



SCC-5774 - Capítulo 2

Busca com Adversários

João Luís Garcia Rosa¹

¹Departamento de Ciências de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
<http://www.icmc.usp.br/~joaoluis>

2020

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Jogos

- Diferentemente da busca tradicional vista até agora, na qual a situação não troca durante a busca, a **busca com adversários** considera que há oponentes hostis em sua trajetória.
- **Jogos** são o exemplo clássico.



Árvore de busca “tradicional” de xadrez

- Tamanho do espaço de estados (10^{120} - considerando uma média de 40 movimentos)
 - $10^{120} >$ Número de átomos no universo!!!
- Considerando que são computadas 200 milhões de posições/seg então seriam necessários 10^{100} anos para avaliar todos os jogos de xadrez
 - Idade do universo = 10^{10} anos.
- Assim, outras técnicas de busca são utilizadas em jogos

Considerações gerais

- Jogos são atrativos para IA
 - Formulação simples do problema (ações bem definidas)
 - Ambiente acessível
 - Abstração (representação simplificada de problemas reais)
 - Sinônimo de inteligência
- Problema desafiador
 - Tamanho (espaço de busca) + limitação de tempo
 - Restrições sobre recursos: difícil encontrar a meta
 - Adversário “imprevisível”: solução é ter um plano de contingência

Considerações gerais

- Jogos: dois tipos básicos de situações
 - Espaço de estados suficientemente pequeno
 - Espaço de estados muito grande
- Problema pode ser formulado como:
 - *Estado inicial*: posições do tabuleiro e indicação do jogador (de quem é a vez)
 - *Estado final*: posições em que o jogo acaba
 - *Operadores*: jogadas legais
 - *Função de utilidade*: valor numérico (pontuação) de uma posição

Caso simples

- Jogo de 2 pessoas
- Movimentos alternados
- Zero-soma: a perda para um jogador representa o ganho do outro
- Informação perfeita: ambos jogadores têm acesso a toda a informação relacionada ao estado corrente do jogo. Não há ocultação de informação para os jogadores.
- Não há “sorte” no jogo (ex., uso de dados)
- Exemplos: jogo da velha, damas, xadrez, go, nim (mas não bridge, poker, backgammon, ...)

Como jogar

- Uma maneira de jogar consiste em:
 - Considerar todos os movimentos legais que podem ser realizados
 - Computar a nova posição resultante de cada movimento legal
 - Avaliar cada posição resultante, decidir e executar o melhor movimento
 - Esperar pelo movimento do oponente e repetir o processo
- Problemas a ser solucionados:
 - Representar todas as “posições” legais (estados)
 - Gerar essas “posições”
 - Avaliar a “posição”

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Função de utilidade

- A função de utilidade é usada para avaliar a qualidade de uma posição no jogo
 - Observar que é diferente da função de avaliação utilizada na busca informada (heurística) na qual a função de avaliação é uma estimativa não-negativa do que “acreditamos” (futuro) que ainda falta fazer para chegar ao estado meta
- Jogo zero-soma permite usar uma única função de utilidade para avaliar a qualidade de uma posição com relação a ambos jogadores
 - $f(n) >> 0$: posição n boa para mim e ruim para o oponente
 - $f(n) << 0$: posição n ruim para mim e boa para o oponente
 - $f(n)$ próximo de 0: posição n neutra
 - $f(n) = +\infty$: eu ganho
 - $f(n) = -\infty$: o oponente ganha

Função de utilidade

- Dá a utilidade de um estado
 - $utility(\text{Estado})$
- Exemplos da função *utility*:
 - -1, 0, e +1, para Jogador 1 perde, empata e ganha, respectivamente.
 - Diferença entre o total de pontos dos dois jogadores.
 - Somas ponderadas de fatores (ex. xadrez)
 - $utility(S) = w_1 f_1(S) + w_2 f_2(S) + \dots + w_n f_n(S)$
 - $f_1(S) = (\text{No. de rainhas Br}) - (\text{No. de rainhas Pr})$, $w_1 = 9$
 - $f_2(S) = (\text{No. de torres Br}) - (\text{No. de torres Pr})$, $w_2 = 5$
 - ...

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- **Método de busca minimax**
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Minimax

- 1944: John von Neumann propõe um método de busca (**Minimax**) para jogos de soma zero que maximiza a sua posição enquanto minimiza a de seu oponente.
- Para implementar esse método necessitamos de medir, de alguma maneira, o quanto é boa a nossa posição.
- Usamos para isso a **função de utilidade** (*utility function*).
- Inicialmente, será um valor que descreve exatamente a nossa posição.

Minimax

- Dois Agentes:

- MAX

- Tem como objetivo **maximizar** o resultado da avaliação da função de utilidade.
 - Estratégia para ganhar se, quando MIN joga, MAX ganha para *todos* os movimentos que MIN pode realizar.

- MIN

- Tem como objetivo **minimizar** o resultado da avaliação da função de utilidade.
 - Estratégia para ganhar se, quando MAX joga, MIN ganha para *todos* os movimentos que MAX pode realizar.

Minimax

- Algoritmo Minimax (informal)
 - ① Gerar a árvore do jogo
 - ② Calcular a função de utilidade de cada estado terminal
 - ③ Propagar a utilidade dos nós terminais para níveis superiores:
 - se no nível superior é a vez de MIN jogar, escolher o menor valor
 - se no nível superior é a vez de MAX jogar, escolher o maior valor
 - ④ No nó raiz, MAX escolhe o movimento que leva ao maior valor (decisão Minimax)

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

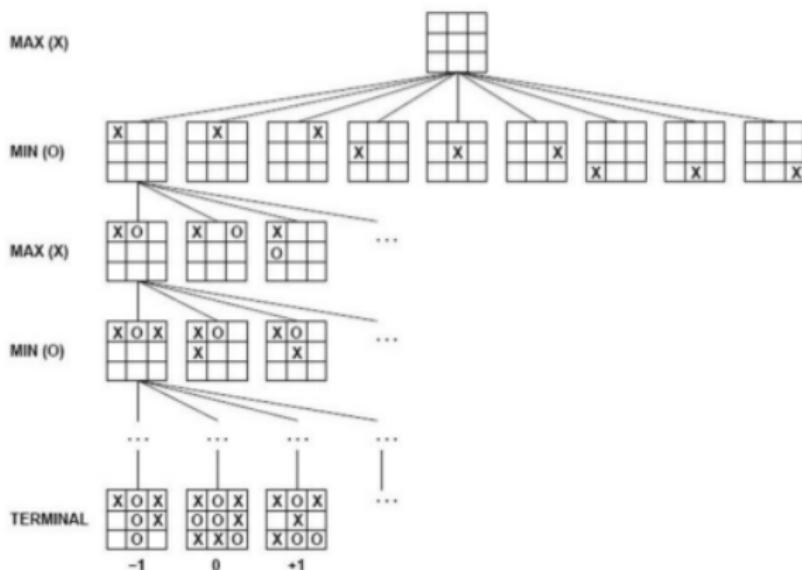
- Método de busca minimax
- **Aplicações**
- Poda da árvore

