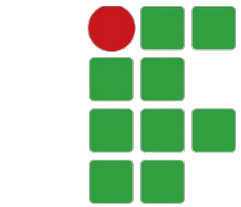
Inteligência Artificial

Busca Competitiva

Felipe Augusto Lima Reis
felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

Introdução

Busca Competitiva

- Ambientes competitivos são aqueles em que os objetivos dos agentes estão em conflito
  - Esses ambientes dão origem a problemas de busca competitiva, conhecidos como jogos [Russel and Norvig, 2013].
- Deve-se separar, no entanto, os algoritmos aprendidos nesta seção dos jogos eletrônicos mais modernos
  - Os algoritmos serão relacionados principalmente à Teoria de Jogos (Matemática / Economia) [Russel and Norvig, 2013];
  - Alguns algoritmos podem ser utilizados em jogos de tabuleiro, como xadrez, gamão e *go* [Coppin, 2004].

Teoria do Jogos

- A Teoria de Jogos corresponde a qualquer ambiente multiagente no qual o impacto de cada agente sobre os outros seja "significativo"
  - A teoria pode ser definida também como o estudo situações estratégicas onde jogadores tomam ações na tentativa de melhorar seu retorno [Nogueira, 2009];
- Destacam-se como pioneiros no desenvolvimento da Teoria dos Jogos, os trabalhos de John von Neumann e John Nash [Nogueira, 2009].

Teoria do Jogos

- Os jogos estudados são determinísticos e completamente observáveis, com revezamento de dois jogadores e soma zero
  - Jogos de Soma Zero são aqueles simétricos, cujos resultados são sempre iguais e opostos (ou simétricos)
    - Se um jogador ganha e o outro perde [Nogueira, 2009] [Russel and Norvig, 2013];
  - Nesses jogos, supõe-se que as decisões tomadas são racionais [Coppin, 2004].

Teoria do Jogos

- Cada jogador pode ter um número finito ou infinito de alternativas (ou estratégias);
  - Cada estratégia possui um valor de *payoff*, pago de um jogador a seu oponente;
  - *Payoffs* podem ser positivos (em caso de vitória) ou negativos (em caso de derrota);
  - A soma dos *payoffs* é zero [Nogueira, 2009];
- A teoria dos Jogos pode ser utilizada para:
  - Campanhas publicitárias, para produtos concorrentes;
  - Estratégias militares;
  - Jogos eletrônicos.

