Université de Caen, Normandie

Intelligence artificielle distribuée

Rapport

# SoccerBots

*Auteurs:*
Lounis Bibi
Boris El Gareh

*Enseignant:*
Grégory Bonnet

24 Mars 2019

# 1 Introduction

Cette année, dans le cadre de l'unité d'enseignement *Intelligence artificielle distribuée*, nous avons développé un système multi-agents simulant une équipe de football à cinq grâce à l'API *SoccerBots*.
L'objectif de ce projet est de produire un comportement non trivial original, en implémentant une architecture multi-agents.
Par ailleurs, des résultats de test contre d'autres équipes seront présentées dans ce rapport.
Enfin, l'équipe combattra dans un tournoi de promotion, cette année.

# 2 *SoccerBots*

L'API *SoccerBots* fait parti d'un programme plus vaste appelé *TeamBots*. *TeamBots* est une collection, basée sur le langage de progammation Java, axée sur la recherche robotique multi-agents.

*SoccerBots* est une simulation de football de la ligue *RoboCup F-180*. Elle reproduit la dynamique et les dimensions d'un jeu de ligue robot RoboCup de petite taille. Deux équipes de cinq robots s'affrontent sur une table de ping-pong en poussant une balle de golf orange dans le but de l'adversaire.

Nous utiliserons *SoccerBots* avec *TBSim*, un outil de simulation. Les mêmes systèmes de contrôle qui s'exécutent en simulation peuvent également utiliser des robots réels avec l'application *TBHard*.

Linterface entre le système de contrôle du robot, ses capteurs et ses actionneurs est fournie par l'API *SocSmall*. Voici les principaux outils fournis par cette API :

- Détecter si le robot est en mesure de tirer
- Obtenir un vecteur pointant vers la balle
- Obtenir un tableau de vecteurs pointant vers tous les autres joueurs
- Obtenir la position du joueur sur le terrain
- Obtenir l'angle de vue du joueur

- Pousser la balle
- Tirer
- Régler la vitesse
- Tourner.

# 3 Architecture multi-agents

## 3.1 Subsomption

Le comportement de nos agents intelligents est régi par une architecture réactive de subsomption, comme vu en cours.
L'architecture de subsomption comporte plusieurs comportements, étant responsable de la réalisation d'une tâche simple. Chaque comportement a une priorité différente, et il en résulte d'une hiérarchie verticale et une relation d'inhibibition totale.

Dans notre réalisation, nous avons défini cinq comportements. Voici la hiérarchie de nos comportements (le premier est prioritaire sur le deuxième, et ainsi de suite) :

1. Unlocker Behaviour
2. Offensive Midfielder Behaviour
3. Defender Behaviour
4. Center Forward Behaviour
5. Default Behaviour

Les comportements seront détaillés dans la section Comportements. On note que, dans notre code, l'implantation de cette architecture de subsomption est faite à l'aide du patron de conception *Chaîne de responsabilité*.

## 3.2 Attraction et répulsion

Dans notre réalisation, nous avons utilisé le concept d'attraction-répulsion, vu en cours.
En fonction du comportement de l'agent, ce dernier calcule une attraction ou une répulsion sur différents éléments du terrain (but allié, but ennemi, joueur allié, joueur ennemi, bords du terrain, etc.).
L'attraction-répulsion est une part essentielle dans la performance de notre équipe car elle permet, non seulement, d'éviter les collision entre les joueurs, mais surtout de prendre la balle dans le sens de l'attaque et, ainsi, d'éviter de pousser la balle vers notre but.

## 3.3 Comportements

Les comportements possèdent une fonction de vérification d'action réalisable et une fonction de réalisation d'action. Si la fonction de vérification d'action renvoie que l'action est réalisable, alors on active la fonction d'action, qui effectue l'action du comportement.
Par ailleurs, chacun de ces comportements utilise des fonctions de calculs vectorielles présentent dans une classe Utilitaires. Dans cette classe, on y retrouve essentiellement des fonctions vectorielles, indispensables pour les calculs de trajectoire et d'attraction-repulsion.

### 3.3.1 Unlocker Behaviour

Le comportement Unlocker Behaviour est celui avec la plus forte priorité. Il est déclenché lorsque le jeu est bloqué trop longtemps, c'est à dire, lorsque le ballon est bloqué par les joueurs. Dans ce cas, tous les joueurs s'écartent du ballon pendant un cours moment. Pour cela, on augmente la force de répulsion entre les joueurs et le ballon.

