Board Games with some Game Theory Cleverness

Julian Stampfli, Marc Rey, 2019.

Context

This is a console app developed during the course "Game Theory" by J. Eckerle at University of Applied Sciences Berne in Spring 2019.

The goal of this app is to implement Game Theory strategies in a play of a board game. The user plays either against another human or against a computer opponent who may either choose his next move randomly or by strategically evaluating his best move.

There are two board games that can be played:

- Pawn Chess / Bauernschach
- Alquerque / Original Checkers

Board Games

Pawn Chess ("Bauernschach")

- The game is a simplification of Chess such that there are only the pawns and no other figures.
- Play happens on a square board of 4 to 8 fields in width and height respectively.
- Initially each player has as many pawns as the board is wide, placed in the second row closest to him.

```
- - - - -
X X X X X
- - - - -
- - - - -
0 0 0 0 0
- - - - -
```

- Players move turn by turn and one pawn per turn.
- Pawns may move by:
    - advancing straight ahead 1 field, if this is not blocked.
    - advancing straight ahead 2 fields, if those are unoccupied and the pawn has not been moved yet.
    - advancing straight ahead 1 field then left or right 1 field, if the target field is occupied by the opponent. The opponent's pawn is killed.
    - stepping left or right 1 field, if this field is occupied by the opponent and the occupying pawn has moved once. (en passant) The opponent's pawn is killed.
- The game is won by the player who manages first to get a pawn to the last row on the opposite of the board.

Alquerque ("Original Checkers" / "Urdame")

- The game is a simplification of the various versions of Checkers played today.

- Play happens on a square board of 4 to 8 fields in width and height respectively.
- Initially each player has as many pawns as half the number of fields on the board, while 1 (uneven board width) or 2 (even board width) fields at the center must remain empty.

```
X X X X X
X X X X X
X X - 0 0
0 0 0 0 0
0 0 0 0 0
```

```
X X X X X X
X X X X X X
X X - - 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

- Players move turn by turn and one pawn per turn.
- Pawns may move by:
    - moving 1 field in any direction including diagonally, if this field is unoccupied.
    - moving 2 fields in any one direction including diagonally, if the first field is occupied by the opponent and the second field is unoccupied. The opponent's pawn on the first field is killed.
- The game is won by the player who's opponent cannot move:
    - either because he has no more pawns
    - or because he has no possible moves, that is he is blocked.

AI Behavior

There are various strategies available to choose the best move:

- Random
- Simple heuristic
- Complex heuristic
- Monte Carlo

All of them are implemented for Pawn Chess. Only random choice and Monte Carlo are implemented for Alquerque.

Random Choice

Simple Heuristic (only Pawn Chess)

Search is performed by Alpha-Beta-Pruning at a deepness of 4.

Complex Heuristic (only Pawn Chess)

Search is performed by Alpha-Beta-Pruning at a deepness of 4.

The heuristic evaluation function is based on various considerations:

- Total pawns:
    - The more pawns one has compared to the number of pawns the opponent has, the better. *(implemented)*
- Position of one pawn relative to the game board:
    - The further advanced a pawn, the better. *(implemented)*
    - A pawn that has an unblocked path to the finish line
        - is great. *(implemented)*
        - is less great, if there is an opponent's pawn in a neighbouring line up ahead and thus could kill our pawn. *(not implemented)*
- Position of one pawn relative to our and opponent's other pawns: *(not implemented since this is covered by the possible moves in the next level in the game tree and the count of pawns)*
    - A pawn is strong if ...
        - he can kill another pawn now
        - without being killed in the next move
        - without another being killed in the next move
    - A pawn is safe if ...
        - he cannot be killed in the next move
        - he can be killed in the next move but the killer himself can be killed in the move after. (both players loose 1 pawn each)

Monte Carlo

Search is performed by Alpha-Beta-Pruning at a deepness of 4.

App Architecture

Game Engines

GUI & User Interaction

Actors taking turns and playing

App usage

- The app is built with Python version 3.7.
- Start by running the app.py in this same directory.

Running the application

User Interaction

- As a user, interact with the app by entering one of the commands shown at each step.

Gameplay

- Note that on the board, you as a user are placed on the bottom side while the opponent is placed on the top side.

```
opponent (human or machine)
X X X X X X
X X X X X X
X X - - O O
O O O O O O
O O O O O O
user (human)
```

Critical Reflection

Challenges

- It is hard to make a clever machine player due to various reasons, foremost:
    - The game tree expands fast. Thus it is a balance between having the user to wait a long time for the machine opponent to be ready on one side and a deep game tree on the other.
    - "Designed" heuristics, in contrast to Monte Carlo, require the developers to find and optimize a proper evaluation function either by thinking and trial and error or by analysing vast amounts of data gathered. Due to the lack of data, we had to accomodate with ideas based trial and error.
- Once you get the knack of it, working with Python is nice.