Inteligência Artificial

Thiago Henrique Leite da Silva, RA: 139920

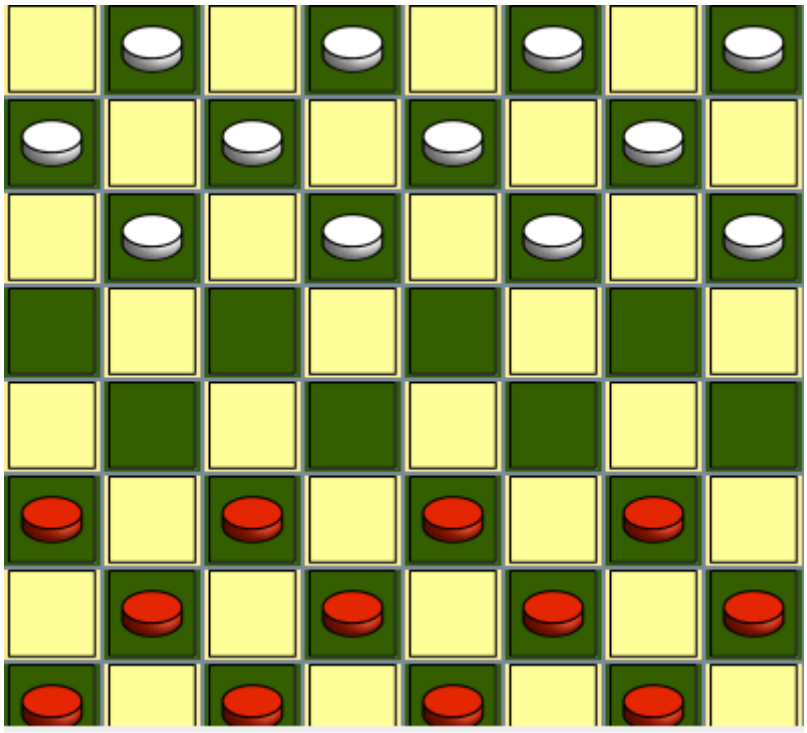
AULA8: Exercício prático jogos minimax e poda alfa-beta

1) (1,0) Pesquise uma implementação usando algumas das técnicas vistas na Aula 08 em jogos (minimax, poda alfa-beta, jogos estocásticos, etc):

Créditos a este brilhante trabalho: ***https://github.com/danielwohlgemuth/damas***

a) Qual o tipo de jogo

O jogo escolhido por mim foi Damas, um jogo de tabuleiro para dois jogadores, sua versão mais popular é a jogada em um tabuleiro 8x8, com 12 peças cada jogador. O objetivo do jogo é capturar ou imobilizar as peças do seu oponente, o jogador que conseguir capturar todas as peças do adversário vence a partida. No início do jogo, as peças são colocadas nas casas escuras e se movimentam na diagonal, de uma em uma casa, a não ser na captura, onde pulamos a casa em que está a peça do adversário, ou quando promovemos uma peça para “Dama”, ou seja, chegamos na primeira linha da direção oposta à qual iniciamos o jogo.



b) Quais as técnicas empregadas

Um ponto bastante interessante desta implementação que encontrei é que o programador dá a opção de escolhemos 4 possibilidades de executar o jogo, utilizando o algoritmo Minimax, a poda Alpha-Beta, aprendizagem por reforço, ou jogadas aleatórias. Podemos executar um jogo por exemplo entre o Minimax e Alpha-Beta, Alpha-Beta e jogadas aleatórias, entre outras combinações. Lembrando que é uma partida assistida, iremos ver nosso algoritmo jogando de acordo com os métodos que definirmos, não seremos nós propriamente jogando contra a máquina. Outro ponto interessante do projeto é que ele tem uma interface gráfica, a imagem acima do tabuleiro foi tirada de lá, podemos optar por ver o jogo nesta interface ou apenas ver a saída com um relatório da partida.

Aprofundando um pouco mais nos algoritmos vistos em aula, a estratégia usada no algoritmo minimax e na poda, foi avaliar em cada rodada todos os possíveis movimentos do jogador, executar o algoritmo minimax para escolhermos aquela em que:

- Diminuímos a quantidade de peças do adversário;
- Não colocamos nossa peça em uma posição de risco;

Já em relação a poda, a estratégia foi a mesma vista em aula para deixar o algoritmo mais rápido, executando menos comparações.

c) Quais outras estratégias foram usadas além dessas vistas em aula (ex: tabela com jogadas prontas, etc)

Nesta implementação não foram usadas outras estratégias além das vistas em aula relacionado a parte do algoritmo minimax e poda alpha-beta; porém uma estratégia já comentada utilizada e que achei muito interessante foi a aprendizagem por reforço, onde nós temos um tempo de aprendizado para que o algoritmo aprenda a jogar o jogo, e assim que ele aprende, o jogo se inicia, no relatório da partida temos o tempo que ele levou para aprender. No aprendizado levamos em conta o tempo de aprendizado, passos sem captura, passos com captura, se conseguimos promover o peão alguma vez, entre outras maneiras de avaliarmos o aprendizado.

d) Analise o código, execute o mesmo e anexe um print da saída

O código foi muito bem escrito em Java e está extremamente organizado, nesta fase de testes, executei o jogo de 4 formas para vermos os desempenhos:

- Alpha-Beta x Minimax

```
thiagoleite@IA:~/damas/src$ javac Main.java Jugador.java Random.java Minimax.java AlphaBeta.java RL.java Tablero.java
thiagoleite@IA:~/damas/src$ java Main
Tiempo entrenamiento (ms): 0.0
Ratio Minimax(2)[0]: 0.0
Tiempo promedio (ms) 68.32000000000002
Expandidos 16884.500000000007
Ratio empates: 1.0
Ratio AlphaBeta(2)[1]: 0.0
Tiempo promedio (ms) 30.669999999999984
Expandidos 7430.499999999989
Jugadas promedio: 145
thiagoleite@IA:~/damas/src$
```