3 *Game IA*

- Histórico
- Técnicas de *game IA*

Minimax: jogo da velha

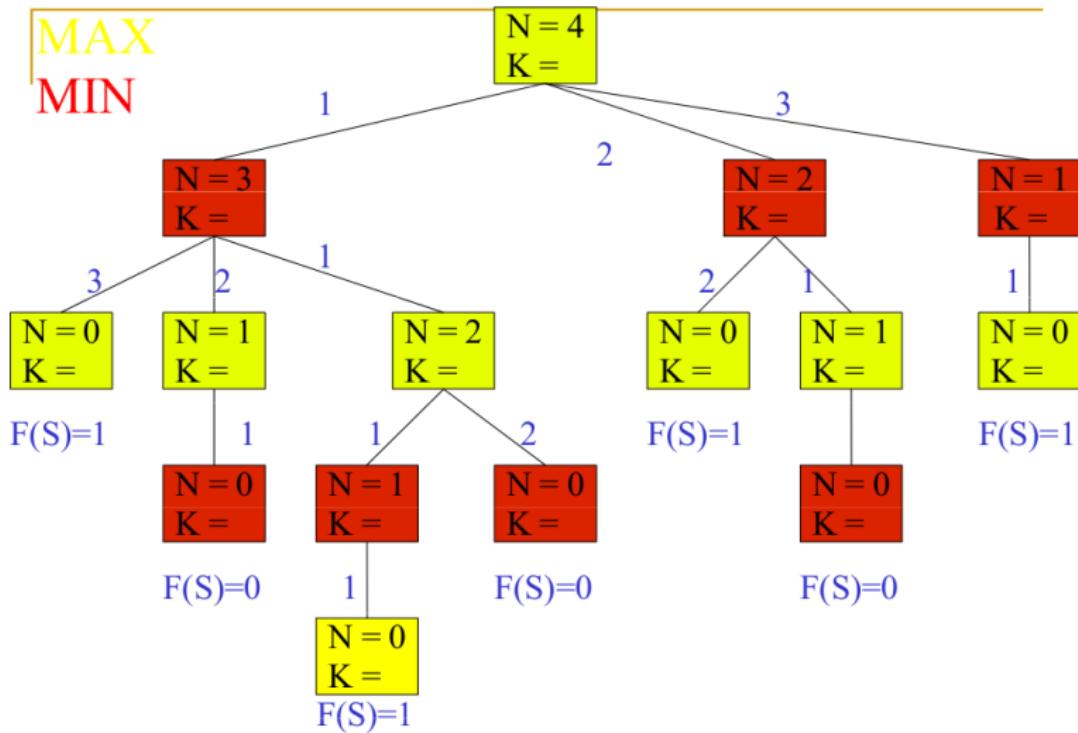
- Árvore minimax para o jogo da velha



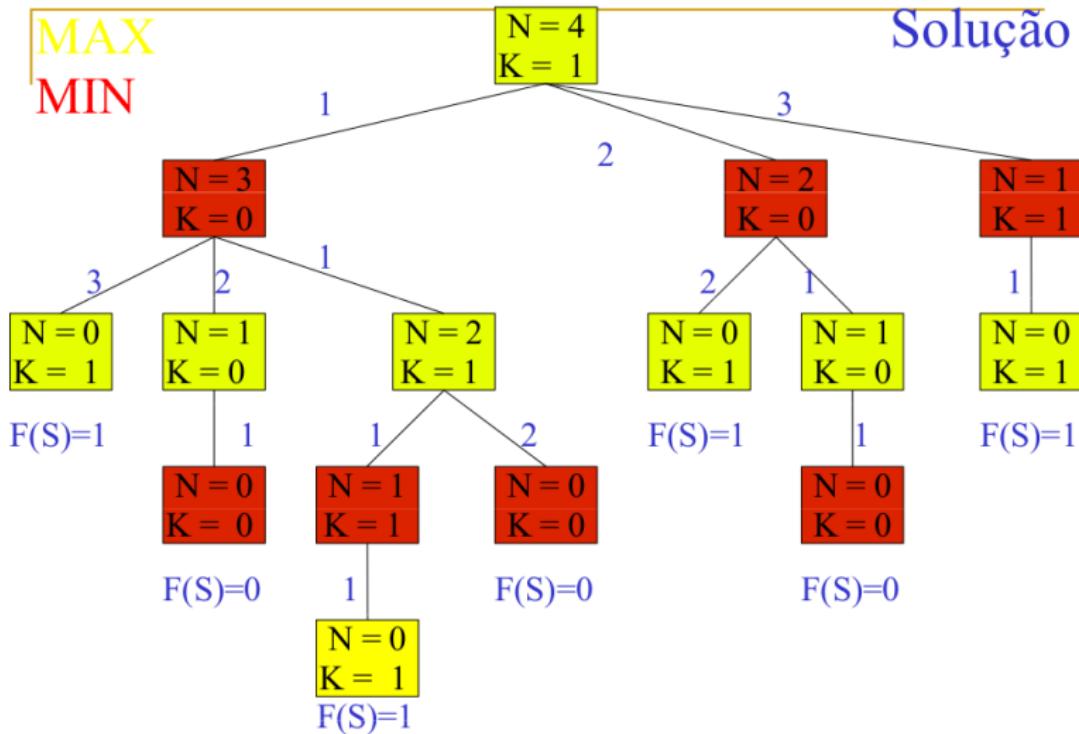
Minimax: jogo das moedas

- Outro exemplo: Jogo das moedas
 - Há uma pilha com N moedas
 - Um por vez, cada jogador retira 1, 2, ou 3 moedas da pilha
 - O jogador que retira a última moeda perde
- Definição formal:
 - **Estado inicial:** número de moedas na pilha
 - **Operadores:**
 - ① Remove uma moeda
 - ② Remove duas moedas
 - ③ Remove três moedas
 - **Fim do jogo:** pilha vazia
 - **Função de utilidade:** $F(S)$
 - $F(S) = 1$ se MAX ganha, 0 se MIN ganha