Árvores de Jogos

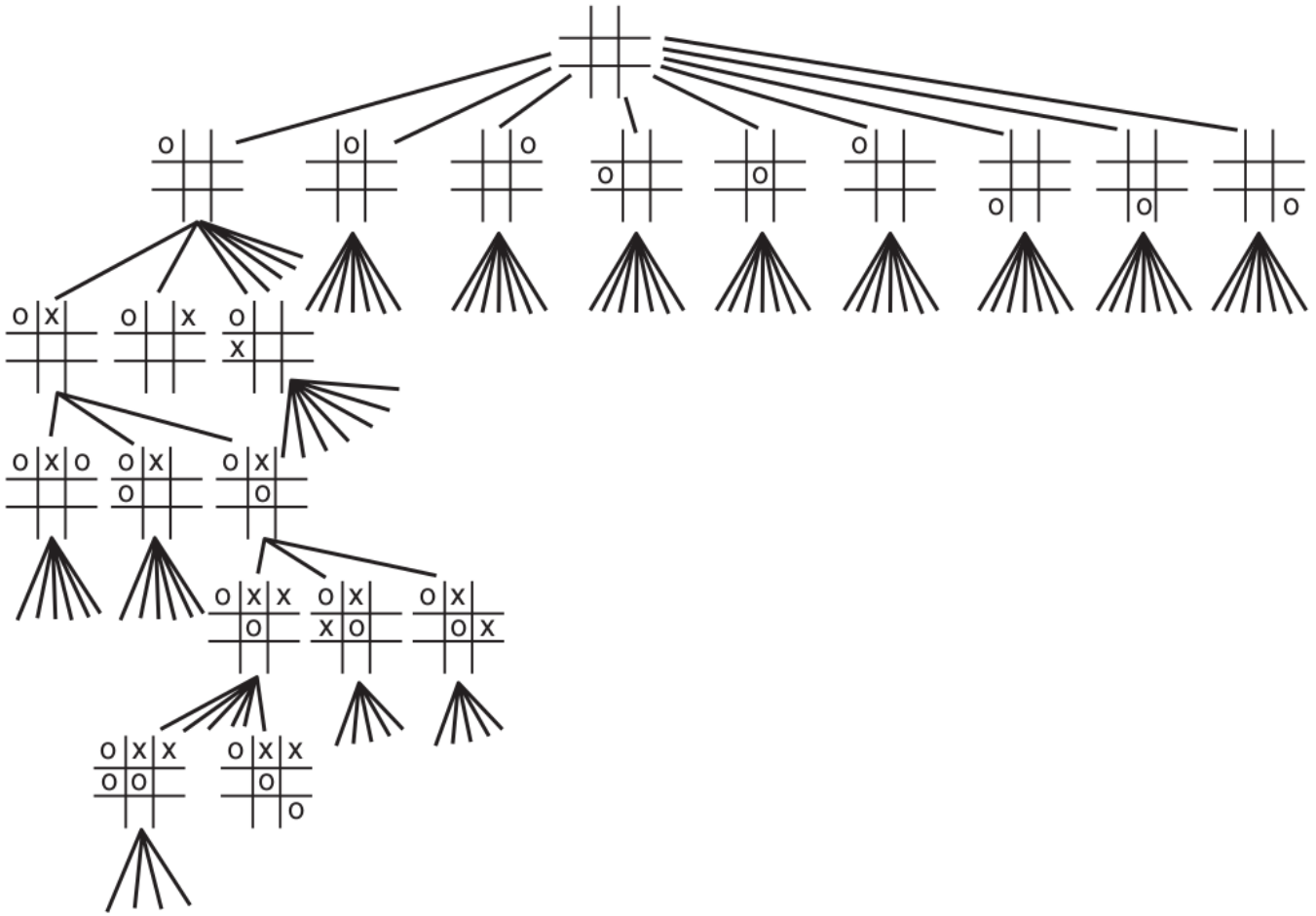
Árvores de Jogos

- Árvores de Jogos podem representar vários tipos de jogos entre dois oponentes de forma eficiente [Coppin, 2004];
  - Nela, a raiz representa o estado inicial, antes da execução de movimentos;
  - Os nós representam os estados ("posições") possíveis;
  - As arestas representam os movimentos;
  - Nós folhas representam os estados finais (vitória, perda ou empate) [Coppin, 2004];
- Para dois jogadores, os níveis pares podem representar os movimentos de um jogador, enquanto ímpares representam os movimentos executados pelo outro [Coppin, 2004].

Árvores de Jogos - Exemplo: Jogo da Velha

INSTITUTO FEDERAL Minas Gerais

- Movimentação de oponentes em um Jogo da Velha.



Fonte: [Coppin, 2004]

Árvores de Jogos

- Devido à competição, abordagens tradicionais como busca em Largura, em Profundidade ou A* não são adequadas
  - Essas abordagens são facilmente derrotadas, devido ao custo de processamento (tempo limitado) [Russel and Norvig, 2013]
    - "Cada movimento" do outro jogador deve ser derrotado, gerando um número muito alto de possibilidades;
  - Além disso, jogos competitivos contam com o fator intrínseco de incerteza [da Silva, 2014];
- Para solução de problemas competitivos são usados algoritmos específicos, como MiniMax e Poda Alfa-Beta [Coppin, 2004] [Russel and Norvig, 2013].

Árvores de Jogos

INSTITUTO FEDERAL
Minas Gerais

- Para manipulação das decisões, os algoritmos contam com funções de avaliação (avaliadores estáticos) [Coppin, 2004];
  - Essas funções tem como objetivo cortar a árvore de busca e avaliar as possibilidades atuais para uma sequência de passos;
  - A busca raramente será executada até a folha, devido a quantidade de possibilidades;
  - Uma função linear ponderada pode ser utilizada, para definir o peso das melhores soluções / características
    - Em um jogo de xadrez, por exemplo, uma função de avaliação pode ponderar e avaliar que é melhor perder um peão que uma rainha [Coppin, 2004].

