Création d'une intelligence artificielle de Poker
Projet d'informatique 2020

Supervisé par Loïc Midy

Eléonore Blanchard
ENSAE Paris
eleonoredblanchard@gmail.com

Yannick Ly
ENSAE Paris
yannick.ly@ensae.fr

Abstract

L'objectif de notre projet[1] consiste à créer une intelligence artificielle pour le Poker Limit Texas Hold'em à deux joueurs. Nous cherchons à trouver l'équilibre de Nash via l'algorithme Counterfactual Regret Minimization. Pour cela, nous travaillerons sur le Poker Kuhn qui est une version simplifiée du Poker car il ne comporte que 2 joueurs, 3 cartes disponibles et une seule incertitude (la carte piochée par l'adversaire). Dans un deuxième temps, nous avons travailler à l'implémentation de cet algorithme au Poker Limit Texas Hold'em, et à ses implications, comme le temps de calcul et la puissance nécessaires pour l'algorithme.

1 Introduction

1.1 Problématique

Notre projet consiste à créer une intelligence artificielle, qui une fois entraînée nous renvoie une stratégie pour jouer au Poker Limit Texas Hold'em.

Les principaux enjeux de notre sujet reposent sur le fait qu'il s'agit d'un jeu à information incomplète car il réside une part d'incertitude car nous ne pouvons pas connaître la main de l'adversaire, ni sa stratégie. De plus, la combinatoire des situations possibles est énorme (10^{161} décisions possibles). De ce fait, les algorithmes développés théoriquement ont longtemps été limités par les puissances de calcul disponibles.

Pourtant au cours des 20 dernières années des progrès énormes ont été réalisés et début 2017 le programme informatique Libratus a battu très largement plusieurs des meilleurs joueurs du monde dans au jeu de poker Texas hold'em à deux joueurs sans limite.

C'est dans ce cadre que nous cherchons à mette au point une stratégie dont l'objectif principal est de minimiser son regret pour chacune des décisions qu'il prendra. Pour cela, nous entraînerons l'algorithme sur un grand nombre de simulations afin qu'il apprenne par renforcement.

1.2 Méthodologie

Tout d'abord, nous allons nous intéresser au Poker Kuhn à deux joueurs qui une forme simplifiée du Poker sur lequel nous implémenterons l'algorithme de Counterfactual Regret Minimization (CRM) afin de mieux comprendre la stratégie sous-jacente.

Puis, nous évaluerons via la simulation de parties les forces et faiblesses de l'intelligence artificielle (IA) CFR en l'opposant à des IA simplistes (stratégies aléatoire et basée sur l'espérance de gain).

Enfin, nous étendrons l'algorithme CRM au Poker Limit Texas Hold'em à deux joueurs et travaillerons sur les difficultés de son implémentation en réduisant le nombre de mains possibles (clustering).

2 Algorithme de Minimisation du Regret (Counterfactual Regret Minimization)

2.1 Théorie des jeux

Nous cherchons à créer une stratégie qui se base sur les informations connues, à savoir les cartes qui ont été distribuées au joueur, les cartes de la rivière révélées au cour du jeu et les informations que l'adversaire lui révèle involontairement dans sa manière de jouer.

Ainsi, du fait de ses caractéristiques, le Poker Limit Texas Hold'em (ou Poker Kuhn) à deux joueurs, peut s'apparenter à un jeu fini à somme nulle à deux personnes car :

- **deux personnes** : dans notre cadre d'étude, il n'y a que deux joueurs

[1] https://github.com/YannickLy/Poker-Project

- **jeu fini** : il y a un nombre fini d'actions possibles (Il y a un nombre limité de tours (preflop, flop, turn et river) et les joueurs ne peuvent miser qu'une fois par tour)
- **jeu à somme nulle** : tous les payoffs somment à zéro
- **information imparfaite** : on ne connaît pas la carte de l'adversaire
- **jeu avec un facteur chance** : il y a de l'incertitude introduite par la distribution des cartes au hasard

Il est possible d'approximer une stratégie pour ce type de jeu à l'aide d'un algorithme de minimisation du regret (counterfactual regret minimization). L'objectif de l'algorithme est alors de trouver un équilibre de Nash[2] pour un tel jeu.

