Report LINFO1361: Assignment 2

Group N°052 (Moodle and Inginious)

Student1: Bette Jonas

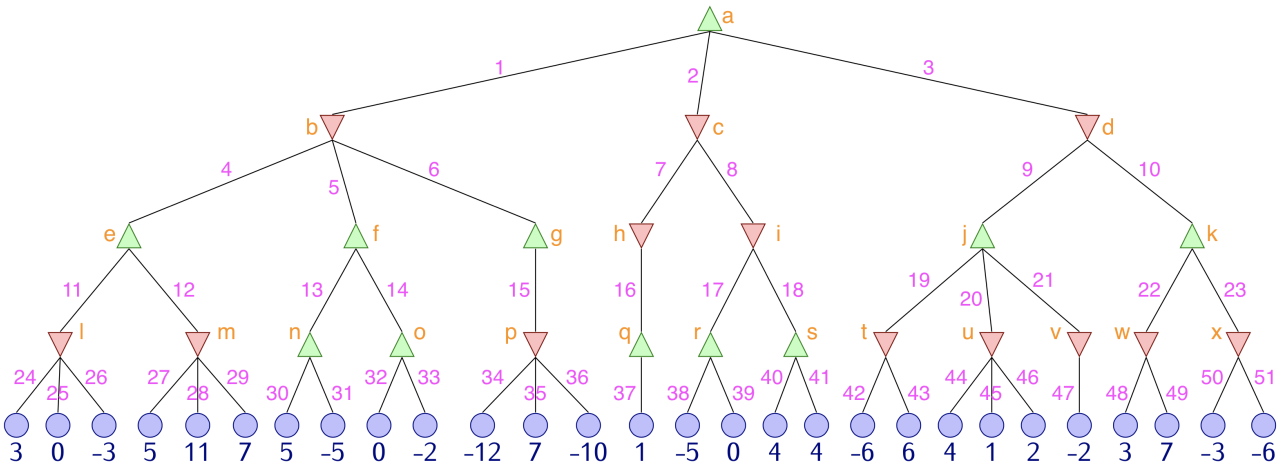
Student2: Huet Anatole

April 6, 2024

Answer to the questions by adding text in the boxes. You may answer in either **French or English**. Do not modify anything else in the template. The size of the boxes indicate the place you **can** use, but you **need not** use all of it (it is not an indication of any expected length of answer). **Be as concise as possible! A good short answer is better than a lot of nonsense!**

1 Exercises (5 pts)

The following figure assigns a unique letter to each node, and a unique number to each branch. Use it to answer the following questions.



1. Perform the MiniMax algorithm on the following tree, i.e. put a value to each node. What move should the root player do? **(1 pt)**

   Assign a numerical value to each node, and indicate the move (i.e. 1, 2, or 3) to perform:

a: 1	**f: 5**	**k: 3**	**p: -12**	**u: 1**
b: -12	**g: -12**	**l: -3**	**q: 1**	**v: -2**
c: 0	**h: 1**	**m: 5**	**r: 0**	**w: 3**
d: 1	**i: 0**	**n: 5**	**s: 4**	**x: -6**
e: 5	**j: 1**	**o: 0**	**t: -6**	**Move: 3**

2. Perform the Alpha-Beta algorithm on the same tree. At each non terminal node, put the successive values of α and β. Cross out the arcs reaching non visited nodes. Assume a left-to-right node expansion. **(1 pt)**

   Indicate the successive α and β values of each node in the table below. Separate successive values by a comma (,). Indicate at the bottom the identifiers of the branches that are cut (in increasing order, separated by a comma) (indicate only the branches where the cuts happen, i.e. don't *indicate the branches that are below a cut).*

Node	α values	β values
a	-∞, -12, 0, 1	+∞
b	-∞	+∞, 5
c	-12	+∞, 1, 0
d	0	+∞, 1
e	-∞, -3	+∞
f	-∞, 5	5
g	-∞	5
h	-12	+∞
i	-12	1
j	0, 1	+∞
k	/	/
l	-∞	+∞, 3, 0, -3
m	-3	+∞, 5

