Fanorona/Fanoron-Tsivy

Tópico 2B - Trabalho prático 1 - IA - Grupo A1_22

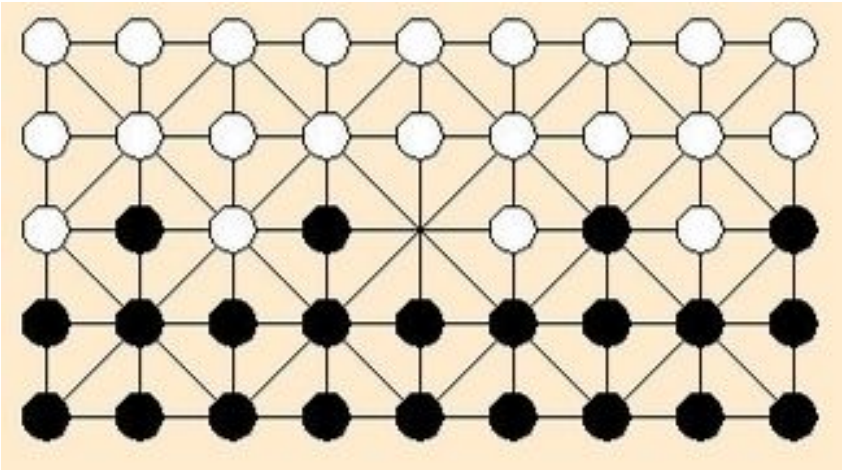
Félix Martins, up202108837
Pedro Lima, up202108806
Pedro Januário, up202108768

O jogo

O Fanorona/Fanoron-Tsivy é um jogo de tabuleiro para dois jogadores, originalmente jogado num tabuleiro de 5 casas por 9. Cada jogador tem 22 peças e o objetivo é capturar todas as peças do adversário. Todas as peças de cada jogador têm o mesmo papel e podem mover-se na horizontal, na vertical ou na diagonal. Há uma restrição adicional que permite movimentos diagonais apenas em determinadas posições.

Ocorre uma captura numa de duas situações: quando uma peça do jogador A “se encosta” a uma ou mais peças consecutivas de B, capturando todas as peças deste na direção e sentido do movimento; ou no caso inverso, quando A afasta uma peça sua na referida situação de “encosto”.

Sempre que um jogador está em condições de capturar uma ou mais peças do adversário, é obrigado a fazê-lo, podendo, no movimento seguinte, escolher entre capturar novamente com a mesma peça ou passar a vez, não sendo permitido o movimento livre.



- Representação do Estado: Tabuleiro (matriz de inteiros), jogador atual e lista de movimentos efetuados durante a jogada atual.
- Estado inicial: Figura acima
- Teste de Objetivo: Um jogador ganha quando o adversário ficar sem peças no tabuleiro.
- Funções heurísticas utilizadas serão detalhadas adiante.

Trabalhos relacionados

- Wikipédia, a Enciclopédia Livre (inglês): https://en.wikipedia.org/wiki/Fanorona
- Regras (inglês): https://www.woodtactics.com/uploads/b/e8bb97b0-bd3b-11e9-8a02-9b8433820a8d/Fanorona.pdf
- Vídeo com explicação das regras: https://www.youtube.com/watch?v=_8Oi7p9TzgA
- Fanorona Online: https://kbhgames.com/game/fanorona-board-game

Esclarecimento prévio

É importante fazermos uma distinção entre os termos “jogada” e “movimento”, no âmbito do nosso problema.

Entendemos por movimento toda a alteração da posição de uma peça por parte do seu detentor.

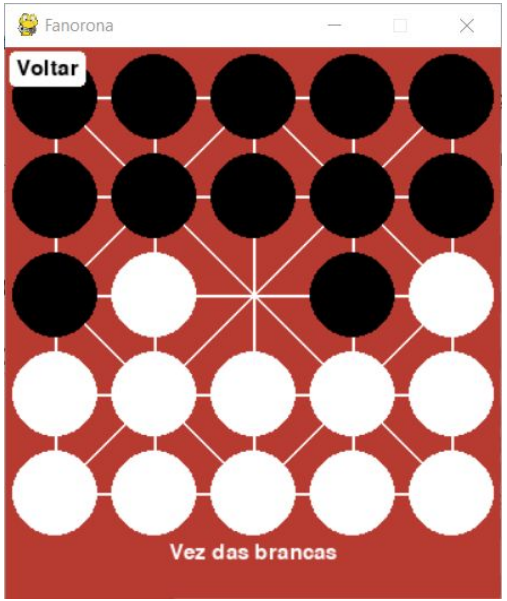
Uma vez que, em certas situações, um jogador pode fazer mais do que um movimento de seguida, no âmbito dos algoritmos de pesquisa, consideramos que uma jogada pode incorporar vários movimentos.

