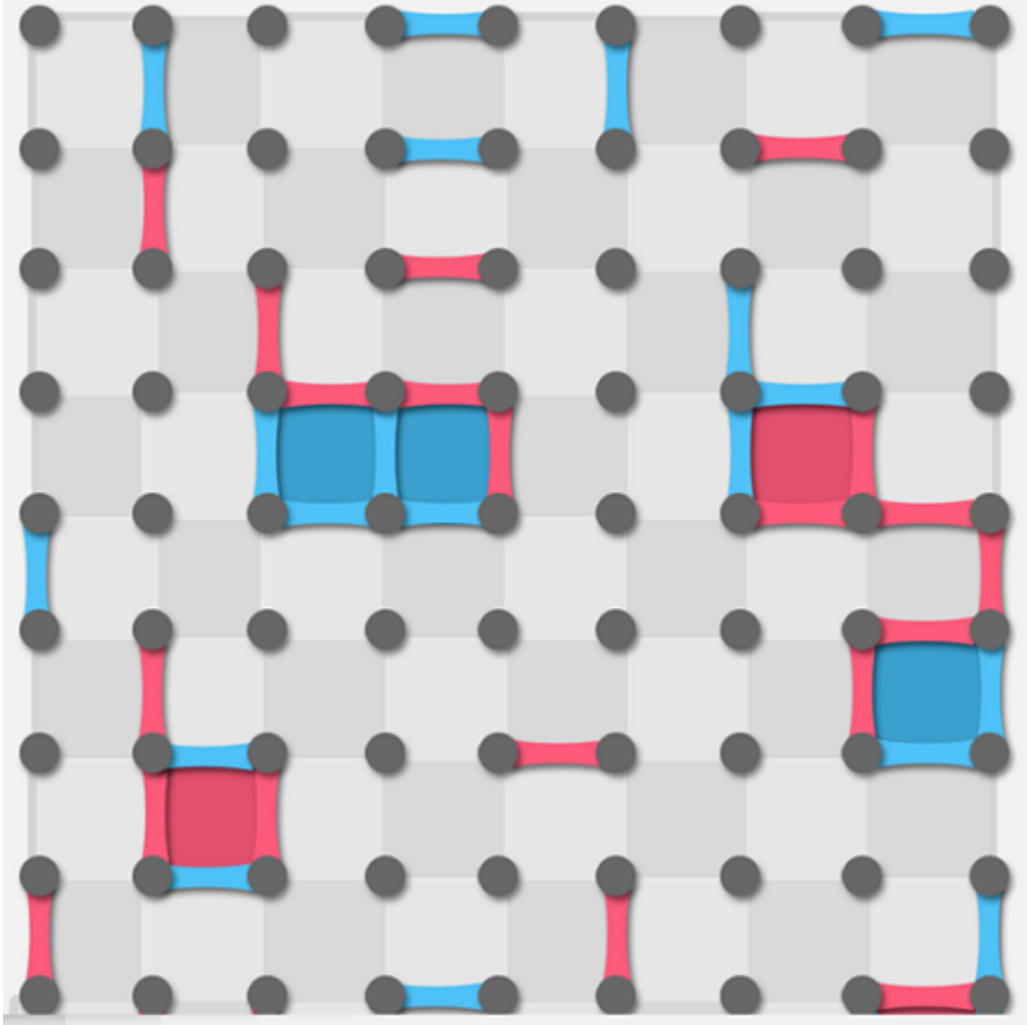
Manual de Utilizador

Dots & Boxes



IPS ESTS - Licenciatura Engenharia Informatica - 2022/2023

Inteligência Artificial
Docente: Joaquim Filipe

Trabalho realizado por:

Nome: Daniel Baptista - Nº:202001990
Nome: Rafael Silva - Nº:202001553

2. Entidades e sua implementação

- ***no*** - entidade que representa o ponto do tabuleiro de jogo.
- ***no-estado*** - entidade que representa o estado em que se encontra o ponto do tabuleiro de jogo.
- ***sucessores*** - entidade que representa todos os sucessores de um nó atual.

3. Algoritmos e sua implementação

Algoritmo MiniMax com cortes alfa-beta

É um algoritmo que faz a procura através do tipo *depth-first*, pelo que em cada instante apenas é necessário considerar os nós ao longo de um ramo da árvore de procura.
Seja **α** o valor da melhor escolha encontrada até ao momento, ao longo do ramo corrente, para um nó **MAX**.
Seja **β** o valor da melhor escolha encontrada até ao momento, ao longo do ramo corrente, para um nó **MIN**.
Este vai atualizando o valor de **α** e **β** ao longo da procura podendo ou não cortar subárvores dependente se se sabe que os valores correntes sejam piores dos que os que já temos.
A complexidade algoritmica é de $O(b^{m/2})$, em que **m** é a profundidade máxima e **b** o fator de ramificação.

