Implementação do jogo Mancala - Grupo 23

Utilizando Métodos de Pesquisa MiniMax com cortes Alpha-Beta em Linguagem java

António Cruz (up201603526)
FEUP, MIEIC
up201603526@fe.up.pt

Helena Montenegro (up201604184)
FEUP, MIEIC
up201604184@fe.up.pt

Juliana Marques (up201605568)
FEUP, MIEIC
up201605568@fe.up.pt

Resumo - Este artigo tem como objetivo introduzir as regras do jogo Mancala que foi implementado no segundo projeto da cadeira de Inteligência Artificial, tal como aspetos relativos ao desenvolvimento do algoritmo de Minimax com cortes Alfa-Beta. Concluiu-se que os cortes Alfa-Beta tornaram o algoritmo mais eficiente em termos de tempo médio de uma jogada e que ambas as heurísticas utilizadas se mostraram eficazes na resolução do problema.

Palavras-chave: Inteligência Artificial, Minimax, com cortes Alfa-Beta, Mancala.

I. Introdução

Este trabalho tem como objetivo o desenvolvimento de capacidades na área da Inteligência Artificial através da implementação do jogo com adversários, neste caso o Mancala, utilizando o algoritmo de pesquisa Minimax com cortes alfa-beta.

Foram desenvolvidos modos de jogo "humano-humano", "humano-computador" e "computador-computador", com diferentes níveis de dificuldade, heurísticas e com vertentes do algoritmo de Minimax: com e sem cortes alfa-beta.

Neste artigo, pretende-se analisar a implementação do trabalho, tal como os seus resultados. Primeiramente apresenta-se uma descrição do jogo selecionado, seguido da formulação do problema como um problema de pesquisa, depois apresentam-se trabalhos semelhantes previamente realizados e procede-se à explicação da implementação do jogo e à análise dos resultados obtidos. O artigo termina com uma pequena conclusão.

II. Descrição do Problema

O tabuleiro do jogo **Mancala** é composto por duas áreas de captura de peças localizadas nas extremidades do tabuleiro que pertencem a cada um dos jogadores, sendo tratadas por "mancala". Existem 12 áreas intermédias, as "casas". Para além disso, existem também as peças do jogo que circulam pelas casas e pelas mancalas.

Figura 1: exemplo de um tabuleiro de Mancala.

Inicialmente, cada casa possui quatro peças.

Na sua vez, o jogador começa por escolher uma casa com peças no seu lado do tabuleiro (uma das fileiras), colocando as peças nas casas subsequentes no sentido contrário ao dos ponteiros do relógio, uma a uma. Se neste processo uma peça calhar na mancala do jogador, é colocada uma peça na mesma, e se calhar na mancala do adversário, não se coloca peça. Quando a última peça da jogada cai na própria mancala do jogador, este pode repetir a jogada. Se a última peça calhar numa casa vazia do lado do jogador, este pode colocar na sua mancala essa peça e as peças presentes na casa oposta.

O jogo termina quando uma fileira fica vazia, sendo que o jogador que tem peças na sua fileira captura-as. Ganha o jogador que possua mais peças na sua mancala.

III. Formulação do Problema

A. Representação do Estado

O jogo é representado por uma matriz de inteiros 2x6, em que cada célula representa as casas do jogo e o seu valor representa o número de peças aí presente. Cada jogador contém a sua própria mancala, que será um inteiro representante do número de peças na mesma.

B. Estado Inicial

A matriz representante do tabuleiro do jogo inicia com todos os elementos com o valor 4.

C. Teste Objetivo

O estado final ocorre quando uma das linhas da matriz do tabuleiro tiver em todos os seus elementos o valor 0.

D. Operadores

O único operador possível é a seleção de uma célula da matriz representativa do tabuleiro.

1) Pré Condições:

A célula escolhida tem de pertencer à linha do jogador e o seu valor tem de ser superior a zero.

2) Efeitos:

O valor presente nessa linha será distribuído da seguinte forma: no sentido oposto ao dos ponteiros do relógio incrementa-se o valor de cada uma das peças que seguem a célula escolhida, subtraindo ao valor da célula escolhida. Se neste processo se chegar ao fim da linha correspondente ao jogador que efetuou a jogada, incrementa-se o valor da sua mancala, subtraindo também ao valor da célula escolhida e continua o processo pela linha do jogador adversário. Se o último valor incrementado tiver calhado na mancala do jogador, este repete a jogada. Se o último valor incrementado

tiver calhado numa casa previamente vazia na linha do jogador, essa célula e a célula oposta ficam vazias, sendo os seus valores incrementados na mancala do jogador.

