Inteligência Artificial
Tópico 03 - Parte 04
Resolução de Problemas por Buscas
Além da Busca Clássica - Busca Local

Profa. Dra. Priscila Tiemi Maeda Saito
priscilasaito@ufscar.br

Roteiro

1. Resolução de Problemas por Buscas
   - Busca Cega ou Exaustiva
   - Busca Heurística
   - Busca Local - Além da Busca Clássica
     - Hill-Climbing
     - Simulated Annealing
     - Local Beam
     - Genetic Algorithms

Resolução de Problemas por Buscas - Clássicas

- **Interesse:** buscar sistematicamente, a partir de um estado inicial, uma sequência de ações que levem à meta
- **Solução:** caminho ao estado-objetivo
- Algoritmos de busca exploram o espaço de estados
  - mantêm um ou mais caminhos na memória
  - registram alternativas exploradas e não exploradas

Resolução de Problemas por Buscas - Clássicas

- Problemas de otimização $\rightarrow$ **caminho** ao objetivo é **irrelevante**
- **Solução** = estado objetivo

Exemplos

- projeto de circuitos integrados
- layout de instalações industriais
- escalonamento de trabalho
- otimização de redes
- gerenciamento de salas de aula
- 8 rainhas (basta mostrar o tabuleiro final)
- ...

Descrição do estado contém toda informação relevante para a solução $\rightarrow$ Caminho até a meta não importa

Além da Busca Clássica

Busca Local ou de melhoria iterativa

Operam em um único estado e move-se para a vizinhança deste estado

Ideia

Começar com o estado inicial (configuração completa, solução aceitável), e melhorá-lo iterativamente

Vantagens

- utiliza pouca memória (usualmente uma quantidade constante)
- frequentemente encontra boas soluções em espaço de estados muito grandes ou mesmo infinitos, nos quais estratégias sistemáticas são inadequadas ou inviáveis

Busca Local

- Úteis para resolver **problemas de otimização**
  - buscar por estados que atendam a uma **função objetivo**

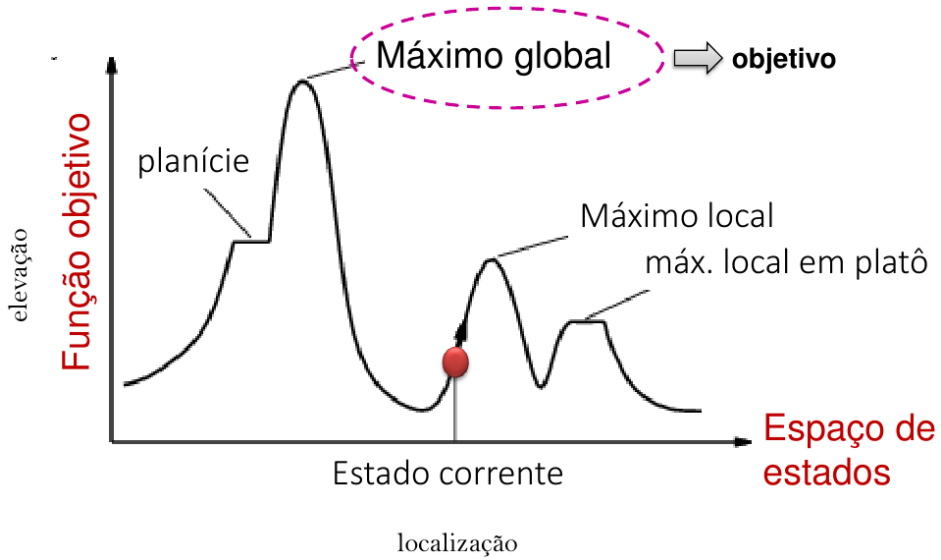
Objetivo

Encontrar o estado que:

- maximize a função objetivo ou
- minimize o custo de uma heurística h(n)

