CSE 318
Artificial Intelligence Sessional
Assignment 2
Adversarial Search

The task of this assignment is to implement a game player which uses adversarial search algorithms to play a two player game. The game we shall consider here is Mancala.

Your implementation would include a minimax **algorithm with alpha-beta pruning**. You also need to

- Experiment with different search strategies (e.g., iterative deepening search).
- Experiment by varying depth-limits and changing move-ordering.

Moreover, you need to implement various heuristics and find out which one is better by running experiments. Determine the win-loss ratio by running 100 games with computer vs computer autoplay. Keep the experiment results and show during the evaluation process.

A heuristic function estimates how good a particular state is for a player. A number of factors determine whether a given state of the game is good for a player.

How to Play:

In Mancala, the board consists of six (6) bins on each side, and a home position (called storage) on the right of the bins. The board is laid out typically starting with four stones in each bin and both storage bins empty.

The following link contains a description of the rules of Mancala.

https://www.thesprucecrafts.com/how-to-play-mancala-409424

You can play Mancala online from one of the following links to get acquainted with it.

https://www.mathsisfun.com/games/mancala.html

https://www.mathplayground.com/mancala.html

Mancala Heuristics:

For Mancala, the following strategic factors can be used to design a good heuristic to determine how favorable a particular position is for a player:

1. First valid move [furthest valid bin (a bin on my side which is not empty) from my storage]
2. How far ahead of my opponent I am now [the difference in stone count between my storage and opponent's storage]
3. How close I am to winning (If my storage is already close to containing half of total number of stones)
4. How close opponent is to winning

5. The total number of stones residing in the six bins of my side
6. The total number of stones residing in the six bins of opponent's side
7. Number of stones close to my storage (a stone, although residing in a bin on my side, is not close to my storage, if it is going to be overflowed to my opponent's side)
8. Number of stones close to my opponent's storage
9. Have I earned an extra move
10. Have I captured any stone

You have to experiment with six heuristics including the following three (i.e., define three of your own heuristics):

Heuristic-1: The evaluation function is

(stones_in_my_storage – stones_in_opponents_storage)

Heuristic-2: The evaluation function is

*W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side)*

Heuristic-3: The evaluation function is

*W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side) + W3 * (additional_move_earned)*

How to represent:

At the end, your code will give the user to play with the computer. After each move (either by human or by the computer), you need to show the updated state of the board. You can do it in the normal output console. But it would be better if you can make a small implementation in GUI.

Additional Resources for Adversarial Search:

You can learn more about Adversarial Search in Chapter 5 (5.1 to 5.4) of the book "Artificial Intelligence A modern Approach (Third Edition)" by Stuart Russell and Peter Norvig

Submission Deadline:

December 18, 2021 (Saturday) 11:50 PM