



## Projet Foot – 2013

---

***Baskiotis Nicolas***

***Eddy Cai / Steve Moukouri***

***Groupe 1***

***Licence Informatique (2017/2018)***

# 1

## Sommaire

---

### ***Introduction***

---

### ***Structure du code***

---

### ***Stratégies***

---

- 1 contre 1
- 2 contre 2

### ***Optimisation***

---

### ***Conclusion***

---

# 2 Introduction

---

Dans le cadre de l'UE 2i013, nous avons été amenés à réaliser un projet sur l'intelligence artificielle autour du thème du football. Une partie du code nous était fournie (l'interface graphique, les règles du jeu) et nous devions ainsi nous occuper de l'implémentation des joueurs. En somme, le but était au préalable de réaliser des stratégies, de les simuler ensuite, et enfin de les confronter à celles des autres binômes. Ces stratégies, bien que basiques au départ, tendaient progressivement à s'améliorer en utilisant des méthodes liées à l'apprentissage statistiques (algorithme génétique, arbre de décisions, etc.).

Dans un premier temps nous vous parlerons brièvement de l'architecture logicielle que nous avons choisie. Par la suite, nous rentrerons dans le vif du sujet en évoquant avec vous les stratégies que nous avons adoptées et ; nous terminerons en vous montrant comment nous avons pu améliorer chacune d'entre elles.

## Structure du code

---

Avant d'aborder la partie « théorie » du sujet, nous avons tenu à faire avec vous un court entracte sur l'organisation de notre code. Ce dernier a été réparti en deux principaux modules : outil et stratégie. Le premier comme son nom l'indique, est une sorte de coffret, de boîte à outils qui nous a servi aussi bien à retrouver les constantes utiles, que les éléments redondants de notre code. C'est en soi, un objet qui peut se mouvoir tantôt en joueur (en intégrant tous les paramètres qui composent l'état de ce dernier), tantôt en balle, mais le plus souvent en utilitaire. Le module stratégie quant à lui se rapporte à l'utilisation directe des fonctions implémentées dans outil pour en faire des stratégies cohérentes et « concrètes ».

# 3

## Stratégies

---

Dans cette partie, nous allons présenter les différentes stratégies que nous avons codées : Fonceur, Dribbleur, Défenseur et Attaquant. Tout au long du projet, nous avons sans cesse cherché la meilleure des stratégies pour les différentes situations (1v1, 2v2, 4v4, etc.). Pour ce faire, nous avons dû les coder, les adapter au jeu, les tester et les optimiser.

### ***1 contre 1***

---

Notre première stratégie a été le fonceur, qui nous a aidés à nous familiariser avec l'environnement. Un fonceur est codé de manière très simpliste. Si le fonceur peut tirer, il le fait en direction du centre du but adverse. À défaut, il court vers la balle. Cette stratégie a rencontré des problèmes dès le début. En effet, lorsque ce fonceur joue contre d'autres stratégies de fonceur, on se retrouve dans une situation où les deux fonceurs courent ensemble vers la balle et, sachant que la vitesse maximale est la même pour tous les joueurs, les deux fonceurs atteignent en même temps la balle et chacun tire sur celle-ci. Ce qui donne une direction aléatoire à la balle et aucun des deux joueurs n'arrive à marquer un but (Figure 1).