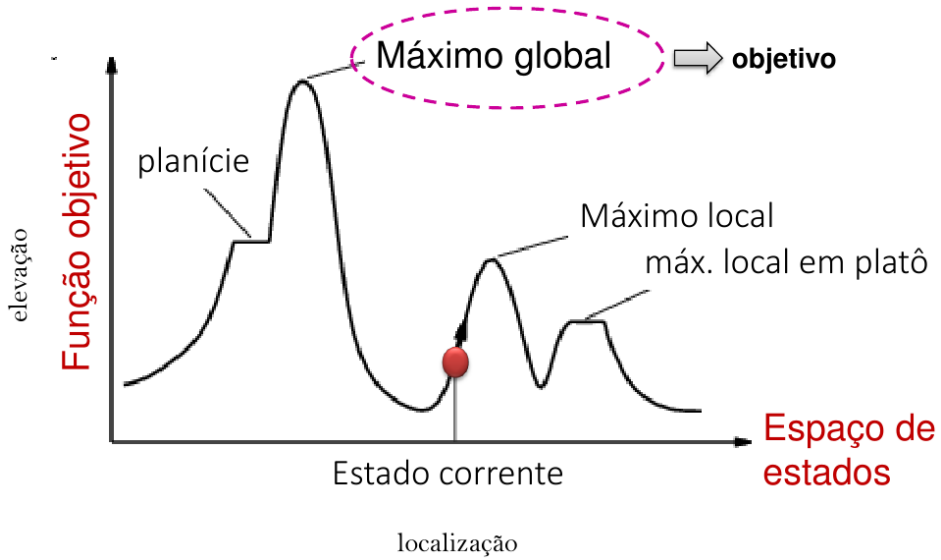
Busca Local

- Estados podem ser representados sobre uma superfície
  - altura de qualquer ponto na superfície corresponde à função de avaliação f(n) do estado naquele ponto
- Algoritmo move pela superfície em busca de pontos mais altos/mais baixos (**solução ótima**)
- f(n) = custo
  - objetivo é encontrar o ponto mais baixo (**mínimo global**)
- f(n) = função objetivo
  - objetivo é encontrar o ponto mais alto (**máximo global**)



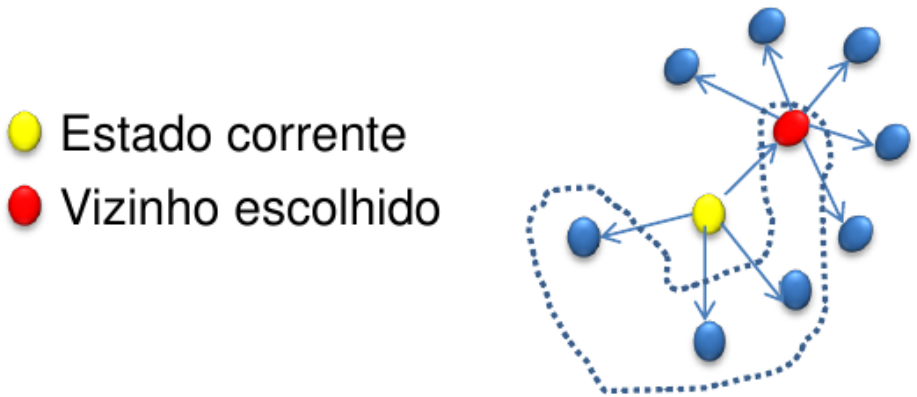
Busca Local

- **Completo:** consegue atingir um estado objetivo, caso exista
- **Ótimo:** consegue encontrar o mínimo/máximo global



Busca Local

- Algoritmos guardam apenas o estado corrente, e não veem além dos vizinhos imediatos do estado
- Muitas vezes são os melhores métodos para tratar problemas reais muito complexos



Busca Local

- Principais algoritmos
  - Hill-Climbing
  - Simulated Annealing
  - Local Beam
  - Genetic Algorithms

Hill-Climbing

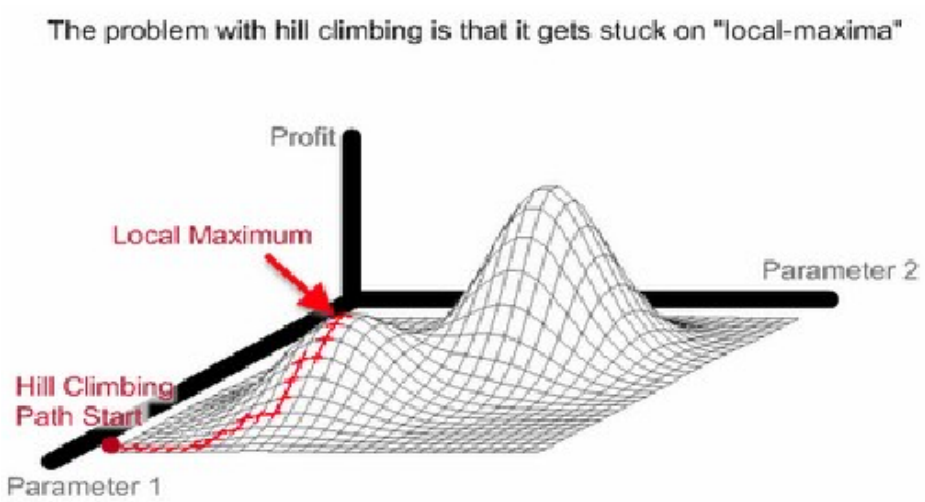
Analogia

Metáfora

- Tentar encontrar o topo do Monte Everest sob uma forte neblina e sofrendo de amnésia

