Pahtum

Projeto EIACD

Realizado por: Catarina Abrantes, Liliana Silva e Mariana Fonseca.

Objetivo do jogo

Formar mais linhas horizontais ou verticais de pedras da cor do jogador do que o adversário

Linhas contínuas, ou seja, compostas por peças da mesma cor e não podem ser interrompidas por peças do adversário ou casas bloqueadas.

Formulação do Problema

Estado inicial

- M[7,7] preenchida com 0 = casas vazias e -1 = buracos negros;
- Turno do jogador com as peças brancas;
- 5 posições aleatorias = os buracos negros, número que pode variar.

Operadores

- Put_white(x,y) = colocar uma peça branca numa casa vazia;
- Put_black(x,y),= colocar uma peça branca numa casa vazia.

	Put_white (x,y)	Put_black (x,y)
Pré-condições	Turno do jogador com peças brancas; A casa (x, y) estar dentro dos limites do tabuleiro (ou seja, x e y estar entre (0,0) e (6,6)); A casa (x, y) não ser um buraco negro (-1) e a casa (x, y) estar vazia, isto é, M[x][y] == 0.	Turno do jogador com peças pretas; A casa (x, y) estar dentro dos limites do tabuleiro ((ou seja, x e y estar entre (0,0) e (6,6)); A casa (x, y) não ser um buraco negro (-1) e a casa (x, y) estar vazia, isto é, M[x][y] == 0.
Efeitos	A casa escolhida passa a ter uma peça branca; atualiza M[x][y] para 1. Ocorre uma mudança de turno para o jogador com peças pretas.	A casa escolhida passa a ter uma peça preta; atualiza M[x][y] para 2.; Ocorre uma mudança de turno para o jogador com peças brancas.
Custo	1	1

Teste objetivo

Verifica se o tabuleiro está completo, ou seja, verifica se não existem mais casas vazias no tabuleiro. Neste sentido, determina se o jogo terminou.Este teste pode ser representado pela seguinte função:

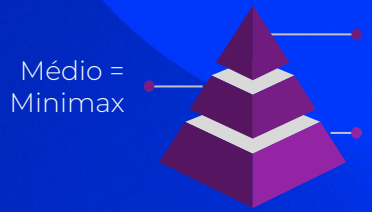
all(M[x][y] != 0 for x in range(7)) for y in range(7))

- Determina-se o vencedor: O jogador com o maior número de pontos ao final do jogo é declarado vencedor. Condição de Empate: Se os jogadores tiverem o mesmo número de pontos, é declarado um empate.

Utility Function

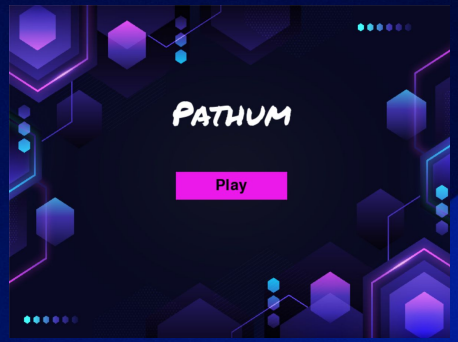
+1	-1	0
Vitória	Derrota	Empate

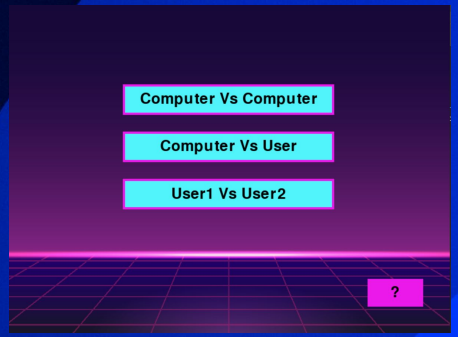
1. Utilizador VS Utilizador
2. Utilizador VS Computador
3. Computador VS Computador