# 4

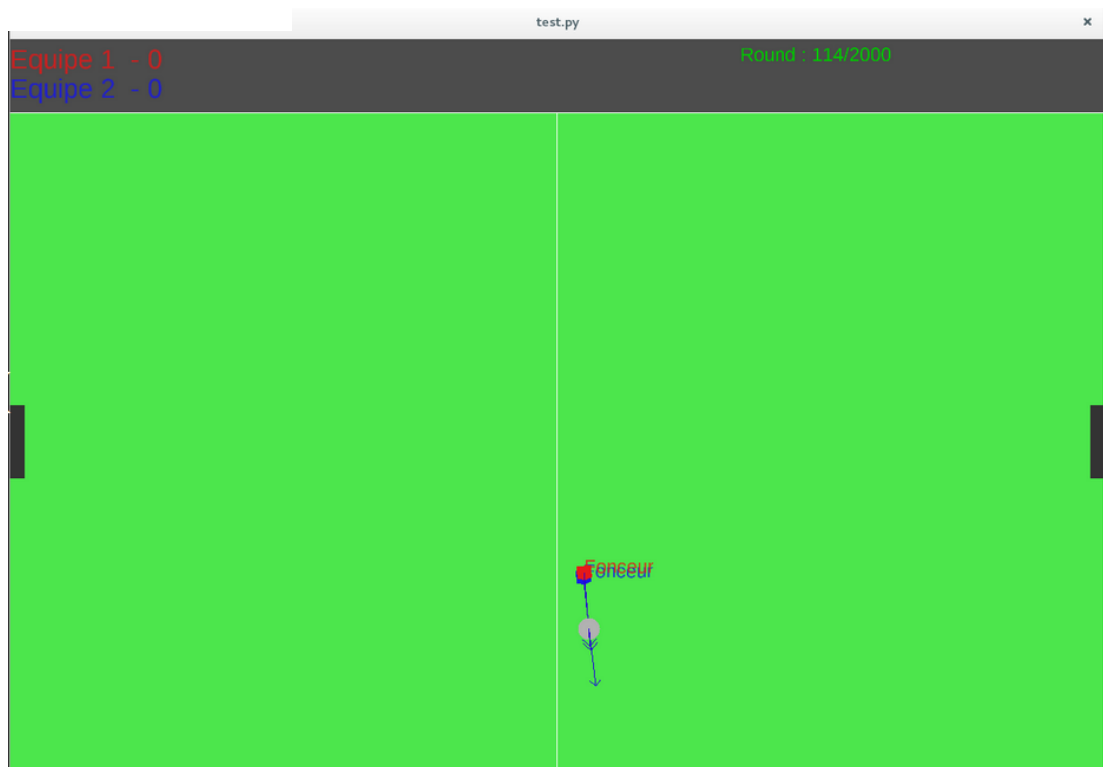


Figure 1.Exemple de deux fonceurs

Pour pallier ce problème, nous avons donc créé une nouvelle stratégie plus élaborée afin de gagner contre des stratégies de fonceur. Pour cela, nous avons codé un dribbleur. Ce dribbleur est codé de manière à agir quand la balle rentre dans son périmètre d'action que nous avons nous même défini. Ainsi, ce dribbleur récupère la balle puisqu'il sera le plus proche de celle-ci puis, dribble le fonceur adverse qui est entrain de courir vers la balle et marque.

Nous allons détailler les situations les plus vues lorsqu'on utilise ce dribbleur (Figure 2).

# 5

| Étape | Notre dribbleur          | Adversaire          |
|-------|--------------------------|---------------------|
| 1     | Ne fais rien             | Court vers la balle |
| 2     | Ne fais rien             | Tir sur la balle    |
| 3     | Récupération de la balle | Court vers la balle |
| 4     | Dribble l'adversaire     | Court vers la balle |
| 5     | Tir sur la balle         | Court vers la balle |
| 6     | BUT                      | Court vers la balle |

Figure 2. Stratégie dribbleur

Grâce à cette stratégie, nous avons pu gagner la majorité de nos rencontres en 1v1. Cette stratégie est très efficace mais renferme elle aussi, quelques inconvénients. Il se peut que l'adversaire fasse la même chose que notre dribbleur, à savoir, attendre la balle et agir après. Pour pallier ce problème, nous pouvons coder notre dribbleur afin qu'il agisse, si l'adversaire ne fait rien pendant un certain temps.

