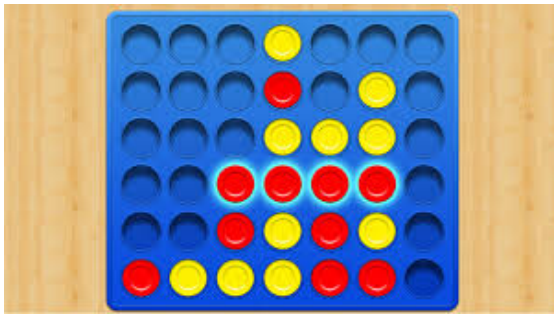
IT University of Copenhagen

IT-University of Copenhagen

Intelligent systems programming

Connect 4

Tróndur Høgnason (thgn@itu.dk)
Kristian Mohr (kvin@itu.dk)



March 1, 2017

1 Connect 4

1.1 Minimax

MidOrFeed uses a minimax algorithm to calculate the optimal move. A minimax algorithm calculates the best move based on the assumption that the adversary has takes his best move previous to our previous move. In plainer terms, the algorithm seeks to maximize his score based the knowledge that the adversary will try to minimize our gain (maximize his gain).

1.2 Cutoff function

When we are using a minimax algorithm for Connect Four, it search through all the possible game states reachable. In Connect Four the searchspace is so large that it is unfeasible to examine all possible move and their consequences until only a few moves are left. Therefore it is necessary to restrict the search depth with a cut-off function. *MidOrFeed's* cut-off function is implemented using only tree depth. There is a hardcoded maximal search depth, and the minimax algorithm will return with best possible move so far, if it has reached the maximal tree depth.

1.3 α-β pruning

Even with a cut-off function the searchspace is still quite large. To further reduce the searchspace *MidOrFeed's* minimax algorithm implements α-β pruning. α-β pruning. α-β pruning allows *MidOrFeed* to end the exploration of a branch early. Let us consider a n example, where the minimax algorithm is in the maximum phase in a state, where branch A will return a score of 4 and branch B leads to another branch, which has two option which give 3 and 5. In this example our α is 4. If we choose to follow branch B, the minimax algorithm will be in the minimum phase. So when the algorithm examine option 3, minimum will always pick that option. So we do not need to examine B's other options, they are pruned, since branch A will always be better (if both players play optimally).

To optimize the α-β pruning *MidOrFeed* explores the middle column first and then out towards the edges. In most cases this speeds up the pruning, since the best moves, especially at the start of the game, are usually found close to or in the middle column. And when the best moves are found earlier, there is a bigger chance that some branches can be pruned. This makes the searchspace smaller and in turn lets us use a higher maximal search depth.

1.4 Heuristics

1.5 Pre coded moves

At the beginning of the game when *MidOrFeed* can not reach the full depth of the search tree, it occasionally makes mistakes. To prevent the bad moves

at the beginning, we compared the starting moves to the actual perfect moves using a connect 4 solver.[1]

We identified the worst moves, and hardcoded *MidOrFeed's* actions in certain early game scenarios. This method is used at most until turn 7.

1.6 Further improvements

While testing *MidOrFeed*, we have seen that the heuristics we use are not optimal. An improvement would be to ....

The cut-off function is also implemented rather naively. A better approach would be to consider if a gamestate was suitable for cut-off rather than just having a fixed tree depth.

[1] `http://connect4.gamesolver.org`