Inteligência Artificial

Relatório do Trabalho Prático

Autores:
Pedro Nunes, 31240
Tiago Martinho, 35735

Docente:
Paulo Quaresma

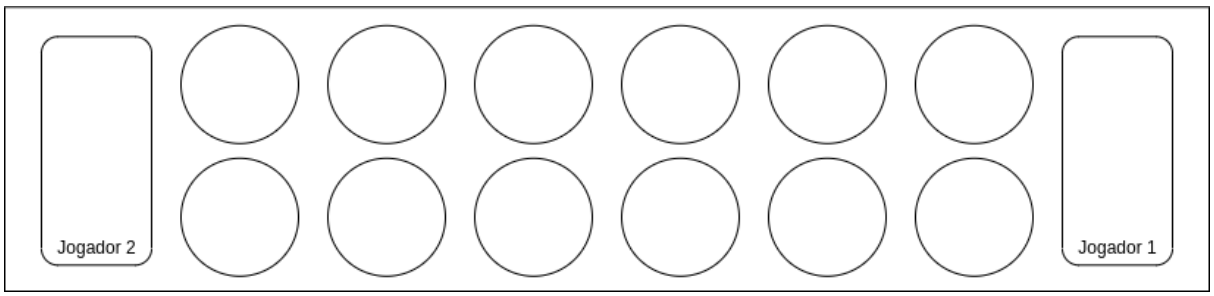
Junho de 2020

Índice

1 Introdução

O jogo "Ouri" é um **jogo de tabuleiro** entre dois jogadores em que temos um tabuleiro constituído por duas filas de 6 "casas"e duas "casas"adicionais, onde cada jogador armazena as peças que vai **capturando**. A primeira evidência conhecida do jogo é um fragmento de um tabuleiro de cerâmica e diversos cortes de rocha encontrados na Eritreia, estas são datadas entre os séculos VI e VII.

Este trabalho aborda a implementação de **dois algoritmos** e é implementado também um **GUI** do tabuleiro (imagem em baixo) de forma a ser possível observar as jogadas realizadas.



Instalar e Executar

- Para instalar os módulos do *python* pode-se correr:

```
sh install.sh
```

- Para correr o programa deve-se correr:

```
python3 main.py [Args]
```

- Argumentos[Args] obrigatórios:
  - alfabeta ou minimax (Algoritmo a usar).
  - -p ou -s (Se o agente é o primeiro a jogar ou o segundo).
  - 1, 2, ou 3 (Tempo de processamento: 1 - 5s, 2 - 15s, 3 - 30s).
- Argumentos extra:
  - true ou false (Mostrar tempo de processamento).
  - gui ou no-gui (Usar GUI).
- Exemplo:

```
python3 main.py alfabeta -p 1 true gui
```

Para correr a versão escolhida para o "I Torneio de Ouri de IA@LEI@UÉ":

```
python3 main.py alfabeta [Args]

Arg 1:
-p -> Primeiro
-s -> Segundo

Arg 2:
1 -> 5s
2 -> 15s
3 -> 30s
```

2 Desenvolvimento

2.1 Escolha das linguagens

Decidiu-se utilizar ***Prolog*** e ***Python*** de modo a tirar partido do melhor de cada linguagem. O *Prolog* foi uma escolha óbvia visto que os algoritmos utilizados são **recursivos** e este tem vantagem, uma vez que utiliza uma *heap* em vez de *stack*. O *Python* por sua vez foi escolhido pela sua simplicidade, pela **existência de módulos** que facilitam a implementação e pela componente de listas que possui. Para fazer a **conecção** entre os dois usou-se o módulo *"pyswip"*.

2.2 Escolha dos algoritmos

Os dois algoritmos que encaixam melhor neste tipo de problema são *MiniMax* e *MiniMax* com cortes *AlphaBeta*, isto porque pretendia-se encontrar a **melhor jogada** possível dado um tabuleiro. Logo a escolha foi direta uma vez que pretendemos calcular as **jogadas do adversário** de forma a escolher a melhor para nós.

3 Estados e Operações

- **Estado Inicial:** O estado inicial é a **descrição atual** do jogo do qual o algoritmo implementado em *Prolog* começa a sua procura. Este estado é **atualizado** usando o *Python* que trata de fazer a jogada de ambos os jogadores.

- **Terminal:** Considera-se **estado terminal** quando um dos jogadores tem **mais de 24 pontos** ou caso não exista peças no tabuleiro.