3) Custo da Solução:

A solução será avaliada de acordo com a pontuação de cada jogador no final do jogo, o tempo que o algoritmo demora a alcançar uma solução para cada jogada e o número de jogadas necessárias para alcançar a vitória.

IV. Trabalho Relacionado

- https://github.com/naigutstein/Mancala - Este repositório contém uma implementação do jogo Mancala usando o algoritmo de MiniMax com cortes Alfa-Beta, programado em Python.

V. Implementação do jogo

O projeto foi desenvolvido na linguagem java. O projeto contém uma representação gráfica de fácil utilização que aparece quando se corre o ficheiro gui/MainMenu.java. Tem também uma representação em texto na consola que é acessível ao correr logic/Main.java.

Foram implementados três modos de jogo: PC vs PC, PC vs Humano e Humano vs Humano, onde o PC tem três níveis de dificuldade, definidas por diferentes graus de profundidade do algoritmo de Minimax.

Num modo de jogo que use o PC, são pedidos ao utilizador o seu nível de dificuldade, a heurística a utilizar para o cálculo do valor de um tabuleiro e se o algoritmo Minimax deve ou não conter cortes alfa-beta.

Em modos de jogo em que o utilizador, é lhe dada a possibilidade de pedir uma pista à aplicação, que executará o algoritmo Minimax com cortes alfa-beta no maior nível de dificuldade existente (oito níveis de profundidade).

O estado do tabuleiro está representado na classe gui/Board, onde constam as funções que implementam as regras do jogo (movement()), que verificam se o tabuleiro é final (is_final()), que atualizam as mancalas dos jogadores quando se chega ao fim do jogo (set_final()), que geram os sucessores do tabuleiro (generate_successors()), que calculam o valor do tabuleiro de acordo com a heurística do algoritmo Minimax escolhida (calculate_value()), que executam o algoritmo Minimax (get_best_board() e get_best_board_alpha_beta()) e que mostrem o tabuleiro (print()). A função do movimento, definida em movement(), recebe o valor correspondente à célula cujas peças serão movidas e as distribui no sentido contrário ao dos ponteiros do relógio, de acordo com as regras enunciadas na secção respetiva à descrição do jogo.

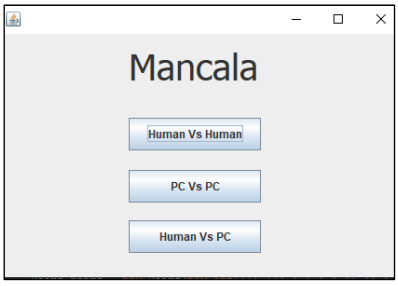

Figura 2: menu principal

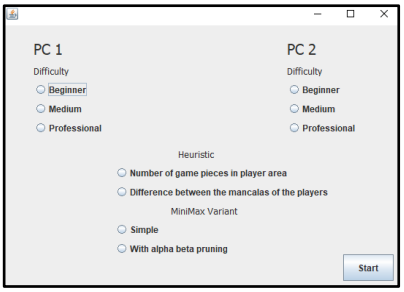

Figura 3: escolha da dificuldade, heurística e algoritmo.

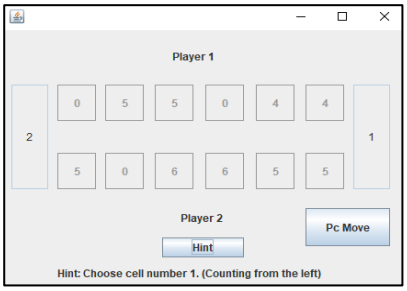

Figura 4: possível cenário de jogo.

VI. Algoritmo Minimax

Foi implementado o algoritmo de pesquisa com adversários Minimax. Na aplicação são apresentadas versões do algoritmo com e sem cortes alfa-beta, e com duas heurísticas diferentes: o número de peças do lado do jogador e a diferença entre o número de peças na mancala do jogador.

