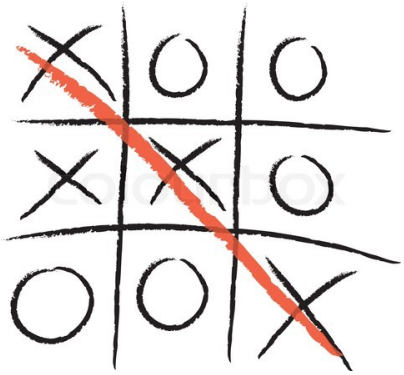

Autores:
Marcus Santos, 29764
Ricardo Fusco, 29263

Professor:
Irene Pimenta Rodrigues

May 4, 2014

Índice

Lista de Figuras

1 Minimax

2 Corte α-β

3 Jogo do Galo

3.1 Questões

3.1.1 Escolha uma estrutura de dados para representar os estado do jogo.

A estrutura utilizada para representar o estado do jogo é representado por um tuplo com uma lista de posições do tabuleiro, onde cada posição contém uma referência(um valor não instânciado), um caractere "x" ou "o" que indica uma jogada e a última peça que foi jogada.A figura 1 ilustra a definição do estado inicial.

```
%(lista com pos, ultima peca jogada)
estado_inicial(([(p(1,1), _),(p(1,2), _),(p(1,3), _),
                 (p(2,1), _),(p(2,2), _),(p(2,3), _),
                 (p(3,1), _),(p(3,2), _),(p(3,3), _)],_)).
```

Figure 1: Estado inicial - Jogo do Galo

3.1.2 Defina o predicado terminal(estado) que sucede quando o estado é terminal.

Um estado é terminal se há uma linha, coluna ou diagonal completa, seja ela com x's ou com o's, ou no ultimo dos casos quando estão todas as posições preenchidas e não há ganhadores(empate).

O predicado terminal/1 verifica a ocorrência de cada um desses casos e se qualquer um for verdade então o resultado é verdade.

```
terminal(E):-
    linhas(E);colunas(E);diagonais(E);empate(E).
```

Figure 2: Predicado terminal - Jogo do Galo

3.1.3 Defina uma função de utilidade que para um estado terminal que deve retornar o valor do estado (ex: -1 perde, 0 empata, 1 ganha).

A função de utilidade verifica a profundidade na árvore de pesquisa e os casos em que o estado é terminal, com a excepção do empate. Os valores devolvidos pela mesma podem ser 1, 0 ou -1 sendo que 0 representa empate, 1 ganha e -1 perde.

```
valor(E, 1, P):- X is P mod 2, X=0 ,linhas(E);colunas(E);diagonais(E),!.
valor(E, 0, _):- empate(E),!.
valor(E, -1, P):-linhas(E);colunas(E);diagonais(E).
```

Figure 3: Função de utilidade - Jogo do Galo

3.1.4 Use a implementação da pesquisa minimax dada na aula prática para escolher a melhor jogada num estado.

Com o algoritmo minmax as jogadas são sempre óptimas, logo, se jogarmos contra o computador, no melhor dos casos conseguimos um empate.

3.1.5 Implemente a pesquisa Alfa-Beta e compare os resultados (tempo e espaço).

Através dos resultados observados, podemos concluir que o minimax, apesar de demorar mais tempo a efectuar as jogadas, faz sempre uma jogada óptima, enquanto que o corte α-β não faz a melhor jogada óptima mas leva menos tempo a efectuar a mesma.

Foram realizadas 10 observações de tempo e nº de nós para jogadas com 1, 3 e 5 posições preenchidas. A média dos resultados obtidos pode ser observada através da tabela 1 para o algoritmo minimax e na tabela 2 para o corte α-β.

nº de posições preenchidas	**Tempo(ms)**	**nº nós**
1	4356,6	59 704
2	83,6	1109,2
3	6,2	50,8

Table 1: Resultados observados para o algoritmo minimax

nº de posições preenchidas	**Tempo(ms)**	**nº nós**
1	29,2	337
2	5	18,3
3	2,8	9,4

Table 2: Resultados observados para o corte α-β

Tal com se pode observar é notável a redução do tempo e nº de nós necessários para efectuar uma jogada. Apesar destes resultados não conseguimos colocar o alfabeta a fazer sempre a melhor jogada para o jogo do galo, por oposição ao jogo do 4 em linha onde com o alfabeta faz sempre as melhores jogadas.
No caso do jogo do galo o jogador irá jogar contra o minimax pois este demora no máximo à volta de 4 segundos e meio a fazer a primeira jogada, o que não é muito visto que nas seguintes irá ser quase instantâneo, e tendo em conta que faz a melhor jogada sempre não faria sentido colocar o jogador a jogar contra o alfabeta.

3.1.6 Defina uma função de avaliação que estime o valor de cada estado do jogo use os dois algoritmos anteriores com corte em profundidade e compare os resultados (tempo e espaço).

