KATE: Kick-Ass-and-Terminate-Effectively

A connect-four AI

Introduction

The purpose of this project was to implement a simple connect-four AI using a combination of textbook algorithms and custom algorithms and heuristics. We started by implementing the naive solution using the minimax algorithm. The solution was further improved by the alpha-beta pruning algorithm described in the textbook. From there we added the Iterative-depth-search combined with a time-cutoff function. Finally, a heuristic was written to evaluate the non-terminated state space. Due to the size limitation of the report, we have chosen not to elaborate on all the implemented textbook algorithms in depth.

Alpha-Beta search (ABS)

This is a textbook algorithm which prunes the branching of the decision tree in a given player-turn. Since the ABS is implemented in conjunction with minimax, each recursion calculates the player and adversary's optimal turn respectively. Given that the adversary employs an optimal playstyle, the ABS prunes the branches when:

- In a max-ply, we can cut the rest of the actions when we find a utility greater than Beta (the min-value of a previous min-ply).
- In a min-ply, we can cut the rest of the actions when we find a utility less than Alpha (the max-value of a previous max-ply).

As a consequence, the ABS might yield a significant reduced branching factor of the intermediate recursions.

Cut-off algorithms

Due to the branching factor and the depth of evaluating a state in a connect-four game, it is necessary to employ certain cutoff mechanisms such that a player-turn doesn't exceed 10 seconds. The following section elaborates on the most important mechanisms that reduce runtime of calculating a player-turn.

Iterative-Deepening-Search (IDS)

The project assignment specifies as a requirement that a turn should take no more than ~10 seconds (on average) to calculate. This poses somewhat of problem unless addressed properly. As the minimax employs a DFS approach, we risk that an asymmetric state space evaluation would occur if we terminate once the time has elapsed. By employing IDS, we guarantee that each action is fully explored for a given depth, before proceeding to next depth. This does result in some overhead, however this is negligible.

Time cutoff

To accommodate the time constraints, we implemented a safeguard, that effectively terminates further branching once a time-threshold has elapsed (~10 sec). This strategy combined with IDS and our state-analysis heuristic ensures that a symmetric evaluation of the state space has been computed at a given depth. This in turn guarantees that a valid action always will be found, and that the action will be optimal according to our heuristic.

Evaluation functions

The utility value of a state is determined by two outcomes. Either the state leads to a terminal state, or reaching a non-terminal during cutoff.

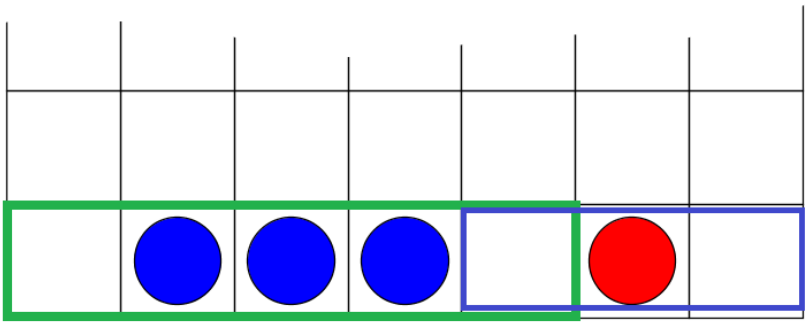
Terminal state evaluation

A terminal state is reached if any of the players achieves four connected cells in any given direction, or no legal moves are left.

Provided a terminal state has been reached, a win will potentially yield a utility of a maximum integer, while a loss will yield the minimum. It is then divided by the depth of when the win/loss occurs. The division ensure that a win in the low depths has higher weight than a win in a high depth. Conversely, the opposite applies for a loss.

Non-terminated state evaluation

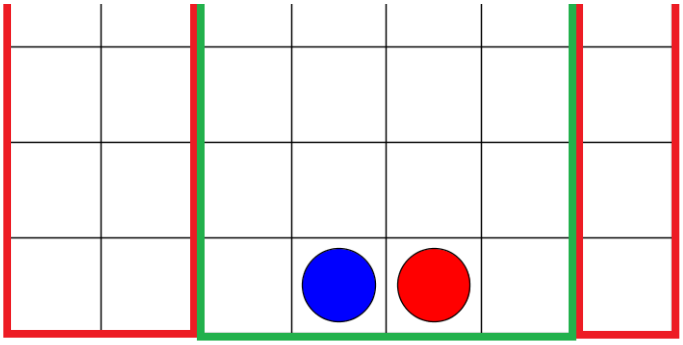
If a given ply is interrupted either due to a depth or time limitation, we are forced to evaluated the utility of the current state. The central idea is that we reward points for potential wins. A potential win is a chain of connected cells that are either free or only occupied by a single player. A chain is only valid if it is longer than or equal to the winning condition (of 4). If so, the player owning the chain, is rewarded points by each cell occupied by a coin. The distribution of points is calculated by counting the number of coins in the chain squared.



In the figure above, the blue player has placed 3 connected coins. The chain is illustrated by the green box, and shows the potential play of the blue player. Likewise, the blue box displays the chain obtained by the red player. In this case, only the blue player has a valid chain, as the chain belonging to the red player is lower than the win-condition. The utility of the blue players chain is in this case 9 points (3^2). It should be noted, that if the opponent is rewarded any points from valid chains, they are deducted from the final utility value.

Additional Optimizations

A final optimization has been implemented, in order to facilitate reaching a higher depth before time-cutoff. As such, we decided to eliminate as many actions as we could before the ABS process, thereby reducing the branching factor of the first iteration.



The idea is that any empty columns with **only empty neighboring columns,** is unnecessary to search through. As seen in the figure above, the columns highlighted with red would be ignored during depth 1, while the columns marked with green would be searched through. Subsequently, at deeper depths all columns will be searched through.

Conclusion

Looking back at the progression of the development of the AI, we see now the benefits it has to prune and optimize the search algorithms, especially considering the time constraints. In the beginning KATE was unable to find a solution on bigger boards than 4x4. Implementing ABS greatly improved this, but it wasn't until the heuristic was applied that she was able to beat us on a 7x6 board. From this point on all optimizations only led to our demise sooner rather than later. We look forward to the competition, hoping KATE will crush everybody else.