**Heuristic Evaluations in Competitive Tournaments**

**Q:** For each of your **three** custom heuristic functions, ***evaluate*** *the performance of the heuristic* using the included **tournament.py** script.

A: As seen in the table below, the **AB\_Custom, AB\_Custom\_2**, and **AB\_Custom\_3** heuristics performed at a level of **60%**, **60%** and **75.1%** win rate respectively when engaged in the tournament as opposed to the **AB\_Improved** Agent which performed at a **65.7%** win rate.

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **Playing Matches**

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

# Match # Opponent AB\_Improved **AB\_Custom AB\_Custom\_2 AB\_Custom\_3**

# Won | Lost Won | Lost Won | Lost Won | Lost

# 1 Random 7 | 3 9 | 1 9 | 1 8 | 2

# 2 MM\_Open 8 | 2 8 | 2 7 | 3 6 | 4

# 3 MM\_Center 7 | 3 6 | 4 6 | 4 6 | 4

# 4 MM\_Improved 8 | 2 6 | 4 4 | 6 6 | 4

# 5 AB\_Open 5 | 5 4 | 6 5 | 5 5 | 5

# 6 AB\_Center 7 | 3 5 | 5 7 | 3 3 | 7

# 7 AB\_Improved 4 | 6 4 | 6 4 | 6 6 | 4

# --------------------------------------------------------------------------

# Win Rate: 65.7% **60.0% 60.0% 57.1%**

**Q**: **Then write up a brief summary of your results**, describing the performance of the agent using the different heuristic functions verbally and using appropriate visualizations.

**Custom Score Agent**

* **Name:** **heuristic\_square\_difference\_available\_moves**
* **Description:** Outputs a score equal to the square of the difference in the number of moves available to the two players.
* **Commentary:** The concept was to experiment with derivations of the difference between the number of moves left between players. The "Improved" evaluation function did well by only taking the simple difference, so the thought was to use something related but different (like the square) and see if any improvements could be made.
* **Evaluation:**  This heuristic actually did OK (60% win rate) but not as well as the heuristic in which it was modeled after.
* **Code:**

own\_moves = len(game.get\_legal\_moves(player))

opp\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

return float((own\_moves - opp\_moves)\*2)



**Custom Score 2 Agent**

* **Name:** **heuristic\_player\_moves\_vs\_player\_distance\_to\_center**
* **Description:** Outputs a score equal to the difference in the number of moves available to the two players multiplied by the active players distance to the center of the board.
* **Commentary:** Remembering that the lessons argued that the first choice being in the center of the board was an excellent one, and the **center\_score** heuristic in **sample\_players.py** calculated it's heuristic relative to the center of the board - the idea was to relate the moves remaining to their distance to the center of the board.
* Evaluation: This heuristic did as well as the first heuristic (60% win rate) - which is interesting.
* **Code:**

# Get the num of moves left for both players and take the difference

own\_moves = len(game.get\_legal\_moves(player))

opp\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

moves\_diff = own\_moves - opp\_moves

# Get the location of the player relative to the center of the board

w, h = game.width / 2., game.height / 2.

y, x = game.get\_player\_location(player)

dist\_to\_center = ((h - y)\*\*2 + (w - x)\*\*2)

return float(moves\_diff \* dist\_to\_center)



**Custom Score 3 Agent**

* **Name:** **heuristic\_player\_opponent\_vs\_center\_distance**
* **Description:** Outputs a score equal to square of the distance from the center of the board relative to the difference of the position of the player vs opponent.
* **Commentary:** This heuristic was an experiment to see if there was any correlation between the location on the board where the players were operating and the center of the board. The heuristic performed alright, better than average at 57.1% win rate, but not as well as the others.
* Code:

**# Get the dimmensions of the Board**

**w, h = game.width / 2., game.height / 2.**

**# Get Player Location**

**py, px = game.get\_player\_location(player)**

**# Get Opponents location**

**oy, ox = game.get\_player\_location(game.get\_opponent(player))**

**# Return the heuristic calculation**

**return float( (h - abs(py -oy))\*\*2 + (w - abs(px - ox))\*\*2 )**



**Evaluation Function Recommendation**

The evaluation function recommendation for usage is **AB\_Custom\_2** for the following reasons:

1. In the table below, the opponents get smarter down the list (1 being the easiest opponent, who plays a random game), and the heuristic AB\_Custom\_2 wins slightly more games with the more difficult opponents - see highlighted scores below.
2. The heuristic AB\_Custom\_2 performs at the same level **overall** as the AB\_Custom heuristic
3. The heuristic AB\_Custome\_2 performs at the same level as AB\_Custom with the **easier** opponents

