Wana - First AI Project

Matilde Silva - up202007928@edu.fe.up.pt
Pedro C.A.B Gomes - up202004986@edu.fe.up.pt
Pedro Ferreira - up202004986@edu.fe.up.pt

Specification of the work to be performed

- Wana is a 2-player Board Game. Each player has 8 pieces of their colour. These pieces are aligned in the intersection lines of the board. 3 of the 6 lines are displayed in parallel and vertically and the other ones are displayed horizontally. The board lines intersect each other in 9 central points. Besides that, there 3 parallel circumferences intersecting the lines.

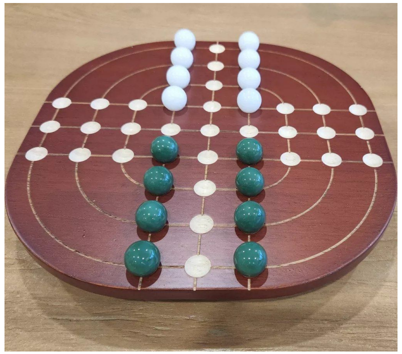
- In each play, a piece must be moved by the player, which can move along any line; however, it can not go through or occupy the space of another one.

- In case it reaches the end of the board, the piece will appear on the other side of the board.

- When a player has a piece which cannot move at the beginning of the play, he/she loses the game.

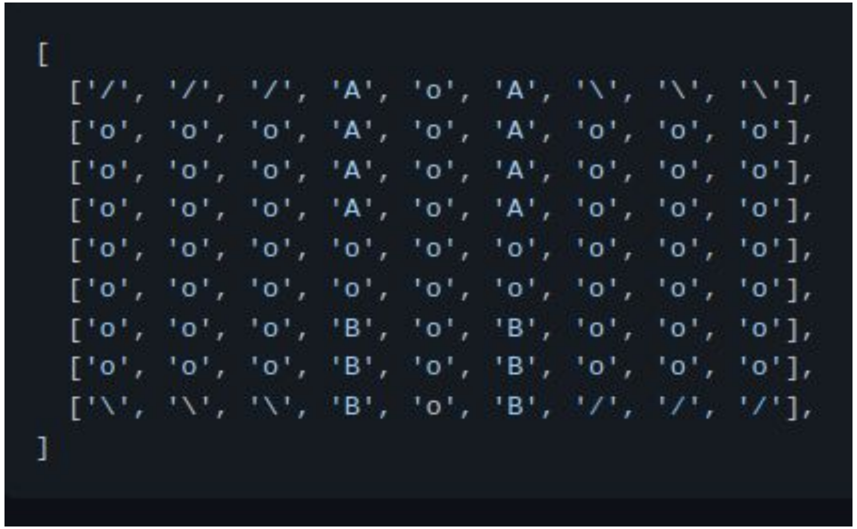
References to works found in a bibliographic search

- Wana board game rules - https://boardgamegeek.com/boardgame/364012/wana
- MiniMax algorithm - https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
- Alpha-beta pruning for MiniMax algorithm - https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
- Our source code - https://github.com/Pedro-CAB/IA-Project



Formulation of the problem as a search problem

- Utility problem- pieces that can not move have less utility;
- State representation: list of lists.
- Initial state:



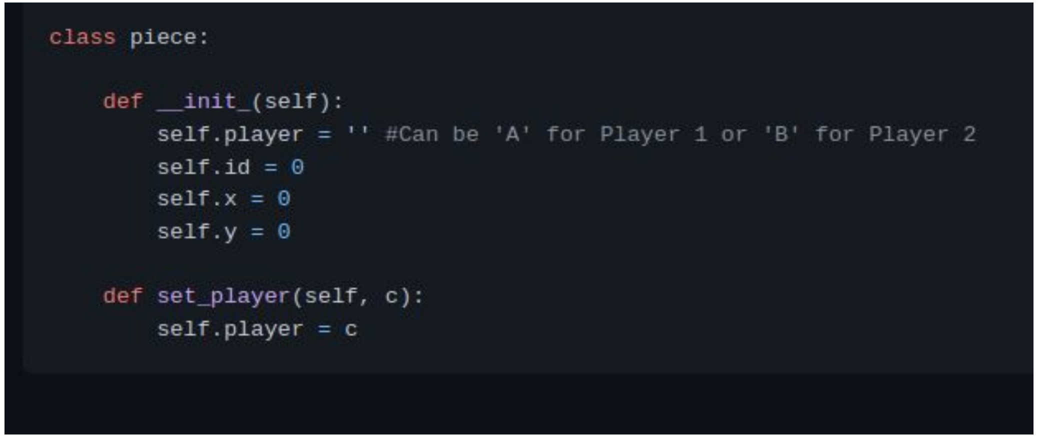
```
[
  ['/', '/', '/', 'A', 'o', 'A', '\', '\', '\'],
  ['o', 'o', 'o', 'A', 'o', 'A', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'A', 'o', 'A', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'A', 'o', 'A', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'B', 'o', 'B', 'o', 'o', 'o'],
  ['o', 'o', 'o', 'B', 'o', 'B', 'o', 'o', 'o'],
  ['\', '\', '\', 'B', 'o', 'B', '/', '/', '/'],
]
```

