CSE 537 Assignment 2 Report: Game Search

Remy Oukaour
SBU ID: 107122849
remy.oukaour@gmail.com

Jian Yang
SBU ID: 110168771
jian.yang.1@stonybrook.edu

Thursday, October 8, 2015

1 Introduction

This report describes our submission for assignment 2 in the CSE 537 course on artificial intelligence. Assignment 2 requires us to implement an AI player for the game of Connect Four that uses minimax search with our own position evaluation function, as well as minimax with alpha-beta pruning. We are also supposed to generalize the game to Connect K, and implement an alternate "longest-streak-to-win" mode with a corresponding evaluation function. In this report, we discuss the implementation details and performance of our solutions.

2 Implementation

We were given seven Python code files implementing the assignment framework. Of these, *basicplayer.py*, *connectfour.py*, and *lab3.py* are essential to the problem; *tree_searcher.py* is only needed for testing our search algorithms; and *tests.py*, *tester.py*, and *util.py* are left over from the original MIT assignment and can be deleted.

For the sake of structured code, we moved the definitions of all player callbacks and their search functions into *basicplayer.py*, leaving the core Connect Four code in *connectfour.py*, and with only the *run_game* function and main testing code in *lab3.py*. (Originally the player callbacks were scattered across all three files, and some of them were only needed by the MIT assignment. We deleted these to leave only the functions we needed.)

TODO: finish this report.

2.1 Minimax algorithm

2.1.1 Data structures

2.1.2 Utility methods

2.2 New evaluation function

2.3 Alpha-beta pruning

2.4 Generalizing the game

2.4.1 Connect K

We added a keyword argument *chain_length_goal* to to the constructor of *ConnectFourBoard*, with a default value of 4. We then modified the method *is_win_from_cell* to use this value instead of a hard-coded 4:

```
def _is_win_from_cell(self, row, col):
    """
```

```
    Return whether there is a winning set of four connected nodes
    containing the specified cell.
    """
    return (self._max_length_from_cell(row, col) >=
        self._chain_length_goal)
```

We also had to modify the methods *do_move* and *clone*, both of which return a new *ConnectFourBoard* derived from the old one, to copy the value of *self._chain_length_goal* to the child board.

2.4.2 Longest-streak-to-win

We added a keyword argument *longest_streak_to_win* to to the constructor of *ConnectFourBoard*, with a default value of False. We then modified the methods *is_win* and *is_tie* to use alternate definitions of winning and tying when *self._longest_streak_to_win* is True:

```
def is_win(self):
    """
    Return the ID of the player who has won this game.
    Return 0 if it has not yet been won.
    """
    if self._longest_streak_to_win:
        if self.num_tokens_on_board() < 20:
            return False
        current_streak = self.longest_chain(
            self.get_current_player_id())
        opponent_streak = self.longest_chain(
            self.get_opposite_player_id())
        if current_streak > opponent_streak:
            return self.get_current_player_id()
        if opponent_streak > current_streak:
            return self.get_opposite_player_id()
        return 0
    for i in xrange(self.board_height):
        for j in xrange(self.board_width):
            cell_player = self.get_cell(i, j)
            if cell_player and self._is_win_from_cell(i, j):
                return cell_player
    return 0
```

```
def is_tie(self):
    """
    Return whether the game has reached a stalemate, assuming
    that self.is_win() returns False.
    """
    if self._longest_streak_to_win:
        return self.num_tokens_on_board() == 20
    return 0 not in self._board_array[0]
```

We also had to modify the methods *do_move* and *clone*, both of which return a new *ConnectFourBoard* derived from the old one, to copy the value of *self._longest_streak_to_win* to the child board.

3 Results

3.1 Test cases

3.2 Benchmarks

3.3 Connect K

Playing a game with different values of *chain_length_goal* behaves as expected. At 3, it is easier to win; at 5, it is harder. Winning at 2 is trivial, and at 6 or 7 is barely possible (with a cooperative opponent). At 1, the player to move first wins instantly, and at 8 or beyond, every game is a tie.

3.4 Longest-streak-to-win

Playing a game in longest-streak-to-win mode also behaves as expected. Each player gets to place 10 tokens, and then the player with the longest chain of tokens wins, or there is a tie if they have equally long chains.

4 Conclusion