Possivel forma de implementação do Alfa-Beta

1. Se **n** no limite de profundidade **d**, devolve $AlfaBeta(n)=f(n)$, caso contrário calcula os sucessores n1, ..., nk, ..., nb (por ordem), faz k=1 e, se **n** é um nó **MAX** começa no ponto **2.**, caso seja nó **MIN** começa no ponto **8.**
2. v = -∞
3. v <- max[v, AlfaBeta(nk; α; β)]
4. α <- max[v, α]
5. Se α ≥ β devolve β (corte)
6. Se k=b devolve α;
7. Caso k≠b então retorna ao passo 3.
8. v = +∞
9. v <- min[v, AlfaBeta(nk; α; β)]
10. β <- min[v, β]
11. Se β ≤ α devolve α (corte)
12. Se k=b devolve β;
13. Caso k≠b então retorna ao passo 9.

Características do MiniMax com cortes alfa-beta

- Examina menos nós do que o MiniMax devido aos cortes.
- Não altera os resultados finais em relação ao NegaMax ou MiniMax.
- É mais rápido para fazer a procura em relação ao MiniMax.

Implementação feita no projeto

- **Função Auxiliar**

```
(defun filtrar-nos-filhos (nos)
  "Função para remover os nós em que o tabuleiro seja NIL"
  (reduce 'append (mapcar (lambda (no)
                            (cond((null (no-tabuleiro no)) NIL)
                                 (t (list no))))
                          nos)))
```

- **Algoritmo**

```
(defun alfabeta (no alfa beta operadores sucessores avaliacao profundidade
jogador)
  "Algoritmo de procura da melhor jogada possivel implementado com MiniMax com
cortes Alfa-Beta"
  (labels ((maximizar (nos alf bet &optional (valor -10000000))
             (cond ((null nos) alf)
                   (t (let* ((temp-valor (max valor (alfabeta (car nos) alf bet
operadores sucessores avaliacao (1- profundidade) (trocar-jogador jogador))))
                             (nos-analisados (inc-nos-analisados))
                             (temp-alfa (max temp-valor alf)))
                        (cond ((>= temp-alfa bet) (let ((cortes-beta (inc-cortes-
beta)))
                                                    bet))
                              (t (maximizar (cdr nos) temp-alfa bet temp-
valor)))))))
           (minimizar (nos alf bet &optional (valor 10000000))
             (cond ((null nos) bet)
                   (t (let* ((temp-valor (min valor (alfabeta (car nos) alf bet
operadores sucessores avaliacao (1- profundidade) (trocar-jogador jogador))))
                             (nos-analisados (inc-nos-analisados))
                             (temp-beta (min temp-valor bet)))
                        (cond ((<= temp-beta alf) (let ((cortes-alfa (inc-cortes-
alfa)))
                                                    alf))
                              (t (minimizar (cdr nos) alf temp-beta temp-
valor)))))))))
    (cond ((or (= 0 profundidade) (tabuleiro-preenchidop (no-tabuleiro no)))
(avaliacao no))
          (t (let ((nos-filhos (filtrar-nos-filhos (sucessores no operadores
jogador))))
               (cond ((= jogador 2) (maximizar nos-filhos alfa beta))
                     (t (minimizar nos-filhos alfa beta))))))))
```

4. Descrição das opções tomadas

Uma das decisões tomadas foi a troca de variaveis globais de contagem de nós e cortes **α** e **β**, sugerida pelo professor, para closures de modo a não infrigir-mos qualquer propriedade da programação funcional. De resto foi feito a implementação através da logica dos ultimos laboratorios disponibilizados na cadeira.

5. Limitações técnicas e ideias para desenvolvimento futuro

Limitações técnicas do programa:

- Tempo de procura do algoritmo Alfa-Beta não é limitado.
- Não é possivel jogar computador vs computador.

Ideias para desenvolvimento futuro poderiam ser:

- Implementar a limitação de tempo para a procura do algoritmo Alfa-Beta.
- Fazer implementação do modo de jogo de computador vs computador.
- Refactoring de modo a obter um maior nivel de abstração.
- Possiveis melhoramentos no desempenho do algoritmo de procura através do uso de hash tables.