Abhinav R Pandey
Giorgi Kharshiladze
February 23 2018
Artificial Intelligence and Machine Learning
Lab B Report

Project Overview

For this project, we have implemented an AI that plays the popular board game *Breakthrough*. The underlying algorithm that powers the game is our implementation of the minimax algorithm which can utilize a variety of utility functions that we've developed. The utility function defines the strategy that is used by the AI player by assigning high values to favorable outcomes and low values to unfavorable outcomes. Although it wasn't formally required, we have also implemented a GUI using the bottle framework in Python.

Representation Scheme

Our state representation for the game is a list of lists (i.e. two-dimensional list) of characters containing 'X', 'O', and '.'. We have chosen to use the two-dimensional list because the game can easily be represented as a plot of X-Y coordinates. In order to move the positions of pieces, we can simply reassign list indices to the updated values. All of the functions required for part 1 have been implemented by using this state representation scheme.

How does Evasive Play?

The Evasive heuristic was one of the default utility functions that we implemented. After testing this utility function with other utility functions, we realized that it isn't too bad at playing. Since the basis of this heuristic is to minimize the number of pieces lost, the heuristic allows the AI to maintain its pieces towards the beginning of the game. However, as the game progresses and the pieces scatter, it can usually be defeated quite easily. Since this heuristic doesn't consider terminal game states and does not value minimizing the opponent's pieces, it probably isn't the most effective heuristic.

Evasive Outcomes

Below are some game ending states of Evasive vs Evasive.

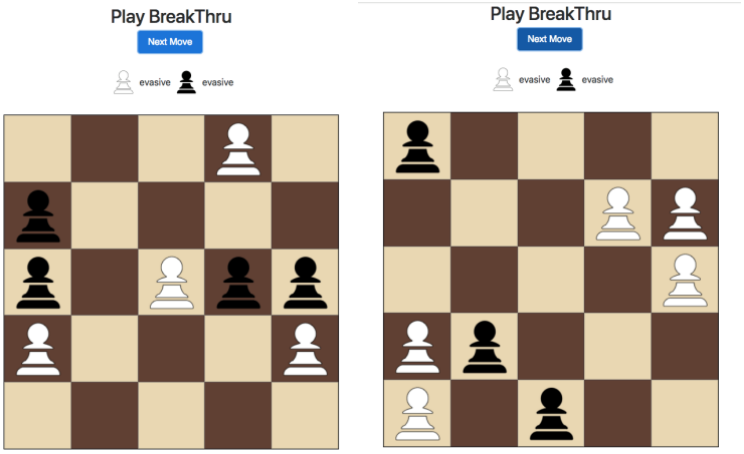

Figure Terminal States of Evasive Vs Evasive
Game 1 (Left): Number of pieces captured: 1, Number of moves (Total): 15
Game 2 (Right): Number of pieces captured: 0, Number of moves (Total): 23

Evasive Vs Conqueror

After making the two play against each other using Evasive and Conquer heuristics, we noticed that conqueror was much more effective than evasive. Since conqueror has a very aggressive heuristic, it was able to move forward towards the end of the board very quickly and capture enemy pieces efficiently. With smaller board sizes, conqueror usually won by reaching the opponent's size, however with larger board sizes, conqueror usually won by defeating all of the opponents pieces. A peculiarity that we noticed was that although conqueror is an aggressive heuristic, it was able to defend much better than the evasive heuristic (since evasive doesn't really attack opponent pieces frequently, and this presents a number of disadvantages when the opponent is closing in on the evasive player's pieces). Below, we show a terminal board state in which the black player used the conqueror heuristic.

