Heuristic Analysis
By: Eric Iacutone

Custom Heuristics

- Sorted by best score descending

Score: 63.9%

```
def percent_occupied(game):
    blank_spaces = game.get_blank_spaces()
    return int((len(blank_spaces)/(game.width * game.height)) * 100)

w, h = game.width / 2., game.height / 2.
y, x = game.get_player_location(player)

if percent_occupied(game) < 75:
    return float(2*((h - y)**2 + 2*(w - x)**2))
else:
    return float(-1*(((h - y)**2 + 2*(w - x)**2)))
```

Score: 62.1%

```
def percent_occupied(game):
    blank_spaces = game.get_blank_spaces()
    return int((len(blank_spaces)/(game.width * game.height)) * 100)

w, h = game.width / 2., game.height / 2.
y, x = game.get_player_location(player)

if percent_occupied(game) < 50:
    return float(2*((h - y)**2 + 2*(w - x)**2))
else:
    return float(-1*(((h - y)**2 + 2*(w - x)**2)))
```

Score: 62.0%

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

return float(2*own_moves - opp_moves)
```

Score: 61.6%

```
    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    return float(own_moves*own_moves - 1.5*opp_moves*opp_moves)
```

Improved Avg Score: 60.9% after ~160 matches

Reasoning

I experimented with many different heuristics for my analysis. Many ideas are derivatives of the center and 'chase' heuristics given in the AIND-Isolation repository. I altered the weights for both the center and 'chase' heuristics in order to determine a good mix of weights to use in the custom heuristics. At first, I made functions in order to determine if the player was close to either the walls or corners and then assigning a score to these positions. These functions were computationally intensive because I needed to loop through both players and then check and assign a score to each player. The center heuristic is ostensibly performing the same wall/corner check. So, I discarded the wall/corner functions and focused on modifying the center heuristic.

After watching the instructive Isolation Agent Competition video, I noticed a pattern emerge. A center move is advantageous at the beginning of the game, while a move near the walls and corners is advantageous near the end of the game. I developed a `percent_occupied` helper method in order to calculate the remaining blank spaces on the board.

With the `percent occupied` helper function, I developed my best heuristic. If the board is more less than 75% occupied, a positive score is given to moves near the center, else a negative score is returned. I recommend using the heuristic listed above with a win rate of 63.9%. This heuristic had the best win rate. Furthermore, it is not a computationally expensive and therefore is fast to compute. Finally, it is easy to interpret the heuristic.

Additional Information

- The `tournament.py` NUM_MATCHES = 40 and TIME_LIMIT = 450
- Processor: 3.1 GHz Intel Core i7
- Memory: 16 GB 1867 MHz DDR3
- Graphics: Intel Iris Graphics 6100 1536 MB