## **2 contre 2**

---

Pour le 2v2, nous avons décidé d'utiliser un défenseur et un attaquant. Nous voulions encaisser le moins de buts possibles. Pour cela, nous avons commencé à développer notre défenseur.

Notre défenseur est positionné en fonction de l'attaquant adverse, il se place entre l'adversaire et le milieu de nos buts. Ainsi, il peut défendre la zone de laquelle l'attaquant peut tirer. Par la suite, si notre défenseur peut renvoyer la balle, il l'a renvoie en fonction du positionnement de l'attaquant adverse. Si l'adversaire se trouve au-dessus de notre défenseur, notre défenseur renvoie la balle vers le bas et réciproquement. Cela force l'adversaire à parcourir un plus long trajet pour récupérer la balle, et nous donne ainsi un avantage pour la récupération de celle-ci.

En ce qui concerne notre attaquant, il se comporte de manière agressive. Il le fait de la façon suivante : il fonce vers la balle à chaque début de jeu, et s'il rencontre un adversaire il essaye de le

# 6

dribbler. S'il réussit et se retrouve dans la zone d'action de tir, il tire. En revanche, s'il perd la balle, il se positionne en fonction de l'attaquant adverse. Il se place ainsi soit sur la ligne médiane vers le haut, soit sur la ligne médiane vers le bas.

Le placement dépend de la position de l'adversaire. Si l'adversaire se trouve au-dessus de notre défenseur, notre attaquant se placera vers le bas et vice-versa. Par conséquent, quand notre défenseur dégage la balle, il la renvoie sur notre attaquant et par la même occasion, crée une plus longue distance entre la balle et l'attaquant adverse.

## Optimisation

---

Comme mentionné à maintes reprises dans les paragraphes précédents, notre code n'est pas infallible et possède ses limites. Cependant, certaines d'entre elles peuvent être atténuées voire rectifiées. En ce sens, bien vite nous avons pu mettre en lumière deux types de failles : celles liées à l'utilisation de données arbitraires d'une part, et de l'autre, toutes celles en relation avec l'utilisation stéréotypée de certaines stratégies.

### **Problèmes en lien avec les données arbitraire :**

Bien qu'utiles dans un premier temps pour visualiser les choses et concrétiser certaines de nos idées, les valeurs prises sans fondement ou test préalable tendaient à être constamment imprécises et diminuaient la performance de chacun de nos joueurs (Figure 3).

