U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3: Adversarial Search

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Jogos como Problemas de Pesquisa

- Agente Hostil (adversário) incluído no mundo!
- Oponente Imprevisível => Solução é um Plano de Contingência
- Tempo Limite => Pouco provável encontrar objetivo! É necessário uma aproximação
- Uma das áreas mais antigas da IA! Em 1950 Shannon e Turing criaram os primeiros programas de Xadrez!
- Xadrez:
  - Todos consideram que é necessário inteligência para jogar
  - Regras simples mas o jogo é complexo
  - Mundo totalmente acessível ao agente
  - Fator de ramificação médio de 35, partida com 50 jogadas => 35^{100} folhas na árvore de pesquisa (embora só existam 10^{40} posições legais)
- Conceitos de corte na árvore de pesquisa e função de avaliação**!**

Tipos de Jogos

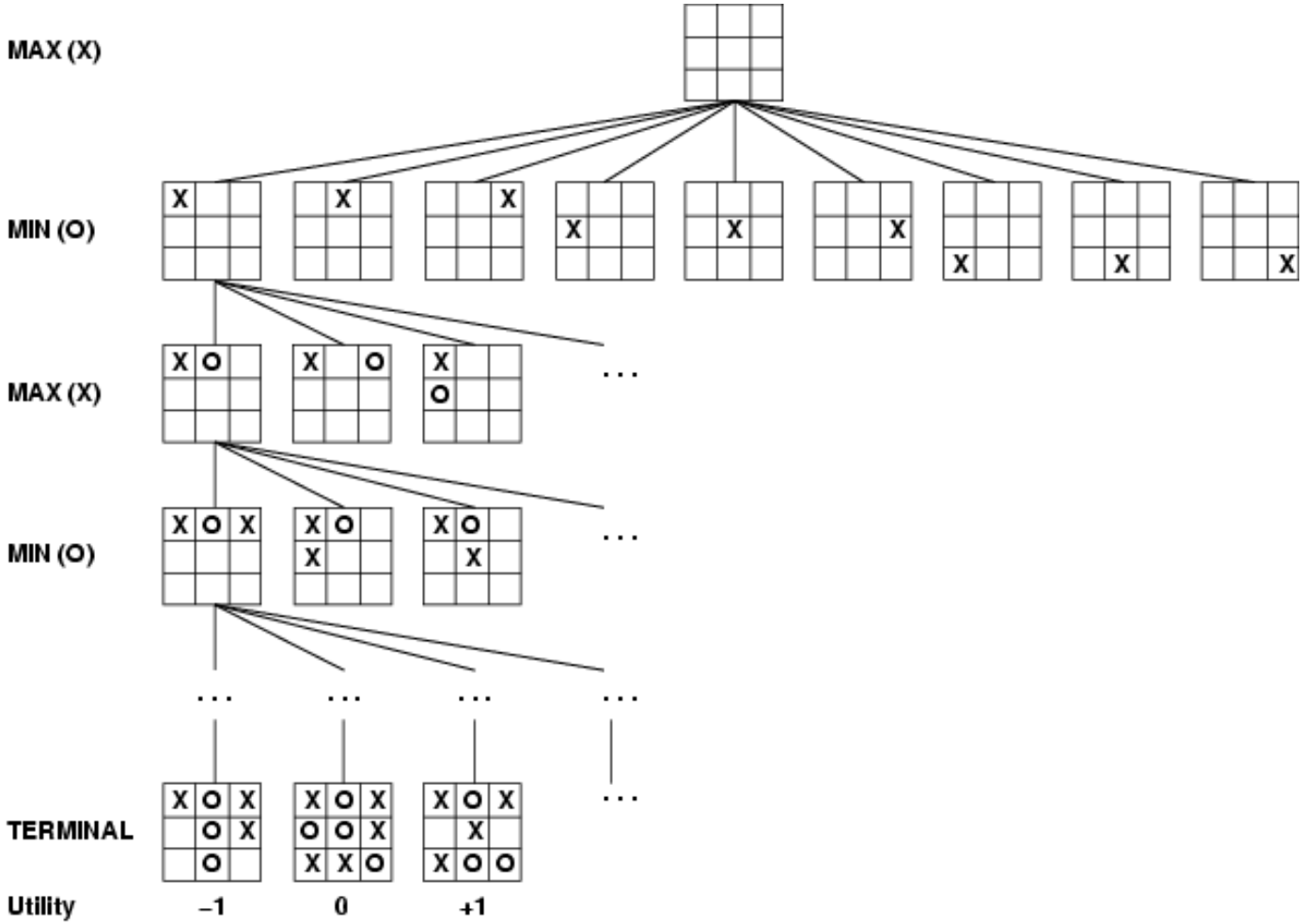
- **Tipos de Jogos:**
  - Informação:
    - Perfeita: Xadrez, Damas, Go, Otelo, Gamão, Monopólio
    - Imperfeita: Poker, Scrabble, Bridge, King
  - Sorte/Determinístico:
    - Determinístico: Xadrez, Damas, Go, Otelo
    - Jogo de Sorte: Gamão, Monopólio, Poker, Scrabble, Bridge, King
- **Plano de "Ataque":**
  - Algoritmo para o jogo perfeito
  - Horizonte finito, avaliação aproximada
  - Cortes na árvores para reduzir custos

Decisões Perfeitas em Jogos com Adversário – Algoritmo MiniMax

- **Jogo: Problema de pesquisa com:**
  - Estado Inicial (posição do tabuleiro e qual o próximo jogador a jogar)
  - Conjunto de Operadores (que definem os movimentos legais)
  - Teste Terminal (que determina se o jogo acabou ou seja está num estado terminal)
  - Função de Utilidade (que dá um valor numérico para o resultado do jogo, por exemplo 1-vitória, 0-empate, -1-derrota)
- **Estratégia do algoritmo Minimax:**
  - Gerar a árvore completa até aos estados terminais
  - Aplicar a função utilidade a esses estados
  - Calcular os valores da utilidade até a raiz da árvore, uma camada de cada vez
  - Escolher o movimento com o valor mais elevado!

