# Rapport IN104

Robin Labbé, Pierre-Gwenaël Pervès

May 2019

## 1 Lien du Repository GitHub

https://github.com/pierregwenaelmoon/SoutenanceHanabi.git
Contenu : 2 dossiers :
- HanabiIA1 : contient une première version d'IA
- HanabiIA2 : contient une deuxième et troisième version d'IA

## 2 Choix de la stratégie

Nous avons opté pour une stratégie de type HatGuessing. Nous supposons que vous connaissez sa théorie. Nous nous contenterons donc d'expliquer son implémentation.

## 3 Choix d'implémentation

Nous vous prions de bien vouloir tester nos 2 versions d'IA. Les 2 ont été implémentées différemment.

### 3.1 1ère version de l'IA HatGuess

L'algorithme du HatGuess nécessite de stocker plusieurs variables temporaires : score des différentes mains, nombres de fois où une carte a été jouée depuis que le dernier indice a été donné... Nous avons choisi de placer certaines variables dans la classe Game du fichier deck.py. Les variables qui y sont stockées sont celles qui ont besoin d'être sauvegardées d'un tour au suivant. Critique : compliqué d'effectuer des tests pour vérifier si ce que l'IA propose est viable.

### 3.2 2ème version de l'IA HatGuess

Globalement la même que la première. Son intérêt résulte dans le fait de pouvoir effectuer des tests sur la fonction de manière plus efficace. L'I.A. est décomposée en 2 fonctions et est plus synthétique. Les variables sont stockées dans des attributs propres à sa classe : la classe Deck n'est donc pas modifée. Les tests

de véracité de notre IA, effectués par des tests unitaires, seront donc plus faciles à effectuer.

### 3.3 3ème version de l'IA HatGuess

Nous avons pu observer sur différentes parties que des scores inférieurs ou égaux à 17 étaient bien souvent dûs à une fin de jeu trop prudente.

Lorsqu'il ne reste plus de cartes dans le deck,qu'il ne manque plus que certains 5 sur la table, que notre recommandation est de jouer une carte et que le nombre de pièces rouges est égal à 2 : il serait intéressant d'obéir à la recommandation sans faire attention au nombre de pièces rouges déjà en jeu.

Nous avons implémenter cette nouvelle I.A. et avons observé que nous nous étions trompés : elle est moins efficace. Réferrez-vous s'il vous plaît à la question 5.2 pour les résultats de cette A.I.

## 4 Tests effectués sur nos Intelligences Artificielles

Vous trouverez les différents tests dans notre Repository Git dans les dossiers Test propres à chaque I.A.

### 4.1 1er test : test d'arrêt du jeu

Notre jeu ne doit s'arrêter que dans 2 cas : 3 pièces rouges OU deck vide.

### 4.2 2ème test : Type du résultat de notre I.A.

Nous testons si ce que retourne notre IA est bien un résultat sous la forme d'une chaîne de caractères de longueur 2 ou 3 et dont la première lettre est bien un p , un d ou un c .

### 4.3 3ème test : Test de calcul de score sur une main

Nous testons ici la capacité de notre IA à lire le jeu. Nous effectuons ce test sur une combinaison de cartes bien précise. Nous regardons le résultat et concluons sur sa crédibilité.

### 4.4 4ème test : Test de la crédibilité de notre IA

Nous testons notre IA sur une situation bien précise du jeu : nous savons donc ce que l'IA doit retourner. Nous regardons ce qu'elle nous renvoie et concluons sur sa crédibilité.

# 5 Résultats des Intelligences Artificielles

## 5.1 Résultats de l'I.A. 1 et de l'I.A. 2

Nous avons choisi de lancer 6000 parties et d'en extraire la répartition des scores. Nous comptabilisons aussi le nombre de parties à 25 et 24 points.

