Go Interactive Problem Solver

Jude Haris

2191920

Project outline

Motivation

The board game of Go is simple to learn to play but difficult to become good at. The game contains enough complex tactics and strategy that it could takes years maybe even decades to comprehend and to be implemented correctly in one's game play. To study and improve in the game there are large collections of problems created to focus on different phases of the game and improve on it specifically. To help players improve in the game an interactive problem solver would be useful. This would display a problem on the screen and allows the user to try and solve it. The program itself should be able to solve the problem to respond correctly to a user's move.

Aims

The aim of this project is to create such a program which would allow for interactive problem-solving experience. The program needs a graphical user interface to interactive with the user, show the current board position and to make moves. Another aim is to create an AI to play against users in Go, which can choose the correct move given a board position. Specifically, the project's aim was to create a program which given a Go life and death problem it would solve it, if solvable or return as unsolvable.

Progress

- The project is being developed and fully be implemented in Java using the Slick2D GUI framework to create the GUI.
- Good portion of the GUI developed and implemented.
- Implemented the game of Go.
- Implemented editor mode which allows problems to be created.
- Implemented saving and loading of problem files
- Implemented simple features to improve quality of the program such as undo and redo move. Rotate and flip board in the editor mode. Reset the board.
- Implemented Alpha-Beta pruning search to allow the AI to pick the right move to solve the given problem.
- Currently in development of heuristic evaluation of the board to allow depth cut-off to alpha-beta pruning algorithm give a better solution.

Problems and risks

Problems

- Some instance of problems forces the brute force searching to get stuck in infinite loop of moves which leads the program to crash. Possible Solution: re-ordering of moves.
- Implementation of heuristic evaluation has been a difficult and slow process so far
- Success rate of heuristic evaluation depends varying on given boundaries of the problem

Risks

- Lack of board evaluation for different problems. Mitigations: add more simpler heuristics to capture wider problems which maybe not be as effective as more complicated heuristics.
- Not 100% clear on evaluation of the project. Mitigations: plan evaluation more thoroughly and research on how similar programs are evaluated.

Plan

Semester 2

- Week 1-2: Work on heuristic evaluation of the board. Keep adding to it.
- Week 3-5: Start work move generator / reordering of moves
  – Deliverable: First fully featured prototype
- Week 6: Standard unit and system testing of the program. Research on how to evaluate the project.
  – Deliverable: Testing results and evaluation plan.
- Week 7-9: Final implementation sprint.
  – Deliverable: Final product passing basic unit and system testing.
- Week 9: Evaluation, beta testing.
- Week 8-10: Write up.
  – Deliverable: First full draft of submitted to supervisor.