ELECTRONIC ASSIGNMENT COVERSHEET

Murdoch UNIVERSITY

Student Number	35355782
Surname	Badal
Given name	Arogya
Email	35355782@student.murdoch.edu.au

Unit Code	ICT206
Unit name	Intelligent Systems
Enrolment mode	Internal
Date	1/11/2024
Assignment number	2
Assignment name	Project
Tutor	David Parry

Student's Declaration:

- Except where indicated, the work I am submitting in this assignment is my own work and has not been submitted for assessment in another unit.
- This submission complies with Murdoch University's academic integrity commitments. I am aware that information about plagiarism and associated penalties can be found at http://www.murdoch.edu.au/teach/plagiarism/. If I have any doubts or queries about this, I am further aware that I can contact my Unit Coordinator prior to submitting the assignment.
- I acknowledge that the assessor of this assignment may, for the purpose of assessing this assignment:
  - reproduce this assignment and provide a copy to another academic staff member; and/or
  - submit a copy of this assignment to a plagiarism-checking service. This web-based service may retain a copy of this work for the sole purpose of subsequent plagiarism checking, but has a legal agreement with the University that it will not share or reproduce it in any form.
- I have retained a copy of this assignment.

I am aware that I am making this declaration by submitting this document electronically and by using my Murdoch ID and password it is deemed equivalent to executing this declaration with my written signature.

Signed Arogya Badal

(Write your name in the space above)

Title

Introduction
This document presents the development, application and evaluation of a chess bot for my ICT206 Intelligent Systems course. I therefore gave attention to how best to incorporate intelligent logic into the bot which I did in the form of a Negamax algorithm with alpha-beta pruning. While the ChessChallenge framework provided the structural elements, I developed custom logic for the algorithm and evaluation functions to make the bot competitive. Testing was done through 10-game matches against both a default "Evil Bot" and customized versions of my bot.

Background

My goal with this project was to understand intelligent decision-making by programming a chess bot capable of evaluating multiple board positions to choose optimal moves. I selected the Negamax algorithm, a variant of Minimax, for its simplicity in this context. This project focused on implementing Negamax with alpha-beta pruning within the ChessChallenge framework from Sebastian Launge, where I customized the logic for decision-making and position evaluation. The ChessChallenge framework provided the foundational components like move generation and board tracking, allowing me to focus on implementing the logic.

AI Method and Tools

Negamax Function

The Negamax function, which is my bots core decision-maker, looked for possible moves to a fixed depth and utilized alpha-beta pruning to minimize search space. With the color parameter, the evaluation scores were altered in a way that displayed the bot moves and the opponent's moves as different.
The algorithm steps included:

1. Move Generation: Constructing an array of the entire possible moves through the framework.
2. Move Execution and Reversal: Applying each move temporarily to explore outcomes.
3. Scoring and Pruning: Applying the evaluation function to give scores to moves and using alpha-beta pruning to make the search window narrower.

Evaluation Function

My evaluation function contained a score variable which was the sum of the piece values and positional values for all the pieces. For the positional values, I used custom piece-square tables enhancing the bot's awareness of board control. Additional points were awarded for checks which encouraged control-oriented play.

Piece-Square Tables

I developed piece-square tables for each piece type (Pawns, Bishops etc.), giving importance to the piece positions helping the bot to control space strategically.

Evaluation Method

The Framework had a system of an EvilBot and MyBot where you could put the bots to play against each other to check how they play and how good they are.
To evaluate my bot, I used the Framework to run 10-game matches between the default EvilBot and various iterations of itself to benchmark improvements.

Furthermore, It also allows the chess bot to play against the user which I recommend the marker to check out if they want to test it out themselves.

Results

- Negamax Depth 1(MyBot) vs Default EvilBot

Initially my Bot was worse than the default EvilBot, losing all 10 games.

- Negamax Depth 4(MyBot) vs Default EvilBot

So, I modified and added a function that would let me search with more depth resulting in winning all games.

- Negamax Depth 4(MyBot) vs Negamax Depth 5(EvilBot)

Here, for Depth 5 the bot would in all cases end up losing in time due to having a bigger search window so I found the bot with the depth 4 to be superior than it.

Conclusion

Creating this chess bot for ICT206 proved to be a good exercise for understanding algorithms. I wanted to make a bot that played good games using custom-defined evaluation functions and Negamax algorithm.

When tested in 10-game matches with instances of the bot, I observed the mistakes that it made and proved to be a huge help towards making adjustment to make the bot better. This project showed how Negamax works in total environment and laid a good foundation for learning intelligent systems.

Acknowledgements

The Framework I used:

GitHub - SebLague/Chess-Challenge: Create your own tiny chess bot!

User Guide

So, to access the program one can simple open the solution file present in the zip through Visual Studio. Then run it. You will then have different ways to test out the bot.

The submission will be marked using the following rubric:

CATEGORY	5	4	3	2	1	0
On Time/Late Without Extension	On time 0 marks	1-2 days late -1 marks	3-4 days late -2 marks	5-6 days late -3 marks	7 or more days late -4 marks	
Presentation (coversheet, diagrams, captions, references in IEEE format, clarity and style)	Excellent +5 marks	Very good +4 marks	Good +3 marks	Acceptable +2 marks	Poor +1 mark	Very Poor 0 marks
Description (problem, background research and goal)	Excellent +5 marks	Very good +4 marks	Good +3 marks	Acceptable +2 marks	Poor +1 marks	Very Poor 0 marks
Solution & Implementation (good solution, good use of tools and resources)	Excellent +5 marks	Very Good +4 marks	Good +3 marks	Acceptable +2 marks	Poor +1 mark	Very Poor 0 marks
Performance of Solution (solution runs according to design, goals reached)	Excellent +5 marks	Very Good +4 marks	Good +3 marks	Acceptable +2 marks	Poor +1 mark	Very Poor 0 marks
Evaluation & Conclusion (quality of evaluation, conclusion)	Excellent +5 marks	Very Good +4 marks	Good +3 marks	Acceptable +2 marks	Poor +1 mark	Very poor 0 marks

Tutor's Comments