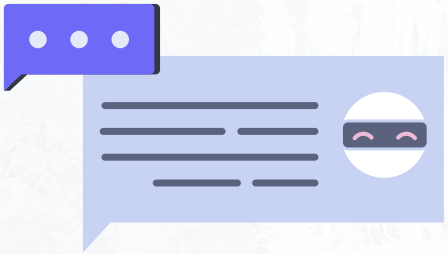
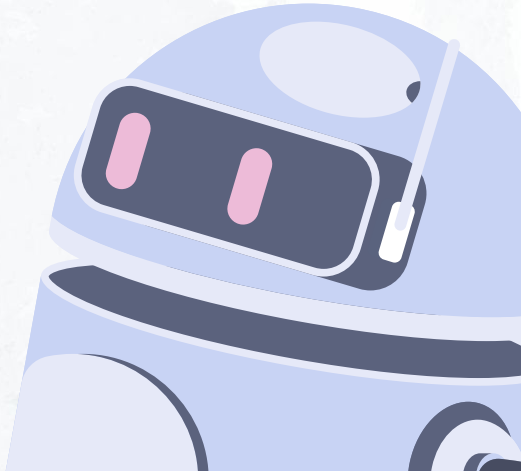
Introdução à Inteligência Artificial

João Paulo Aires

(IA)





Objetivos

- Busca não-determinística
- Busca Adversária

Recap

Busca

- Cobrimos uma série de estratégias de busca que operam gerando sistematicamente novos estados e testando-os em relação a um objetivo
- Geralmente são altamente ineficientes:
  - Além do espaço de estados e do custo, eles não levam em consideração nenhum conhecimento do problema
  - Eles representam **pesquisa desinformada**
- Ao usar conhecimento específico do problema (ou seja, uma **pesquisa informada**), podemos ter um desempenho melhor do que os algoritmos já encontrados

Recap

Busca Informada

- Ideia: selecione o nó para expansão com base em uma função de avaliação, *f(n)*
  - estimativa de “desejabilidade”
  - expandir o nó não expandido mais desejável
- Esta função pode medir a distância do objetivo, então expanda o nó com avaliação mais baixa
- Normalmente implementado por meio de uma fila de prioridade
- Nossa busca é do melhor primeiro; expandimos o nó que (acreditamos) é o melhor primeiro

Recap

A*

- A busca A∗ é provavelmente o tipo de busca heurística mais utilizado
- Combina o custo para chegar a um nó (g(n)) com o custo para ir do nó até o objetivo (h(n))
- f(n) = g(n) + h(n)
- f(n) é o custo estimado da solução mais barata através de n
- Como estamos tentando encontrar a solução mais barata, faz sentido tentar primeiro o nó com o f(n) mais baixo
- • Sob certas condições de h(n), a pesquisa A∗ é completa e ótima

Recap

Busca Local

- Certas classes de problemas não se preocupam com o caminho para uma solução ou com seus estados intermediários
- Chamamos esses métodos de busca local: eles consistem em otimizar uma função objetivo
- Classes inteiras de algoritmos concentram-se em movimentos estocásticos em direção a máximos locais/globais
- (Estocástico) Hill Climbing / Gradient Descent (Ascent)
- Algoritmos genéticos

Índice

01 →

Busca Não-Determinísitca

(IA)

Busca Não-Determinística

Ações Não-Determinísticas

- Vimos algoritmos de busca para ambientes que são determinísticos e totalmente observáveis
- Quando uma ou ambas as suposições não podem ser feitas, um agente precisa levar em conta as percepções em cada movimento para restringir os estados onde está
- Uma solução não pode mais ser uma sequência plana de ações, mas sim um **plano de contingência** (também conhecido como **estratégia**)