Hill Climbing

- Subida de encosta, subida de encosta mais íngreme, subida da colina, subida da montanha, **busca local gulosa**, ...
- Move de forma contínua em direção a valores crescentes da função objetivo (encosta acima)
- Examina apenas estados vizinhos imediatos ao corrente
  - não mantêm árvore de busca
  - armazena apenas estado corrente e tenta melhorá-lo
- Termina quando um pico (mínimo/máximo) é alcançado
  - nenhum vizinho tem um valor maior



Hill Climbing

Algoritmo

Função Hill-Climbing(Problema)
Início
  EstadoAtual ← EstadoInicial(Problema)
  Repita
    Vizinho ← SucessorMaiorValor(EstadoAtual)
    Se aval(Vizinho) <= aval(EstadoAtual) **então**
      retorna EstadoAtual // não consegue encontrar algo melhor na vizinhança
    EstadoAtual ← Vizinho
  Fim_Repita
Fim

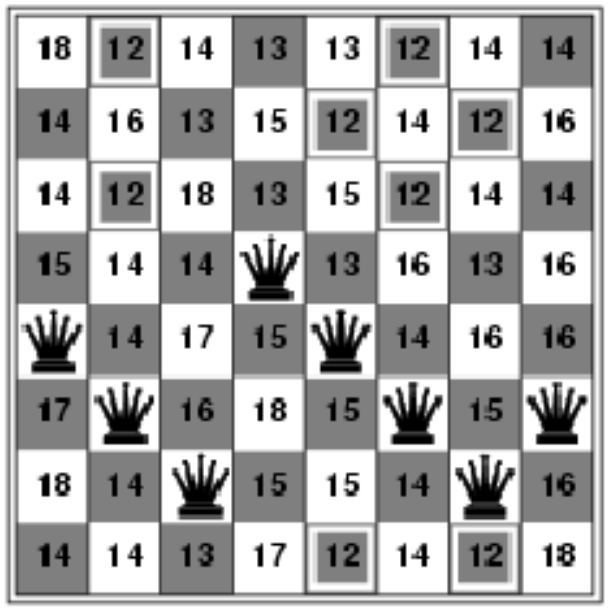
Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- **Formulação:** estados completos
  - cada estado tem as 8 rainhas no tabuleiro, uma por coluna
- **Ações:** mover cada rainha para qualquer posição na mesma coluna
- **Função Sucessora:** dado um estado e uma ação, retorna o estado alcançado
  - para cada estado, há 56 estados sucessores: 8 rainhas x 7 posições
  - escolhe melhor sucessor corrente (aleatório, se existir vários)
- **Função de custo/heurística h:** número de pares de rainhas que se atacam
- **Estado objetivo:** configuração com h=0 (mínimo global)
  - somente para as soluções perfeitas (nenhuma rainha sendo atacada)

Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- **Estado corrente:** h=17 (há 17 ataques entre rainhas)

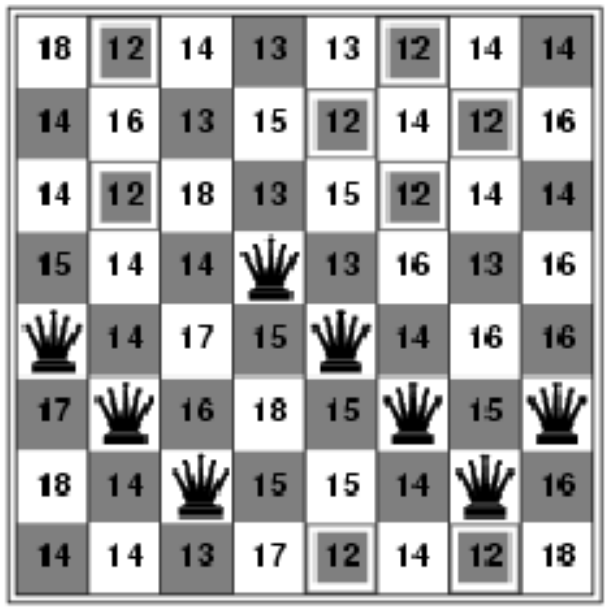
Qual a melhor ação (por uma escolha gulosa)?



Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- **Estado corrente:** h=17 (há 17 ataques entre rainhas)
- Em cada coluna é exibido o valor de h para cada possível sucessor

Qual a melhor ação (por uma escolha gulosa)?



