Benchmark Jeu de Dames

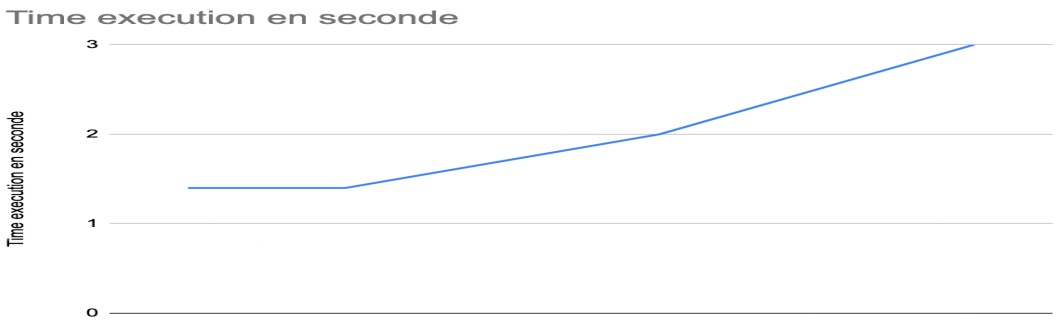
I.Créer un script de test pour comparer le modèle minimax avec le random (différentes profondeurs)

Nous allons comparer le modèle minimax avec le modèle random avec différentes profondeurs.
Par soucis de temps d'exécution nous allons faire le benchmark sur un jeu de 6x6 avec qu'une ligne de pions pour chaque côté.

profondeur	Minimax	Random	Nulle	**Total**	Time execution en seconde
1	95	4	1	100	1.4
2	100	0	0	100	1.4
3	95	5	0	100	1.7
4	99	1	0	100	2
5	98	1	1	100	2.5
6	99	1	0	100	3
Total	586	12	2	600	
Proportion	97.67%	2.00%	0.33%		

D'après 600 tests avec 6 profondeurs différentes, la proportion de victoire du modèle Minimax face au random est de 97%. La profondeur ne fait pas grandement varier la proportion de victoires, parce que l'écart moyen du nombre de victoires en moyenne selon la profondeur est de 2.

De plus le temps d'exécution des 100 parties joué ne varie pas énormément avec un écart moyen de 0.6 secondes , plus il y a de profondeur plus le temps d'exécution est long mais reste assez linéaire. Voici un graphique qui représente le temps d'exécution.



Sur ce graphique en bleu la courbe originale du temps d'exécution .Comme nous pouvons remarquer le temps d'exécution croît de plus en plus vite avec une courbe qui s'assimile au début à une courbe constante et ensuite à une courbe linéaire , l'algorithme qui est Minimax vs random avec une profondeur qui augmente linéairement est approximativement en grand O(n).

