

Inteligência Computacional

INF8286

Aula 05 – Busca Local

Augusto Baffa
<augusto.baffa@uva.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.

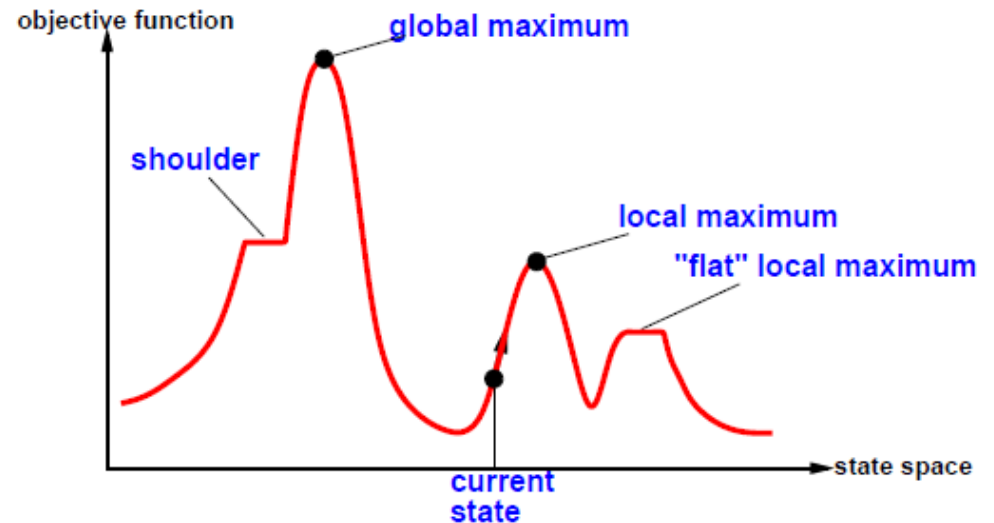
Busca Local

- **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

Início

EstadoAtual \leftarrow FazNó(*Problema*[EstadoInicial])

loop do

Vizinho \leftarrow SucessorDeMaiorValor(EstadoAtual)

se Vizinho[Valor] for menor ou igual EstadoAtual[Valor] **então**

retorna EstadoAtual

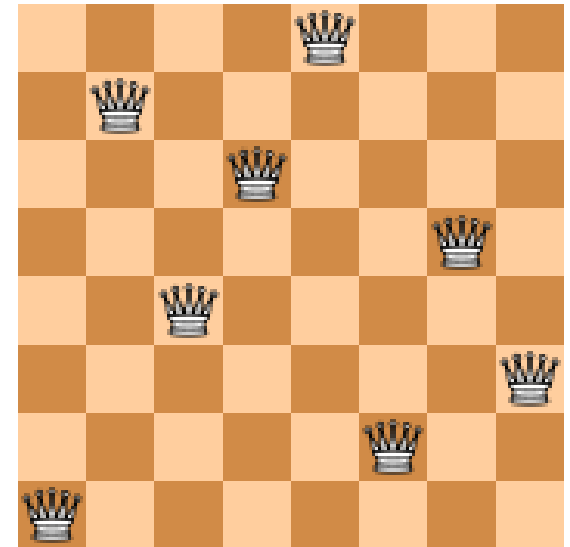
EstadoAtual \leftarrow Vizinho

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.

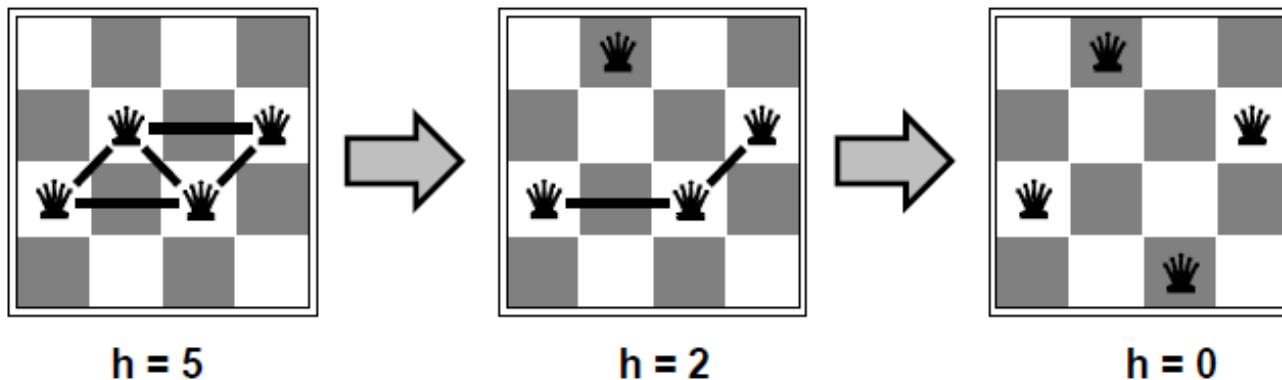
Hill Climbing – Exemplo

- **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 $8 \times 7 = 56$ Vizinhos
- **Função Heurística (h):**
 - Número de rainhas sendo atacadas.



Hill Climbing – Exemplo

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

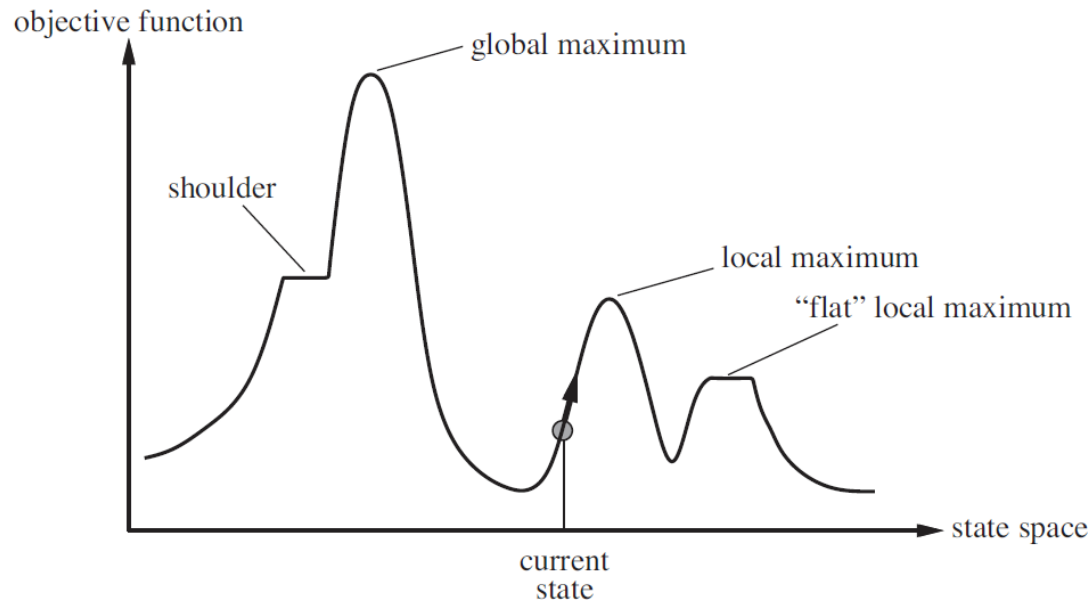
18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♔	13	16	13	16
♔	14	17	15	♔	14	16	16
17	♔	16	18	15	♔	15	♔
18	14	♔	15	15	14	♔	16
14	14	13	17	12	14	12	18

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

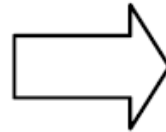
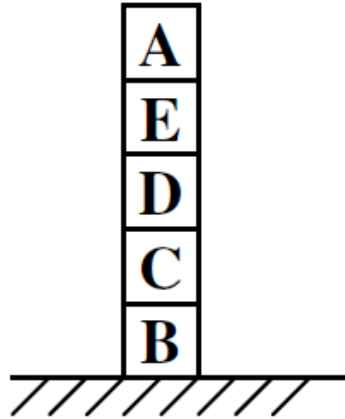
Hill Climbing

- Máximos Locais
- Planícies

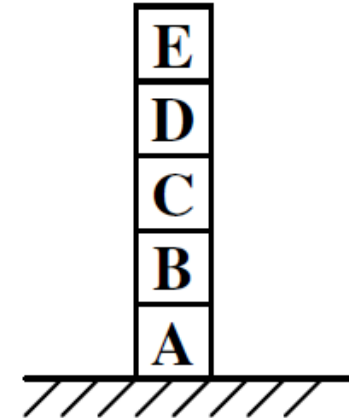


Hill Climbing – Exemplo

Estado Inicial

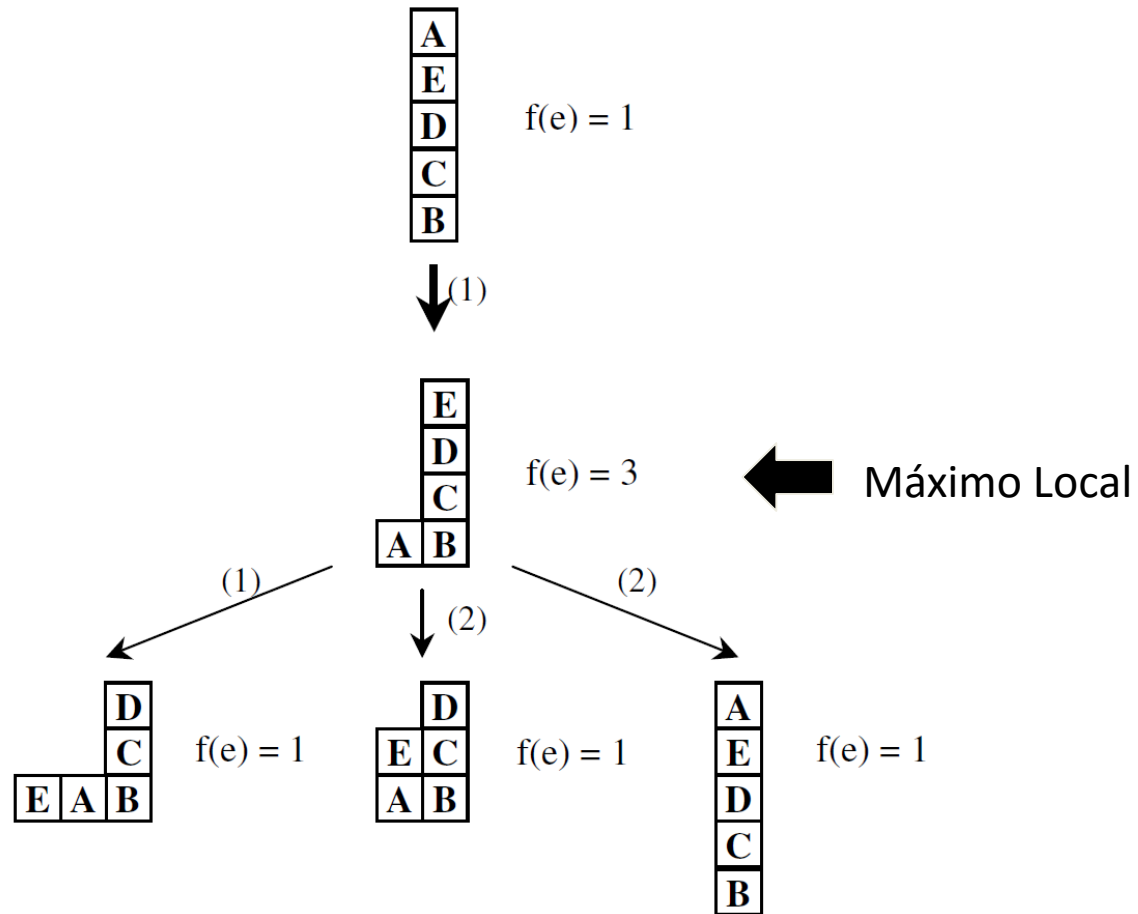


Estado Meta



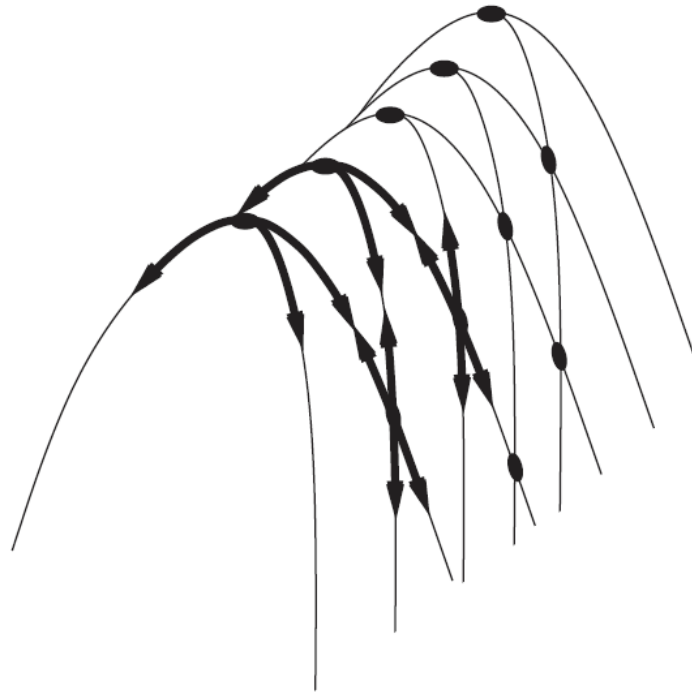
- **Ações Possíveis:**
 - Pegar um bloco e colocar ele sobre a mesa.
 - Pegar um bloco e colocar ele sobre outro bloco.
- **Heurística:**
 - +1 para cada bloco em cima do bloco onde ele deve estar.
 - -1 para cada bloco em cima do bloco errado.

Hill Climbing – Exemplo



Hill Climbing

- Encostas e Picos

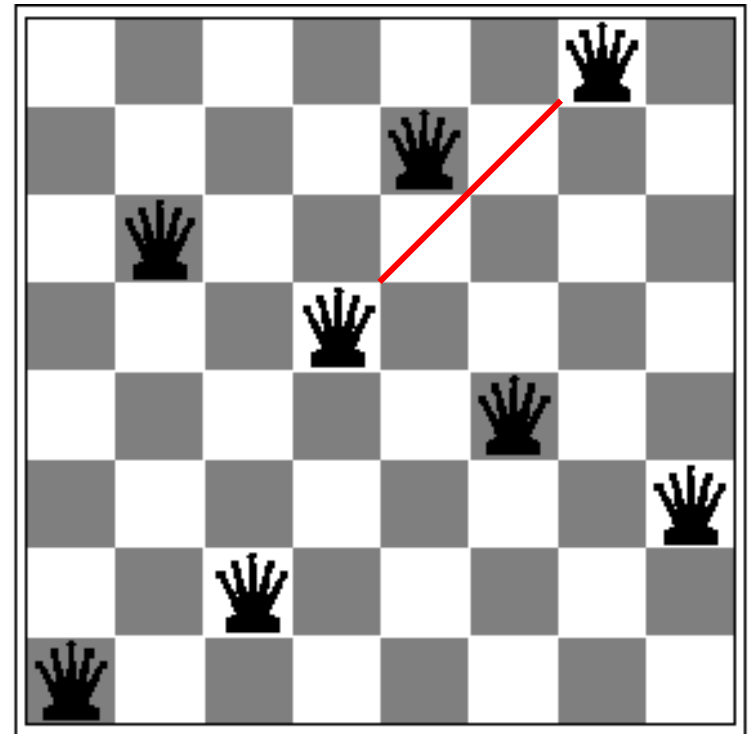


Hill Climbing – Exemplo

- **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

Início

EstadoAtual \leftarrow Criar-Nó(*Problema*[EstadoInicial])

loop do

$t = t + 1$

$T \leftarrow$ Escalonamento[t]

Se $T = 0$ **então retorna** EstadoAtual

 Próximo \leftarrow seleciona um sucessor do EstadoAtual aleatoriamente

$\Delta E \leftarrow$ Próximo[Valor] - EstadoAtual[Valor]

se $\Delta E > 0$ **então** EstadoAtual \leftarrow Próximo

senão EstadoAtual \leftarrow Próximo somente com probabilidade $e^{\Delta E/T}$

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

Início

Inicia *k* estados aleatoriamente

loop do

gera todos os sucessores para os *k* estados

Se algum sucessor é a solução

então retorna sucessor

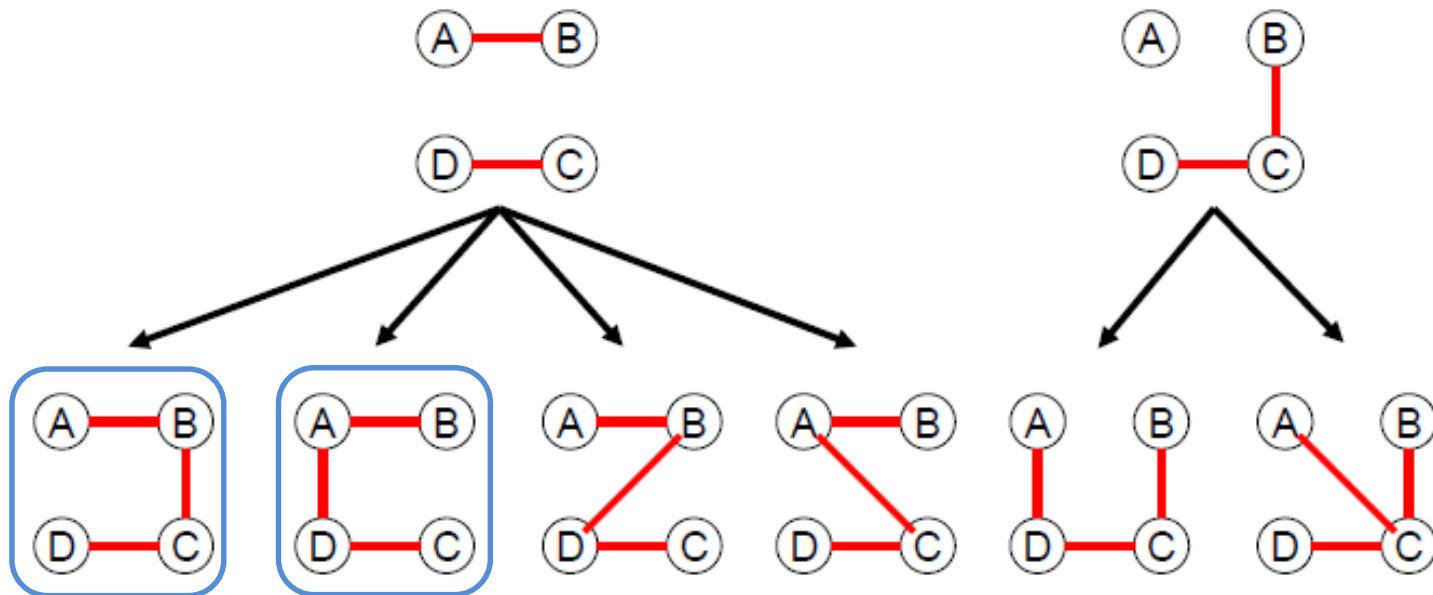
senão seleciona os melhores *k* sucessores

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

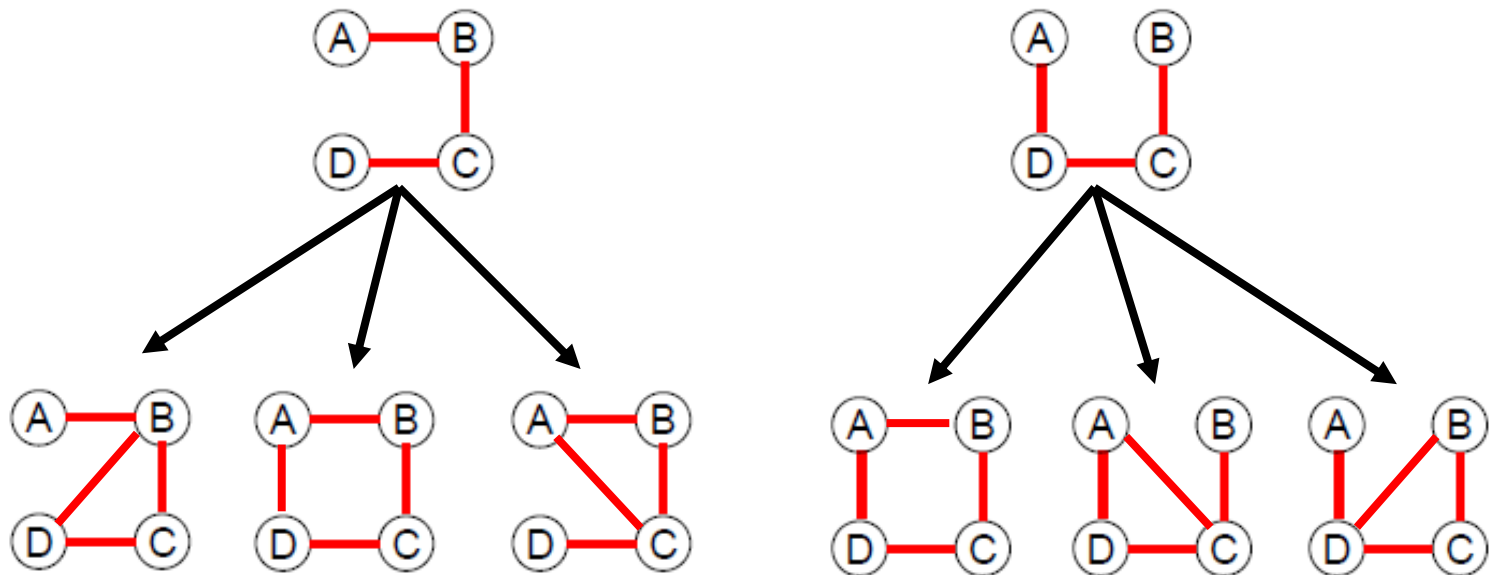
- Problema do Caixeiro Viajante ($k = 2$)



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



Qual Algoritmo Usar?



Leitura Complementar

- Russell, S. and Norvig, P. **Inteligência Artificial**, 3a Edição, Campus, 2013.
 - **Capítulo 4: Além da busca clássica**
- Coppin, B. **Inteligência Artificial**, 1ª Edição, LTC, 2010.
 - **Capítulo 5: Busca avançada**
- Luger, G. **Inteligência Artificial**, 6a Edição, Pearson, 2014.
 - **Capítulo 4: Busca Heurística**

