Projet : Cooperative Path-Finding

Nemanja Kostadinovic, Rebecca Leygonie

I. Présentation du problème

A. Description de l'environnement

Le problème (Cooperative Path-Finding), noté *CPF*, est décrit par la donnée des éléments suivants :

- Un environnement décrit par une matrice binaire M de taille $n \times m$. La variable binaire M_{ij}indique la présence d'un obstacle aux coordonnées (i, j) pour $1 \leq i \leq n$, $1 \leq j \leq m$.
- Un ensemble de N joueurs décrits par les coordonnées $\{(x_k^0, y_k^0) | 1 \leq k \leq N\}$ indiquant leurs positions de départ.
- Un ensemble de N objectifs décrits par les coordonnées $\{(x_k^T, y_k^T) | 1 \leq k \leq N\}$ indiquant les positions à atteindre pour chaque joueur.

B. Description des actions

Tous les joueurs sont positionnés à leurs points de départ à $t = 0$. À l'étape t, chaque joueur peut prendre l'une des cinq actions $\{(0, 0), (1, 0), (0, 1), (-1, 0), (0, -1)\}$ pour choisir d'aller dans l'une des cases adjacentes ou de ne rien faire. Cependant, une action n'est valide que si elle respecte les règles suivantes :

- L'action ne peut mener ni vers une case qui sort du cadre de l'environnement ni vers un obstacle
- Une case ne peut contenir plus d'un joueur à un instant t
- Deux joueurs ne peuvent pas interchanger leurs positions entre t et $t + 1$

C. Objectif du problème

L'objectif du problème est de trouver des chemins valables pour chaque joueur tout en miminimsant le temps total nécessaire pour que chaque joueur atteigne son objectif. Ce problème est difficile car le nombre de chemins possibles pour chaque joueur est exponentiel en la taille de l'environnement $n \times m$. Dans la suite de ce projet, on va explorer différentes heuristiques pour résoudre ce problème. Pour cela, on va d'abord utiliser l'algorithme A^* pour résoudre le problème dans le cas $N = 1$, *i.e* un seul joueur, avant de passer au cas général.

II. Cas particulier $N = 1$

A. Algorithme A*

Pour ce cas particulier avec un seul joueur, on ne se soucie pas des collisions et on peut donc effectuer la recherche du plus court chemin sans l'action stationnaire $a = (0, 0)$.
L'algorithme utilisé est celui de A^* qui consiste à utiliser une fonction d'évaluation f pour développer le chemin du point de départ au point d'arrivée. Cette fonction d'évaluation f est la somme d'un coût réel g et d'une heuristique d'estimation h. Étant donnée une position, g est calculée récursivement en utilisant la valeur de son parent. On stocke des positions dans une file prioritaire (par rapport à la fonction d'évaluation).

B. Fonction d'évaluation

- **Coût réel** g : Ce coût réel est calculé récursivement en utilisant la valeur du parent d'un noeud. Étant donné un certain chemin, le coût réel d'une position est la longueur du chemin partiel entre le point de départ et cette position.
- **Coût heuristique** h: Ce coût heuristique estime la distance séparant la position actuelle du point d'arrivée. Diverses heuristiques sont possibles :
  - **Euclidienne**

    $$\text{dist}[(x_1, y_1), (x_2, y_2)] = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

  - **Maximum**

    $$\text{dist}[(x_1, y_1), (x_2, y_2)] = \max(|x_1 - x_2|, |y_1 - y_2|)$$

  - **Manhattan**

    $$\text{dist}[(x_1, y_1), (x_2, y_2)] = |x_1 - x_2| + |y_1 - y_2|$$

III. Cas général $N \in \mathbb{N}^*$

Dans cette section, l'algorithme A^* est utilisé pour calculer tout chemin, partiel ou total.

A. Les stratégies

1) Stratégie opportuniste de portionnement de chemins: Cette stratégie consiste à suivre un trajet dont une partie est recalculée à chaque fois qu'un obstacle est rencontré. Tout joueur détermine le chemin à prendre sans prendre en compte les chemins des autres joueurs, *i.e* sans coopération. Ces chemins sont exécutés séquentiellement et en parallèle. Toutefois, à chaque étape t, si une collision est détectée à $t+1$ entre deux joueurs k_1 et k_2, l'un des deux devra modifier son trajet. Supposons que k_2 à la position (x_t, y_t) doit modifier son trajet. Le chemin initial était de la forme :

$$(x_t, y_t) \to (x_{t+1}, y_{t+1}) \to (x_{t+2}, y_{t+2})$$

Le joueur k_2 recalcule un autre chemin en prenant en compte l'obstacle rencontré pour aller de (x_t, y_t) à (x_{t+2}, y_{t+2}).

