EPL - ECOLE POLYTECHNIQUE DE LOUVAIN

LINGI2261 - ARTIFICIAL INTELLIGENCE

Report of third assignement
Quoridor

Professor :

Yves DEVILLE

Students : (Group 1)

Benoît BAUFAYS 22200900

Julien COLMONTS 41630800

Program :

SINF21MS/G

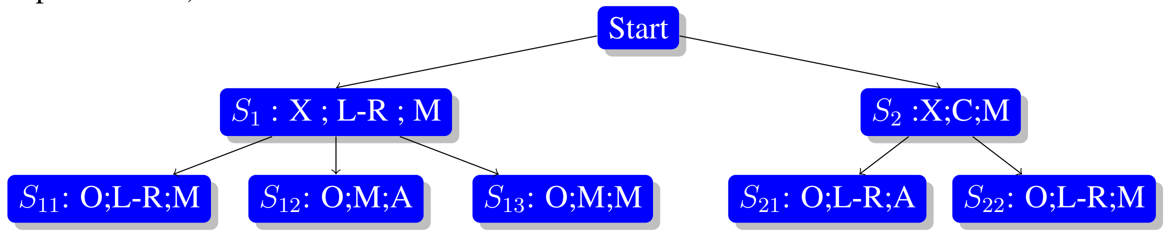
Academic year 2013-2014

Contents

1 Scipion

Question 1 : Draw the game tree for a depth of 2, i.e. one turn for each player. In the game tree, if a node has expanded symmetrical facts, you should only consider one of them. For instance, the first level has only two nodes.

We built the tree using triplets. First element means which player is going to make the action. The second one means which pawn on the board will do an action. Letters L and R mean left and right (states are grouped to avoid symmetrical states), letter M means middle. The third one shows which action is done. M stands for move action while A means attack action (or capture action).



Question 2 : Evaluate the value of the leaves using the heuristic function (on the figure you draw in question 1, you don't have to implement it!).

Values of the leaves for the tree shown above :

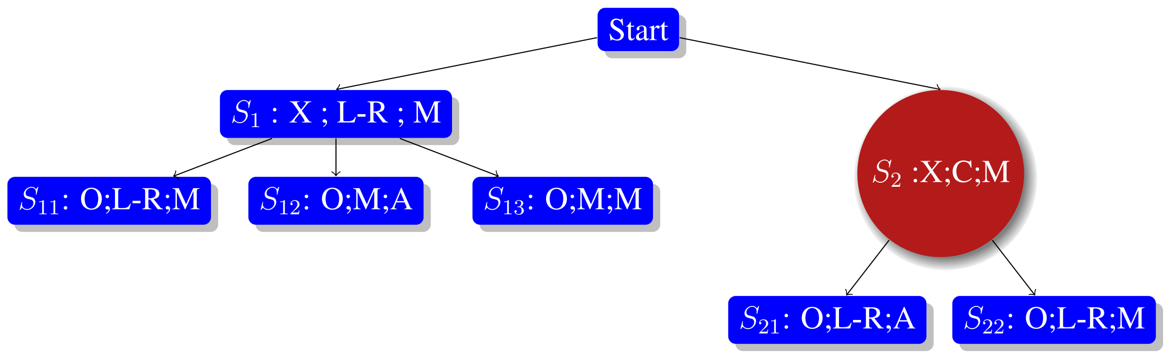
State :	S_{11}	S_{12}	S_{13}	S_{21}	S_{22}
Value :	0	-2	-4	-3	4

Question 3 : Using the MiniMax algorithm, find the value of the other nodes (no implementation needed, do it by hand).

Value of the other nodes :

State :	S_1	S_2
Value :	-4	-3

Question 4 : Circle the move the root player should do (i.e. circle the move on the first level of the game tree that the minimax algorithm would choose).



2 Alpha-Beta search

Question 1 : Perform the MiniMax algorithm on the tree in Figure 4, i.e. put a value to each node. Circle the move the root player should do.

