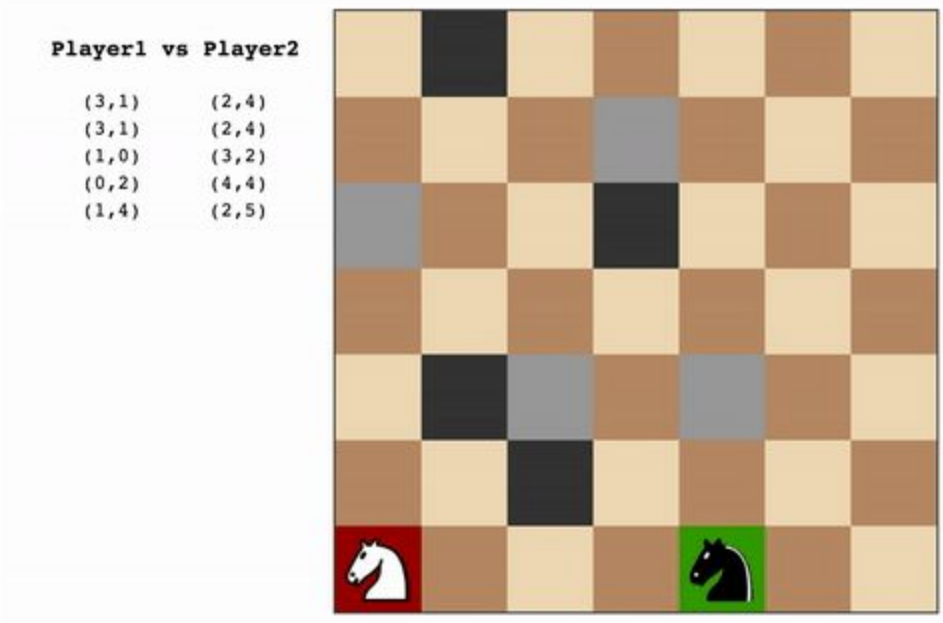
Artificial Intelligence Nanodegree Program

Manceñido Agustín

Build an Adversarial Game Playing Agent



Introduction

In the game Isolation, two players each control their own single token and alternate taking turns moving the token from one cell to another on a rectangular grid. Whenever a token occupies a cell, that cell becomes blocked for the remainder of the game. An open cell available for a token to move into is called a "liberty". The first player with no remaining liberties for their token loses the game, and their opponent is declared the winner.

In knights Isolation, tokens can move to any open cell that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. On a blank board, this means that tokens have at most eight liberties surrounding their current location. Token movement is blocked at the edges of the board (the board does not wrap around the edges), however, tokens can "jump" blocked or occupied spaces (just like a knight in chess).

Finally, agents have a fixed time limit (150 milliseconds by default) to search for the best move and respond. The search will be automatically cut off after the time limit expires, and the active agent will forfeit the game if it has not chosen a move.

Build an agent using advanced search techniques: Monte Carlo Tree Search

Create a performance baseline: alpha-beta

A performance baseline algorithm was developed and tested against all the available sample players. The player was tested with fair option enabled. If fair_matches is true, then the agents repeat every game they played, but the agents switch initiative and use their opponent's opening move. In some games, picking a winning move for the opening guarantees the player a victory.

Playing "fair" matches this way will balance out the advantage of picking perfect openings (the player would win the first time, and then lose when their opponent uses that move against them). The results of the test are shown in Table 1 and Figure 1.

Evaluate the effectiveness

To evaluate the effectiveness of the algorith a series of games were simulated against all the available sample players provided and compared the results against the Baseline alogrithm. For this work the following algorithms were selected:

- Monte Carlo Tree Search (MCTS)
- **Baseline**: Minimax Alpha-Beta Pruning

Both algorithms were tested against the available sample players:

- Random Algorithm
- Greedy Search Algorithm
- MiniMax algorithm

The percentage of games won against the sample players and the different parameters used are shown in the following Table 1:

	RAND (NF)	**RAND (F)**	**GREEDY (NF)**	**GREEDY (F)**	**MINIMAX (NF)**	**MINIMAX (F)**
Baseline (deph = 3)	90.60%	91%	64.30%	60.90%	39.70%	38.90%
Baseline (deph = 5)	90.40%	89.80%	69.40%	69.00%	45.80%	44.00%
MCTS (epochs = 40, C=0.5)	89.00%	87.70%	63.90%	63.60%	45.50%	42.50%
MCTS (epochs = 80, C=0.5)	91.50%	91.80%	74.20%	71.40%		
MCTS (epochs = 80, C=2)						

Table 1 - % of WON games

The results of the Table 1 were obtained over a 1000 games to make sure of getting consistent results over the sample players.