Formulação do Problema - Operadores

Nome	Pré-condições	Efeitos
Captura por aproximação	Tem que existir exatamente um espaço entre a nossa peça e a peça adversária (na diagonal, horizontal ou vertical). O movimento pode ser diagonal se for feito numa posição com diagonais definidas. Não pode sair do tabuleiro. A posição final não pode estar na lista de movimentos já efetuados. No caso de uma jogada composta por mais que um movimento, o movimento não pode ser efetuado na mesma direção que o último movimento.	Movimenta uma casa no sentido da peça adversária e captura todas as peças adversárias consecutivas no sentido do movimento. O jogador ganha o direito de efetuar mais capturas com a mesma peça, mas pode escolher não o fazer. O movimento efetuado é adicionado à lista de movimentos efetuados durante a jogada.
Captura por afastamento	Tem que ter exatamente uma peça adversária adjacente (na diagonal, na horizontal ou na vertical) e um espaço livre no sentido oposto. O movimento pode ser diagonal se for feito numa posição com diagonais definidas. Não pode sair do tabuleiro. A posição final não pode estar na lista de movimentos já efetuados. No caso de uma jogada composta por mais que um movimento, o movimento não pode ser efetuado na mesma direção que o último movimento.	Movimenta uma casa para o sentido oposto, captura das peças adversárias consecutivas no sentido do movimento. O jogador ganha o direito de efetuar mais capturas com a mesma peça, mas pode escolher não o fazer. O movimento efetuado é adicionado à lista de movimentos efetuados durante a jogada.
Movimento livre	Nenhuma captura disponível (por afastamento ou aproximação). O movimento pode ser diagonal se for feito numa posição com diagonais definidas. Não pode sair do tabuleiro.	A peça movimenta-se uma casa sem capturar nenhuma peça adversária.

Implementação

- Implementação da lógica do jogo
  - Tabuleiro com tamanho “livre”
  - Jogadas/movimentos
- Implementação dos algoritmos e de simulações
  - Heurísticas
  - Minimax com cortes α, β
  - Monte Carlo Tree Search
- Interface (pygame)
  - Diferentes tamanhos do tabuleiro
  - Escolha do modo de jogo de cada jogador
    - Humano, aleatório, Minimax com diferentes heurísticas e variantes de Monte Carlo Tree Search



Heurísticas implementadas

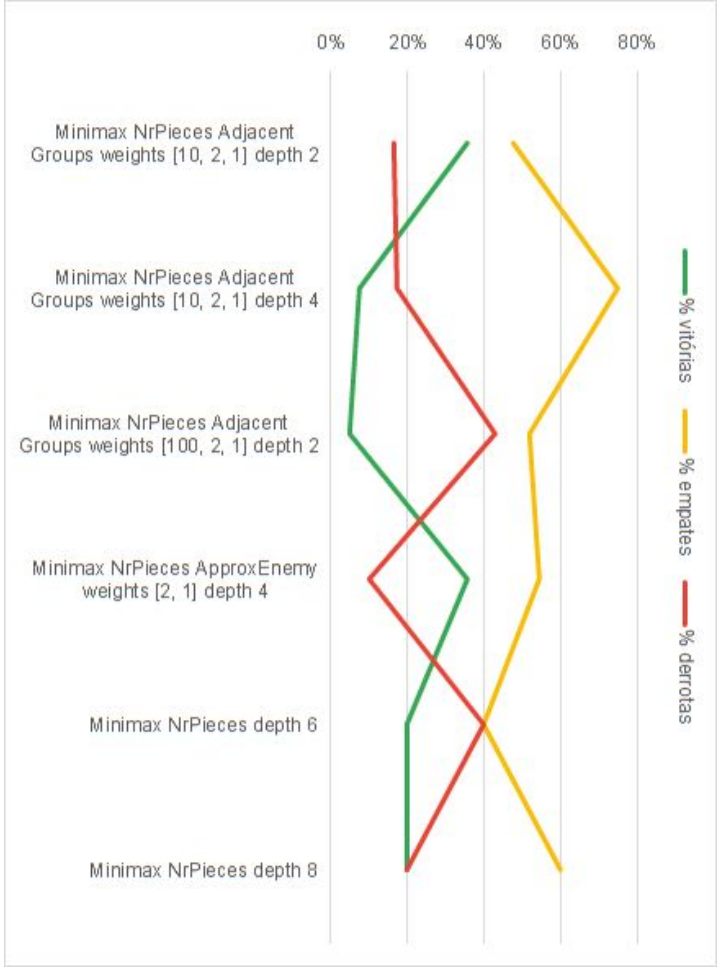
- **Vitória**
- Número de **peças**
  - Notar que a diferença entre os números de peças de cada equipa é mais “valiosa” quando existem menos peças
- Número de **grupos** de peças
  - Com o objetivo de colocar as peças mais “separadas”
- Número máximo de peças **adjacentes** em linha (diagonal, vertical ou horizontal)
- **Aproximação** ao inimigo:
  - Média da distância mínima das peças do jogador às peças do inimigo
  - Aproxima-se se estiver a ganhar, afasta-se se estiver a perder
- Controlo do **centro**
  - Média da distância de cada peça ao centro

