

Inteligência Artificial

Além da Busca Clássica

Prof. Fabio Augusto Faria

Material adaptado de Profa. Ana Carolina Lorena e livro
“Inteligência Artificial, S. Russell e P. Norving”

1º semestre 2015

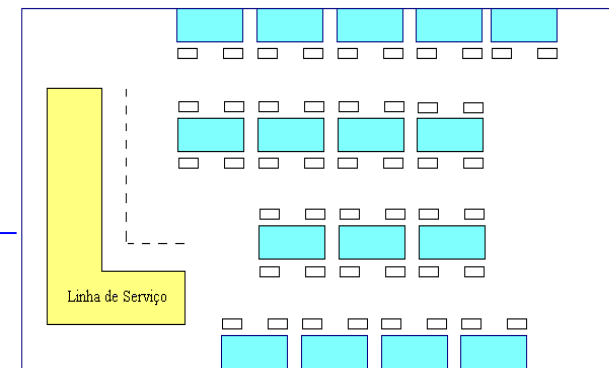


Algoritmos de busca

- Até agora: **exploração sistemática** do espaço de busca
 - Mantendo um ou mais caminhos na memória
 - Registrando alternativas exploradas e não exploradas
- Há problemas em que *caminho é irrelevante* e o **estado final em si é a solução**

Exemplos:

- projeto de circuitos,
- layout de instalações,
- escalonamento de trabalho,
- otimização de rede,
- gerenciamento de carteiras
- 8 rainhas: basta mostrar o tabuleiro final



Meta-heurísticas

Meta-heurísticas

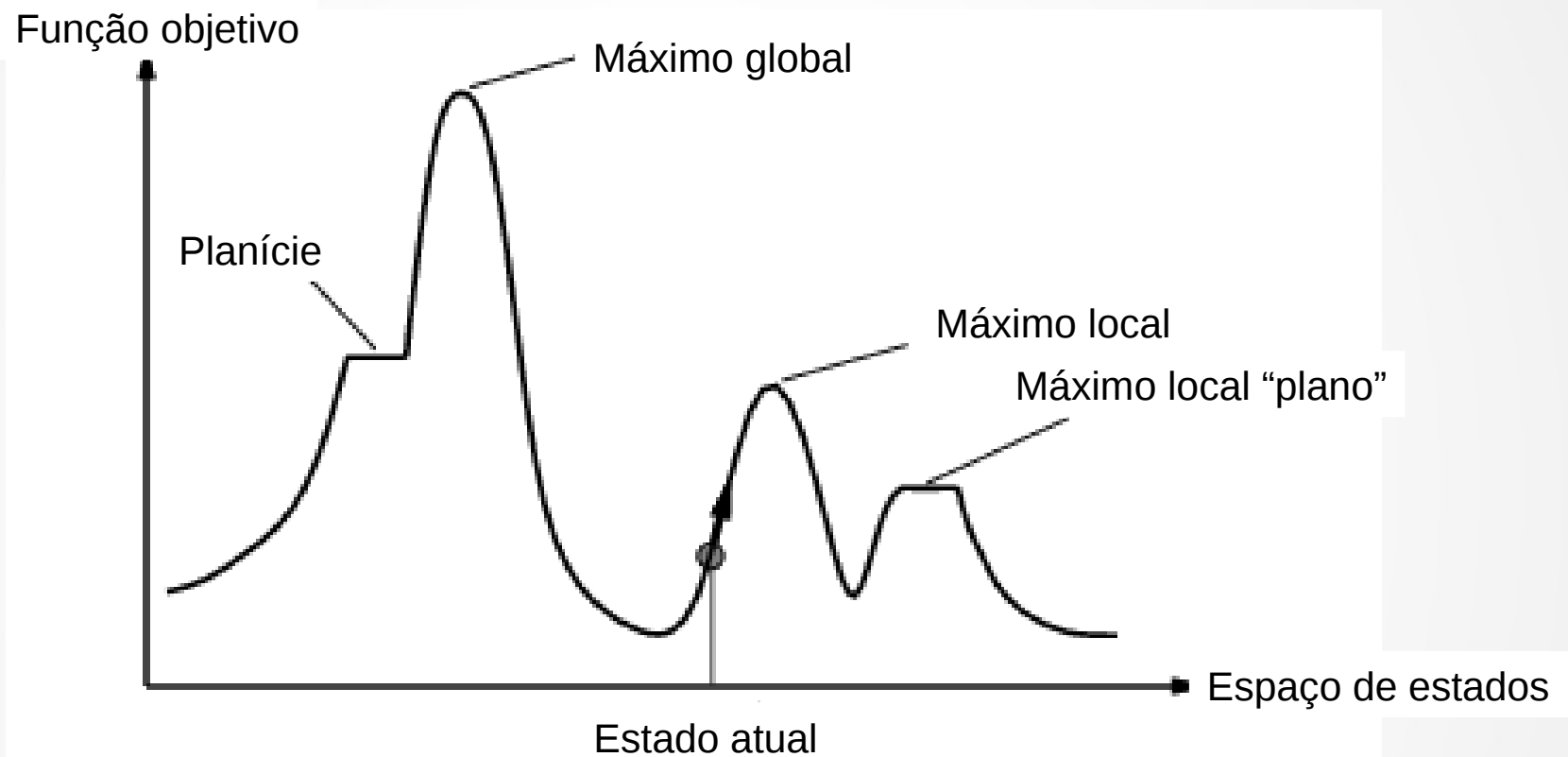
Método heurístico para resolver de forma genérica problemas de busca e otimização

