# Adversarial Search in Gomoku

Kapple, Huff, Olds
Southern Oregon University
kapplej@sou.edu, huffm@sou.edu,
oldss@sou.edu

## INTRODUCTION

### What Was Asked

For this assignment, we were put into groups of three and we were asked to write a program that plays Gomoku. The goal of this assignment was to learn how adversarial search through the process of implementing Alpha-Beta pruning. Our goal was to create a program that could compete against other classmates' code in a tournament.

### Objectives

We were given a list of objectives:

- Acquire some experience implementing game search techniques.
- Improve programming skills.
- Have fun.

### Requirements

There were six requirements asked of us:

1. Game-state representation. We needed to decide on our own internal representation of game states. We could use any data structure we wanted as long as it implemented the appropriate game state.
2. Generate moves. We were required to implement functionality that, when given a current game-state, the program can generate all legal moves.
3. Game-state evaluations. We needed to implement a heuristic evaluation function that estimates the utility of a given board state.
4. Alpha-Beta pruning. Implement Alpha-Beta search, this is the heart of the assignment.
5. Manage your time. The program has 2.0 seconds to make a move or it will forfeit the game.
6. Play a game. The tournament involves connecting and playing moves. Connecting is simply establishing a socket connection with the port at the given host.

### Choice of Language

We decided to use java as our language to code the agent since that was the language our group had the most experience with. We all happened to have eclipse and jGRASP installed on our computers, so that is the software we decided to use.

## OUR WORK

Our first objective of the assignment was to get our program to connect through GomokuDriver.java. We got the program to connect but the game instantly ended once our client and the other client connected.

We set up a skype session to talk over where we were at with the code and what our next steps were. We used github.com to so we could share code and used numerous emails to try and problem solve the errors we were getting.

Our next step was to implement a minimax search with MiniMax.java. Minimax.java instantiates a small tree from the text book and runs through it. Node.java implemented a basic tree with n-children. We then added a method to get the information provided by Gomoku and turn it into an arraylist with a 2D char array for the grid and strings for the pieces of information provided.

We then started to implement the Alpha-Beta pruning. We used some pseudo-code to figure out how to write the proper code for the Alpha-Beta. We created an evaluation function in BoardState.java to calculate scores for horizontal rows.

Once we got our code up and running, we decided to evaluate how it would play against the random player. We could beat it most of the time, therefore we decided to fine tune the code to make it so we could have a better chance to beat people in the tournament.

### Expanding

Our first code was only able to work horizontally so we had to spend a fair amount of time debugging AlphaBeta to see where it was failing. Once we got it to work, we then worked on evaluating the board state to add vertical, horizontal, diagonal, and forwards/backwards checks. We had to refactor our evaluate function so we could reuse our switch statement instead of repeating it.

One problem we kept running into was the program would have difficulty if the depth limit was too much. We eventually narrowed it down to 4 because that was all that it could handle.

We also worked on implementing code that would block opponents if they got up to 3 in a row because if they can get 4 in a row with nothing on either end, the opponent wins.

## REFLECTION

Looking back on the lab, we believe as a group we put in countless hours on the project to make the most effective Gomoku player we could. We looked up multiple pages on the internet for help and used the book for references to pseudo code. We were able to create a code that is able to beat other code, so we feel we were successful. Throughout this project we all learned a fair amount about the implementation of game search techniques and further improved our programming skills. This lab has helped us become better programmers and learned that the world of artificial intelligence requires a lot of hard wor