The Polarized Ladder Game

COMP6721 Introduction to Artificial Intelligence

Project Report

Ede Cameron 2952270

Dimitri Tiago 3978227

Concordia University, April 11 2013

Program Description

The Polarized Ladder game is a turn based game similar to Tic Tac Toe or Connect Four. The program is designed for three possible game scenarios, Human versus Human, Computer versus Human, and Human versus Computer , the later two really the same with the first and second player reversed. The object of the game is to build a ladder consisting of five points with four connected edges, the first one to do so wins the game. Much like Tic Tac Toe the objective of a defensive strategy is to block or stop the other player from building a ladder. This can be done by blocking an opponent's move or polarizing the opponent's ladder.

The game is programmed in Java and there are two methods to run the game one is to double click on the jar file in the main folder PolarizedLadder and the other in the PLGame folder is there to compile the game using javac both methods are described in the README.txt located in the main directory of this submission, PolarizedLadder. The game is played and runs in the terminal (Linux, Mac OS X), or the command prompt in Windows. There is no GUI the user simply inputs point coordinates into the command line, for example 1A. If playing against the computer after a turn is inputted by the human player the computer automatically takes a move. If playing against another person the prompt shows the next player turn, or error messages if the player makes an invalid move.

There are four main sections to the game there is the Player (Player.java), a human Player and the Artificial Intelligence Player (AIPlayer.java) which represents the computer player. Both Human and Computer follow the basic rules of the game. That is placing discs at allowed locations, following the turn based rules of each player playing after the other. Winning ladders and Polarized Rules are the same for both human and computer. The Human Player incorporates little more than the rules of the game and receives input from the player through the prompt, and other two sections of the game are designed to specifically run the AI Player.

The Heuristics of the game are used to measure the value of each possible move the computer could take from a specific state in the game, and the MiniMax Algorithm defines the way in which these heuristics affect moves from both player (max) and opponent (min) perspectives. The data structures used for the MiniMax Algorithm is an n-ary tree whose leaf size depends on the number of possible moves in the present state of the game, and a hash table which is used to contain the points that haven't been played yet.

The Heuristics of the game are found in the class LadderPatternStrategies and are described in detail in Section 2. The AIPlayer is the primary class for the calculation of each turn for the AI Player. It is in this class that the search tree is created for each turn and the calculations of the heuristics are called. The primary methods are move(..), createStateSpace(..) which builds the Search tree and calls the recursive ethod miniMaxMove() which depending on the depth (even or odd) calls getMinMove(..) or getMaxMove(..) in the MiniMaxAIPlayer Class.

The MiniMax Algorithm is the standard algorithm where initial heuristics are calculated on the leaves and then a recursive min or max value calculated from the leaves to the root. The moves are calculated using an open list (openList in the Class SearchLists) which uses a hash table of Points (keys) and a string (the value) stating whether the point is open. After a move has been completed the Point is removed from the open list in order to avoid redundant searches.

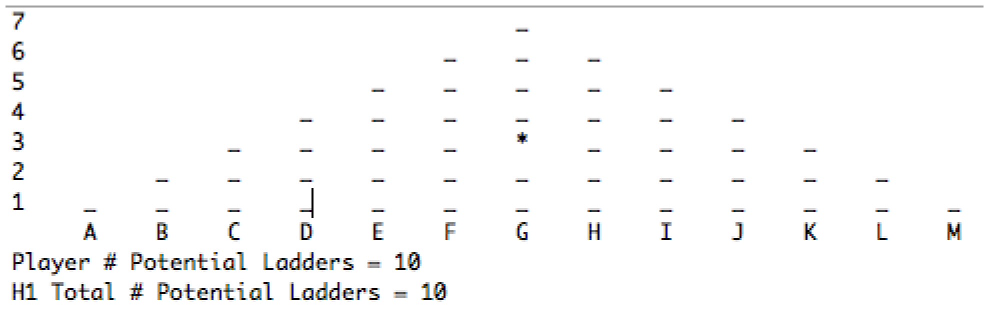
When a move is executed by the AI it gets the state of the current board, it then builds a tree where nodes represent all next possible moves. This at the beginning of the game means that in three seconds (as required from the outline of the game), the tree can only be searched to depth three as the game continues the search is deepened to depth 5 at its maximum. After calculating the next moves from the open list, when the leaf nodes are reached the heuristic is calculated and then minimax is called recursively on the tree, when the root is reach it has been returned the next maximum move to take.

