Homework 2

How large is the state space for Connect 3? You should answer this by running your program to generate the full game tree and recording how many nodes are generated.

The state space is 799 824 nodes. I recorded the state space by outputting the size of the minimaxCache. This was recorded after the final minimaxValue was output.

Assuming Player 1 plays first, do you know who will win, Player 1 or Player 2? If so, who? Explain your answer using the results of running minimax.

If Player 1 plays first then he should win the game. The minimax value outputted when we start with an empty board and it is Player 1’s turn is 1, which means that there is a path on the tree that allows Player 1 to win against Player 2.

What move should Player 1 open the game with? How should Player 2 respond? In the case of multiple equally good choices, list all possible options.

Player one should open the game by placing its sign in positions (3,1),(3,2),(3,3) (Figures of the three states can be found in the appendix). This would allow him/her to have the most opportunities to make a Connect3. The algorithm that I am using does not find any possible positions that Player2 can respond with because they are all equally bad. I believe that because it is such an early stage of the game and player one has played all very well chosen positions so the possibility of Player 2 to minimize Player 1’s game are very little. Also Player 1 chooses particularly central positions, which leave him a lot of space to expand in order to make a three line so this minimizes the opportunity of Player two to beat him.

Assume that the following moves have already been made: P1-2, P2-3, P1-2, P2-2, P1-1. It is now Player 2’s turn. What move should they take? In case of multiple equally good choices, list all possible options. Note: the moves already taken above describe the player and the column they drop their piece into. The columns are 1-indexed, so P1-1 would mean Player 1 dropped a piece in the first column.

There is only one position outputted by the minimax algorithm, the highlighted position is the one that was chosen by the algorithm:

00000

02000

01000

11220

This is logical because if we put a 2 in the position (3,2) then 1 will win. If we put it on top of the highest 2 then we cannot really make a line of three. If we put it in the position of (2,0) then we can potentially make a diagonal with the two positioned in (1,1) but that would be later in the game or we can try to fill up the column to make a line of three, which is still pretty hard. If we put a 2 at the position (3,4) then we could be easily intercepted by i=one in the position (3,3). While if we put the 2 where the algorithm indicates then we have two choices for winning the game so no matter where 1 positions its play Player 2 will always win.

In 1.a. you reported the total size of the full game tree. How many nodes can be pruned by αβ pruning? Note: you can compute this by tracking how many nodes are evaluated while running Minimax w/ αβ, then subtracting the number of evaluated nodes from your answer in 1.a.

After using alpha beta pruning the state space decreased to: 1276 nodes. This means that 798548 nodes were pruned by αβ pruning. This shows how effective alpha beta pruning.

Consider the problem of placing k knights on a n x n chessboard such that no two knights are attacking each other, where k and n are given k<=n2.

1. Develop a CSP formulation for the knight cover problem. In your formulation, what are the variables?

X={X1,X2,…Xk} where for 1=< i <= k , Xi =[xi,yi] where x and y represent the x coordinate o andh4 y coordinate of the chess board.

1. In your formulation, what are the possible values of each variables?

In my formulation each variable can hold any of the coordinates on the board. The domain is:

D =Dx,y where x=[1,2..n] and y=[1,2..n] for every i

1. What are the constraints?

For the constrains set all of the nights that appear in the L-shaped coordinates away from the assigned one should not be there.

For each Xi we will have Ci={Ci1….Ci8}, where Ci1=[xi-1; yi+2] and Ci1!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci2=[xi+1; yi+2] and Ci2!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci3=[xi-1; yi-2] and Ci3!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci4=[xi+1; yi-2] and Ci4!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci5=[xi+2; yi+1] and Ci5!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci6=[xi+2; yi-1] and Ci6!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci7=[xi-2; yi-1] and Ci7!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Ci8=[xi-2; yi+1] and Ci8!== for every X in the set{ X1..Xi-1, Xi+1…Xk}

Explain why it is good during CSP search to choose the variable that is most constrained, but the value that is least constrained.

