Departamento de Informática

Inteligência Artificial

Ouri

3º Trabalho

(2010/2011)

Trabalho Realizado por:
Cindy Silva 21565
Nuno Croino 22802

Introdução:

O objectivo deste trabalho consiste em implementar um jogo de dois jogadores. O Ouri foi o jogo escolhido e consiste em passar feijões, de uns buracos para outros, o objectivo é conseguir encher o pot com mais de 24 feijões.

Após a implementação do jogo, pretende-se determinar a melhor jogada com os algoritmos min-max e com o alfa-beta.

Ouri:

Movimentos:
O Ouri consiste em ter um tabuleiro com 48 sementes. O tabuleiro contem 6 buracos para cada jogador, ou seja 12 buracos no total, que no inicio do jogo cada buraco tem 4 sementes. O jogador irá escolher 1 buraco e pegar em todas as sementes desse buraco e ira distribuir uma a uma semente nos buracos seguintes no sentido anti-horario, incluindo nos buracos do adversário.

Recolha:
O jogador recolhe as sementes para o seu pot caso a última semente caia em um buraco que contenha no total 2 ou 3 sementes e verificará nas imediatamente anteriores se estas também têm 2 ou 3 sementes e irá colocá-las no pot, termina quando encontra a primeira que não tem 2 ou 3 sementes.

Terminar:
O jogo termina quando um dos jogadores tem mais de 24 sementes no seu pot, ou quando não há mais possibilidades de o jogador efectuar a sua jogada. Ganha o jogador que tem mais de 24 sementes.

Desenvolvimento:

Estado Inicial:

```
estado_inicial((j1,([4,4,4,4,4,4,4,4,4,4,4,4],0,0))).
```

O estado inicial é composto por 1 tuplo que contem o jogador que irá jogar, a lista dos buracos com o número de sementes e os pot's dos jogadores 1 e 2. O primeiro jogador a jogar é o jogador 1, inicia-se com 4 sementes cada buraco e os pot's estão a 0.

Terminal:

```
terminal((_,([0,0,0,0,0,0,0,0,0,0,0,0],_V1,_V2))).
terminal((_,(_,V1,_))):- V1>24.
terminal((_,(_,_,V2))):- V2>24.
terminal((J,(A,B,C))):- verificaJogada(J,(A,B,C,_B1,_C1)).
```

Os terminais do jogo Ouri são quando um jogador tem mais que 24 sementes, ou não existe qualquer semente a jogo ou quando o jogador não consegue efectuar qualquer jogada.

Função de utilidade:

```
valor((J,(A,B,C)),1,_P):- verificaJogada(J,(A,B,C,B1,C1)), B1>C1.
valor((J,(A,B,C)),-1,_P):- verificaJogada(J,(A,B,C,B1,C1)), B1<C1.
valor((J,(A,B,C)),0,_P):- verificaJogada(J,(A,B,C,B1,C1)), B1=C1.
valor((J,(T,V1,V2)),1,_P):-terminal((J,(T,V1,V2))), V1>V2.
valor((J,(T,V1,V2)),-1,_P):-terminal((J,(T,V1,V2))), V2>V1.
valor((J,(T,V1,V2)),0,_P):-terminal((J,(T,V1,V2))), V1=V2.
```

A função utilidade sucede quando se encontra em estado terminal, e retorna 1 se o primeiro jogador ganha, -1 se o segundo jogador ganha e 0 se existir empate.

Min-Max

O algoritmo Min-Max tem como objectivo escolher a jogada com maior min-max. Para isso é necessário expandir a árvore com todas as possibilidades e calcular o valor do min-max. O algoritmo utilizado foi o fornecido pela docente da disciplina nas aulas práticas. Este algoritmo quando o número de jogadas possíveis é muito grande, o min-max tem que guardar em memória muitos nós, e portanto para poder testar este algoritmo convém que o estado inicial se encontre em um estado quase terminal, senão o computador não contem memória suficiente para guardar toda a informação necessária e cria um Stack Overflow.

<u>Alguns Estados Possíveis:</u>

```
estado_inicial((j1,([0,0,0,0,1,0,0,0,0,0,0,0],24,23))).
estado_inicial((j1,([0,0,0,0,0,2,0,0,0,0,0,0],23,23))).
estado_inicial((j2,([0,0,0,0,0,0,0,0,0,0,1,0],23,24))).
```

Para poder então testar o problema podemos ter uma função avaliação que irá de certo modo limitar o número de nós expandidos, e assim conseguir terminar o jogo.