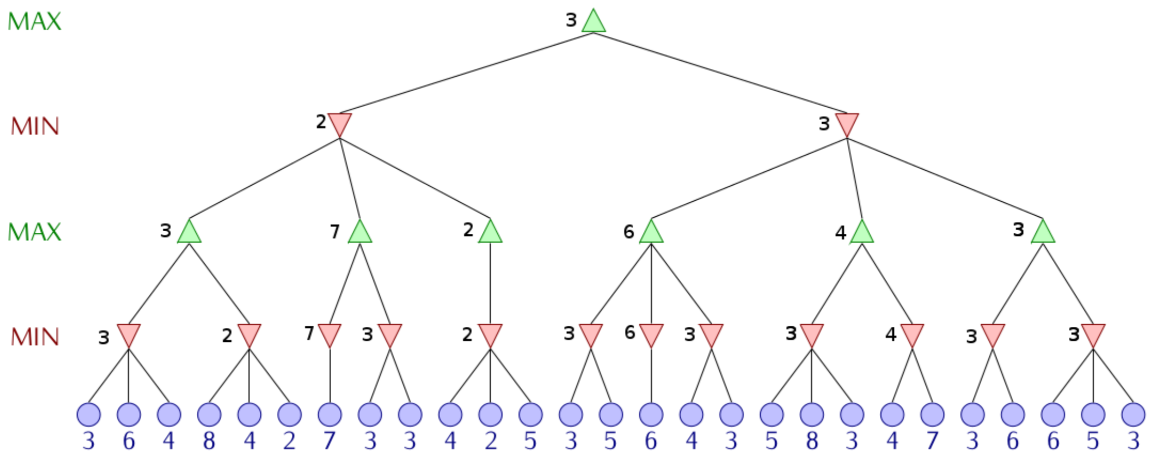
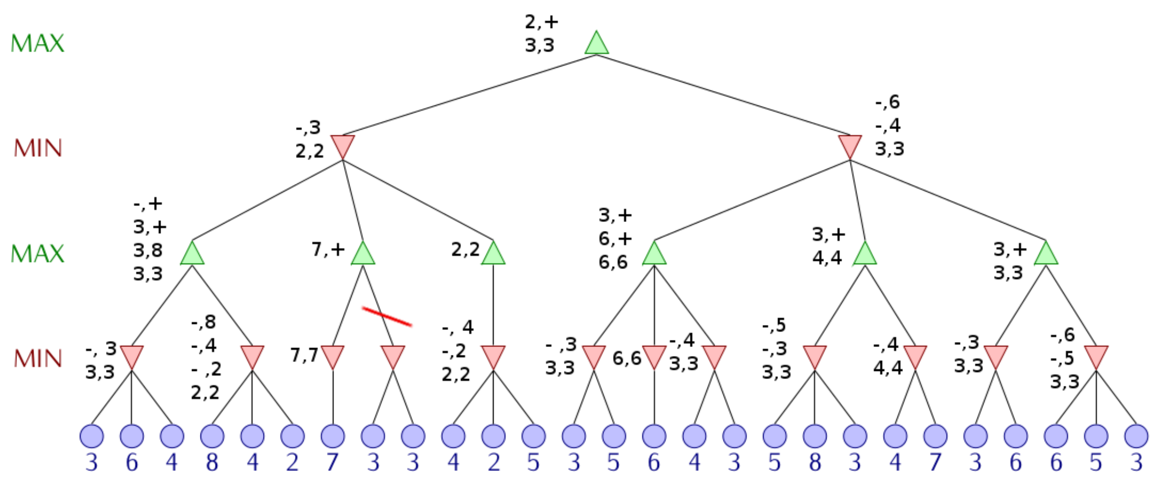
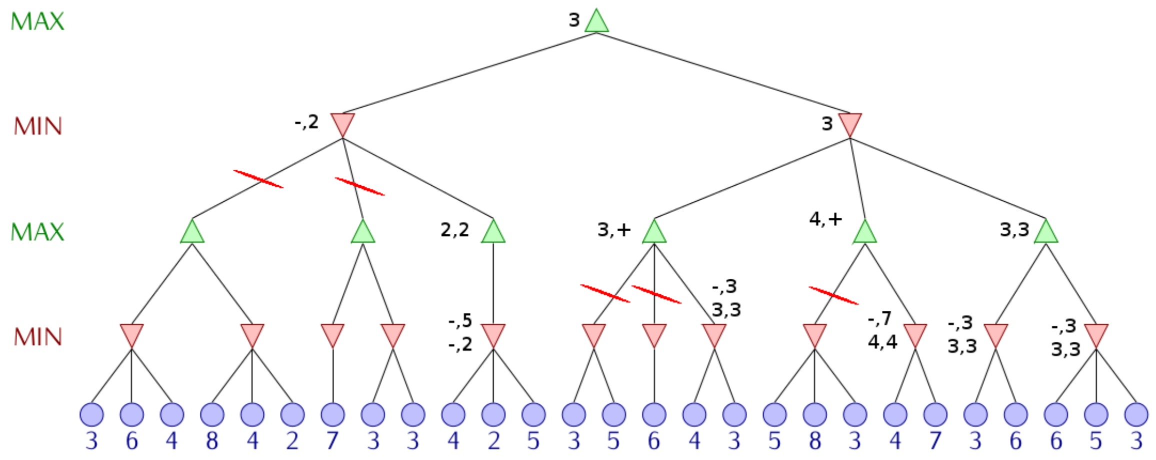


Figure 4: MiniMax

Question 2 : Perform the Alpha-Beta algorithm on the tree in Figure 5. 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.



