Introduction to AI - Connect5

Nassim Maluli, Bachelor Wirtschaftsinformatik, Matr-Nr 3216710
Philipps University of Marburg
Marburg, Germany
Maluli@students.uni-marburg.de

I. Background

For the purpose of this challenge we used a combination of the MiniMax algorithm with Alpha-Beta pruning and a Depth control variable. we will elaborate on the evaluation-Function, n-Streak detection and Terminal tests we implemented for the project in this README file.

II. The implementation of the MiniMax algorithm

The connect5 game is obviously a ”contingency Problem” i.e. there is no way to control opponent’s actions, which have a tremendous impact on the ”perfect way” to play the Game(at least from the Max-Player point of view). This means the ”Other” player or ”Min” always has something to say about the course of the game through his actions of course. The MiniMax algorithm is designed to search for a solution (or at least a favourable state that could lead to a solution) despite of Min’s best effort or ”perfect play” to win or to hinder us from achieving a winning state.

1) The main function ”alpha-beta-search()” with parameters (State : the starting state, Depth : to what depth we want to search, Alpha , Beta) :
    a) The approach we took was: to let the program of PlayerA run against the ”random” PlayerB given an evaluation function f0 and observe if the program conducts any blocks, changes in used slots or columns or actually wins against a the ”random” PlayerB. The evaluation function plays obviously a crucial role in making some states favourable for the search algorithm, therefore a practical approach was taken (like explained above) to fine tune it.
    b) the depth 5 has shown a good and efficient performance for the program.
    c) Alpha.
    d) Beta.
2) maxValue function: returns the maximum values of the successor nodes.
3) minValue function: returns the minimum values of the successor nodes.

III. The n-Streak detection functions

To identify possible n-Streaks of coins for player ’A’ or ’B’ we use multiple functions that look in different directions like Down, Right, Up-Right, and Down-Right to test for these occurrences and return a player that achieved these streaks. Blocked streaks (streaks that cannot be extended to winning streaks because of an opponent’s Coin) of 2 , 3 or 4 are unfavorable to us because they give a false evaluation of a desirable state but are in fact not helpful to evaluate a state objectively. These functions are based primarily on the functions provided to us in the Win5 project like (hasWon, IsDraw, TestRight, TestDown ...etc) but slightly tweaked to look for blockages and to be general in propose, hence the Parameter ”n”.

1) 5-Streaks: are based on the function ”hasWon” and returns who of the two players ’A’ or ’B’ has won by achieving a 5-coin streak.
2) 4-Streaks: ”check-Fours(state)” searches the board starting from (0,0) for each position in 4 directions Down, Right, Down-Right, Up-Right and identifies the player that achieved a Streak of 4.
3) 3-Streaks: ”check-threes(state)” is similar to 4-Streaks but searches for Streaks of 3.
4) 2 in a row: ”check-twos(state)” gives the program incentives in shallow depths of d smaller or equal to 5 to pursue states containing them and build upon these streaks in later turns to finally reach winning Streaks.

IV. The Evaluation function

The evaluation function is a crucial part of the search mechanism. In our case we used the following weighted Function:

$$f(x) = w_1 f_1(x) + w_2 f2(x) + w_3 f_3(x) + w_4 f_4(x) - w_1 f_1(x) - w_2 f_2(x) - w_3 f_3(x)$$

$$f_1 : \textit{identifies } 5 - \textit{streaks (winning states) returns } +1000 \textit{ for AI and } -1000 \textit{ for opponent.}$$
$$f_2 : \textit{identifies } 4 - \textit{ streaks and returns } +50 \textit{ for AI and } -50 \textit{ for opponent.}$$
$$f_3 : \textit{identifies } 3 - \textit{ streaks and returns } +25 \textit{ for AI and } -25 \textit{ for opponent.}$$
$$f_4 : \textit{identifies } 2 - \textit{in a row (only for AI) and returns } +10 \textit{ for AI}$$

$$w_1 := 1$$
$$w_2 := \textit{ the number of } 4 - \textit{streaks the AI / Opponent has}$$
$$w_3 := \textit{ the number of } 3 - \textit{streaks the AI / Opponent has}$$
$$w_4 := \textit{ the number of } 2s - \textit{in a row that the AI has}$$

Fig. 1. The Evaluation Function

1) The largest value goes to the Winning states, obviously this should have the highest priority if encountered while searching in the state space. Equally, when the opponent reaches a winning state i.e. we reach a Losing state. We evaluate this case by giving it the worst score.
2) Immediately after the Winning/Losing states comes the states which have a streak of 4s. These Streaks are surely

not blocked i.e. extendable to winning states (Streaks of 5).

3) A little further down the priority hierarchy comes the Streaks of 3s. These are although not crucial for the Game, also impact the formation of the winning Streaks. So they naturally have a lighter value and weight.
4) The Streaks of 2s have the lowest evaluation but are there to give the AI incentives to ”start somewhere” forming a winning Streak. We found them very helpful in shallow depths.

References