Activity 3: Heuristic Analysis

Name: Amrita Nambiar
Matrics ID: 17173025/1

Introduction

The objective of this project is to develop an adversarial agent to play the “Isolation” game. In this report, I will be discussing about the heuristic used in A* Search for minimax and alphabeta pruning.

Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. In a situation where either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent wins. This project uses a version of Isolation where each agent is restricted to L-shaped movements (like a knight in chess) on a rectangular grid (like a chess or checkerboard). The agents can move to any open cell on the board that is 2- rows and 1-column or 2-columns and 1-row away from their current position on the board. Movements are blocked at the edges of the board (the board does not wrap around), however, the player can "jump" blocked or occupied spaces (just like a knight in chess).

Also, agents will have a fixed time limit for each turn to search for the best move and respond. If the time limit expires during a player's turn, that player forfeits the match, and the opponent wins.For this project, I have modified the code in game_agent.py by adding a heuristic function to it. I included the function in a custom player agent and added it to the list of test agents in tournament.py. The goal is to develop a heuristic that outperforms the default baseline ID_Improved and compare its performance with other team members.

Custom Heuristic

Discussion:

When the player selects the very center position, it forces the player to take it. The center position makes the player have the inside track. It is always better to move to the center position if its depth is 3. The game is set to pick random positions for players. For the current value, the heuristic counts the player's moves, opponent's moves and depth.

Implementation:

```
def custom_score_1(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))

    approx_depth = 49 - len(game.get_blank_spaces())

    center_spaces = [(3, 3)]
    center_value = 0

    if approx_depth == 3:
        if game.get_player_location(player) in center_spaces:
            center_value = 99999

    return float(center_value + my_moves - opponent_moves - approx_depth*0.01)
```

Evaluating Heuristic

To evaluate the effectiveness of the heuristics, tournament.py script is used. The script measures relative performance of player in a round-robin tournament against several other pre-defined agents.

The performance of time-limited iterative deepening search is hardware dependent. Therefore, I used Google Colab to execute my program. Also, to accommodate to the needs and limitations faced by myself and my members, we decided to set the number of matches to 5.

The tournament opponents are listed below:

- Random: An agent that randomly chooses a move each turn.
- MM_Null: CustomPlayer agent using fixed-depth minimax search and the null_score heuristic
- MM_Open: CustomPlayer agent using fixed-depth minimax search and the open_move_score heuristic
- MM_Improved: CustomPlayer agent using fixed-depth minimax search and the improved_score heuristic
- AB_Null: CustomPlayer agent using fixed-depth alpha-beta search and the null_score heuristic
- AB_Open: CustomPlayer agent using fixed-depth alpha-beta search and the open_move_score heuristic
- AB_Improved: CustomPlayer agent using fixed-depth alpha-beta search and the improved_score heuristic
- ID_Improved: CustomPlayer agent using iterative alpha-beta search and the improved_score heuristic
- Student8: CustomPlayer agent using iterative alpha-beta search and the custom_score_1

Results

Agent	Performance (%)	Rank
Improved Score	60.00	5
Custom Heuristic (Amrita)	68.57	2
Custom Heuristic (Dong)	64.25	4
Custom Heuristic (Athar)	70.70	1
Custom Heuristic (Eng)	67.86	3

The table shows the performance of my agent, the default baseline ID_Improved and of my group members.

It can be observed that all the Custom Heuristics performed better compared to ID_Improved. My groupmate, Athar got the highest performance score, with the value of 70.70%. Meanwhile, I got the second highest, with a score of 68.57%. Several reasons my agent scored a fairly high performance, include:

1. The presence of positional advantage. For instance, the player has to take the very center position when they have select it. The center position gives the advantage to the player to have the inside track.
2. It counts depth. Counting depth keeps the play competitive and it's always better to move to the position in the center if the depth is 3.
3. It counts the opponent's move.

Snippets of my evaluation result and the default baseline, ID_Improved can be found in the Appendix section.

Appendix

A. EVALUATION RESULT

```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 16 to 4
  Match 2: ID_Improved vs   MM_Null     Result: 14 to 6
  Match 3: ID_Improved vs   MM_Open     Result: 9 to 11
  Match 4: ID_Improved vs MM_Improved   Result: 8 to 12
  Match 5: ID_Improved vs   AB_Null     Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved   Result: 10 to 10


Results:
----------
ID_Improved         60.00%
```

```
*************************
  Evaluating: Student8
*************************

Playing Matches:
----------
  Match 1:  Student8   vs   Random      Result: 18 to 2
  Match 2:  Student8   vs   MM_Null     Result: 13 to 7
  Match 3:  Student8   vs   MM_Open     Result: 12 to 8
  Match 4:  Student8   vs MM_Improved   Result: 13 to 7
  Match 5:  Student8   vs   AB_Null     Result: 14 to 6
  Match 6:  Student8   vs   AB_Open     Result: 14 to 6
  Match 7:  Student8   vs AB_Improved   Result: 12 to 8


Results:
----------
Student8            68.57%
```