HEURISTIC ANALYSIS

Adversarial Game Playing Agent for Isolation

Submitted by

Muhamad Amin Hakim bin Sazali

ARTIFICIAL INTELLIGENCE GAME PROGRAMMING: ACTIVITY 3

TABLE OF CONTENTS

SYNOPSIS

In this project, I need to develop an adversarial search agent to play the game "Isolation". Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent is declared the winner. These rules are implemented in the isolation.Board class provided in the repository.

This project uses a version of Isolation where each agent is restricted to L-shaped movements (like a knight in chess) on a rectangular grid (like a chess or checkerboard). The agents can move to any open cell on the board that is 2-rows and 1-column or 2-columns and 1-row away from their current position on the board. Movements are blocked at the edges of the board (the board does not wrap around), however, the player can "jump" blocked or occupied spaces (just like a knight in chess).

Additionally, agents will 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. In my case, I use 150 milliseconds for agent to search for the next move.

CUSTOM HEURISTICS

1. HEURISTIC 1: WEIGHTED COMBINATION OF HEURISTICS (

Can be mathematically expressed as:

$$\frac{len(my\ available\ moves)}{len(available\ opponent\ moves)}, \quad \frac{len(available\ opponent\ moves)}{len(my\ available\ moves)} - \alpha \qquad where\ \alpha \in (1, \infty)$$

Maximizing above equation is equivalent to maximizing:

$$[len(my\ available\ moves)]^2 - \beta\ [len(available\ opponent\ moves)]^2, \qquad where\ \beta \in (1, \infty)$$

The latter form has been implemented in the code with β chosen as 1.5 empirically.

2. CUSTOM HEURISTIC 1: DIFFERENT IN PLAYER MOVES AND OPPONENTS

The heuristic is based on the logic that player should have more next moves in comparison to opponent. In this heuristic, I increase opponent's move using factor of 2. It can be mathematically expressed as:

$$len(\text{total player moves}) - 2{*}len(\text{total } opponent\ moves) \qquad where\ \alpha \in (1, \infty)$$

3. CUSTOM HEURISTIC 2: DISTANCE BETWEEN PLAYER AS A REWARD

The heuristic is based on the logic that player should move further away from the opponents. Distance is calculated using x and y coordinates of player and opponent. Distance is use as reward player for being further away.
Height and width of triangle use Pythagorean theorem to get distance. It can be mathematically expressed as:

$$len(available\ \text{player}\ moves) + (\text{distance player to opponent }) - len(available\ opponent\ moves)$$

EVALUATING HEURISTICS

The tournament.py script is used to evaluate the effectiveness of heuristic. The script measures relative performance of player in a round-robin tournament against several other pre-defined agents.

The performance of time-limited iterative deepening search is hardware dependent (faster hardware is expected to search deeper than slower hardware in the same amount of time). Hence, I use Google Colab to run the tournament.

The tournament opponents are listed below:

- Random: An agent that randomly chooses a move each turn.
- MM_Null: CustomPlayer agent using fixed-depth minimax search and the null_score heuristic
- MM_Open: CustomPlayer agent using fixed-depth minimax search and the open_move_score heuristic
- MM_Improved: CustomPlayer agent using fixed-depth minimax search and the improved_score heuristic
- AB_Null: CustomPlayer agent using fixed-depth alpha-beta search and the null_score heuristic
- AB_Open: CustomPlayer agent using fixed-depth alpha-beta search and the open_move_score heuristic
- AB_Improved: CustomPlayer agent using fixed-depth alpha-beta search and the improved_score heuristic
- ID_Improved: CustomPlayer agent using iterative alpha-beta search and the improved_score heuristic
- **Custom Heuristics 1: CustomPlayer 1 agent using iterative alpha-beta search and the Custom Heuristics 1**
- **Custom Heuristics 2: CustomPlayer 2 agent using iterative alpha-beta search and the Custom Heuristics 2**
-

My team and I agree to use 10 matches as a comparison. This is because we have limited computing resources to run the experiments. The paper use 2000 matches because they have the computing resources, and they want to get the most accurate results.

RESULTS

The performance of various agents is as follow:

Agent	Performance	Rank
ID_Improved (Baseline)	61.79%	6
Custom Heuristics 1 (Amin)	66.79%	5
Custom Heuristics 2 (Amin)	67.89%	3
Custom Heuristics (Alif)	68.21%	2
Custom Heuristics (Akif)	67.86%	4
Custom Heuristics (Fadelic)	79.64%	1

All the custom heuristics perform better than ID_Improved by a reasonable margin as can be seen in the above table.

The table also includes the heuristic performance of my team members.

I use ID_Improved as a baseline method as a comparison. Plus, I have prepared 2 heuristic function with the mathematical formula stated on page 3. The comparison table also includes the results from my team member. We share our results on group chat and discuss what can be improved more. Fadelic managed to get the highest score with a win rate of 79.64%.

The raw evaluation result can be found in A. Appendix: Evaluation Result.

APPENDICES

A. APPENDIX: EVALUATION RESULT

```
This script evaluates the performance of the custom heuristic function by
comparing the strength of an agent using iterative deepening (ID) search with
alpha-beta pruning against the strength rating of agents using other heuristic
functions.  The `ID_Improved` agent provides a baseline by measuring the
performance of a basic agent using Iterative Deepening and the "improved"
heuristic (from lecture) on your hardware.  The `Student` agent then measures
the performance of Iterative Deepening and the custom heuristic against the
same opponents.


*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random        Result: 30 to 10
  Match 2: ID_Improved vs   MM_Null       Result: 32 to 8
  Match 3: ID_Improved vs   MM_Open       Result: 19 to 21
  Match 4: ID_Improved vs MM_Improved     Result: 18 to 22
  Match 5: ID_Improved vs   AB_Null       Result: 26 to 14
  Match 6: ID_Improved vs   AB_Open       Result: 22 to 18
  Match 7: ID_Improved vs AB_Improved     Result: 26 to 14


Results:
----------
ID_Improved         61.79%


*************************
Evaluating: Custom Heuristics 1
*************************

Playing Matches:
----------
  Match 1: Custom Heuristics 1 vs   Random      Result: 35 to 5
  Match 2: Custom Heuristics 1 vs   MM_Null     Result: 28 to 12
  Match 3: Custom Heuristics 1 vs   MM_Open     Result: 27 to 13
  Match 4: Custom Heuristics 1 vs MM_Improved   Result: 20 to 20
  Match 5: Custom Heuristics 1 vs   AB_Null     Result: 25 to 15
  Match 6: Custom Heuristics 1 vs   AB_Open     Result: 26 to 14
  Match 7: Custom Heuristics 1 vs AB_Improved   Result: 26 to 14
```

```
Results:
----------
Custom Heuristics 2     66.79%

************************
Evaluating: Custom Heuristics 2
************************

Playing Matches:
----------
  Match 1: Custom Heuristics 2 vs   Random         Result: 33 to 7
  Match 2: Custom Heuristics 2 vs   MM_Null        Result: 34 to 6
  Match 3: Custom Heuristics 2 vs   MM_Open        Result: 26 to 14
  Match 4: Custom Heuristics 2 vs MM_Improved      Result: 21 to 19
  Match 5: Custom Heuristics 2 vs   AB_Null        Result: 32 to 8
  Match 6: Custom Heuristics 2 vs   AB_Open        Result: 24 to 16
  Match 7: Custom Heuristics 2 vs AB_Improved      Result: 20 to 20


Results:
----------
Custom Heuristics 5     67.86%
```