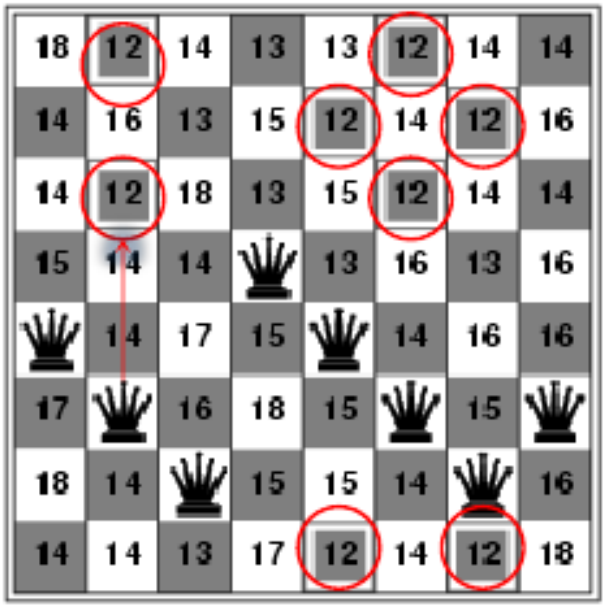
$$(C_1\,C_2\,C_3\,C_4\,C_5\,C_6\,C_7\,C_8) = (5\ 6\ 7\ 4\ 5\ 6\ 7\ 6)$$

Hill Climbing - Exemplo

- Problema: 8 rainhas

Resposta:

Qualquer movimento que leve uma das rainhas na sua coluna a uma posição marcada com 12



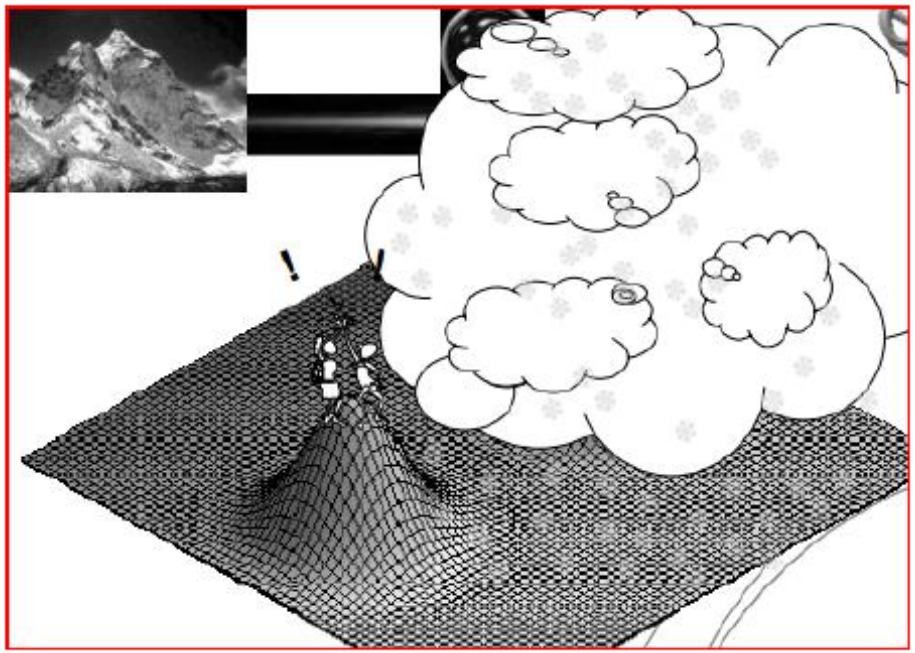
Hill Climbing

- **Busca Gulosa Local** - escolhe sempre o primeiro melhor vizinho para progredir na busca
- Bons resultados em alguns problemas
- Capaz de progredir rapidamente para solução do problema

Problemas ?

