U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Eximo

Inteligência Artificial

Afonso Carvalho Pereira de Sá

André Filipe da Silva Moutinho

Sérgio Bruno Rodrigues Dias

31 de março de 2020

Eximo

O Eximo é um jogo da família das damas, cujo objetivo é capturar todas as peças do oponente, saltando por cima delas, ou forçar uma situação de impasse em que o oponente não pode realizar nenhuma jogada.

Quando uma das peças atinge a última linha, esta removida do tabuleiro e o jogador recebe duas novas peças para serem colocadas de imediato na *drop zone* (duas primeiras linhas do lado do jogador, excetuando as extremidades).

Figura 1 - Tabuleiro no estado inicial do jogo

Movimentos

- **Mover** – Existem 3 movimentos possíveis: norte, noroeste e nordeste. Um jogador pode **mover** a sua peça para um quadrado livre adjacente ou **saltar** por cima de outra peça sua, situada à frente ou nas diagonais da frente da peça, desde que haja um quadrado livre na direção do movimento que vai realizar. Caso o jogador opte pela segunda, deverá continuar a movimentar a sua peça dessa maneira enquanto for possível.
- **Capturar** – Para capturar uma peça existem 5 movimentos possíveis: norte, noroeste, nordeste, este e oeste. Deste modo, o jogador pode **saltar** com uma peça por cima de uma peça adversária desde que haja uma casa vazia na direção do seu movimento, **capturando** assim a peça do oponente. Dentro da mesma jogada, o jogador deverá continuar a capturar enquanto for possível. É de notar que a captura, sendo possível, é **obrigatória**.

Estado do Jogo

O estado inicial é representado por um *array* bidimensional de inteiros de 8x8 elementos (representante do *board* 8x8), em que um 0 representa uma célula vazia, um 1 uma célula com uma peça branca e um 2 uma peça preta.

Função de Avaliação

A função de avaliação inicialmente considerada levaria sobretudo em conta o número de peças no tabuleiro, sendo que a avaliação seria tanto maior quanto maior for esse número. No entanto, com menor peso, seria também avaliada a posição das peças (sendo mais favorável quando se encontram mais perto da outra ponta do tabuleiro) e o número de movimentos possíveis (proporcional à avaliação). As diversas heurísticas que foram de facto implementadas encontram-se descritas mais à frente.

Operadores

Nomes	Pré-Condições	Efeitos	Custo
moveN	y<7 && board[x+(y+1)*8]=0	board[x+1+y*8]=1 && board[x+y*8]=0	1
moveNW	x<7 && y<7 && board[x+1+(y+1)*8]=0	board[x+1+(y+1)*8]=1 && board[x+y*8]=0	1
moveNE	x>0 && y<7 && board[x-1+(y+1)*8]=0	board[x-1+(y+1)*8]=1 && board[x+y*8]=0	1
jumpOverN	y<6 && board[x+(y+2)*8]=0 && board[x+(y+1)*8]=1	board[x+(y+2)*8]=1 && board[x+y*8]=0	1
jumpOverNW	y<6 && x<6 && board[x+2+(y+2)*8]=0 && board[x+1+(y+1)*8]=1	board[x+2+(y+2)*8]=1 && board[x+y*8]=0	1
jumpOverNE	y<6 && x>1 && board[x-2+(y+2)*8]=0 && board[x-1+(y+1)*8]=1	board[x-2+(y+2)*8]=1 && board[x+y*8]=0	1

Operadores (cont.)

Nomes	Pré-Condições	Efeitos	Custo
captureN	y<6 && board[x+(y+2)*8]=0 && board[x+(y+1)*8]=2	board[x+(y+2)*8]=1 && board[x+(y+1)*8]=0 && board[x+y*8]=0	1
captureNW	y<6 && x<6 && board[x+2+(y+2)*8]=0 && board[x+1+(y+1)*8]=2	board[x+2+(y+2)*8]=1 && board[x+1+(y+1)*8]=0 && board[x+y*8]=0	1
captureNE	y<6 && x>1 && board[x-2+(y+2)*8]=0 && board[x-1+(y+1)*8]=2	board[x-2+(y+2)*8]=1 && board[x-1+(y+1)*8]=0 && board[x+y*8]=0	1
captureW	x<6 && board[x+2+y*8]=0 && board[x+1+y*8]=2	board[x+2+y*8]=1 && board[x+1+y*8]=0 && board[x+y*8]=0	1
captureE	x>1 && board[x-2+y*8]=0 && board[x-1+y*8]=2	board[x-2+y*8]=1 && board[x-1+y*8]=0 && board[x+y*8]=0	1

Operadores (cont.)

Os operadores que são aplicados sobre os tabuleiros são efetivamente os descritos anteriormente. Porém, devido a poderem ser executados movimentos sequenciais (captura ou jumpOver), uma jogada é representada por uma lista de tabuleiros, que são o resultado da aplicação dos operadores de modo sequencial num turno apenas. Caso não exista nenhum movimento a ser executado além do inicial (portanto obrigatório), esta lista contém um único tabuleiro.

