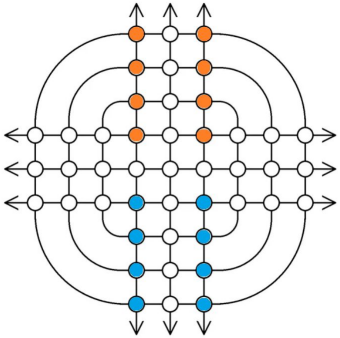
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Wana

Artificial Intelligence
L.EIC 2022/2023

Group 15
Alexandre Costa - up 202005319
Ana Beatriz Fontão - up202003574
Ana Rita Oliveira - up202004155

Project Specification



Our goal for this project is to implement a two-player board game called Wana with different player modes with ai. In order to implement these versions, we will use implement algorithms based on the ones covered in the lectures.

The board for this game is shaped like a cross and each player has 8 pieces. In their turn, a player has to move a single piece as far as they want along a line. When moving a piece, they can't move through other pieces but are allowed to travel off the board and appear from the other side.

The game ends when a player can't move a piece.

Formulation of the problem as a search problem

- **State Representation**:
  - Nparray, where each cell has a number representing its state, [[Integer]].
- **Initial State**:
  - 16 cells set to 1 or 2 representing the players' pieces, 4 in each column adjacent to the middle column (on every row but the middle one). Empty cells set to 0 and forbidden cells set to -1
- **Objective Test**:
  - Nparray, such that at least a cell with value 1 or 2 doesn't have a single valid move
- **Operators**:
  - move
    - Pre-conditions:
      - There isn't another piece in the way of the wanted position
      - The wanted position is empty
      - The cell has the player whose turn it is
    - Effects:
      - The current position of the piece becomes empty (replace with 0)
      - The piece occupies the wanted position by changing the cell's value to 1 or 2

Implementation

- We're using Python to implement the structure and Pygames to display the game
- Because of the inefficiency of classes in Python, we chose to alter the game state representation
- Instead of having the classes 'Piece' and 'Board', we now have single class, 'State', that contains a nparray.
- This array stores the state of the game. Each cell can have one of the following values: -1 if the cell is forbidden; 0 if the cell is empty; 1 if the cell contains a piece of player 1 and 2 1 if the cell contains a piece of player 2
- We also added a features that allows human players to get hints by clicking the hint icon displayed on the screen
- The AI moves are decided by the minimax algorithm that is explained in the following slides

Algorithms

- As mentioned earlier, we used the minimax algorithm to determine the AI moves.
- We implemented the minimax algorithm with alpha-beta cuts inside the ‘minimax’ function that can be found in the Minimax.py file.
- This function takes as parameters the state of the game, the depth, the values of alpha and beta, a bool that identifies if the goal is to maximize or minimize and the evaluation function.
- This algorithm uses recursion and for each iteration it decreases the value of the depth until zero is reached or the game is over and switches the parameter ‘maximizing’ to the opposite value
- In order to create different difficulty levels, we use different evaluation functions and depths.

Approach

- We implemented the following evaluation functions:
  - (Minimize) the number of moves of the opponent (min_num_enemy_moves)
  - (Minimize) the number of moves of the opponent's piece that has the least amount of moves (piece_enemy_moves)
  - (Maximize) the difference between the total of moves of the current player and the total of the opponent's ones (move_dif_eval)
- For the different difficulties of AI we used the following combination of functions and depth:
  - Easy AI: move_dif_eval, depth 2
  - Medium AI: piece_enemy_moves, depth 3
  - Hard AI: min_num_enemy_moves, depth 3

Experimental Results

player1	player2	depth1	depth2	time	num_rounds	winner
move_dif_eval (AI easy)	move_dif_eval (AI easy)	2	2	2.472761631011963	3	1
move_dif_eval (AI easy)	move_dif_eval (AI easy)	2	4	37.029996395111084	3	1
move_dif_eval (AI easy)	move_dif_eval (AI easy)	4	2	119.98047184944153	5	1
piece_enemy_moves (AI medium)	piece_enemy_moves (AI medium)	3	3	84.89894700050354	5	1
piece_enemy_moves (AI medium)	piece_enemy_moves (AI medium)	2	4	272.06515526771545	3	2
piece_enemy_moves (AI medium)	piece_enemy_moves (AI medium)	4	2	204.73730158805847	3	1
min_num_enemy_moves (AI hard)	min_num_enemy_moves (AI hard)	3	3	145.1507432460785	5	1
min_num_enemy_moves (AI hard)	min_num_enemy_moves (AI hard)	2	4	446.2521667480469	2	2
min_num_enemy_moves (AI hard)	min_num_enemy_moves (AI hard)	4	2	699.3413891792297	3	1

Table 1: Comparison of the same evaluation function with different depths

player1	player2	depth1	depth2	time	num_rounds	winner
random	random			0.2541954	28.4	2(1)/3(2)
random	move_dif_eval(AI easy)		2	3.41513	6.2	1(1)/4(2)
random	piece_enemy_moves(AI medium)		3	46.540408	5.4	0(1)/5(2)
random	min_num_enemy_moves(AI hard)		3	46.501678	3.6	0(1)/5(2)
move_dif_eval(AI easy)	piece_enemy_moves(AI medium)	2	3	27.871801614761353	4	1
piece_enemy_moves(AI medium)	move_dif_eval(AI easy)	3	2	57.43560767173767	6	1
piece_enemy_moves(AI medium)	min_num_enemy_moves(AI hard)	3	3	121.6748628616333	8	2
min_num_enemy_moves(AI hard)	piece_enemy_moves(AI medium)	3	3	51.68491005897522	2	1
move_dif_eval(AI easy)	min_num_enemy_moves(AI hard)	2	3	18.705236196517944	2	2
min_num_enemy_moves(AI hard)	move_dif_eval(AI easy)	3	2	58.68344187736511	4	1

Table 2: Comparison of different evaluation functions

Conclusions

- After examining the results, we came to the conclusion that depths larger or equal to 4 are too time consuming for a game, even though they produce better results.
- Initially we thought that move_dif_eval would be the best evaluation function, since the strategy tends to work well for other games, but through testing, we realized this was not the case. For this reason, we used it as the easy ai mode. We think this might be the case since the strategy is more defensive than the other ones and the depths used are not big enough to make a difference.
- Though similar, after testing we realized that min_num_enemy_moves is overall better than piece_enemy_moves for the depths tested.

Work References

- https://boardgamegeek.com/boardgame/364012/wana
- https://www.pygame.org/docs/
- Slides available in Moodle
- Notebooks used in the TP classes (AI2 AdversarialSearch)