CS 227B Final Report

Paul Mitalipov, Shimea Bridgewater, Ricky Thai

Overall Architecture – Grounded MCTS with special interpreter

Metagaming

During the start clock our player grounds the rules, replacing all of the variables in the rules and creating a new grounded rule set that consists of every combination of those variables. This means that at runtime we are then able to use a special interpreter to more efficiently read a game's rules by using string equality for testing purposes. Ultimately, with our MCTS player, grounding allows it to be able to expand its game tree to greater depths each turn. While there are some drawbacks with grounding, such as always ending up with a larger ruleset compared to ungrounded rules, the cost of sifting through the rules is outweighed by the sheer efficiency of string comparisons.

Development and Testing Methodology

During development, we spent a lot of time modifying the given code for MCS and Greedy, trying to mold the two to get a result of MCTS, which was difficult to get functioning at first. We combined these given players with code taken from lecture notes, and were eventually able to get a functioning player that we tested on simple games such as Hamilton and Alquerque. We spent a lot of time in testing figuring out what was the optimal count of depth charges in combination with updating the tree, while making sure that the player would actually output a move in time. Most of our testing consisted of a lot of console statements to check when depth charges were being dropped and to check the utility of each state.

Development while trying to implement grounding was a bit tricky because we had to figure out the specific syntax in how to actually ground the rules at the beginning of the game. Some posts on Ed were particularly helpful, if not having to make our own post. It took a long time to get grounding to function properly, also because of some errors when using functions given to us, and some parameters being off between given code and what was shown in lecture.

Strengths and Weaknesses

Monte Carlo Tree Search uses simple random probes and statistics to calculate the approximate value of a move. These random probes do not need to spend any time calculating an optimal move at any junction – they pick a random one and move down the tree. This allows MCTS to make inferences about larger portions of the game tree than standard minimax, which must evaluate all moves at each junction and pick the optimal one for whichever player is in control, a process which can take quite a while depending on the depth of the tree or its branching value. The upshot of this is that on games with complex trees, MCTS seems to "think" further ahead than minimax does.

This is particularly noticeable on games whose rules do not give helpful intermediate state values, such as chinese checkers. Without these state values, the minimax family of players tends to flounder at the beginning of chinese checkers because its limited search prevents it from recognizing if it is making progress towards the goal or not. MCTS, on the other hand, is able to send out is inexpensive random probes all the way to the bottom of the tree, and can make quick progress towards the goal.
A weakness of MCTS is that its statistical evaluation of states does not take forced wins seriously. It may make a move that gives the opponent an opportunity for a forced win, or miss a forced win of its own just because the average values of the states after its blunder are higher than the rest. A minimax player lacks this weakness, since it assumes that the opponent will always make the move that is the worst for the player, so it will not choose a move that will allow the opponent to win on their next turn, even if the states that come from the opponent choosing something else have high terminal values.

If there was more time?
If given more time we would have loved to figure out how to implement the other metagaming techniques we were taught in class. Techniques like simplifying and logical optimization, and figuring out how to make all of these techniques work in a seamless manner all while making sure we were ready once the start clock was over. We also would have worked on curbing a lot of the weaknesses that grounding has, especially for games with rule sets too large to ground, by switching to another technique in those cases. Finally, if given more time we think it would have been fun to give our player a bit of a personality by taunting our opponent through messages to the console log (even if only we could see it on our end)!