Minimax: jogo das moedas



Minimax: jogo das moedas

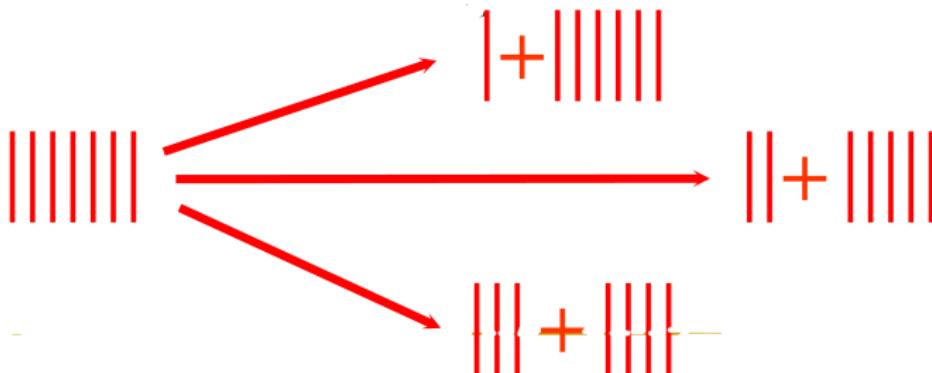


Minimax: jogo das moedas - Análise

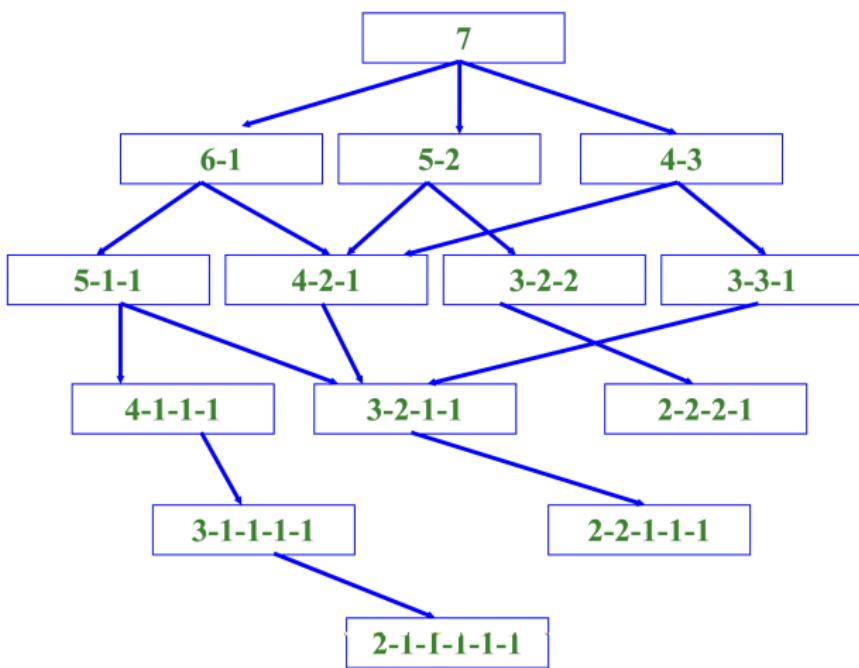
- Profundidade máxima: 5
- Grau da árvore: 3
- Número de nós: 15
- Ainda neste exemplo trivial é possível observar que essa árvore pode ser muito grande
 - Geralmente, a busca envolve $\mathcal{O}(b^d)$ nós
 - b = grau: número máximo de movimentos para cada nó
 - d = profundidade: altura da árvore
 - Tempo exponencial para executar o algoritmo!
 - Como podemos melhorar esse tempo?

Minimax: jogo Nim

- Uma pilha de fichas é colocada sobre uma mesa entre dois oponentes
- A cada movimento, um jogador deve dividir a pilha de fichas em duas pilhas não vazias de tamanhos diferentes
- O primeiro jogador que não puder mais realizar um movimento perde o jogo



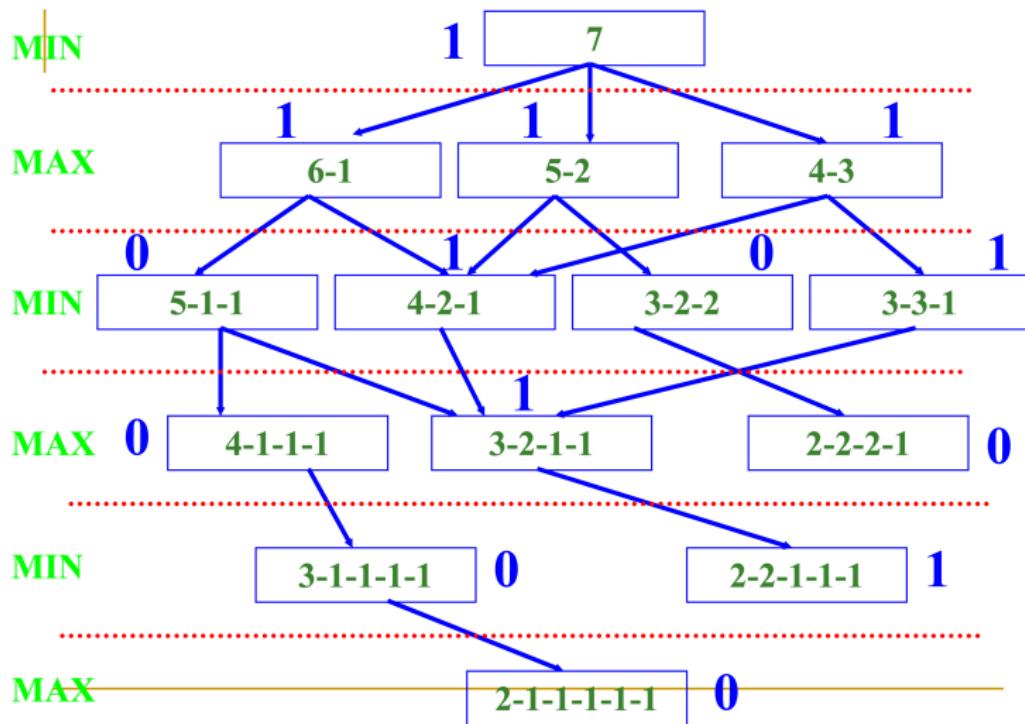
Minimax: jogo Nim



Minimax: jogo Nim

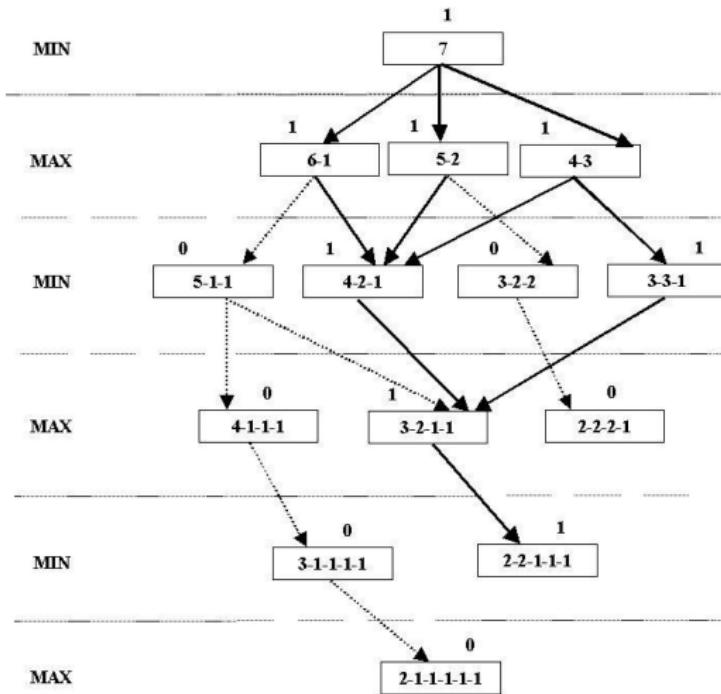
- Considerando que MIN joga primeiro, completar a árvore minimax.
- Lembrar que
 - jogador MIN quer minimizar a chance de vitória do oponente
 - jogador MAX quer maximizar sua chance de vitória
- Considerar a seguinte função de utilidade
 - 0 = ganha MIN
 - 1 = ganha MAX