Formulation of the problem as a search problem

- Objective-test: Check if a piece cannot move;
- Operators: up, down, left, right;
- Pre-conditions: The space immediately next in the direction we want to move is empty.
- Utility of operations: It depends on how many points it will add to the board (provisional values):
  - Piece can move in all directions => 0 Points
  - Piece cannot move in 1 direction => 10 Points
  - Piece cannot move in 2 directions => 30 Points
  - Piece cannot move in 3 directions => 60 Points
  - Piece cannot move in 4 directions => 100 Points
- Evaluation function: Each board will have a value, which is the result of by the player's move. A player will try to maximize this value affecting the opponent, preferentially to 80 points, while the other one will try to make a play which minimizes this effect.

Implementation work already carried out

- Programming language: Python;
- Development Environment: Spyder, displays will be on console;
- Data structures:
    - 2D-array for the board;
    - Search trees for the play's choice;
    - Pieces:



```
class piece:

    def __init_(self):
        self.player = '' #Can be 'A' for Player 1 or 'B' for Player 2
        self.id = 0
        self.x = 0
        self.y = 0

    def set_player(self, c):
        self.player = c
```

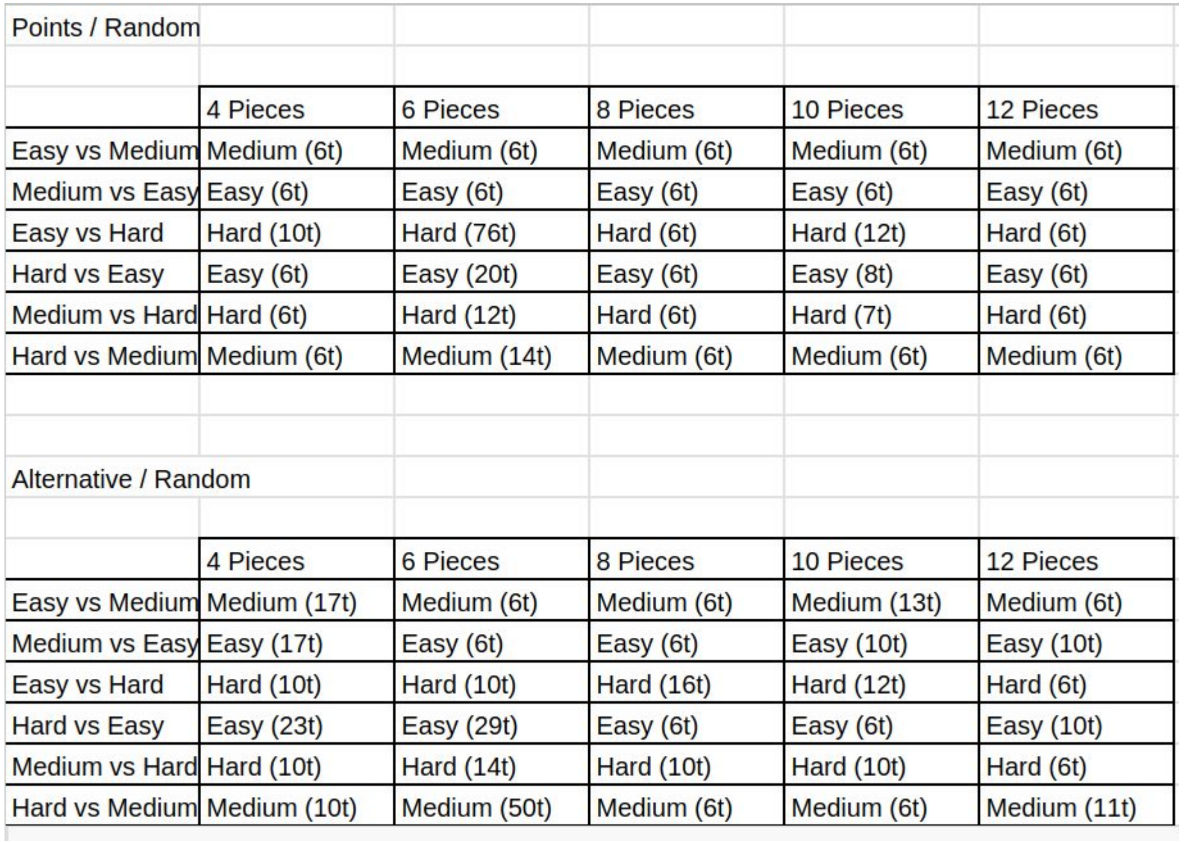
Our approach to the problem

- **Evaluation Function:** Evaluate all pieces of the opponent, grading them with a specific amount of points depending of how many directions each one can move. Then subtract the value of the same evaluation made for our pieces.
- **MiniMax:** The objective of our agent is to minimize the value obtained from the board evaluation, as the value represents the leverage we have over our opponent.
- **Difficulty Depths:** We have three difficulty levels, with different depth limits of exploration for our MiniMax's algorithm tree:
    - **Easy:** Depth Limit = 2
    - **Medium:** Depth Limit = 4
    - **Hard:** Depth Limit = 6

Implemented Algorithms

- **Selection of Next Move:** Minimax with Alpha-Beta Cuts