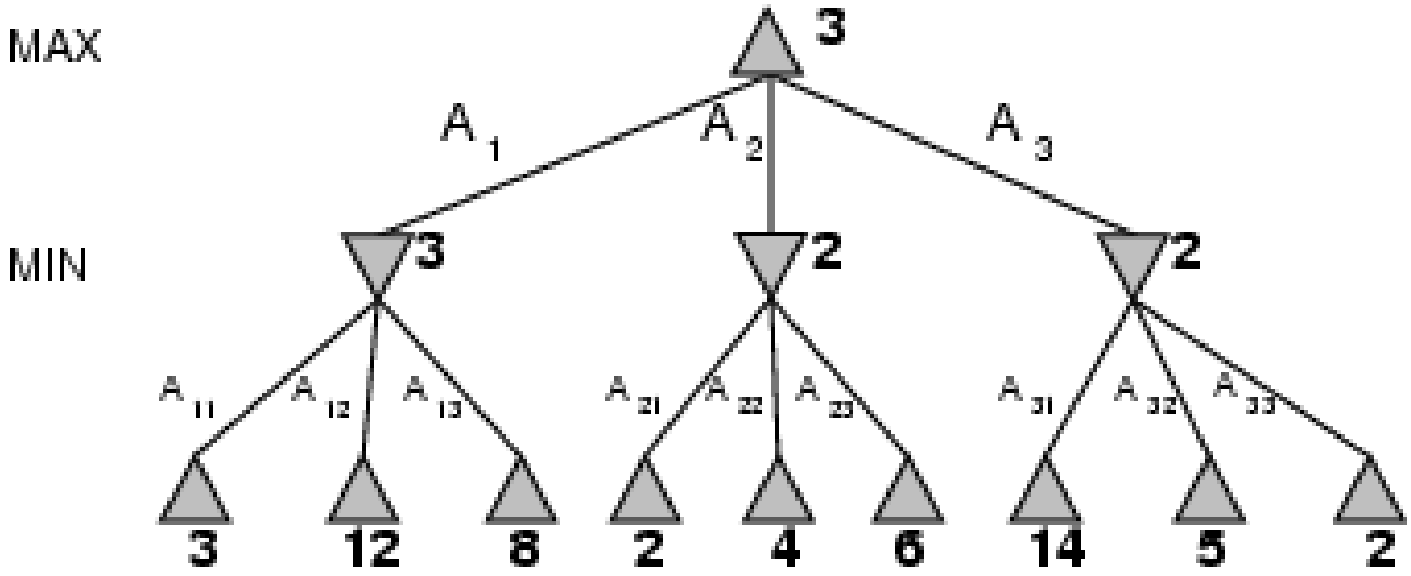
Minimax - Exemplo para o Jogo do Galo

Fácil resolver o Jogo do Galo utilizando Minimax



Minimax - Exemplo Geral

- **Estratégia:**
  - Escolher o movimento que tem o maior valor minimax = melhor que se pode conseguir contra as melhores respostas do adversário!