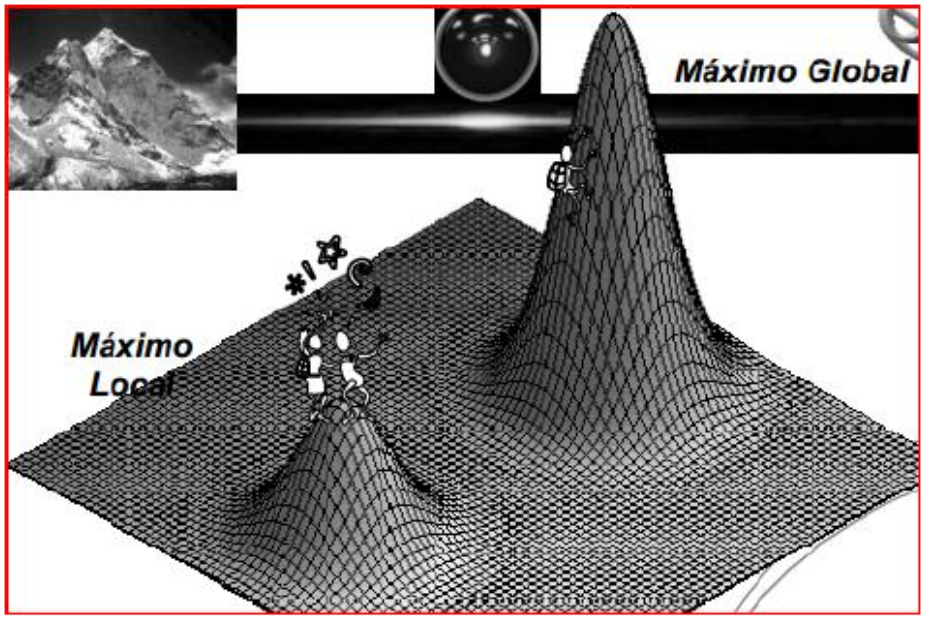
Hill Climbing

Problemas ?



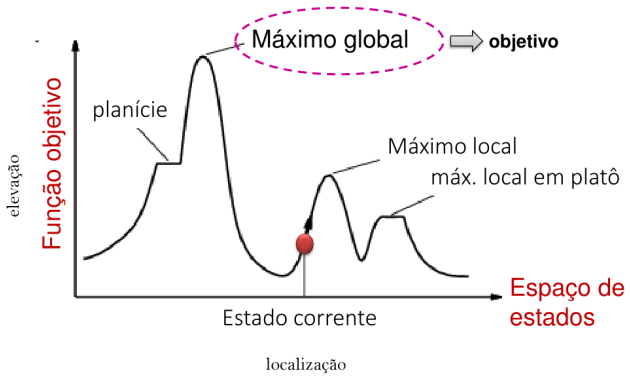
Hill Climbing - Problemas

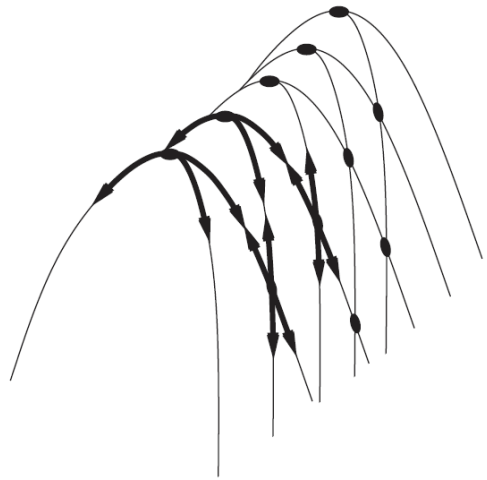
- **Máximos locais**



Hill Climbing - Problemas

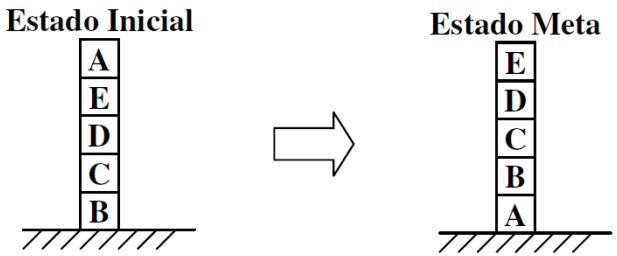
- **Máximos locais**
  - picos máximos/mínimos de abrangência local → não atingem máximo/mínimo global
- **Planícies/platôs**
  - região no espaço de busca → f(n) fornece mesmo resultado
- **Encostas e picos/cordilheiras/cristas**
  - região longa e estreita → sequência de máximos locais





Hill Climbing - Exemplo

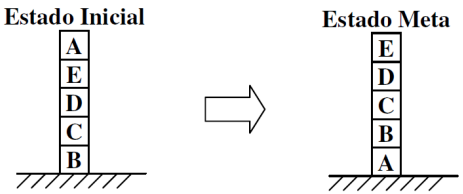
- Ações possíveis:
  - pegar um bloco e colocar ele sobre a mesa
  - pegar um bloco e colocar ele sobre outro bloco
- Heurística
  - $+1$ para cada bloco em cima do bloco onde ele deve estar
  - -1 para cada bloco em cima do bloco errado

