Pontifícia Universidade Católica
do Rio de Janeiro

INF 1771 – Inteligência Artificial

Aula 05 – Busca Local

2016.2

Prof. Augusto Baffa
<abaffa@inf.puc-rio.br>

Métodos de Busca

- **Busca Cega ou Exaustiva:**
  - Não sabe qual o melhor nó da fronteira a ser expandido. Apenas distingue o estado objetivo dos não objetivos.

- **Busca Heurística:**
  - Estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas.

- **Busca Local:**
  - **Operam em um único estado e movem-se para a vizinhança deste estado.**

Busca Local

- Em muitos problemas o **caminho para a solução é irrelevante**.
  - **Jogo das n-rainhas:** o que importa é a configuração final e não a ordem em que as rainhas foram acrescentadas.
  - **Outros exemplos:**
    - Projeto de Circuitos eletrônicos;
    - Layout de instalações industriais;
    - Escalonamento de salas de aula;
    - Otimização de redes;
- Se o caminho para a solução não importa, podemos utilizar um algoritmo de **busca local**.

Busca Local

- Algoritmos de busca local operam sobre um **único estado corrente**, ao invés de vários caminhos.

- Em geral se movem apenas para os **vizinhos** desse estado.

- O caminho seguido pelo algoritmo não é guardado.

Busca Local

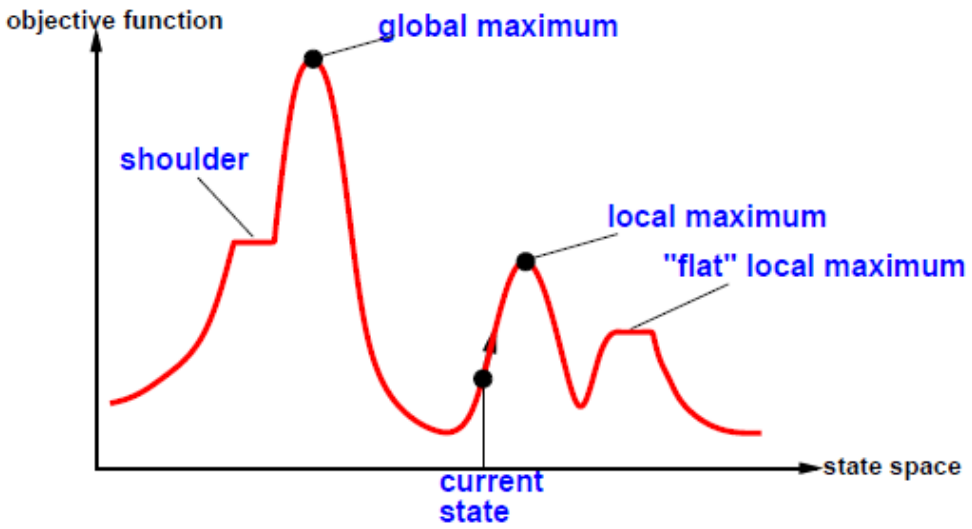
- **Vantagens:**
  - Ocupam pouquíssima memória (normalmente constante).
  - Podem encontrar soluções razoáveis em grandes ou infinitos espaços de estados.

- **São uteis para resolver problemas de otimização.**
  - Buscar por estados que atendam a uma função objetivo.

Busca Local

- **Panorama do Espaço de Estados**:

- **Local** = Estado;
- **Elevação** = Valor de custo da função heurística;
- Busca-se o máximo ou mínimo global;



Busca Local

- **Principais Algoritmos:**
  - Hill Climbing
  - Simulated Annealing
  - Local Beam Search
  - Algoritmos Genéticos
- **Outros Algoritmos:**
  - Iterated Local Search
  - Colônia de formigas

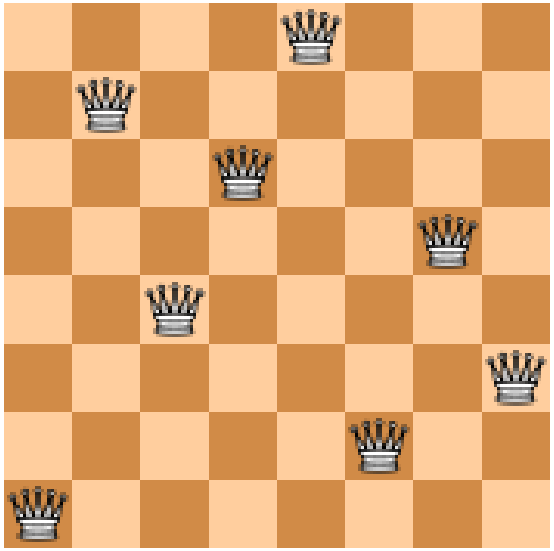
Pseudocódigo – Hill Climbing

```
Função Hill-Climbing(Problema) retorna um estado que é o máximo local
Inicio
  EstadoAtual ← FazNó(Problema[EstadoInicial])
  loop do
    Vizinho ← SucessorDeMaiorValor(EstadoAtual)
    se Vizinho[Valor] for menor ou igual EstadoAtual[Valor] então
      retorna EstadoAtual
    EstadoAtual ← Vizinho
Fim
```

- Consiste de de um loop que continuamente move-se para os estados que aumentam o valor em sua função de avaliação.
- Termina quando atinge um "pico" onde nenhum vizinho tem um valor maior.
- Não mantem uma árvores de busca.