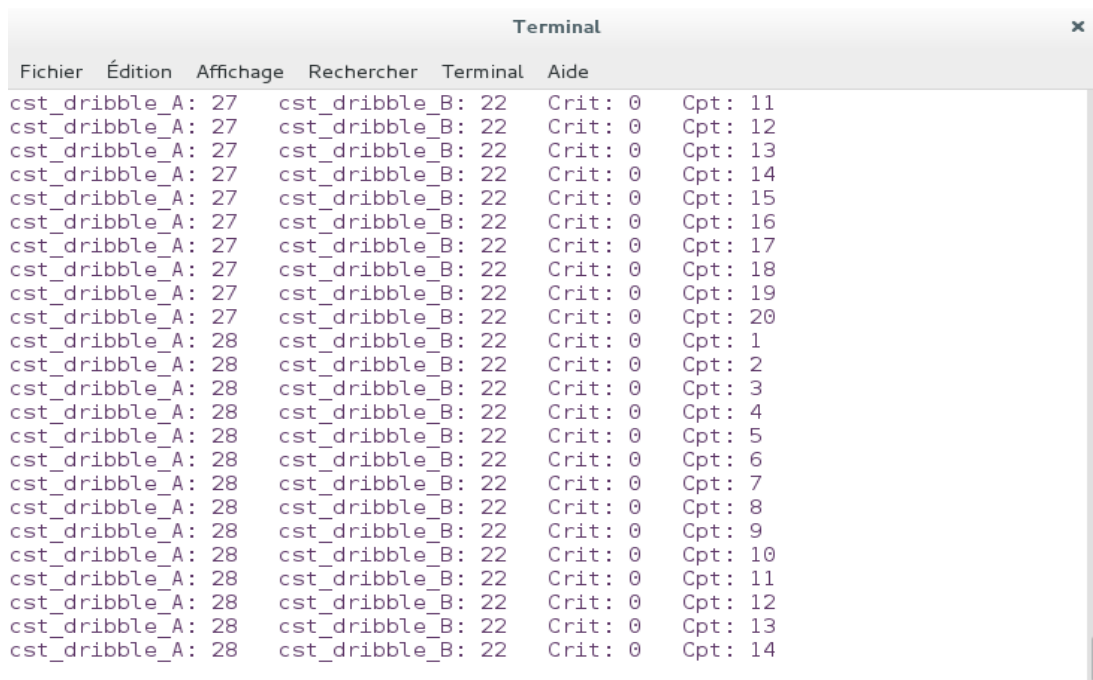


Figure 3.Tir imprécis

Pour pallier ce problème, l'emploi de la recherche séquentielle a été un outil redoutable. Le principe est simple et repose essentiellement sur le test répétitif d'un large panel de valeurs (d'une variable définie au préalable) dans des conditions données. Pour notre projet, ce fut dans un premier choix « l'angle » du dribble : la valeur étant mal choisie, notre joueur avait tendance à tirer la plupart du temps sur le joueur adverse plutôt que dans une position qui lui permettait de conserver la balle, puis ; dans un second, la prédiction de la trajectoire de la balle : celle-ci étant fonction à la fois de la vitesse de la balle, de notre distance vis-à-vis d'elle, mais aussi de critères non maîtrisés tels que l'accélération initiale donnée à la balle par un autre joueur par exemple.(Figure4).



# 8



| Fichier           | Édition           | Affichage | Rechercher | Terminal | Aide |
|-------------------|-------------------|-----------|------------|----------|------|
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 11    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 12    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 13    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 14    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 15    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 16    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 17    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 18    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 19    |          |      |
| cst_dribble_A: 27 | cst_dribble_B: 22 | Crit: 0   | Cpt: 20    |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 1     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 2     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 3     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 4     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 5     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 6     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 7     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 8     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 9     |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 10    |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 11    |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 12    |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 13    |          |      |
| cst_dribble_A: 28 | cst_dribble_B: 22 | Crit: 0   | Cpt: 14    |          |      |

Figure 4. Recherche séquentielle

## Problèmes en lien avec l'utilisation stéréotypée de stratégie :

L'un de nos plus gros problèmes était de réussir à rendre nos joueurs suffisamment intelligents dans leur appréhension du jeu afin qu'ils puissent : non seulement, utiliser la bonne action dans les conditions définies au préalable, mais en outre et mieux encore ; s'adapter en fonction de la situation de jeu. Pour ce faire nous nous sommes servis de l'apprentissage supervisé via l'utilisation d'arbre de décision. En quelques mots, le principe est le suivant : recréer des pseudos classes ou classes intermédiaires qui reprendront le principe de nos classes de base, puis ; les soumettre à des conditions pour qu'elles soient choisies successivement en fonction de situations de jeu (d'un état) et donc selon qu'elles sont plus ou moins plus favorables (Figure 5).