The Game Heuristics

The Polarized Ladder game assigns a heuristic total to each potential next state of the game at the leaves of the state space tree under analysis. The heuristic total = H1 + H2.

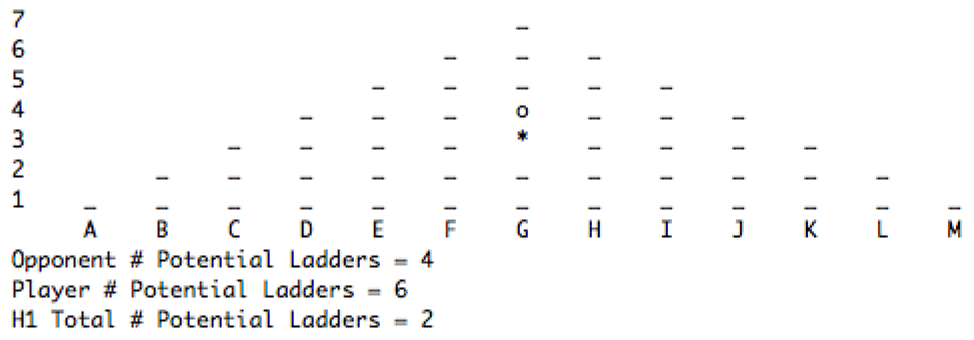
2.1. H1 - Potential number of ladders

The game calculates the potential number of ladders that can be constructed for each token on the board. This is done by comparing a token position against an array of potential ladder patterns (e.g. (0,0), (1,0), (1,1), (2,1), (2,2)). The game assigns 1 point for each ladder pattern that is not blocked (see Figure 1), taking into account polarity blocks of unpolarized ladders. This is carried out for the current player, and then separately for the opponent player. The heuristic total H1 = Current Player # Ladders - Opponent Player # Ladders (see Figure 2).



```
7                           _
6                       _   _   _
5                   _   _   _   _   _
4               _   _   _   _   _   _   _
3           _   _   _   _   *   _   _   _   _
2       _   _   _   _   _   _   _   _   _   _   _
1   _   _   _   _   _   _   _   _   _   _   _   _   _
    A   B   C   D   E   F   G   H   I   J   K   L   M
Player # Potential Ladders = 10
H1 Total # Potential Ladders = 10
```

Figure 1: Player Potential # Ladders at point 3G



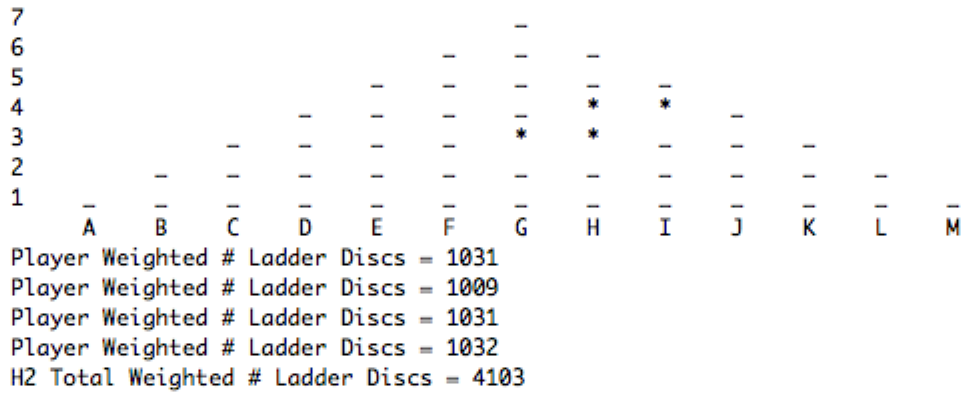
```
7                                           _
6                                     _     _     _
5                               _     _     _     _     _
4                         _     _     _     o     _     _     _
3                   _     _     _     _     *     _     _     _     _
2             _     _     _     _     _     _     _     _     _     _     _
1       _     _     _     _     _     _     _     _     _     _     _     _     _
        A     B     C     D     E     F     G     H     I     J     K     L     M
Opponent # Potential Ladders = 4
Player # Potential Ladders = 6
H1 Total # Potential Ladders = 2
```

Figure 2: H1 Heuristic Total Showing Player and Opponent Tokens

The Potential number of ladders heuristic allows the AI to evaluate how many ladders are available to be constructed at each position occupied with a token. This informs the AI about how favorable a position is in relation to ladder construction.