### 3.3.2 Offensive Midfielder Behaviour

Le comportement Offensive Mifielder Behaviour est un comportement offensif. Ce comportement est paramétré pour trois agents. Ces trois agents forment alors une formation de trio offensif, où le possesseur de la balle peut tirer. Trois tirs sont possibles : le tir à longue distance, le tir à moyenne distance et le tir à courte distance. Pour valider chacun de ces tirs, on vérifie s'ils sont réalisables selon un angle de vue, la présence d'ennemi dans cet

angle et la distance par rapport au but ennemi.
Cette stratégie est activée par les joueurs les plus proches du ballon. Ils sont alors fortement attiré par le ballon et par le but adverse.
Dans un premier temps, l'idée fut de développer un algorithme de nuée d'oiseaux pour simuler le déplacement de groupe du trio, comme vu en cours. Et plus particulièrement, un algorithme de nuée en "V". Or, nous nous sommes rendu compte qu'en utilisant seulement l'attraction-répulsion qu'on a développé et en fixant exactement trois joueurs pour ce comportement, cette formation de trio en "V" était exactement réalisée.
On note que, grâce aux fonctions d'attraction-répulsions développées, ces joueurs pousse tout le temps la balle dans le sens de l'attaque.

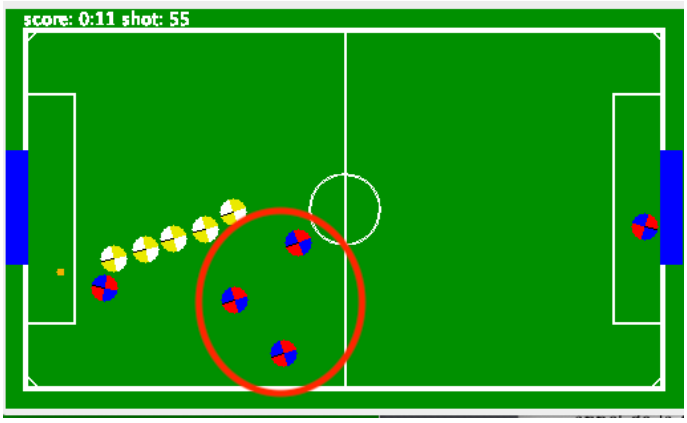


Figure 1: Formation du trio offensif

### 3.3.3 Defender Behaviour

Ce comportement est celui d'un défenseur. Le défenseur est seul et se positionne au niveau du but allié. Ainsi, on pourrait dire qu'il est un gardien de football. Mais, lors d'une charge adverse offensive, et s'il est seule dans la surface allié, il part au contact pour récupérer le ballon.
Son attraction au but allié est la plus forte.

### 3.3.4 Center Forward Behaviour

Le comportement Center Forward Behaviour est celui d'un avant-centre. Il a le rôle d'un renard. Il se place tout le temps dans le camp adverse, et

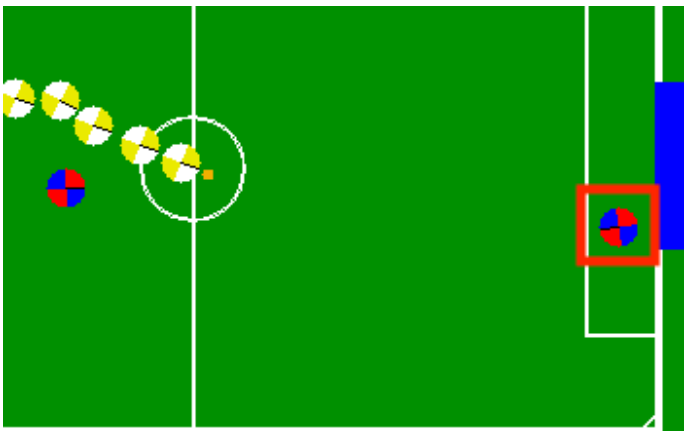

Figure 2: Défenseur en tant que gardien

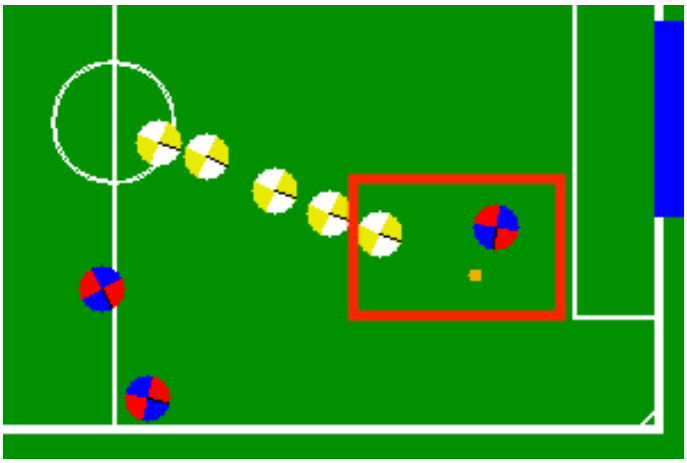

Figure 3: Interception de la balle par le défenseur

patrouille autour du centre du terrain afin de récupérer la balle lorsque celle-ci est remise au centre, pour prendre le comportement d'un Offensive Midfielder Behaviour, foncer vers le but et tirer. C'est un vrai sournois !