A implementação deste algoritmo passa por duas fases: a geração dos tabuleiros descendentes do tabuleiro atual, até uma profundidade máxima definida no tabuleiro, e a procura do melhor valor entre os tabuleiros descendentes.

Na classe logic/Board, encontra-se a função recursiva get_best_board(), que procura entre os descendentes do tabuleiro o melhor tabuleiro, de acordo com o algoritmo Minimax sem cortes alfa-beta. Em primeiro lugar a função verifica se o tabuleiro se encontra no último nível de profundidade, e, se esse for o caso, é calculado o seu valor, de acordo com a heurística escolhida. Verifica-se também se o tabuleiro é final. Nestes dois últimos casos, não se verificam os sucessores do tabuleiro. Caso o tabuleiro não seja final nem esteja no último nível de profundidade da pesquisa, chama-se a função get_best_board() para cada um dos sucessores, guardando o melhor valor encontrado (o maior valor quando o tabuleiro estiver no estado de MAX, ou seja, o jogador do tabuleiro corresponde ao que irá fazer a jogada no tabuleiro inicial, ou o menor valor quando o tabuleiro estiver no estado MIN).

Para o algoritmo Minimax com cortes alfa-beta, encontrado na função get_best_board_alpha_beta(), a função é semelhante à anterior, com a diferença de se guardarem os valores alfa e beta, que são passados como argumento da função. Alfa corresponde ao melhor valor encontrado quando o estado do tabuleiro é MAX e beta corresponde ao melhor valor encontrado quando o estado do tabuleiro é MIN. Os seus valores iniciais são Integer.MIN_VALUE e Integer.MAX_VALUE, respetivamente. Quando se

verificam os valores dos sucessores do tabuleiro, guardam-se os melhores valores nas variáveis alfa e beta, consoante o estado do tabuleiro, e, se alfa se tornar maior que beta, não se verificam mais sucessores, sendo estes cortados.

Os cortes alfa-beta não alteram o resultado do algoritmo Minimax, apenas o torna mais eficiente.

VII. Experiências e Resultados

Nesta secção serão analisados os resultados obtidos em termos de pontuação, número de jogadas para chegar ao final do jogo e média do tempo de jogada, em segundos. Para tal recolhemos os dados resultantes do modo de jogo PC vs PC.

Os níveis de dificuldade indicados nas tabelas que se seguem apresentam a seguinte correspondência: 1 - fácil; 2 - médio; 3 - difícil.

Nas tabelas que dizem respeito à pontuação e ao número de movimentos realizados pelos jogadores, os algoritmos Minimax com e sem cortes alfa-beta mostraram resultados iguais, até porque os cortes alfa-beta não alteram o resultado apenas tornam o algoritmo mais eficiente, pelo que apenas serão comparadas as heurísticas utilizadas.

Pontuação dos jogadores usando como heurística o número de peças na área do jogador:

Dificuldade	Pontuação da mancala do jogador 1	Pontuação da mancala do jogador 2
1 vs 1	44	4
1 vs 2	19	29
2 vs 2	21	27
1 vs 3	22	26
3 vs 3	30	18
2 vs 3	20	28

Pontuação dos jogadores usando como heurística a diferença entre as peças nas mancalas:

Dificuldade	Pontuação da mancala do jogador 1	Pontuação da mancala do jogador 2
1 vs 1	19	29
1 vs 2	15	33
2 vs 2	34	14
1 vs 3	21	27
3 vs 3	41	7
2 vs 3	19	29

Nas tabelas relativas às pontuações obtidas pelos computadores, verifica-se que para ambas as heurísticas, o computador com o nível de dificuldade mais elevado vence sempre ao computador com o nível de dificuldade mais baixo.

Comparando ambas as heurísticas, verifica-se também que na segunda heurística a diferença entre a maior pontuação e a menor pontuação é superior à encontrada na primeira heurística, no que toca a jogos onde ambos os PCs têm níveis de dificuldade diferentes. Este facto, por si só, leva a pensar que a segunda heurística é melhor, apesar de ambas se mostrarem eficazes.