2) Stratégie coopérative de base: Cette stratégie consiste à suivre un trajet qui ne croise le chemin d'aucun joueur à aucun instant. Ces chemins peuvent donc être exécutés en parallèle sans risque de collision. Un ordre de passage est déterminé pour chaque joueur. Ce dernier calcule son chemin, en considérant les chemins pris par les précédents comme un obstacle à éviter.

3) Stratégie coopérative avancée: Cette stratégie consiste à suivre un trajet qui n'entre pas en collision avec celui d'un autre joueur à chaque instant. Elle diffère de la stratégie précédente du fait que deux joueurs peuvent occuper la même position à des instants différents. Toutefois, il faut aussi prendre le deuxième type de collision. Concrètement, l'environnement est maintenant décrit par une matrice binaire à 3 dimensions pour inclure la dimension temporelle. La variable binaire M_{ijt} indique la présence d'un obstacle ou qu'un joueur occupe cette position à l'instant t. Un ordre de passage est déterminé pour chaque joueur. Ce dernier effectue une recherche de son chemin et modifie la matrice M_{ijt} pour inclure ses positions.

B. Procédures de sélection

Pour que les stratégies précédentes aboutissent à une solution, il faut déterminer l'ordre dans lequel les joueurs effectuent leurs actions : à chaque étape t pour la première stratégie ou à l'étape initiale pour les deux stratégies suivantes . Un ordre de passage de la forme $k_1, k_2, k_3, \cdots, k_N$ implique que le joueur k_i recalcule si nécessairement son chemin en prenant en compte les obstacles des joueurs précédents $k_1, \cdots, k_{i-1}$.
On détermine alors l'ordre de passage selon l'une des procédures de sélection suivantes :

- **Aléatoire** : On choisit aléatoirement et de manière uniforme l'odre de passage des joueurs.
- **Plus proche** : Dans cette procédure, les joueurs sont ordonnées par la distance les séparant à leur objectif. On favorise le joueur le plus proche pour qu'il arrive le plus rapidement possible et ainsi ne plus imposer des obstacles pour les autres joueurs.
- **Plus loin** : Dans cette procédure, les joueurs sont ordonnées par l'inverse de la distance les séparant à leur objectif. On favorise le joueur le plus loin en estimant que la modification du trajet d'un joueur plus proche engendre un moindre coût.

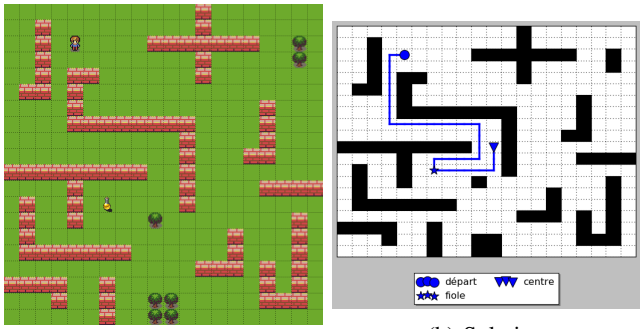
IV. Résultats expérimentaux

A. Mesures de performances

- **Temps total** T**:** Le temps total pour que chaque joueur atteigne sa destination
- **Itérations** n_{iter}**:** Nombre d'itérations total dans la boucle de l'algorithme A^*

B. Résultats

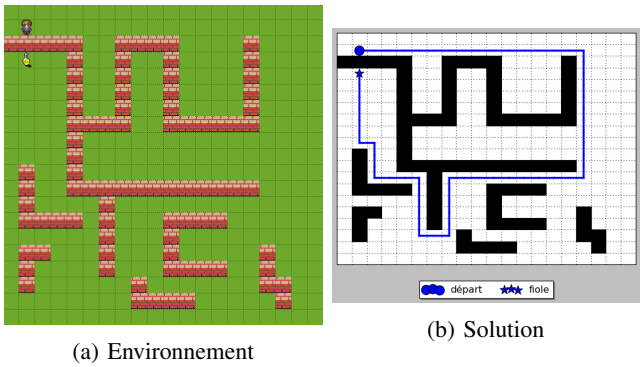
1) Cas particulier $N = 1$*:*
Exemple 1 : Dans cet exemple, le joueur doit récupérer la fiole et revenir au centre. La solution trouvée : $T = 26$, $n_{iter} = 209$