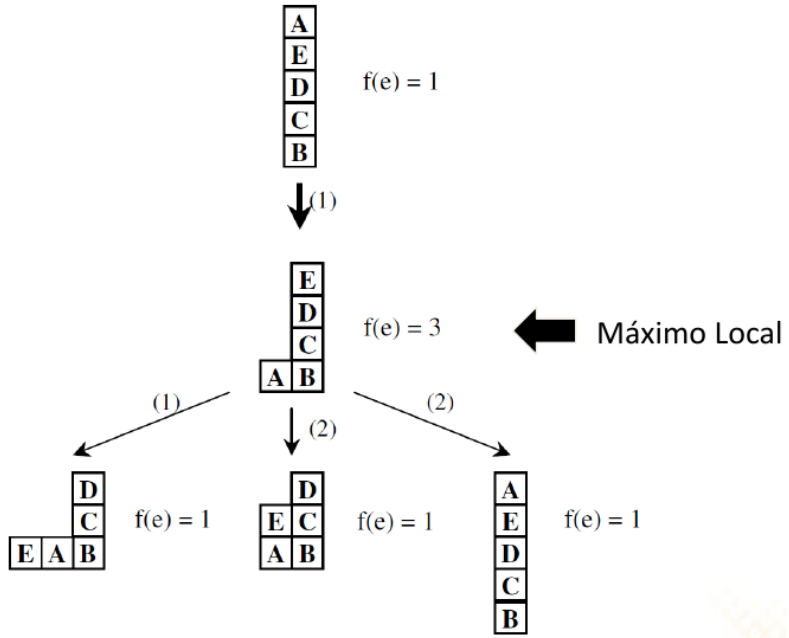


$f(e) = ?$

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

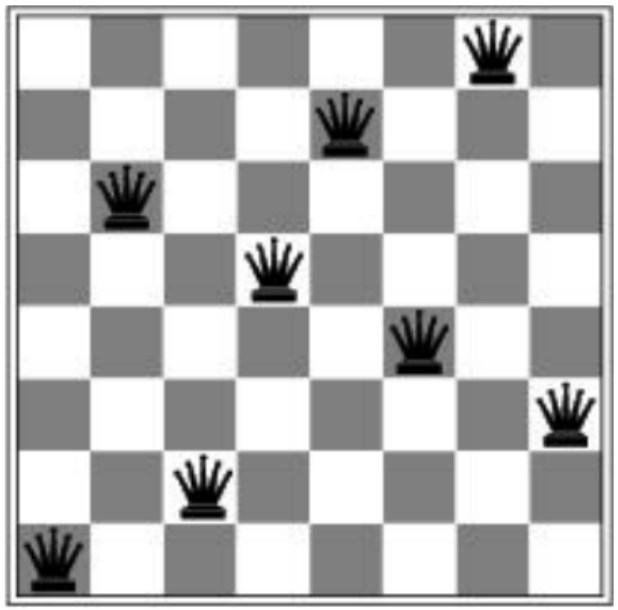




Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- **Estado Corrente:** alcançada em 5 passos a partir do estado inicial

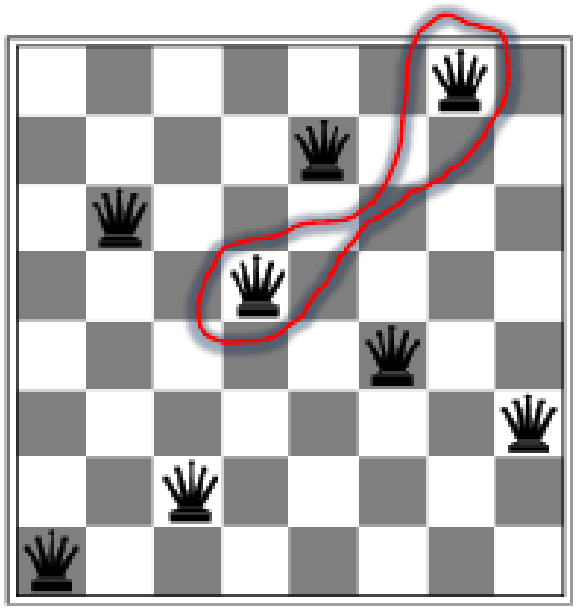
Mínimo Global?



Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- **Estado Corrente:** alcançada em 5 passos a partir do estado inicial

Mínimo Local: todo sucessor tem custo mais alto



Um mínimo local com $h = 1$

Hill Climbing - Exemplo

- **Problema:** 8 rainhas
- A partir de estados iniciais aleatórios
  - 86% $\rightarrow$ busca paralisada
  - 14% $\rightarrow$ busca bem sucedida
- No entanto, é rápida
  - 4 passos em média quando tem sucesso
  - 3 passos em média quando fica paralisada
- Em espaço que tem $\approx$17 milhões de estados

Hill Climbing - Solução

- **Problema:** mudar de estado em platôs (quando estados têm avaliações iguais)
  - pode ser uma boa ideia prosseguir, permitir um movimento lateral?
  - cuidado para evitar repetições infinitas
  - **solução:** considerar limite de movimentos laterais consecutivos permitidos
- Ex.: problema das 8 rainhas
  - permitindo até 100 movimentos laterais consecutivos
  - passa a resolver de 14% para 94% das instâncias do problema
  - no entanto, demora mais
    - 21 passos em média quando tem sucesso
    - 64 passos em média quando falha

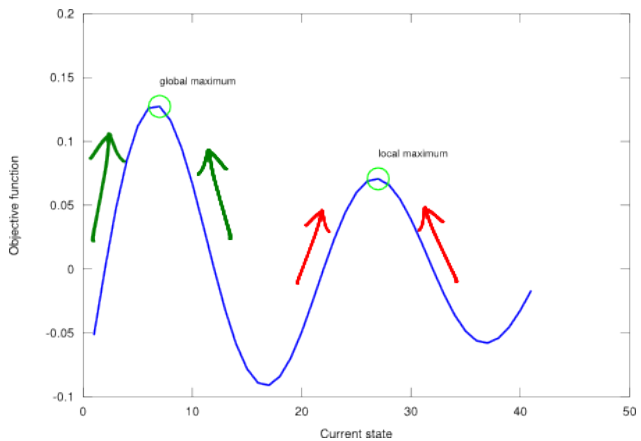
Hill Climbing - Variações

Diferentes versões:

- **Original (maior passo):** move-se para o estado vizinho que proporciona a maior melhoria (redução ou aumento) no valor da função objetivo
- **First-Choice:** move-se para algum estado vizinho sorteado aleatoriamente e que esteja entre aqueles (se existirem) que melhoram o valor corrente da função objetivo
  - interrompe a busca após um limite pré-especificado de tentativas de movimento frustradas
  - interessante em problemas nos quais a quantidade de vizinhos é grande

Hill Climbing - Variações

- **Random-Restart:** múltiplas buscas hill climbing, a partir de diferentes estados iniciais gerados aleatoriamente
  - tenta evitar mínimos/máximos locais
  - interessante em problemas com superfícies de otimização complexas (muitos mínimos ou máximos locais não satisfatórios)
- Cada busca é executada
  - até que um número máximo estipulado de iterações seja atingido, ou
  - até que os resultados encontrados não apresentem melhora significativa
- Algoritmo escolhe o melhor resultado obtido com as diferentes buscas (diferentes reinícios)
- Cada execução produz apenas uma solução!



Hill Climbing

- Sucesso depende da topologia do espaço de estados
- Se houver poucos máximos locais e platôs
  - Hill climbing com reinício aleatório encontrará boa solução com rapidez
  - mesmo para mais complexos, pode encontrar máximo local razoavelmente bom
    - ★ com um pequeno número de reinícios

Hill Climbing

- Problema 1: Hill Climbing nunca realiza movimentos "encosta abaixo" em direção a estados com valor mais baixo (ou de custo mais alto)
  - **incompleto** $\rightarrow$ porque pode ficar preso em um máximo local
- Problema 2: percurso aleatório, movendo para um sucessor escolhido ao acaso a partir do conjunto de sucessores
  - **completo** $\rightarrow$ extremamente ineficiente
- Solução: ?

Hill Climbing

- Problema 1: Hill Climbing nunca realiza movimentos "encosta abaixo" em direção a estados com valor mais baixo (ou de custo mais alto)
  - **incompleto** $\rightarrow$ porque pode ficar preso em um máximo local
- Problema 2: percurso aleatório, movendo para um sucessor escolhido ao acaso a partir do conjunto de sucessores
  - **completo** $\rightarrow$ extremamente ineficiente
- Solução: combinar Hill Climbing com um percurso aleatório que resulte de algum modo em eficiência e completeza
  - Simulated Annealing

Simulated Annealing

- Têmpera simulada
- Oferece meios para se escapar de ótimos locais (mínimos/máximos)

Metáfora 1

- Metalurgia $\rightarrow$ têmpera de metais e de vidro
- Aquecimento em alta temperatura e posterior resfriamento gradual

Metáfora 2

- Colocar uma bola na fenda mais profunda em uma superfície acidentada
- Agitar a superfície com força suficiente para fazer com que a bola saia de mínimos locais e evitar desalojá-la do mínimo global

Solução

- Começar a agitar com força (alta temperatura) e depois reduzir gradualmente a intensidade da agitação (baixa temperatura)