Algoritmo MiniMax

Algoritmo MiniMax

- Para avaliar Árvore de Jogos, podemos assumir que o computador tenta maximizar sua pontuação, enquanto o oponente busca minimizá-la [Coppin, 2004]
  - Essa premissa dá origem a um jogo de soma zero;
- O algoritmo MiniMax é usado para escolha dos movimentos
  - O algoritmo assume que é possível alcançar a máxima pontuação a partir de um determinado nó;
  - A função de avaliação será usada nos nós-folhas, cujos valores serão filtrados pela árvore de soluções, para escolha do melhor caminho [Coppin, 2004];

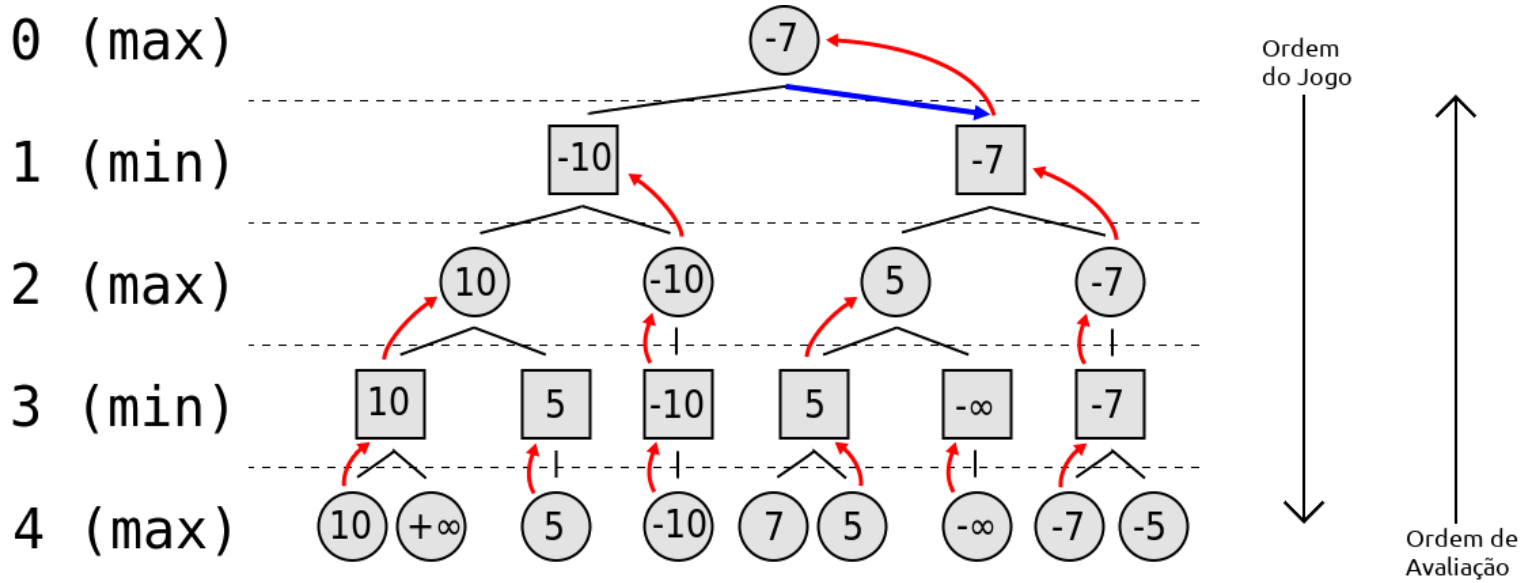
Algoritmo MiniMax

- O algoritmo avalia os custos da folha para a raiz, variando maximização e minimização, para prever ações do adversário
  - As ações previstas são baseados na premissa de que as ações do adversário são racionais;
    - Se o adversário não jogar de forma racional, a vitória será obtida de forma mais fácil;
  - Após maximizar as possibilidades, o algoritmo executa uma ação na direção do máximo valor possível.

Nesta seção, iremos aplicar o algoritmo MiniMax utilizando a Busca em Profundidade.

Algoritmo MiniMax

- Suponha que os círculos representam os movimentos do algoritmo e os quadrados, os movimentos do oponente
  - O algoritmo busca maximizar resultados (o adversário, minimizar);
  - O algoritmo inicia uma jogada no nível 0.



Fonte: Adaptado de [Wikipedia contributors, 2020]

Algoritmo MiniMax

- O algoritmo MiniMax também pode ser representado em função do *payoff*, pago de um adversário a outro
  - Nesse caso, é gerada uma matriz de *payoff*, correspondente às ações de cada oponente;
  - Para minimização, o algoritmo é chamado de MaxiMin.

