Université de Bordeaux - Master 2

— Algorithmique appliquée —

Projet stratégies de défense à la RoboCup

Adrien Maurin, Florian Simba

Encadré par Ludovic Hofer

2 octobre 2020

1 Définition formelle du problème

Dans cette section, nous allons introduire les différentes variables du problème et définir formellement les objets utilisés en entrée et en sortie.

1.1 Input

La constante *posstep* Il s'agit de la valeur qui indique l'écart minimal entre deux points de $\mathbb{R}^2$. Cela permet de discrétiser $\mathbb{R}^2$. On a alors $posstep \in \mathbb{R}_+^*$.

La constante *thetastep* Il s'agit de la valeur minimale de l'angle que doivent former deux droites pour ne pas être confondus. Cela permet de discrétiser les angles (compris dans $]-\pi;\pi]$).

La constante *radius* Il s'agit du rayon des cercles qui modélisent les robots. On a alors $radius \in \mathbb{R}_+^*$.

Limites du terrain Le terrain de jeu est un rectangle délimité par deux points (x_{min}, y_{min}) et (x_{max}, y_{max}). Il s'agit donc de l'ensemble de points F tels que :

$$F = \{(x,y) \in \mathbb{R}^2 \ / \ x \leqslant x_{max} \wedge x \geqslant x_{min} \wedge y \leqslant y_{max} \wedge y \geqslant y_{min}\}.$$

Le terrain Le terrain est modélisé par l'ensemble de points G tel que :

$$G = \{(x,y) \in F \ / \ fmod(x, posstep) = 0 \wedge fmod(y, posstep) = 0\}.$$

Autrement dit, il s'agit de la grille de points contenant l'origine du plan et dont la taille des cases est de *posstep*.

Les robots Les robots sont modélisés par l'ensemble de points suivants avec p_0 le point central du robot :

$$R_{p_0} = \{p \in \mathbb{R}^2 \ / \ ||p_0 - p||_2 \leqslant radius\}.$$

Les positions des adversaires Il s'agit d'un ensemble de points p_i, avec $p_i \in G$. Cet ensemble A est défini par :

$$A = \{\forall i, j \in [\![1, |A|]\!], i \neq j, \neg collision(a_i, a_j)\}.$$

La fonction *collision* peut être exprimée de la manière suivante :

$$collision\colon F^2 \to \{0,1\}$$
$$d_i, d_j \mapsto collision(d_i, d_j) = \begin{cases} 0 & \text{si } ||d_i - d_j||_2 < 2radius \\ 1 & \text{sinon.} \end{cases}$$

Le(s) goal(s) Il s'agit d'un ensemble de segments délimités par deux points. Un dernier paramètre permet d'indiquer le sens du goal, i.e. par quel « côté » du segment la droite de tir doit passer. On notera ce vecteur g. Le segment délimitant le goal sera noté T.

Les tirs d'un attaquant Il s'agit d'un ensemble T_i de demi-droites ayant pour point d'origine celui d'un attaquant i tel que :

$$T_i = \{t_i \subset \mathbb{R}^2 \ / \ p_i \in t_i \wedge fmod(angle(t_i), thetastep) = 0\}.$$

La fonction *angle* retourne l'angle formé par la demi-droite donnée et l'axe des abscisses.

Les tirs cadrés Pour qu'un tir soit cadré, il faut que le tir d'un attaquant traverse le segment du goal dans le bon sens (le produit scalaire entre le vecteur du goal et celui de la droite de tir est strictement négatif). Cette représentation montre les tirs qui atteignent le but sans prendre en compte les défenseurs. L'ensemble B_c des tirs cadré est exprimé ainsi :

$$B_c = \{t_{ij} \ / \ \forall i \in [\![1, |A|]\!], \forall j \in [\![1, |T_i|]\!], t_{ij} \cap T \neq \emptyset \wedge (t_{ij}|g) < 0\}$$

