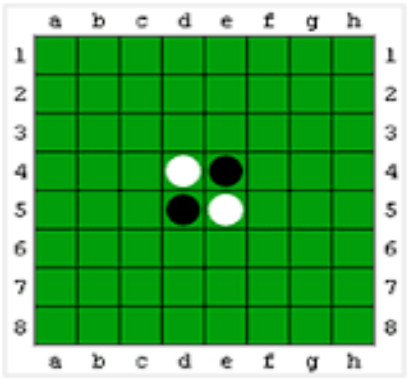
ARTIFICIAL INTELLIGENCE (CS323)

REVERSI

Submitted by: Ashish Sahu (B14CS009)
Rishabh Shukla (B14CS028)

Description



Reversi is a strategy board game for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces (disks) which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up.

The rules of the game are:-

1) A player should play in such a position that there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new disk and another disk of same color, with one or more contiguous disks of other color between them. If there doesn't exist such move then his chance is skipped and the other player gets the chance to play.
2) During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.
3) The player with the most pieces on the board at the end of the game wins.

Choice of Heuristic

We have developed a heuristic which is a function of three parameters.

Heuristic Value = F (Cell Utility Value, Disk Parity, Mobility)

1) Cell Utility Value: Every cell (square) is assigned a value depending upon the stability of the disks placed on that square. For example, Corners are given the highest value because once captured, they cannot be flanked by the opponent. The cells adjacent to corners have negative values as they can be flanked easily if the other player occupies corner. Individual cell utility values are assigned as below:

25	-5	14	10	10	14	-5	25
-5	-7	-4	1	1	-4	-7	-5
14	-4	3	2	2	3	-4	14
10	1	2	-6	-6	2	1	10
10	1	2	-6	-6	2	1	10
14	-4	3	2	2	3	-4	14
-5	-7	-4	1	1	-4	-7	-5
25	-5	14	10	10	14	-5	25

2) Disk Parity: The move which increases the number of disks of the player on the board by a greater amount is considered a better move. So, this parameter of heuristics takes into account the relative difference in number of disks occupied by max player and the min player after a move.

Disk Parity =
(Max Player Disks - Min Player Disks) / (Max Player Disks+ Min Player Disks)

3) Mobility: It is good to have greater number of possible moves so that the player could choose the best move among them. Also, it is good to play a move which decreases other player's moves. Thus a move should restrict the opponent's mobility and increase one's own mobility. So this parameter finds the relative difference between the number of possible moves for the max and the min players.

Mobility =
(Max Player Moves - Min Player Moves) / (Max Player Moves+ Min Player Moves)

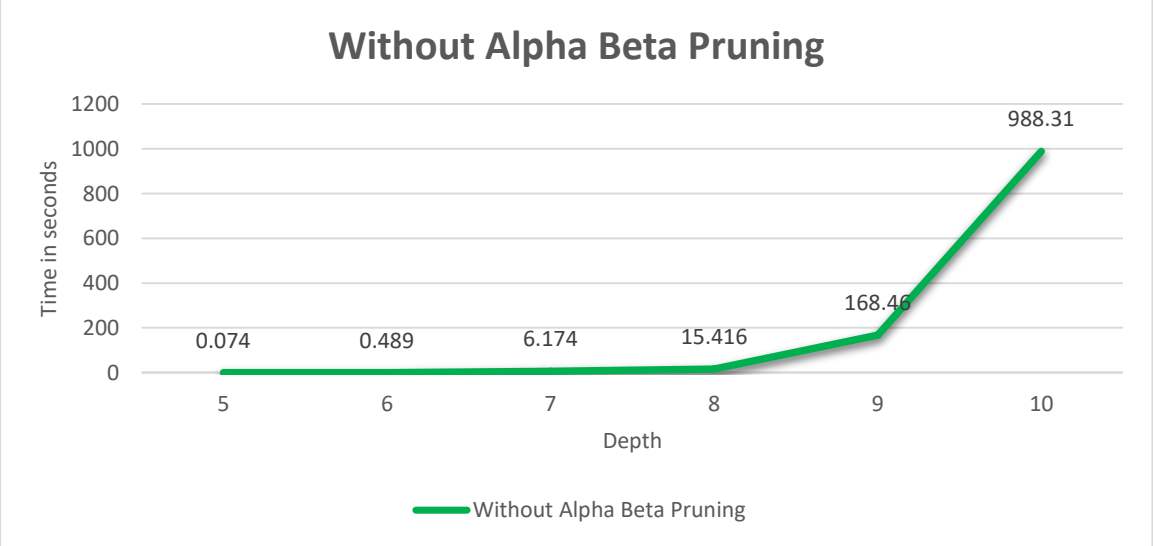
Issues with the depth of the Search Tree