	B_1	B_2	B_3	B_4	
A_1	8	-2	9	-3	
A_2	6	5	6	8	**Maximin**
A_3	-2	4	-9	5	
			Minimax		

Fonte: Adaptado de [Nogueira, 2009]

Algoritmo MiniMax

- No exemplo abaixo, a tabela de *payoff* contém o ganho máximo possível a partir da escolha do oponente
  - Se o oponente B escolhe B1, o oponente A deve escolher o valor que maximiza a função (A1);

	B chooses B1	B chooses B2	B chooses B3
A chooses A1	+3	−2	+2
A chooses A2	−1	0	+4
A chooses A3	−4	−3	+1

Payoff matrix for player A

Fonte: [Wikipedia contributors, 2020]

- Supõe-se que o oponente B tenha uma tabela oposta à tabela do oponente A (tabela de A com os sinais invertidos).

Algoritmo MiniMax

- Características:
  - Completo: executa uma exploração completa em profundidade da árvore de jogo[1];
  - Ótimo: retorna o estado ótimo [Russel and Norvig, 2013][2];
- Complexidade:
  - Tempo: $\mathcal{O}(b^m)$;
  - Armazenamento:
    - $\mathcal{O}(b^m)$, se o algoritmo gera todas as soluções;
    - $\mathcal{O}(m)$, se o algoritmo gera uma solução por vez[3].

[1] Na prática, o algoritmo pode não ser gerar todas as soluções, devido ao tempo de execução.

[2] O desempenho será melhor ainda quanto pior for o desempenho do oponente.

[3] Fonte: [Russel and Norvig, 2013]

Poda Alfa-Beta

Poda Alfa-Beta

- A Poda Alfa-Beta busca reduzir o custo de processamento da busca MiniMax
  - Devido a quantidade de estados, o algoritmo MiniMax têm custo exponencial em relação ao número de movimentos;
  - Para reduzir a complexidade, é calculado, então, um MiniMax "parcial", sem examinar todos os nós na árvore;
  - Uma poda é feita de forma a desconsiderar grandes partes da árvore - essa poda é chamada de Poda Alfa-Beta [Russel and Norvig, 2013].

Poda Alfa-Beta

- A Poda Alfa-Beta busca desconsiderar qualquer movimento que seja potencialmente pior que outro previamente analisado
  - Quando aplicada a uma árvore MiniMax, o algoritmo retorna o mesmo movimento que o MiniMax padrão retornaria;
  - No entanto, a Poda Alfa-Beta remove qualquer ramo que não influencie a decisão final;
  - O método pode podar subárvores inteiras no lugar de podar apenas folhas [Russel and Norvig, 2013];
- A efetividade da poda é altamente dependente da ordem em que os estados são examinados [Russel and Norvig, 2013].

Nesta seção, iremos aplicar o algoritmo Poda Alfa-Beta utilizando a Busca em Profundidade.

Poda Alfa-Beta

- Segundo [Russel and Norvig, 2013] e [Coppin, 2004], a poda tem seu nome devido a dois parâmetros do algoritmo
  - α: valor da melhor escolha encontrada (máximo), em qualquer ponto ao longo do caminho para MAX;
  - β: valor da melhor escolha encontrada (mínimo), em qualquer ponto ao longo do caminho para MIN.

Poda Alfa-Beta

INSTITUTO FEDERAL
Minas Gerais

- A propagação de valores na Poda Alfa-Beta é feita da seguinte maneira:
  - Nós máximos: o valor β mínimo para todos os ancestrais de nó mínimo é armazenado como β;
  - Nós mínimos: o valor α máximo para todos os ancestrais de nó máximo é armazenado como α;
  - Nós não-folhas terão um valores alfa e beta armazenados;
  - O nó raiz recebe um valor alfa de infinito negativo e um valor beta de infinito positivo [Coppin, 2004].

