U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Artificial Intelligence

Adversarial Search Method for Line of Actions

Realizado por:

- Fabio Huang up201806829
- Ivo Ribeiro up201307718
- Maria Beirao up201806798

Specification

Board and Pieces:

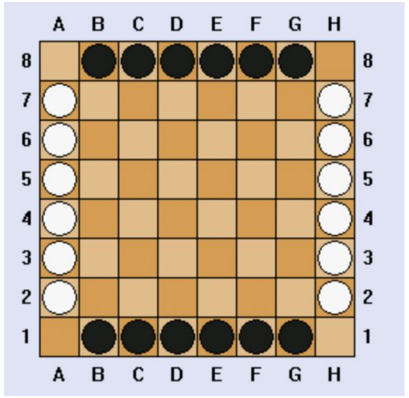
- Simple checkers board (8x8)
- Each player has 12 pieces on the board (black and white)

Game Objective:

- Connect all your pieces together in a group (including diagonals)

Movements:

- Each turn, the player moves one of his pieces, in a straight line (diagonal included), exactly as many squares as there are pieces of either color anywhere along the line of movement.
- Players may jump over their own pieces
- Players may not jump over their opponent's pieces, but can capture by landing on them



State representation

The state will be represented with an 8 x 8 matrix (board[8,8], or in general board[rows, columns]), filled with elements of the following type:

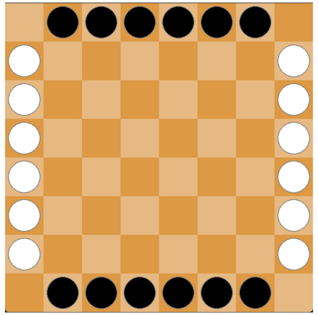
- Black Piece (B);
- White Piece (W);
- Empty Cell (0).

The player's turn to move (turn) will also be represented.

Initial State

The black checkers are placed in two rows along the top and bottom of the board, while the white stones are placed in two rows at the left and right of the board.

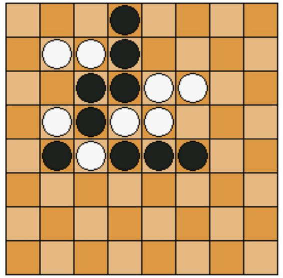
```
turn = BLACK                    // Black begins
board =
[ [0, B, B, B, B, B, B, 0],
  [W, 0, 0, 0, 0, 0, 0, W],
  [W, 0, 0, 0, 0, 0, 0, W],     // B - Black Piece
  [W, 0, 0, 0, 0, 0, 0, W],     // W - White Piece
  [W, 0, 0, 0, 0, 0, 0, W],     // 0 - Empty Cell
  [W, 0, 0, 0, 0, 0, 0, W],
  [W, 0, 0, 0, 0, 0, 0, W],
  [0, B, B, B, B, B, B, 0] ]
```


Graphical Representation of Initial State

Objective Test

The game ends when all pieces of either player are connected together in a group, where diagonals are also considered to be connected.

- If a player is reduced to a single piece, that is a win for the captured player, because a single piece counts as a connected group.
- Another way to win is if the other player has no possible moves (all pieces blocked on all sides).
- If a move made by a player creates a win for both the victory is given to the player who made the move, and we control this, by checking the win condition after a move, first for that player.


Typical finished game, where black won

```
// returns 1:Black Wins; 2:White Wins; -1:Game continues
def check_gameover():  // checks if the current state creates an ending condition
        …
```