Número de movimentos dos jogadores usando como heurística o número de peças na área do jogador:

Dificuldade	Número de movimentos do jogador 1	Número de movimentos do jogador 2
1 vs 1	5	5
1 vs 2	10	9
2 vs 2	24	23
1 vs 3	8	8
3 vs 3	28	20
2 vs 3	22	18

Número de movimentos dos jogadores usando como heurística a diferença entre as peças nas mancalas:

Dificuldade	Número de movimentos do jogador 1	Número de movimentos do jogador 2
1 vs 1	11	10
1 vs 2	12	9
2 vs 2	26	11
1 vs 3	14	11
3 vs 3	17	9
2 vs 3	17	13

Através da análise das tabelas relativas ao número de movimentos realizados verifica-se que, a segunda heurística necessita de mais movimentos para alcançar uma solução quando os PCs se encontram em níveis baixos de dificuldade (fácil e médio), no entanto, com PCs com níveis mais elevados, já apresenta melhores resultados com menos movimentos (2 vs 3 e 3 vs 3). Isso poderá significar que a segunda heurística é mais eficiente quando usada em conjunto com níveis de profundidade mais elevados no algoritmo Minimax, e que quando se usam níveis mais baixos, a primeira heurística se torna mais apropriada.

Tempo médio, em segundos, de uma jogada usando cortes alfa-beta e com heurística relativa ao número de peças na área do jogador:

Dificuldade	Tempo total do jogador 1	Tempo total do jogador 2
1 vs 1	0.000	0.000
1 vs 2	0.000	0.000
2 vs 2	0.001	0.000
1 vs 3	0.000	0.269
3 vs 3	0.121	0.026
2 vs 3	0.000	0.139

Tempo médio, em segundos, de uma jogada sem cortes alfa-beta e com heurística relativa ao número de peças na área do jogador:

Dificuldade	Tempo total do jogador 1	Tempo total do jogador 2
1 vs 1	0.000	0.000
1 vs 2	0.000	0.001
2 vs 2	0.001	0.000
1 vs 3	0.000	0.307
3 vs 3	0.143	0.090
2 vs 3	0.000	0.126

Ao analisar as tabelas relativas ao tempo médio, em segundos, necessário para a realização de uma jogada, para a primeira heurística, verifica-se que os tempos totais para o algoritmo que utiliza cortes alfa-beta é inferior aos tempos totais para o algoritmo sem estes.

Tempo médio, em segundos, de uma jogada usando cortes alfa-beta e com heurística relativa à diferença entre as peças nas mancalas:

Dificuldade	Tempo total do jogador 1	Tempo total do jogador 2
1 vs 1	0.000	0.000
1 vs 2	0.000	0.000
2 vs 2	0.000	0.000
1 vs 3	0.000	0.181
3 vs 3	0.140	0.099
2 vs 3	0.000	0.195

Tempo médio, em segundos, de uma jogada sem cortes alfa-beta e com heurística relativa à diferença entre as peças nas mancalas:

Dificuldade	Tempo total do jogador 1	Tempo total do jogador 2
1 vs 1	0.000	0.000
1 vs 2	0.000	0.002
2 vs 2	0.000	0.002
1 vs 3	0.000	0.215
3 vs 3	0.202	0.074
2 vs 3	0.000	0.233

Ao analisar as tabelas relativas ao tempo médio, em segundos, necessário para a realização de uma jogada, para a segunda heurística, verifica-se que os tempos totais para o algoritmo que utiliza cortes alfa-beta são inferiores aos tempos totais para o algoritmo sem estes.

Deste modo conclui-se que os cortes alfa-beta melhoram o algoritmo Minimax em termos de eficiência, visto que lhe permitem verificar menos nós sem alterar o seu resultado.

Ao comparar ambas as heurísticas em termos de tempo, a primeira heurística utilizada mostra ser ligeiramente mais rápida do que a segunda, no entanto as diferenças não são significativas, sendo ambas eficientes.

VIII. Conclusões e Perspetivas de Desenvolvimento

A partir da análise dos resultados de utilização do algoritmo Minimax com e sem cortes alfa-beta e com diferentes heurísticas para a resolução do jogo Mancala, conclui-se que os cortes alfa-beta não alteram o resultado do algoritmo, apenas o tornam mais eficiente. Ambas as heurísticas utilizadas, relativas ao número de peças na área do jogador e à diferença entre o número de peças nas mancalas dos jogadores, mostraram resultados positivos, sendo que com ambas, computadores com níveis de dificuldade superior venceram sempre computadores com níveis de dificuldade inferior.