Effects of increasing search depth and improving evaluation function:

I first started with a depth of 10 for this algorithm. On average the time seemed ok and only took a second or two to make a move, but in some scenarios it would take 4 or 5 seconds. I felt it took a little too long so I decided to decrease it to 5. Once I did this I found the agent was a little too easy to beat. It would make moves that were ok but I could see there were better ones it could've chosen from. Thus I ended up at 8 for a depth max, this seems to be a good balance in speed and optimal move choices.

For the evaluation function I first started by only scoring the terminalStates. My original thought was to have the agent choose moves based on the total amounts of possible end game scenarios that led to a win, I. E. if a move led to 50 different wins, and another move only led to 20, choose the move that led to 50. I did find this hard to implement at the start of games though as typically the depth of 8 wasnt enough for the agent to see an end, and it would take forever to play out all the scenarios. Thus I adjusted the evaluation function to look at the board in each state, giving value to regular and king pieces, so then the agent understood that taking pieces and getting kings leads to wins. I tweaked these numbers a bit but ultimately led to kings = .5 score and regular pieces = .25. This made the AI agent very hard to beat (I never beat it).

MCTS agent, performance using the theoretical optimal value and a different one:

The value I set for C was sqrt(4) = 2 to see the behavior change compared to the sqrt(2).
From what I saw it seems increasing this constant made the agent choose a more aggressive approach to the game, trying harder to capture my pieces than to move its back pieces, for example the agent moved its kings 3 spaces just to jump a piece of mine, while still have 4 pieces that had barely moved from its spot.

It seems theoretical optimal value gives the agent a much more balanced play style where it strategically develops the board when there isn't a potential jump in its near future.

Comparing the agents based on performance and the hybrid agents:

Given the time constraint of MCST I found it easier to beat on average, it took a lot more tweaking of its evaluation and simulation function to make it more difficult to play against. After putting in some print statements I found it does explore a LOT of scenarios, probably more than the alpha-beta search does.

On the other hand I found Alpha-Beta Search to be absolutely brutal to play against. It seems quicker overall and it really puts you into tough scenarios quickly. To be honest I couldn't beat it in probably 50 attempts. I'd imagine the reason for this stems from it "pruning" the possible good moves for the player which basically shuts down any advantageous moves.
The hybrid agent was also difficult to beat, you can tell which agent it chooses based on the time delay for the move. Typically the MCTS will always take 1 second to think, while alpha beta can be almost instant or a bit longer.