2.2 Présentation de l'algorithme

L'algorithme de minimisation du regret est en fait issu de la théorie de l'apprentissage sans regret (no-regret learning). C'est une stratégie qui apprend en jouant contre lui même

L'algorithme apprend en jouant de manière répétitive contre lui même. Il commence avec une stratégie qui est uniformément aléatoire, où il jouera chaque action à chaque noeud de décision avec une probabilité égale. Il simule ensuite des jeux contre lui-même. Après chaque jeu, il re-parcours chaque noeud de décision et cherche comment améliorer sa stratégie. Il répète ce processus pour un grand nombre de jeux, en améliorant sa stratégie à chaque fois. A chaque partie, il se rapproche d'une stratégie optimale pour le jeu.

2.3 Implémentation

L'algorithme améliore sa stratégie dans le temps, en sommant le regret total que chaque action peut engendrer à chaque noeud décision. Un regret positif (resp. négatif) associé à une action, signifie que l'algorithme améliore (resp. détériore) sa situation en choisissant cette action. Après chaque simulation, le CRM ajuste les probabilités de choisir telle action, plutôt qu'une autre, en augmentant celles qui minimisent le regret. Autrement dit, plus une action a été bénéfique dans le passé, plus elle sera choisie dans le futur.
L'équilibre de Nash est alors obtenu en calculant la moyenne des probabilités cumulées sur toutes les simulations.
En théorie, pour obtenir un équilibre de Nash, il faudrait simuler un milliard de parties.

[2]Situation dans laquelle aucun joueur ne regrette son choix étant donné le choix des autres joueurs.

3 Poker Kuhn

3.1 Règles du jeu

Le Poker Kuhn se joue a deux joueurs. Dans ce jeu, le paquet ne contient que 3 cartes : roi, dame et valet. Le jeu se déroule de la façon suivante : Chaque joueur doit miser 1 (la blinde) pour entrer dans le jeu. Chaque joueur reçoit une carte et la troisième carte reste cachée.

Le premier joueur peut check ou bet.
Si le premier joueur check, alors le deuxième joueur peut check ou bet.

- Si le deuxième joueur check, alors les joueurs montrent leurs cartes, et celui qui a la meilleure remporte le pot de 2.
- Si le deuxième joueur bet, alors le premier joueur peut fold ou call.
  - Si le premier joueur fold, alors le deuxième joueur gagne et remporte le pot de 3.
  - Si le premier joueur call, alors les joueurs montrent leurs cartes, et celui qui a la meilleure remporte le pot de 4.

Si le premier joueur bet, alors le deuxième joueur peut fold ou call.

- Si le deuxième joueur fold, alors le premier joueur gagne et remporte le pot de 3.
- Si le deuxième joueur call, alors les joueurs montrent leurs cartes, et celui qui a la meilleure remporte le pot de 4.

3.2 Équilibre de Nash

Ce jeu dispose de plusieurs équilibres de Nash en stratégie mixte. Cela signifie que si les deux joueurs jouent ces stratégies, alors le premier joueur devrait perdre a un taux de -1/18. Il n'y a pas de stratégie pure. On peut définir les équilibres de Nash comme suit :
Pour le premier joueur, il existe un nombre infini de stratégies d'équilibre car sa stratégie ne dépend

que d'un seul paramètre (sa carte). Soit une probabilité libre $\alpha \in [0; 1/3]$.

- S'il a un roi, il doit bet avec la probabilité 3α (et s'il check, il doit toujours call si le deuxième joueur bet).
- S'il a une dame, il doit toujours check, puis call avec la probabilité $\alpha + 1/3$ si le deuxième joueur bet.
- S'il a un valet, il doit bet avec une probabilité α (et s'il check, il doit toujours fold si le deuxième joueur bet).

Pour le deuxième joueur, en revanche, il n'y a qu'une stratégie d'équilibre :

- S'il a un roi, il doit toujours bet ou call.
- S'il a une dame, il doit check si possible, ou call avec une probabilité de 1/3.
- S'il a un valet, ne jamais call et bet avec une probabilité de 1/3.

