Jonathan Dolan

Project proposal

Abstract

Chess, the Immortal Game, has been regarded as an excellent test of intelligence for centuries. It is no surprise then, that early work in Artificial Intelligence and game theory often investigated ways for algorithms to play chess and mimic the creativity and planning that is required of a human grandmaster. Since a computer beat the greatest human chess player in 1997, the field of chess algorithms has stagnated, I hope my project ignites new ideas and discussion in this field. I plan to create a chess engine that is specifically designed to beat other chess engines, showing a weakness of the algorithm and generating more discussion on the subject.

Introduction

For my senior capstone project, I plan to create a computer program to play chess, which is also called a chess engine. My chess engine will differ from others in that it will be specifically designed to beat other chess engines, not humans. I believe that there is a false assumption in the algorithm that most chess engines use to make decisions, which I plan to exploit.

My chess engine will use several different strategies in order to beat other chess engines, such as non-standard opening moves, varying levels of aggression, an advanced evaluation function, and machine learning to hone its strategies across multiple games.

Background

Chess engines have been around for a long time, with The Turk being the first mechanical chess machine in 1770. The Turk turned out to be an illusion, and a human was actually playing the game, but the idea persisted. Alan Turing created the first algorithmic chess engine in 1948, but since no computer could run it, a human had to perform the calculations according to the algorithm's rules. As digital computers became more and more common, dedicated chess engines became popular, but even a moderate chess player could still beat them. In 1997, a supercomputer created by IBM beat Gary

Kasparov, the greatest human chess player to ever live. This marked a turning point in the evolution of technology, because now a computer could beat a human at something that required planning and creativity. Since this match, the field of computer chess has progressed only slightly, with variations on the common algorithms, but the advancement of computer hardware has accounted for most of the increase in performance. There are many strategies that could be used to beat a computer, and this project will exploit several of them.

Proposed Work

In order to complete this project, I will accomplish the following tasks, in approximate order:

- **Create user interface**

  The first step will be to create a way to talk to the application. There are several programs such as xboard which provides a basic chess interface, along with a communications protocol for the program to interface with.

- **Implement standard chess engine**

  Creating a standard chess engine (designed to play against humans) will provide a walking skeleton of the entire project, in which additions can be added and tested individually.

- **Add custom strategies**

  In this step, my own ideas and strategies will be added to the chess engine. This will make the chess engine optimized to play against other chess engines, not against humans, and will also take up the bulk of the time.

- **Testing and evaluation**

  I will test my program against other chess engines at different levels of difficulty. This step will determine if the program meets its goals or not.

Rationale

Following these steps will help the development process, because it breaks the tasks into small

sections, and provides a walking skeleton on which to add enhancements and customizations. Following these steps in order will also allow them to build upon each other, allow me to focus on only one aspect of the program at a time, and will lead to a better program.

Plan of work

The following steps will be completed in order to finish this project:

1. **Preparation for development (2 hours)**

   I will research into what communication protocol to use and how to have two chess engines play each other. The Go programming language will be used, because of its concurrency features and speed.

2. **Implement user communication protocol for user interface (4 hours)**

   This will provide an abstraction of the interface, and will provide an easy way to deal with input and output. Simple hardcoded moves will be used to test that it is functioning properly.

3. **Implement standard chess engine (10 hours)**

   The next task will be to create a standard chess engine based on established strategies and algorithms. This will provide a walking skeleton for later addition and modification. This step will also set up the model for concurrency and parallel processing.

4. **Add custom strategies to standard chess engine(25 hours)**

   This step will take the most time, and will be the body of the project. Additional strategies and optimizations for playing against a computer will be added. Included in this will be a multi variable evaluation function, with weights assigned to each variable.

5. **Machine learning (15 hours)**

   A simple machine learning algorithm will be implemented to optimize the weights of each variable in the evaluation function, using decision trees.

6. **Testing and evaluation (4 hours)**

   The final step will be to test my chess engine against other programs to see if it is effective.

Testing will occur with many different chess engines, at each of their difficulty levels, and the results will be tracked. This step may take much longer than 4 hours, but most of it will be waiting for the game to finish.

Total estimated time: 60 hours

Timeline

January 12th – 19th

Preparation and interface development

January 19th – 29th

Implement standard chess engine

January 29th – March 2nd

Addition of custom strategies

March 2nd – April 24th

Machine learning

April 24th – May 1st

Testing and evaluation

Preparation for presentation

Evaluation

Evaluation of the success of this project will be pretty straightforward. Since a chess game always resolves to a three way value (win, lose, draw), the key will be to play a large number of games, and then count how many wins and losses each chess engine scored. Many chess engines have difficulty ratings, so each one will be tested at each difficulty level. It will also be trivial to write a script to start a game and then record the results, which would allow thousands of games to be played, which would lead to better data. There are also several competitions for chess engines that I might enter my program into.