Pivit

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

Faculdade de Engenharia da Universidade do Porto

Grupo Pivit:

Pedro Leite Galvão - 201700488

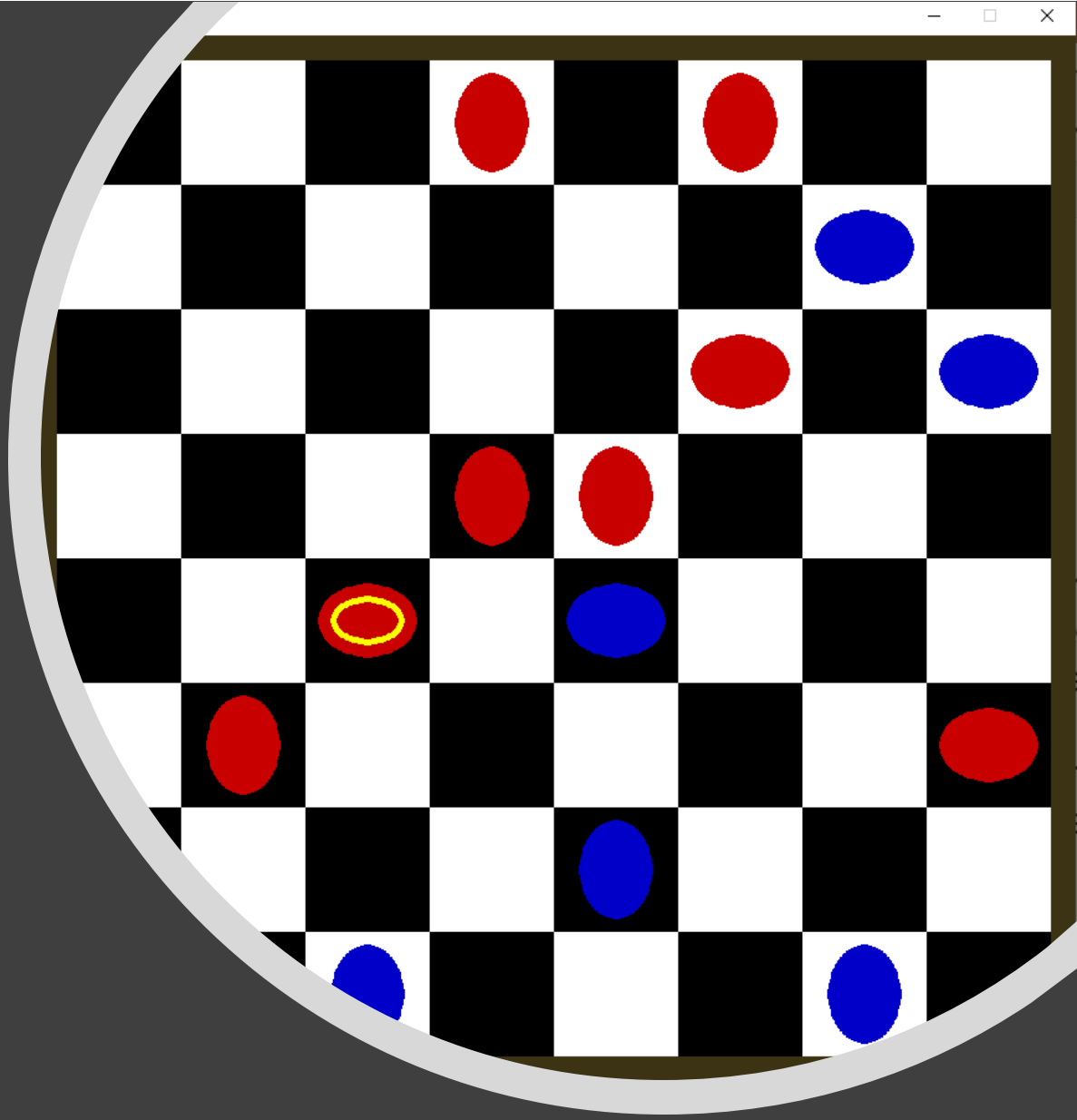
Vasco Marques Lopes Teixeira - 201802112

Decrição do Jogo

O nosso trabalho centra-se na replicação do jogo de tabuleiro “Pivit”, que é um jogo de tabuleiro para dois jogadores.