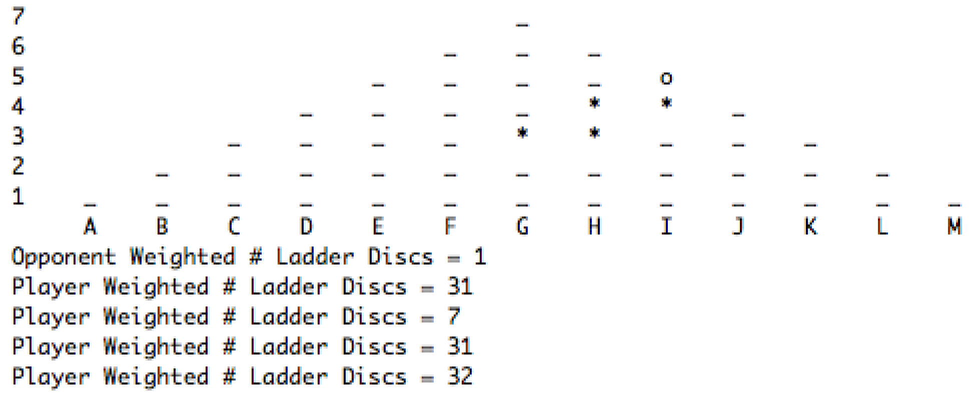
2.2. H2 - Weighted number of ladder Discs

The game calculates the number of discs on each potential ladder that can be constructed for each token on the board. This is done by comparing a token position against an array of potential ladder patterns (e.g. (0,0), (1,0), (1,1), (2,1), (2,2)) and counting the number of discs on each ladder. The game assigns a weight to each disc that is detected on a potential ladder. The weights of the discs are summed up to obtain the heuristic score for the current player (see Figure 3). In the case that a ladder is blocked with an opponents disc, or neutralized with a polarity block, the heuristic score for the ladder is 0 (see Figure 4). This is repeated for the opponent player. The heuristic total H2 = Current Player Weighted # Ladder Discs - Opponent Player # Ladder Discs.



```
7                                   _
6                              _    _    _
5                         _    _    _    _    _
4                    _    _    _    _    *    *    _
3               _    _    _    _    *    *    _    _    _
2          _    _    _    _    _    _    _    _    _    _    _
1     _    _    _    _    _    _    _    _    _    _    _    _    _
      A    B    C    D    E    F    G    H    I    J    K    L    M
Player Weighted # Ladder Discs = 1031
Player Weighted # Ladder Discs = 1009
Player Weighted # Ladder Discs = 1031
Player Weighted # Ladder Discs = 1032
H2 Total Weighted # Ladder Discs = 4103
```

Figure 3: Player Weighted Number Of Ladder Discs



```
7                                   _
6                              _    _    _
5                         _    _    _    _    o
4                    _    _    _    _    *    *    _
3               _    _    _    _    *    *    _    _    _
2          _    _    _    _    _    _    _    _    _    _    _
1     _    _    _    _    _    _    _    _    _    _    _    _    _
      A    B    C    D    E    F    G    H    I    J    K    L    M
Opponent Weighted # Ladder Discs = 1
Player Weighted # Ladder Discs = 31
Player Weighted # Ladder Discs = 7
Player Weighted # Ladder Discs = 31
Player Weighted # Ladder Discs = 32
```

Figure 4: Opponent Blocks a Potential Ladder at 5I

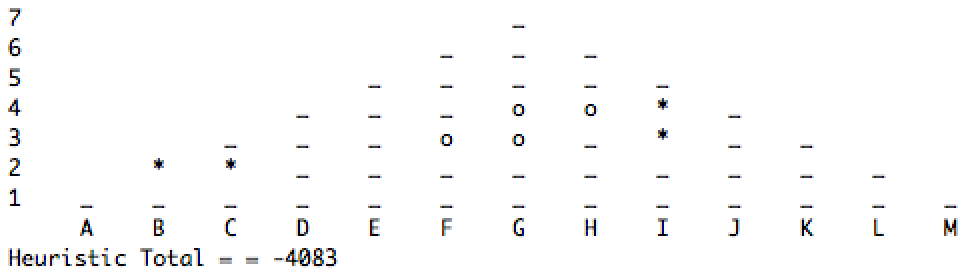
The following weights were used in the Polarized Ladder game implementation:

Disc 1 =	1	
Disc 2	=	2
Disc 3	=	20
Disc 4	=	1000
Disc 5	=	10000

This heuristic allows the game to evaluate a state taking into account how close a player is to constructing a complete ladder. The ladder disc weight values are used to allow the game to give more importance to ladders with more discs. This informs the construction of ladders as well as ladder blocking.