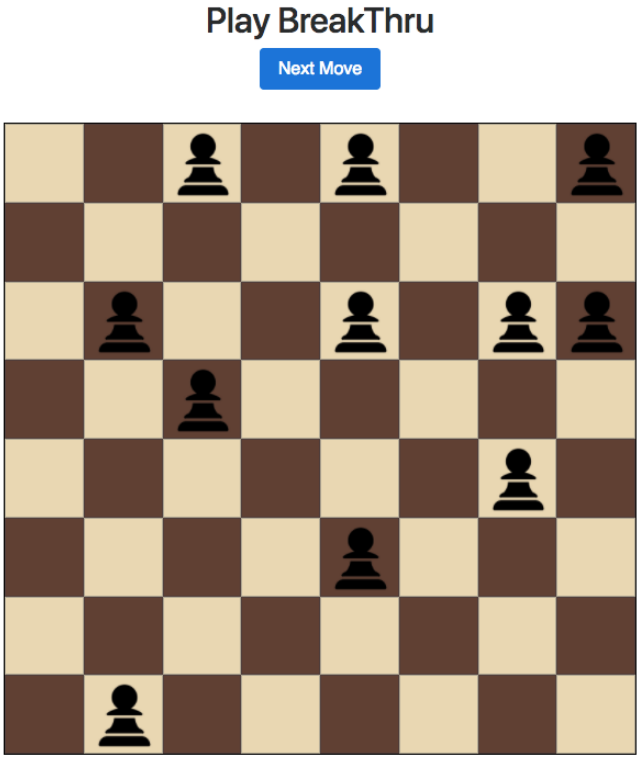

Figure 8x8 Conqueror Vs Evasive Terminal State

Our Heuristics

After experimenting with a variety of ideas, we came up with two heuristics that were working decently well against both conqueror and evasive. Our heuristics are called *Block* and

Enhanced. Both of heuristics incorporate slightly less aggressive and slightly more defensive styles.

Block is a heuristic that assigns a high utility value to game states in which the opponent pieces are far from the player's end. This is achieved by choosing game states in which the maximum 1st dimension index value of the opponents pieces is minimized. This function also utilizes a helper function called PlayerWinCheck which determines whether a given game state is a terminal state for the given player. If this function returns true, then the winning game state is assigned a very high utility value. So, this function is concerned with winning the game, preventing the opponent from achieving a terminal state, and minimizing the opponent's movement towards the player's end. After testing this heuristic multiple times, we found that it was very effective against the evasive heuristic but less so against the conqueror heuristic.

Enhanced is a heuristic that assigns a high utility value to a particular defensive state. While researching defensive strategies online (mainly through YouTube videos of real competitions), we found that it is beneficial to have two pieces in the columns 1 and 2 and two pieces in columns 5 and 6 (in an 8x8 game), as this creates a very good defensive structure. The pieces in these columns are able to attack an incoming enemy piece from any given direction. Although this heuristic isn't fool-proof, it's better to have these structures throughout the board so as the facilitate the blockade of enemy pieces. The heuristic assigns high values to game states with pieces in the aforementioned coordinates, and assigns the conqueror heuristic to game states that do not possess this formation. By doing this, in case there are no options in which these formations exist, the heuristic will become more aggressive. Similar to *Block, Enhanced* also uses the helper function PlayerWinCheck to assign high utility values to winning states and low utility values to losing states. All in all, *Enhanced* is a compound heuristic that considers a number of variables.

Our Function's Performances

Below are terminal outcomes with our heuristic functions.

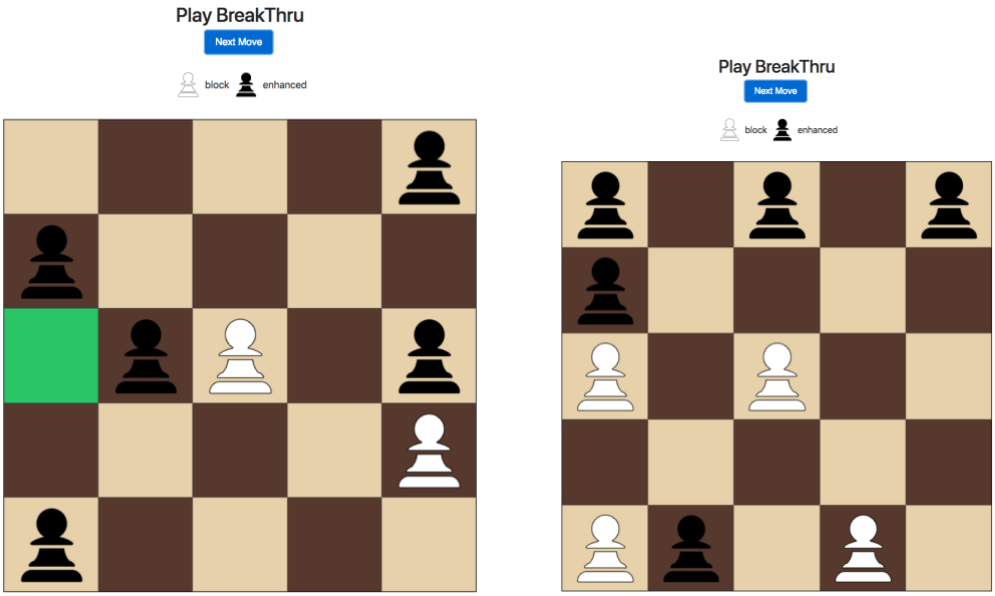

Figure Terminal States of block vs enhanced
Game 1 (Left): Number of pieces captured: 3, Number of moves (Total): 17
Game 2 (Right): Number of pieces captured: 1, Number of moves (Total): 10