(a) Environnement (b) Solution

Fig. 1: Exemple 1

Exemple 2 : Dans cet exemple, le joueur doit seulement récupérer la fiole. L'exploration est nécessaire car le coût heuristique donne une mauvaise estimation à premier abord de la distance nécessaire pour arriver à la fiole. La solution trouvée : $T = 60$, $n_{iter} = 312$

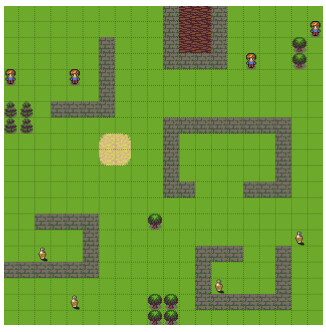


(a) Environnement (b) Solution

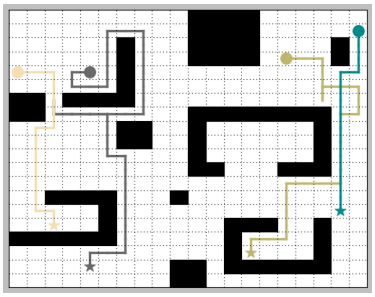
Fig. 2: Exemple 2

2) Cas général $N \in \mathbb{N}^*$*:*
Dans ces exemples, la borne inférieure $T_{\min}$ dénote le coût max des chemins optimaux calculés de manière individuelle. Pour la solution, on prend celle fournie par la meilleure procédure de sélection.
Exemple 1 :

- *Carte utilisée :* $T_{\min} = 22$

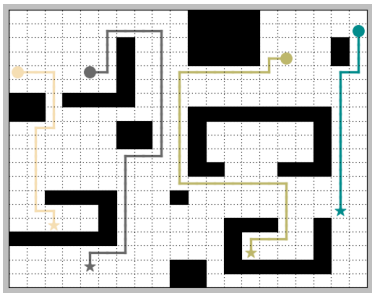


- *Stratégie opportuniste de portionnement de chemins :* $T = 41, n_{iter} = 841$



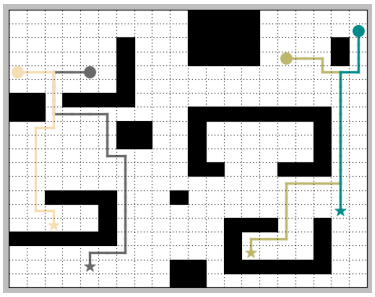
On remarque que le joueur 2, une fois entré en collision avec le joueur 1, a du faire un grand détour pour reprendre son chemin initial.

- *Stratégie coopérative de base :* $T = 28, n_{iter} = 622$



On remarque que les joueurs 2 et 3 ont dû complètement modifier leurs chemin individuels pour ne pas entrer en collision avec leurs voisins respectifs.

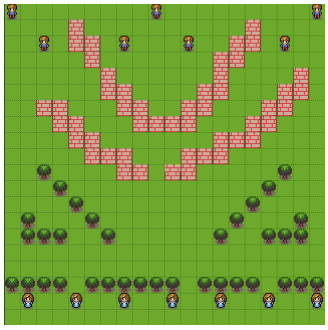
- *Stratégie coopérative avancée :* $T = 23, n_{iter} = 6198$



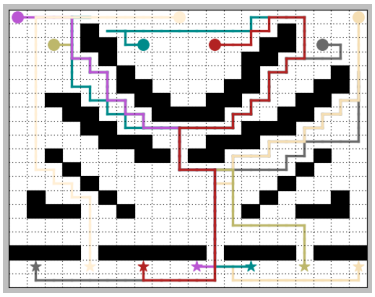
Cette stratégie fournit une solution optimale mais avec un très grand nombre d'appels à l'algorithme A^*.

Exemple 2 :

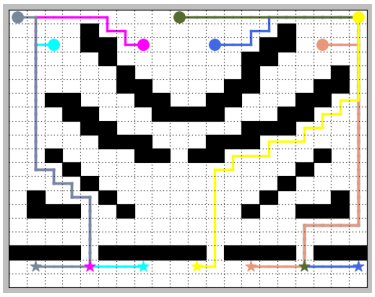
- *Carte utilisée :* $T_{\min} = 34$



- *Stratégie opportuniste de portionnement de chemins :* $T = 75, n_{iter} = 2518$



- *Stratégie coopérative de base :* Cette stratégie ne fournit aucune solution car il n'est pas possible de trouver des chemins qui ne se croisent pas.
- *Stratégie coopérative avancée :* $T = 35, n_{iter} = 28499$



Cette stratégie fournit une solution optimale mais avec un très grand nombre d'appels à l'algorithme A^*.