CSE 3117: Artificial Intelligence

Project 1

Due Date: Dec 03, 2025

Mid-project evaluation: via Github link

Overview

This project consists of developing and implementing a computer program that can play a game against a human opponent. There are three such games to be implemented, each group will work on one of these games as assigned. The games are: **Gomoku**, **Minichess,** and **4-connect.** The project will exemplify the minimax algorithm with alpha-beta pruning, and designing good evaluation functions.

Game description: Gomoku

Gomoku, also known as "five in a row", is a board game similar to tic-tac-toe but on a larger board. It is a two-player, fully observable, deterministic, zero-sum game. Players take turn in placing 'stone's in the board until one player can connect five stones of the same color in a row horizontally, vertically or diagonally. It is typically played in a 15 X 15 board, but for this project a **10 X 10 board** is sufficient. Detailed information about the game can be found in: https://en.wikipedia.org/wiki/Gomoku

Game description: Minichess

Minichess is shorter version of Chess. It is also a two-player, fully observable, deterministic, zero-sum game. There are several versions of Minichess available. You will work with any version of the **6 X 5 board**. Detailed information about the game can be found in: https://en.wikipedia.org/wiki/Minichess#5%C3%976_chess

Game description: Connect Four

is a game in which the players choose a color and then take turns dropping colored tokens into a six-row, seven-column vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own tokens.
https://en.wikipedia.org/wiki/Connect_Four

Project Requirements

- Your AI program must be able to play a reasonable game against a human or AI opponent
- The program must have the following components
    - Searching the **game tree** using a suitable search algorithm
    - Running **minimax** algorithm on the game tree to select the right move
    - Implementation of **alpha-beta pruning** to make search faster
    - Design and implementation of an effective **evaluation function** to evaluate the goodness of any intermediate node (state of the board).
    - An **early stopping** mechanism to be able to abort search and return the minimax values based on the evaluation function.
- A functional **user interface** is expected. An aesthetic/graphical user interface is encouraged, and will bear extra credits.
- You must maintain a **Git** (or similar version control system) repository of your project.

Grades distribution

Game tree search + Minimax	30%
Alpha-Beta pruning	20%
Design and implementation of Evaluation function	20%
Early stopping of search	10%
Playing "reasonably well"	10%
Interacting well with user	10%

There will be a **mid-project evaluation** where 25% of the marks of the project will be assigned. The remaining 75% will be assigned on the final project evaluation.

Deliverables

1. A fully functional software (with Git repository)
2. A 1-page report explaining your evaluation criteria and other implementation details

Deadline

The deadline is **Wednesday, Dec 03**. The projects will be checked during class time. No submissions after the deadline will be accepted. Absolutely **no exceptions** will be made.

Academic Integrity Policy

The code of the project **must be your own.** Any libraries or code snippets that you use from other sources must me mentioned explicitly. Any kind of academic dishonesty will result in you getting a zero for this project, and possibly further repercussions.