Algoritmo Minimax

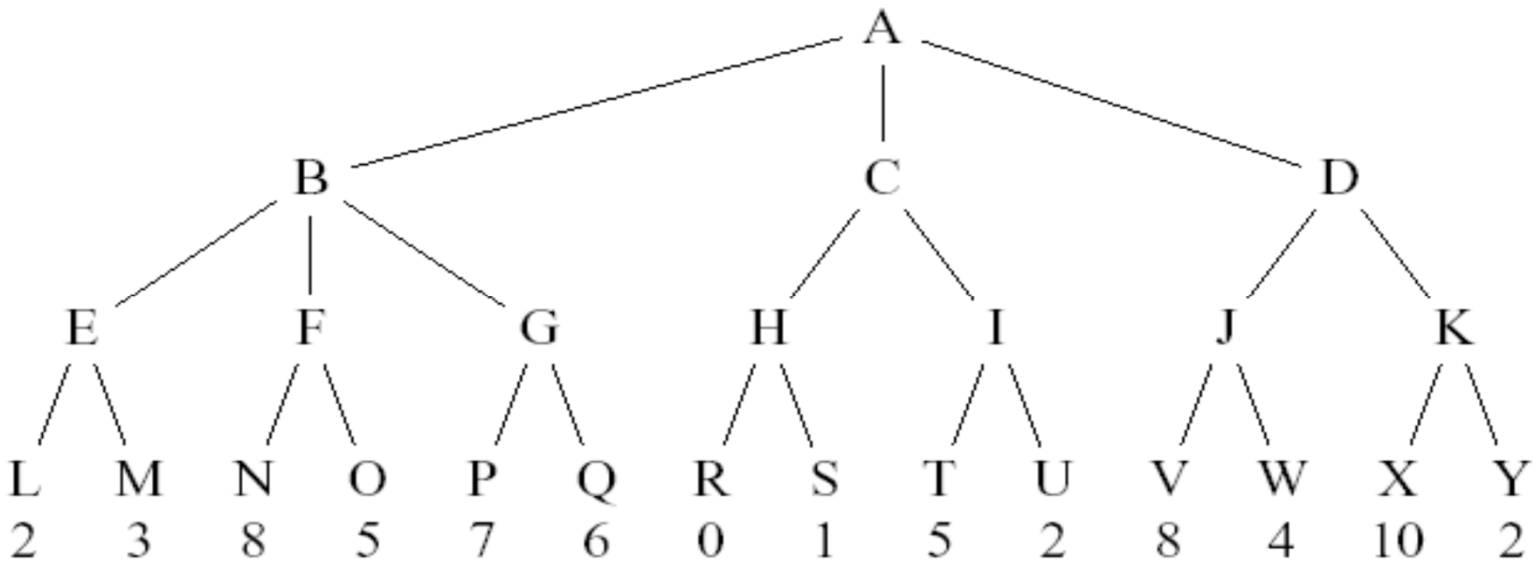
```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Exercício – MINIMAX

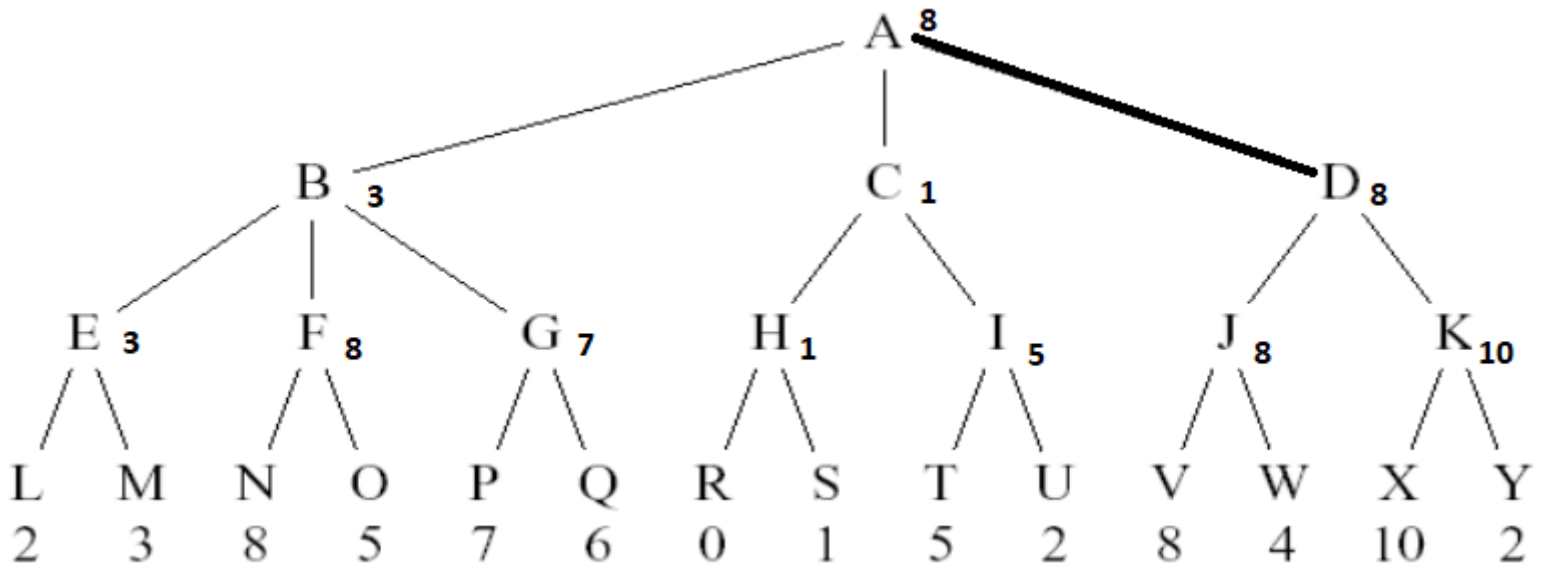
- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax à seguinte árvore, indicando o movimento selecionado pelo algoritmo e o respetivo valor estimado.



Exercício – MINIMAX

Solução:

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax à seguinte árvore, indicando o movimento selecionado pelo algoritmo e o respetivo valor estimado.



Propriedades do Minimax

- **Propriedades:**
  - Completo? Sim se a árvore for finita!
  - Ótimo? Sim contra um adversário ótimo! Senão?
  - Complexidade no Tempo? $O(b^m)$
  - Complexidade no Espaço? O(bm) (exploração primeiro em profundidade)

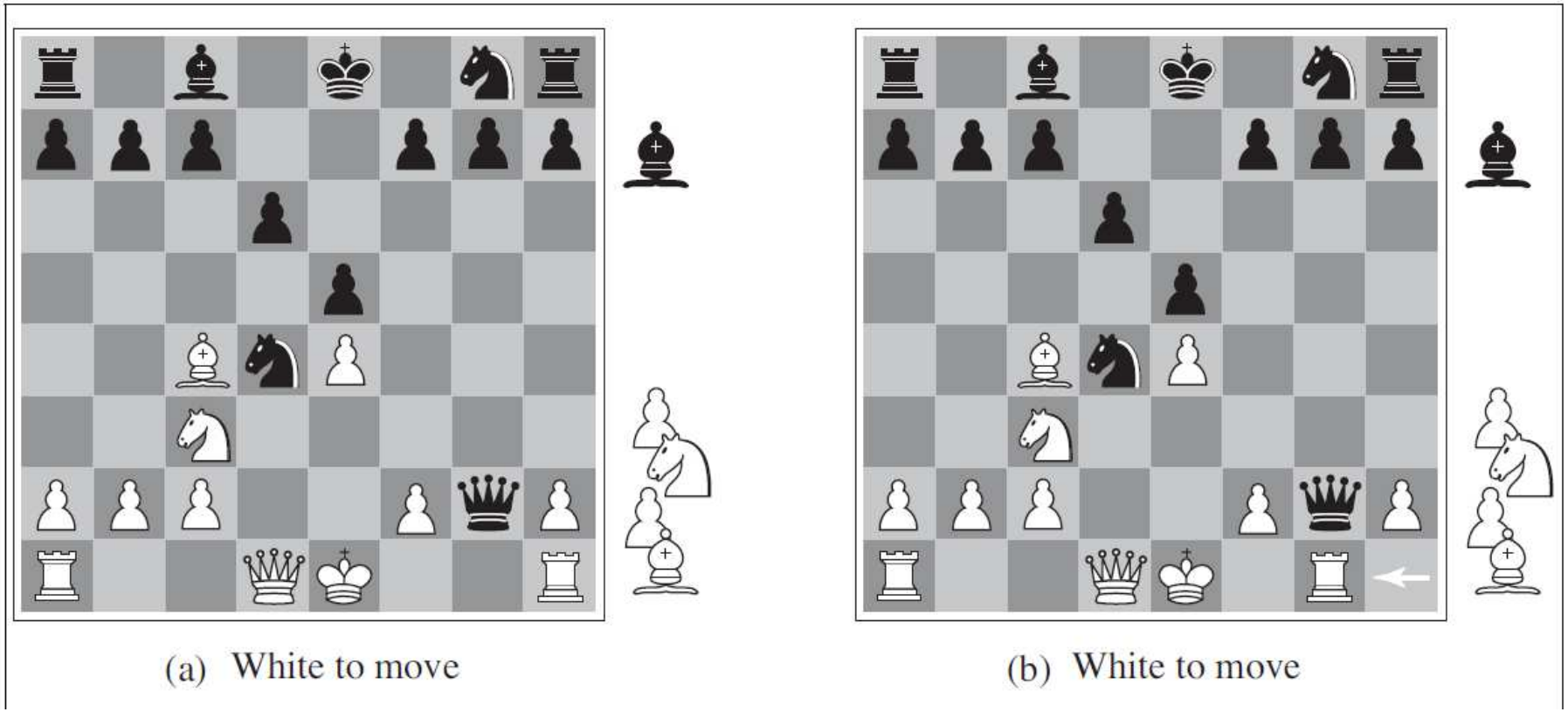
- **Problema:**
  - Inviável para qualquer jogo minimamente complexo
- **Exemplo:**
  - Para o xadrez (b=35, m=100), $b^m=35^{100}=2.5*10^{154}$
  - Supondo que são analisadas 450 milhões de hipóteses por segundo => $2*10^{138}$ anos para chegar à solução!

Recursos Limitados

- Supondo que temos 100 segundos e exploramos 10^4 nós/segundo, podemos explorar 10^6 nós por movimento