Nous pourrons vérifier après l'entraînement de nos algorithmes que ce sont bien ces équilibres de Nash qui sont obtenus.

3.3 Implémentation

3.3.1 Poker Kuhn

L'implémentation du Poker Kuhn nécessite de comprendre la structure du jeu. Elle est définie par un arbre (Figure 1) composé :

- **d'un noeud chance** dont la sortie est issue d'un tirage aléatoire parmi les combinaisons suivantes : KQ, KJ, QK, QJ, JK, JQ.
- **de noeuds décision** dont la sortie est le résultat de la décision d'un des deux joueurs : bet ou check, call ou fold.
- **de noeuds terminaux** déterminant les gains et pertes de chacun des deux joueurs.

Pour implémenter l'arbre, nous avons créé une classe Noeud dont héritent les classes NoeudChance et NoeudDecision. La racine de l'arbre est créée en initialisant un NoeudChance dont les enfants sont toutes les combinaisons possibles (KQ, KJ, QK, QJ, JK, JQ), chacun de ces enfants ayant eux même pour enfants des NoeudDecision. Par récursivité, nous créons ainsi l'intégralité de l'arbre jusqu'à arriver à un noeud terminal.

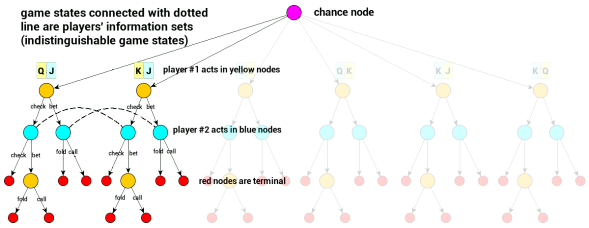

Figure 1: Arbre du Poker Kuhn.

A chaque NoeudDecision est rattaché un set d'informations (information set) qui contient les informations suivantes : carte du joueur et historique des actions.

3.3.2 CRM

Une fois l'arbre construit, on peut enfin s'intéresser à l'algorithme CRM appliqué au Poker Kuhn. Afin de parcourir l'arbre, nous avons décidé d'implémenter deux solutions :

- Parcourir l'intégralité de l'arbre à chaque simulation;
- Ne parcourir que les branches de l'arbre issues d'un seul des enfants du NoeudChance choisi aléatoirement à chaque simulation

Une fois les simulations réalisées, nous calculons les équilibres de Nash comme expliqué précédemment.

3.3.3 Autres stratégies

Afin d'évaluer la performance des stratégies générées par l'algorithme CRM, nous avons également créé une stratégie dont les probabilités des actions sont aléatoires (Random).
Nous avons également implémenté une stratégie optimale espérance gain (OptEspGain), qui consiste à prendre les décisions qui maximisent les gains.
Enfin, nous avons également implémenté une fonction qui nous permet de jouer contre l'une des stratégies au choix, afin de jouer notre propre stratégie de manière interactive.

3.4 Comparaison des différentes stratégies

Dans cette section nous avons cherché à évaluer la performance du CRM face aux autres stratégies.

3.4.1 Comparaison entre CRM et lightCRM

Dans un premier temps, nous avons comparé les algorithmes CRM et lightCRM afin de voir si les équilibres de Nash sont identiques. En effet, si tel

lightCRM VS. CRM (100 000 parties)		
Nombre d'itérations pour l'entraînement de l'IA	**lightCRM**	**CRM**
Payoff		
1 000	1008	-1008
5 000	-2094	2094
10 000	-1928	1928
100 000	-1384	1384
Nombre de victoires		
1 000	50 433	49 567
5 000	49 536	50 464
10 000	49 813	50 187
100 000	49 887	50 113
Temps d'exécution en secondes		
1 000	0,5	0,8
5 000	2,5	3,9
10 000	4,9	7,8
100 000	49,7	79,1

Figure 2: Comparaison lightCRM VS CRM.

est le cas, il nous sera plus intéressant de garder l'algorithme lightCRM pour l'implémenter sur le Poker Limit Texas Hold'em car il est beaucoup moins gourmand en calcul et donc plus rapide à faire tourner.