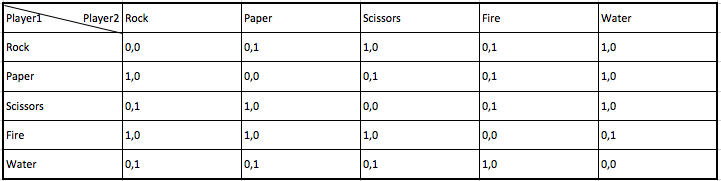
It is best to choose the variable that is most constrained because on the CSP graph representation it will have the most edges that connect it to other nodes and thus you would have less layers to go through afterwards. By starting form there it will be easier to assign values to the other nodes because they will be dependent on this most constrained one. It is best to choose the value that is least constrained because the number of variables that we can use it for depends on the extent of constraint. If we choose to start search with the least constrained value and assign it to a certain variable then we will have more options for other values to nodes that are adjacent to this starting node.

Prove (informally) the following assertion: For every game tree, the utility obtained by the MAX player using minimax decisions against a suboptimal MIN player will never be lower than the utility obtained playing against an optimal MIN player.

Suboptimal means that it doesn’t always take the best decision in his/her favor.

Suppose that the utility obtained by the MAX player in a minimax decision against a suboptimal MIN player is lower than the one against an optimal MIN player. This would mean that the suboptimal MIN player is making better decisions than the optimal in at least one case, which would mean that they would provide a better defense against the MAX player and therefore lower the overall minimax output. This is a contradiction because we know that the suboptimal MIN player doesn’t always take the best decision in his/her favor, while the optimal MIN player does. Therefore, the utility obtained by the MAX player using minimax decisions against a suboptimal MIN player will never be lower than the utility obtained playing against an optimal MIN player.

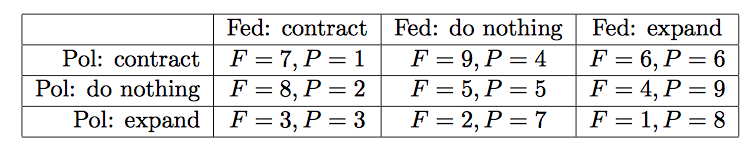
In the children’s game of rock-paper-scissors each player reveals at the same time a choice of rock, paper or scissors. Paper beats rock, rock beats scissors, and scissors beats paper. In the extended version rock-paper-scissors-fire-water, fire beats rock, paper and scissors; rock, paper, and scissors beat water; and water beats fire. Write out the payoff matrix for this game. If you were playing this game, what strategy would you employ? Why? Explain your answer using concepts from class.



**Table 1**. Payoff matrix for the game Rock, Paper, Scissors, Fire, Water. The utility of each player is denoted in the form “Player1, Player2”

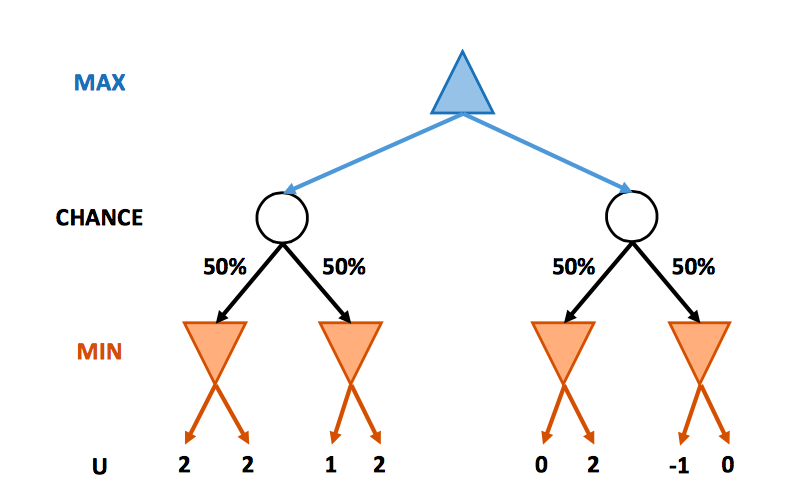
The strategy that I would obtain is to play Fire because it beats 3 out of the 4 elements left, so I will have 75% chance of winning if the other player chooses one of the other elements by a random strategy. This is a type of a mixed strategy. If I believe that my other player is inclined to play Fire than I might choose Water but this is very risky since all of the elements except for Fire beat water. In addition, if I believe that the other player is inclined to choose Water then I can pick any of the Rock, Paper, Scissors.

