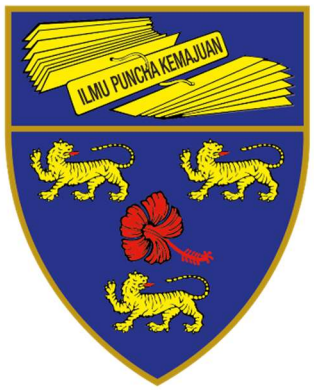

UNIVERSITY OF MALAYA

Heuristic Analysis Report

For an Adversarial Game Playing Agent for Isolation

Name: Aqiff Mursyideen bin Shamsul Safuan

Matric Number: WID170005 @ 17141207/1

Lecturer: Professor Dr. Loo Chu Kiong

Table of Contents

Synopsis

The project aims at developing an adversarial search agent to play the game "Isolation". This project report focusses on the heuristics to be used in A* Search for minimax and alphabeta pruning. 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. 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. These rules are implemented in the isolation. Board class provided in the repository.

Custom Heuristics

1. Default Heuristic

The heuristic is based on the logic that the difference between the player's moves and opponent's moves. It can be mathematically as:

$$len(my\ available\ moves) - len(available\ opponent\ moves)$$

2. Custom Heuristic

The heuristic is based on the logic that for each player and opponent's moves are considered. First, we will iterate for each move for player and make it move and increment the initial total moves of the player with legal moves of current position. The action is then repeated for the opponent moves with its initial total. At the end, the logic used can be mathematically expressed as:

For each player moves, we will use game.forecast_move(player):

$$len(player\ total\ moves) = len(player\ total\ moves) + len(player\ legal\ moves)$$

It is then repeated with the opponent, where for each opponent moves, we will use game.forecast_move(opponent):

$$len(opponent\ total\ moves) = len(opponent\ total\ moves) + len(opoonent\ legal\ moves)$$

Maximizing above equation is equivalent to maximizing:

$$len(player\ total\ moves) - a\ len(opponent\ total\ moves) + len(current\ available\ moves)$$

The value of a was empirically chosen as 2

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). The script controls for these effects by also measuring the baseline performance of an agent called "ID_Improved" that uses Iterative Deepening and the improved_score heuristic from sample_players.py.

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
- Student1: CustomPlayer agent using iterative alpha-beta search and the heuristic 1
- Student2: CustomPlayer agent using iterative alpha-beta search and the heuristic 2
- Student3: CustomPlayer agent using iterative alpha-beta search and the heuristic 3
- Student4: CustomPlayer agent using iterative alpha-beta search and the heuristic 4
- Student5: CustomPlayer agent using iterative alpha-beta search and the heuristic 5
- Student6: CustomPlayer agent using iterative alpha-beta search and the heuristic 6
- Student7: CustomPlayer agent using iterative alpha-beta search and the heuristic 7
- my_custom: CustomPlayer agent using iterative alpha-beta search and the Custom Heuristic

Since, running only a few matches gave different results, the number of matches is set to 5, with all group members agreed to it because faster execution time. Timeout value is 150 which is the the default value

Results:

Results are compared between the baseline default (ID_Improved), 7 students and 3 of my group members. The results as follow:

Agent	Performance	Rank	Rank (Group members)
ID_Improved	65.71%	3	
Student1	62.14%	6	
Student2	65.71%	4	
Student3	62.14%	7	
Student4	62.14%	8	
Student5	60.00%	9	
Student6	75.00%	1	
Student7	63.57%	5	
my_custom	71.43%	2	1
Group Member 1 (Fawwaz)	65.71%		2
Group Member 2 (Lukman)	64.29%		4
Group Member 3 (Syafik)	65.00%		3

The custom heuristic performs better than Id_Improved by a reasonable margin which can be seen in the table. Heuristic 6 outperforms all other heuristic algorithm including me and my group members. There's slight percentage difference between my_custom and heuristic 6.