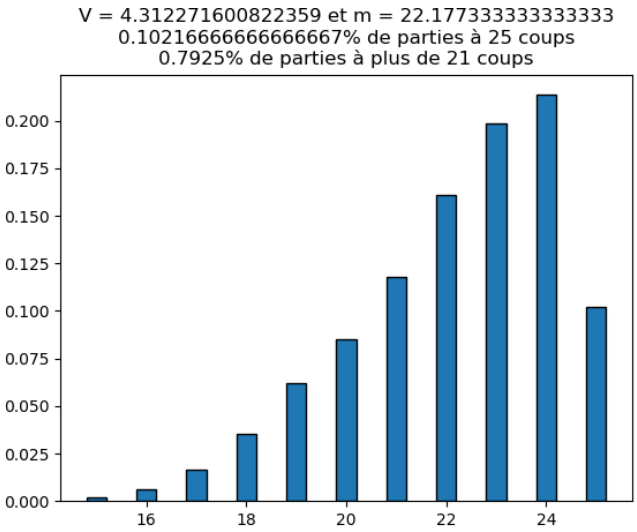


Figure 1: Simulation de Hanabi pour l'I.A. 1

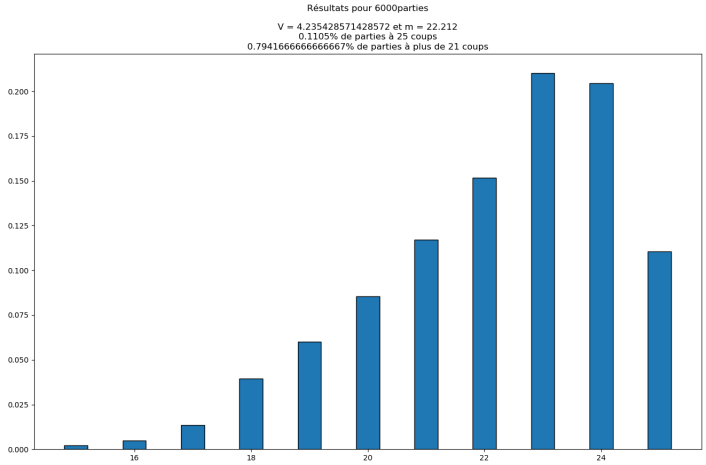


Figure 2: Simulation de Hanabi pour l'I.A. 2

### 5.2 Une 3ème I.A. moins performante...

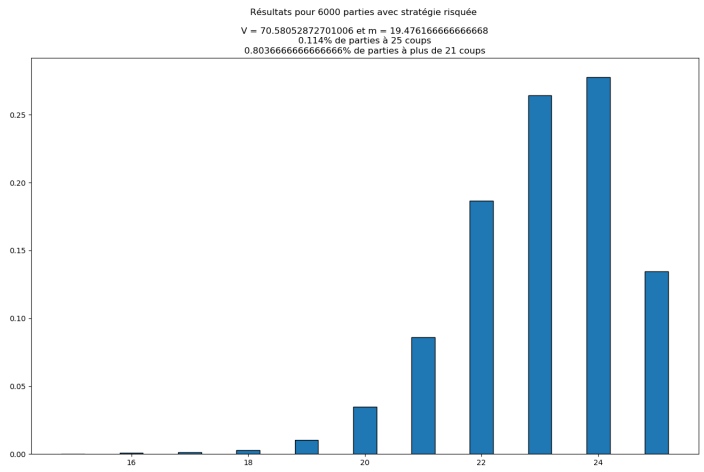


Figure 3: Simulation de Hanabi pour l'I.A. 3

## 6 Problèmes rencontrés

Nous avons rencontré certaines difficultés dans l'utilisation du deck. Voici un de nos problèmes : Nous cherchions (dans notre IA) à obtenir le nom du joueur actuel. Nous utilisions la variable game.currentPlayer.name qui ne nous renvoyait pas exactement le nom du joueur mais un nom entouré de divers symboles. Cette mauvaise utilisation du deck nous a bloqué pendant un sacré moment...

## 7 Perspectives d'évolution

Nous aimerions pouvoir proposer à l'I.A. de jouer contre une autre I.A. ou contre une personne. Elle ne peut pour le moment que jouer avec elle-même.

Nous remarquons que nous aurions pu mener de manière plus efficace le projet. La création de la 1ère I.A. s'est faite sans fonction test, et ne marchait pas. C'est d'ailleurs pour cette raison que nous avons commencé à coder différemment une autre I.A. : les variables étaient plus facilement accessibles et les fonctions tests étaient déjà faites.

Nous avons tout de même par la suite réussi à debugger la première I.A. en comprenant une erreur que nous avions aussi commise sur la 2ème.

## References

Le PDF proposé par J.D. Garaud.