Introduction To AI (67842) | Ex3

June 21, 2024

Question 4

(1)

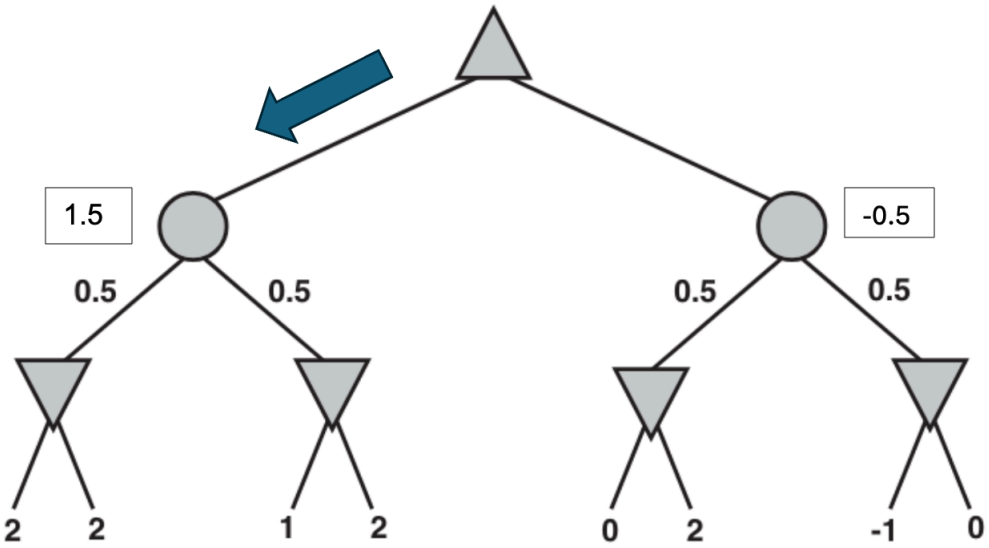

Figure 1:

(2)

- We will not need the value of all leaves.
- For example in the next graph:

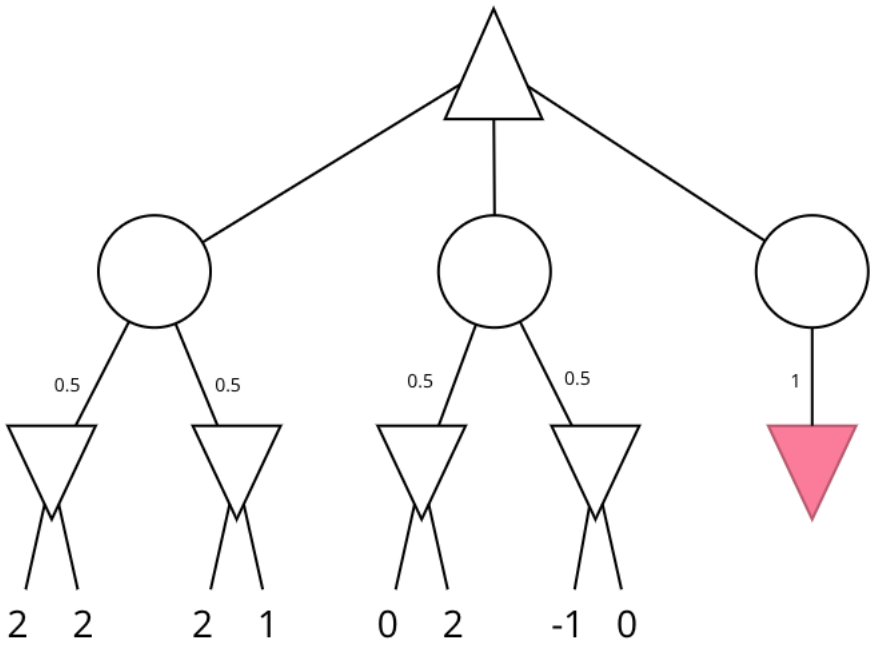

Figure 2:

- If we assign the evaluation of the red Min node to 400 and 400, we would get from the Min 400, and from the expectation function the value of 400, which the Max player would choose.
- However, we do not need both nodes. If we see that the 7th leaf is smaller than the minimum expectation value up to now, we know for sure that min will return a value that is smaller than the current expectation value, and Max wont choose it. In that case we can return the 7th value as the expectation value without needing to know the 8th value.

- In conclusion, every time we enter a new node for evaluation, we do not necessarily need to know all its leaves values. If one of them is smaller than the current evaluation value, we do not need to keep evaluating this node.

(3)

- In the graph with the given assumption, after evaluating the first 2 nodes, the Min player will choose the minimum between the values.
- We will refer to this minimum value as x.
- There are now $probability^{-1} - 1$ leaves left in this node.