O campo de jogo é um tabuleiro quadrado como o de xadrez, 8x8 ou 6x6. As peças podem capturar-se umas às outras como no xadrez e ser promovidas de *minions* a *masters*.

Cada peça tem na sua face setas que apontam na direcção em que se pode mover – horizontal ou vertical. Todas as vezes que uma peça se move, ela gira 90 graus no sítio onde ficou, mudando de direcção

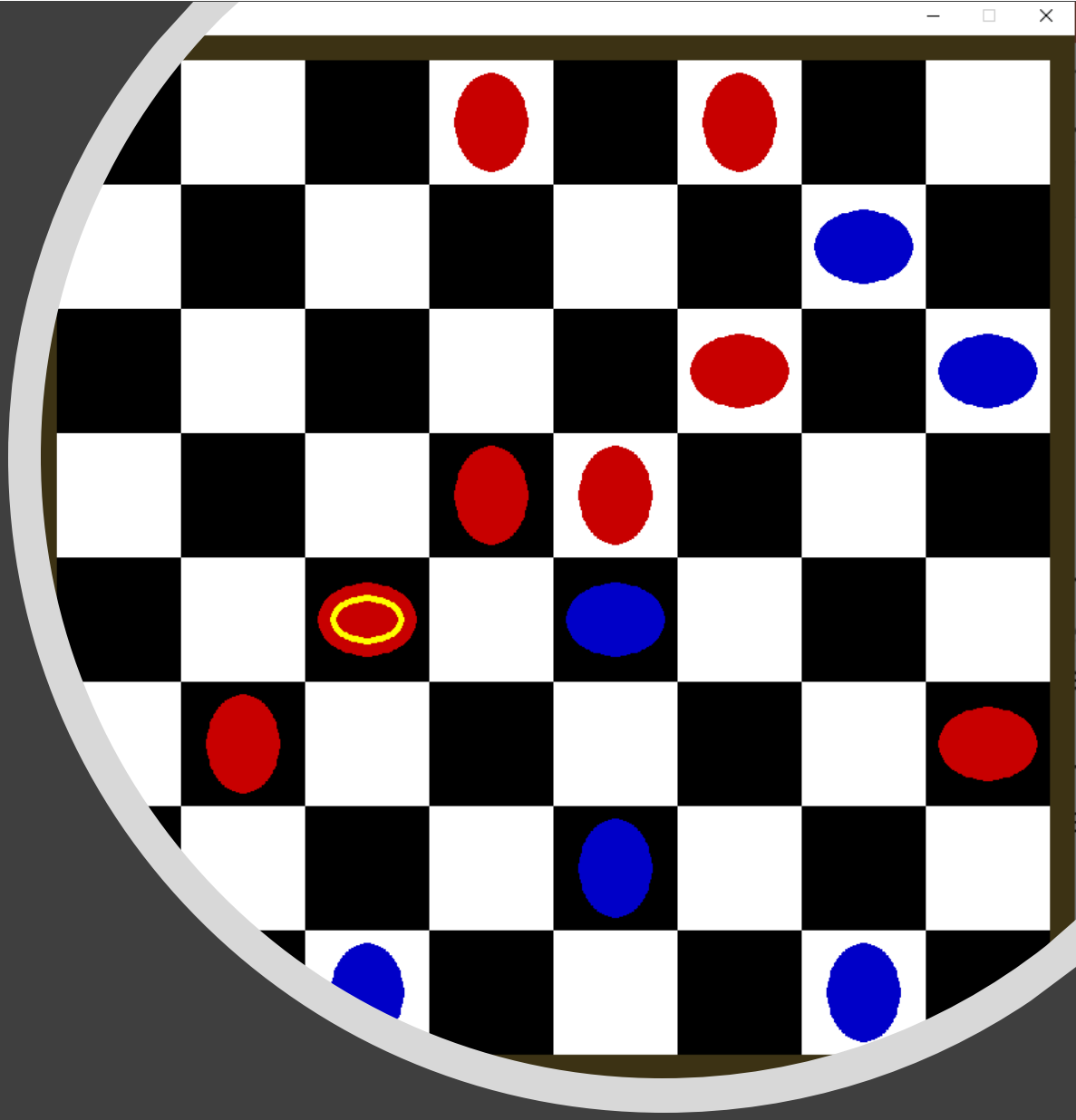


Descrição do Jogo

As peças básicas, chamadas de *minions*, devem movimentar-se para um quadrado de uma cor diferente. Ou seja, se eles começam num quadrado preto, devem movimentar-se para um quadrado branco. Peças promovidas, chamadas *masters*, podem pousar em qualquer lugar ao longo da sua linha.

