

# **Udacity**

## **Artificial Intelligence (AI) Nanodegree**

### **Project 3 -**

### **Build an Adversarial Game Playing Agent**


**Arka Ghosh**

**October 11**

# Summary


In this project, I have chosen Option 1: Developing a custom heuristic. The performance/effectiveness of the custom heuristic has also been evaluated by comparing it with a baseline heuristic. I have also combined minimax search with alpha-beta pruning and iterative deepening as explained in the lectures.

**Baseline Heuristic:** This heuristic is based upon a simple formula, which is *the difference between 'our moves left' and 'opponent moves left'*.




```
1 def baseline(self, state):
2     return len(state.liberties(state.locs[self.player_id])) - \
3           len(state.liberties(state.locs[1-self.player_id]))
```

**Central Heuristic:** This heuristic takes into account the sum of the straight line (euclidean) distance between the current locations of each player and the centre-most cell of the board [5, 4]. This assumes that as closer to the centre of the board the player is, the more their chances of winning, since they have much more free cells to explore in more directions, as opposed to edges, where one side is completely blocked off.



```
1 def central(self, state):
2     pos_1 = state.locs[self.player_id]
3     pos_2 = state.locs[1-self.player_id]
4
5     x1, y1 = (pos_1%(11+2), pos_1//(9+2))
6     x2, y2 = (pos_2%(11+2), pos_2//(9+2))
7
8     # center coordinates: [(11-1)/2, (9-1)/2] = [5, 4]
9     return -(x1-5)**2 - (y1-4)**2 + (x2-5)**2 + (y2-4)**2
```

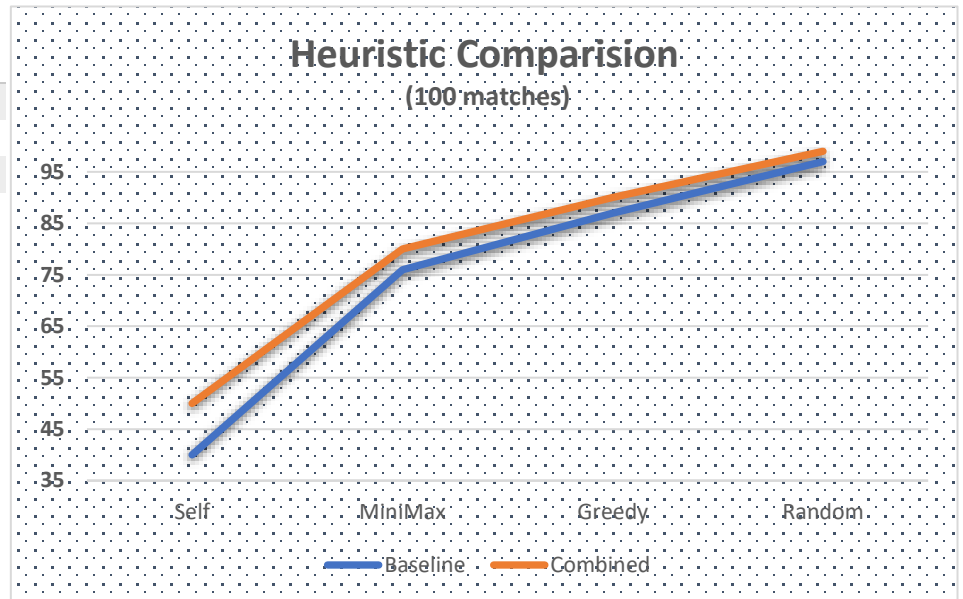
**Custom Heuristic (Combined):** This heuristic considers both the above-mentioned heuristics, by assigning certain weights to them individually and adding them up together. The value obtained from the baseline heuristic is multiplied by 0.4, and the value obtained by the central heuristic is multiplied by 1.2, and they are added up together to get a new heuristic. This heuristic will be in a sense better than the previous 2 since it derives its values from and is dependent on both of them. The values 0.4 and 1.2 were chosen by initially selecting them as 1 and then gradually changing them randomly to check which values give a higher success rate.



```
1 # combining both heuristics to create a final heuristic
2 def combined(self, state):
3     return 0.4 * self.baseline(state) + 1.2 * self.central(state)
```

# Visualization

Opponent	Baseline (%)	Custom (%)
Self	40	50
MiniMax	76	80
Greedy	87	90
Random	97	99



**Observation:** The combined heuristic, outperforms the baseline heuristic.

## Q&A

**1. What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?**

The heuristic includes the liberties accessible to each player during each turn, along with their proximity to the central cell of the board. The final combined heuristic value is determined by multiplying each sub-heuristic by constant values of 0.4 and 1.2, respectively, to enhance performance. The combined significance of these features is evident: the baseline heuristic decreases the number of available liberties at each iteration, while the central heuristic calculates the sum of the Euclidean distances between both players' cells and the centre of the board. This is crucial, as players have greater access to open cells from the centre compared to the edges, where one side is entirely obstructed. The final heuristic is more sophisticated as it incorporates both calculated values.

**2. Analyze the search depth your agent achieves using your custom heuristic. Does search speed matter more or less than accuracy to the performance of your heuristic?**

After examining the agent's depth, I discovered that the lowest depth attained was 8 and the maximum depth reached was 18. When the time limit was decreased to 100ms, the agent's success rate remained consistent with that of 150ms, achieving 90% while operating in greedy mode with fair matches for 20 games. Upon raising the duration to 500 milliseconds, the success rate rose to 95%. This suggests that the precision of the heuristic is contingent upon its velocity and the established time constraints. Given the heuristic's elevated success rates, I attempted to augment the number of games to 100 to evaluate the revised success rates.