Nous avons fait jouer les algorithmes CRM et lightCRM l'un contre l'autre (en choisissant le premier joueur aléatoirement) afin de comparer leur performance. Les résultats (Figure 2) nous montre qu'autant en nombre de victoires, qu'en payoff, les deux algorithmes obtiennent des résultats très similaires (il n'y a pas un algorithme qui se fait battre à plat de couture par l'autre).

En revanche, en ce qui concerne le temps d'exécution de l'entraînement, le lightCRM est bien plus rapide que le CRM (presque 2 fois plus rapide). Pour le Poker Kuhn, ce point peut sembler insignifiant car les temps d'exécution sont très courts, mais il sera bloquant pour le Poker Limit Texas Hold'em car étant donné le grand nombre de combinaisons possibles (même si réduites sous formes de clusters, comme nous le verrons plus tard), les temps d'exécution exploseront.

De plus, nous pouvons voir (Figure 3) que l'équilibre de Nash attendu est atteint au bout de 100 000 itérations par l'algorithme lightCRM. Cela nous conforte dans notre décision de choisir cet algorithme plutôt que le CRM, car il renvoie des équilibres de Nash similaires au CRM et de manière plus rapide.

```
{'.': {'JK': 0.1666666666664488,
       'JQ': 0.1666666666664488,
       'KJ': 0.1666666666664488,
       'KQ': 0.1666666666664488,
       'QJ': 0.1666666666664488,
       'QK': 0.1666666666664488},
 '.J.': {'BET': 0.12136186425855044, 'CHECK': 0.8786381357414597},
 '.J.BET': {'CALL': 5e-06, 'FOLD': 0.999995},
 '.J.CHECK': {'BET': 0.3293175459742382, 'CHECK': 0.6706824540257631},
 '.J.CHECK.BET': {'CALL': 0.0, 'FOLD': 1.0},
 '.K.': {'BET': 0.3721071188651892, 'CHECK': 0.6278928811347742},
 '.K.BET': {'CALL': 0.99998, 'FOLD': 2e-05},
 '.K.CHECK': {'BET': 0.99997, 'CHECK': 3e-05},
 '.K.CHECK.BET': {'CALL': 0.999995, 'FOLD': 5e-06},
 '.Q.': {'BET': 8.99135293238099e-05, 'CHECK': 0.9999100864706761},
 '.Q.BET': {'CALL': 0.33591270190409, 'FOLD': 0.6640872980959146},
 '.Q.CHECK': {'BET': 0.0001425, 'CHECK': 0.9998575},
 '.Q.CHECK.BET': {'CALL': 0.45750727927340007, 'FOLD': 0.542492720726596 3}}
```

Figure 3: Equilibre de Nash lightCRM.

lightCRM VS. Random (100 000 parties)		
Nombre d'itérations pour l'entraînement de l'IA	**lightCRM**	**Random**
Payoff		
1 000	42 734	-42 734
5 000	10 797	-10 797
10 000	29 544	-29 544
100 000	36 439	-36 439
Nombre de victoires		
1 000	49 177	50 823
5 000	47 212	52 788
10 000	52 508	47 492
100 000	52 378	47 622

Figure 4: Comparaison lightCRM VS Random.

3.4.2 Comparaison entre lightCRM et Random

Une fois que nous avons sélectionné l'algorithme lightCRM, nous voulons voir si les équilibres de Nash obtenus permettront réellement à l'IA de gagner face à une stratégie aléatoire.

Pour ce faire, nous avons fait jouer la stratégie lightCRM contre une stratégie Random, et les résultats obtenus (Figure 4) sont sans appel : si le nombre de victoires entre les deux stratégies sont relativement similaires, les payoffs sont toujours positifs pour la stratégie lightCRM (et inversement, ils sont toujours négatifs pour la stratégie Random).

On en conclut donc que la stratégie lightCRM est meilleure qu'une stratégie Random.