Les tirs non-cadrés Pour qu'un tir soit non-cadré, il faut que le tir d'un attaquant ne traverse pas le segment du goal, ou dans le mauvais sens sinon. Il s'agit du complémentaire de l'ensemble précédent. L'ensemble B_{nc} est exprimé ainsi :

$$B_{nc} = \{t_{ij} \ / \ \forall i \in [\![1, |A|]\!], \forall j \in [\![1, |T_i|]\!], t_{ij} \cap T = \emptyset \vee (t_{ij}|g) \geqslant 0\}$$

Les tirs interceptés Pour qu'un tir soit intercepté, il doit dans un premier temps être cadré et rencontrer un obstacle sur le chemin (robot allié ou adverse). L'ensemble B_i est exprimé ainsi :

$$B_i = \{t_i \ / \ \forall i \in [\![1, |B_c|]\!], \exists! j \in [\![1, |A|]\!], R_{a_j} \wedge t_i \neq \emptyset \wedge t_i \cap D = \emptyset\}$$

Les tirs réussis Pour qu'un but soit marqué, il faut qu'une demi-droite de tir d'un attaquant traverse le segment du goal dans le bon sens sans rencontrer de robots sur le trajet. L'ensemble B est exprimé ainsi :

$$B = \{t_{ij} \ / \ \forall i \in [\![1, |A|]\!], \forall j \in [\![1, |T_i|]\!], t_{ij} \in B_c \wedge t_{ij} \notin B_i\}$$

1.2 Output

Les positions des défenseurs Il s'agit d'une liste de points p_i dans D tels que :

$$D = \{\forall i, j \in [\![1, |D|]\!], i \neq j, \neg collision(d_i, d_j) \wedge B = \emptyset\}.$$

Le cardinal de l'ensemble D est par ailleurs borné par le nombre de joueurs. On a ainsi $|D| \leqslant 8$.

2 Modélisation du problème

On va modéliser le problème que l'on vient de définir de manière formelle comme un problème de graphe.

2.1 Construction du graphe

Les nœuds représentant les tirs Un tir est est une combinaison entre la position d'un attaquant et un angle (Att, θ). On ne considère que les tirs cadré. Pour chaque tir cadré, on lui associe un nœud sur notre graphe. On note l'ensemble de ces sommets V_t.

Les nœuds représentant les position des défenseurs La position d'un défenseur est modélisée par un point du terrain. On ne considère que les positions disponibles, i.e. l'ensemble des points de $G \backslash A$. Pour chaque position de défenseur, on lui associe un nœud sur notre graphe. On note l'ensemble de ces sommets V_d.

L'ensemble des sommets du graphe est alors égale à $V = V_t \cup V_d$.

Les interceptions Une interception entre un tir et un défenseur est modélisée comme une arête reliant le tir d'un attaquant à une position de défenseur. On note cet ensemble E. Le graphe que l'on obtient est alors biparti entre l'ensemble de nœuds V_d et V_t.

2.2 Simplification du graphe

On simplifie le graphe que l'on vient d'obtenir en retirant les sommets isolés (correspondant à des positions de défenseurs n'interceptant aucun tir).

2.3 Résolution du problème

Résoudre le problème initial, i.e. trouver le nombre minimal de défenseurs permettant de défendre le but se ramène dans notre modélisation à résoudre le problème de graphe suivant :

Problème Il faut trouver le plus petit sous ensemble de V_t tel que cet ensemble domine V_d.

2.4 Exemple d'application

La figure 2.4 montre un exemple de la modélisation du problème. Ici, il existe 5 trajectoires de tirs permettant aux adversaires de marquer. On dispose de 5 emplacements pour placer nos défenseurs. On constate que placer un défenseur à la position Def_1 et un défenseur à la position Def_4 permet de bloquer la totalité des tirs (les arêtes en bleu). L'ensemble des tirs est alors dominé par notre groupe de défenseurs. On remarque dans cet exemple qu'on ne peut pas faire mieux que 2 défenseurs.