Question 3 : Do the same, assuming a right-to-left node expansion instead (Figure 6).



Question 4 : Can the nodes be ordered in such a way that Alpha-Beta pruning can cut off more branches (in a left-to-right node expansion)? If no, explain why; if yes, give the new ordering and the resulting new pruning.

Yes. The nodes have to be ordered in an increasing way for MIN levels and in a decreasing way for MAX levels.

Build the new tree

3 Quoridor

3.1 A basic Alpha-Beta agent

3.2 Comparison of two evaluation functions

Question 4 : Launch two instances of your basic agent and make them play against each other. Use the replay options to have a look at the two first moves of both payers. What do you observe?

Blue player is going on the right direction. Red player is moving two same board positions.

Question 5 : Replace the basic evaluation function such that it returns directly the result of Board.get_score instead of -1, 0 or 1. Launch again two instances of your agent and make them play against each other. Does their respective behaviour radically change? What do you observe?

In this situation, Red player wins. Both players went to the right direction.

Question 6 : Explain why the respective behaviours of the instances of your agent changes when considering the two different evaluation function. To help you and illustrate your answer, draw and compare the search trees of depth 2 for the second move of the second player for both evaluation functions. Perform the MiniMax algorithm (not the alpha-beta) on these two trees, i.e. put a value to each node. Circle the move the root player should do.

Put tree here

3.3 Evaluation function

Question 7 : Describe your evaluation function.

In our evaluation function, we compute the cost to every goal tile with a dijkstra algorithm and we take the minimal one. We compute it for both player and opponent. A special value is also evaluated if player is on a goal (to force player to go on this tile first).

Question 8 : In Alpha-Beta, the evaluation function is used to evaluate leaf nodes (when the cut-

off occurs). As seen in previous questions, the pruning of Alpha-Beta depends on the order of the successors. Explain how your evaluation function could be used to (we hope) obtain more pruning with Alpha-Beta. Are there any drawbacks to your approach?
We could improve our evaluation function by sorting results at each depth of the tree in an increasing order for opponent playing depths and in a decreasing order for player playing depths. Even if it isn't the real value for the node, it could be a good approach to know what will probably happens in the next depths.

Question 9 : Make an agent using the successor function of the basic player (section 3.1), using your new evaluation function and cutting the tree at its root to use the evaluation function on its direct successors (you can achieve this by making cutoff always return True). Let this agent play against another similar agent using Board.get_score as evaluation function. Try out multiple matches and vary who plays first. How well does your evaluation function fare?

3.4 Successors functions

Question 10 : Give an upper and a lower bound on the branching factor for a search tree on the Quoridor game. Justify your answer.

Question 11 : How does the branching factor evolve after a pawn has moved? How does the branching factor evolve after a wall has been placed? Explain.

Question 12 : From random games (at least 100), compute the average number of possible actions at each step of the game (end the game after 100 steps). A step represents the turn of one player; i.e. if both player play one move each, two steps have been performed. Plot the results in a graph. What do you observe?

Question 13 : Are there special moves in the game after which your branching factor changes radically? What are these moves? Explain.

Question 14 : Are all these successors necessary to be exhaustive (think about symmetry)? Why? If not, how will you consider only the necessary states?

Question 15 : If the number of successors is still too large, can you think of states that might be ignored, at the expense of loosing completeness?

Question 16 : Describe your successors function.

Successors are computed in relation with current score (computed with get_score). If current player is losing, it tries to place a wall around opponent. Instead of trying all possible walls on the map (which would take a large computation time), our algorithm only tries walls at four places around opponent. There are pro and cons with this strategy : in some cases, it's really good because it waits the opponent to reach the end of a board side before putting walls, so he has to make the all way back. In some cases, it's bad because it could let the opponent going further before blocking him.

3.5 Cut-off function

Question 17 : The cutoff method receives an argument called depth . Explain precisely what is called the depth in the minimax.py implementation.

The depth is a number representing anticipated actions computed. A depth of one will only look at all possible actions for the next move of the player. A depth of two will also look at all corresponding actions of the opponent after the first move. The decision will so be more meaningful.

Question 18 : Explain why it might be useful (for the Quoridor contest) to cut off the search for another reason than the depth.

We can stop searching in the tree if one of the player won.

Question 19 : Describe your cut-off function.

In the cut-off function, we just check that game is not finished yet. Then we check that current depth is not deeper than maximum depth allowed. We compute depth in function of time and number of walls. Depth will increase till half time is spent than it will decrease. It starts at 3 and the maximum is 8. If current player has no more walls, depth is put to one because it only needs shortest path to a goal to play.