AIND Project: Building a Game Playing Agent

Heuristic analysis

Naruhiko Nakanishi

In this project, the purpose is to develop an adversarial search agent to play the game "Isolation". Agents have a fixed time limit 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. The code in the game_agent.py file is modified to complete the project. Additional files include example Player and evaluation functions, the game board class, and a template to develop local unit tests.

To evaluate the effectiveness of the custom heuristics the tournament.py script is used. The script measures relative performance of the agent. The agent uses time-limited Iterative Deepening along with the custom heuristics. The script controls for these effects by also measuring the baseline performance of an agent called "ID_Improved" that uses Iterative Deepening and the improved_score heuristic defined in sample_players.py. The goal is to develop a heuristic such that Student outperforms ID_Improved. The following three heuristics were used and tested.

Heuristic 1

Here is the implementation for Heuristic 1:

```
def custom_score(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    own_moves = len(game.get_legal_moves(player))
    opp_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 + own_moves - opp_moves - approx_depth*0.01)
```

Analysis:

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, Heuristic 1 counts the player's moves, opponent's moves, depth.

Heuristic 2

Here is the implementation for Heuristic 2:

```
def custom_score_2(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")
    player_legal_moves = game.get_legal_moves(player)
    player_total_moves = len(player_legal_moves)
    for move in player_legal_moves:
        board = game
        board = game.forecast_move(move)
        player_total_moves += len(board.get_legal_moves())
    opp_legal_moves = game.get_legal_moves(game.get_opponent(player))
    opp_total_moves = len(opp_legal_moves)
    for move in opp_legal_moves:
        board = game
        board = game.forecast_move(move)
        opp_total_moves += len(board.get_legal_moves())
    return float(player_total_moves - 2*opp_total_moves)
```

Analysis:

The possibility of limiting future moves for the opponent is focused. The weight of 2 with the opposition legal moves means the penalizing highly for a board state, in which the opponent is fairly limited in it's movements.

Heuristic 3

Here is the implementation for Heuristic 3:

```
def custom_score_3(game, player):
    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
```

```
        return float("inf")
    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    app_depth = 50 - len(game.get_blank_spaces())
    return float(own_moves - opp_moves - app_depth*0.01)
```

Analysis:

When the depth between moves are different and the scores is same, the move which has not deeper would rather be chosen, because it can finish the game earlier. In case that some moves have same player's moves and opponent's moves, the lower depth would rather selected because it increases the chance of the winning by finishing the game early. Heuristic 3 counts the player's moves, opponent's moves, and the depth for the current value.

Performance

Of the three heuristics, Heuristic 1 performed the best.

```
                      *************************
                          Playing Matches
                      *************************

Match #   Opponent    AB_Improved    AB_Custom    AB_Custom_2   AB_Custom_3
                      Won  | Lost    Won  | Lost   Won  | Lost   Won  | Lost
    1       Random     10  |   0     10  |   0     9  |   1     9  |   1
    2       MM_Open     5  |   5      8  |   2     6  |   4     6  |   4
    3      MM_Center    9  |   1      9  |   1     8  |   2    10  |   0
    4     MM_Improved   6  |   4      7  |   3     7  |   3     8  |   2
    5       AB_Open     5  |   5      6  |   4     5  |   5     5  |   5
    6      AB_Center    5  |   5      7  |   3     7  |   3     6  |   4
    7     AB_Improved   7  |   3      4  |   6     5  |   5     3  |   7
--------------------------------------------------------------------------
          Win Rate:      67.1%        72.9%        67.1%        67.1%
```

Recommendation

Heuristic 1 is recommended. The reasons for the recommendation are as follows. 1) There is positional advantage. 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. 2) Counting depth keeps the play competitive. It is always better to move to the center position if its depth is 3. 3) Heuristic 1 counts for opponent's move.