PROJECT REPORT

CSE 537: Project 2

GAME SEARCH-Connect-4

Submitted by:

Aakriti Mittal- 110562375

Shubhi Rani-110455118

Introduction

This project focuses on the game Connect In this game, the board is a 7x6 grid of possible positions

```
  0 1 2 3 4 5 6
0 * * * * * * *
1 * * * * * * *
2 * * * * * * *
3 * * * * * * *
4 * * * * * * *
5 * * * * * * *
```

Two players take turns alternately adding tokens to the board. Tokens can be added to any column that is not full .When a token is added, it immediately falls to the lowest unoccupied cell in the column. The game is won by the first player to have four tokens lined up in a row, either vertically or horizontally, or along a diagonal.

In this homework, we have successfully implemented Connect-4 game using following adversarial search techniques:

1. Minimax Search Algorithm
2. Alpha – Beta Search

A new Evaluation function with better heuristic for finding out the utility value at terminal nodes has been implemented.

Further, the game is extended for connect-k problem (k=3, 4, 5) and longest-Streak-To-Win Problem.

Implementation section highlights the changes done in the code.

The results of new_player (using the better evaluation function) vs. Minimax and Alpha-Beta search have been obtained and shown in the result section of the report which also includes number of nodes expanded and execution time .

Implementation

1. Minimax Search :

- This algorithm is implemented in basicplayer.py file as :

```
def minimax(board, depth, eval_fn,
        get_next_moves_fn=get_all_next_moves,
        is_terminal_fn=is_terminal,
        verbose=True):
```

- It calls max_value() and min_value() functions which return the maximum and mimimum values at a node respectively.
- get_all_next_moves() is called to generate successors at a given node. If a terminal node is reached, basic_evaluate() is called which returns the utitlity value of the terminal node.
- Minimax() returns the column number which defines next move of the player
- ***How To Run***: In order to run Minimax search uncomment the line:48 in lab3.py.

```
run_game(human_player,basic_player)
```

2. Better Evaluation Function:

- A better heuristic has been used for utility function to calculate value at the terminal nodes.
- The heuristic emphasizes on number of consecutive moves for a player to win the game. Highest weight is given to longest consecutive move in any direction (horizontal, vertical or diagonal). For a particular board configuration, values have been assigned as follows:
  - Number of '4' streak : weightage of 100000
  - Number of '3' streak : weightage of 100
  - Number of '2' streak : weightage of 1
- The final score is calculated as :

  Score = (Number of 4 streaks)*100000 +(Number of 3streaks)*100 +(Number of 2 streaks)
- If opponent player has any 4 consecutive moves, then a large negative value from terminal function is returned.
- At max node, we take the positive value of score and vice-versa at min node.
- New_player uses better evaluation function defined as:

```
def new_evaluate(board):
```

- The functionality of new player and better evaluation function, line:191 has been added in basicplayer.py

- ***How To Run***: In order to use new_player and basic evaluation function, line:49 in lab3.py has been added

```
run_game(new_player,basic_player)
```

3. Alpha Beta Search:

- This algorithm is implemented in lab3.py file as :

```
def alpha_beta_search(board, depth, eval_fn,
        get_next_moves_fn=get_all_next_moves,
        is_terminal_fn=is_terminal,
        verbose=True):
```

- It calls alpha_beta_max_value() and alpha_beta_min_value() functions which return the maximum and mimimum values at a node respectively.
- get_all_next_moves() is called to generate successors at a given node. If a terminal node is reached, basic_evaluate() is called which returns the utitlity value of the terminal node.
- Minimax() returns the column number which defines next move of the player
- ***How To Run***: In order to run alpha beta search uncomment the line: 147 in lab3.py.

```
run_game(alphabeta_player, new_player)
```

4. Generalization of the Game

a. Connect –k Problem:

- The connect-4 game has been generalized for k=3, 4, 5.
- The winning number k can be considered a variable of the game.
- Following changes have been made in code to generalise the game:
  - ‘k’ has been added as a member of class ConnectFourBoard with default value k=4
  - ‘k’ has been passed to the constructor of class ConnectFourBoard

```
def __init__(self, board_array = None,
board_already_won = None, modified_column = None,
current_player = 1, previous_move = -1, k=4)
```

  - Function get_k() has been defined as:

```
def get_k(self):
```

  - Wherever ConnectFourBoard object has been instantiated, an additional parameter has been passed as :

```
k=self.get_k()
```

- ***How To Run***: In order to run, the generalised version of the game, set k =3, 4, 5 at line:71 and line:87

b. Longest – Streak- to-win Problem

- The game does not stop at a fixed number k.
- Instead, each player has 10 tokens in total, and in turn they put the tokens all in.
- After 20 rounds, the one who has the longest horizontal, vertical or diagonal streak wins.
- The win condition has been modified to incorporate Longest –Streak-to-win problem.
- Following changes have been made in code:
    - New function has been defined in class ConnectFourBoard as:
      ```
      def _is_win_longest_streak(self, row, col,p):
      ```
    - Is_win() has been modified to call _is_win_longest_streak().

- ***How To Run***: In order to run this, set k =20 at line:71 and line:87

Results

1.) Minimax search :

```
run_game(new_player,basic_player)
```

```
Player 1 (X) puts a token in column 5
Win for X!

  0 1 2 3 4 5 6
0 X O O O X   X
1 X X O O O   O
2 O X O O O   X
3 X O X X X X O
4 X X X O X O X
5 X X O O O X O

Number of expanded nodes are:63868
Execution Time: 72

Process finished with exit code 0
```

2.) Alpha Beta Search:

```
run_game(alphabeta_player, new_player)
```

```
Player 2 (O) puts a token in column 0
Win for O!

  0 1 2 3 4 5 6
0       X
1       X
2   X   X
3 O O O O
4 X O O X
5 O O X X X O

Number of expanded nodes are:27468
Execution Time: 23

Process finished with exit code 0
```

References

- https://en.wikipedia.org/wiki/Negamax
- http://www.ics.uci.edu/~jlam2/connectk.pdf
- http://roadtolarissa.com/connect-4-ai-how-it-works/