Jeremy Newman
UNI: jrn2144
Professor Levatich
Project Proposal

Parallelizing Connect 4 AI in Haskell

Overview

The purpose of this project is to implement an AI player for the Connect 4 game, and then to parallelize the algorithm to improve the performance. The AI player will use the Minimax algorithm with alpha-beta pruning as its strategy. I will use Haskell's built in parallelism libraries to create performance improvements. The goal is to see how much of a speedup the decision-making algorithm can experience using search-tree parallelism. I will use Threadscope and other measurement tools to quantify the performance across varying numbers of CPU cores/threads.

Background

Connect 4 is a two-player turn based game that is played in a 6x7 grid. Each player is assigned a color and takes turns placing discs into columns that drop all the way down. The objective is to form a line of 4 of your color's discs in a row, either vertically, horizontally, or diagonally. The Minimax algorithm is a common algorithm used by AI in turn-based games and searches the game tree to choose the optimal move. The algorithm assumes that both players will always play optimally. Alpha-beta pruning is a technique that builds upon the Minimax algorithm to eliminate branches of the search tree that will never be chosen. This is an interesting choice of an algorithm as the recursive subtree exploration lends itself well to Haskell's ability to evaluate independent subtrees in parallel.

Approach and Methodology

I will first implement the Minimax algorithm with alpha-beta pruning in Haskell as a sequential algorithm first. I will then add code to the algorithm in order to parallelize the evaluation of optimal move selection. In each game state, there are 7 possible moves that a player can make, and they will all be evaluated in parallel. I will use constructs from the Haskell parallel library such as par and pseq to perform these parallelism improvements. The input will have automated input with predefined board states that represent different positions in the game. I will include positions at early, middle, and late game situations. The format will be a 2D list representing a list of rows of positions with 0 meaning no disc, 1 meaning one player's disc, and 2 meaning the other player's disc.

Performance Measurement

I will be using a combination of timing and Threadscope graphs to measure the performance of the algorithm. I will report speedups across varying numbers of cores/threads as well as across multiple stages of improvement, starting from a sequential algorithm to varying levels of parallelism.