Node	α values	β values
n	-∞, 5	5
o	/	/
p	-∞	5, -12
q	-12	+∞
r	-12, -5, 0	1
s	-12	0, 4
t	0	+∞
u	0	+∞, 4, 1
v	1	+∞
w	0	1
x	/	/

Cuts: 14, 23, 31, 43

3. Do the same, assuming a right-to-left node expansion instead. **(1 pt)**

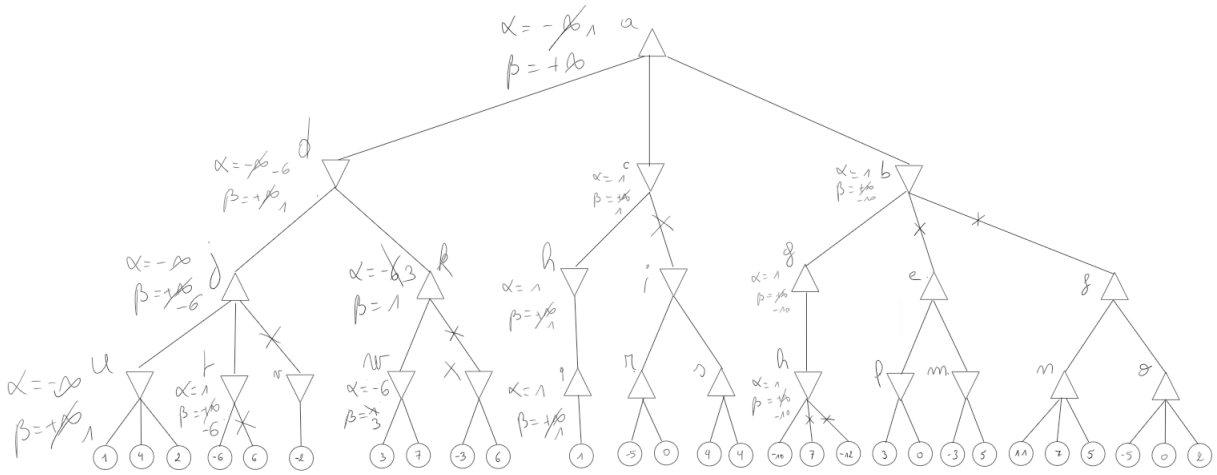
Node	α values	β values
a	-∞, 1	+∞
b	1	+∞
c	1	+∞
d	-∞	+∞, 3, 1
e	/	/
f	/	/
g	1	+∞
h	/	/
i	1	+∞, 4
j	-∞, -2, 1	3
k	-∞, -6, 3	+∞
l	/	/
m	/	/

Node	α values	β values
n	/	/
o	/	/
p	1	+∞
q	/	/
r	1	+∞
s	1	4
t	1	3
u	-2	3, 2, 1
v	-∞	3, -2
w	-6	+∞, 7, 3
x	-∞	+∞, 6

Cuts: 4, 5, 7, 34, 35

4. Is there a node ordering that can lead to a more optimal pruning of the tree (in the sense where the algorithm prunes more branches than in the two other considered cases)? If no, explain why. If yes, give a new node ordering and the resulting new pruning. **(1 pt)**

   Insert an image below containing the reordered tree, with successive α/β values indicated next to each node, and where the branches that are cut by the algorithm are crossed out. This may either be an edited version of `minimax_empty.png` *(using paint, gimp, etc.), a photograph of a drawing you made by hand, etc. In any case, the image must be* ***clear*** *in order to be graded.*