Minimax: jogo Nim



Minimax: jogo Nim

Figure 1:
 Se repas-
 sarmos
 adequadamente as
 decisões
 dos estados
 finais aos
 seus
 ancestrais,
 a cada
 jogada se
 sabe qual o
 melhor
 caminho a
 se seguir



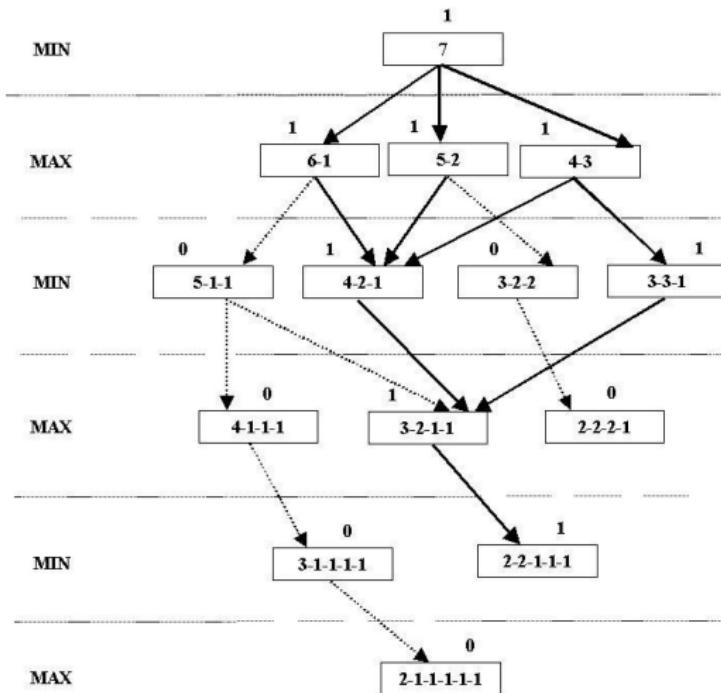
Minimax: jogo Nim

- Raciocínio dos jogadores

- O jogador MIN sempre vai seguir o caminho que dê menos vantagem para o MAX... no caso, escolherá os estados marcados com 0, se houver algum
- O MAX sempre vai seguir o caminho que lhe dê mais vantagem... no caso, escolherá os estados marcados com 1, se houver algum

Minimax: jogo Nim

Figure 2:
 Note que o jogador 2 (MAX) sempre ganha, supondo-se boas decisões (indicadas pelas setas mais escuras)



Minimax, de novo

- **Passos:**

- Gera a árvore inteira até os estados terminais
- Aplica a função de utilidade nas folhas
- Propaga os valores subindo a árvore através do minimax
- Determinar qual o valor que será escolhido por MAX

Minimax, de novo

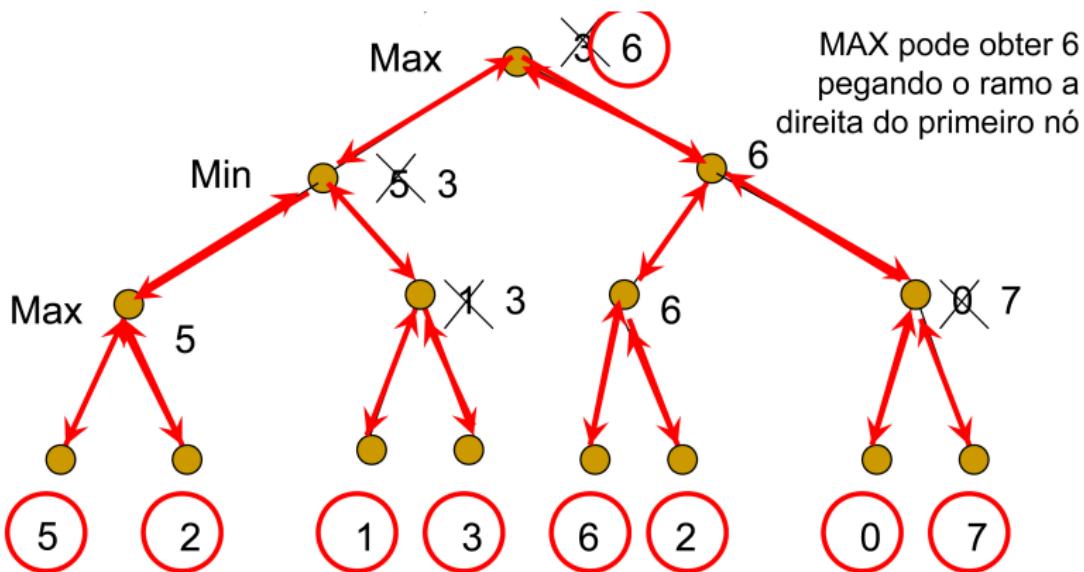


Figure 3: Veja que árvore de busca é construída em profundidade: MAX pode obter 6 pegando o ramo a direita do primeiro nó

Busca para jogos

- Em espaços de estados suficientemente pequenos, a estratégia anterior é muito boa
- Para espaços grandes e/ou em que há recursos limitados, temos que limitar a exploração realizada no espaço
 - Antecipação de n níveis

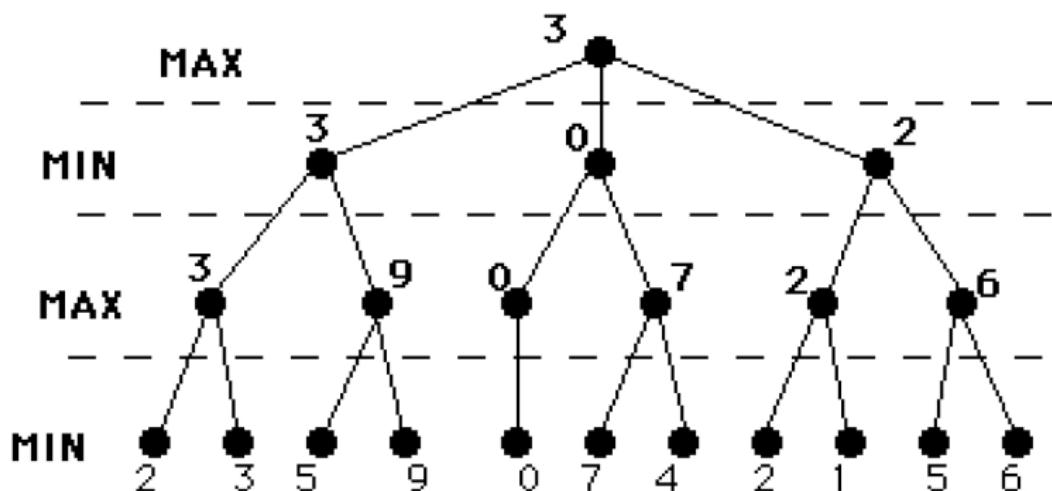
Minimax para níveis antecipados

- **Algoritmo:**

- Determina-se o número n de níveis (*ply*) a serem investigados
- Para cada jogada
 - Geram-se os n níveis
 - Como não se garante que há estados finais e, portanto, não há certeza de vitória ou derrota, usa-se uma função de avaliação, que retorna um valor heurístico para cada estado
 - Repassam-se os valores do último nível aos seus ancestrais
 - Decide-se que jogada fazer com base nos valores heurísticos

