fioles respectives en utilisant A*. A chaque étape du parcours, le joueur vérifie la présence d'un autre joueur sur sa prochaine case. Dans ce cas, l'agent calcule un chemin pour contourner ce nouvel obstacle toujours en utilisant A*. Dans le cas où le détour est trop long (deux fois la longueur du chemin restant), le chemin recalculé ne sera plus vers la case suivante à la collision, mais plutôt directement vers la fiole.

Stratégie Coopérative de Base

Pour cette stratégie, les chemins sont calculés avec A*. On identifie les chemins ne partageant aucune case, ces derniers seront exécutés en parallèle.

Strategie Cooperative avancée

Une autre façon de traiter le problème est de planifier les chemins d'un point de vue spatio-temporel. Autrement dit, il n'y a collision que si deux joueurs sont sur la même case au même moment. Au niveau de l'implémentation, cela se fait avec une table de réservation : c'est une structure de données tridimensionnelle stockant les chemins des joueurs. Elle est transmise au A*, afin que le joueur évite les croisements lors du calcul de son chemin. Elle est mise a jour par chaque joueur. Nous utilisons deux heuristiques de distance.

Distance de Manhattan

La distance de manhattan est simplement le somme des distances en x et en y jusqu'à la destination. Le problème qui se pose en utilisant la distance de manhattan est que les obstacles ne sont pas considérés. De ce fait le joueur est parfois contraint d'explorer toute la carte (temps de calculs considérables) car la distance de manhattan est courte mais la distance réelle en prenant en compte les obstacles est plus importante.
Nous choisissons donc d'explorer une autre heuristique.

True Distance

Afin de résoudre les problèmes rencontrés en utilisant la distance de manhattan. Nous utilisons une heuristique pour laquelle A* explore en premier les chemins les plus pertinents (en prenant en compte les murs). On remarque qu'en utilisant A* inversé (depuis l'objectif jusqu'au joueur),le cout (g-$value$) de chaque nœud représente la distance réelle vers l'objectif. En pratique l'implémentation de cette idée est réalisée par l'exécution de deux recherches. Le cheminement est effectué en utilisant A* spatio-temporel. A chaque emplacement exploré l'heuristique true-distance est appelée : elle consiste en l'exécution du A* spatial depuis l'objectif vers l'emplacement. Nous conservons tous les nœuds explorés par la true distance afin de pouvoir reprendre la recherche de la true distance et éviter les calculs redondants.

Limitation de la profondeur de recherche

Dans le cas ou un grand nombre d'agents sont présents, l'algorithme A* spatio-temporel explore tant que la fiole n'est pas atteinte, or dans ce cas, la profondeur de recherche est très grande ce qui ralentit la stratégie. En limitant la profondeur de recherche à d nœud, un chemin partiel est retourné même si la fiole n'est pas atteinte, l'agent devra calculer a nouveau une fois arrivé a la fin du chemin partiel. Ceci nous permet de limiter le niveau de coopération entre les agents.

Analyse Comparative

Nous avons testé les différentes stratégies sur plusieurs cartes significatives. Les critères d'évaluations sont le nombre d'itérations pour que chaque joueur atteigne sa fiole et le temps de cheminement total (en comptant le calcul des détours pour la stratégie 1 et le calcul des chemins partiels pour la stratégie searchDepth).
Nous avons procédé aux différents tests sur une machine disposant d'un processeur intel i7. La profondeur de recherche pour la dernière stratégie à été mise a 12.

Dans cette partie, nous présentons le comportement des stratégies face à des aspects spécifiques de la problématique.

Grand nombre de joueurs

Nous utilisons la carte de la figure 1. Elle est intéressante du fait du nombre de joueurs qui est assez grand (9 joueurs).

Les résultats :

FIGURE 1 – carte

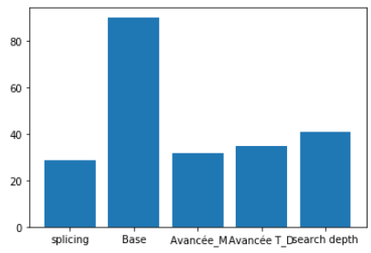

FIGURE 2 – itérations

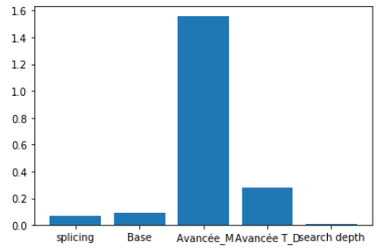

FIGURE 3 – temps de cheminement

Analyse des résultats

Par rapport au nombre d'itérations

Nous observons qu'en utilisant la stratégie coopérative de base, le nombre d'itérations nécessaire est très élevé par rapport aux autres algorithmes. Ceci s'explique par le fait que des croisement sont fréquents donc peu de joueurs s'exécutent en même temps. En limitant la profondeur de recherche, puisque les agents ne coopèrent qu'a un certain degré, ceci peut introduire des boucles qui, parfois se résolvent(comme ici). Dans certains cas où la profondeur est trop petite, la stratégie ne termine pas a cause de la présence de boucles. Les Stratégies avancées avec la distance de manhattan ou la *true-distance* sont équivalentes.