Algoritmos implementados

- Minimax
  - Cortes α, β
  - Uso e teste com diferentes heurísticas
- Monte Carlo Tree Search
  - Extra: Monte Carlo Tree Search com heurística em vez de *rollout*

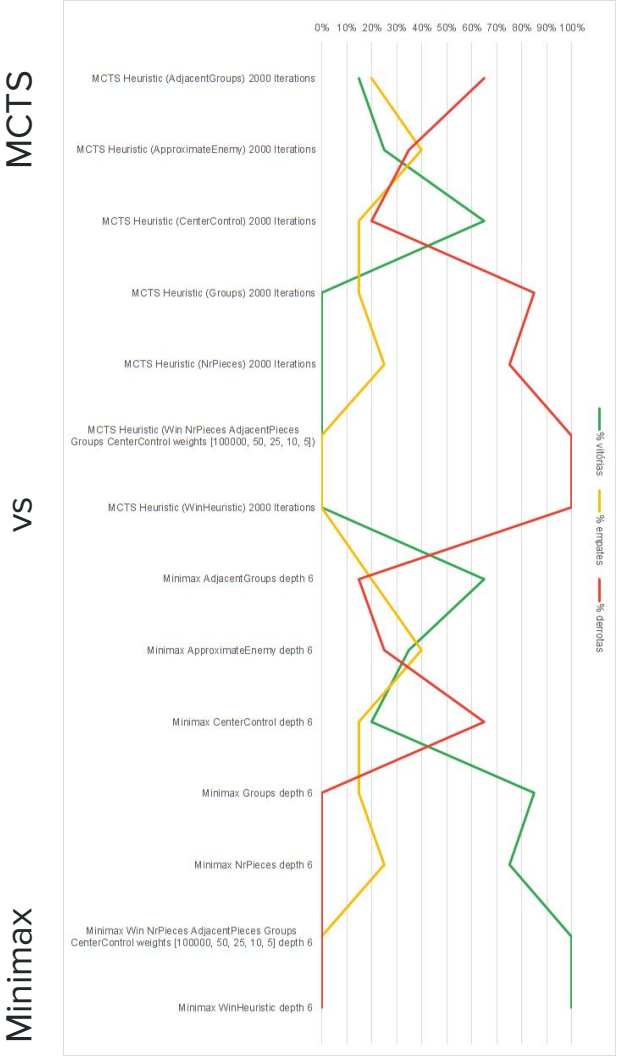
Resultados experimentais

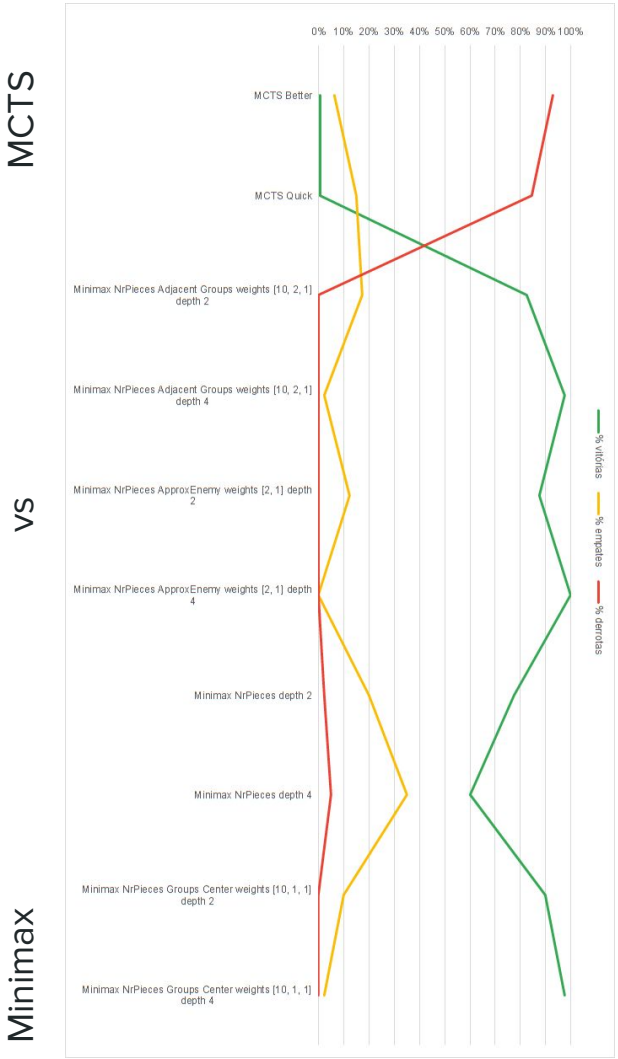
Aleatório vs MiniMax/MCTS



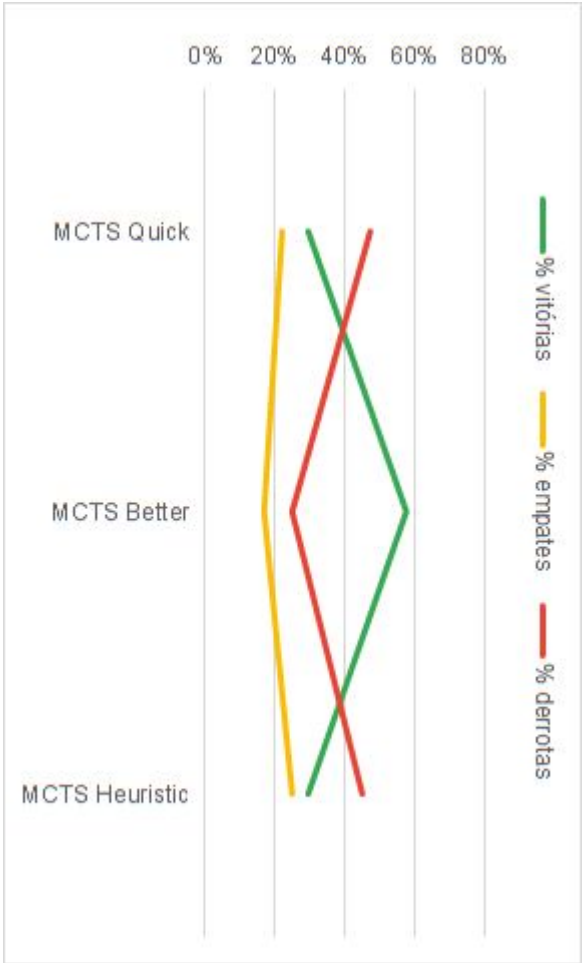
Minimax vs Minimax

Resultados experimentais





Resultados experimentais



Para colmatar a colisão entre a quantidade de dados obtidos e a escassez de espaço neste documento, disponibilizamos, em anexo, um ficheiro com as medições recolhidas, bem como os gráficos apresentados, numa melhor resolução.

Conclusões

Heurísticas

- Como esperado, a heurística de **número de peças** levou aos melhores resultados. O **número de grupos** e a **aproximação ao inimigo** resultaram bem.
- A heurística de **controlo de centro** deu resultados positivos, mas a de **aproximação ao inimigo** foi superior.
- O **número de peças adjacentes** não funcionou como esperado e levou a resultados relativamente piores.

Minimax

- Funcionou bem e relativamente rápido.
- Resultou melhor com boas heurísticas e menos profundidade.
