Kusterskiy Dmitriy
Lapin Andrew
Markeeva Larisa
Usvyatsov Mikhail1

Assignment 2

Exercises 6.1

his problem exercises the basic concepts of game playing, using tic-tac-toe (noughts and crosses) as an example. We define X , as the number of rows, columns, or diagonals with exactly n X's and no 0's. Similarly, 0, is the number of rows, columns, or diagonals with just n 0's. The utility function assigns +1 to any position with X g = 1 and -1 to any position with $O3 = 1$. All other terminal positions have utility 0. For nonterminal positions, we use a linear evaluation function defined as $Eva1(s) = 3X2(s)+X1(s)-(302(s)+O1(s))$.

a) Approximately how many possible games of tic-tac-toe are there?

b) Show the whole game tree starting from an empty board down to depth 2 (i.e., one X and one 0 on the board), taking symmetry into account.

c) Mark on your tree the evaluations of all the positions at depth 2.

d) Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0, and use those values to choose the best starting move.

e) Circle the nodes at depth 2 that would not be evaluated if alpha-beta pruning were applied, assuming the nodes are generated in the optimal order for alpha-beta pruning.

Solution

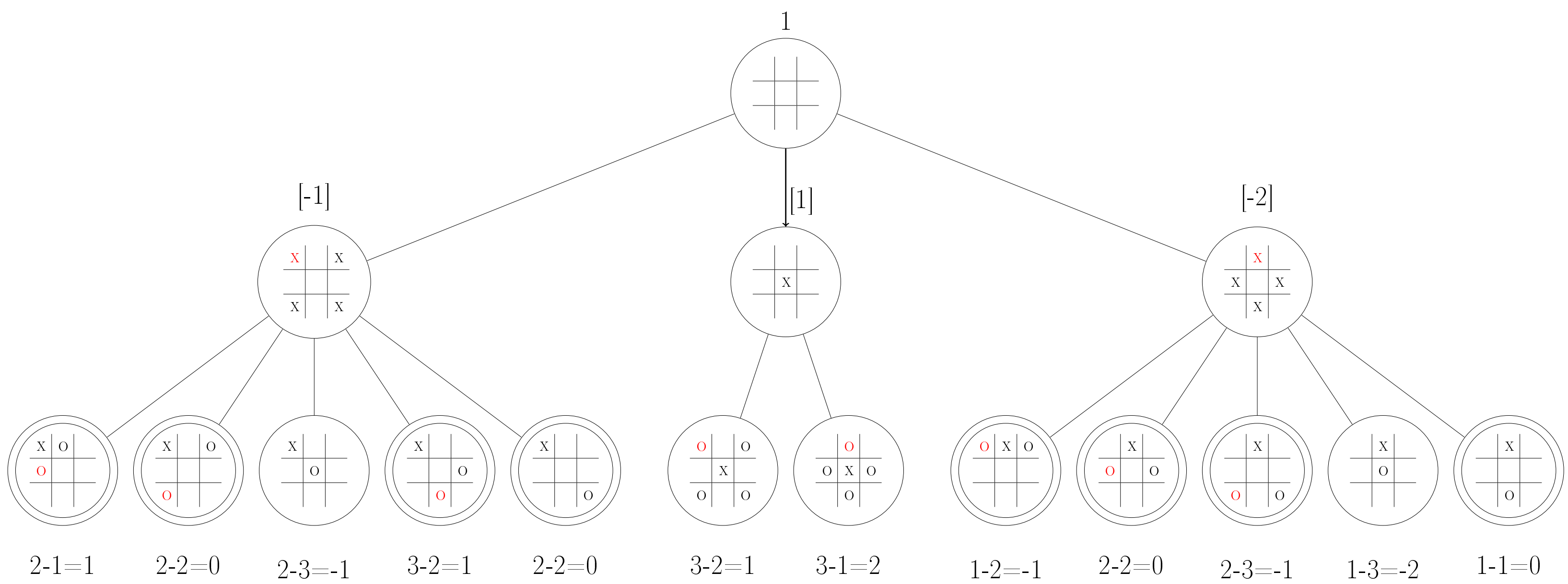
a) 9! = 362880

b)

c)

d)

e)



1
[-1]
[1]
[-2]
2-1=1
2-2=0
2-3=-1
3-2=1
2-2=0
3-2=1
3-1=2
1-2=-1
2-2=0
2-3=-1
1-3=-2
1-1=0

Exercises 6.5

Develop a formal proof of correctness for alpha-beta pruning. To do this, consider the situation shown in Figure 6.15. The question is whether to prune node n_j, which is a max-node and a descendant of node n_1. The basic idea is to prune it if and only if the minimax value of n_1 can be shown to be independent of the value of n_j.

a) The value of n_1 is given by

   $n_1 = max(n_2, n_{2_1}, \cdots, n_{2_{b2}})$

   Find a similar expression for n_2 and hence an expression for n_1 in terms of n_j.

b) Let l_i be the minimum (or maximum) value of the nodes to the left of node n_i at depth i, whose minimax value is already known. Similarly, let r.i be the minimum (or maximum) value of the unexplored nodes to the right of n_i at depth i. Rewrite your expression for n_l in terms of the l_i and r_i values.

c) Now reformulate the expression to show that in order to affect n_l, n_j must not exceed a certain bound derived from the l_i values.

d) Repeat the process for the case where n_j is a min-node.

Solution

a) $n_2 = max\left(n_3, n_{3_1}, ..., n_{3_{b3}}\right)$

   $n_1 = min\left(max\left(n_3, n_{3_1}, ..., n_{3_{b3}}\right), n_{2_1}, \cdots, n_{2_b 2}\right)$

   Then we replace n_3 the same way and so on recursively

b) $n_1 = min\left(l_2, max\left(l_3, n_3, r_3\right), r_2\right)$

   Then we again replace n_3 until we reach the step where $n_j - 1 = min\left(l_j, n_j, r_j\right)$

c) L nodes contains all the values that are less then n_j, n_2 node will only increase while going down. So it will be rejected. It is the thing what alpha-beta does.

d) The corresponding bound for min nodes $n_k = max\left(l_3, l_5, ..., l_k\right)$

Exercises 6.8 Consider the following procedure for choosing moves in games with chance nodes:

a) Generate some die-roll sequences (say, 50) down to a suitable depth (say, 8).

b) With known die rolls, the game tree becomes deterministic. For each die-roll sequence, solve the resulting deterministic game tree using alpha-beta.

c) Use the results to estimate the value of each move and to choose the best.

Will this procedure work well? Why (not)?

Solution

Minmax algorithm for nonzero sum will works similarly to minmax for zero sum games. But we will represent the weight of node as a vector. Alpha beta pruning here is impossible because there are states which weight we cannot compute(doesn't clear about their profit).

Exercises 6.15

Describe how the minimax and alpha-beta algorithms change for two-player, non-zero-sum games in which each player has his or her own utility function. You may assume that each player knows the other's utility function. If there are no constraints on the two terminal utilities, is it possible for any node to be pruned by alpha-beta?

Solution

The evaluation function is a vector of values, one for each player, and the backup step selects which ever vector has the highest value for the player whose turn it is to move. Alpha-beta pruning is not possible in general non-zero-sum games, because an unexamined leaf node might be optimal for both players