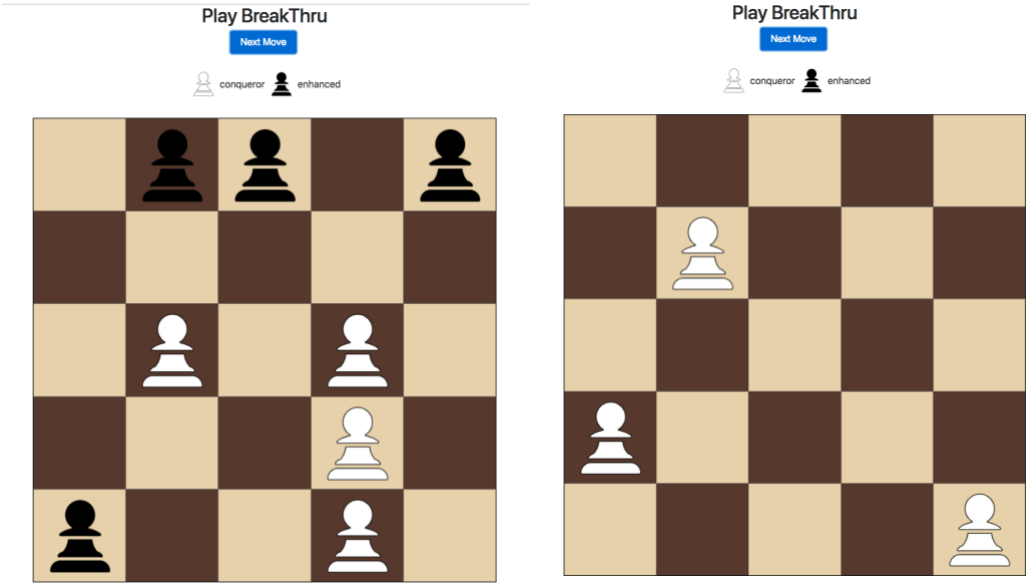

Figure Terminal States of conqueror vs enhanced
Game 1 (Left): Number of pieces captured: 2, Number of moves (Total): 13