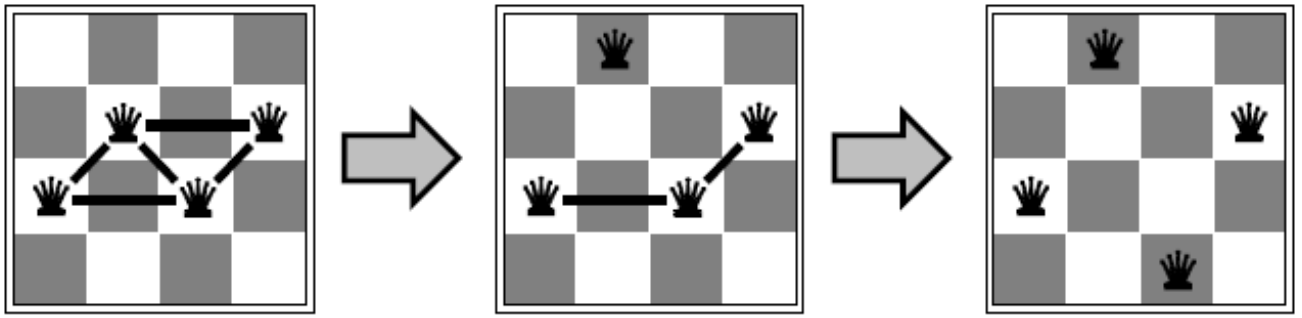
Hill Climbing – Exemplo

- **Problema:** 8 Rainhas (estados completos)
  - Em cada estado: 8 rainhas no tabuleiro, uma em cada coluna.
- **Ações:**
  - Mover uma rainha para outro quadrado na mesma coluna.
- Objetivo:
  - h = 0 (nenhuma rainha sendo atacada)



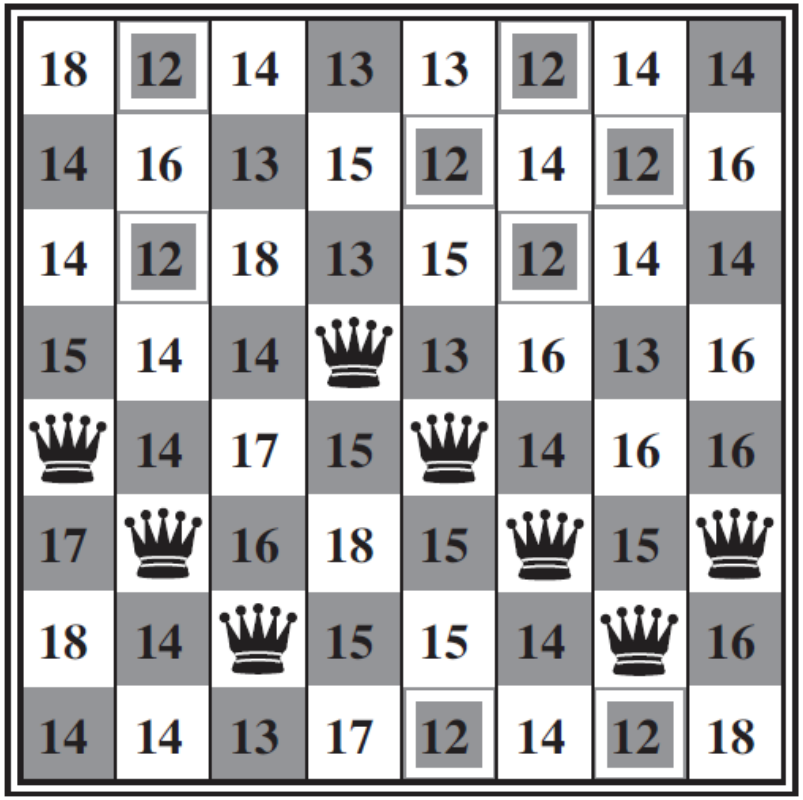
Hill Climbing – Exemplo

- **Iterações:**
  - Inicia com uma rainha em cada coluna
  - Move a rainha em sua coluna para reduzir número de conflitos
    - Cada estado possui 8 x 7 = 56 Vizinhos
- **Função Heurística (h):**
  - Número de rainhas sendo atacadas.