- **Aproximação usual:**
  - Teste de Corte: Profundidade Limite
  - Função de Avaliação: Utilidade (interesse) estimada para a posição
- **Exemplo (Xadrez):**
  - Teste de Corte: Profundidade de Análise n
  - Função de Avaliação simples = soma dos valores das peças brancas em jogo menos a soma dos valores da peças negra em jogo!
  - Função de avaliação só deve ser aplicada a posições estáveis (em termos do seu valor). Por exemplo, posições com possíveis capturas devem ser mais exploradas…
  - Outro Problema: Problema do horizonte!

Decisões Imperfeitas

- **Em (a) as pretas vencem mas em (b) as brancas vencem!**

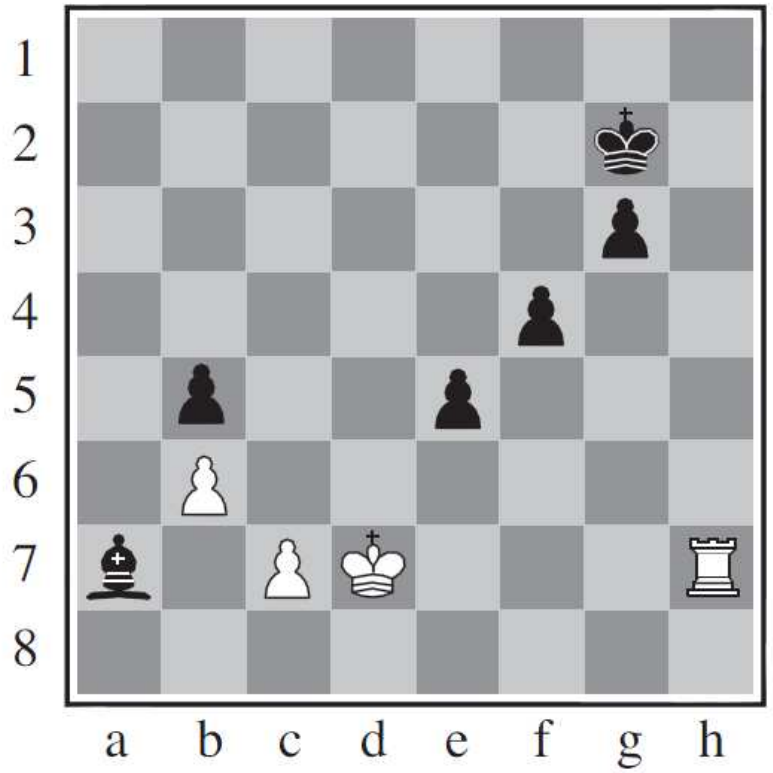


(a) White to move

(b) White to move

Xadrez - Problema do Horizonte

- Com um horizonte limitado parece interessante para as pretas dar cheque com os peões sendo que atrasam a captura do bispo (para além do horizonte)

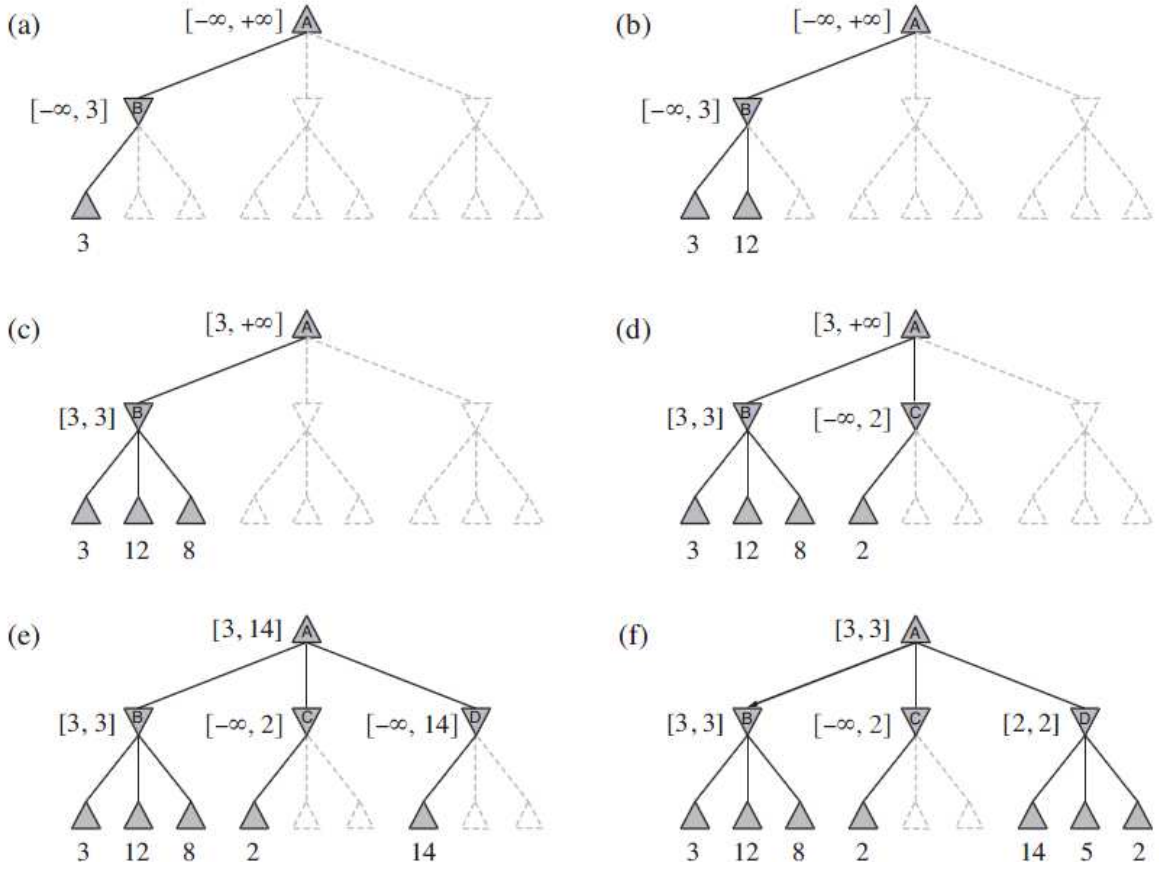


Cortes à Pesquisa

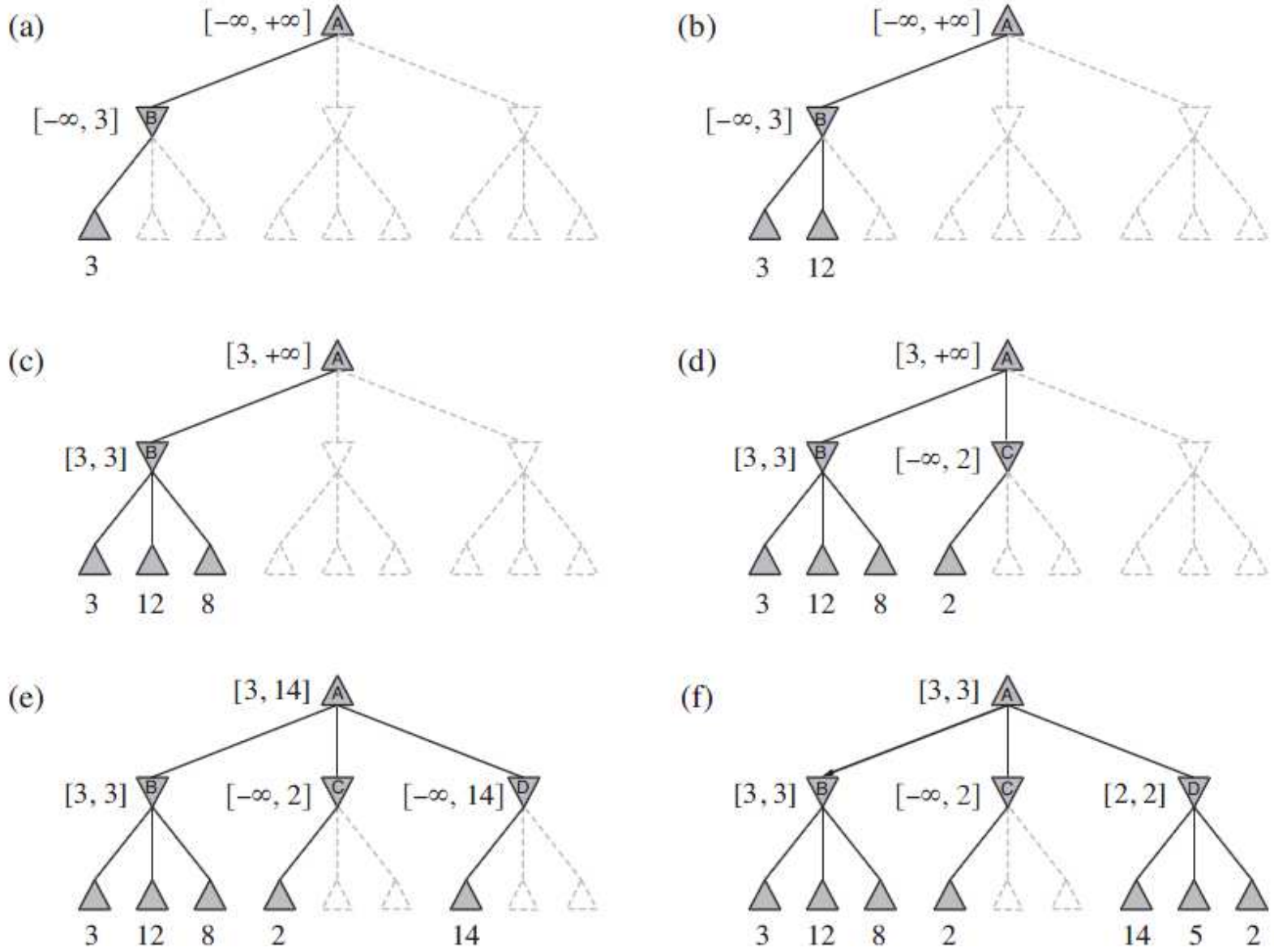
- ***MinimaxCutoff* é idêntico ao *MinimaxValue* excepto:**
  - *Terminal-Test* é substituído por *Cutoff*