- We conclude from this that the range of the expectation value of the node is:

$$\left[\frac{x + probability^{-1} \cdot (-2)}{probability^{-1}}, \frac{x + probability^{-1} \cdot 2}{probability^{-1}}\right]$$

- For example:
  - In the graph given in the question, after evaluation of the first two nodes, the Min player returns the value of **2**.
  - From this reason, the range of values that the left-handed expectation node will return is $\left[\frac{2+(-2)}{2}, \frac{2+2}{2}\right] = [0, 2]$.
  - As we showed in the first subsection, the expectation value returned is 1.5 which is in range.

(4)

- We will always need to check the values for all of the left-handed expectation node in order to get a expectation value.
- From now on, in each expectation node, we can check after evaluating the first Min value if the node range is smaller than the maximum of the expectation nodes evaluated up until now.
- If the range is smaller than the maximum expectation value:
  - We know that the Max player will not choose this path.
  - At this point we can return the value of the maximum range of the current node, and move on to the next evaluation node.
- Otherwise we expand and evaluate the next leaf. (after the evaluation we check the range again)
- For exapmple:
  - In the graph given in the question, after evaluating the first two leaves the Min player chooses **2**.
  - We continue to evaluate the next **2** leaves, as we said that we need to get a expectation value.
  - The min player returns the value of **1**, which means the expectation value is **1.5**.
  - We continue to the next expectation node:
    - The Min value is **0**.
    - The range that is possible is $\left[\frac{0+(-2)}{2}, \frac{0+2}{2}\right] = [-1, 1]$.
    - This range is smaller than **1.5**, which is our current maximum expectation value, so we dont evaluate any more leaves and return from the expectation node **1**.

* Now the Max player chooses between 1.5 and 1, it will choose 1.5 as we know it does.

Question 6

Success In The Game

(1)

- The most suitable algorithm for the 2048 game is the expectimax algorithm.
- The expectimax is more suitable for random occurences, because it takes in consideration all of the possible events. After that consideration it calculates the value according to the probability of an event to occur. In the 2048 game, placing a new tile shoud be random rather than choosing the worst place for the player.
- The Minimax algorithm chooses to place the tile in the worst possible spot for the player. This doesn't suit the game, because we want the player to be able to win, and not try to make him fail.

(2)

- The highest tiles in each game:
Algorihtm \ Game Number	1	2	3	4	5	6	7	8	9	10
Minmax	512	256	512	512	512	1024	1024	512	512	256
Alpha-Beta	512	512	512	256	256	512	512	1024	512	512
Expectmax	512	512	512	1024	512	512	1024	512	256	1024
- The scores:
Algorihtm \ Game Number	1	2	3	4	5	6	7	8	9	10
Minmax	5424	3560	6712	6580	5296	12212	12084	6732	6412	3088
Alpha-Beta	5600	5104	5516	3440	3412	7100	5560	11520	5296	6884
Expectmax	7192	7268	7112	12248	7072	7408	16460	7052	3064	12456
- The average highest tile:
  - Minmax - 563.2
  - Alpha-Beta - 512
  - Expectmax - 640
- The average scores:
  - Minmax - 6810
  - Alpha-Beta - 5943.2
  - Expectmax - 8733.2

(3)

- As we see from the averages, the Expectmax algorithms higest tile average and score average is higher than the other algorithms, which means the theoretical and empirical results are consistent.

(4)

Algorihtm \ Standard Deviation Of	Highest Tile	Score
Minmax	250.8	2930.06
Alpha-Beta	198.3	2180.99
Expectmax	232.32	3640.72

- As we see from this table, which displays the standard deviation of each algorithm according to its highest tile and score, the Alpha-Beta algorithm is more consistent than the Expectmax algorithm. The Minmax algorithm without pruning is a little less consistent than expect max in the term of highest tile, but is more consistent in with the score, which is roughly the same result with pruning.
- This is something we would expect to see:
  - The Minmax (with and without pruning) algorithm is more consistint in its essence. It will always choose the worst placement for the player, which the player can expect.
  - On the other hand, the Expectmax algorithm shouldn't be consistent as Alph-Beta, because it doesn't know what to expect for in the next move. The algorithm may not be random, but randomness is expressed by evaluating the probability of each outcome that could happen after the expectation node, and summing the products of each leaf with its probability.

Alternative Games

(1)

- The Minmax algorithm will after each turn will take the "boom" option. This is in order to make the player fail.
- The Expectmax algorithm will take in accoount the probability of getting "boom" option, which is 0, and try to calculate its options for other probabilities.

(2)

- In a game of 2048 where the new tile is placed 100% of the times in the worst possible place for the player, the Minmax algorithm will outpreform the Expectmax.
- The reason is because the Minmax will know for sure where the tile placement would be, and do its turn in order to minimize failure, while the Expectmax doesn't know where the placement would be, hence the failure is more probable.