Hill Climbing – Exemplo

- **h = 17**
- Melhor movimento: 12



Hill Climbing

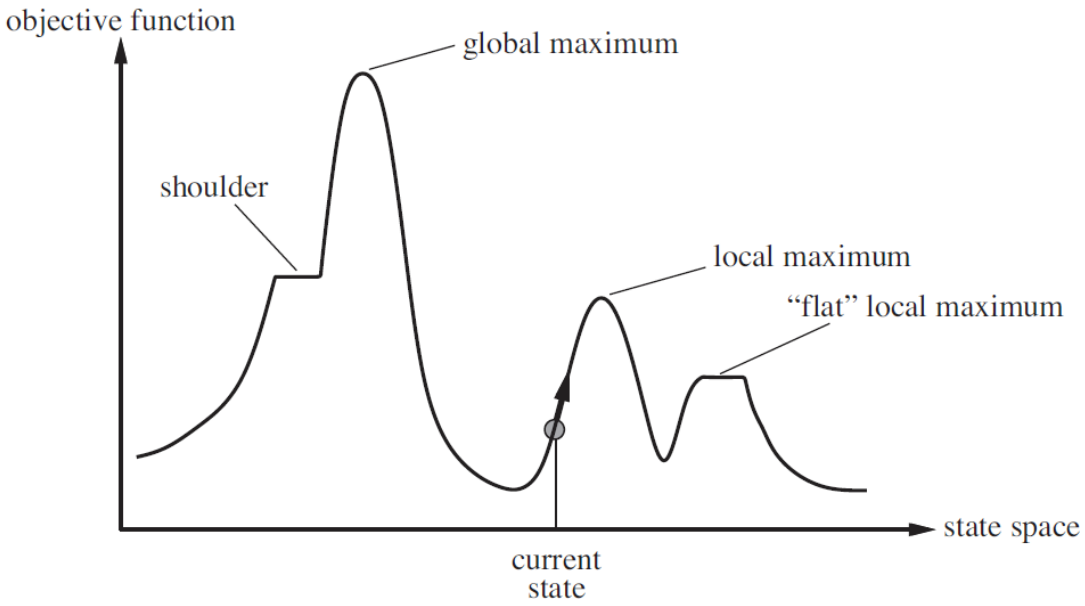
- É um **algoritmo guloso** – escolhe sempre o primeiro melhor vizinho para progredir na busca.

- Essa abordagem pode ter **bons resultados em alguns problemas**. Sendo capaz de progredir rapidamente para a solução problema.

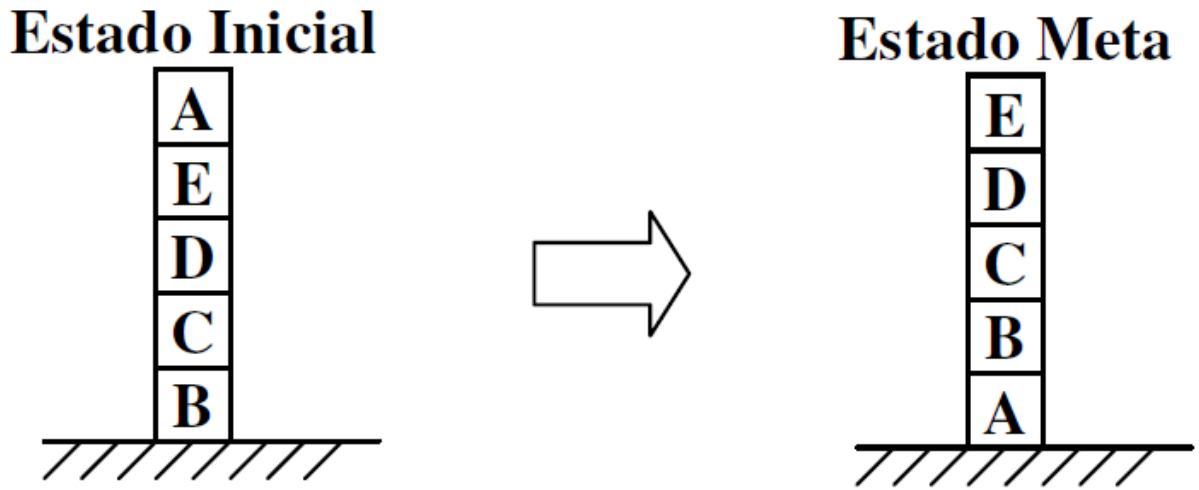
- Mas, sofre de três sérios **problemas**:
  - Máximos locais
  - Planícies
  - Encostas e Picos

