Multirobot Wars: rapport

Kim-Anh Laura Nguyen, Fatemeh Hamissi

April 7, 2018

1.1 Introduction

Ce projet consiste en l'implémentation de stratégies permettant à une équipe de robots de gagner le maximum de cases dans une arène.
Pour ce faire, nous considérons dans notre équipe un robot "explorateur" qui parcourt le plus de cases possibles, et des robots "suiveurs" qui suivent ou bloquent des adversaires.
Nous définissons donc deux stratégies, la première étant composée uniquement de comportements; la deuxième introduisant l'évolution artificielle.

1.2 Première stratégie: Architecture de subsomption

Nous définissons une architecture de subsomption dans le fichier *strategy1.py*.
L'algorithme définissant la prochaine action du robot à chaque pas de temps est le suivant:

```
Si le robot est celui devant explorer alors
    Si un de ses senseurs détecte un robot ou un obstacle alors
        rotation ← angle opposé à celui de ce senseur
    Sinon
        rotation ← 0
Sinon
    Si un de ses senseurs détecte un robot alors
        Si c'est un robot de l'équipe adverse alors
            rotation ← angle de ce senseur
        Sinon
            rotation ← angle opposé à celui de ce senseur
    Sinon
        rotation ← 0
translation ← 1
```

1.3 Deuxième stratégie: Évolution artificielle

Dans cette partie, nous considérons l'approche de la robotique évolutionniste pour définir un agent explorant le maximum de cases possibles.
Pour construire le réseau de neurones correspondant, nous procédons tout d'abord à une optimisation sur les contrôleurs du robot afin d'extraire les meilleurs paramètres . Nous utilisons une fonction fitness qui évalue la distance parcourue à chaque pas de temps, en pénalisant les commandes de rotation, ainsi qu'un génome composé de 14 paramètres (12 associés aux capteurs, 2 servant pour les neurones de biais).
L'algorithme d'optimisation utilisé est l'algorithme (1+1)-ES avec la règle des 1/5, pour accélérer la convergence lorsqu'il y a progression et la freiner dans le cas contraire. Nous fixons le nombre d'évaluations maximale à 200 et σ à 10^{-7}. Le pseudocode est donné ci-dessous.

Initialisation des paramètres du génome à des valeurs aléatoires dans [-10,+10]
Pour i allant de 0 à maxEvaluations
v ← Tirage aléatoire selon une loi gaussienne
Pour chaque paramètre du génome
paramètre ← paramètre + $\sigma * v$
Si la fitness du génome obtenu est meilleure que l'ancienne alors
Mettre à jour le génome avec ces paramètres
$sigma \leftarrow sigma * 2$
Sinon
$sigma \leftarrow sigma * 2^{-1/4}$
i ← i + 1

À la sortie de cet algorithme, nous obtenons les meilleurs paramètres pour définir un robot "explorateur". Ils sont recopiés tels quels dans l'algorithme définissant la prochaine action du robot (dans *strategy2.py*), et utilisés en tant que poids des neurones :

Si le robot est celui devant explorer alors
P ← tableau des poids récupérés grâce à l'étape d'optimisation effectué au préalable

S ← tableau des valeurs des senseurs
translation ← tangente hyperbolique de $P_7 + \sum_{i=1}^{6} P_i * S_i$
rotation ← tangente hyperbolique de $P_{14} + \sum_{i=8}^{13} P_i * S_i$
Sinon
mêmes étapes que dans la première stratégie