- **Utility Function:** Evaluates the board piece by piece, summing a specific score by piece depending on the number of directions that piece can't move. After it calculates both the the value for the AI and its opponent, the utility value considered is (AI points - opponent points)
- **Minimax Goal:** Our Utility Agent's objective is to minimize the utility value whereas the objective of the opponent would be to maximize it.
- **Randomization of Operators:** Every time the boards are generated over the current board, the order in which the operators are called is randomized, so that our AI plays in a slightly more unpredictable way.

Experimental Results



Points / Random

	4 Pieces	6 Pieces	8 Pieces	10 Pieces	12 Pieces
Easy vs Medium	Medium (6t)	Medium (6t)	Medium (6t)	Medium (6t)	Medium (6t)
Medium vs Easy	Easy (6t)	Easy (6t)	Easy (6t)	Easy (6t)	Easy (6t)
Easy vs Hard	Hard (10t)	Hard (76t)	Hard (6t)	Hard (12t)	Hard (6t)
Hard vs Easy	Easy (6t)	Easy (20t)	Easy (6t)	Easy (8t)	Easy (6t)
Medium vs Hard	Hard (6t)	Hard (12t)	Hard (6t)	Hard (7t)	Hard (6t)
Hard vs Medium	Medium (6t)	Medium (14t)	Medium (6t)	Medium (6t)	Medium (6t)

Alternative / Random

	4 Pieces	6 Pieces	8 Pieces	10 Pieces	12 Pieces
Easy vs Medium	Medium (17t)	Medium (6t)	Medium (6t)	Medium (13t)	Medium (6t)
Medium vs Easy	Easy (17t)	Easy (6t)	Easy (6t)	Easy (10t)	Easy (10t)
Easy vs Hard	Hard (10t)	Hard (10t)	Hard (16t)	Hard (12t)	Hard (6t)
Hard vs Easy	Easy (23t)	Easy (29t)	Easy (6t)	Easy (6t)	Easy (10t)
Medium vs Hard	Hard (10t)	Hard (14t)	Hard (10t)	Hard (10t)	Hard (6t)
Hard vs Medium	Medium (10t)	Medium (50t)	Medium (6t)	Medium (6t)	Medium (11t)

Conclusions

- Player 2 seems to have an advantage, as a lot of PC vs PC matches end with the victory of player 2;
- Randomization does not have great effect on it, only making the number of turns of the game vary considerably more;
- Only great oscillations of the number of turns to win the game with 12 pieces (6 for each player); in the other ones it is practically constant;
- We had a better perception of the the concepts of AI in games and some of the existent algorithms work.