COMP 424 – Artificial Intelligence

FINAL PROJECT

***HUS***

Implementing a Player

New agents are created by extending the class hus.HusPlayer.

Skeleton is provided by the class student.player.StudentPlayer and you should proceed by directly modifying that file.

Two primary responsibilities:

1. Change the constructor of the StudentPlayer class so that it returns your student number.
2. Change the code in the method chooseMove to implement your agent’s strategy for choosing moves.

Hus Rules:

1. Player selects one of the pits they control that **contains more than 1 seed**, and scoops all the seeds in that pit.
2. The player then begins sowing the scooped seeds, moving **CounterC from the scooped pit,** placing one seed in each pit until running out of seeds. (Players only ever place seeds in their own pits)
3. At the end of sowing 3 things can occur depending on the contents of the last pit sowed (called the end pit)
   1. 3.1 IF the end pit was previously empty, then the player's turn is over.
   2. IF the end pit was previously occupied, and it is in the player's "INNER" row of pits, AND the opponents INNER pit directly opposite is occupied, then a CAPTURE takes place. The player scoops the seeds from both the INNER and OUTER pits directly opposite the endpit and begins sowing using these stolen seeds placing the first seed in the first pit CC from the endpit.
   3. If the endpit was occupied but the conditions for capture are not met, then the player scoops the seeds from the end pit, and begins a round of sowing. This is called relay sowing.
4. The player’s turn continues until a round of sowing ends in a pit that was previously empty (i.e. option 1 (3a) occurs.
5. On the first turn, neither player may capture, if a situation arises where a player would have captured, relay sowing is performed instead.
6. A player wins when it is their opponents turn to play, but their opponent has no valid moves (all their pits contain either 0 or 1 seeds).
7. Draw occurs if neither player has won after 5000 turns.

Evaluation Function:

A key part in designing my AI Agent and its ability to make the best decisions in order to win the game, is coming up with a accurate evaluation function. An evaluation function represents the “goodness” board state or in other words the chance of winning the game from that position. To come up with my evaluation function I need to determine which features of the board I want to look at as being “good”.

1. Maximize number of seeds on my side of the board.
2. If last seed planted comes from previously occupied spot and there are seeds opposite in opponents inner row.
3. If all opponent’s pits contain only 0 or 1 seeds. (then choose this move)

**Current Issues with Hus Project and Actions Taken.**

**ISSUE 1 - on V2**

* Can run my Minimax function with depth = 3 without any problems. --> WIN all games in ~35 moves.
* When I try and run my Minimax function with depth = 4, 6.
* What I don’t understand is that within HusBoardState this code is given:

/\*\* Detect when a player has won. Called at the end of a turn. A player

\* wins when their opponent is about to play but has no legal moves.\*/

private void updateWinner(int next\_to\_play){

if(winner != NOBODY){

return;

}

if(!hasValidMoves(next\_to\_play)){

winner = (next\_to\_play + 1) % 2;

return;

}

if(turn\_number > MAX\_TURN){

winner = DRAW;

}

}

* How am i possibly running games that reach values greater than 5000 turns? Doesn’t make any sense.

Game 188 special case depth 6, worked, won in 19 moves total.

Game 196 worked too

WANT TO IMPLEMENT:

Want to implement whether depth = 0 || if no possible moves.

If no possible moves the game is over and either I have won or lost. If I win then this is definitly the path leading to this move is definitly the one I want to chose. If I lose, definitely don’t want to chose this.

🡪 within MyTools.EvaluationFunction

Do minimax to a certain depth + have monte carlo rollout and see percentage of games won from that.

Incorporate both results into an evaluation function.

Reasons for Monte Carlo,

“Go is a hard game for computers to play: it has a high branching factor, a deep tree, and lacks any known reliable heuristic value function for non-terminal board positions. “ Survey MCTS Methods Pg. 2

Hus has a high branching factor, and given the lack of known reliable heuristic value functions for non-terminal boards is a great strategy for me to use.

Need to figure out my Tree Policy:

Pg 6. Survey MCTS Methods.

These may be grouped into two distinct policies:

1) *Tree Policy*: Select or create a leaf node from the nodes already contained within the search tree (se- lection and expansion).

2) *Default Policy*: Play out the domain from a given non-terminal state to produce a value estimate (sim- ulation).

Tradeoff between rollouts vs. depth.

Realized that at a greater depth, rollouts score is exponentially growing.

Associazione italiana per l'intelligenza artificiale., & Esposito, F. (2001). *AI\*IA 2001: Advances in artificial intelligence : 7th Congress of the Italian Association for Artificial Intelligence, Bari, Italy, September 2001 : proceedings*. Berlin: Springer.