- Não necessariamente guardam caminho à solução
 - Usando então pouca memória
- Podem encontrar soluções razoáveis para problemas grandes
 - Até mesmo infinitos

Problema de otimização:

- Estado: vetor de variáveis
- Função objetivo: $f: estado \rightarrow \mathbb{R}$
- Objetivo: achar estado que maximize a função objetivo ou minimize o custo de uma heurística $h(n)$

Espaço de busca



Algoritmo **completo** sempre encontra uma solução, caso ela exista

Algoritmo **ótimo** sempre encontra mínimo/máximo global

Busca com melhoria iterativa

A idéia é começar com o *estado inicial* e *melhorá-lo iterativamente*

Os estados podem ser representados sobre uma superfície

A altura de qualquer ponto na superfície corresponde à *função de avaliação* do estado naquele ponto

O algoritmo se “move” pela superfície em busca de pontos mais altos/baixos

Ponto mais alto/baixo (máximo/mínimo global) corresponde à solução ótima

- Estado onde a função de avaliação atinge seu valor máximo/mínimo

Busca com melhoria iterativa

Esses algoritmos *guardam apenas o estado atual*, e não veem além dos *vizinhos imediatos* do estado

Contudo, muitas vezes são os melhores métodos para tratar problemas reais muito complexos

Duas classes de algoritmos:

Hill-Climbing: Subida de Encosta

- Só faz modificações que melhoram o estado atual

Simulated Annealing: Têmpera Simulada

- pode fazer modificações que pioram o estado temporariamente para possivelmente melhorá-lo no futuro