As peças não podem saltar por cima de outras e capturam ao aterrar numa casa ocupada pelo inimigo. A promoção ocorre quando um *minion* se movimenta para um espaço do canto do tabuleiro.

O jogo termina quando todos os *minions* de ambos os jogadores são capturados. O vencedor é o jogador com mais *masters* no tabuleiro.



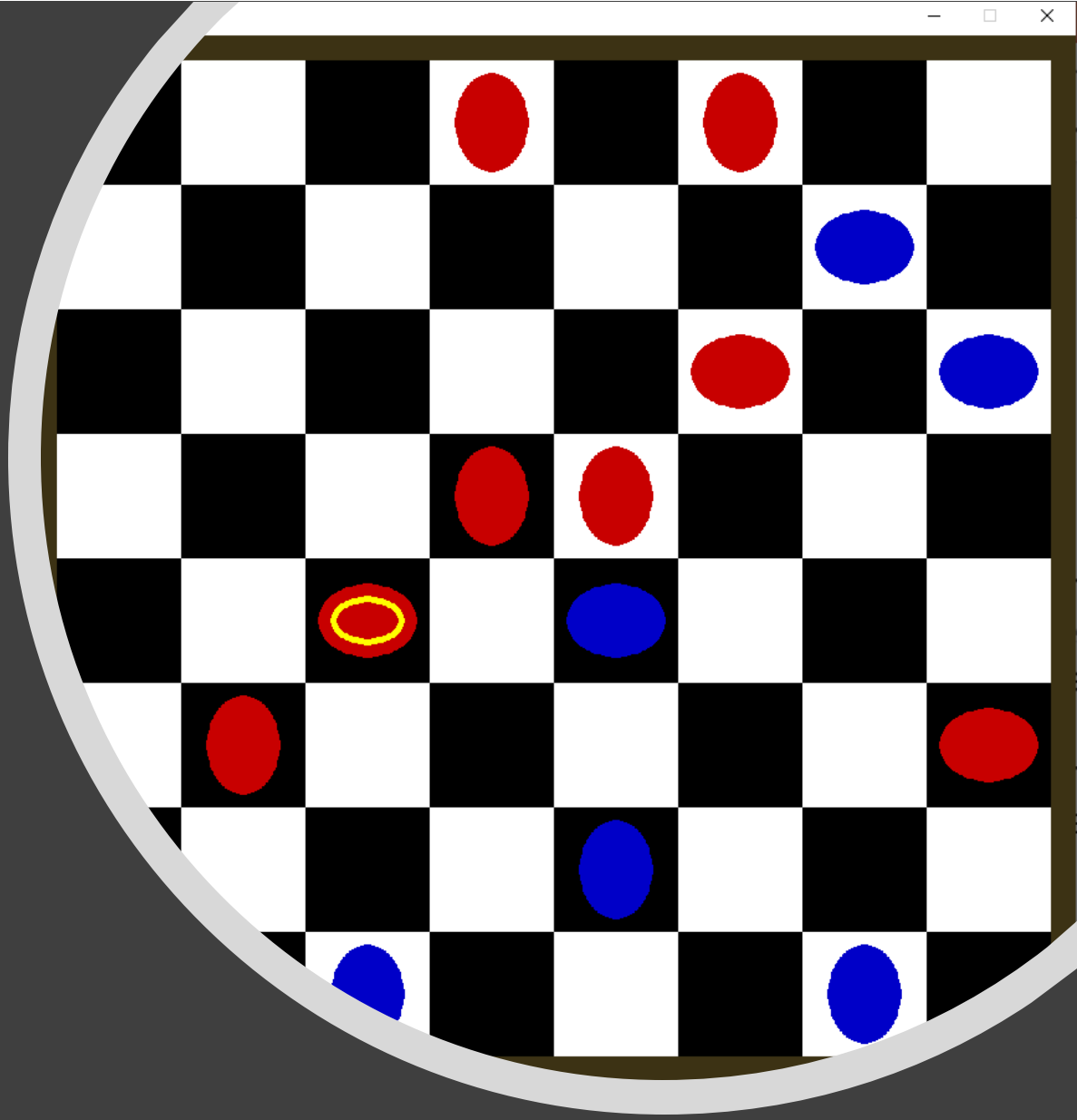
Formulação do Problema

- **Representação do estado:**

O jogo é representado por um tabuleiro com peças nas casas das extremidades menos nos cantos. Para isso usamos uma lista de listas para representar o tabuleiro em que cada índice representa uma casa. Dentro de cada posição da lista pode estar 0 que representa uma casa vazia, ou um objecto "Piece" que representa uma peça. Utilizamos também uma variável "active_player" booleana que muda a cada movimento para determinar qual o jogador a jogar.

- **Estado inicial:**

No estado inicial as peças dos dois jogadores ficam colocadas nas linhas e colunas encostadas às extremidades. As peças são colocadas alternadamente pela cor e os cantos do tabuleiro ficam vazios assim como todas as restantes casas.



Formulação do Problema

- **Operadores:**

As possíveis operações a serem realizadas a partir de um estado são determinadas seguindo as regras de jogo já descritas. Dado que há muitos movimentos possíveis, estes operadores e seus efeitos devem ser determinados a cada jogada. Não existem custos envolvidos nas jogadas.