3.4.3 Comparaison entre lightCRM et OptEspGain

Nous avons cherché à comparer l'algorithme lightCRM avec une simple stratégie où les décisions sont prises pour maximiser l'espérance de gain. Nous observons que la stratégie OptEspGain bat à plat de couture la stratégie lightCRM (Figure 5).

lightCRM VS. OptEspGain (100 000 parties)		
Nombre d'itérations pour l'entraînement de l'IA	lightCRM	OptEspGain
Payoff		
1 000	-24 680	24 680
5 000	-31 631	31 631
10 000	-28 896	28 896
100 000	-31 424	31 424
Nombre de victoires		
1 000	42 474	57 526
5 000	40 285	59 715
10 000	41 550	58 450
100 000	40 594	59 406

Figure 5: Comparaison lightCRM VS OptEspGain.

OptEspGain VS. Random (100 000 parties)		
	OptEspGain	Random
avec seulement la carte Roi pour OptEspGain		
Payoff	365 370	-365 370
Nombre de victoires	100 000	0
avec seulement la carte Reine pour OptEspGain		
Payoff	90 862	-90 862
Nombre de victoires	66 000	34 000
avec seulement la carte Valet pour OptEspGain		
Payoff	**-115 192**	**115 192**
Nombre de victoires	35 958	64 042

Figure 6: Comparaison OptEspGain VS Random (avec carte de OptEspGain fixée).

Ce résultat nous ayant étonné, nous avons décidé d'approfondir notre étude en comparant le comportement des deux IA si elles disposent d'une carte forte (roi), moyenne (reine) ou faible (valet) contre une stratégie aléatoire (Figure 6 et 7). A nouveau, c'est OptEspGain qui affiche de meilleurs résultats. Cela s'explique par le fait qu'un équilibre de Nash, par définition, est la stratégie telle qu'aucun joueur ne regrette son choix. Or elle suppose que l'autre joueur joue de manière optimale (stratégie forte), et donc il ne fonctionne pas bien contre une stratégie faible, comme le Random. C'est pourquoi la stratégie OptEspGain renvoie des meilleurs résultats que la stratégie jouant selon l'équilibre de Nash obtenu avec lightCRM.

3.5 Conclusion

Pour conclure, même si OptEspGain semble une meilleure stratégie que lightCRM, nous pensons que ce ne sera pas le cas sur le Poker Limit Texas Hold'em car le jeu est beaucoup plus complexe que le Poker Kuhn. En effet, cette stratégie est beaucoup trop prévisible car elle ne consiste qu'à bet/call tout le temps. De ce fait, nous avons éprouvé plus de difficulté à gagner contre lightCRM que contre OptEspGain car lightCRM est plus imprévisible.

lightCRM VS. Random (100 000 parties)		
	lightCRM	Random
avec seulement la carte Roi pour lightCRM		
Payoff	342 363	-342 363
Nombre de victoires	100 000	0
avec seulement la carte Reine pour lightCRM		
Payoff	-36 804	36 804
Nombre de victoires	40 063	59 937
avec seulement la carte Valet pour lightCRM		
Payoff	-213 887	213 887
Nombre de victoires	8 181	91 819

Figure 7: Comparaison lightCRM VS Random (avec carte de lightCRM fixée).

4 Poker Limit Texas Hold'em à deux joueurs

Une fois que nous avons correctement implémenté l'algorithme CRM sur le jeu simplifié du Poker Kuhn, nous pouvons désormais l'appliquer sur le Poker Limit Texas Hold'em. En revanche, l'implémentation nécessite un autre travail préliminaire portant sur la réduction des combinaisons de mains au travers de méthodes de clustering.

4.1 Règles du jeu

Dans le jeu du Poker Limit Texas Hold'em à deux joueurs, le paquet comporte 52 cartes : 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A pour chacune des quatre couleurs (coeur, carreau, trèfle, pique). Chaque joueur doit miser 1 (la blinde) pour entrer dans le jeu. Le jeu se déroule ensuite de la façon suivante :

- **Préflop** : Une paire de cartes est distribuée aléatoirement à chaque joueur parmi les 52 cartes et 5 cartes (la rivière) face cachée sont posées au milieu de la table. Les joueurs prennent des décisions comme dans le Poker Kuhn. Si un des joueurs se couche, le jeu s'arrête, sinon il poursuit avec le flop.