  - *Utility* é substituída por *Evaluation* (que calcula uma avaliação da posição atingida)
- **Será que funciona na prática?**
  - Se $b^m=10^6$ com b=35 => m=4
- **Um jogar de xadrez com profundidade 4 é absolutamente miserável!**
  - Profundidade 4 => Jogador Novato
  - Profundidade 8 => PC, M.Bom Jogador Humano
  - Profundidade 12 => Deep Blue, Kasparov

Cortes Alfa-Beta

- α é o melhor valor (para Max) encontrado até agora no caminho corrente
- Se V for pior do que α, Max deve evitá-lo => cortar o ramo
- β é definido da mesma forma para Min



Cortes Alfa-Beta (2)



Cortes Alfa - Beta (3)

- **Cortes Alfa-Beta não afetam o resultado final**
- **Boa ordenação melhora a eficiência dos cortes**
  - Essencial raciocinar sobre a ordenação
- **Com ordenação perfeita: Complexidade no Tempo = $O(b^{m/2})$**
  - Duplica a profundidade de pesquisa
  - Profundidade 8 => Bom jogador de Xadrez
- **Bom exemplo do valor de raciocinar sobre que computações são relevantes:**
  - Isto é essencial em Sistemas Inteligente/Sistemas Baseados em Conhecimento

Cortes Alfa - Beta (4)

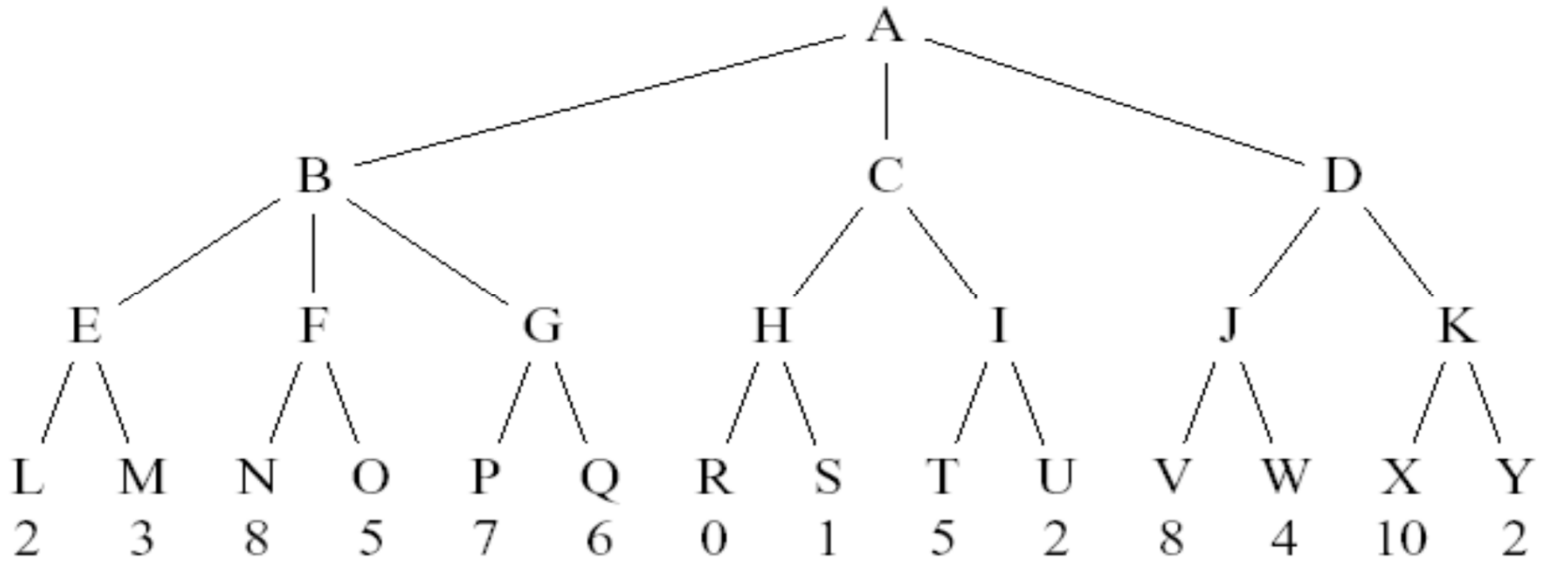
```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Exercício – MINIMAX com Cortes