Algoritmos implementados

Tal como era o objetivo deste trabalho prático, implementou-se o Minimax como algoritmo de pesquisa de soluções. Foram realizadas otimizações para encurtar o tempo de pesquisa, nomeadamente cortes alpha/beta e ordenação previa dos nós, tendo-se recorrido para o efeito à função sort da biblioteca Collections do java (Merge Sort, com O(N*log(N))).

Heurísticas e funções de avaliação

A função de avaliação mais bem-sucedida a que chegamos resultou de uma vasta análise empírica do comportamento de inúmeras outras funções previamente implementadas. Foram-se anotando os resultados e observando as lacunas de cada uma, para que se pudesse chegar a um ótimo compromisso. De seguida encontram-se as heurísticas descritas de uma forma resumida:

Heurística	Variáveis a considerar
Random	Heurística aleatória
Basic	Número de peças do jogador
First	Número e posicionamento das peças do jogador
Improved	Posicionamento das próprias peças e número das peças do adversário e do próprio
Advanced	Igual a Improved, com o acréscimo de estratégias de defesa e ataque mais eficazes

Por fim, é de acrescentar que foi implementado um tempo máximo para obter uma jogada, para que o computador não levasse demasiado tempo a executar o algoritmo de pesquisa quando deparado com profundidades elevadas. Ao fim do tempo estipulado, a melhor jogada encontrada até ao momento é retornada. Estas heurísticas encontram-se detalhadas nos anexos.

Resultados

Tempo máximo de pesquisa por jogada

Tempo de execução

Profundidade		Tempo	Heurística	Tempo (5 ocorrências)	Número de jogadas	Vencedor
3	Jogador 1	3 segundos	Advanced	5/4/5/5/4 (segundos)	29/29/29/29/29	Jogador 1 (Advanced)
	Jogador 2		Improved		24/24/24/24/24	
	Jogador 1	1 segundo	Advanced	11/5/5/5/6 (segundos)	46/29/29/29/29	Jogador 1 (Advanced)
	Jogador 2		Improved		44/24/24/24/24	
5	Jogador 1	2 segundos	Advanced	1:47/1:24/1.29/1:15/1:04 (minutos : segundos)	42/42/34/38/32	Jogador 1 (Advanced)
	Jogador 2		Improved		35/37/32/29/23	
	Jogador 1	1 segundo	Advanced	40/40/53/44/35 (segundos)	34/35/42/34/32	Jogador 1 (Advanced)
	Jogador 2		Improved		25/29/37/31/27	
3	Jogador 1	5 segundos	Improved	15/15/16/15/14 (segundos)	63/63/63/63/63	Jogador 1 (Improved)
	Jogador 2		First		39/39/39/39/39	
	Jogador 1	5 segundos	Advanced	16/16/17/17/16 (segundos)	67/67/67/67/67	Jogador 1 (Advanced)
	Jogador 2		Basic		64/64/64/64/64	
3	Jogador 1	2 segundos	Basic	2:02/0:37/1:00/1:12/1:36 (minutos : segundos)	95/38/63/52/88	Jogador 2 (Improved)
	Jogador 2		Improved		131/46/74/73/96	

Resultados (cont.)

	Profundidade	Tempo	Heurística	Tempo (5 ocorrências)	Número de jogadas	Vencedor
Jogador 1	3	2 segundos	Advanced	15/14/15/15/13 (segundos)	60/57/63/63/56	Jogador 1 (Depth 3)
Jogador 2	1		Advanced		55/53/51/51/51	
Jogador 1	1	2 segundos	Improved	5/4/4/5/4 (segundos)	27/27/27/27/27	Jogador 2 (Basic, Depth 3)
Jogador 2	3		Basic		33/33/33/33/33	
Jogador 1	4	2 segundos	Advanced	1:15/1:29/1:36/1:14/0:54 (minutos : segundos)	226/293/318/226/206	Jogador 2 (Advanced)
Jogador 2	1		Basic		236/295/319/236/209	

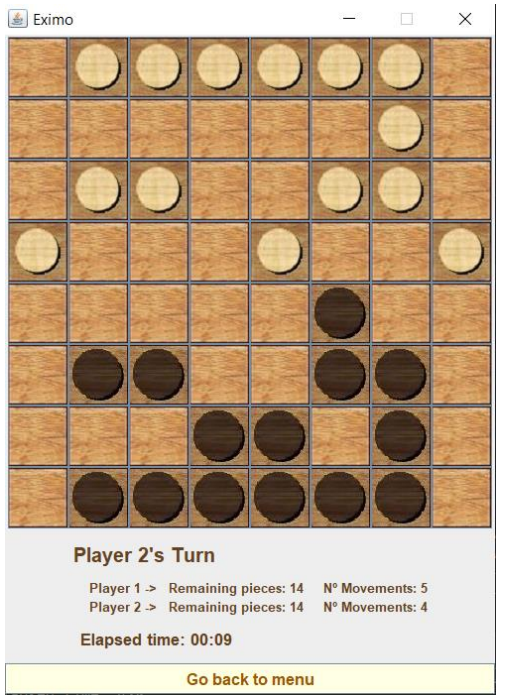
A tabela a seguir apresenta os resultados da aplicação do algoritmo Minimax sem cortes alfa beta:

	Profundidade	Tempo	Heurística	Tempo (5 ocorrências)	Número de jogadas	Vencedor
Jogador 1	3	5 segundos	Advanced	30/29/29/42/31 (segundos)	29/29/29/43/29	Jogador 1 (Advanced)
Jogador 2			Improved		24/24/24/41/24	
Jogador 1	3	5 segundos	Advanced	4:16/2:06/4:07/3:03/2:18 (minutos : segundos)	89/87/118/98/67	Jogador 1 (Advanced)
Jogador 2			Basic		79/77/110/90/61	

Resultados (cont.)

Fazendo uma análise breve e geral dos resultados, é possível verificar a diferença de tempo decorrido entre utilizar o algoritmo Minimax com e sem cortes alpha beta. É também de notar a eficácia da heurística Advanced, apenas perdendo quando utilizando uma profundidade mais baixa que a do adversário, o que nos leva a observer que de facto a profundidade tem um impacto particularmente grande no que toca à inteligência do computador.

Final de jogo – Advanced vs Basic s/cortes



Estado intermédio do jogo – Advanced vs Improved

Conclusões

A realização deste projeto foi bem-sucedida, tendo sido cumpridos todos os objetivos propostos. Conseguiu-se ainda implementar uma interface gráfica apropriada que facilita tanto a visualização como a jogabilidade. Os integrantes do grupo sentem-se satisfeitos com o trabalho realizado e destacam a evolução do seu conhecimento desde o início do semestre no que à área **Inteligência Artificial** diz respeito. Para tal, contribuiu largamente a aplicação de conceitos neste trabalho prático, que aumentou ainda o interesse pela área. Foi, portanto, um projeto prazeroso de executar, o que motivou um maior esforço e dedicação.

Por fim, espera-se que os conhecimentos adquiridos e aplicados possam vir a ser úteis ao longo da nosso percurso enquanto engenheiros informáticos e que sirvam de base à aprendizagem de conhecimentos futuros.

Referências

Os integrantes do grupo consideraram suficiente a informação presente no site disponibilizado pelo professor para a realização do jogo, tendo como base para a criação de heurísticas a análise empírica, dada a complexidade do jogo e a escassez de informação. Quanto ao software utilizado, este projeto foi desenvolvido no Eclipse, tendo a interface gráfica sido desenvolvida com recurso à biblioteca Swing de java.

Webgrafia

https://www.boardgamegeek.com/boardgame/137916/eximo

https://boardgamegeek.com/thread/925957/new-game-checkers-family-eximo

Anexos

Aqui encontram-se as heurísticas já referidas anteriormente mas explicadas com maior detalhe:

- **Random**: esta função retorna uma avaliação aleatória;
- **Basic**: esta função apenas tem em consideração o número de peças do próprio jogador;
- **First**: esta foi a primeira heurística que implementamos com o intuito de ser eficaz. Leva em consideração o critério anterior mas introduz um outro com menor peso (distribuição de estimadamente 70%-30%), consistindo na avaliação uniforme da distância das peças ao fim do tabuleiro.
- **Improved**: esta heurística, por sua vez, surge como um melhoramento da anterior, mantendo o peso dos critérios já enunciados mas levando também em conta o número de peças do oponente, procurando minimizá-lo. De referir que o ganho/conservação de peças do jogador e a captura de peças do adversário são valorizadas de igual forma.
- **Advanced**: esta foi, naturalmente, a heurística mais trabalhosa de se implementar, sendo a única capaz de vencer as restantes em várias situações e de forma consistente. Introduziu-se uma estratégia de defesa, que privilegia a permanência de uma certa quantidade de peças na linha inicial do tabuleiro, salvaguardando-as e dificultando ainda a multiplicação de peças por parte do oponente. Além disso, implementou-se uma estratégia de ataque cauteloso pelas laterais, aumento a chance de chegar ao fim do tabuleiro e consequentemente ganhar peças extra. Além disso, em certos cenários, esta estratégia assume um papel defensivo, evitando que o adversário chegue facilmente ao fim do tabuleiro (algo que se verificou na heuristica básica, que por vezes vencia outras mais avançadas). A distribuição de pesos foi afinada para otimizar e dar sentido à heurística, sendo contudo dificil de apurar em percentagem.