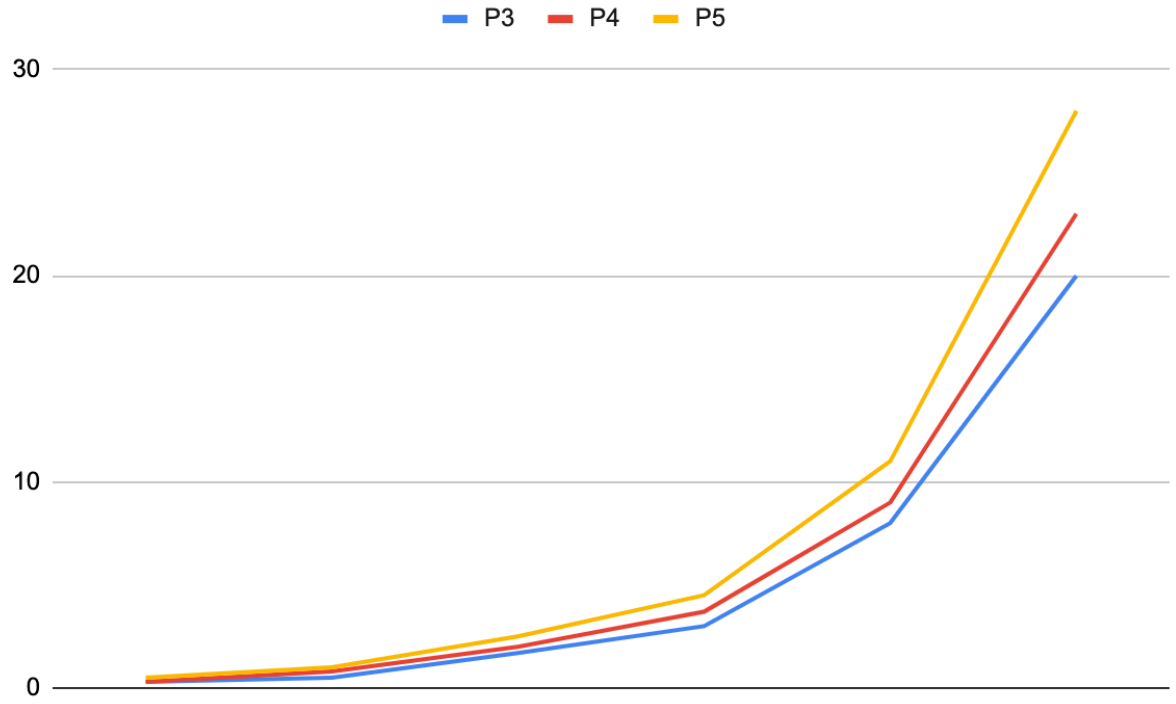
III.Créer un script de test pour comparer le modèle minimax avec lui même (différentes profondeurs)

II.Calculer le temps d'exécution moyen en fonction du nombre de pièces et de la profondeur de l'algo

Voici le temps d'exécution moyen en fonction du nombre de pièces , de la profondeur mais également de la taille du plateau.

100 test	Minimax vs random					
nb_piece\Profondeur	1	2	3	4	5	dimension
4	0.3	0.2	0.3	0.3	0.5	4x4
4	0.3	0.5	0.5	1	1	5x5
6	1.4	1.4	1.7	2	2.5	6x6
12	2	2.5	3	3.7	4.5	6x6
8	5	6	8	9	11	8x8
10	13	15	20	23	28	10x10
STD	4.89	5.66	7.61	8.69	10.55	

La profondeur maximale appliquée est de 5, et comme nous l'avons remarqué auparavant le temps d'exécution est assez linéaire quand seul la variation de la profondeur varie, en effet l'écart moyen est de 1.7 secondes.
Cependant quand le nombre de pièces change l'écart moyen entre les différentes dimensions et nombre de pièces est de 8 secondes en moyenne.



Le graphique ci-dessus nous montre les différentes évolution de temps d'exécution avec une profondeur fixe mais un nombre de pièce et dimension qui varie (analyse verticale de la table), a la différence de la variation de temps d'exécution avec un nombre de pièce fixe (analyse horizontale de la table), le graphique ci dessus nous montre une variation exponentielle.

III.Implémenter l'algorithme de Monte Carlo, Faire différents tests de performances

L'algorithme de Monte Carlo a pour but de faire des mouvements aléatoires et ensuite avec ces mouvements aléatoires voir si elle implique une victoire. Si un mouvement aléatoire implique une victoire alors l'utilisation de ce mouvement sera plus conseillé qu'un autre. La démarche est de faire des coups random, si ce mouvement implique une victoire, alors son score va augmenter, à la fin nous aurons une effectué tous meilleurs coups avec la meilleur probabilité de victoire. Voici une table résumant un duel Monte-carlo classique vs Random avec des profondeur comprise [1,8].

profondeur	Montecarlo/random	Random	Nulle	**Total**	Time execution en seconde
1	20	0	0	20	22
2	20	0	0	20	25
3	20	0	0	20	37
4	20	0	0	20	40
5	20	0	0	20	50
6	20	0	0	20	70
7	20	0	0	20	63
8	20	0	0	20	81
Total					388
Proportion	100.00%	0.00%	0		

Voici le résultat de 20 parties avec le monte carlo random, contre un algorithme random, 100% de win rate. On remarque que le temps d'exécution augmente en fonction de la profondeur mais pas de manière exponentielle.Voici une représentation graphique du temps d'exécution.

L'écart moyen de temps d'exécution est de 18 secondes.