AI Lab Report 5

Om Patil (200010036)
Hrishikesh Pable (200010037)

February 2022

1 Brief Description of the Algorithm

The Basic Algorithm ,that drives the 2 bots, is the Minimax Algorithm. The Minimax bot uses purely the minimax algorithm to find the next best move to be played, whereas the AlphaBeta bot implements Alpha-Beta pruning on top of the Minimax algorithm, in order to achieve greater speed in finding the next best move.

Following is the pseudocode for Minimax Algorithm:

```
function MINIMAX($position$, $depth$, $maximizingPlayer$)
    if $depth == 0$ or game over in $position$ then
        return static evaluation of $position$
    end if
    if $maximizingPlayer$ then
        $maxEval = -\infty$
        for each $child$ of $position$ do
            $eval$ = MINIMAX($child$, $depth - 1$, $false$)
            $maxEval = \max(maxEval, eval)$
            return $maxEval$
        end for
    else
        $maxEval = +\infty$
        for each $child$ of $position$ do
            $eval$ = MINIMAX($child$, $depth - 1$, $true$)
            $minEval = \min(minEval, eval)$
            return $minEval$
        end for
    end if
end function
```

Following is the pseudocode for Alpha-Beta Pruning Algorithm:

```
function ALPHABETA($position$, $depth$, $alpha$, $beta$, $maximizingPlayer$)
    if $depth == 0$ or game over in $position$ then
        return static evaluation of $position$
    end if
    if $maximizingPlayer$ then
        $maxEval = -\infty$
        for each $child$ of $position$ do
            $eval$ = ALPHABETA($child$, $depth - 1$, $alpha$, $beta$, $false$)
            $maxEval = \max(maxEval, eval)$
            $alpha = \max(alpha, eval)$
            if $beta \leq alpha$ then
                break
            end if
            return $maxEval$
        end for
    else
        $maxEval = +\infty$
        for each $child$ of $position$ do
            $eval$ = ALPHABETA($child$, $depth - 1$, $alpha$, $beta$, $true$)
            $minEval = \min(minEval, eval)$
            $beta = \min(beta, eval)$
            if $beta \leq alpha$ then
                break
            end if
            return $minEval$
        end for
    end if
end function
```

2 Heuristic Function

The heuristic function that we implemented to find the heuristic value of the leaf nodes in the minimax tree is the difference in the number of pieces of the player and the number of pieces of the opponent. For the red player, it would be calculated as:

$$heuristic = (number\ of\ red\ pieces) - (number\ of\ black\ pieces)$$

And for the black player, it is calculated as:

$$heuristic = (number\ of\ black\ pieces) - (number\ of\ red\ pieces)$$

Since the winning criteria of the game depends on number of pieces that each player has, at the end of the game, the difference in the number of pieces at any point of the game is a good measure of the heuristic value.

3 Tree of nodes

```
===================
| |0|1|2|3|4|5|6|7|
|a| | | | | | | | |
|b| | | | | | | | |
|c| | | | | | | | |
|d| | | |X|O| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
===================
Blacks: 02 Reds: 02
===================
| |0|1|2|3|4|5|6|7|
|a| | | | | | | | |
|b| | | | | | | | |
|c| | | | |X| | | |
|d| | | |X|X| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
===================
Blacks: 04 Reds: 01
===================
| |0|1|2|3|4|5|6|7|
|a| | | | | | | | |
|b| | | | | | | | |
|c| | | |O|X| | | |
|d| | | |O|X| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
===================
Blacks: 03 Reds: 03
```

```
==========================
| |0|1|2|3|4|5|6|7|
|a| | | | | | | | |
|b| | |X| | | | | |
|c| | | |X|X| | | |
|d| | | |O|X| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
==========================
Blacks: 05 Reds: 02
==========================
| |0|1|2|3|4|5|6|7|
|a| | | | | | | | |
|b| | |X|O| | | | |
|c| | | |O|X| | | |
|d| | | |O|X| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
==========================
Blacks: 04 Reds: 04
==========================
| |0|1|2|3|4|5|6|7|
|a| | |X| | | | | |
|b| | |X|X| | | | |
|c| | | |O|X| | | |
|d| | | |O|X| | | |
|e| | | |O|X| | | |
|f| | | | | | | | |
|g| | | | | | | | |
|h| | | | | | | | |
==========================
```

4 Comparison of Minimax and AlphaBeta bots

4.1 Based on Space and Time complexity

4.1.1 Time complexity

Since alpha-beta pruning,in general, does not generate some nodes (which are generated by the minimax algorithm), the overall time taken by the AlphaBeta bot is, in general, lesser as compared to that of the Minimax bot. But we can have examples in which the nodes explored by alpha-beta bot are exactly same as the nodes explored by the Minimax bot (since there is no scope for pruning any node in the search tree). Hence the worst case time complexity of AlphaBeta bot is the same as that of the Minimax bot.

4.1.2 Space Complexity

Here too, in general, the overall space required for AlphaBeta bot is lesser than the space required for Minimax bot because of the fewer number of nodes explored. But, again, we can have examples in which the nodes explored by alpha-beta bot are the same as the nodes explored by the Minimax bot (since there is no scope for pruning any node in the search tree).So worst case space complexity of Alphabeta bot is the same as that of the same as that of the Minimax bot.

4.2 Winning criteria

If the depth (untill which the nodes are explored) and the heuristic function are the same, then the Minimax bot and the AlphaBeta bot perform in the same way. This is because the alpha-beta algorithm only prunes some nodes (in most of the cases) which leads to better speed, but the move returned by both the algorithm is the same.