Busca subida de encosta

Hill-climbing

Se move de forma contínua em valor crescente

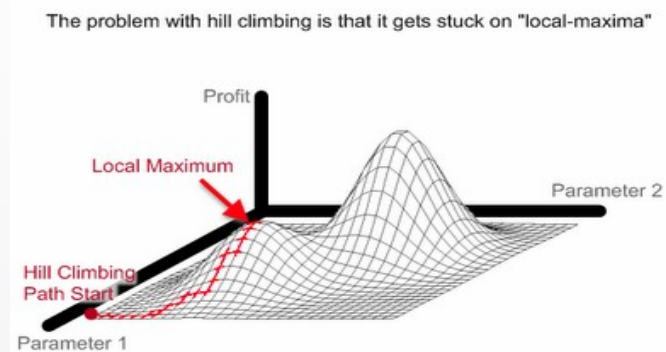
- Encosta acima

Termina quando alcança um pico

- Em que nenhum vizinho tem valor mais alto

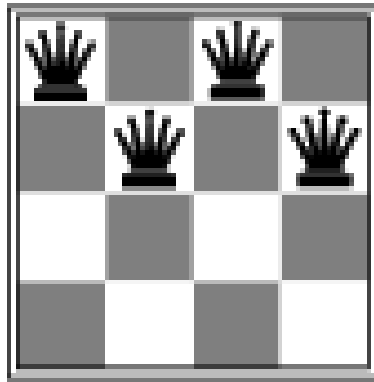
Examina vizinhos imediatos

- Não precisa manter árvore de busca
- Guarda só estado corrente e tenta melhorá-lo

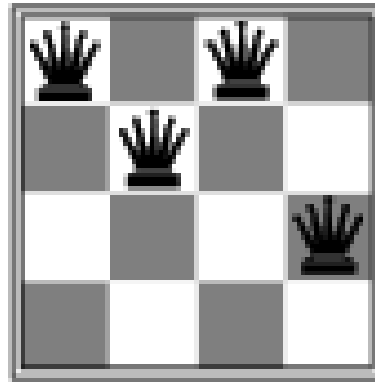
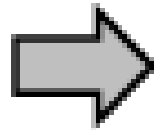


Ex.: n-rainhas

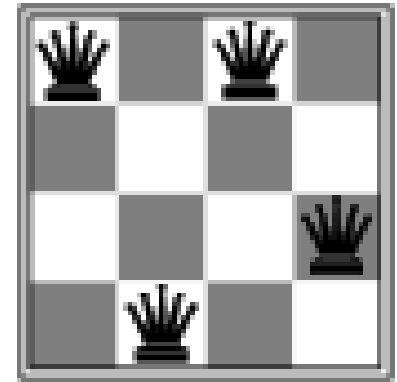
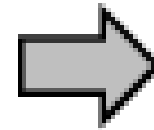
Começa por um estado e vai tentando melhorá-lo sucessivamente



5 pares de rainhas se atacam



3 pares de rainhas se atacam

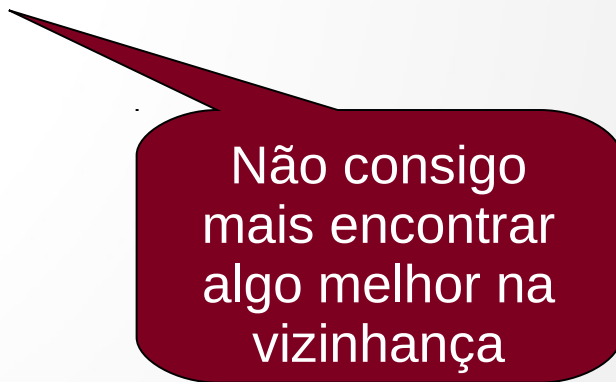


1 par de rainhas
se atacam

Algoritmo

`Subida_em_encosta(problema)`

1. `Nó_atual = Estado_inicial(problema)`
2. **Repita**
 - 2.1 `Nó_vizinho = sucessor de nó_atual com melhor valor de avaliação`
 - 2.2 **Se** `aval(nó_vizinho) <= aval(nó_atual)` **então**
 retorne como solução o estado do nó corrente
 - 2.3 `Nó_atual = nó_vizinho`
3. **Fim_repita**