Busca Não-Determinística

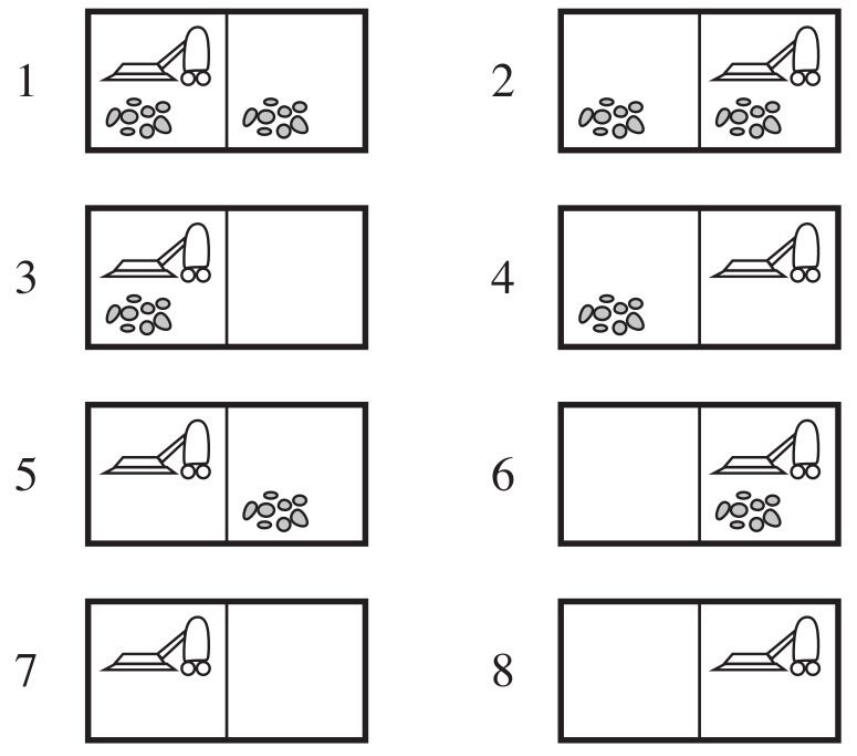
Vacuum World Errático

- Lembrem-se do Vacuum World, mas com uma ação de aspirar não determinística:
  - Quando aplicada a um quadro sujo, a ação limpa o quadro atual, mas às vezes também **limpa um quadro adjacente**
  - Quando aplicada a um quadro limpo, a ação às vezes **suja** o quadro
- Isto exige
  - **Modelo de transição** diferente
  - Conceito de **solução** diferente

Busca Não-Determinística

Transições Não-Determinísitcas

- **Resultado** de um estado agora se torna **Resultados**:
- Resultados (1, aspira) = {5, 7}
- Aspira aplicado ao estado 1 resulta no agente chegando nos estados 5 ou 7



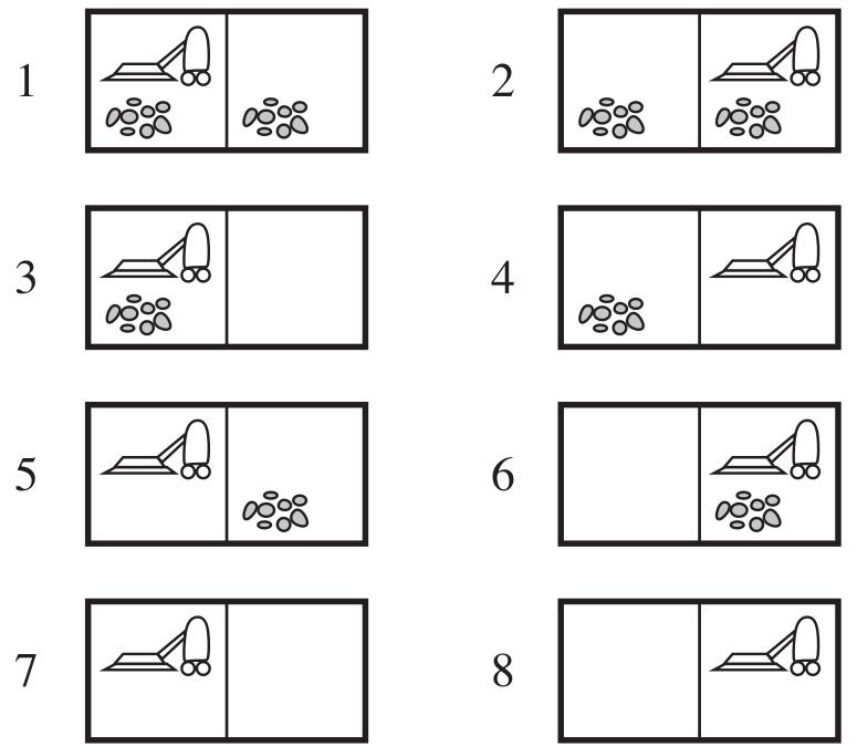
Busca Não-Determinística

Transições Não-Determinísitcas

- O plano é agora um plano de contingência (uma árvore):
- [**Aspirar**,

  se Estado = 5 então [Direita, Aspirar]

  senão []]