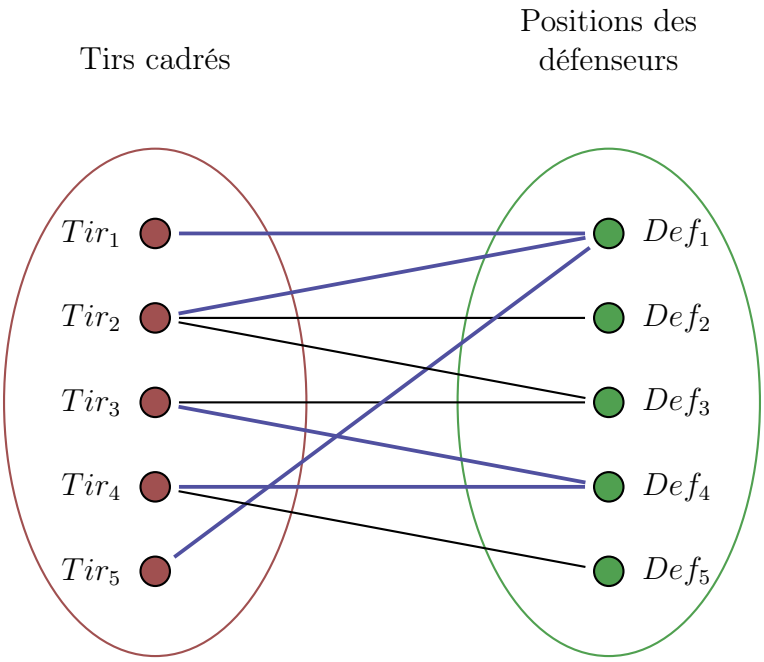

FIGURE 1 – Exemple du problème où les 5 tirs sont bloqués par deux défenseurs 1 et 4.

3 Pistes de résolution du problème

Cette section présente la méthode de calcul pour savoir si une arête se situe entre deux sommets du graphe ainsi que différents algorithmes de résolution à notre problème.

3.1 Interception de tir

Un tir étant défini à l'aide d'un angle θ et d'un point (x_0, y_0). Pour déterminer si ce dernier est intercepté par un point, on peut dans un premier retrouver l'équation de la droite :

$$\begin{aligned}
&y = y_0 + (x - x_0)\tan\theta && \text{avec } x \geqslant x_0 \text{ si } \theta \in]-\frac{\pi}{2}; \frac{\pi}{2}[\\
&y = y_0 + (x - x_0)\tan\theta && \text{avec } x \leqslant x_0 \text{ si } \theta \in]-\pi; \frac{\pi}{2}[\cup]\frac{\pi}{2}; \pi] \\
&x = x_0 && \text{avec } y \geqslant 0 \text{ si } \theta = \frac{\pi}{2} \\
&x = x_0 && \text{avec } y \leqslant 0 \text{ si } \theta = -\frac{\pi}{2}
\end{aligned}$$

Une fois qu'on a l'équation de la demi-droite modélisant le tir, pour vérifier s'il y a une interception d'un tir à une position d'un défenseur on calcule la distance minimale entre la demi-droite de tir et le point position du défenseur. Cette distance est calculée à partir de la formule suivante :

$$d = \frac{Ax_0 + By_0 + C}{\sqrt{A^2 + B^2}}$$

avec $Ax + By + C = 0$ l'équation de la droite et (x_0, y_0) la position du défenseur.
Si cette distance est inférieure à R (rayon du robot), alors le tir est intercepté.

3.2 Utilisation d'un algorithme exhaustif

Pour résoudre le problème d'ensemble dominant dans notre graphe, on peut dans un premier temps utiliser un algortihme exhaustif, qui, pour toute configuration d'ensemble de sommets dans V_d, vérifie s'il est dominant, et le conserve si c'est le cas. Ensuite, l'algorithme renvoie un ensemble dominant de taille minimale. L'avantage est qu'on aura la meilleure solution possible.

Complexité La complexité de cet algorithme est en $\mathcal{O}(2^{|V_d|})$ [1].