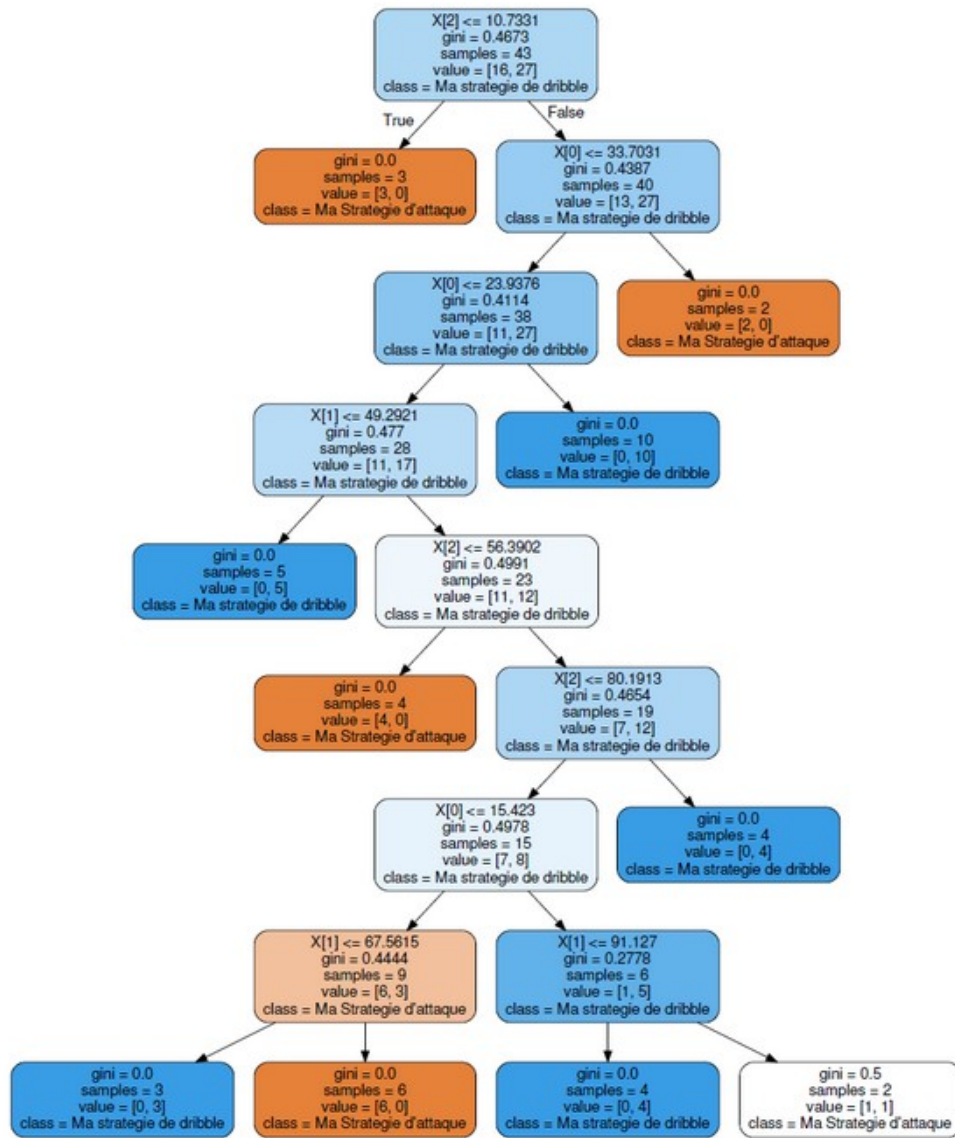


Figure 5. Arbre de décision

# 10

## Conclusion

---

Ce projet a donc été pour nous une expérience enrichissante. En effet, non seulement, il nous a permis de nous familiariser avec de nouvelles méthodes de travail (github) mais aussi, et surtout, il nous donné la possibilité d'apprendre de nouveaux concepts théoriques (apprentissage statistique, intelligence artificielle).

Par ailleurs il nous a amené à mettre en pratique ces nouvelles connaissances dans un contexte ludique.

Quelques regrets demeurent cependant, dans la mesure où nous aurions souhaité : avoir plus de temps pour approfondir certaines notions (machine Learning), coder plus de stratégies (notamment en 4 vs 4) mais surtout, pouvoir utiliser de manière efficace toutes les méthodes d'apprentissages statistiques mises à notre disposition (algorithme génétique).