Rapport : Methodes et outils de l'ia et la ro

projet : multirobot wars

Becirspahic Lucas

Université Pierre et Marie Curie - 18 avril 2017

1 Introduction

L'objectif de ce projet est de réaliser des intelligences artificielles pour jouer a un jeu de capture de case par equipes. Chaque équipes est composées de 4 robots. Il faut maximiser le nombres de cases possédées à la fin de la partie. Pour ce faire, on peut définir des algorithmes à la main ou des algorithmes évolutionistes.

2 Algorithmes génétiques

On cherche à trouver les poids associés aux différents senseurs de notre robot dans des neurones. On a deux neurones, un pour avoir la valeur de transltation et l'autre pour celle de la rotation. Chaque senseur à une valeur d'activation comprise entre 0 et 1 et est l'entrée de notre neurone. On cherche à déterminer les valeurs optimales des poids. Pour ce faire, on définit une fonction $fitness$, qui permet de voir à quel point l'objectif de notre robot est respecté pour des paramètres donnés. On cherche donc a trouver les valeurs optimales de notre généome qui permet de maximiser cette fonction.
La permière aproche, la plus simple consiste à tirer aléatoirement les parametres et garder le champion : celui avec la meilleur fitness. Néenmoins cette methode prend beaucoup de temps à converger et de ce fait donne souvent un optimum local. Par exemple dans le cas d'un éviteur d'obstacle, l'optimum local est un robot qui tourne en rond .
Une approche plus intelligente est de faire une mutation gaussienne sur notre champion afin de faire varier les parametres tout en restant proche du génome du champion.
On introduit un σ pour faire varier l'amplitude de la mutation. Néenmoins avec un σ fixe, on s'approche de la valeur optimal mais trop brutalement , les mutations changent trop notre génome. C'est pourquoi on fait bouger le σ de la manière suivante :
Si la fitness trouvée est meilleur que la précédante, on augmente σ pour qu'il soit plus grand : $\sigma = 2\sigma$
Sinon on veut avoir des variations plus fines, donc on réduit σ : $\sigma = 2^{-1/4}\sigma$
On peut développer des comportements variés en fonction de la fonction fitness choisit par exemples :
– **Eviteur d'obstacle** $fitness = vt * (1 - vr) * MinSenseurValue$
– **Colle mur** $fitness = vt * (1 - vr) * (1 - MinSenseurValue)$
– **Traqueur** $fitness = vt * (1 - vr) * MinSenseurEnemis$
remarque : seul l'éviteur d'obstacle à été implémenté.

3 Mise en oeuvre

Dans le cadre du projet, nous avons utilisé une architechture de subsomption, constituée de stratégies à la main et d'algorithmes génétiques.
Nous avons définit un objet $Action$, qui est constitué d'une action , d'une condition et d'une durée.
La condition permet de savoir quand il faut utiliser l'action et la durée permet de faire une action de manière continue par exemple pour reculer sufisament avant de re-avancer.
Parmis les différents comportements, nous avons implémenté un random quand le robot est coincé, un traqueur qui suit un robot adversaire, une action d'immobilisation si notre robot detecte qu'il est suivit.
Afin de faciliter la programation, nous avons implémenté un $GameDecorator$ qui contient des attributs plus facile à manipuler comme information de jeu tel que les senseurs avant, ou l'adversaire le plus proche détecté. Celui ci est recalculer lors de chaque $step$ du jeu. L'architecture de subsomption choisit une action dans la liste d'action et l'exécute si c'est possible lors de chaque étape de jeu.

4 Ouverture

Une stratégie envisagable qui n'a pas été implémenté, aurait été de définir la fitness comme le score final d'une partie.
Puis d'utiliser un algorithme génétique pour trouver les paramètres adaptés.
En revanche, il faut faire attention car cela entraine le robot a jouer contre une équipe advserse spécifique et non pas toutes les équipes possibles.