- **Flop** : Trois cartes de la rivière sont dévoilées. Les joueurs prennent à nouveau

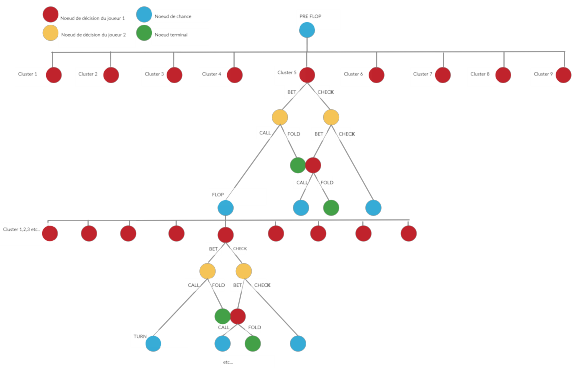

Figure 8: Arbre du Poker Limit Texas Hold'em avec clusters.

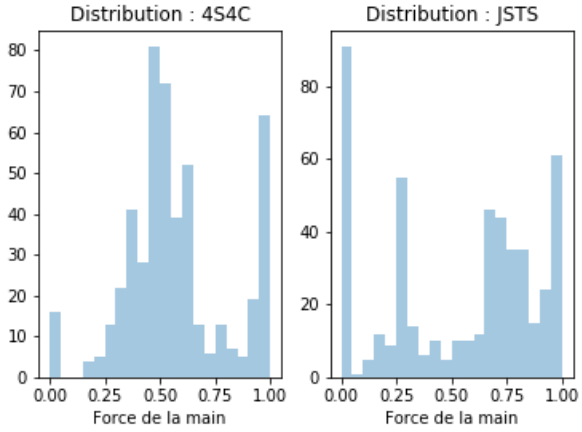

Figure 9: Histogramme des mains 4S4C et JSTS

des décisions. Si un des joueurs se couche, le jeu s'arrête, sinon il poursuit avec le turn.

- **Turn** : La quatrième carte de la rivière est dévoilée. Les joueurs prennent à nouveau des décisions. Si un des joueurs se couche, le jeu s'arrête, sinon il poursuit avec la river.

- **River** : La dernière carte de la rivière est dévoilée. Les joueurs prennent leurs dernières décisions. Si un des joueurs se couche, le jeu s'arrête, sinon les cartes sont dévoilées et le gagnant est ainsi déterminé.

4.2 Implémentation

L'implémentation de l'arbre du Poker Limit Texas Hold'em à deux joueurs (Figure 8) est plus compliqué que celui du Poker Kuhn au vu du grand nombre de combinaisons de cartes possibles.
La principale différence avec l'arbre du Poker Kuhn est que l'arbre du Poker Limit Texas Hold'em possède plusieurs noeuds chance : au preflop, au flop, au turn et à la river. A chaque noeud chance, le nombre d'enfants est exponentiel. Cela rend donc impossible l'implémentation de l'arbre avec les capacités de calcul dont nous disposons.
Pour pouvoir l'implémenter, il est nécessaire au préalable de réduire le nombre de combinaisons possibles à chaque noeud chance (preflop, flop, turn et river) par des méthodes de clustering.

4.3 Clustering

Dans cette partie nous allons déterminer des clusters de combinaisons de cartes similaires. L'idée ici n'est pas de regrouper des combinaisons de cartes ayant des espérance de gain similaire, mais pouvant s'améliorer de la même manière. Ce travail est à effectuer au preflop, au flop, au turn et à la river.
Pour le preflop, nous avons réalisé les histogrammes de la manière suivante :

- Pour chacune des paires de cartes possibles du premier joueur, nous simulons 500 fois les 5 cartes de la rivière, et pour chacune de ces 500 simulations nous simulons 500 fois les cartes du deuxième joueur.

- Nous calculons pour chacune des paires de cartes du premier joueur, le nombre de fois qu'il a gagné.

- Nous traçons les histogrammes de ces victoires.

La Figure 9 représente les histogrammes de deux paires (4S4C et JSTS). En moyenne ces deux paires ont la même espérance de gain (environ 57%) mais nous observons des distributions totalement différentes. Comme nous pouvons l'observer la paire 4S4C gagne souvent quelque soit la rivière (peu de chance d'amélioration), tandis que la paire JSTS gagne seulement avec certaines rivières (grande chance d'amélioration).