Hill Climbing

- Máximos Locais
- Planícies



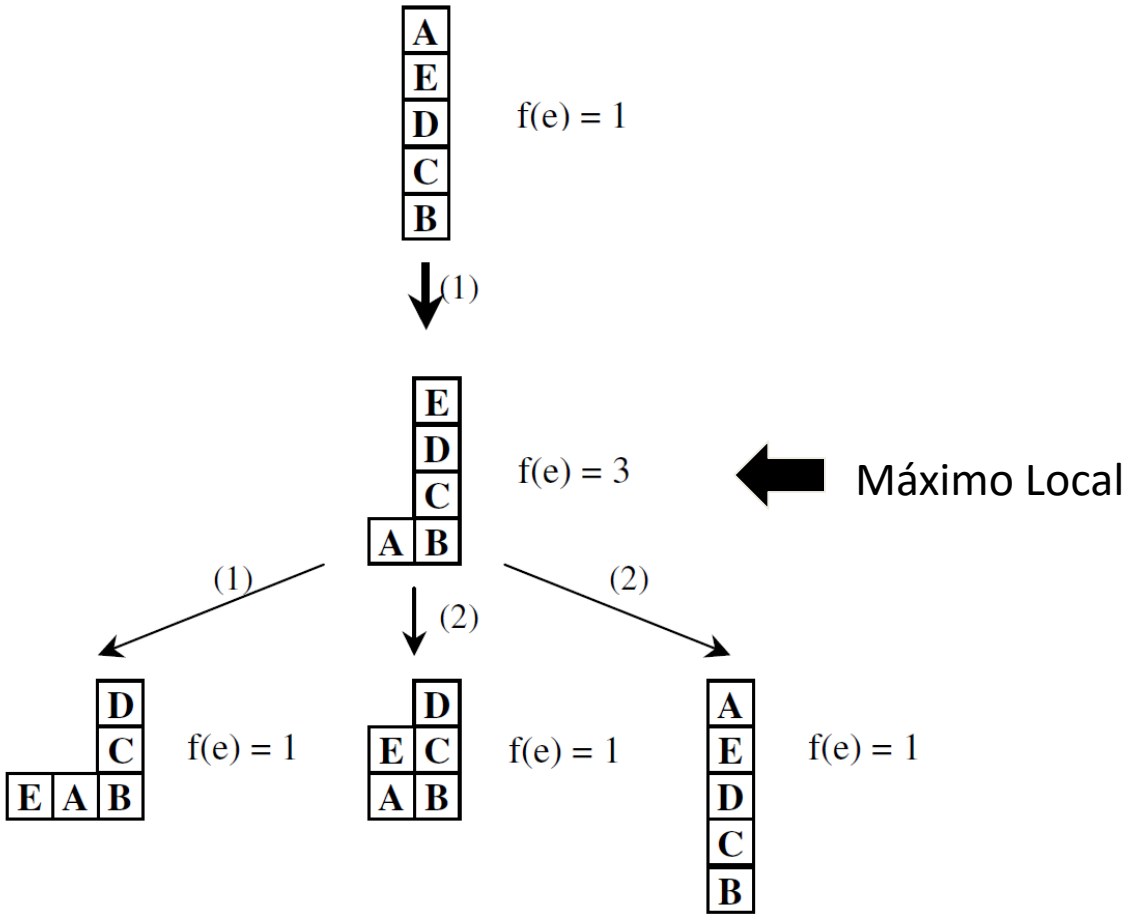
Hill Climbing – Exemplo



- **Ações Possíveis:**
  - Pegar um bloco e colocar ele sobre a mesa.
  - Pegar um bloco e colocar ele sobre outro bloco.

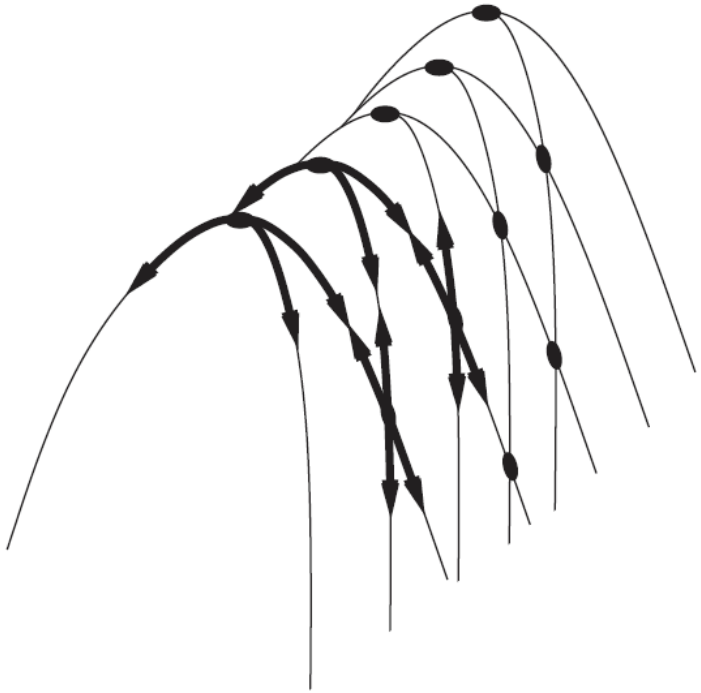
- **Heurística:**
  - +1 para cada bloco em cima do bloco onde ele deve estar.
  - -1 para cada bloco em cima do bloco errado.

