TP1 IA

ABDELMOUMENE Djahid

October 25, 2019

1 Fonction d'evaluation

La fonction d'evaluation depend de deux critéres. Le premier c'est la difference des sommes des valeurs des pions de joueur et l'adversaire. Cette heuristique capture le fait qu'on a plus de chance de gagner si on a plus des pions avec des valeur grandes.

On prend N comme la nombre des lignes, M nombre des colonnes et $Pions$ l'ensemble des pions, où pour chaque pion on peux récupérer la couleur (1 ou -1), la valeur et les indices dans le plateau x et y.

$$valDiff(Pions) = \sum_{p \in Pions} p.col * p.val$$

Et le deuxiéme c'est la difference des sommes des distances inversés (*ie*: $N - dist$) vers la ligne de fond. La valeur de la distance est inversé parce qu'on veux que l'evaluation soit grande quand les pions sont proche de fond et petite lorsque les pions sont loin. Cette heuristique encourage les pions a se rapprocher vers les pions de l'adversaire et vers la ligne de fond ou on peux gagner.

$$indiceFond(couleur) = \begin{cases} N-1 & couleur = -1 = \mathrm{o} \\ 0 & couleur = 1 = \mathrm{x} \end{cases}$$

$$distDiff(Pions) = \sum_{p \in Pions} p.col * (N - |p.x - indiceFond(p.col)|)$$

Pour combiner ces deux critéres on choisit un facteur λ pour multiplier $valDiff$ et on fait la somme, cette valeur doit indiquer le facteur d'importance de la $valDiff$ de la $distDiff$. C'est à dire qu'on veux prioriser l'attack des pion de l'avancement si $\lambda > 1$.

Alors la fonction d'evaluation:

$$H(Pions, joueur) = joueur * (valDiff(Pions) * \lambda + distDiff(Pions))$$

2 Complexité

On calcule la fonction de coût $c(p)$ où p est la profondeur maximale et F est le facteur de branchement - *ie* moyenne des nombre des bouges possible à chaque coup -, et cst un constant décrivant les conditions et operations unitaires pour effectuer le minimax.

$$c(p) = \begin{cases} N * M & p = 0 \\ F * c(p-1) + cst & p \geq 1 \end{cases}$$

Si on prend $u(p) = c(p) - \frac{cst}{1-F}$ alors:

$$\begin{aligned} u(p) &= \begin{cases} N * M - \frac{1}{1-F} & p = 0 \\ F * (c(p-1) - \frac{1}{1-F}) + cst & p \geq 1 \end{cases} \\ &= \begin{cases} N * M - \frac{cst}{1-F} & p = 0 \\ F * c(p-1) \end{cases} \end{aligned}$$

Alors U_p est un suite géometrique ou le terme génerale est:

$$u(p) = u(0) * F^p \qquad = (N * M - \frac{cst}{1-F}) * F^p$$

Alors on peux déduire $c(p)$:

$$\begin{aligned} c(p) &= u(p) + \frac{cst}{1-F} \\ &= (N * M - \frac{cst}{1-F}) * F^p + \frac{cst}{1-F} \end{aligned}$$

Alors la complexité est:

$$c(p) = \mathcal{O}(F^p)$$

Alors si on prend $F = 30$ - *estimation empirique* -

$$c(p) \approx 30^p \tag{1}$$

3 Analyse expérimentale

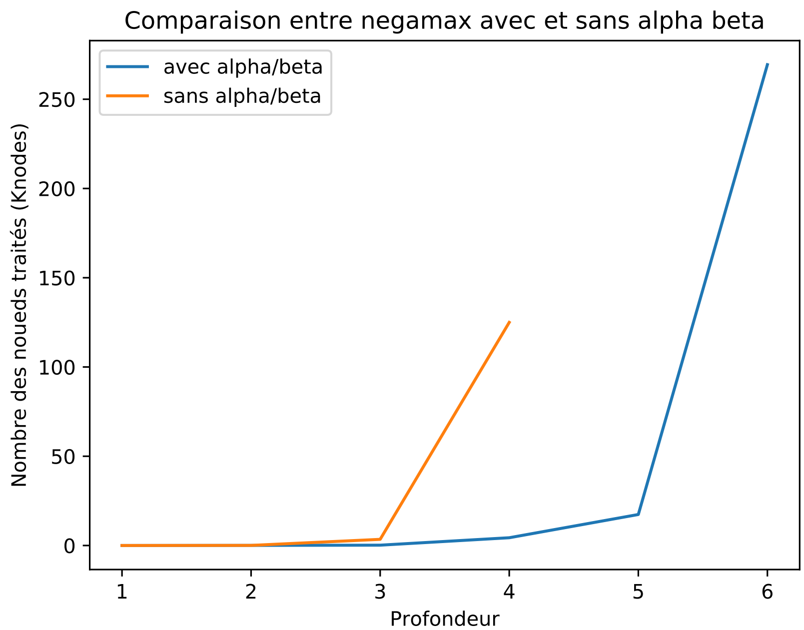

Figure 1: Comparaison entre negamax et negamax avec l'élagage α/β au niveau de nombre des noeuds traités

En théorie on doit avoir un gain moyenne de l'ordre de racine carrée pour la version avec l'élagage alpha-bêta:

$$\begin{aligned} c(p) &= \mathcal{O}(\sqrt{F^p}) \\ &= \mathcal{O}(F^{\frac{p}{2}}) \end{aligned}$$

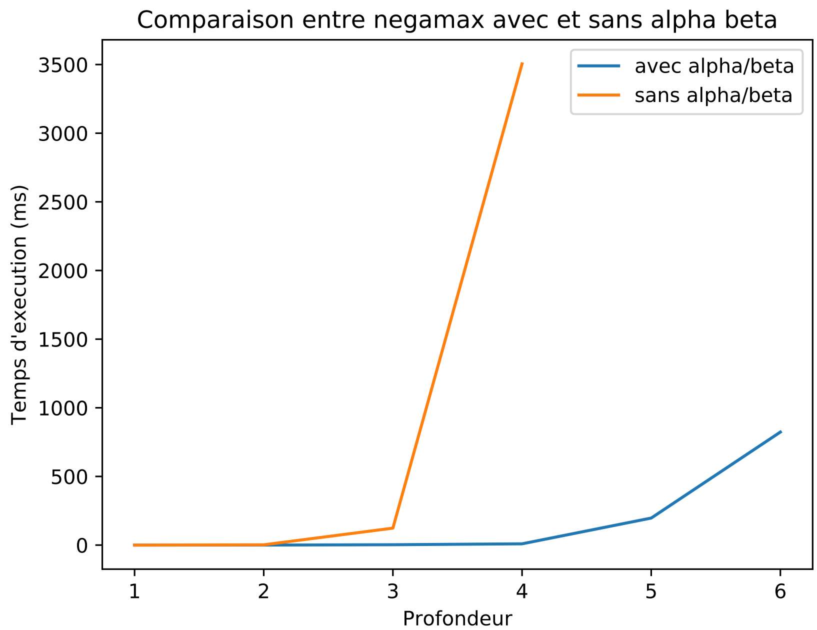

Figure 2: Comparaison entre negamax et negamax avec l'élagage α/β au niveau de temps d'éxecution

Pour maximiser le nombre de coupure avec l'algorithme α/β. On peut utiliser une technique de la programmation dynamique qui s'appelle la **Mémoïsation** ou on garde un tableau de mappage entre les noeuds et le résultat de l'appel minimax. et a chaque fois qu'on essaye de calculer le minimax d'un plateau, on cherche d'abord dans ce tableau si la noeud est déja stocké on récupére directement le résultat. Mais cette technique ne marchera pas toujours car elle consomme trop de memoire $\mathcal{O}(2^n)$ pour les jeux ayant un facteur de branchement grand.

Une autre optimisation c'est de trier les bouges possibles avant commencer, Ce tri dois prioritiser les bouges ayant plus de chance d'avoir un evaluation minimax grande, pour cela on peux utiliser l'heuristique existant pour tries ces bouges. Ca doit améliorer le performance puisque les bouges qui ont un bonne heuristique dés les premier coups devront au moyenne avoir un heuristique meilleur en generale, alors le nombre des coupures doit s'augmenter car l'interval $[\alpha, \beta]$ sera rognés plus rapidement.