A função de avaliação feita para este jogo foi fazer o cálculo do número de 1 peça isolada somando ao número de 2 peças juntas, fazer a mesma soma para o oponente e subtrair um valor ao outro. Quando se soma o numero de 2 peças dá-se um peso extra a este numero multiplicando-o por 2.

```
% avalia(Estado, Tipo_peca, Avaliacao)
% dados um estado, um tipo de peca, returna em C o valor da avaliacao
func_aval((E,J), Val):-
        inverteJog(J, J2),
        aval(E,J2,Val).

aval(E, J, Val):-
        find_all_1peca(E, J, V1),
        find_all_2pecas(E, J, V2),
        Val1 is V1+(2*V2),
        inverteJog(J, J2),
        find_all_1peca(E, J2, V3),
        find_all_2pecas(E, J2, V4),
        Val2 is V3+(2*V4),
        Val is Val1-Val2.
```

Figure 4: Função de avaliação

3.1.7 Implemente um agente inteligente que joga o jogo que escolheu usando a pesquisa definida na alínea anterior.

Implementámos um agente inteligente que joga contra o jogador num ciclo até que se verifique o terminal.

```
ciclo_jogada(_,(E,J)):- (linhas(E);colunas(E);diagonais(E)), print_(E),write('Vencedor: '),write(J),!.
ciclo_jogada(_,(E,_)):- empate(E), print_(E),write('Empate!'),nl,!.

ciclo_jogada('c',(E,J)):-
        print_(E),
        nl,statistics(real_time,[Ti,_]),
        minimax_decidir((E,J),Op),
        statistics(real_time,[Tf,_]), T is Tf-Ti,
        nl,
        write('Tempo: '(T)),
        nl,
        n(N),
        write('Numero de Nós: '(N)),
        initInc,
        nl,
        write(Op),
        nl,
        nl,
        op1((E,J),Op,Es),
        ciclo_jogada('j',Es).

ciclo_jogada('j',(E,J)):-
        print_(E),
        nl,
        write('Escreva a linha da posicao onde deseja jogar: '),
        read(X),
        write('Escreva a coluna da posicao onde deseja jogar: '),
        read(Y),
        inverteJog(J,J1),
        op1((E,J),insere(p(X,Y),J1),Es),
        ciclo_jogada('c',Es).
```

Figure 5: Agente inteligente

3.1.8 Apresente uma tabela com o número de nós expandidos para diferentes estados do jogo (10 no mínimo) com os vários algoritmos.

4 4 em Linha

4.1 Questões

4.1.1 Escolha uma estrutura de dados para representar os estado do jogo.

À semelhança do jogo do galo, o estado inicial do 4 em linha é representado por um tuplo composto por uma lista de posições, com uma diferença que é o tamanho do tabuleiro é de 6x7, e a última peça jogada.

```
%(lista com pos., ultima peca jogada)
estado_inicial(([(p(1,1), _),(p(1,2), _),(p(1,3), _),(p(1,4), _),(p(1,5), _),(p(1,6), _),(p(1,7), _),
                 (p(2,1), _),(p(2,2), _),(p(2,3), _),(p(2,4), _),(p(2,5), _),(p(2,6), _),(p(2,7), _),
                 (p(3,1), _),(p(3,2), _),(p(3,3), _),(p(3,4), _),(p(3,5), _),(p(3,6), _),(p(3,7), _),
                 (p(4,1), _),(p(4,2), _),(p(4,3), _),(p(4,4), _),(p(4,5), _),(p(4,6), _),(p(4,7), _),
                 (p(5,1), _),(p(5,2), _),(p(5,3), _),(p(5,4), _),(p(5,5), _),(p(5,6), _),(p(5,7), _),
                 (p(6,1), _),(p(6,2), _),(p(6,3), _),(p(6,4), _),(p(6,5), _),(p(6,6), _),(p(6,7), _)], _)).
```

Figure 6: Estado inicial - 4 em linha

4.1.2 Defina o predicado terminal(estado) que sucede quando o estado é terminal.

O predicado terminal assenta exactamente no mesmo principio, vê as linhas, colunas, diagonais e de seguida verifica o empate

4.1.3 Defina uma função de utilidade que para um estado terminal que deve retornar o valor do estado.

A função de utilidade que devolve o valor do estado é exactamente igual à do jogo do galo com uma pequena diferença apenas, dado o vencedor, se este for "o" devolve 100 se for "x" devolve -100, em caso de empate devolve 0. Foi feita esta alteração no jogo do 4 em linha pois após realizados uns testes com os valores -1, e 1 e depois com -100 e 100 chegamos à conclusão que com os valores 100 e -100 faz sempre a melhor jogada (com profundidade limitada a 20)

4.1.4 Indique e descreva um algoritmo para escolher a melhor jogada num estado.

O algoritmo que escolhe a melhor jogada num estado seria o minimax, mas devido ao problema que tem no caso do 4 em linha, este demora demasiado tempo a fazer uma jogada no início sem qualquer tipo de corte pois as hipóteses são (7*6)!, pelo que o mais indicado para poder jogar em tempo real seria o alfabeta, e após alguns testes verificamos que com o alfabeta, além de ser rápido nas suas jogadas (demora a volta de 1 segundo a fazer a primeira jogada com profundidade limitada a 20), ainda executa sempre a melhor jogada o que é perfeito para o jogo do 4 em linha.
Para correr o programa basta consultar o ficheiro main.pl e executar o predicado

main/0.

Na figura seguinte pode-se observar o menu principal do programa onde se pode escolher qual o jogo que quer jogar.

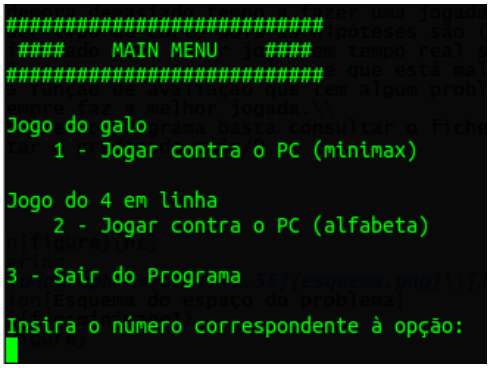

Figure 7: Main menu