Anuroop Bisaria - 2021FC04237

Vipin Indoria – 2021FC04245

Rahul Kant Tripathy – 2021FC04539

Maneesh Kumar Shrivastava - 2021FC04011

# Artificial and Computational Intelligence Assignment 2

## Game Playing

### Part 1 – Choosing a static evaluation function

The evaluation function chosen in this circumstance is based on conventional chess position evaluation based on force strength and piece value. Since positional value is dynamic, a general sense of the relative strength of a chess position is usually gleaned from the available pieces for either side of the board. Each piece is assigned a certain value, with pawns valued the least and queens valued the most (the king is always on the board and has infinite value, as such it is not considered in such evaluations).

In the sample game state provided in the question where the human player is White and the computer is Black, the net board evaluation would return (1+3+5)-(1+3) = 5, which is symbolic of the fact that White has a piece advantage on Black worth 5 points i.e. they are up a Rook. Interesting to note that even if White were to lose the Pawn and Knight the board would evaluate to 5-(1+3) = 1, which is still an advantage since a Rook is considered in conventional chess theory to be a more potent piece than a Pawn and Knight combined.

### Part 2 – MinMaxing the Game of Chess

The minimax algorithm in this implementation is three levels deep, i.e. it will minimize the opponent’s position, maximize the player’s position, then minimize the opponent’s position to find the optimal board position for all three constraints.

The implementation does not account for alpha-beta pruning and is instead searching through all available moves. This is not a concern seeing as the tree size will be quite small, compared to a full-size chess board.

### Part 3 – Expectiminimax and The Steps Involved

Since chess is a game with perfect information without an element of randomness, expectiminimax can be hard-coded into static evaluation functions by understanding the distribution of the eval function. Since the function is abstracting the likelihood of win as a function of the relative strength of the chess armies, it is possible to directly compare the function values to probabilities of winning.

“Knight on the corner means 0.5 chance of win/loss” implies that the evaluation of the position is even. Given the definition of the eval function, this would result in a value of 0. Therefore, an override on the evaluation function would have to be created, where the corners of the play space would be checked for a Knight, and then the value of the function would be set to 0.

“Rook in the game means 0.8 chance of win” would mean that any board position involving a rook would automatically be evaluated to 0.3 times the maximum value of the evaluation function. In the case of the sample game state it would be 2.7 rounded up to 3 but for a full game board it could be much larger.