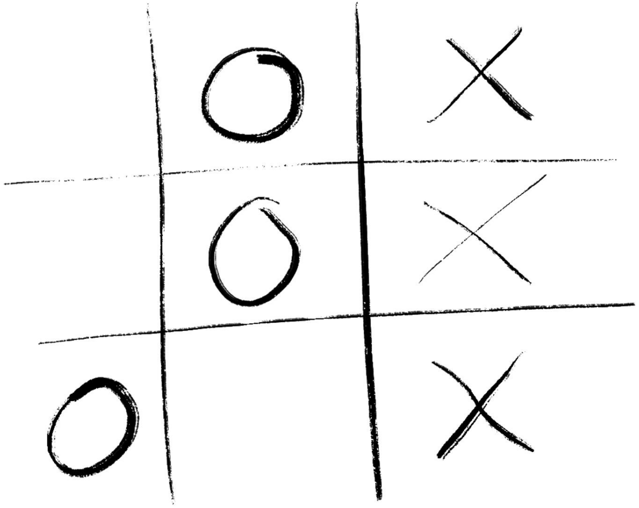
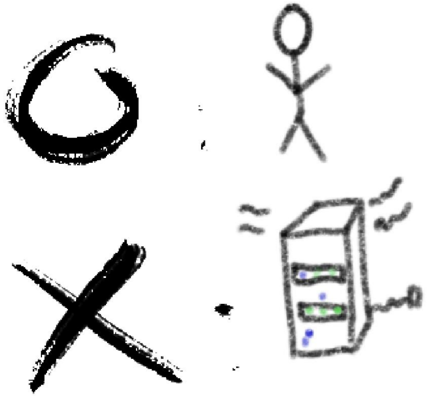
Tic-Tac-Toe using MCTS

Group 7





The problem

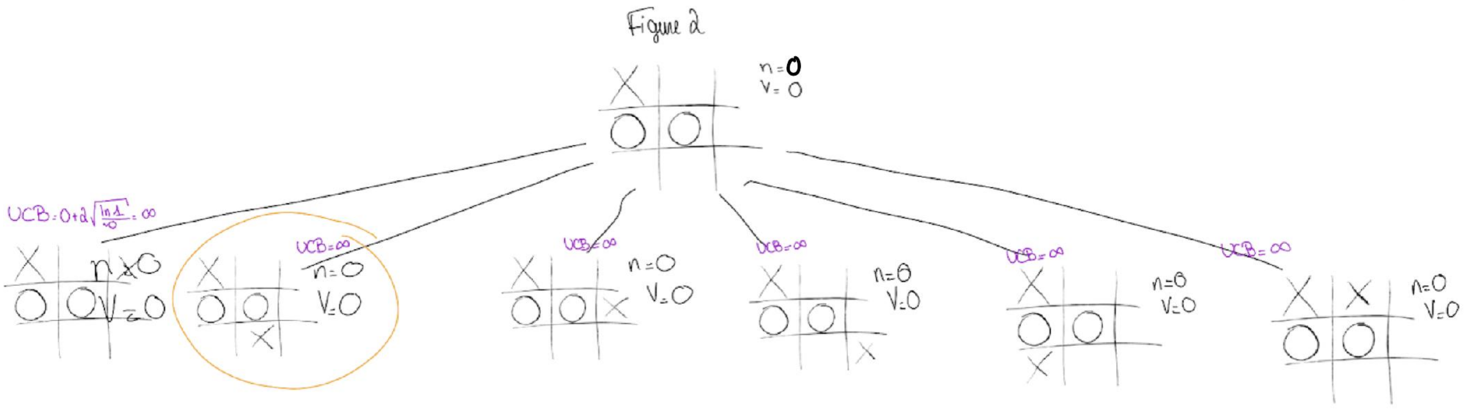
- Minimax searches through every path on tree for best move
  - Breadth of tree = # of possible moves
- Usable but slightly inefficient for simpler game like Tic-Tac-Toe
- Problematic for complex games like chess and Go
  - Larger game space and more complex rules lead to more possible moves
  - Tree too large to go through every path in reasonable amount of time