Minimax para níveis antecipados

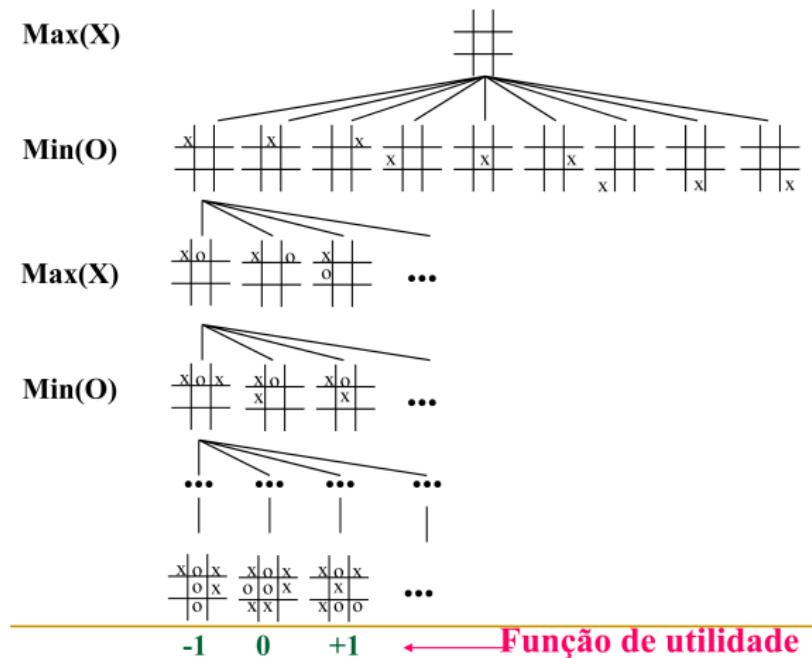
- Exemplo de um espaço de estados hipotético, com antecipação de 4 níveis/camadas:



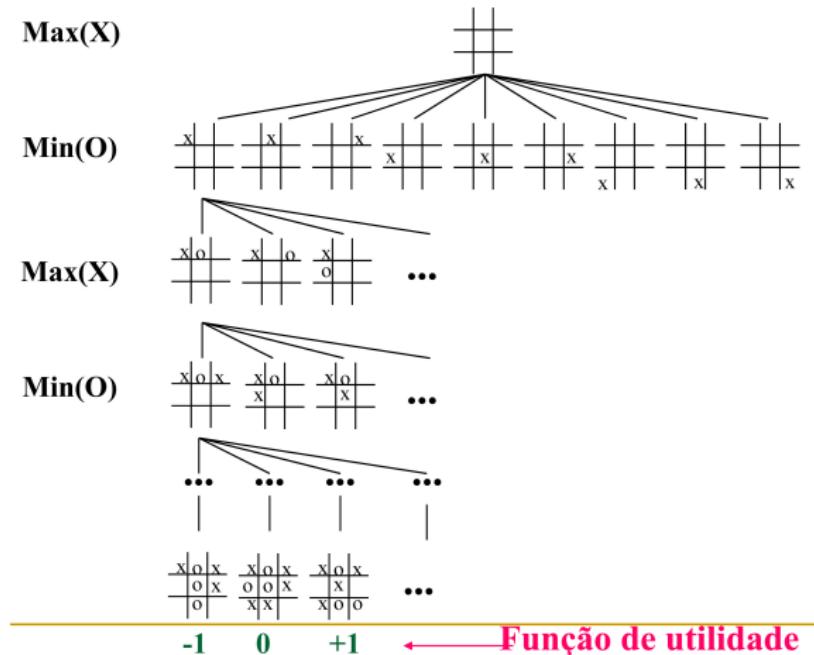
Minimax para níveis antecipados

- **Como definir a heurística?**
- Muitas possibilidades:
 - Em geral, medem a vantagem de um jogador sobre o outro
 - Em damas e xadrez, a vantagem de peças é importante: heurística que calcula a diferença do número de peças no tabuleiro
 - Em alguns casos, como xadrez, as peças têm importâncias diferentes e poderiam ser ponderadas de forma diferente na heurística (por exemplo, rainha > peão)
- Qual seria uma boa heurística para o jogo da velha?

Jogo da velha: minimax

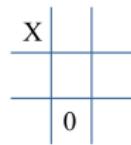


Jogo da velha: minimax

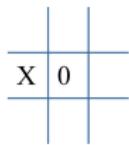


Precisa ser resolvido com antecipação de níveis?

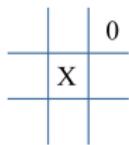
Jogo da velha: minimax



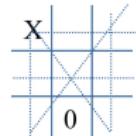
$$h = 6 - 5 = 1$$



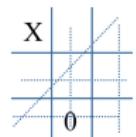
$$h = 4 - 6 = -2$$



$$h = 5 - 4 = 1$$

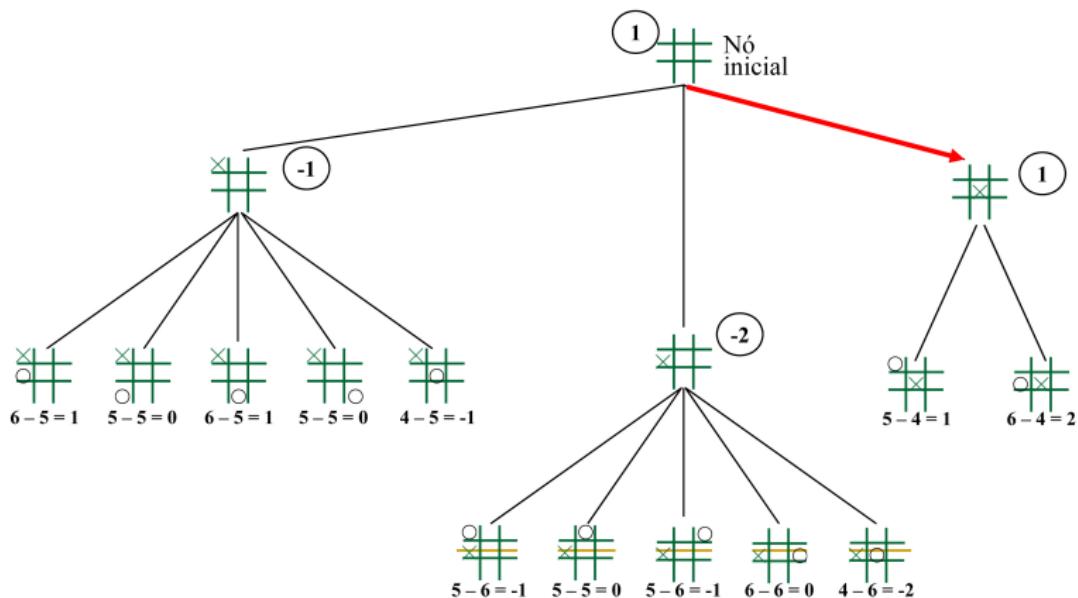


X tem 6 possibilidades



0 tem 5 possibilidades

Jogo da velha: minimax



Minimax para níveis antecipados

● Problema dessa técnica?