Não consigo
mais encontrar
algo melhor na
vizinhança

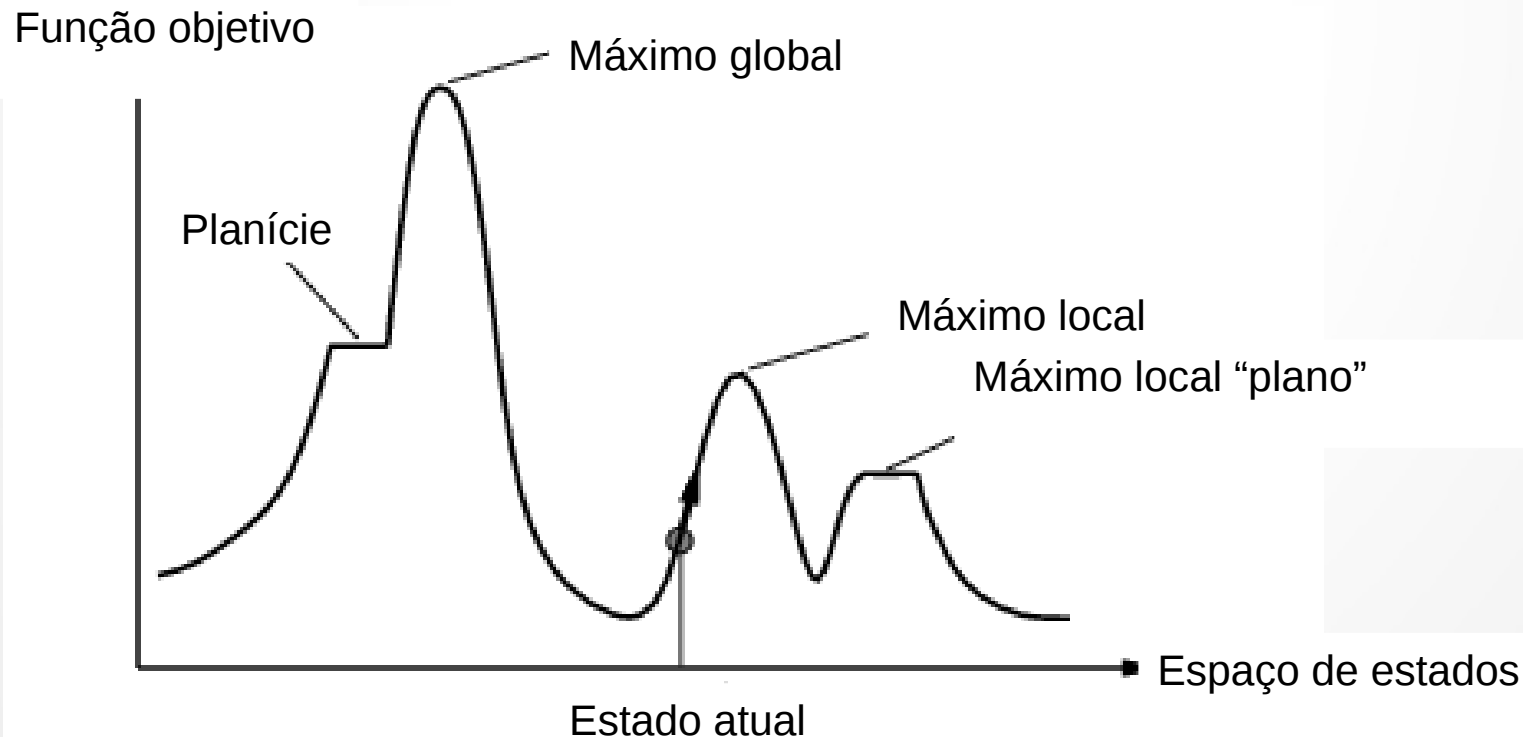
Busca subida em encosta

Chamada de busca gulosa local

Captura um bom estado vizinho

Porém, pode ficar paralisada:

- Em máximos/mínimos locais
- Em platôs



Busca subida em encosta

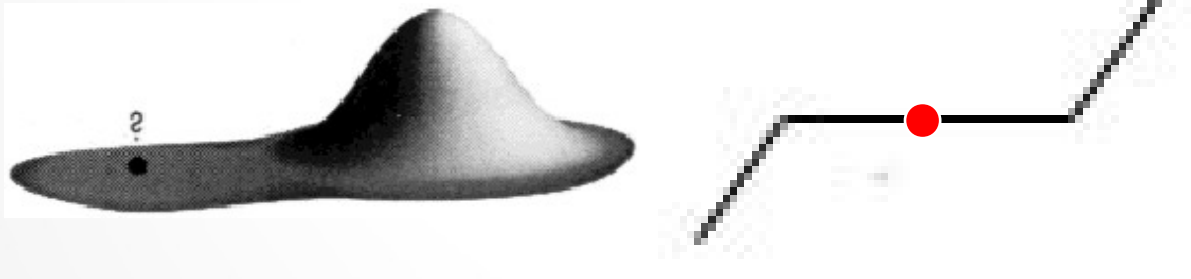
- **Máximos Locais:**

- em contraste com *máximos globais*, são picos mais baixos do que o pico mais alto no espaço de estados (solução ótima)
- a função de avaliação leva a um valor máximo para o caminho sendo percorrido: essa função utiliza informação local
- porém, o nó final está em outro ponto mais alto
- isto é uma consequência das decisões irrevogáveis do método
 - e.g., xadrez: eliminar a Rainha do adversário pode levar o jogador a perder



Busca subida em encosta

- **Platôs:**
 - uma região do espaço de estados onde a função de avaliação dá o mesmo resultado.



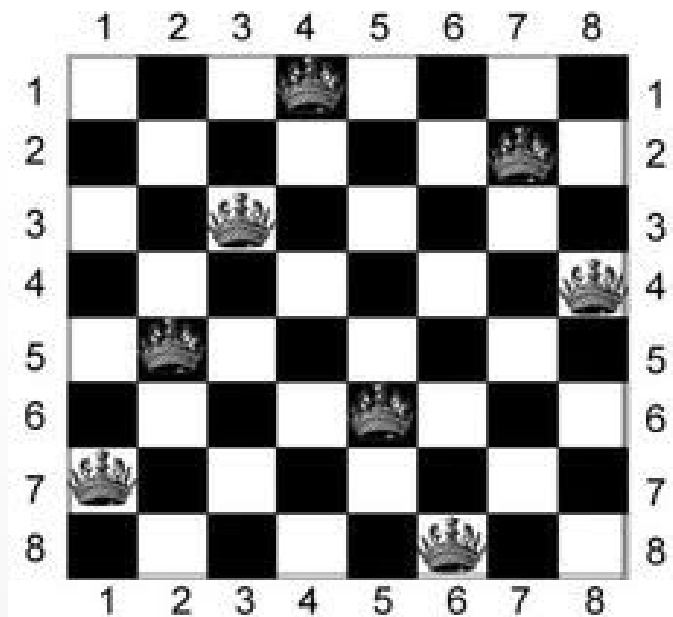
Busca subida de encosta

Problema das oito rainhas: formulação de estados completos

Cada estado tem 8 rainhas no tabuleiro, uma por coluna

Novos estados são gerados movendo uma rainha para outro quadrado na mesma coluna

- Cada estado tem $8 \times 7 = 56$ sucessores



Busca subida de encosta

Problema das oito rainhas:

Função **heurística** h

- h = número de pares de rainhas se atacando
 - Neste caso, quanto menor, melhor
- Mínimo global é 0
- Escolhe melhor sucessor corrente
 - Se houver vários, escolhe aleatoriamente um deles