Le désaventage de cet algorithme est le coût en temps qui, si le nombre de sommets de positions des défenseurs est grand, peut être problématique pour obtenir une solution

1. On code chaque sommet des défenseurs par 0 s'il ne fait pas parti de l'ensemble dominant et 1 sinon. Tester ensuite toutes les combinaisons donne cette complexité exponentielle.

rapidement. L'algorithme évoqué dans la section suivante permettra de réduire les temps de calcul.

3.3 Utilisation d'un algorithme Branch and Reduce

Une piste de résolution possible est l'utilisation de l'algorithme Branch and Reduce, présenté dans l'article « A Branch-and-Reduce Algorithm for Finding a Minimum Independent Dominating Set in Graphs » [2] par Serge GASPERS et Mathieu LIEDLOFF.

Dans cet article, l'algorithme travaille sur des graphes marqués, i.e. des graphes définis par le triplet : $G = (M, F, E)$ ou M est un ensemble de sommets marqués (les tirs), F un ensemble de sommets libres (les positions des défenseurs) et E l'ensemble des arêtes. L'algorithme calcule le sous-ensemble minimal (s'il existe) de F qui domine le graphe $G' = (M \cup F, E)$.

Complexité La complexité de cet algorithme est égàle à $\mathcal{O}(1.3575^{|V_d|})$ ce qui est nettement mieux que la complexité de l'algorithme exhaustif.

4 Compatibilité de la modélisation choisie avec les extensions

Cette section les avantages et inconvénients de notre modélisation par rapport aux extensions à développer.

Plusieurs buts S'il y a plusieurs buts, le nombre de trajectoires ne fera que s'accroître. Si V_{ti} correspond aux tirs cadrés dans le goal i et qu'il y a n buts, alors $V_t = V_{t1} \cap \cdots \cap V_{tn}$.

Position initiale des joueurs Dans le cas ou la position initiale des défenseurs est connue, on peut introduire un paramètre *movestep* qui symbolise la distance maximale qu'un défenseur peut parcourir pour intercépter le tir. Les changements dans notre modèle sont les suivants :

- Les positions des défenseurs ne sont plus tous les points de $G \backslash A$ mais tous les points du terrains à portée de déplacement des défenseurs.
- Chaque défenseur doit pouvoir être placé dans sa zone d'action.

Distance minimale entre les robots Dans le cas ou les robots nécéssitent de respecter une distance minimale, nous pouvons introduire un paramètre D modélisant la distance minimale de deux robots.

Pour prendre en compte cette contrainte, il faut légèrement changer notre modélisation. Á partir du sous-graphe G' comportant les sommets V_d, on place une arête

2. `https://www.univ-orleans.fr/lifo/Members/Mathieu.Liedloff/publications/GL06.pdf`

entre deux sommets si les positions des défenseurs sont à une distance inférieure à D. La solution de positionnement des défenseurs doit être un stable de G'.

Gardien Pour pouvoir prendre en compte cette extension dans notre modèle, il faut prendre en compte les modifications suivantes :

— L'ensemble des sommets de position des défenseurs situés dans le goal est égal à V_g. L'ensemble des sommets V du graphe est alors égal à $V_t \cup V_g \cup V_d$.

— On place des arêtes entre des sommets de V_t et V_d et entre des sommets de V_t et V_g s'il y a des interceptions de tir.

La solution de positionnement des défenseurs ne doit autoriser le placement que d'un unique défenseur dans le goal (un seul sommet dans V_g).

Trajectoires courbées Pour pouvoir gérer des trajectoires courbées, il faut introduire un nouveau paramètre *alpha* qui est la courbure maximale de la courbe. Lors du calcul des trajectoires des tirs cadrés, il ne faudra pas seulement prendre en compte les tirs rectilignes mais aussi les tirs courbées (discrétisés par *thetastep*). Cette extension ne nécessite pas vraiment de changements dans notre modélision à part à la définition des tirs. Un tir est maintenant modélisé par un point, un angle et une courbure $((x, y), \theta, \alpha)$. Cela aura tendance à angmenter le nombre de sommets modélisant des tirs seulement, mais la résolution de l'algorithme reste similaire.