- Com tabuleiros maiores e na fase final do jogo, profundidades maiores levam a tempos de processamento bastante elevados, devido ao elevado *branching factor*.

Monte Carlo Tree Search

- Monte Carlo Tree Search teve um pior desempenho do que o esperado.
- Leva muitas iterações para encontrar uma boa solução (> 1000), o que demora muito tempo. Quanto maior o tabuleiro, mais grave é este problema, uma vez que tanto o *branching factor* como o tempo de cada *rollout* aumentam.
- A utilização de heurísticas, em vez de *rollouts*, diminui o tempo de cada iteração, mas não resolve o problema.

Bibliografia

1. Schadd, Maarten & Winands, Mark & Uiterwijk, Jos & Herik, H. & BERGSMA, MAURICE. (2008). Best Play in Fanorona Leads to Draw. New Mathematics and Natural Computation. 04. 369-387. 10.1142/S1793005708001124.
2. Fanorona, https://en.wikipedia.org/wiki/Fanorona [acedido 30 mar 2024]
3. How To Play Fanorona Tsivy Edition, https://www.woodtactics.com/uploads/b/e8bb97b0-bd3b-11e9-8a02-9b8433820a8d/Fanorona.pdf [acedido 30 mar 2024]
4. Fanorona Online, https://kbhgames.com/game/fanorona-board-game [acedido 30 mar 2024]