The following payoff matrix, from Blinder (1983) by way of Bernstein (1996), shows a game between politicians and the Federal Reserve.



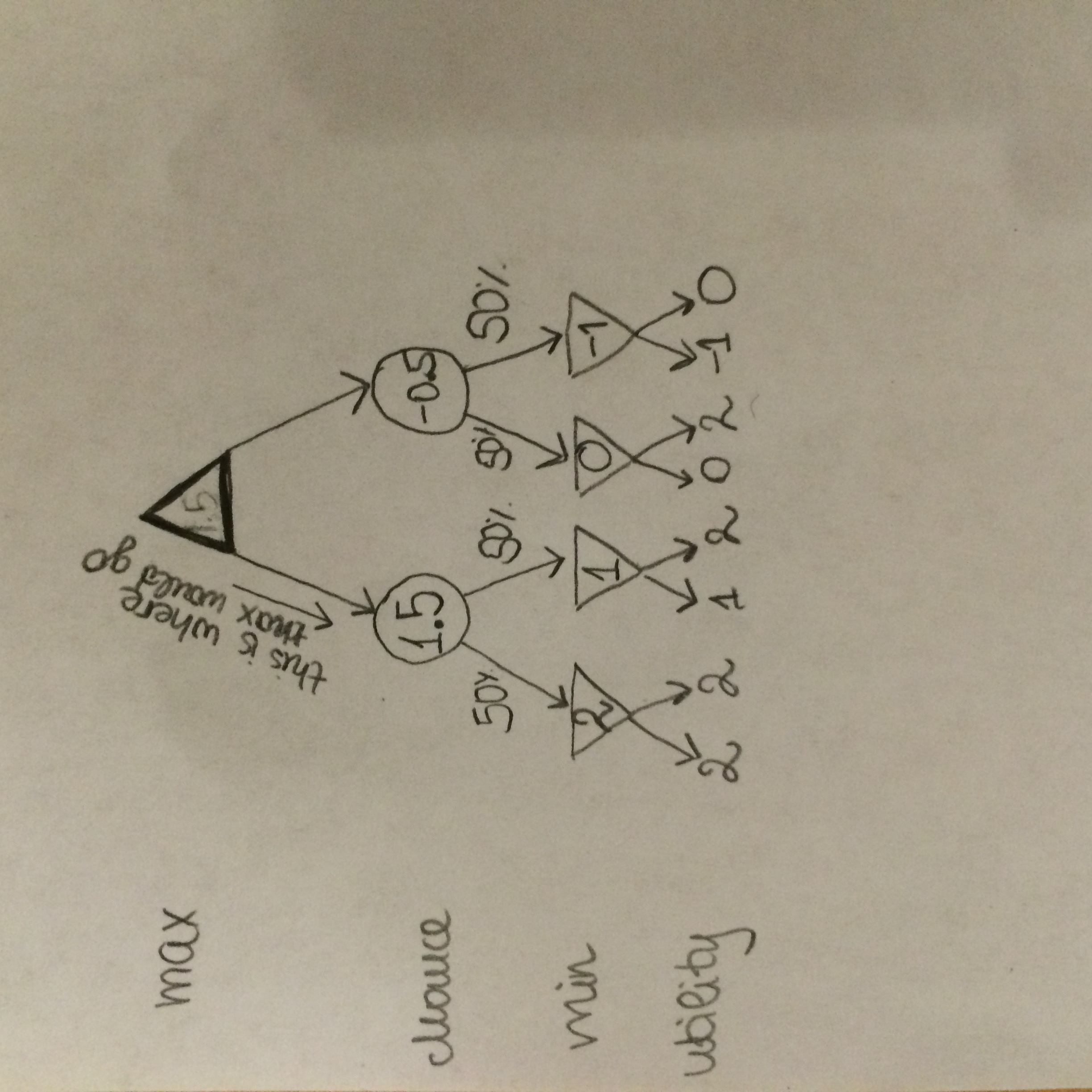
Politicians can expand or contract fiscal policy, while the Fed can expand or contract monetary policy. (And of course, either side can do nothing.) Each side also has preferences for who should do what: neither side wants to look like the bad guys. The payoffs shown are simply the rank orderings: 9 for first choice through 1 for last choice. Find the Nash equilibrium of the game in pure strategies. Is this a Pareto-optimal solution?