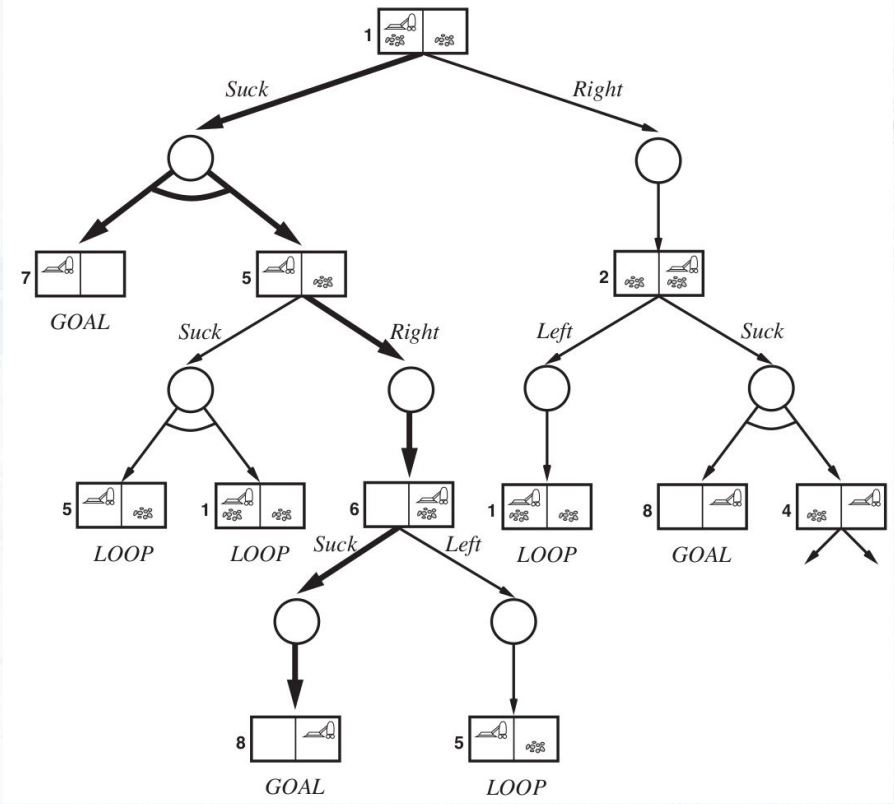
Busca Não-Determinística

Árvore de Busca AND-OR

- Em ambientes determinísticos, a ramificação ocorre apenas devido à escolha do agente (nós OR)
- Em ambientes não determinísticos, a escolha do ambiente também deve ser levada em consideração (AND Nodes)
- Solução é uma subárvore da árvore AND-OR que:
    - Tem um nó objetivo em cada folha
    - Especifica uma ação em cada nó OR
    - Inclui todas as ramificações de resultados de seus nós AND

Busca Não-Determinística

Árvore de Busca AND-OR



Busca Não-Determinística

Árvore de Busca AND-OR

- Em ambientes determinísticos, a ramificação ocorre apenas devido à escolha do agente (nós OR)
- Em ambientes não determinísticos, a escolha do ambiente também deve ser levada em consideração (AND Nodes)
- Solução é uma subárvore da árvore AND-OR que:
    - Tem um nó objetivo em cada folha
    - Especifica uma ação em cada nó OR
    - Inclui todas as ramificações de resultados de seus nós AND

02 →

Busca Adversária

(IA)

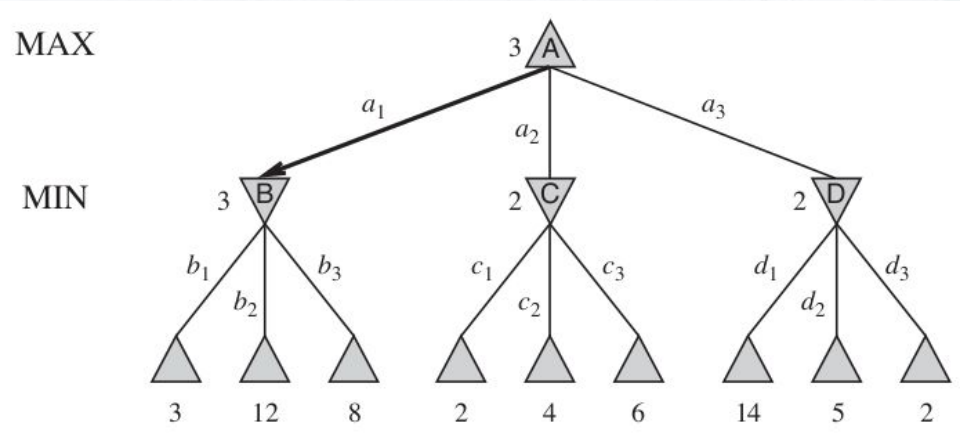
Busca Adversária

Decisões Adversárias Ótimas

- Cada movimento que nós MAX fazem tem uma resposta do MIN
- Então, o plano que procuramos é uma estratégia contingente com movimentos para:
  - O estado inicial; e
  - Todas as respostas possíveis do MIN
- Podemos calcular o valor do jogo recursivamente, assumindo que ambos os jogadores joguem de forma otimizada usando o valor **Minimax**

Busca Adversária

Decisões Adversárias Ótimas



$$\text{Minimax}(s) = \begin{cases} \text{Utilidade}(s) & \text{if EstadoFinal}(s) \\ \max_{a \in \text{Ações}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if Player}(s) = \text{Max} \\ \min_{a \in \text{Ações}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if Player}(s) = \text{Min} \end{cases}$$