Operators

Name	PreConditions	Effects	Cost
Move (left, Piece)	- Check the number of pieces on the same horizontal line, which will be the *Nmoves* - Check if new position inside the board - Check no opponent piece blocking it - Cannot land on own piece	- Move Nmoves to the left - Opponent piece on new position, then capture it	1
Move (right, Piece)	- Check the number of pieces on the same horizontal line, which will be the *Nmoves* - Check if new position inside the board - Check no opponent piece blocking it - Cannot land on own piece	- Move Nmoves to the right - Opponent piece on new position, then capture it	1
Move (top, Piece)	- Check the number of pieces on the same vertical line, which will be the *Nmoves* - Check if new position inside the board - Check no opponent piece blocking it - Cannot land on own piece	- Move Nmoves to the top - Opponent piece on new position, then capture it	1
Move (bottom, Piece)	- Check the number of pieces on the same bottom line, which will be the *Nmoves* - Check if new position inside the board - Check no opponent blocking it	- Move Nmoves to the bottom - Opponent piece on new position, then capture it	1
Move (botleft, Piece)	- Check the number of pieces on the same diagonal line, which will be the *Nmoves* - Check if new position inside the board - Check no opponent piece blocking it	- Move Nmoves to the bottom left diagonal - Opponent piece on new position, then capture it	1
…	…	…	

State Space:

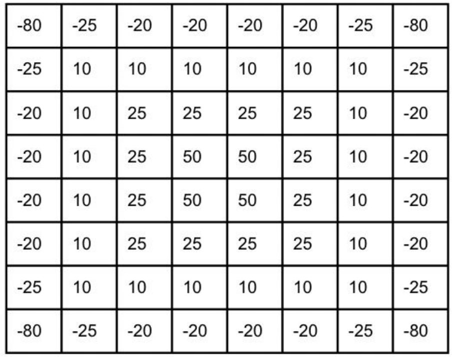
- 8x8 board game:
  - $3^{8x8} = 3.43 \times 10^{30}$
  - However, this number also contains a set of invalid moves
- Better estimate:
  - $$\sum_{B=1}^{12} \sum_{W=1}^{12} Num(B,W) - Num(1,1) \qquad \text{where} \quad Num(B,W) = \binom{64}{B}\binom{64-B}{W}$$
  - $\approx 1.3 \times 10^{24}$

Heuristics/Evaluation Function

eval = concentration + centralisation

Centralisation evaluation function:

- Pieces closer to the center will get a higher evaluation value
- Further features can be added to the evaluator (for better heuristic), such as: center-of-mass position, quads, mobility, walls, connectedness [4]



-80	-25	-20	-20	-20	-20	-25	-80
-25	10	10	10	10	10	10	-25
-20	10	25	25	25	25	10	-20
-20	10	25	50	50	25	10	-20
-20	10	25	50	50	25	10	-20
-20	10	25	25	25	25	10	-20
-25	10	10	10	10	10	10	-25
-80	-25	-20	-20	-20	-20	-25	-80

Centralisation scores

Concentration evaluation function:

Rewarding pieces that are closer together, this can be achieved by:

- Calculate the center of mass of the pieces for each side
- Compute the average distance of the pieces from the center of mass.
- The lower the average distance of the center of mass, the higher the concentration
- The inverse of this value is considered as concentration

Implemented Work:

Programming Language:

- Python (pygame library used)

Development Environment:

- Visual Studio Code,
- PyCharm
- Python Interpreter 3.10.4
- Git

Data Structure:

- Board, Game, Pieces represented as Classes
- The Board Class contains a list that has the information of the pieces positions (board[8][8])

File Structure:

- The current file structure a main class (where the game is called) and inside the folder game contains a “classes” module with all the classes (board, game, piece) including a file constants.py, which contains constants used throughout the other files (pieces colors, window size, board size…)

Bibliography:

[1] **Algorithms Explained – minimax and alpha-beta pruning**
https://www.youtube.com/watch?v=l-hh51ncgDI

[2] **Lines of Action**
https://www.boardspace.net/loa/english/index.html

[3] **THE QUAD HEURISTIC IN LINES OF ACTION**
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.3549&rep=rep1&type=pdf

[4] **An Evaluation Function for Lines of Action**
https://www.researchgate.net/publication/220717032_An_Evaluation_Function_for_Lines_of_Action/download

[5] **Analysis and Implementation of Lines of Action**
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.566.5504&rep=rep1&type=pdf

[6] **ryangmolina/lines-of-action-minimax**
https://github.com/ryangmolina/lines-of-action-minimax