The Nash equilibrium is when F=6 and P=6. This is so because both Fed and Pol try to get the maximum number depending on their choice. Since they are both trying to maximize their outcome they must find some optimal decision, which would be represented by the Nash equilibrium. When Fed expands and Pol contracts they achieve Nash equilibrium and each of their strategies is locally optimal, since it is the maximum in this case. Fed has the highest value, six, when Pol contracts once he expands because Fed gets four when Pol does nothing and Fed gets one when Pol expands as well. The same can be done for Pol. Pol has the highest value, six, when Fed expands once he contracts because Pol gets four is Fed does nothing and Pol gets one is Fed contracts. In addition, this would be a Pareto-optimal solution because there is no other outcome that the players would prefer. There is no other possible option for Pol and Fed that both of them they will have a value greater than six.



This question considers pruning in games with chance nodes. The figure above shows the complete game tree for a trivial game. Assume that the leaf nodes are to be evaluated in left-to-right order, and that before a leaf node is evaluated, we know nothing about its value. The range of possible values is {−∞, ∞}.

1. Copy the figure, mark the value of all the internal nodes according to expectiminimax and indicate the best move at the root with an arrow.



1. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves?

Yes, we do because the min value coming out of those two nodes will be accounted as 50% into the chance node. If both the seventh and eight leave are very high values then the chance node will have a very high value and then the MAX would want to pick that chance node.

Given the values of the first seven leaves, do we need to evaluate the eighth leaf?

We wouldn’t need it because if the value is very small we would still have a very low chance node and the MAX is going to pick up the other one. If the value of the eight leaf is very high it will not be picked by the MIN node. So there is no point in knowing it.

1. Suppose the leaf node values are known to lie between -2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?

The leftmost MIN node will have a value of two, which will contribute in total with one to the chance node because it has 50% chance that is going to be picked up. The other MIN node (the one that has edges with the two leaf nodes we are exploring) can have the lowest value of -2 if either of the leaf nodes is -2 and the highest value of 2 if both of the leaf nodes have values of 2. This will contribute to the chance node with -1 or 1 respectively. Therefore, the range of the chance node will be between 0 and 2.

1. Which leaves do not need to be evaluated under the assumption in part c. above?

We would definitely need the values of the first 4 leftmost leafs in order to compute the value of the leftmost chance node. Then if we look at the fifth node we can see it is 0. Therefore, in the worst case scenario the MIN node above it is going to have a value of 0, so that max value of the chance node above it is going to have a value of 1 assuming the seventh and the eight nodes hold the maximum values in the range,2. We know that our leftmost chance node will have a value of 1.5 and that the maximum value of the other one given that the fifth node is 0 is 1. Therefore, no matter the sixth, seventh or eight nodes, the MAX player is always going to pick up the leftmost chance node. On the other hand if we do not evaluate the fifth node the chance node can have a maximum value of 2, which is higher than 1.5 and therefore there is the possibility that the MAX node will choose the right chance node. Therefore, we need to evaluate the fifth node.

Appendix:

Positions for Question 1b:

Turn: 2

00000

00000

00000

01000

Turn: 2

00000

00000

00000

00100

Turn: 2

00000

00000

00000

00010