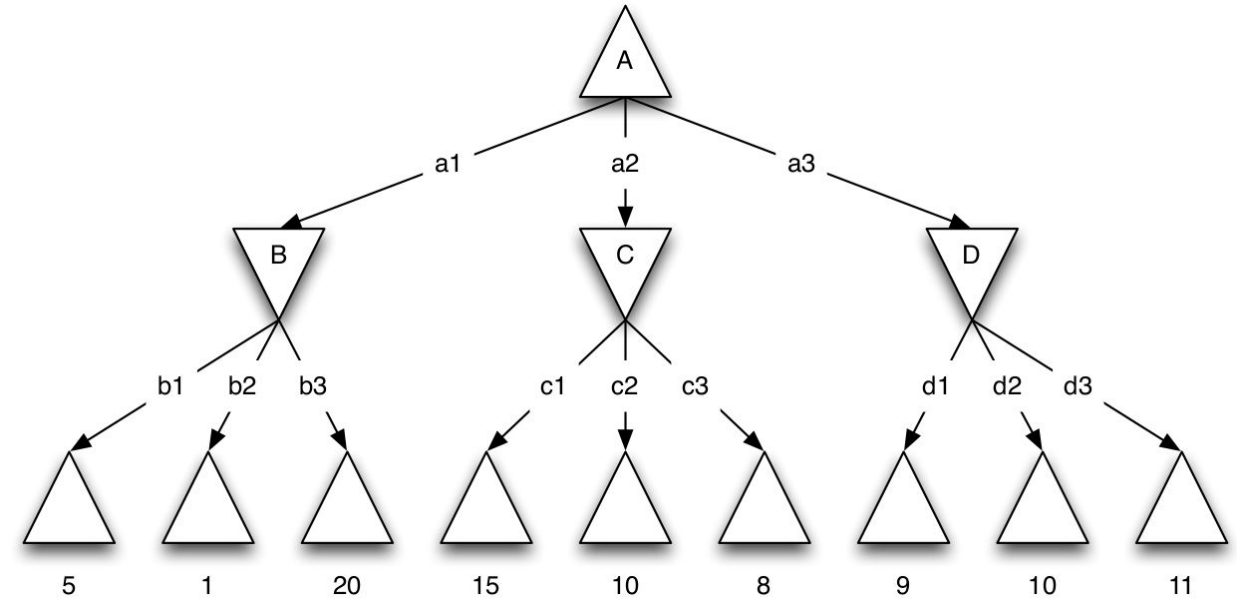
Busca Adversária

Decisões Adversárias Ótimas

```
1:  function Minimax-Decision(state) returns an action
2:      return arg max_{a∈Actions(s)} Min-Value(Result(state, a))
---
3:  function Max-Value(state) returns a utility value
4:      if Terminal-Test(state) then return Utility(state)
5:      v ← −∞
6:      for each a in Actions(state) do
7:          v ← Max(v, Min-Value(Result(state, a)))
8:      return v
---
9:  function Min-Value(state) returns a utility value
10:     if Terminal-Test(state) then return Utility(state)
11:     v ← ∞
12:     for each a in Actions(state) do
13:         v ← Min(v, Max-Value(Result(state, a)))
14:     return v
```