Nous construisons donc nos clusters à partir de ces histogrammes et non des espérance de gain. Il est possible de comparer deux distributions avec la distance de Wasserstein. Les clusters sont construits en minimisant cette distance entre les histogrammes grâce à l'algorithme des k-means.

4.3.1 Réduction des combinaisons

Si nous appliquons ce travail sur l'ensemble des combinaisons possibles, nous observons que les

Sans réduction des combinaisons				
	Nombre de combinaisons possibles du J1	Nombre de simulation de la rivière	Nombre de simulation des cartes du J2	Nombre de simulations total
pre flop	1 326	500	500	331 500 000
flop	2 598 960	500	500	649 740 000 000
turn	20 358 520	500	500	5 089 630 000 000
river	133 784 560		500	66 892 280 000
Temps de calcul sans parallélisation				
0,000124 seconde par simulation	Secondes	Minutes	Heures	Jours
pre flop	41 106	685	11,4	0,5
flop	80 567 760	1 342 796	22 380	932
turn	631 114 120	10 518 569	175 309	7 305
river	8 294 643	138 244	2 304	96
Temps de calcul avec parallélisation (6 cœurs physiques)				
9,4675E-06 seconde par combinaison	Secondes	Minutes	Heures	Jours
pre flop	3 138	52	0,9	0
flop	6 151 413	102 524	1 709	28,5
turn	48 186 072	803 101	13 385	223
river	632 584	10 543	176	2,9

Figure 10: Temps d'exécution sans réduction des combinaisons de mains.

Avec réduction des combinaisons				
	Nombre de combinaisons possibles du J1	Nombre de simulation de la rivière	Nombre de simulation des cartes du J2	Nombre de simulations total
pre flop	169	500	500	42 250 000
flop	331 682	500	500	82 920 500 000
turn				
river				
Temps de calcul sans parallélisation				
0,000124 seconde par simulation	Secondes	Minutes	Heures	Jours
pre flop	5 239	87	1,5	0,1
flop	10 282 142	171 369	2 856	119
turn				
river				
Temps de calcul avec parallélisation (6 cœurs physiques)				
9,4675E-06 seconde par combinaison	Secondes	Minutes	Heures	Jours
pre flop	400	7	0	0
flop	785 050	13 084	218	3,6
turn				
river				

Figure 11: Temps d'exécution avec réduction des combinaisons de mains.

temps de calcul (Figure 10) sont irréels (par exemple, pour le turn il faudrait 7305 jours pour calculer les clusters...). Une des solutions possibles pour réduire le temps de calcul est d'effectuer du multiprocessing[3]. Or nous voyons que même si les temps de calculs décroissent largement (pour la turn on passe de 7305 jours à 223 jours avec un processeur 6 coeurs), ils restent non-implémentables pour les machines que nous possédons.
Une autre solution pour réduire le temps de calcul est de réduire le nombre de combinaisons possibles. Par exemple pour le preflop, nous avons remarqué qu'il était inutile de considérer l'intégralité des combinaisons de paires, il suffit de conserver tous les couples de valeurs possibles avec une même couleur et une couleur différente. L'exercice réalisé sur le flop est sensiblement le même. Par contre, pour le turn et la river la tâche est plus compliquée. Nous n'avons donc pas pu établir une réduction des combinaisons robuste. Dans la Figure 11, nous voyons que les temps d'exécution avec réduction des combinaisons et parallélisation sont réalistes. Pour le flop, nous passons de 28,5 jours de calcul à 3,6 jours.

