Artificial Intelligence Nanodegree

Build an Adversarial Game Playing Agent

NGUYEN Duc Huy

1. Introduction

In this project, I will experiment with adversarial search techniques by building an agent to play knights Isolation. Unlike the examples in the lecture where the players control tokens that move like chess queens, this version of Isolation gives each agent control over a single token that moves in L-shaped movements--like a knight in chess

2. Numerical study

In the experiments, I followed the Option 1 to implement and evaluate my agent.

Option 1: Develop a custom heuristic (must not be one of the heuristics from lectures, and cannot only be a combination of the number of liberties available to each agent)

Firstly, I combine the techniques from the lectures together (MinMax search, Alpha-Beta Pruning, and Depth Limited Search). Then, I perform the baseline of your agent using the heuristic from lecture:

#my_moves - #opponent_moves

Next, I used the same process to evaluate the effectiveness of my agent using your own custom heuristic. The heuristics are given as follows:

Baseline: *#my_moves - #opponent_moves*

Custom1: 2**#my_moves - #opponent_moves*

Custom2: *#my_moves - 2*#opponent_moves*

Custom3: *#my_moves - 0.5*overlap_moves - #opponent_moves*

Custom4: *#my_moves - 0.5*overlap_moves + #opponent_moves*

Custom5: *# ply_count* #my_moves – 2.5*#overlap_moves - #opponent_moves*

Custom6: *#my_moves – 2.5*#overlap_moves - # ply_count* #opponent_moves*

Note that:

- *#ply_count:* Cumulative count of the number of actions applied to the board
- *#overlap_moves:* common moves (liberties) between my opponent and me
- *Depth is set as 5*

3. Results

Performance Baseline Table (Baseline vs other opponents)

Opponents	# Matches	Time Limits (ms)	Wining rate (%)
RANDOM	200	1500	98.5
GREEDY	*200*	*1500*	*100.0*
MINMAX	*200*	*1500*	*46.0*
SELF	*200*	*1500*	*50.0*

Effectiveness Evaluation of my agent

(Time Limits =1500ms, #Matches = 200, against MINMAX player)

Heuristics	Winning rate (%)
Baseline	46.0
Custom1	48.0
Custom2	46.5
Custom3	51.5
Custom4	48.5
Custom5	53.0
Custom6	49.5

The above table shows that the Custom5 seems to be the best choices.

4. Discussion:

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

In my opinion, the remaining liberties is one of the most important features because the more liberties player has, the more chance he has to win this game. That is why I implement heuristics 2, 3, 4

Other features are the common liberties (moves) and the cumulative count of the number of actions applied to the board. These features correspond to the reduction of opponents' future moves.

b. 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?

The performance increased slightly when we augmented the depth, but the running time also increased significantly. It is a trade-off between performance and search speed.

DEPTH	**Run Time (s)**	**Wining rate (%)**
2	80	19.8
3	*98*	*43.0*
4	*120*	*45.2*
5	*223*	*46.0*