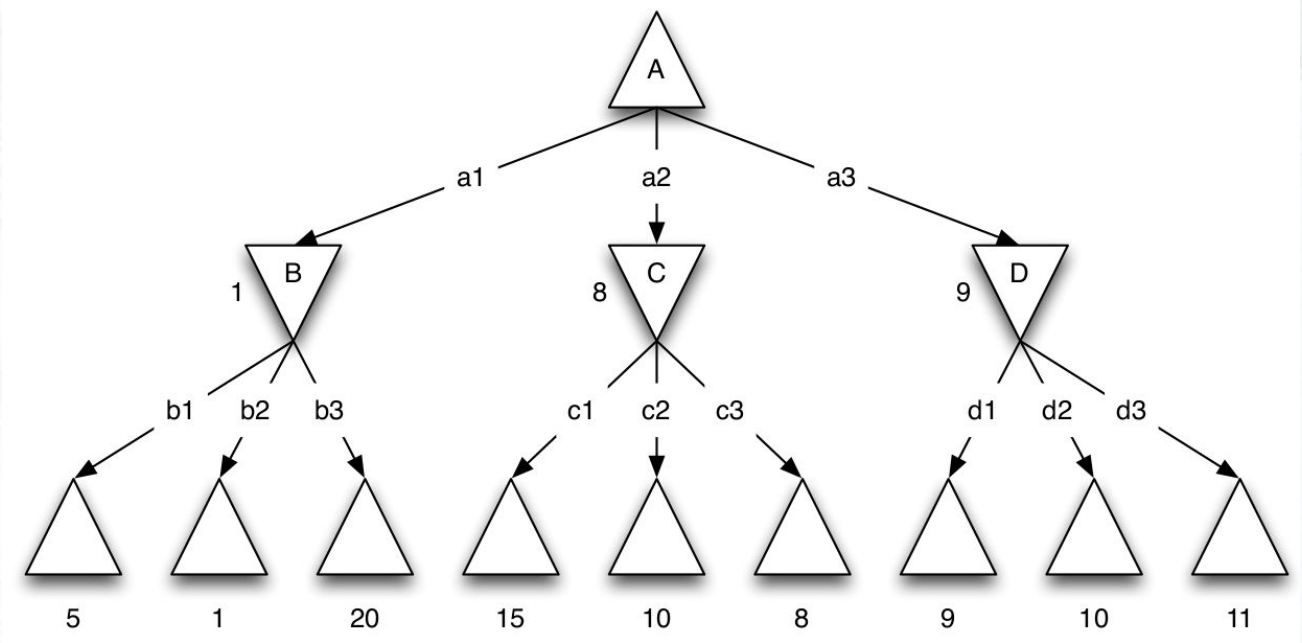
Busca Adversária

Decisões Adversárias Ótimas



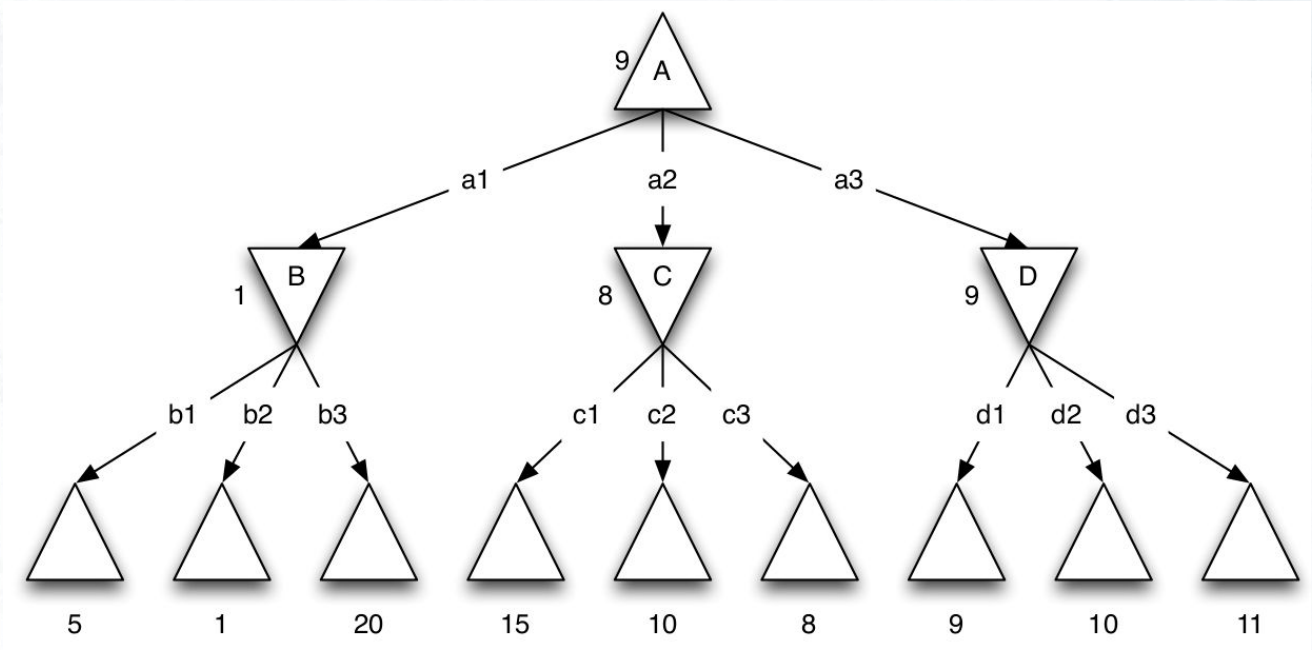
Busca Adversária

Decisões Adversárias Ótimas



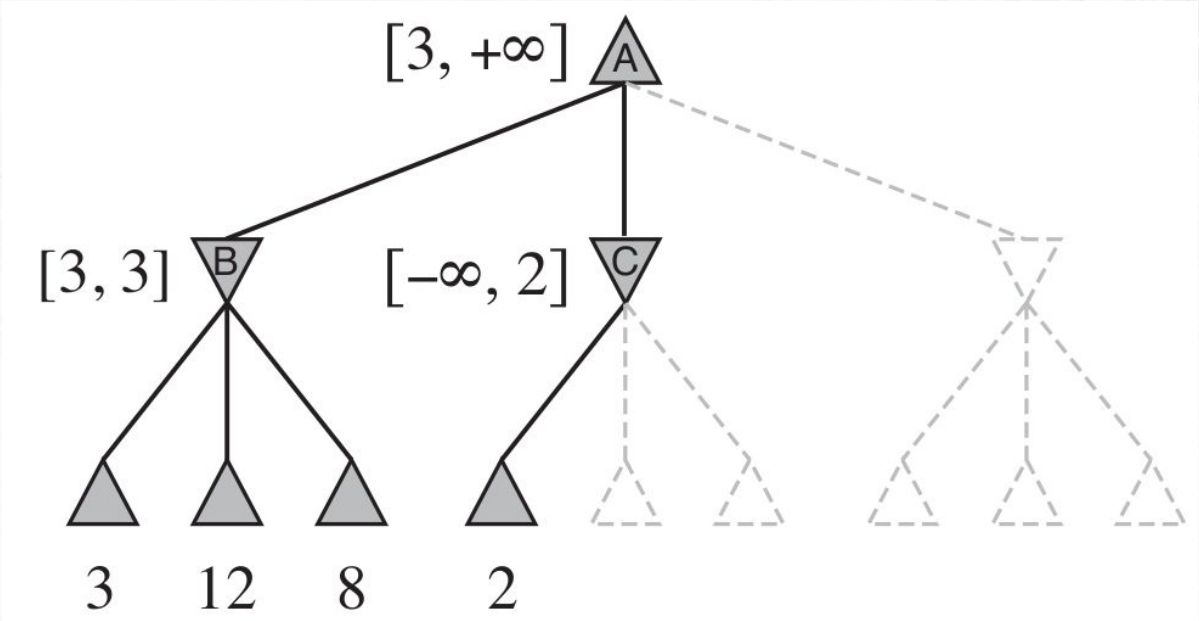
Busca Adversária

Decisões Adversárias Ótimas



Busca Adversária

Reduzir o Fator de Ramificação



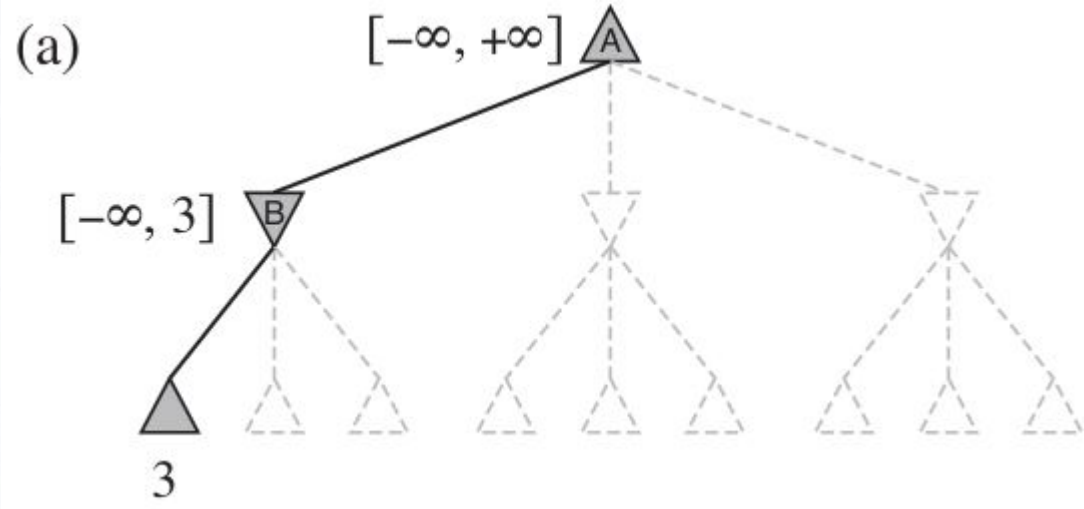
Busca Adversária

Reduzir o Fator de Ramificação

- Poda Alfa-Beta
  - Avaliar quais nós/ramos não afetariam a decisão de MIN/MAX
  - Baseado no acompanhamento de dois parâmetros:
    - α - valor da melhor (maior) escolha que temos no caminho de MAX
    - β - valor da melhor (menor) escolha que temos no caminho do MIN
- Atualiza esses valores à medida que avançamos na árvore