- Os níveis antecipados podem enganar, pois um caminho aparentemente promissor pode levar a derrota mais tarde
 - Bons jogadores de xadrez sabem usar isso a seu favor
- Efeito de horizonte

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

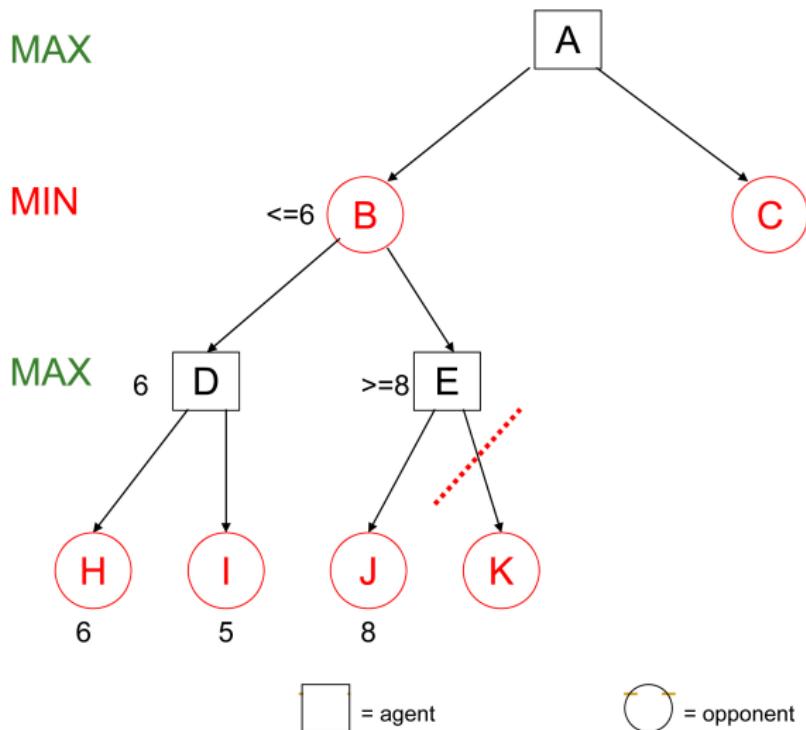
3 Game IA

- Histórico
- Técnicas de game IA

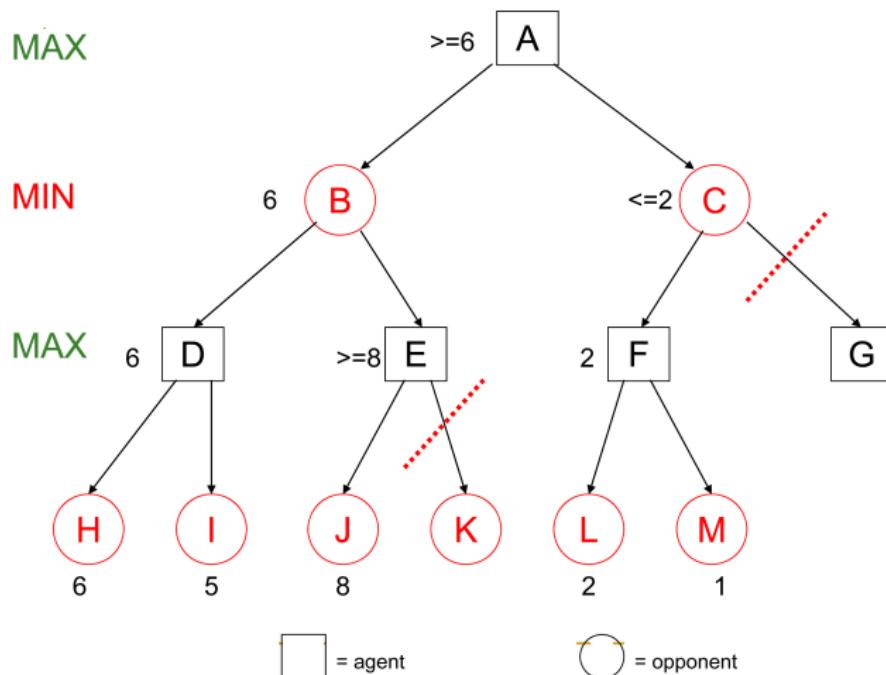
Poda da árvore de busca

- O minimax é feito em 2 passos
 - Descida em profundidade e aplicação da heurística
 - Retropropagação dos valores
- Alguns ramos não precisam ser analisados, podendo ser **podados**.

