**CS 6364.0U1 - ARTIFICIAL INTELLIGENCE**

**Programming Assignment - 2**

**CHESS PROGRAMMING**

Mehavardhini Vijaya Kumar

mxv141730

**Introduction**

The goal of this project is to design a simple chess engine that plays against humans. The chess engine is a basic chess engine which provides strategies for the user to increase the chances of winning.

**Architecture**

The chess game basically comprises of the following parts,

**Board representation**

A chess board consists of a 64 squares, starting with a dark square in the left top corner and a light square in the left bottom. The color alternates between dark and light. The thumb rule is White king should be placed on a black square and vice versa. The chess board in the engine is represented using an 8\*8 two dimensional array with pieces set for initial view. The pieces in the board are represented as follows,

A: White King, a: Black King

Q: White Queen, q: Black Queen

K: White Knight, k: Black Knight

B: White Bishop, b: Black Bishop

R: White Rook, r: Black Rook

P: White Pawn, p: Black Pawn

**Moves**

Move engine which has all the possible moves for each distinct item on the board. The rules for each item on the board is defined in the moves. A chess game is considered complete when either of the King is captured. Once either king is captured, no more moves are allowed in the game.

**Evaluation/Rating**

Evaluation/Rating – How optimal a move is, and what would be the net effect of this move, this is done using the Alpha Beta pruning. Every piece is given a weight, the lesser the weight, the higher is the chances of making it hit by the opposite player. In other words, as the piece value is higher, higher is the protection.

**Chess Engine**

The main program which instantiates the other classes. The logic used for every move is the Alpha-Beta pruning. This is the main logic behind the entire game play. Alpha Beta is an adversarial search algorithm, which takes the maximum value from the move evaluation program and prunes all the other options. The goal here is assumed to be capturing the other player’s king, or a check mate/stale mate.

**Graphical User Interface**

Represent the board and the coins pictorially using Java swings. The moves possible in the chess board are basically represented using, mouse click, drag and release. If a move is valid, this operation results in successful movement of the piece. Various game strategies are provided to the user through this interface.

**Action Listener**

An action listener, which is a swing component, responds to every move and also reacts to a change in heuristics.

**Strategies**

The following strategies have been implemented in the chess engine, using the alpha beta pruning algorithm.

1. Search Depth

As the depth increases, the number of nodes for evaluation increases. As a result, the time taken for one best move after Alpha Beta search is more. Ideally, the depth of 4 is the maximum that this Chess Engine could hold. The value of 5 makes it very slow.

1. Sort the nodes after generation to make the pruning easier

Alpha Beta pruning makes the evaluation of the nodes faster if the nodes are sorted. Only one node has to be searched when sorting is in place. This will considerably decrease the time taken by the adversarial search to return one best move.

1. Change the weight for each item on the board

The weight of each coin in a chess board is different. The king has the maximum weight and the pawn has the least. By increasing the weight of one item, we can protect that item from being hit by the opposite player

1. Increase the weight of one bishop/one knight when the other is lost

When one of the knight or bishop is lost, we can increase the weight of the other knight or bishop. This can also be done to the rook.

1. Save the rook for the next ten moves

The rook is been saved just in case we have a disadvantage in the game. But the strategy is to save the rook for just 10 moves. This also can be extended for any piece. Since, there is a strategy for Knight and Bishop, this strategy is applied only to Rook.

1. Branching Strategy

Decrease the no of branches, when we are focused on a specific move. This also considerably decreases the number of nodes examined and hence decreases the time in returning the best move.

**References**

AIMA code for heuristics

You tube – Alpha Beta pruning

**Technology**

Java/Java Swings