Monte Carlo Tree Search (MCTS)

- Alternative to Minimax for exploring game tree
- Efficient sampling
  - Doesn’t evaluate all moves equally
  - Focuses more on reevaluating promising or uncertain moves, ignores less promising moves
- Consists of 4 phases:
  - Selection
  - Expansion
  - Simulation
  - Backpropagation

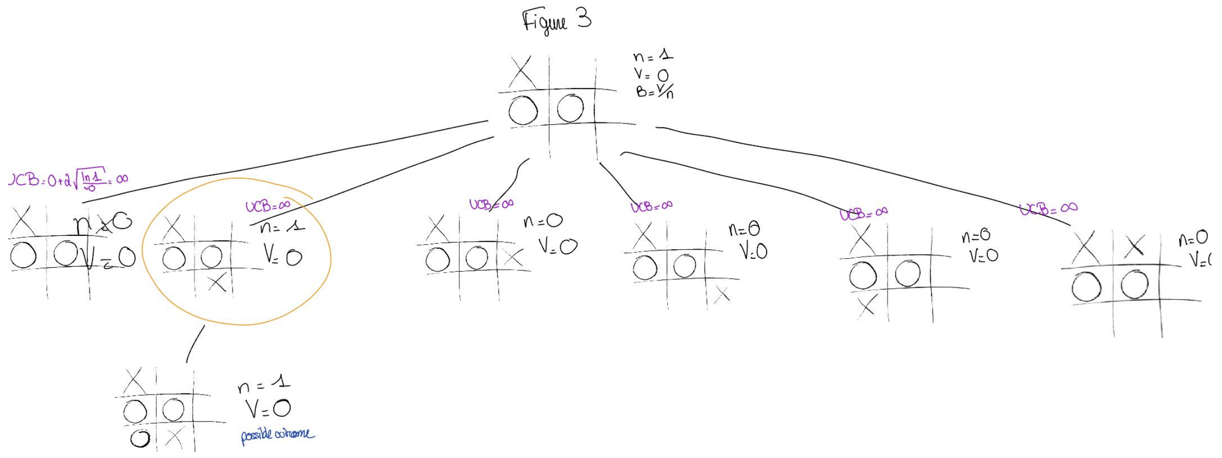
Implementation of MCTS - selection

- Every single children of the board state has a value V and a visit counts n
- Select the path with the max $UCB(k,p) = V(k) + C * \sqrt{\ln(N) \;/\; n}$
- k is the state, p the player, C = 2, N = parent visits, n = state visits