Hill Climbing – Exemplo



Hill Climbing

- Encostas e Picos



Hill Climbing – Exemplo

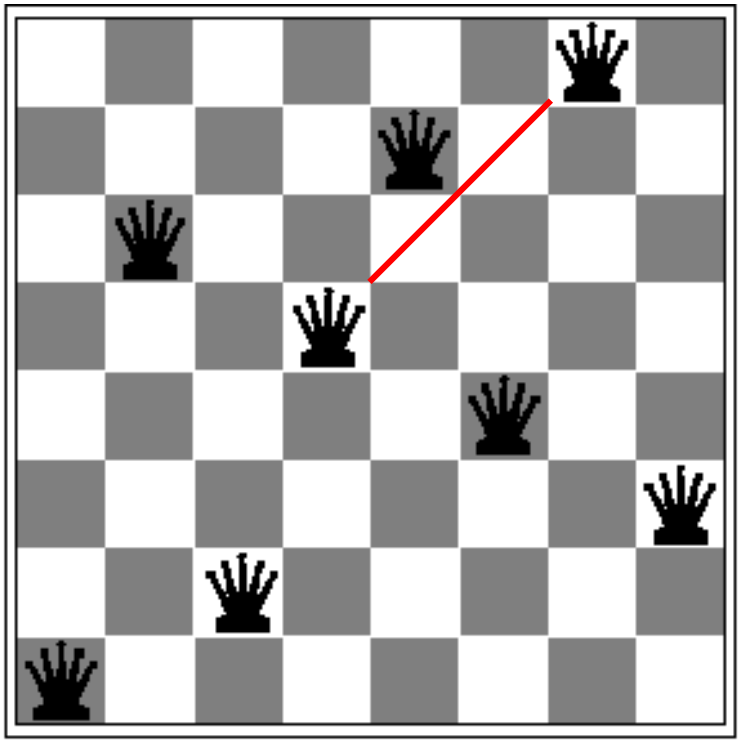
- **Problema:** 8 Rainhas

- **Inicializando aleatoriamente o estado inicial:**
  - O algoritmo fica em **preso em um máximo local** em 86% das vezes.
  - Resolve apenas 14% das instancias.

- Quando tem sucesso, resolve o problema em **aproximadamente 4 passos** – nada mal para um espaço de estados com 17 milhões de estados.

Hill Climbing – Exemplo

- Máximo local em 8 Rainhas
- **h = 1**
  - Mas qualquer movimento piora a solução



Hill Climbing

- **Variações:**
  - Random-Restart Hill Climbing;

- **Não é ótimo e não é completo.**

- O desempenho do Hill Climbing depende muito do **formato do panorama** do espaço de estados.

Pseudocódigo – Simulated Annealing

```
Função Simulated-Annealing(Problema, Escalonamento) retorna um estado que é
    o máximo local
Inicio
  EstadoAtual ← Criar-Nó(Problema[EstadoInicial])
  loop do
    t = t + 1
    T ← Escalonamento[t]
    Se T = 0 então retorna EstadoAtual
    Próximo ← seleciona um sucessor do EstadoAtual aleatoriamente
    ΔE ← Próximo[Valor] – EstadoAtual[Valor]
    se ΔE > 0 então EstadoAtual ← Próximo
    senão EstadoAtual ← Próximo somente com probabilidade e^(ΔE/T)
Fim
```

- Combina a subida de encosta com um **percurso aleatório** resultando em eficiência e completeza.
- Subida de encosta dando uma “chacoalhada” nos estados sucessores;
- Idéia: **Escapar do máximo local** aceitando alguns movimentos “ruins” mas gradualmente diminuindo sua frequência usando o critério de aceitação de **Boltzmann** $e^{(\Delta E/T)}$