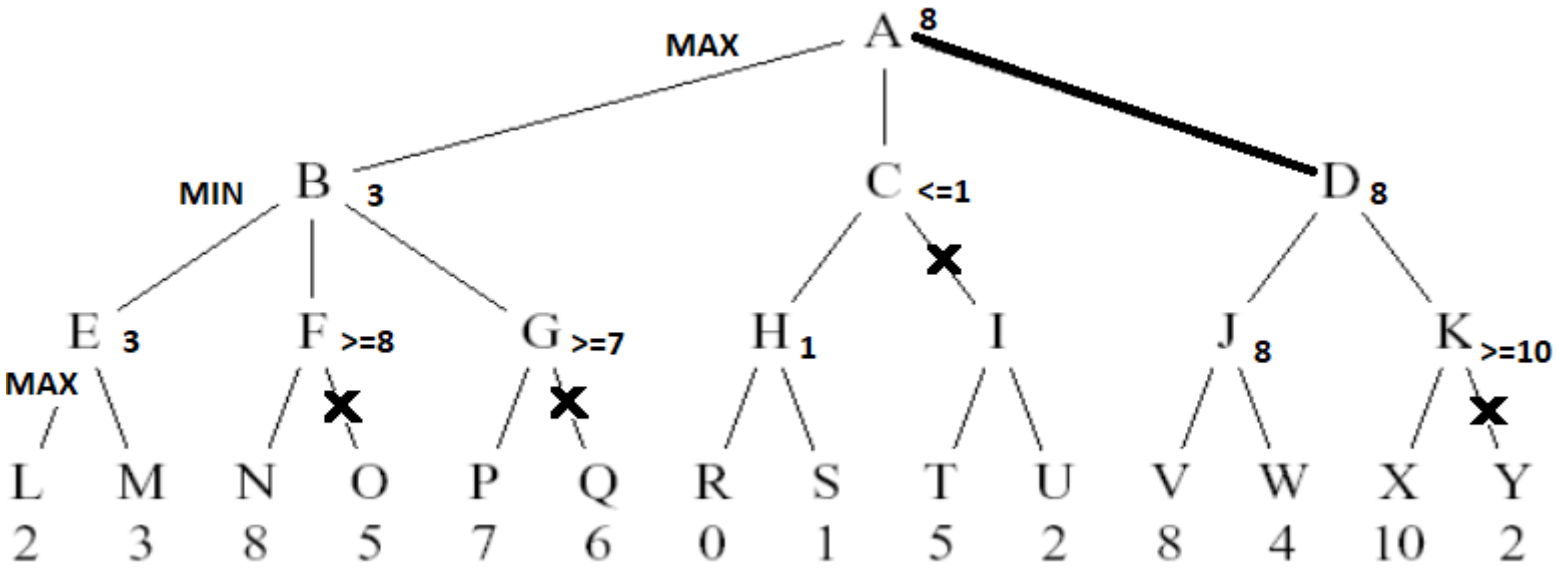
- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax com cortes Alfa-Beta à seguinte árvore, indicando o movimento selecionado pelo algoritmo. Indique graficamente e justifique todos os cortes que efetuar na aplicação do algoritmo Minimax