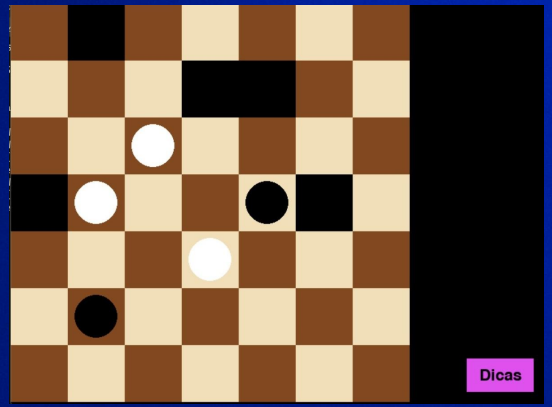


Expert vs Average → Expert vs Beginner → Beginner vs Average

Interface gráfica do jogo







The game is played by players alternately placing white and black stones on the board. Each player places one stone per turn. A stone can only be placed on squares which do not contain another stone or a black hole. The goal of the game is to create more connected lines of stones than an opponent. A connected line must be either vertical or horizontal and may contain only stones of the same colours, no opponent stones or black holes. Points are awarded for lines of three or more at the end of the game as follows:

- 3 stones = 3 points
- 4 stones = 10 points
- 5 stones = 25 points
- 6 stones = 56 points
- 7 stones = 119 points

The game is finished when the board is full of stones and black holes. The player with higher amount of points is the winner. If both players have the same points, the game is a draw.

EXIT

Algoritmos

Minimax

- Projetada para determinar recursivamente o melhor movimento, explorando possíveis estados futuros do jogo até uma profundidade determinada.
- Utiliza a alpha-beta cuts para otimizar o processo de busca, alternando entre estratégias de maximização e minimização com base no jogador.

Monte Carlo Tree Search

- Simula múltiplas jogadas aleatórias, a partir do estado atual do jogo para determinar estatisticamente o movimento mais promissor.
- Usa o resultado dessas simulações para escolher movimentos, particularmente útil em cenários de jogo complexos e probabilísticos, onde calcular todos os resultados possíveis é computacionalmente inviável.

Heuristícas

		Descrição	Implementação
1	**Controlo do centro**	Posicionar as peças no centro do tabuleiro. O centro é estrategicamente valioso para formar linhas pontuáveis e bloquear adversários.	Na função evaluate_board, as peças no centro recebem um bónus de pontos (center_bonus).
2	**Bloqueio**	Valoriza jogadas que bloqueiam os adversários de completar linhas pontuáveis., reduzindo a pontuação do adversário.	Na função evaluate_board, o jogador ganha pontos (block_bonus) por bloquear linhas do adversário.
3	**Avaliação Dinâmica**	Ajustar a avaliação, baseada na proximidade de vitória.	Na função evaluate_boad, se um jogador está próximo de ganhar, a pontuação aumenta para refletir a urgência estratégica.

Funções de avaliação

Função calculate_points

- Encontra-se na classe Pathum Game.
- erifica cada posição no tabuleiro para identificar peças de um jogador específico, calculando pontos para cada uma dessas peças usando a função points_for_line. Essa abordagem soma todos os pontos atribuídos para determinar a pontuação total do jogador.

Função points_for_line

- Encontra-se na classe Pathum Game.
- Avalia diretamente os pontos para uma linha específica a partir de uma posição dada, aplicando o sistema de pontuação predefinido (3 pontos para 3 em sequência, etc.). Este método é crucial para avaliar e pontuar movimentos potenciais durante o jogo.

Função evaluate_board

- Objetivo: Avaliar o estado do tabuleiro com base nos pontos atuais e nos fatores estratégicos que podem influenciar futuras jogadas.
- Heurísticas Implementadas: Controle do Centro, Bloqueio, Avaliação Dinâmica
- Aumenta a pontuação total, se a pontuação calculada indicar que o jogador está muito próximo da vitória.
- Método: Começa com o cálculo da pontuação base, usando calculate_points. Aplica as heurísticas para ajustar essa pontuação, adicionando bônus.

Operadores

1

Método make_move:

Para atualizar o estado do jogo colocando uma peça no tabuleiro. Está na classe PathumGame. Garante que os movimentos são feitos apenas em locais vazios e regista cada movimento para possíveis desfazimentos, o que é crucial para simulações como as usadas no algoritmo Minimax

2

Método undo_move:

Permite reverter o último movimento feito Crítico para a exploração recursiva dos estados do jogo no algoritmo Minimax.

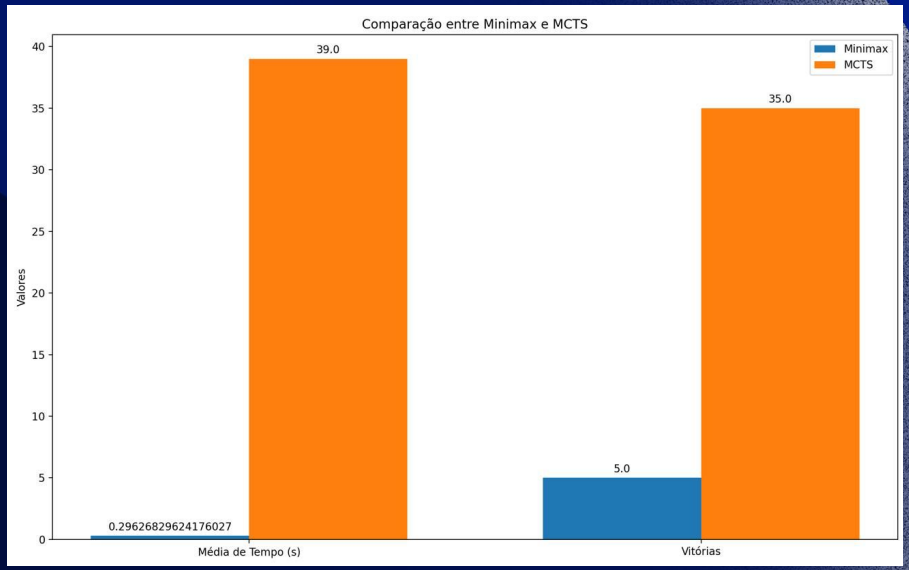
3

Método get_legal_moves:

Fornece uma lista de todos os movimentos legais possíveis a partir do estado atual do jogo, necessário para ambos os algoritmos Monte Carlo e Minimax funcionarem.

Resultados Experimentais


```
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.289355039596558, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.1394548416137695, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.245429039001465, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.208889007568359, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.245879888534546, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.282668828964233, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.520954847335815, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.36109471321106, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.379362106323242, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.3632378578186035, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.354353904724121, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.639996290206909, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.408056974411011, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.4646687507629395, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.466424226760864, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10},
{'algoritmo': 'MCTS', 'tempo_de_execucao': 4.458159923553467, 'numero_de_movimentos': 44, 'vencedor': 'white',
'simulacoes': 10}]
resultados_minimax = [{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.8409500122070312, 'numero_de_movimentos': 44,
'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.9951348304748535, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.8962960243225098, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.9115111827850342, 'numero_de_movimentos': 44, 'vencedor': 'black'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.8503942489624023, 'numero_de_movimentos': 44, 'vencedor': 'black'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.8683860301971436, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.0521848201751, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.9286229610443115, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.0656721591949463, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.0138559341430664, 'numero_de_movimentos': 44, 'vencedor': 'black'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.085237979888916, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.074779748916626, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.1559019088745117, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.1214499473571777, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.0723071098327637, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.050926923751831, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.4647688865566162, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.1478347778320312, 'numero_de_movimentos': 44, 'vencedor': 'draw'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 1.0659220218658447, 'numero_de_movimentos': 44, 'vencedor': 'white'},
{'algoritmo': 'Minimax', 'tempo_de_execucao': 0.9980590343475342, 'numero_de_movimentos': 44, 'vencedor': 'draw'}]  #
```

Tempo de execução: Monte Carlo tree search > Minimax

Número de vitórias: Monte Carlo tree search >> Minimax

Referências e Materiais

- Apontamentos das aulas
- Modelos de linguagem conversacional
- https://brainking.com/en/GameRules?tp=72
- https://br.freepik.com/fotos-vetores-gratis/fundo-jogos

Conclusões

- Trabalhar Python (Pygame)
- Desensolver interface gráfica
- https://brainking.com/en/GameRules?tp=72
- https://br.freepik.com/fotos-vetores-gratis/fundo-jogos