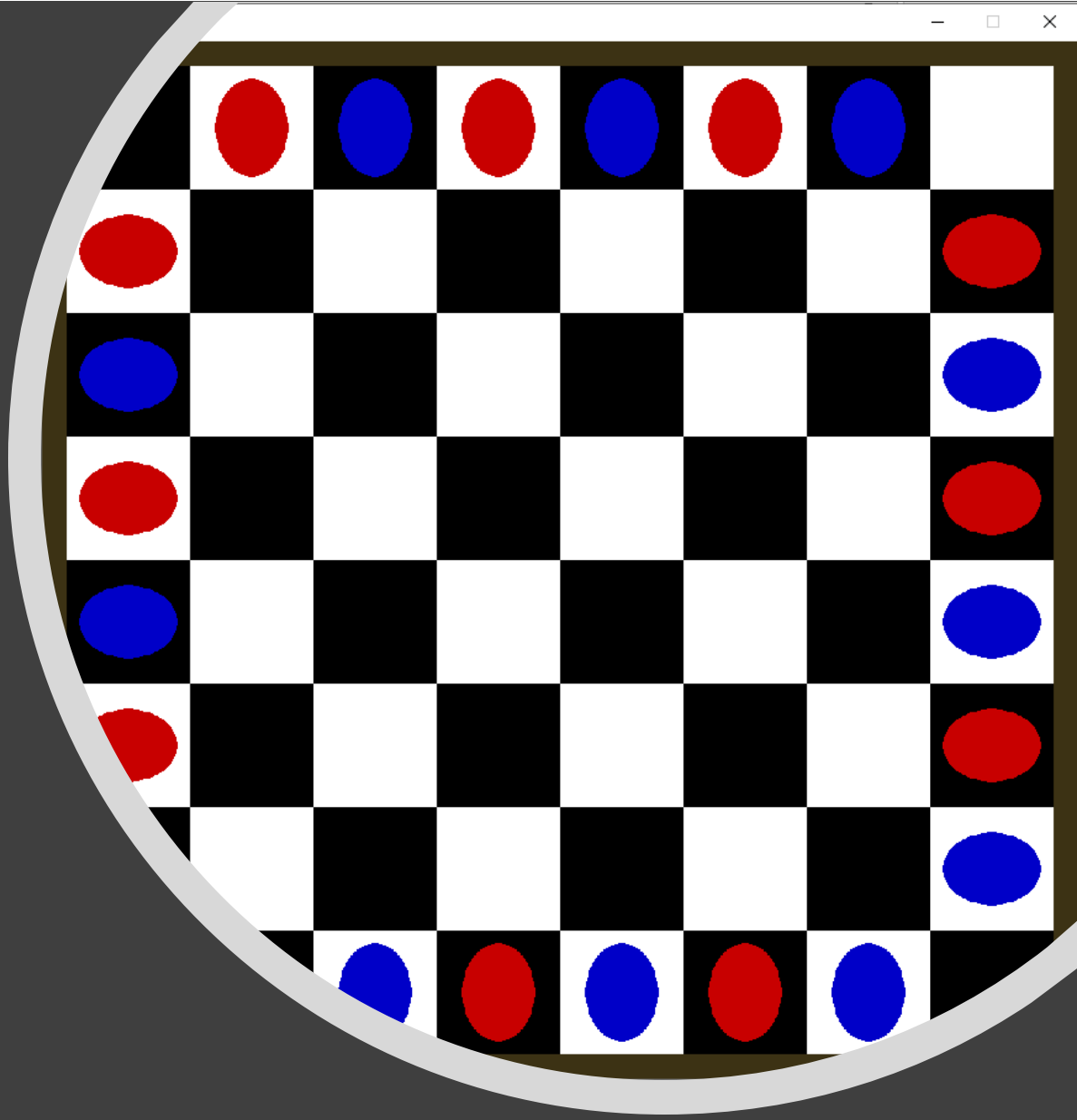
- **Teste Terminal:**

A condição para o jogo terminar é que haja apenas masters no tabuleiro, e neste caso vence o jogador com mais masters, ou um jogador não tenha mais movimentos válidos, e neste caso este jogador perde.

- **Função de Avaliação:**

A função de avaliação que implementamos faz a soma dos valores de todas peças de um jogador e subtrai pelas do oponente. Os minions valem 1 ponto e os masters valem 3.

Em um estado de fim de jogo a função avalia o valor como 0, 10e+5 ou –10e+5, dependendo se o estado representa empate, vitória ou derrota.



Implementação

- Utilizamos o algoritmo minimax com cortes alpha e beta com a adição de que a pesquisa não pode parar em um nó no qual possa haver capturas de peças. Utilizamos uma variante do algoritmo denominada negamax, que se aproveita da propriedade de que o jogo tem soma zero para simplificar a implementação.

- Durante a pesquisa o algoritmo separa as jogadas em dois grupos: as jogadas que envolvem capturas de peças e as que não envolvem. O primeiro grupo é verificado primeiro pois é mais provável encontrar jogadas de maior valor neste.

- Testamos nosso algoritmo com 5 níveis de profundidade. O jogo possui duas opções de tamanho de tabuleiro: 6x6 e 8x8. Executamos testes combinando diferentes níveis de profundidade e tamanho de tabuleiro.

Testes

Medimos o tempo médio tomado pelo algoritmo em cada jogada. Além do fato de que o tempo aumenta com a profundidade e o tamanho do tabuleiro percebemos também que costuma variar muito ao longo do jogo de acordo com o número de peças presentes, podendo ir de menos de 1 segundo a mais de 30 segundos em algumas jogadas com profundidade 4 ou 5.