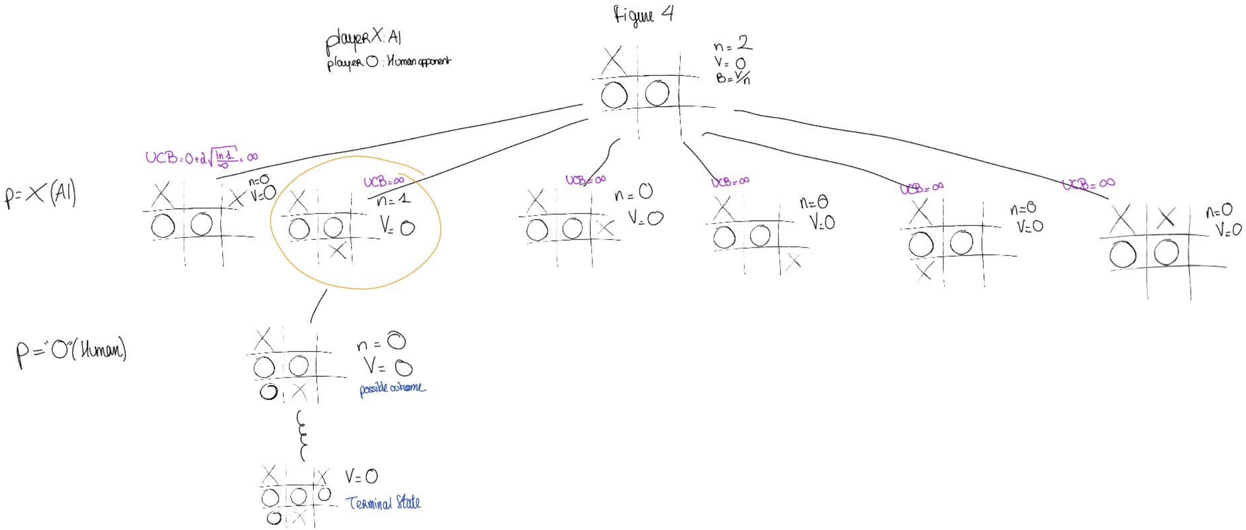
Implementation of MCTS - expansion

- The node chosen is being expanded. A new children is created with a n of 0 and a value V of 0.



Implementation of MCTS - simulation

- Simulate a game from the new expanded node
    - Play the game randomly until the game is terminated without saving it
    - Record the outcome : -1 : loss 0: draw 1:win

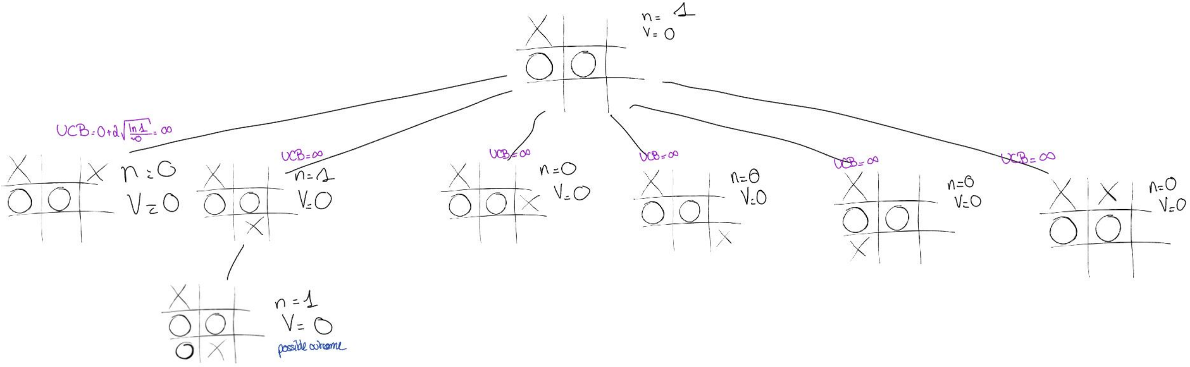


Implementation of MCTS - back propagation

- Back propagate the values to the root of the tree
    - V = V(old) + V(new terminal state),

    - n +=1.

Figure 5



Problems encountered along the way

KeyError: trying to access a key in a dictionary that does not exist. The children dictionary does not contain the key for the node object you are trying to access.

Solution: check if the node has a children and if not create the children of that node.

The influence of the number of iterations

- The timeout time t will directly influence the number of iterations
- The more iterations of MCTS, the more accurate the play should be
- Test was performed with different iterations, the outcome was as follows:
    - 100 iterations :

1st game	2nd game	3rd game	4th game	5th game
Human wins	draw	draw	draw	Human wins

    - 1000 iterations:

1st game	2nd game	3rd game	4th game	5th game
draw	draw	draw	draw	draw

Limitations of using MCTS

- Performance depends on the number of iterations performed
    - If a game has a higher branching factor, it would require a higher iteration count to come up with a viable play

- Susceptible to “trap states”
    - Superficial strong moves, might lead to loss via subtle line of play known by an expert player

Conclusion

- Advantages over alpha-beta pruning and similar algorithm
- Strengths and weaknesses of creating the asymmetric tree
- More powerful when coupled up with other types of architecture
- Overfitting

References

https://courses.cs.washington.edu/courses/cse599i/18wi/resources/lecture19/lecture19.pdf

https://en.wikipedia.org/wiki/Monte_Carlo_tree_search