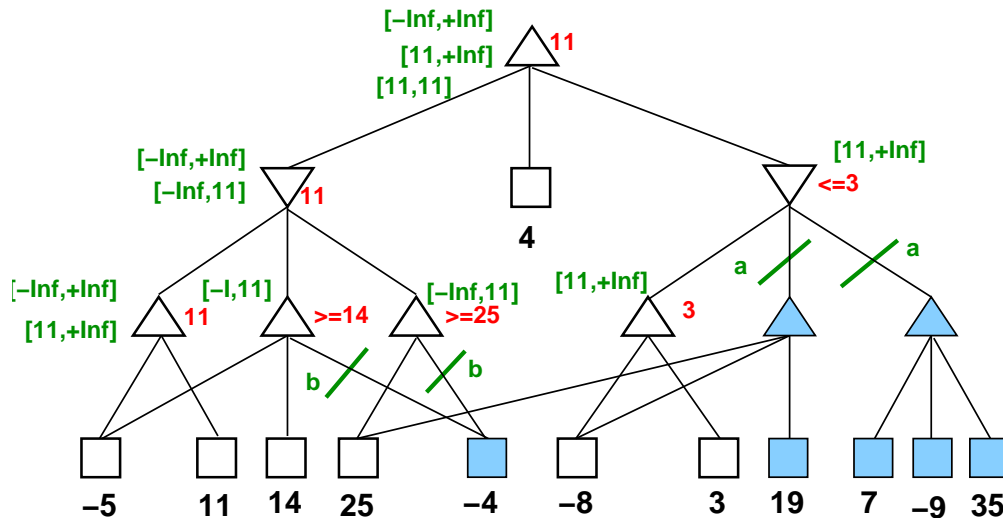


Homework 4

Due: Apr 17, 2003

- Run the minimax and alpha-beta game search algorithms on the following game graph, assuming a left-to-right expansion of children. \triangle nodes are played by MAX and ∇ are played by MIN. For each node, label it with its minimax value, the $[\alpha, \beta]$ bounds, and the optimal action from that state (for the appropriate player, of course). Indicate which arcs are pruned and why (i.e., whether it was pruned because of α or β). How many total nodes (including internal nodes) are pruned? What is the value of the game? (Hint: it might be worth your effort to make a couple of copies of this diagram as a worksheet before you do your final version. Also, colored pens are useful for marking all the different necessary information.)



- Show that if you replace every leaf payoff x in a two-player game tree with a linear transformation of that payoff $ax + b$, then the minimax optimal strategy remains the same.

This is actually a bit of a trick question, as there are two cases to consider. First, consider $a > 0$. The important property to ensure that the minimax strategy remains unchanged is that, when comparing any set of nodes, the maximum and minimum of those sets remain the same. One way to do that is to ensure that the order remains the same. I.e., if $x_1 < x_2$ then $f(x_1) < f(x_2)$ for all nodes x_1 and x_2 . For $f(x) = ax + b$ with $a > 0$, this is clearly the case. Therefore for strictly positive a , a linear transform doesn't change the minimax strategy. If $a \leq 0$, however, the order does change and so does the strategy. (I gave most of the credit if you simply didn't think about the $a \leq 0$ case.)

- You're in the new "Neon Swamp" casino and, as you wander through the crowds and beeping machines, you come to a series of tables, each of which is offering a different form of bet. Each table costs \$1 per play. For each bet, specify which action you should take (in a decision-theoretic sense) and justify why.

- The first table offers a \$5 prize if you roll exactly a 3 on one standard die. Should you take the bet?

Walk away. $U(\text{walk}) = 0$; $U(\text{bet}) = -1/6$. Therefore you will, in expectation, lose money on the bet.

- (b) The second table offers a \$2 prize if you can draw a red card (i.e., heart or diamond) from a standard deck with no jokers. Should you take the bet?

$U(\text{walk}) = U(\text{bet}) = 0$. In this case, the two actions have the same expected return (though the variance is higher for betting). According to the Maximum Expected Utility principle, you can choose either. Usually, we say that you are “indifferent” between the two choices.

- (c) The third table offers a more complex bet. You can choose whether to roll a die or draw a card. If you roll the die and get a prime number¹, the payoff is \$2.25. If you draw a card and get a face card (Jack, King, Queen of any suit), the payoff is \$4. If you draw an Ace, then you must roll a die and if you roll a 1 then the payoff is \$50. Otherwise, the payoff is 0 and you lose your initial money. Should you take the bet? If so, which action should you take (i.e., initially roll the die or draw a card)?

This one is a bit more complicated to analyze. It may be simplest if you draw the full tree out to see what all the possibilities and their relative rewards are. But it boils down to:

$$\begin{aligned}
 U(\text{walk}) &= 0 \\
 U(\text{roll}) &= 1/2(2.25 - 1) + 1/2(-1) = 1/8 \\
 U(\text{draw}) &= 9/13(-1) + && (\text{lose}) \\
 &\quad 3/13(4 - 1) + && (\text{face card}) \\
 &\quad 1/13(1/6(50 - 1) + 5/6(-1)) && (\text{ace}) \\
 &= 22/39 > 1/8
 \end{aligned}$$

Therefore you should take the bet and draw the card.

4. Write pseudocode for minimax and alpha-beta search routines for a two-player *non-zero-sum* game. I.e., assume that each leaf n has an arbitrary real vector payoff $\langle v_a(n), v_b(n) \rangle$ giving the reward for players a and b , respectively. If there are no bounds on v_a and v_b , can alpha-beta still prune nodes?

A number of people suggested simply mapping leaf values to $v' = v_a(n) - v_b(n)$ and then proceed with standard minimax. This effectively defeats the purpose of offering a non-zero-sum game in the first place, and is incapable of representing most of the situations/behaviors that we care about in that class of games. The problem is basically that it makes each player vengeful rather than greedy. While greedy isn't exactly a virtue in daily life, it is the principle that we're using to analyze these systems. To see the difference, consider a choice

¹Recall that 1 is not prime.

point that leads to two nodes n_1 and n_2 with

$$v_a(n_1) = 35$$

$$v_b(n_1) = 30$$

$$v_a(n_2) = 10$$

$$v_b(n_2) = 0$$

Which node should player a choose? The greedy strategy leads to n_1 , while the vengeful strategy leads to n_2 – player a is willing to sacrifice \$25 in order to cost player b \$30.

To implement a greedy strategy in a two player, general sum game, you must pass vectors of results around. Each player then maximizes his or her own return with respect to that vector:

// Solve 2-player general sum game

// Args: n≡current game node; p≡current player (a or b)

// Returns: list, (v,a), where:

// v=vector of values; a=best action for player p at n

function maximax(n,p)

if Terminal-test(n) **then**

return (Utility(n),nil)

// Utility(n) returns vector of values

end

$v := -\infty$

for a, s in Successors(n,a) **do**

$(cv, ca) := \text{maximax}(s, \neg p)$

if $(cv_p > v)$ **then**

$bestCv := cv$

$bestA := a$

$v := cv_p$

end

end

return $(bestCv, bestA)$