Nível (profundidade)	Tempo 6x6	Desvio Padrão 6x6	Tempo 8x8	Desvio padrão 8x8
1	0.0062	0.007946663571098977	0.0229	0.0741
2	0.0270	0.0282	0.1355	0.1881
3	0.1206	0.1649	0.9454	0.9454
4	0.4678	0.6266	4.0997	7.2495
5	1.1239	1.8673	-	-

Testes

Contamos vitórias, derrotas e empates combinando os níveis de profundidade. Contamos também o número médio de jogadas em cada partida.

Vimos claramente que níveis superiores tendem a vencer os inferiores.

Estes são os resultados obtidos em um tabuleiro 6x6

Vermelho/Azul	1	2	3	4	5
1	Vermelho: 6 Azul: 6 Empate: 8 Jogadas: 63.25	Vermelho: 2 Azul: 14 Empate: 4 Jogadas: 55.5	Vermelho: 0 Azul: 17 Empate: 3 Jogadas: 52.4	Vermelho: 0 Azul: 20 Empate: 0 Jogadas: 58	Vermelho: 0 Azul:20 Empate:0 Jogadas: 65.4
2	Vermelho: 13 Azul: 4 Empate: 3 Jogadas: 53	Vermelho: 7 Azul:6 Empate:7 Jogadas: 56.6	Vermelho: 1 Azul: 16 Empate: 3 Jogadas: 50.1	Vermelho: 0 Azul:19 Empate:1 Jogadas: 41	Vermelho: 0 Azul: 20 Empate: 0 Jogadas: 47.5
3	Vermelho: 17 Azul:1 Empate:2 Jogadas: 66.5	Vermelho: 17 Azul:2 Empate:1 Jogadas: 51.6	Vermelho: 7 Azul:5 Empate:8 Jogadas: 50.5	Vermelho: 1 Azul:16 Empate:3 Jogadas: 51.6	Vermelho: 0 Azul:17 Empate:3 Jogadas: 54.7
4	Vermelho: 20 Azul:0 Empate:0 Jogadas: 57.8	Vermelho: 20 Azul:0 Empate:0 Jogadas: 43.2	Vermelho: 17 Azul:1 Empate:2 Jogadas: 50.2	Vermelho: 6 Azul:6 Empate:8 Jogadas:43.0	Vermelho: 0 Azul:18 Empate:2 Jogadas: 55.5
5	Vermelho: 20 Azul: 0 Empate: 0 Jogadas: 64.6	Vermelho: 20 Azul: 0 Empate: 0 Jogadas: 46.7	Vermelho: 18 Azul:0 Empate:2 Jogadas: 66.0	Vermelho: 17 Azul:1 Empate:2 Jogadas: 57.5	Vermelho: 6 Azul:8 Empate:6 Jogadas: 45.1