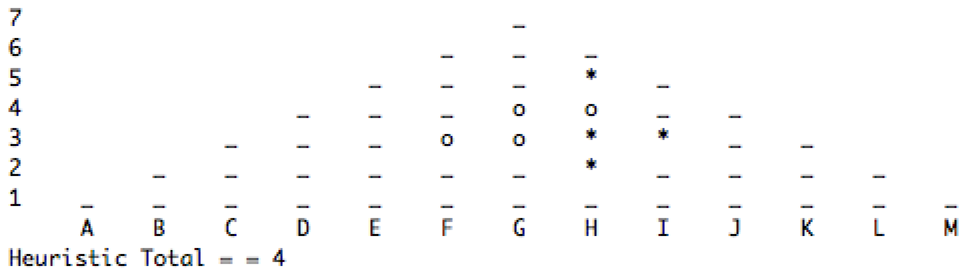
2.3. H = H1 + H2

The total heuristic score for a generated game state is calculated by adding the total of heuristics H1 and H2. This allows the AI to determine how favorable a position is to the current player, taking into account how it impacts the state of the board (see Figures 5 and 6).



```
7                           _
6                       _   _   _
5                   _   _   _   _   _
4               _   _   _   o   o   *   _
3           _   _   _   o   o   _   *   _   _
2       *   *   _   _   _   _   _   _   _   _   _
1   _   _   _   _   _   _   _   _   _   _   _   _   _
    A   B   C   D   E   F   G   H   I   J   K   L   M
Heuristic Total = = -4083
```

Figure 5: * Player Position 2C Heuristic



```
7                           _
6                       _   _   _
5                   _   _   _   *   _
4               _   _   _   o   o   _   _
3           _   _   _   o   o   *   *   _   _
2       _   _   _   _   _   _   *   _   _   _   _
1   _   _   _   _   _   _   _   _   _   _   _   _   _
    A   B   C   D   E   F   G   H   I   J   K   L   M
Heuristic Total = = 4
```

Figure 6: * Payer Position 4H Heuristic

Using this heuristic, the AI is able to determine which move it should consider in order to get closer to a winning state, or to block an opponent’s ladder which is close to a winning state (see Figure 6).

Tournament Results

The implementation of the Polarized Ladder game that our team used in the Tournament experienced one win and one loss.

Us vs Team 1

In the first match, our AI was the 2nd player. The result of this match was a tie. For the second match, our AI was the 1st player. The result of this match was a win. The reasons for this win are the following: 1. Our implementation gradually increased the depth of the minimax search from depth 2 to depth 4 as the game progressed. The opponents implementation did not seem to go farther than depth 2 for the whole game. 2. Our implementation heuristics were more informed and used more appropriate number of disc weights in its heuristic. This was seen as the win came very easily during the 1st half of the game where our AI built a ladder without opposition.

Us vs Team 2

In the first match, our AI was the 2nd player. The result of this match was a tie. For the second match, our AI was the 1st player. The result of this match was a loss. There are two reasons that contributed to this loss: 1. we modified our implementation to delay going deeper than depth 2 (i.e. examine depth 0, 1, and 2) to avoid exceeding the 3 second limit. 2. The opponents AI used a modification of minimax which selected the 4 best moves at depth 2, and then examined potential moves resulting from those moves up to a depth of 4, this was done from the start. The team that won against us ended up winning the tournament. I believe that if we had not delayed going into level 3, our implementation would have at least tied theirs, as we were using the same heuristic ideas.

There were numerous websites looked at to understand the minimax algorithm and the best measurements for game heuristics but two websites were looked at extensively:

1. http://www.ntu.edu.sg/home/ehchua/programming/java/JavaGame_TicTacToe_AI.html
This website offered an informative view of the minimax algorithm.
2.
http://www.authorstream.com/Presentation/mysyasir-149155-artificial-intelligence-connect-4-game-ai-four-heuristic-model-presentation1-science-technology-ppt-powerpoint/
Which gave insight into methods to measure heuristics.