- Minimax x Random

```
thiagoleite@IA:~/damas/src$ javac Main.java Jugador.java Random.java Minimax.java AlphaBeta.java RL.java Tablero.java
thiagoleite@IA:~/damas/src$ java Main
Tiempo entrenamiento (ms): 0.0
Ratio Minimax(2)[0]: 0.9
Tiempo promedio (ms) 29.38
Expandidos 4753.009999999999
Ratio empates: 0.1
Ratio Random[1]: 0.0
Tiempo promedio (ms) 0.33000000000000007
Expandidos 0.0
Jugadas promedio: 57
thiagoleite@IA:~/damas/src$
```

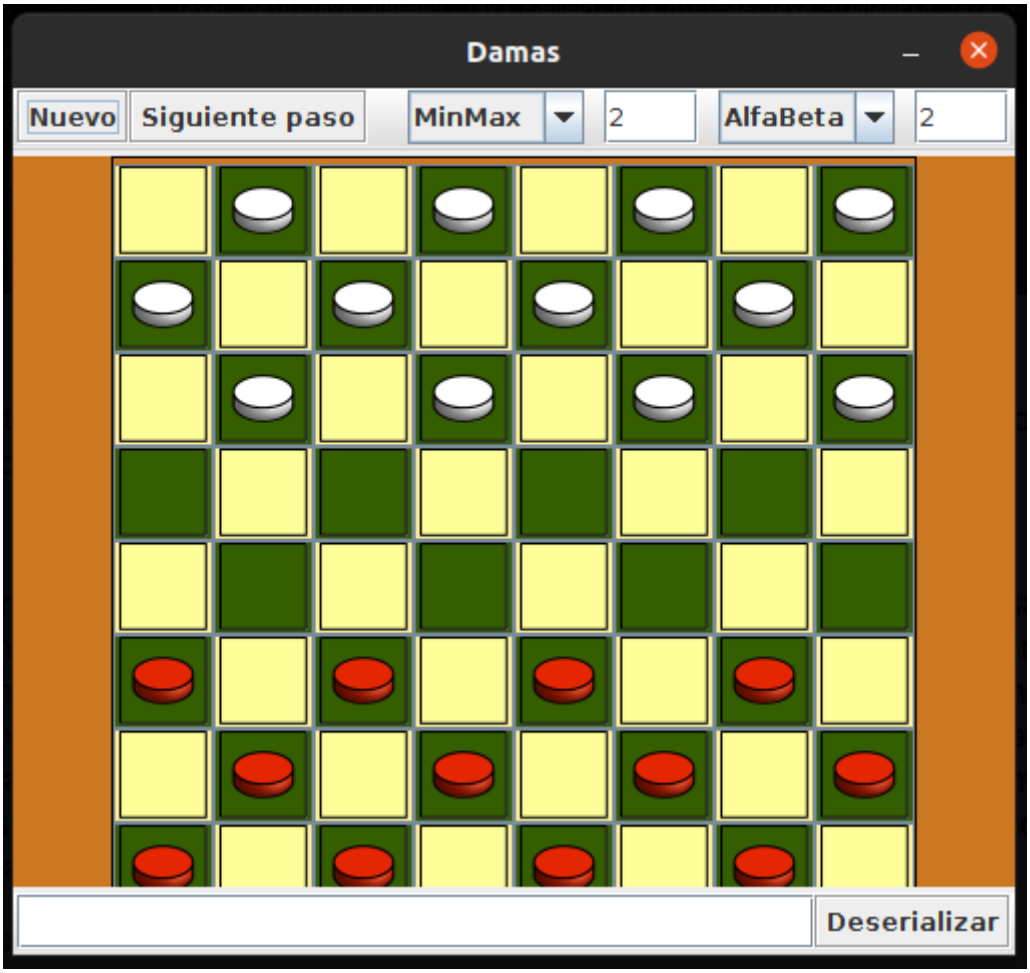
- Alpha-Beta x Random

```
thiagoleite@IA:~/damas/src$ javac Main.java Jugador.java Random.java Minimax.java AlphaBeta.java RL.java Tablero.java
thiagoleite@IA:~/damas/src$ java Main
Tiempo entrenamiento (ms): 0.0
Ratio AlphaBeta(2)[0]: 0.9
Tiempo promedio (ms) 18.07
Expandidos 2899.04
Ratio empates: 0.1
Ratio Random[1]: 0.0
Tiempo promedio (ms) 0.41000000000000014
Expandidos 0.0
Jugadas promedio: 65
thiagoleite@IA:~/damas/src$
```

Também tentei testar alguma das possibilidades contra a aprendizagem por reforço, mas o algoritmo colocou o I7 pra chorar.

Em linhas gerais, podemos observar que, quando temos dois algoritmos que sabem o que estão fazendo jogando um contra o outro, como foi o caso do teste Alpha-Beta x Minimax, o jogo é mais longo, diferentemente quando os algoritmos vistos em aula enfrentaram o Random, em que eles conseguiram vencê-lo rapidamente. Um ponto curioso é que apesar do Minimax ter feitas menos jogadas contra o Random, ele expandiu quase duas vezes mais nós que o algoritmo Alpha-Beta, que fez 8 jogadas a mais; isso acontece também pois o adversário faz movimentos completamente diferentes a cada jogada.

O algoritmo executado com a interface gráfica é da seguinte forma:



Para rodar o projeto localmente basta seguir a descrição fornecida no GitHub. :)