Checker solver overview

David Ma

March 2nd, 2023

Program Design

For each terminal state, we calculate the utility by considering 2 cases.

Case 1 is that one of the players have no pieces left. If the current player have no pieces left, we return $-\infty$ to indicate that the player lost. If the opponent have no pieces left, we return ∞ to indicate that the player won.

Case 2 is that one of the players have no legal moves left. If the current player has no moves left, return $-\infty$, if the opponent has no moves left, return ∞.

For each non-terminal state, we estimate the utility by comparing the differences in the current player's pieces and the opponent's pieces. Each normal piece is worth 1 point, and each king is worth 2 points. We compute the differences by summing up the current player's total points and subtracting it with the opponent's total points.

For optimization, I chose state caching, because it ensures that we don't have to compute/estimate the utility for a existing state, which can prevent the algorithm to waste unnecessary time to perform repetitive computations.