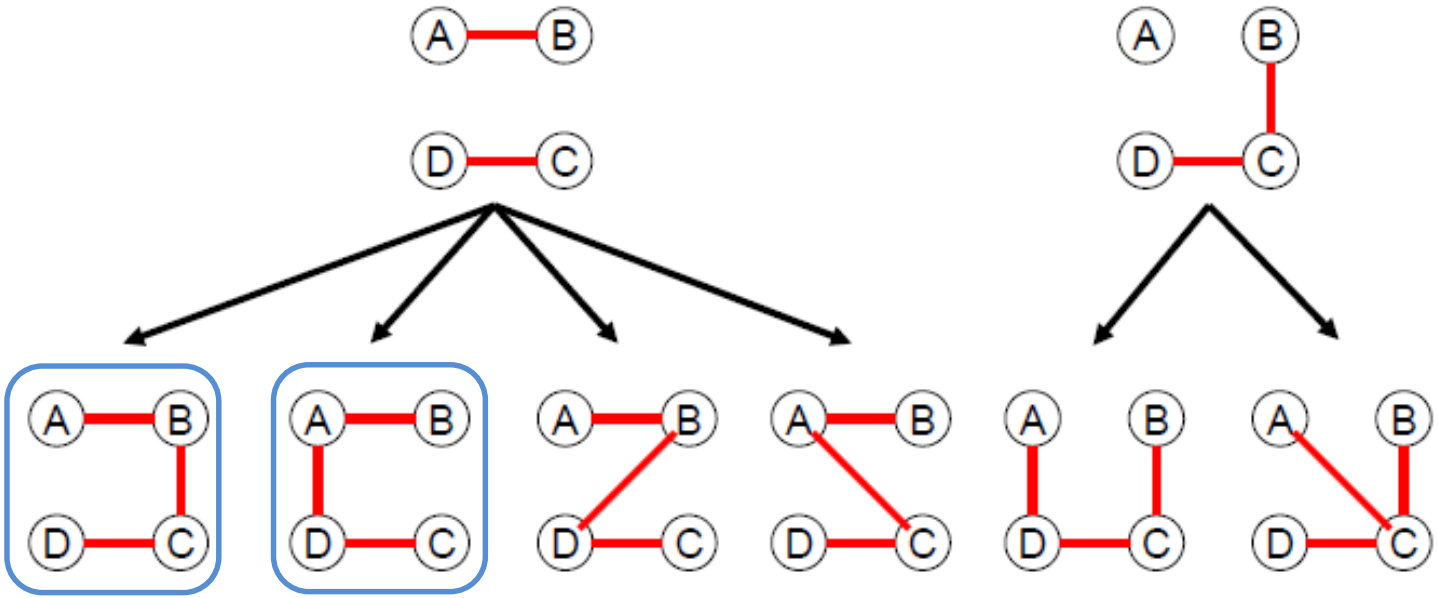
Local Beam Search

```
Função Beam-Search(Problema, k) retorna um estado que é o máximo local
Inicio
  Inicia k estados aleatoriamente
  loop do
    gera todos os sucessores para os k estados
    Se algum sucessor é a solução
       então retorna sucessor
    senão seleciona os melhores k sucessores
Fim
```

- Mantém ***k* estados em memória** – melhor que apenas um.
- Começa com *k* estados gerados aleatoriamente – cada **execução pode gerar resultados diferentes.**
- Análogo a seleção natural.
- Desvantagem: todos os ***k* estados podem parar em um máximo local**

Local Beam Search - Exemplo

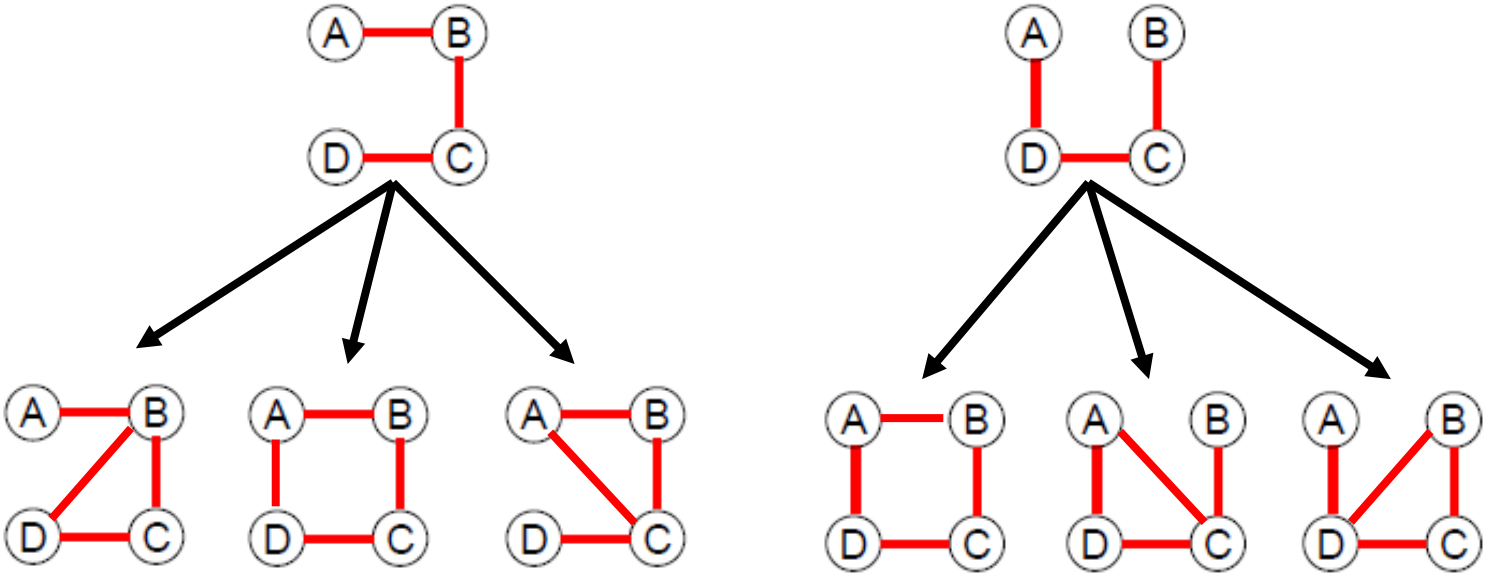
- Problema do Caixeiro Viajante (k = 2)



Seleciona os melhores *k* sucessores a partir da lista completa de nós

Local Beam Search - Exemplo

- Problema do Caixeiro Viajante (k = 2)
  - Repete o processo até encontrar a solução



Busca Online

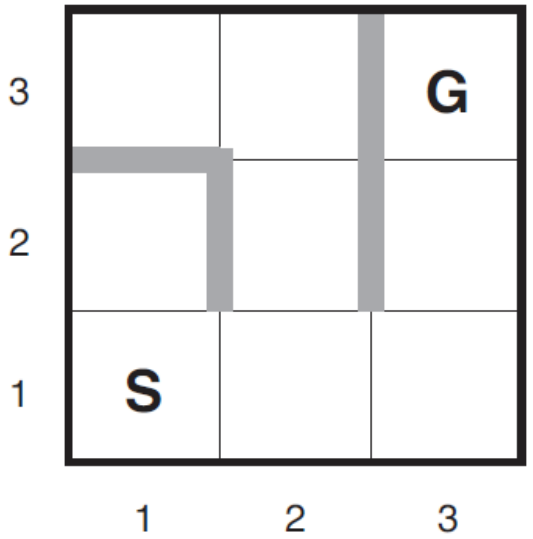
- Todos os métodos de busca vistos até o momento são **offline**, ou seja, o agente calcula todos antes passos antes de agir.

- Agentes de busca online devem **intercalar entre busca e execução das ações**.

- Apropriado para **ambientes dinâmicos, desconhecidos** e também para **espaços de busca muito grandes**.

Busca Online - Exemplo

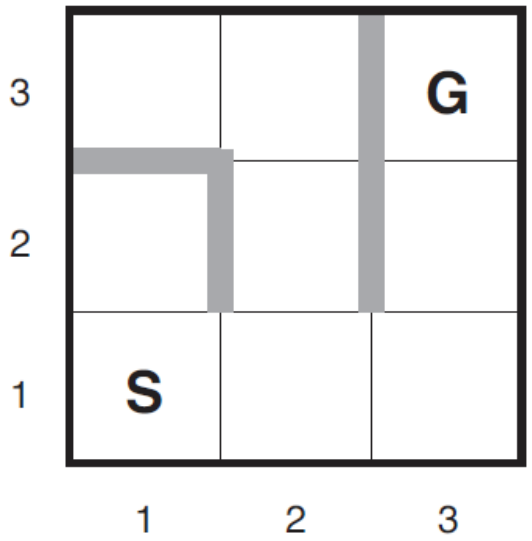
- **Informações:**
  - **Ações(s)**
    - Lista de ações possíveis no estado s.
  - **C(s, a, s')**
    - Custo da execução da ação a no estado s resultando no estado s'.
  - **TesteObjetivo(s)**
    - Testa se s é o estado objetivo.
- O agente somente tem acesso aos estados sucessores após executar as ações.



Busca em Profundidade Online

- Realiza uma busca em profundidade executando as ações fisicamente durante o processo até chegar no estado objetivo.

- Se todos os vizinhos de um estado já foram visitados é necessário retornar fisicamente para a posição anterior.



Busca Local Online

- O **hill climbing** é naturalmente um algoritmo de busca online.

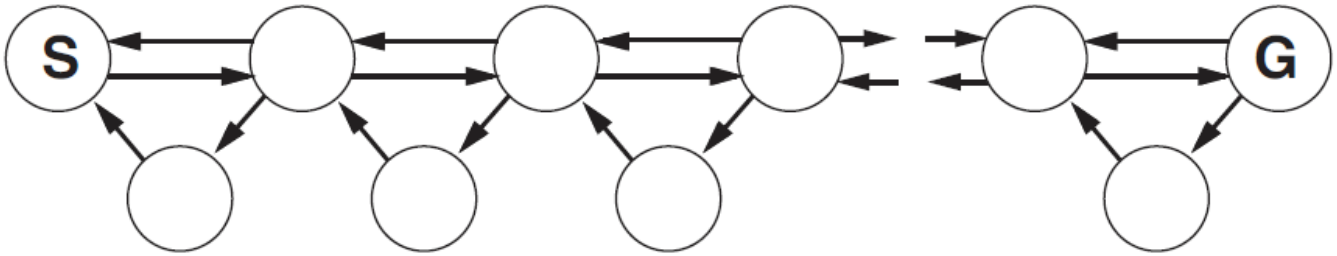
- **Problema:**
  - Na sua forma mais simples, pode deixar o agente parado em um **máximo local** sem ter para onde ir.
  - Não é possível **reiniciar o processo randomicamente** porque o agente não pode teletransportar.

- **Solução:**
  - Caminhada aleatória ao chegar em uma máximo local.

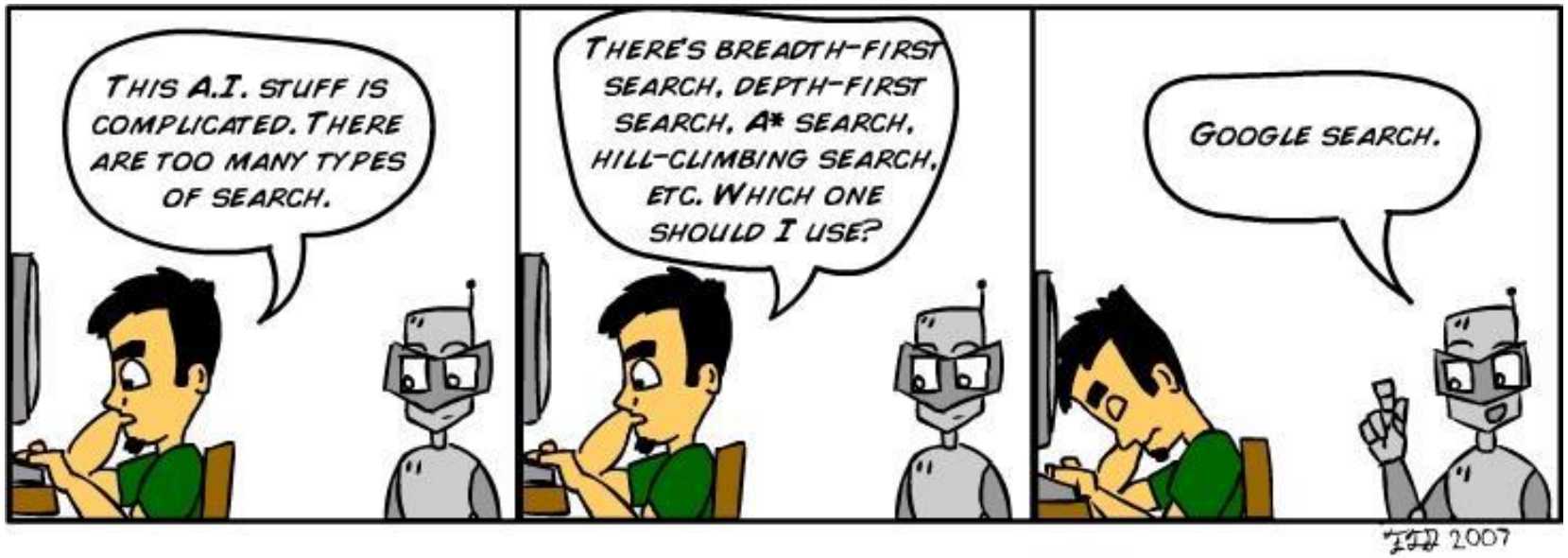
Busca Local Online

- A caminhada aleatória garante que o agente eventualmente vai encontrar o objeto ou completar o processo de exploração.

- Esse processo pode ser lento.

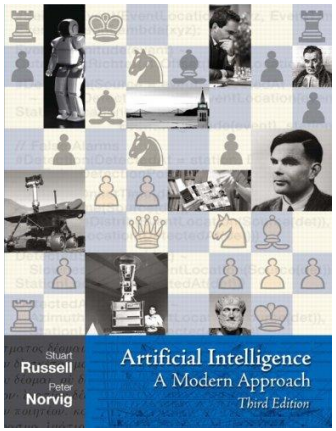


Qual Algoritmo Usar?



Leitura Complementar

- Russell, S. and Norvig, P. **Artificial Intelligence: a Modern Approach**, 3nd Edition, Prentice-Hall, 2009.
  - **Capítulo 4: Informed Search and Exploration**



- Coppin, B. **Artificial Intelligence Illuminated**, Jones & Bartlett Learning, 2004.
  - **Capítulo 4: Search Methodologies**
  - **Capítulo 5 : Advanced Search**