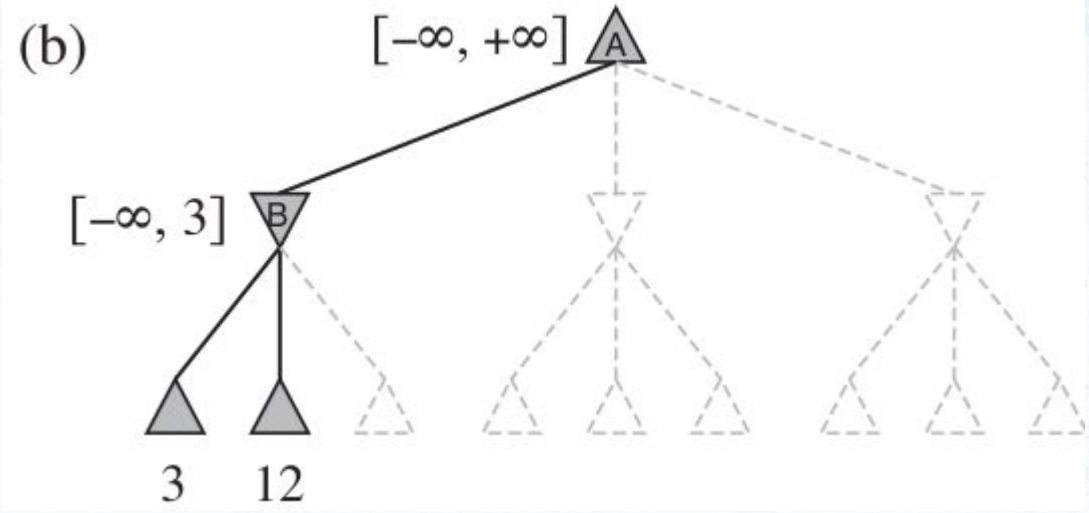
Busca Adversária

Reduzir o Fator de Ramificação



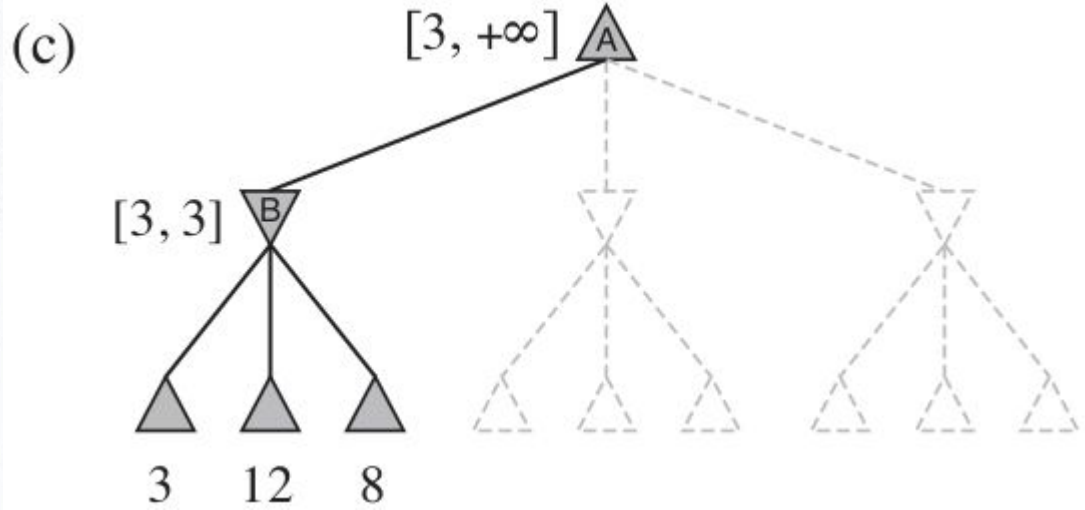
Busca Adversária

Reduzir o Fator de Ramificação



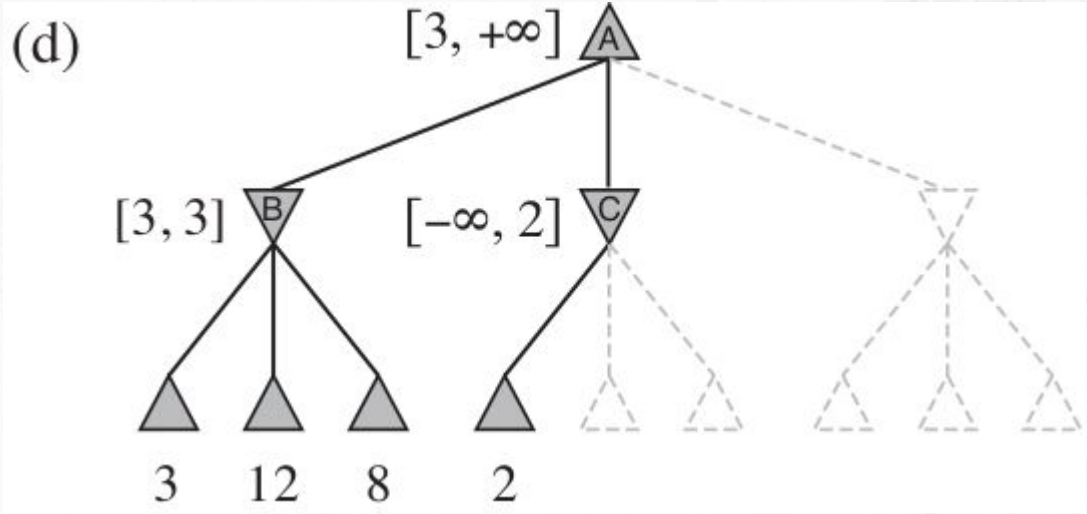
Busca Adversária

Reduzir o Fator de Ramificação



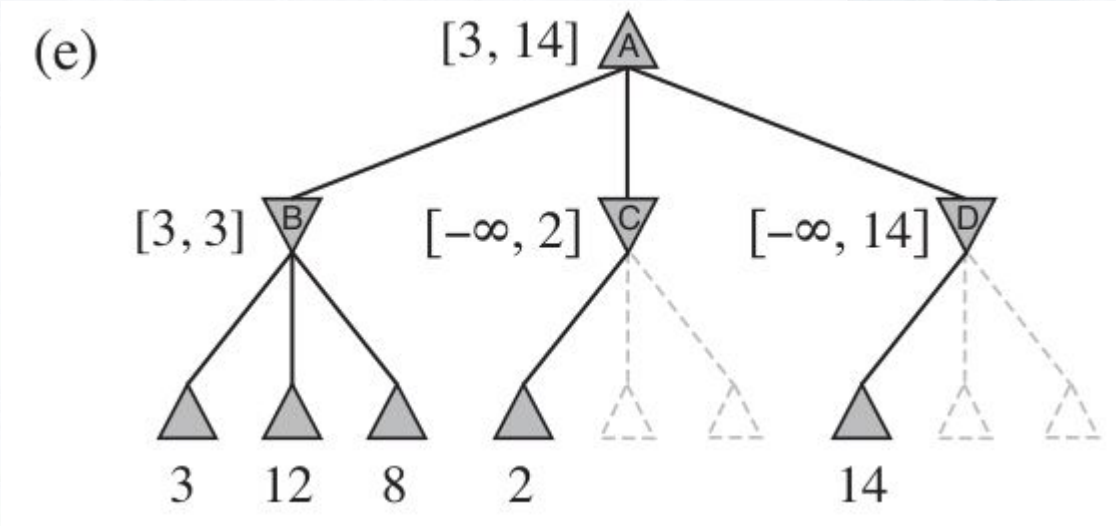
Busca Adversária

Reduzir o Fator de Ramificação



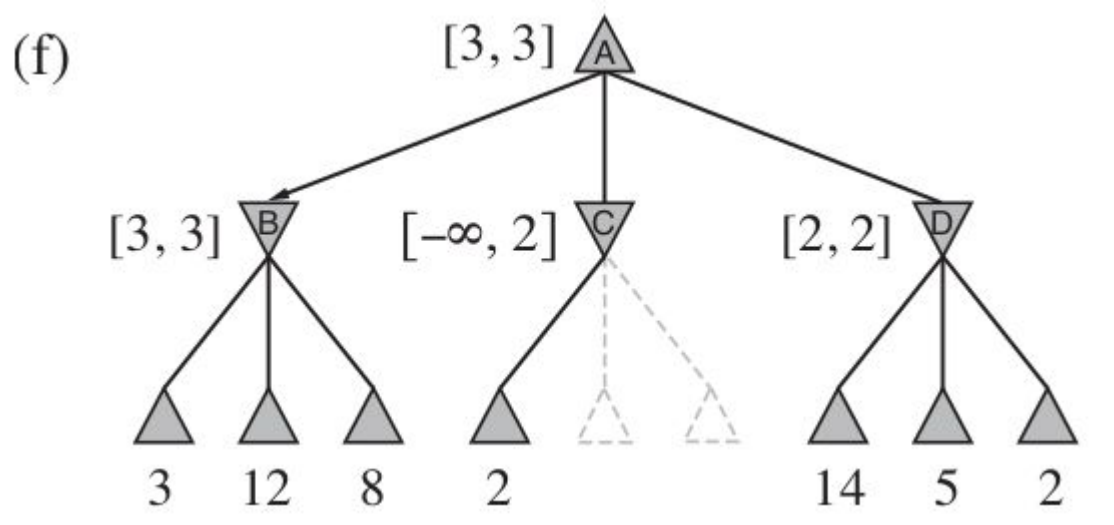
Busca Adversária

Reduzir o Fator de Ramificação



Busca Adversária

Reduzir o Fator de Ramificação



Busca Adversária

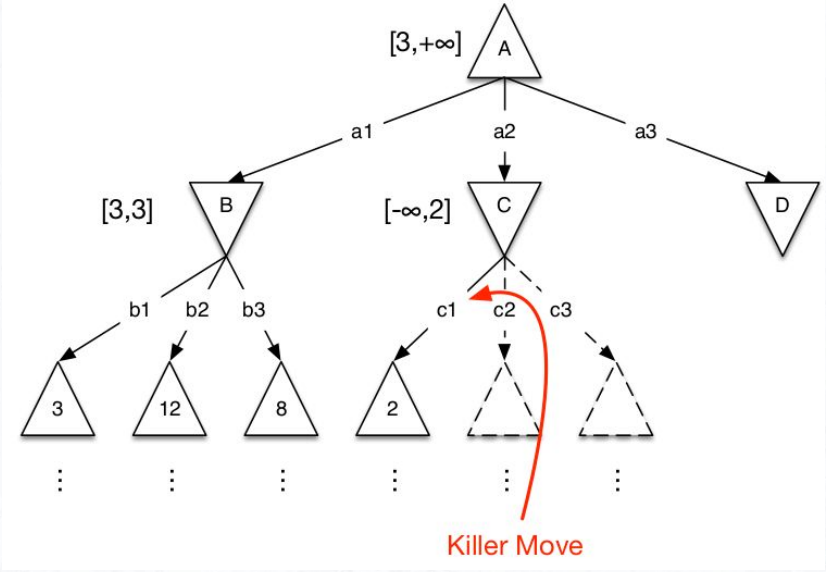
Busca Alpha-Beta: Ordenação dos Movimentos

- A poda é fortemente afetada pela ordem dos movimentos da árvore
- Uma boa ordenação* nos permitiria podar muitos nós
- A ordem dos movimentos geralmente depende do conhecimento do jogo (heurística)
- Ordenação dinâmica de movimentos (heurística de movimento assassino)

Busca Adversária

Busca Alpha-Beta: Ordenação dos Movimentos

- Heurística dinâmica para determinar uma ordem “boa”
- Pesquise duas camadas à frente até
  - Max (alt. Min) causa um corte beta (alt. alfa)
- O lance que causou o corte é o lance matador



Busca Adversária

Busca em Grafo

- Tal como na busca não adversária, muitos estados serão revisitados
- No entanto, apenas registrar os estados visitados não é suficiente (já que o MIN pode divergir no futuro)
- Necessidade de armazenar caminhos de loop reais (uso intenso de memória)
- Requer estratégia de “cache”