Simulated Annealing

- Agitação == aleatoriedade
- Semelhante à Hill-Climbing $\rightarrow$ diferenças
  - movimentos encosta abaixo são permitidos
  - não escolhe o melhor estado sucessor, escolhe um movimento aleatório
  - se o movimento aleatório melhora a situação, ele é aceito
  - senão aceita-se um movimento com alguma probabilidade < 1
    - ★ probabilidade decresce exponencialmente com a "má qualidade" do movimento
    - ★ probabilidade decresce a medida que a temperatura reduz (movimentos ruins têm maior probabilidade de serem permitidos no início e tornam mais improváveis conforme a temperatura diminui
- Se a temperatura decresce de maneira suficientemente lenta, a busca por têmpera simulada encontrará um ótimo global com probabilidade próxima de 1 (100%)
- Na prática, decisão da taxa de redução da temperatura $\rightarrow$ problema prático com este método

Simulated Annealing

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"
  local variables: current, a node
                   next, a node
                   T, a "temperature" controlling prob. of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]   // T schedule é a função que retorna T dado um tempo t
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← VALUE[next] – VALUE[current]
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(Δ E/T)   // faz uma ação ruim de tempos
                                                            // em tempos
```

Simulated Annealing

- Considerar três movimentos possíveis, com mudanças na função objetivo: $d_1 = -0.1$, $d_2 = 0.5$, $d_3 = -5$ e $T = 1$
- Escolher um movimento aleatório
  - se d_2 é escolhido, aceita o movimento
  - se d_1 ou d_3 é escolhido, verifica probabilidade do movimento
  - probabilidade = exp(d/T)
  - movimento 1: prob1 = exp(-0.1) = 0.9
    - ★ 90% das vezes aceita o movimento
  - movimento 3: prob3 = exp(-5) = 0.05
    - ★ 5% das vezes aceita o movimento

Beam Search

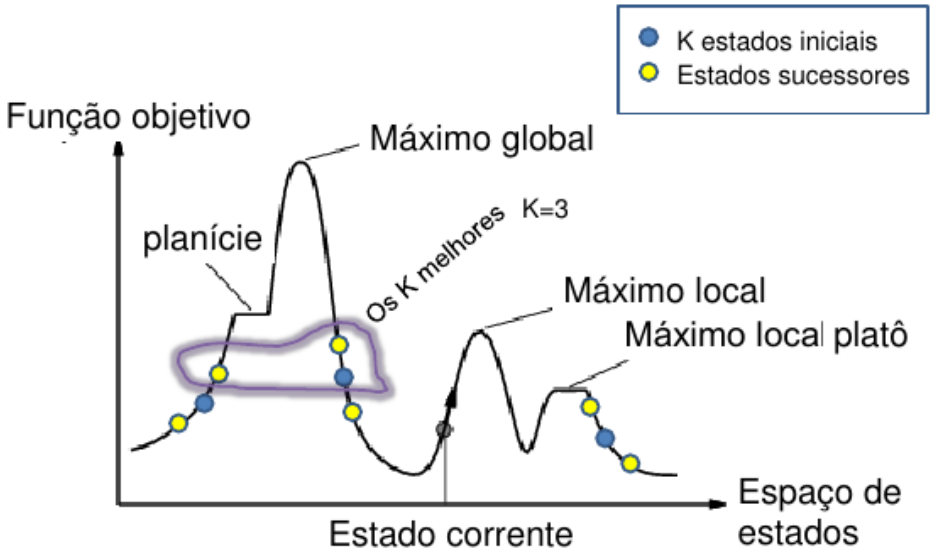
Manter apenas um estado na memória $\rightarrow$ abordagem extrema para resolver problema de limitação de memória

- Beam Search ou Busca em feixe local
- Estratégia mantém o controle de k estados por vez
  1. Inicia com k estados gerados aleatoriamente
  2. Expande os sucessores de cada um dos k estados
  3. Se objetivo, sucesso
  4. Senão, seleciona os k melhores dentre todos os estados e repete o processo

Beam Search

Busca com reinício aleatório $\neq$ beam search?

- Busca com reinício aleatório $\rightarrow$ cada busca é independente
- Beam Search $\rightarrow$ informações úteis são repassadas entre os k processos paralelos da busca
  - estados "conversam" e tendem a abandonar caminhos infrutíferos
  - deslocam recursos para o processo em que estiver sendo realizado maior progresso



Qual Algoritmo Usar?

Referências e Leituras Complementares

- Cap. 04 $\rightarrow$ livro Russel e Norvig
- Cap. 05 $\rightarrow$ livro Ben Coppin