5. How does Alpha-Beta need to be modified for games with more than two players? **(1 pt)**

Plutôt que de comparer les valeurs α et β pour chaque joueur, il faut imaginer que chacun des joueurs est un joueur *max* et que, de ce fait, chacun cherche à maximiser son score. Chaque joueur vont assumer que leurs adversaires cherchent à maximiser leur score aussi et donc jouer les meilleurs coups possibles.
L'arbre de recherche utilisé sera plus ou moins identique à celui d'Alpha-Beta où chaque noeud représente un état du jeu. On va donc devoir évaluer les noeuds en fonction des joueurs et non plus en fonction d'un seul joueur.
On a les noeuds finaux qui sont des tuples avex n valeurs, n étant le nombre de joueurs. Le noeud parent indique à quel joueur c'est le tour de jouer. On peut donc prendre le noeud final qui maximise le score du joueur actuel.
Exemple : si on a deux noeuds finaux : (2, 8, 1) et (1, 7, 3) que c'est le tour du joueur 3, on prendra le tuple (1, 7, 2) parce que 3 est plus grand que 1. Si c'est le tour du joueur 1, on prendra le tuple (2, 8, 1) parce que 2 est plus grand que 1. Et si c'est le tour du joueur 2, on prendra le tuple (2, 8, 1) parce que 8 est plus grand que 7.
On peut aussi réduire l'arbre avec le pruning : si c'est au joueur i de jouer et que le ième élément du tuple de l'un de ses enfants est égale à la borne supérieure sur la somme de toutes les éléments du tuples. On peut alors couper l'arbre à ce niveau là.

2 Shobu (35 pts)

2.1 Alpha-Beta agent (4 points, to be submitted on Inginious)

2.2 Monte-Carlo Tree Search agent (5 points, to be submitted on Inginious)

2.3 Warm-up questions (3 points)

1. What is the branching factor at the start of the game? What is the mean empirical branching factor? (The branching factor is considered here as the number of possible moves that a player can do from a given state). **(1 point)**

Au début de la partien le nombre de coup possibles est au nombre de 116. Le facteur de branchement au début de la partie est de 116.
Au cours de la partie, dans le cas extrème où il n'y a que deux cailloux sur le plateau (un pour chaque joueur sinon ça sinifierait que la partie est finie),le nombre de coups possibles est de 9. Le facteur de branchement moyen est donc de 62,5.

2. What would be one (of the many) drawbacks of the simple heuristic that is imposed for the basic alpha-beta agent above? **(1 point)**

L'heuristique ne prennant en compte que le nombre de pions restants sur le plateau. Cela signifie que l'agent ignore la position des pions sur le plateau et donc il n'y a aucune stratégie de blocage ou de prise de pions adverses. Cela peut mener à des situations où l'agent ne voit pas qu'il peut gagner, protéger ses pions ou bloquer l'adversaire.
Lors de certaines parties, l'agent aurait pu gagner en un coup mais il a continué à jouer sans voir la possibilité de terminer la partie.

3. Considering the Monte-Carlo Tree Search algorithm, what is the hypothesis supporting the use of random simulation to estimate the win rate? Is this hypothesis always valid? **(1 point)**

L'idée derrière l'utilisation de la simulation aléatoire est qu'on peut obtenir une estimation raisonnable du taux de victoire.
En effet, cette approche s'appuie sur la loi des grands nombres stipulant que la moyenne des résultats obtenus à partir d'un grand nombre d'essais devrait être proche de la valeur attendue du taux de victoire.
Le soucis avec cette hypothèse est que si on a un trop petit nombre de simulation, le résultat ne reflètera psa forcément la réalité. On peut n'avoir eu que des simulations où agent adverse a joué de manière sub-optimale et donc notre agent peut prendre un peu trop la confiance et perdre quand l'adversaire jouera des coups qu'il n'avait pas prévu.
De plus, rien n'empêche de simuler plusieur fois une partie qui tourne en boucle, le résultat obtenu ne sera alors pas du tout efficace pour calculer le taux de victoire.

2.4 Description of your agent (8 points)

Describe in the boxes below what you have implemented for your contest agent. You can also mention things that you tried and did not work but focus on what you have submitted in the end.

2.5 Comparison of agents (5 points)

Describe here the comparison of the different agents: random, Alpha-Beta, MCTS and your agent (and others if you want!). Remember that it should be a statistical comparison. Describe how you compare the agents. Draw some observations and conclusions based on the results you have obtained.

2.6 Contest (10 points, to be submitted on Inginious)