Algoritmo Poda Alfa-Beta

INSTITUTO FEDERAL
Minas Gerais

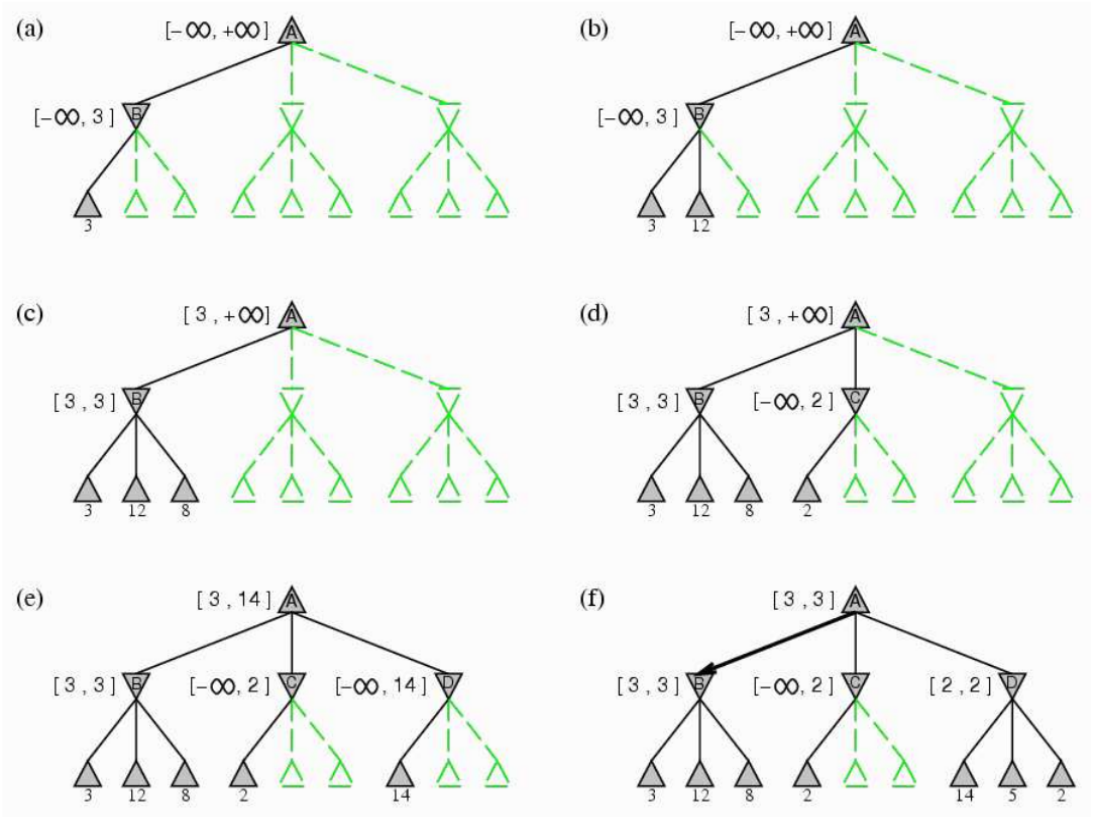
- A figura abaixo contém o algoritmo de Poda Alfa-Beta.

```
Function alpha_beta (current_node, alpha, beta)
{
     if is_root_node (current_node)
     then
     {
          alpha = -infinity
          beta = infinity
     }

     if is_leaf (current_node)
     then return static_evaluation (current_node);

     if is_max_node (current_node)
     then
     {
          alpha = max (alpha, alpha_beta (children, alpha, beta));
          if alpha >= beta
          then cut_off_search_below (current_node);
     }

     if is_min_node (current_node)
     then
     {
          beta = min (beta, alpha_beta (children, alpha, beta));
          if beta <= alpha
          then cut_off_search_below (current_node);
     }
}
```

Fonte: [Coppin, 2004]

Poda Alfa-Beta

- A figura abaixo contém um exemplo de Poda Alfa-Beta
  - Considere $\triangle$ como MAX e $\triangledown$ como MIN.



Fonte: [da Silva, 2014]

Poda Alfa-Beta

- Características:
  - Completo: sim, garante a seleção do estado-objetivo;
  - Ótimo: retorna o estado ótimo [Russel and Norvig, 2013][4];
- Complexidade:
  - Tempo: $\mathcal{O}(b^{\frac{m}{2}})$;
  - Armazenamento:
    - $\mathcal{O}(b^{\frac{m}{2}})$, se o algoritmo gera todas as soluções.

[4] O desempenho será melhor ainda quanto pior for o desempenho do oponente.

Referências I

Coppin, B. (2004).
Artificial intelligence illuminated.
Jones and Bartlett illuminated series. Jones and Bartlett Publishers, 1 edition.

da Silva, D. M. (2014).
Inteligência Artificial - Slides de Aula.
IFMG - Instituto Federal de Minas Gerais, Campus Formiga.

Nogueira, F. (2009).
Teoria do jogos - notas de aula.
[Online]; acessado em 20 de Outubro de 2020. Disponível em:
`https://www.ufjf.br/epd042/files/2009/02/jogos.pdf`.

Russel, S. and Norvig, P. (2013).
Inteligência artificial.
Campus - Elsevier, 3 edition.

Wikipedia contributors (2020).
Minimax.
[Online]; acessado em 20 de Outubro de 2020. Disponível em: `https://en.wikipedia.org/wiki/Minimax`.