The following commands were used to produces these results:

- `python run_match.py -f -r 500 -o RANDOM`
- `python run_match.py -r 500 -o RANDOM`
- `python run_match.py -f -r 500 -o GREEDY`
- `python run_match.py -r 500 -o GREEDY`
- `python run_match.py -f -r 500 -o MINIMAX`
- `python run_match.py -r 500 -o MINIMAX`

The results from the tests are shown on the following Figure 1:

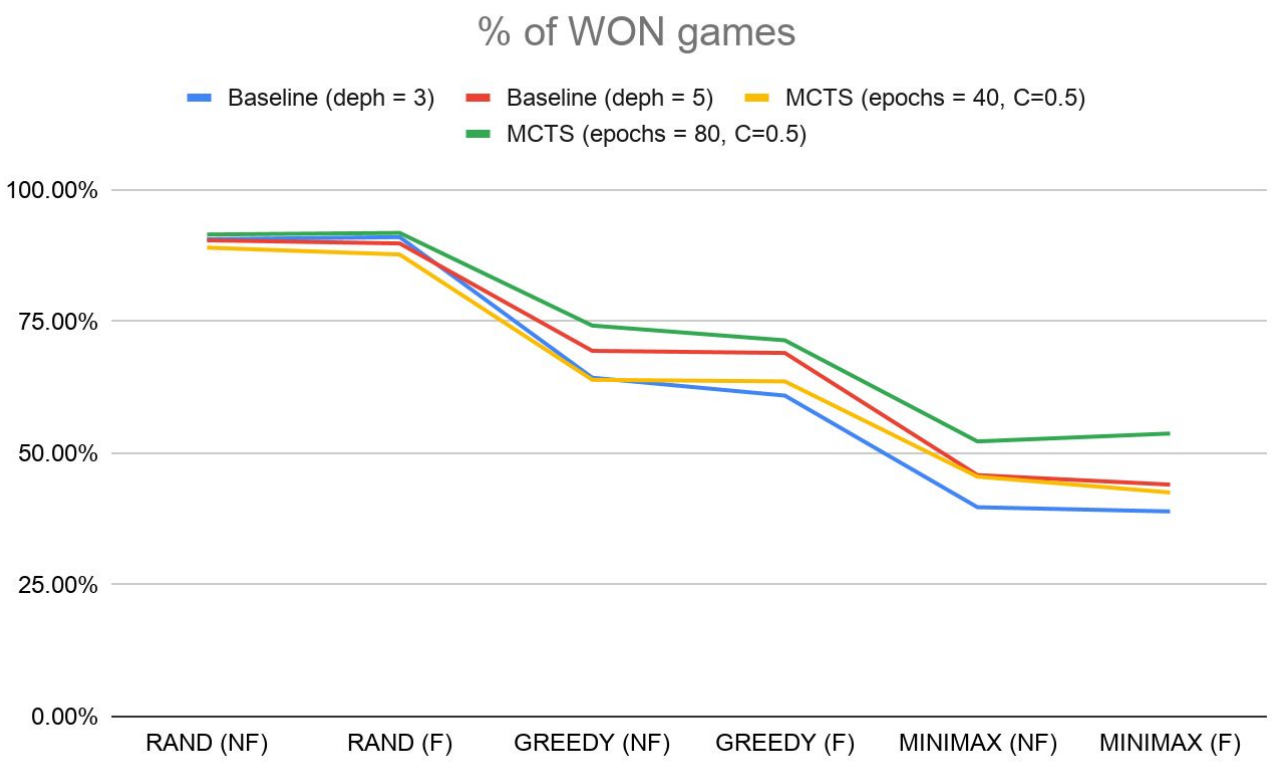


Figure 1 - % of WON games

Based on the results of the test is it possible to conclude that the GREEDY algorithm has a better performance than RANDOM and MINIMAX than both. Also it is possible to observe that in general the results of the NON-FAIR (NF) games are slightly lower than the FAIR (F) games, showing that there is an advantage on the initial conditions.

We can see that the MCTS algorithm improves considerably the performance of the baseline algorithm against the MINIMAX and GREEDY when using a higher number of epochs (80), even when the with a higher depth on the baseline model.