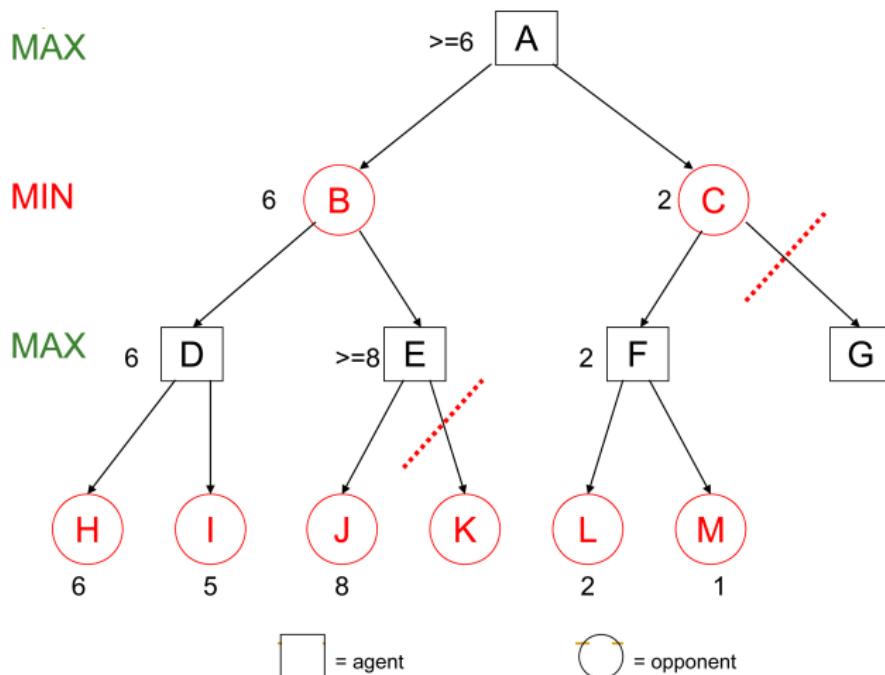
Poda da árvore de busca



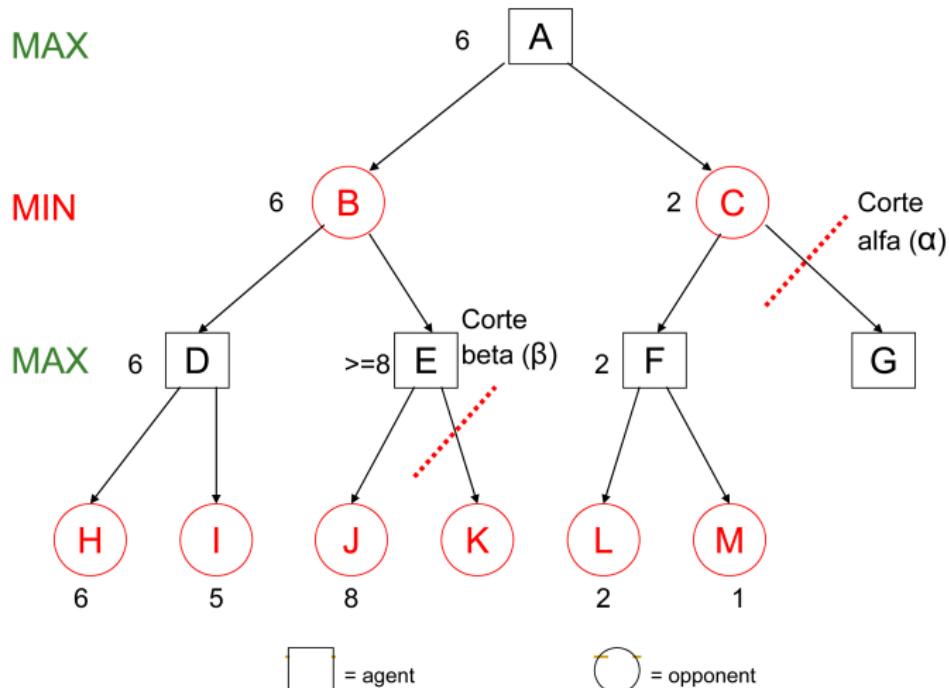
Poda da árvore de busca



Poda da árvore de busca



Poda da árvore de busca: poda alfa-beta



Poda alfa-beta

- A esse tipo de poda se deu o nome de **poda alfa-beta**:
 - O valor α está associado aos nós MAX e não pode decrescer
 - O valor β está associado aos nós MIN e não pode aumentar
- *Idéia principal:* não processar sub-árvores que não afetam o resultado.
- Dois novos parametros:
 - α : o melhor valor de MAX encontrado até agora
 - β : o melhor valor de MIN encontrado até agora
- α é usado nos nós MIN e é atribuído nos nós MAX
- β é usado nos nós MAX e é atribuído nos nós MIN

Poda alfa-beta

- MAX (não no nível 0)
 - Se for encontrada uma sub-árvore com um valor $k > \beta$, então não é necessário continuar a busca nessa sub-árvore
 - MAX pode ir tão bem quanto k nesse nó, portanto MIN nunca vai escolher para chegar aí!
- MIN
 - Se for encontrada uma sub-árvore com um valor $k < \alpha$, então não é necessário continuar a busca nessa sub-árvore
 - MIN pode ir tão bem quanto k nesse nó, portanto MAX nunca vai escolher para chegar aí!

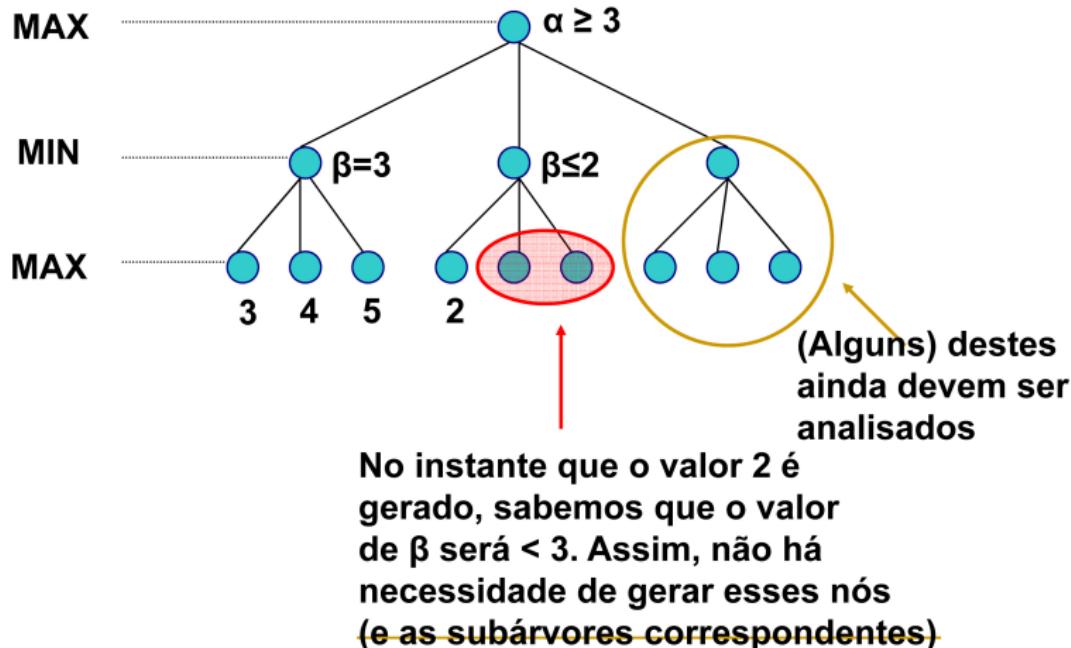
Poda alfa-beta

- Visite a árvore de busca em profundidade
 - Para cada nó n MAX, $\alpha(n) = \text{máximo valor até agora}$
 - Para cada nó n MIN, $\beta(n) = \text{mínimo valor até agora}$
- **Obs:** o valor de α começa em $-\infty$ e somente aumenta, enquanto que o valor de β começa em $+\infty$ e somente diminui.
- **Corte β :** dado um nó n , corte a busca após n (i.e., não gere ou examine os filhos de n se $\alpha(n) \geq \beta(i)$ para algum nó i MIN ancestral de n).
- **Corte α :** corte a busca abaixo de um nó n MIN se $\beta(n) \leq \alpha(i)$ para algum nó i MAX ancestral de n .