4.3.2 Résultats

Après avoir fait tourner l'algorithme des k-means, nous obtenons le dendogramme de la Figure 12. Le choix du nombre de clusters étant arbitraire, nous choisissons une distance maximale égale à 1.8, afin d'obtenir au maximum 9 clusters. La matrice représentant les clusters (Figure 13) est en

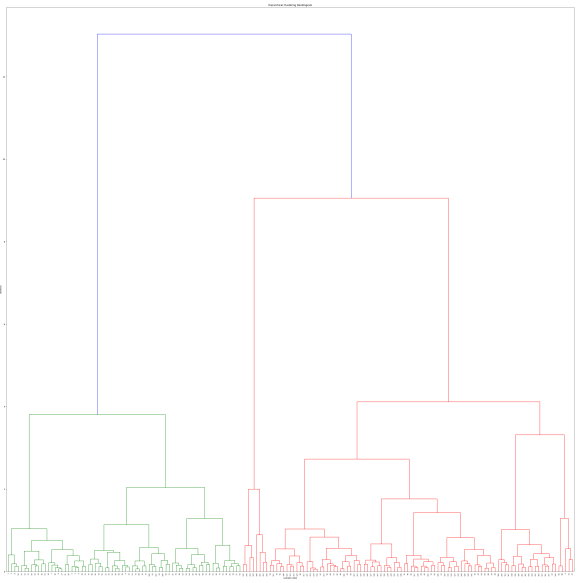

Figure 12: Dendogramme du preflop.

fait la somme d'une matrice triangulaire inférieure qui représente les clusters de paires de couleurs différentes, et une matrice triangulaire supérieure qui représente les clusters de paires de mêmes couleurs.
Selon nous les clusters créés semblent cohérents et font sens de par notre expérience de jeu. Ces clusters créés par un processus mathématique sont les mêmes clusters que nous aurions créé si nous avions du déterminer des clusters manuellement.

Malgré le fait que nous ayons réduit le nombres de combinaisons et donc le temps de calcul, nous n'avons pas pu implémenter les clusters du flop. En effet, le temps de calcul reste encore important pour une machine locale comme la nôtre. Mais en considérant les différents services de cloud comme AWS ou Google, nous aurions pu largement mener

[3] Attention à ne pas confondre avec le multi-threading qui consiste à utiliser un seul coeur et plusieurs thread en parallèle. Le multiproccessing consiste à utiliser de manière parallèle tous les coeurs du processeur.

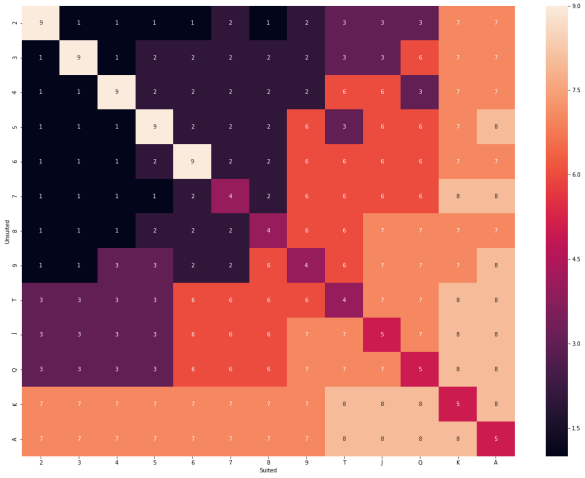

Figure 13: Clusters du preflop.

à bien notre calcul de clusters.
Pour l'instant nous n'avons pas trouvé de solution viable pour les clusters du turn et de la river.

5 Discussion/Conclusion

L'algorithme Counterfactual Regret Minimization est un framework puissant permettant de résoudre des jeux à information imparfaite, tel que le Poker. Il converge vers un équilibre de Nash en parcourant l'arbre du jeu de manière itérative.
Appliqué au Poker Kuhn, le CRM présente des résultats satisfaisants et cela pour un temps de calcul moindre. Il aurait été intéressant de pouvoir l'appliquer au Poker Limit Texas Hold'em, dont l'arbre de jeu est beaucoup plus complexe. Au préalable, il est nécessaire de s'intéresser à des méthodes de clustering et de réduction des temps de calculs, comme nous l'avons fait. Malheureusement nous n'avons pas pu passer à l'étape suivante, qui est l'implémentation de l'arbre et l'application de l'algorithme CRM. Néanmoins, nous avons implémenté toutes les fonctions nécessaires pour faire fonctionner le jeu. Il nous a malheureusement manqué du temps et de puissance de calcul pour assembler l'intégralité des pièces du puzzle.