Par rapport au temps de calcul

La stratégie Path Splicing et la stratégie de base pressentent des temps de calculs peu élevés, ceci s'explique par le fait que pour le premier, A* spatial est utilisé et de petits détours sont calculés. Pour le deuxième, A* n'est utilisé qu'une seule fois par joueur. La stratégie présentant la plus grand temps de calcul est de loin, A* avancée en utilisant la distance de manhattan. La performance est grandement améliorée en changeant l'heuristique, car on économise sur le plan des nœuds explorés. En limitant la profondeur de recherche, le temps de calcul est très amoindrit.

Passages étroits

Nous utilisons la carte de la figure 4. Elle est intéressante du fait de la présence de passages sous forme de tunnels.

Les résultats :

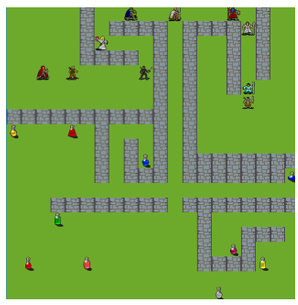

FIGURE 4 – carte

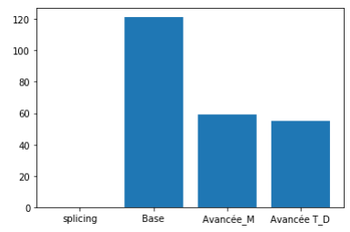

FIGURE 5 – itérations

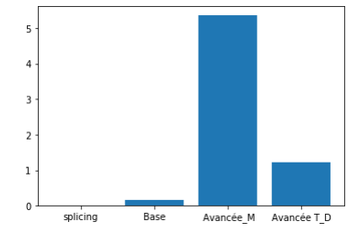

FIGURE 6 – temps de cheminement

Analyse des résultats

Par rapport au nombre d'itérations

il est évident que la stratégie Path Splicing ne termine pas car en calculant un détour pendant qu'un joueur est présent dans le tunnel, A* ne trouve pas de chemin vers la fiole. Nous remarquons que la stratégie avancée ne change pas sensiblement en changeant l'heuristique de distance. Quant à la stratégie coopérative de base, elle met le plus de temps car les joueurs sont obligés de passer un par un a cause des passages étroits.

Par rapport au temps de calculs

On remarque d'abord que la stratégie de base a le plus petit temps de calculs mais en contrepartie le nombre d'itérations est très élevé. La stratégie avancée en utilisant la stratégie de manhattan a un temps de calcul très élevé ($5s$) du fait du grand niveau de coopération demandé, l'algorithme explore un grand nombre de nœuds. En utilisant l'heuristique true Distance, le temps de calcul est amoindrit car le fait de reprendre l'exécution de A* économise un certains nombre d'opérations.

Distance réelle supérieure à la distance de manhattan

Nous utilisons la carte de la figure 7. Elle est intéressante pour notre étude car les fioles sont proches des joueurs en utilisant la distance de manhattan mais bien plus éloignés en utilisant la distance réelle.

Les résultats :

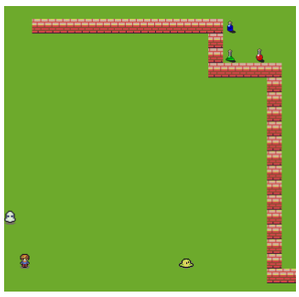

FIGURE 7 – carte

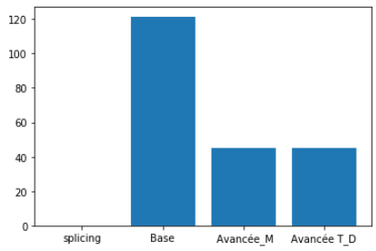

FIGURE 8 – itérations

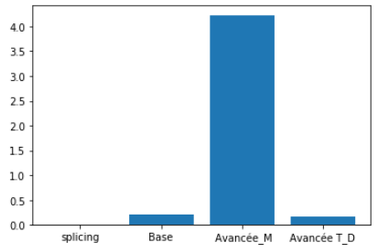

FIGURE 9 – temps de cheminement

Analyse des résultats

Par rapport au nombre d'itérations

En utilisant la distance de manhattan et la true-distance, les deux algorithmes ont le même nombre d'itérations.

Par rapport au temps de calculs :

En utilisant la distance de manhattan le temps de calcul est très élevé car le A* favorise les mauvais nœuds durant l'exploration, donc une grande partie de la carte est explorée avant de trouver la fiole. En changeant l'heuristique, le temps de calcul est considérablement amélioré car les nœuds qui sont réellement plus proches sont explorés en premier.

Conclusion

A travers les différentes stratégies étudiées durant ce projet, nous avons pu caractériser le problème de cheminement sous différents points de vues. Nous pourrions envisager plusieurs types d'autres approches à ce problème comme dans le cas ou des agents vont plus vite que d'autres ou si des agents sont plus prioritaire que d'autres.