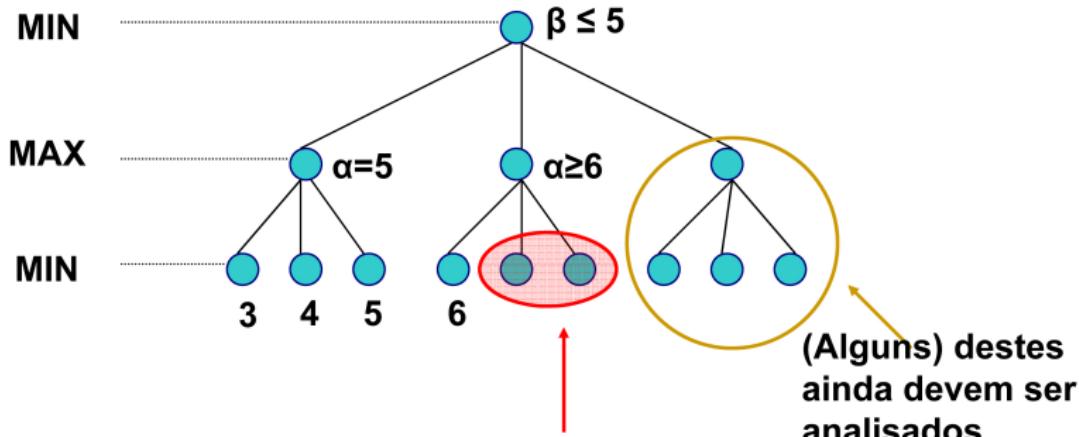
Poda alfa-beta

- **Regras para interromper a busca:**
 - A busca pode ser interrompida abaixo de qualquer nó MIN que tenha um valor β menor ou igual ao valor α de qualquer um de seus ancestrais MAX
 - A busca pode ser interrompida abaixo de qualquer nó MAX que tenha um valor α maior ou igual ao valor β de qualquer um de seus ancestrais MIN
- **Atenção:** a poda alfa-beta expressa uma relação entre nós no nível n e $n+2$, sob o qual podem ser eliminadas sub-árvoreas inteiras com raízes no nível $n+1$

Exemplo com MAX



Exemplo com MIN



No instante que o valor 6 é gerado, sabemos que o valor $\alpha \geq 6$. Assim, não há necessidade de gerar esses nós (e as subárvores correspondentes)

Jogos modernos

- Jogos podem ser **bem mais complicados**
- Pode haver $n > 2$ jogadores
- Pode haver sorte envolvida (por exemplo, jogo com dados)
- Pode envolver comportamento e inteligência distribuída
- Pode se adaptar ao usuário

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 Game IA

- Histórico
- Técnicas de game IA

Histórico

- Até as décadas de 1960 e 1970, os jogos eletrônicos não utilizavam técnicas de IA
- A indústria percebeu que a inclusão dessas técnicas poderia atrair um público maior, aumentando, assim, os lucros
- Havia também a necessidade da inclusão de elementos virtuais que imitassem o comportamento humano, para que os jogadores pudessem jogar sozinhos
- No meio acadêmico, já havia muitas técnicas capazes de oferecer a entidades virtuais características de autonomia e raciocínio, potencialmente úteis aos jogos
- Em 1974, com os jogos *Pursuit* e *Qwak*, os jogadores tinham que atirar em alvos móveis
- Em 1978, o jogo *Space Invaders* implantou as primeiras entidades inteligentes em jogos

Histórico

- Em 1980, *Pac-man* conta com movimentos padronizados dos inimigos, porém cada fantasma tem um modo diferente de caçar o jogador
- Em 1990, O primeiro jogo de estratégia em tempo real, *Herzog Wei*, é lançado. A busca de caminho apresentada nesse jogo era de baixa qualidade
- Em 1993, *Doom* é lançado como primeiro jogo de tiro em primeira pessoa
- Em 1996, *BattleCruiser: 3000AD* é o primeiro jogo a utilizar redes neurais comercialmente
- Em 1998, *Half-Life* é lançado como a melhor Game IA até o momento
- Em 2001, o jogo *Black & White* é alvo da mídia a respeito de como as criaturas aprendem com as decisões do jogador

Sumário

1 Busca com adversários

- Introdução
- Função de utilidade

2 Minimax

- Método de busca minimax
- Aplicações
- Poda da árvore

3 *Game IA*

- Histórico
- Técnicas de *game IA*

IA acadêmica versus game IA

- O termo **game IA** surgiu para diferenciar os estudos em IA para jogos eletrônicos dos elaborados na academia
- A principal diferença entre a IA acadêmica e a *game IA* é que a primeira tem por objetivo a solução de problemas difíceis, como reconhecimento de padrões, enquanto a segunda tem por objetivo a diversão dos jogadores, seja pelo aumento do grau de verossimilhança dos jogos, ou pelo nível de desafio apresentado. Para isso, são utilizadas algumas das soluções pesquisadas e encontradas no meio acadêmico.
- Técnicas de *game IA*:
 - Algoritmos determinísticos e padrões de movimentos
 - Máquinas de estados
 - Sistemas baseados em regras
 - Algoritmos de busca
 - Algoritmos genéticos

Algoritmos determinísticos e padrões de movimentos

- Os algoritmos determinísticos e padrões de movimento foram utilizados nos primeiros jogos eletrônicos da história:
 - Movimentos aleatórios
 - Algoritmos de perseguição e evasão
- Exemplo: *Gun Fight* (1975) - personagens com movimentos aleatórios.

Máquinas de estados

- São utilizadas para determinar os estados de um personagem e suas mudanças
- Recurso de fácil entendimento, implementação e depuração
- Utilização de lógica fuzzy para resultados de ações menos previsíveis
- Exemplo: no *pac-man*, uma máquina de estados é utilizada para cada fantasma, sendo os estados possíveis: procurando o jogador, perseguindo o jogador e fugindo do jogador

Sistemas baseados em regras

- Alguns fenômenos não são fáceis de serem modelados em termos de estados e transições
- Para a modelagem de comportamento global, a utilização de um sistema de regras se faz mais útil que um sistema baseado apenas em máquinas de estados

Algoritmos de busca

- Um dos problemas básicos de *game IA*
- A*, Dijkstra, *waypoints*, ...
- Utilização de caminhos pré-calculados para a redução dos custos da busca

Algoritmos genéticos

- Técnicas de algoritmos genéticos são utilizadas para que agentes “evoluam” baseado em seus desempenhos
- Os agentes podem aprender elementos como a estratégia do usuário, assim como mapear perfis de usuários
- Técnicas bastante custosas

Considerações finais

- Os jogos ilustram vários pontos importantes da IA
- Muita coisa mudou ... E ainda vai mudar!
- Segundo Russell [1],
os jogos estão para a IA assim como as corridas estão para os projetos de automóveis.

Referências I

- [1] Russell, S., Norvig, P.
Inteligência Artificial. 2a. Edição.
Editora Campus, 2004.