#### 3.3.5 Default Behaviour

C'est le comportement par défaut. C'est le joueur qui a une position neutre : ni proche du ballon, ni proche du but ennemi, ni proche du but allié. C'est le comportement basique, toujours déclenchable.

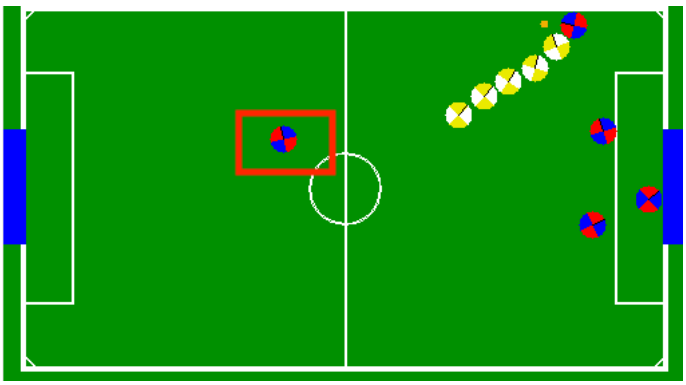

Figure 4: Avant-centre qui vagabonde entre le centre du terrain et le but adverse

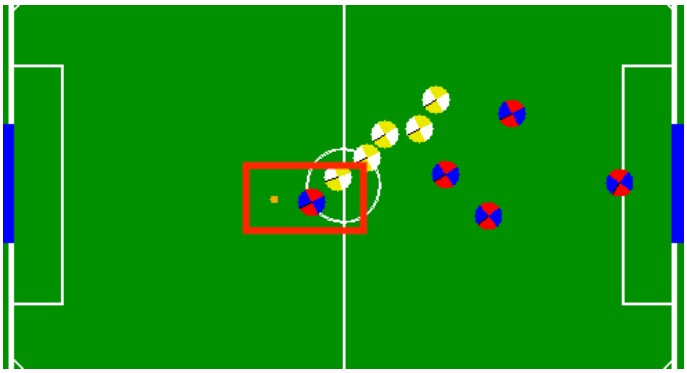

Figure 5: Avant-centre qui récupère la balle au centre du terrain

# 4 Résultats

## 4.1 Notre équipe à gauche du terrain

BibiElGarehTeam *versus* Basic Team : **26 − 0**
BibiElGarehTeam *versus* Go To Ball : **24 − 1**
BibiElGarehTeam *versus* Jun Team Hetero G : **42 − 0**
BibiElGarehTeam *versus* Matti Hetero : **2 − 0**

### 4.2 Notre équipe à droite du terrain

Basic Team *versus* BibiElGarehTeam : **0 − 25**
Go To Ball *versus* BibiElGarehTeam : **0 − 26**
Jun Team Hetero G *versus* BibiElGarehTeam : **3 − 40**
Matti Hetero *versus* BibiElGarehTeam : **0 − 3**

## 5 Conclusion

Grâce à ce projet, qui s'est inscrit dans l'unité d'enseignement *Intelligence artificielle distribuée*, nous avons pu manipuler les notions d'intelligence artificielle distribuée liées aux agents réactifs (architecture de subsomption, attraction-repulsion, etc.), vues en cours. Nous tenons à remercier les acteurs du projet *TeamBots*, sans qui nous n'aurons pas pu nous exercer d'une telle manière.
Par ailleurs, tout au long du développement de ce projet, nous avons été rigoureux sur la qualité du code. En effet, nous avons nommer avec soin nos variables, méthodes, classes, etc. Nous avons utiliser des packages pour la team, les comportements et les actions. Et, nous avons utiliser les *patterns* qui nous ont parus utiles. Puis, nous avons généré une JavaDoc, pour facilité la compréhension et la maintenance du code.
Ce projet fut complexe à prendre en main sur l'aspect des calculs vectoriels. C'est pourquoi, les valeurs utilisées dans le code, et notamment les facteurs donnés pour les calculs, restent empiriques. Par ailleurs, on note que notre réalisation a produit une équipe très offensive. Et, nous pourrions avoir de gros problèmes contre des équipes défensives, jouant sur des contre-attaques furieuses.
Nous avons beaucoup d'idées d'amélioration du projet. Mais si on ne devrait en retenir qu'une majeur, ce serait celle de la communication pour un système de *passe-et-va*. En effet, l'idée serait de produire une communication entre les agents (comme vu en cours), afin de leur permettre de faire des passes entre eux, et plus particulièrement des appels et contre-appels (je passe et je me positionne aussitôt devant pour faire un appel et recevoir la balle).