- **Operação: Existem 14 operações** 7 para cada jogador que são para o primeiro jogador: 0,1,2,3,4,5,6 e para o segundo jogador: 0,7,8,9,10,11,12. Existem **3 casos** para cada operação:

  - **1º Caso:** O jogador adversário **não tem peças** portanto ao fim desta jogada tem de se **introduzir peças no seu lado**.
  - **2º Caso:** O tabuleiro do agente tem **pelo menos** uma casa com **mais que uma peça** e a posição escolhida tem mais que uma peça.
  - **3º Caso:** O tabuleiro do agente tem **no máximo uma peça** em cada casa e a posição escolhida tem uma peça.

- **Distribuição:** Se a casa que estamos a distribuir tiver **mais do que 12 peças**, o jogador dá uma **volta completa** ao tabuleiro. Para fazer esta operação calculamos o **quociente** da divisão do valor das peças por 11 (número de casas retirando a casa de partida), e adicionamos o valor do quociente a todas as casa do tabuleiro. Falta então tratar do **resto** da divisão que é distribuido usando uma lista auxiliar, que começa na casa exactamente a seguir à escolhida e acaba na casa anterior. Depois é **inserido** um zero na posição escolhida e procede-se à captura.

- **Captura:** A captura **só é válida** caso a ultima peça distribuida seja no **lado do adversário**, ainda mais, só se pode capturar em **casas do adversário**. Usando uma lista auxiliar que começa na ultima casa modificada e segue o sentido do **ponteiro do relógio** capturando casas com 2 ou 3 sementes. Estes pontos são depois adicionados ao jogador correspondente.

4 Heuristica

Para calcular o **valor de cada jogada** definiu-se 3 formulas:

H1

Tem como prioridade as **vitórias**.

```
valor([X,_,_], 10) :- X @> 24.
valor([_,Y,_], -10) :- Y @> 24.
valor([X,Y,_], 0) :- X @< 25, Y @< 25.
```

- **Vantagens:** O objectivo final é **ganhar** o jogo, sendo que poderá existir uma jogada que inexplicavelmente pode ser melhor.
- **Desvantagens:** Como o algoritmo tem um limite de tempo **não** se consegue **prever** todas as jogadas, pelo que resulta num fraco desempenho.

H2

Tem como prioridade o **maior** número de pontos.

```
valor([X,Y,_], 10) :- X @> Y.
valor([X,Y,_], -10) :- Y @> X.
valor([X,Y,_], 0) :- X @< 25, Y @< 25.
```

- **Vantagens:** É mais realista face ao objectivo final e à limitação de **tempo**.
- **Desvantagens:** Isto pode ser uma visão um pouco redutora do jogo em **completo**. Isto pode levar a algumas jogadas que seriam vantajosas a **não serem realizadas**.

H3

Tem como prioridade **maximizar a diferença** de pontos entre os dois jogadores.

```
valor([X,Y,_], Z) :- Y @> X, Z is Y - X.
valor([X,Y,_], Z) :- X @> Y, Z is Y - X.
valor([X,Y,_], 0) :- X @< 25, Y @< 25.
```

- **Vantagens:** Neste caso o agente vai sempre procurar estar em **constante vantagem** de pontos face ao oponente.
- **Desvantagens:** Apesar de serem **mínimas**, as desvantagens apresentadas anteriormente (na segunda formula) podem ocorrer.

	H1 Vs H2			H2 Vs H1			H2 Vs H3			H3 Vs H2		
Grau	1º	2º	3º	1º	2º	3º	1º	2º	3º	1º	2º	3º
Vencedor	H2	H2	H2	H2	H2	H1	H3	H3	H3	H3	H3	H3
Jogadas	35	31	35	49	22	18	9	9	6	27	13	24

5 Torneio

Resultados do Torneio

	Minimax Vs Alfabeta			Alfabeta Vs Minimax		
Grau	1º	2º	3º	1º	2º	3º
Vencedor	Alfabeta	Alfabeta	Alfabeta	Alfabeta	Alfabeta	Alfabeta
Jogadas	16	7	16	12	21	12

6 Conclusão

Por o algoritmo minimax com cortes alfabeta conseguir **explorar mais nós**, consegue então escolher melhores jogadas que o algoritmo minimax. Devido a isto decidiu-se escolher o **algoritmo alfabeta** como a **versão participante** no famoso "I Torneio de Ouri deIA@LEI@UÉ".