Testes

Estes são os resultados obtidos em um tabuleiro 8x8

	1	2	3	4	5
1	Vermelho: 6 Azul: 8 Empate: 6 Jogadas: 112.5	Vermelho: 0 Azul: 16 Empate: 4 Jogadas: 107.3	Vermelho: 0 Azul: 20 Empate: 0 Jogadas: 95.4	Vermelho: 0 Azul: 20 Empate: 0 Jogadas: 91.0	Vermelho: 0 Azul: 20 Empate: 0 Jogadas: 92.3
2	Vermelho: 17 Azul:2 Empate:1 Jogadas: 104.5	Vermelho: 6 Azul:10 Empate:4 Jogadas: 93.5	Vermelho:14 Azul:2 Empate:4 Jogadas: 78.2	Vermelho: 0 Azul:19 Empate:1 Jogadas: 71.5	Vermelho: 0 Azul:20 Empate:0 Jogadas: 77.5
3	Jogadas: 124.5 Vermelho: 14 Azul: 4 Empates: 2	Jogadas: 75.7 Vermelho: 17 Azul: 1 Empates: 2	Jogadas: 81.64 Vermelho: 1 Azul: 11 Empates: 6	Jogadas: 79.2 Vermelho: 2 Azul: 16 Empates: 2	Não foi testado
4	Jogadas: 83.7 Vermelho: 20 Azul: 0 Empates: 0	Jogadas: 69.3 Vermelho: 16 Azul: 0 Empates: 4	Jogadas: 70.2 Vermelho: 14 Azul: 1 Empates: 5	Jogadas: 83.0 Vermelho: 8 Azul: 10 Empates: 2	Não foi testado

Conclusões

Conseguimos desenvolver um programa que cumpre o todas as exigências do projeto.

Aplicamos o algoritmo minimax e cortes alpha-beta e verificamos que o resultado obtido é melhor que um jogador humano mediano.

Uma possível melhoria no algoritmo, para economizar tempo, seria guardar parte dos resultados da pesquisa em uma jogada para aplicar na jogada seguinte, pois assim não seria necessário estar sempre a reconstruir a ávore de pesquisa do início.

Observações sobre a Interface

No menu principal são exibidas as opções HxH, HxC, CxC (H para humano e C para Compputador).

Após selecionado o tipo de jogo deve se escolher os níveis de dificuldade dos bots, caso haja algum. As opções apresentadas correspondem a diferentes profundidades maiores.

Após escolher a dificuldade deve-se escolher o tamanho do tabuleiro.

Não recomendamos utilizar as profundidades maiores no tabuleiro 8x8, pois o processamento é muito demorado.

Além dos botões exibidos na interface é possível utililizar as teclas R e M para, respectivamente, reiniciar o jogo no mesmo modo de jogo, retornar ao menu principal.

Para um jogador humano é possível exibir uma dica de jogada pressionando uma tecla de número de 1 a 5. O número escolhido corresponde à profundidade da análise.

É possível desfazer jogadas utilizando tecla de seta para a esquerda.

Referências

- Chess Programming wiki – Alpha-Beta. Acessado em 28/03, disponível em https://www.chessprogramming.org/Alpha-Beta
- Chess Programming wiki – Negamax. Acessado em 28/03, disponível em https://www.chessprogramming.org/Negamax