Game 2 (Right): Number of pieces captured: 7, Number of moves (Total): 16

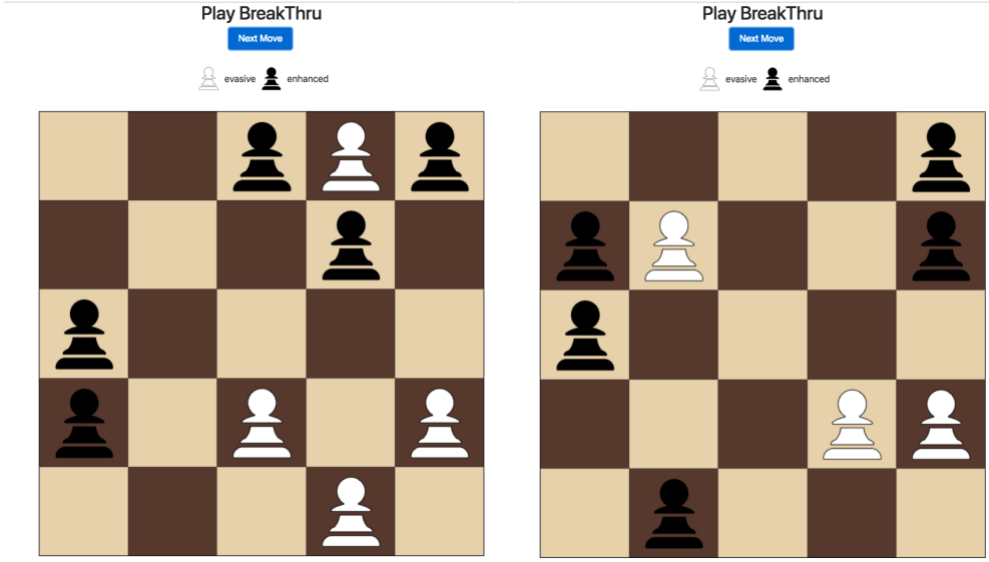

Figure Terminal States of evasive vs enhanced

Game 1 (Left): Number of pieces captured: 1, Number of moves (Total): 9

Game 2 (Right): Number of pieces captured: 2, Number of moves (Total): 11

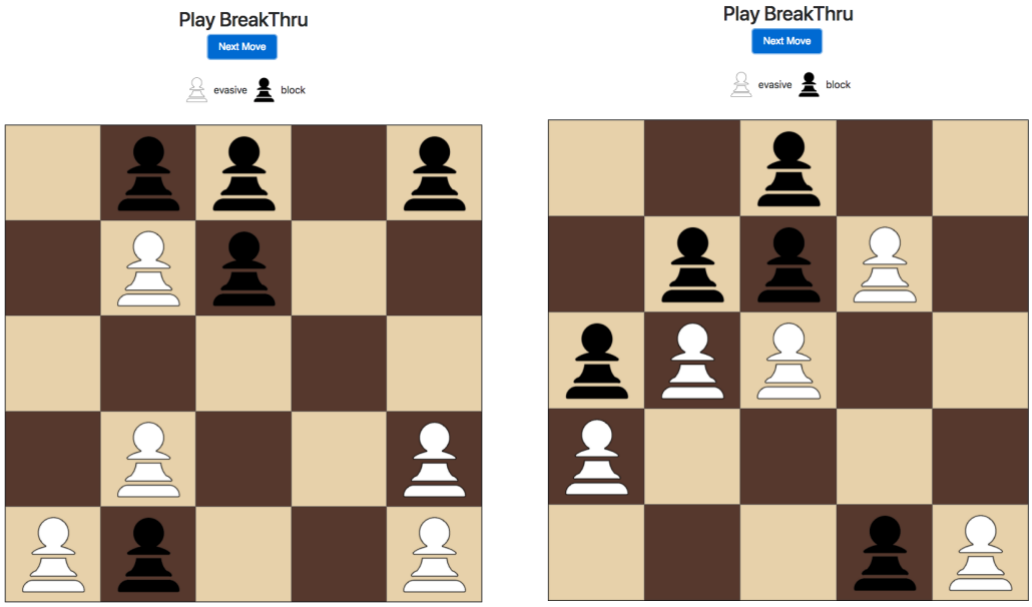

Figure Terminal States of evasive vs block

Game 1 (Left): Number of pieces captured: 0, Number of moves (Total): 10

Game 2 (Right): Number of pieces captured: 0, Number of moves (Total): 13

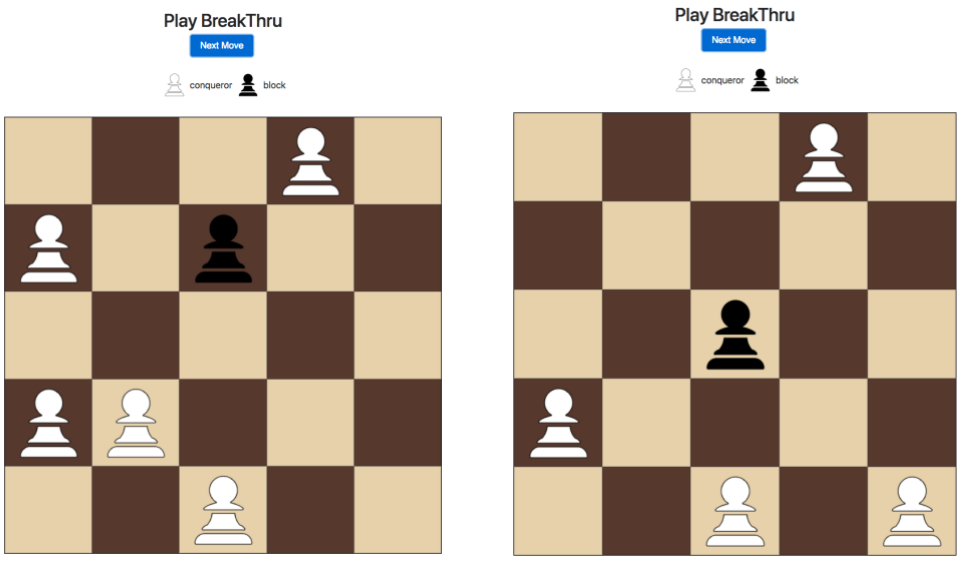

Figure Terminal States of evasive vs block
Game 1 (Left): Number of pieces captured: 4, Number of moves (Total): 17
Game 2 (Right): Number of pieces captured: 6, Number of moves (Total): 15