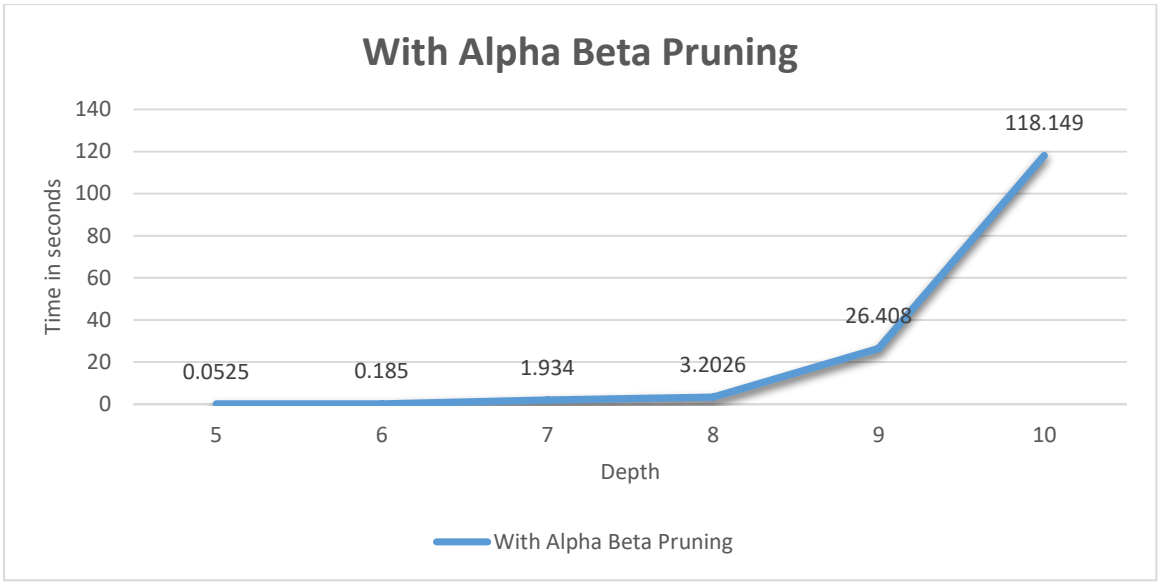
The average game length (total number of moves by both players to win the game) is 58 and the average branching factor of the game is 10. With these, the min-max algorithm took a large time to play a move, even when the depth was small.
To overcome this difficulty, we used Alpha Beta Pruning.

Alpha Beta Pruning

Alpha–beta pruning decreases the number of nodes that are evaluated by the minimax algorithm in its search tree.
When we implemented the Alpha-beta pruning in the game, the average branching factor was reduced to almost 8 from 10. This critically reduced the time taken by the bot to play a move.

The following is the graphical representation of the improvements through Alpha-beta Pruning.




With Alpha Beta Pruning
140
120
100
80
60
40
20
0
Time in seconds
0.0525
0.185
1.934
3.2026
26.408
118.149
5
6
7
8
9
10
Depth
With Alpha Beta Pruning