Exercício – MINIMAX com Cortes

Solução:

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax com cortes Alfa-Beta à seguinte árvore, indicando o movimento selecionado pelo algoritmo. Indique graficamente e justifique todos os cortes que efetuar na aplicação do algoritmo Minimax



Exercício – MINIMAX com Cortes

- Supondo que MAX é o primeiro a jogar, aplique o Algoritmo Minimax com cortes Alfa-Beta a uma árvore com três níveis, um factor de ramificação 3 e com os seguintes valores da função avaliação para a linha final:

[8 6 3 1 10 3 15 20 6 7 4 6 25 10 4 4 3 5 1 20 4 12 1 10 4 22 10]

- Indique graficamente todos os cortes que efetuar na aplicação do algoritmo.

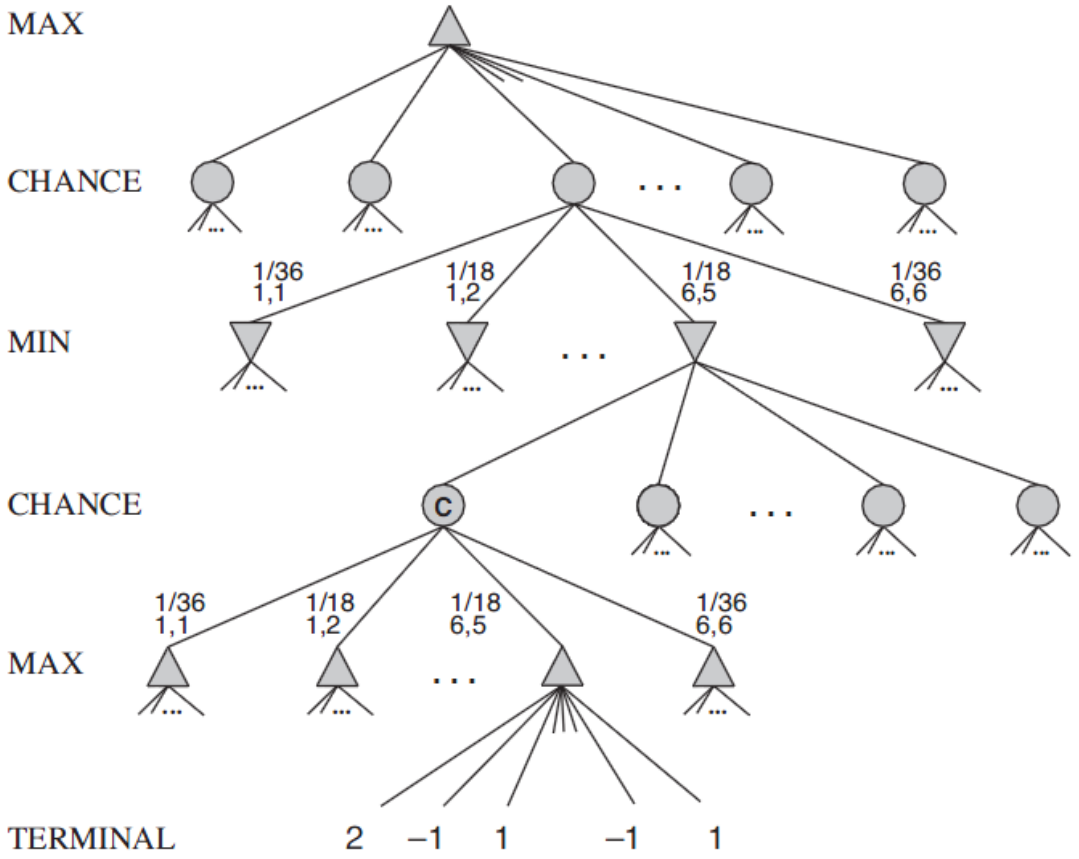
Jogos Determinísticos

- **Damas:**
  - **Chinook** acabou com o reinado de 40 anos do campeão humano Marion Tinsley em 1994. Usava uma base de dados para finais de partida definindo a forma perfeita de vencer para todas as posições envolvendo 8 ou menos peças (no total de 443748401247 posições). Hoje em dia é um **problema resolvido**.
- **Xadrez:**
  - **Deep Blue derrotou** o campeão do mundo humano **Gary Kasparov** num jogo com 6 partidas em 1997. Deep Blue pesquisava 200 milhões de posições por segundo e usa uma função de avaliação extremamente sofisticada e métodos (não revelados) para estender algumas linhas de pesquisa para além da profundidade 40!
- **Otelo:**
  - Campeões **humanos recusam-se a competir com computadores** pois não têm qualquer hipótese! (b entre 5 e 15)
- **Go (2015):**
  - Campeões humanos recusam-se a competir com computadores pois as **máquinas não conseguem jogar razoavelmente (b>300)**
- **Go (2017):**
  - AlphaGo (Fan, Lee), AlphaGo Master, AlphaGo Zero and AlphaZero! Máquinas vencem 100-0 campeões humanos e máquinas anteriores.