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

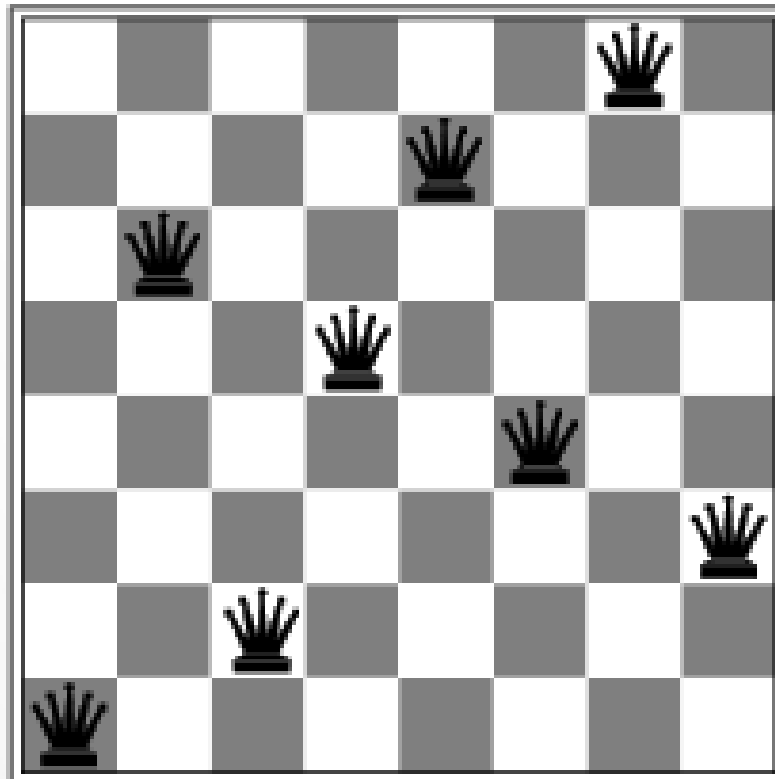
$h = 17$ nesse estado
melhor sucessor $h = 12$

Busca subida em encosta

Mínimo local com $h = 1$

Todo sucessor tem custo mais alto

Alcançada em 5 passos a partir do estado inicial



Busca subida em encosta

Problema das 8 rainhas

Estados iniciais aleatórios

- 86% das vezes busca fica paralisada
 - Resolve apenas 14% das instâncias do problema
- Mas é rápida
 - 4 passos em média quando tem sucesso
 - 3 passos em média quando fica paralisada
 - Em espaço que tem cerca de 17 milhões de estados

Busca subida em encosta

Mudar de estado em platôs

- Quando estados têm avaliações iguais
- Colocando um limite de vezes

Ex. **Problema das 8 rainhas**

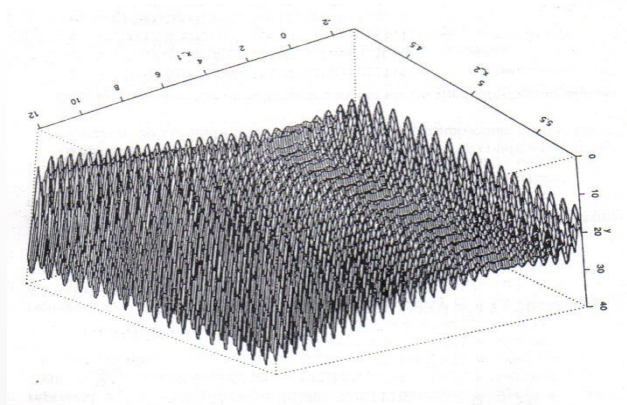
- Com estados iniciais aleatórios, usando essa estratégia
 - Passa a resolver 94% das instâncias do problema
 - Mas demora mais
 - » 21 passos em média quando tem sucesso
 - » 64 passos em média quando falha

Busca de subida em encosta

Sucesso depende da topologia do espaço de estados

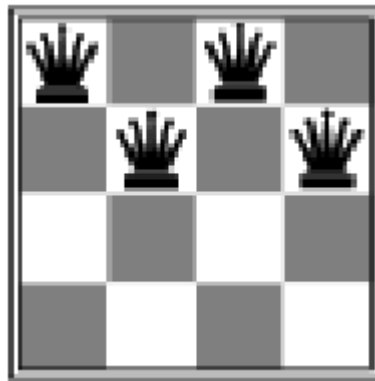
Se houver poucos máximos locais e platôs

- Subida de encosta com reinício aleatório encontrará boa solução com rapidez
- Mesmo para mais complexos, pode encontrar máximo local razoavelmente bom
 - Com poucos reinícios



Busca de subida em encosta

Exercício: aplique a busca de subida em encosta ao seguinte tabuleiro, usando a heurística de minimizar o número de pares de rainhas que se atacam



Referências

- Livros:
 - Russel e Norvig: Inteligência Artificial, cap 4.1 e 4.2.