Alfa-Beta

O algoritmo Alfa-Beta baseia-se no algoritmo Min-Max, mas à medida que vai verificando os nós este algoritmo não necessita expandir mais nós, se o nó que está a verificar é uma pior jogada que o anterior, e assim sendo os movimentos seguintes não necessitam ser analisados.

Alguns Exemplos de estados e os respectivos números de nós visitados

1. Neste exemplo o jogo irá pegar no operador 6, ou seja, vai pegar no elemento que se encontra na posição 6 pertencente ao jogador 1, e irá visitar num total 9 nós:

estado_inicial((j1,([0,0,0,0,0,2,0,0,0,0,0,0],23,23))).
operador: 6
nos visitados: 9

2. Neste exemplo o jogo irá pegar no operador 3, ou seja, vai pegar no elemento que se encontra na posição 3 pertencente ao jogador 1, e irá visitar num total 2 nós:

estado_inicial((j1,([0,0,1,0,0,0,0,0,0,0,0,0],23,23))).
operador: 3
nos visitados: 2

3. Neste exemplo o jogo irá terminar, porque o jogador 2 não tem qualquer semente para jogar e como tal não irá visitar qualquer nó:

estado_inicial((j2,([0,0,1,0,0,0,0,0,0,0,0,0],23,23))).
operador: terminou
nos visitados: 0

4. Neste exemplo o jogo irá pegar no operador 11, ou seja, vai pegar no elemento que se encontra na posição 11 pertencente ao jogador 2, e irá visitar num total 11 nós:

estado_inicial((j2,([0,0,0,0,0,0,0,0,0,0,2,0],23,23))).
operador: 11
nos visitados: 11

5. Neste exemplo o jogo irá pegar no operador 10, ou seja, vai pegar no elemento que se encontra na posição 10 pertencente ao jogador 2, e irá visitar num total 73 nós:

estado_inicial((j2,([0,0,0,0,0,0,0,0,0,3,0,0],23,23))).
operador: 10
nos visitados: 73

6. Neste exemplo o jogo irá pegar no operador 10, ou seja, vai pegar no elemento que se encontra na posição 10 pertencente ao jogador 2, e irá visitar num total 185 nós:

estado_inicial((j2,([1,0,0,0,0,0,0,0,0,3,0,0],23,23))).
operador: 10
nos visitados: 185

7. Neste exemplo o jogo irá terminar, porque o jogador 1 não tem qualquer semente para jogar e como tal não irá visitar qualquer nó:

estado_inicial((j1,([0,0,0,0,0,0,0,0,0,4,0,0],23,23))).
operador: terminou
nos visitados: 0

8. Neste exemplo o jogo irá pegar no operador 10, ou seja, vai pegar no elemento que se encontra na posição 10 pertencente ao jogador 2, e irá visitar num total 73 nós:

estado_inicial((j2,([0,0,0,0,0,0,0,0,0,3,0,0],23,24))).
operador: 10
nos visitados: 73

9. Neste exemplo o jogo irá terminar, porque o jogador 2 já tem no seu pot 25 sementes e como tal, já ganhou:

estado_inicial((j2,([1,0,0,0,0,0,0,0,0,3,0,0],23,25))).
operador: terminou
nos visitados: 0

10. Neste exemplo o jogo irá pegar no operador 4, ou seja, vai pegar no elemento que se encontra na posição 4 pertencente ao jogador 1, e irá visitar num total 2 nós:

estado_inicial((j1,([0,0,0,2,0,0,0,0,0,0,0,0],23,23))).
operador: 4
nos visitados: 2

Conclusão:

Este projecto esteve aquém das expectativas, não nos foi possível implementar o algoritmo de alfa-beta, a função de avaliação e ainda o agente de jogo. Devido a dificuldades a conseguir colocar o Min-Max a funcionar com o jogo do Ouri, perdeu-se muito tempo a tentar colocar o jogo a funcionar.

Para testar tem basta carregar o ficheiro minmax.pl no Prolog e depois chamar a função g(ouri).