Appendices

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: 19 to 1

Match 2: ID_Improved vs MM_Null Result: 12 to 8

Match 3: ID_Improved vs MM_Open Result: 10 to 10

Match 4: ID_Improved vs MM_Improved Result: 10 to 10

Match 5: ID_Improved vs AB_Null Result: 17 to 3

Match 6: ID_Improved vs AB_Open Result: 13 to 7

Match 7: ID_Improved vs AB_Improved Result: 11 to 9

Results:

ID_Improved 65.71%

****

Evaluating: Student1

****

Playing Matches:

Match 1: Student1 vs Random Result: 20 to 0

Match 2: Student1 vs MM_Null Result: 13 to 7

Match 3: Student1 vs MM_Open Result: 8 to 12

Match 4: Student1 vs MM_Improved Result: 7 to 13

Match 5: Student1 vs AB_Null Result: 16 to 4

Match 6: Student1 vs AB_Open Result: 11 to 9

Match 7: Student1 vs AB_Improved Result: 12 to 8

Results:

Student1 62.14%

****

Evaluating: Student2

****

Playing Matches:

Match 1: Student2 vs Random Result: 17 to 3

Match 2: Student2 vs MM_Null Result: 16 to 4

Match 3: Student2 vs MM_Open Result: 10 to 10

Match 4: Student2 vs MM_Improved Result: 8 to 12

Match 5: Student2 vs AB_Null Result: 14 to 6

Match 6: Student2 vs AB_Open Result: 16 to 4

Match 7: Student2 vs AB_Improved Result: 11 to 9

Results:

Student2 65.71%

****

Evaluating: Student3

****

Playing Matches:

Match 1: Student3 vs Random Result: 14 to 6

Match 2: Student3 vs MM_Null Result: 13 to 7

Match 3: Student3 vs MM_Open Result: 14 to 6

Match 4: Student3 vs MM_Improved Result: 8 to 12

Match 5: Student3 vs AB_Null Result: 17 to 3

Match 6: Student3 vs AB_Open Result: 12 to 8

Match 7: Student3 vs AB_Improved Result: 9 to 11

Results:

Student3 62.14%

****

Evaluating: Student4

****

Playing Matches:

Match 1: Student4 vs Random Result: 19 to 1

Match 2: Student4 vs MM_Null Result: 13 to 7

Match 3: Student4 vs MM_Open Result: 13 to 7

Match 4: Student4 vs MM_Improved Result: 8 to 12

Match 5: Student4 vs AB_Null Result: 13 to 7

Match 6: Student4 vs AB_Open Result: 10 to 10

Match 7: Student4 vs AB_Improved Result: 11 to 9

Results:

Student4 62.14%

****

Evaluating: Student5

****

Playing Matches:

Match 1: Student5 vs Random Result: 18 to 2

Match 2: Student5 vs MM_Null Result: 14 to 6

Match 3: Student5 vs MM_Open Result: 8 to 12

Match 4: Student5 vs MM_Improved Result: 8 to 12

Match 5: Student5 vs AB_Null Result: 16 to 4

Match 6: Student5 vs AB_Open Result: 11 to 9

Match 7: Student5 vs AB_Improved Result: 9 to 11

Results:

Student5 60.00%

****

Evaluating: Student6

****

Playing Matches:

Match 1: Student6 vs Random Result: 17 to 3

Match 2: Student6 vs MM_Null Result: 16 to 4

Match 3: Student6 vs MM_Open Result: 13 to 7

Match 4: Student6 vs MM_Improved Result: 10 to 10

Match 5: Student6 vs AB_Null Result: 19 to 1

Match 6: Student6 vs AB_Open Result: 16 to 4

Match 7: Student6 vs AB_Improved Result: 14 to 6

Results:

Student6 75.00%

****

Evaluating: Student7

****

Playing Matches:

Match 1: Student7 vs Random Result: 17 to 3

Match 2: Student7 vs MM_Null Result: 14 to 6

Match 3: Student7 vs MM_Open Result: 12 to 8

Match 4: Student7 vs MM_Improved Result: 9 to 11

Match 5: Student7 vs AB_Null Result: 13 to 7

Match 6: Student7 vs AB_Open Result: 14 to 6

Match 7: Student7 vs AB_Improved Result: 10 to 10

Results:

Student7 63.57%

****

Evaluating: my_custom

****

Playing Matches:

Match 1: my_custom vs Random Result: 16 to 4

Match 2: my_custom vs MM_Null Result: 18 to 2

Match 3: my_custom vs MM_Open Result: 13 to 7

Match 4: my_custom vs MM_Improved Result: 12 to 8

Match 5: my_custom vs AB_Null Result: 15 to 5

Match 6: my_custom vs AB_Open Result: 13 to 7

Match 7: my_custom vs AB_Improved Result: 13 to 7

Results:

---

my_custom 71.43%