Jogos Determinísticos - Go

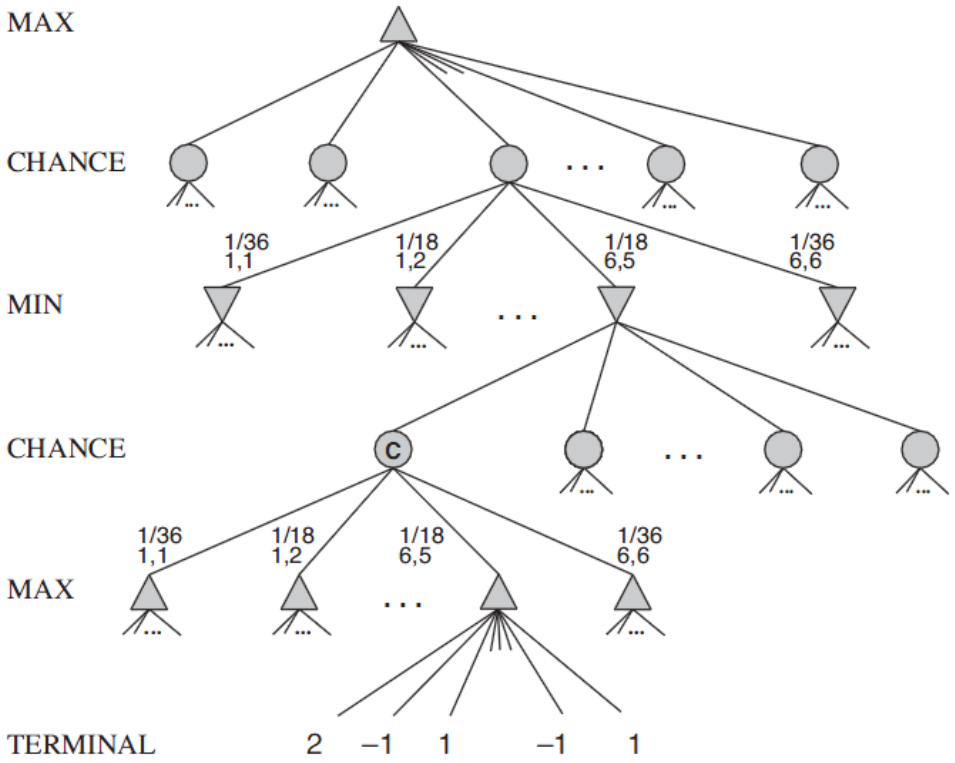
Jogos de Azar

- Em muitos jogos, ao contrário do xadrez, existem eventos externos que afetam o jogo, tais como tirar uma carta ou lançar um dado!
- Exemplos: Jogos de cartas, Gamão, Scrabble, ...
- Árvore de pesquisa deve incluir nós de probabilidade!
- Decisão é efetuada com base no valor esperado!
- *ExpectiMiniMax*



Jogos de Azar

- Expecti Minimax
  - Árvore de pesquisa tem nós de probabilidade!
  - Decisão efetuada com base no valor esperado!



$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \\ \sum_r P(r)\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if PLAYER}(s) = \text{CHANCE} \end{cases}$$

Sumário - Jogos

- **Trabalhar com jogos é extremamente interessante (mas também é perigoso…)**
  - Fácil testar novas ideias!
  - Fácil comparar agentes com outros agentes
  - Fácil comparar agentes com humanos!
- **Jogos ilustram diversos pontos interessantes da IA:**
  - Perfeição é inatingível => é necessário aproximar!
  - É boa ideia pensar sobre o que pensar!
  - Incerteza restringe a atribuição de valores aos estados!
- **Jogos funcionam para a IA como a Formula 1 para a construção de automóveis!**

Exercícios - Jogos

Formular um dos seguintes jogos de tabuleiro para 2 jogadores como um jogo e projetar um agente inteligente capaz de o jogar, utilizando Minimax com cortes alfa-beta:

1. Xadrez (Chess)
2. Shogi (Xadrez Japonês)
3. Damas (Checkers)
4. 4 em Linha (Connect 4)
5. Attaxx
6. Damas Chinesas
7. Otelo (Reversi)
8. Abalone
9. Hex
10. Jogo do Galo 3D (4x4)
11. Diplomacy
12. Jogo do Futebol
13. Quarto (Gigamic)
14. Quixo (Gigamic)
15. Quoridor(Gigamic)
16. Sahara (Gigamic)
17. Pentaminós (8x8)
18. Estratégia (Stratego)
19. Link Five
20. Mancala
21. Fanorana
22. Nine Mens Morris
23. Alquerque
24. Tablut
25. Surakarta
26. Terrace
27. Go
28. Dots and Boxes
29. Dots and Hexagons
30. Amazons
31. Scrabble (pec. visíveis, letr.conh.)
32. Jogo do Galo (normal, memória e movimento)
33. Dominós (peças visíveis e sort. conhecido)
34. Gamão (sort. conhecido)
35. Paper & Pencil Racing
36. Arimaa
37. Gipf
38. Lines of Action
39. Mancala 4x8
40. Connections
41. Omega Chess
42. Tori Shogi
43. Fanorama
44. Hexxagon
45. Jungle Game (J.da Selva)
46. Seega (Tab. dim. Var,)
47. Halma (2 jog., Tab. dimensão variável)
48. Quits (Gigamic) (Tab. dimensão variável)
49. Pylos (Gigamic)
50. Tantrix (2 Jog., 7 Pecas, 3 Cores)
51. Ticket to Ride (2 jog., ordem cartas conhecida)
52. Carcassone (2 jog. Sort. peças prévio))
53. Settlers of Catan (2 jog. sem trocas de recursos)
54. Blokus
55. Sputnik (Gigamic)
56. Katamino (Gigamic)
57. Gobblet (Gigamic)
58. Quads (Gigamic)
59. Quoridor Kid (Gigamic)

Exercícios – Jogos (2)

60. Tortuga (Gigamic)
61. AutomatonWars
62. Trax
63. Cathedral
64. Virus (Desinfect the Core)
65. Labirinto Mágico (Ravensburg)
66. Zertz (Gipf Project)
67. Jogo da Escola
68. Raposa e Gansos
69. Continuo
70. Proximity
71. Jogo do O
72. Dameo
73. Congo
91. Havannah
74. Croda
75. Froglet
76. Batalha de Tanques
77. Xadrersi 08/09)
78. Luta de Cavalos
79. PahTum
80. Spider 4 em Linha
81. Pente (tab.o reduzido)
82. 6 em Linha
83. Xadrez Maharajah
84. PhutBall
85. Cats and Dogs
86. Conspirateur
87. Kamisado
88. Hijara
89. Susan
90. Overboard
91. Buffalo
92. Plateau (sem informação escondida)
93. TriOminos (2 jog, sem info. desconhecida)
94. Camelot
95. Volcano
96. Tumbling Down
97. Aboyne
98. Hexdame
99. Focus
100. CrossFire
101. Spangles
102. Xiangqi
103. Lines Of Action
104. Santorini
105. Traboulet
106. Bagh Chal
107. Corrida de Reis
108. Halma
109. Jogo de Y
110. Nosferatu
111. O Último Capitão
112. Accasta
113. Macadam
114. Photonic Attack
115. Guarda e Torres
116. Absorção

Exercícios – Jogos (3)

1. Implementar o algoritmo Minimax (sem e com cortes alfa-beta) utilizando uma linguagem convencional (C++/Java/Python)
2. Implementar o algoritmo Minimax (sem e com cortes alfa-beta) utilizando Prolog
3. Descrever e/ou implementar descrições do estado, geradores de movimentos e funções de avaliação para os seguintes jogos com fator sorte:

    Jogos de tabuleiro: Gamão, Monopólio, Scrabble

    Jogos de cartas: Viúva Negra (Hearts), King, Poker e Bridge
4. Implementar agentes para jogar os jogos da alínea anterior.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 3: Adversarial Search

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence