Inteligência Artificial

Estratégias de busca local

Prof. Chauã Queirolo

Sumário

Introdução

Introdução

- Um problema pode ser definido por 5 componentes
  - Estado inicial
  - Ações
  - Modelo de transição
  - Teste de objetivo
  - Custo do caminho

Introdução

- Uma solução é uma sequência de ações que levam do estado inicial para o estado objetivo

- Uma solução ótima é uma solução com o menor custo de caminho

Algoritmo Geral de Busca em árvore

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
        fringe ← INSERTALL(EXPAND(node, problem), fringe)

---

function EXPAND( node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node;  ACTION[s] ← action;  STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Busca local

Busca local

Definição

- Em muitos problemas de otimização o caminho para o objetivo é irrelevante

- Queremos apenas encontrar o **estado objetivo**

- Não importa a sequência de **ações**

Busca local

Definição

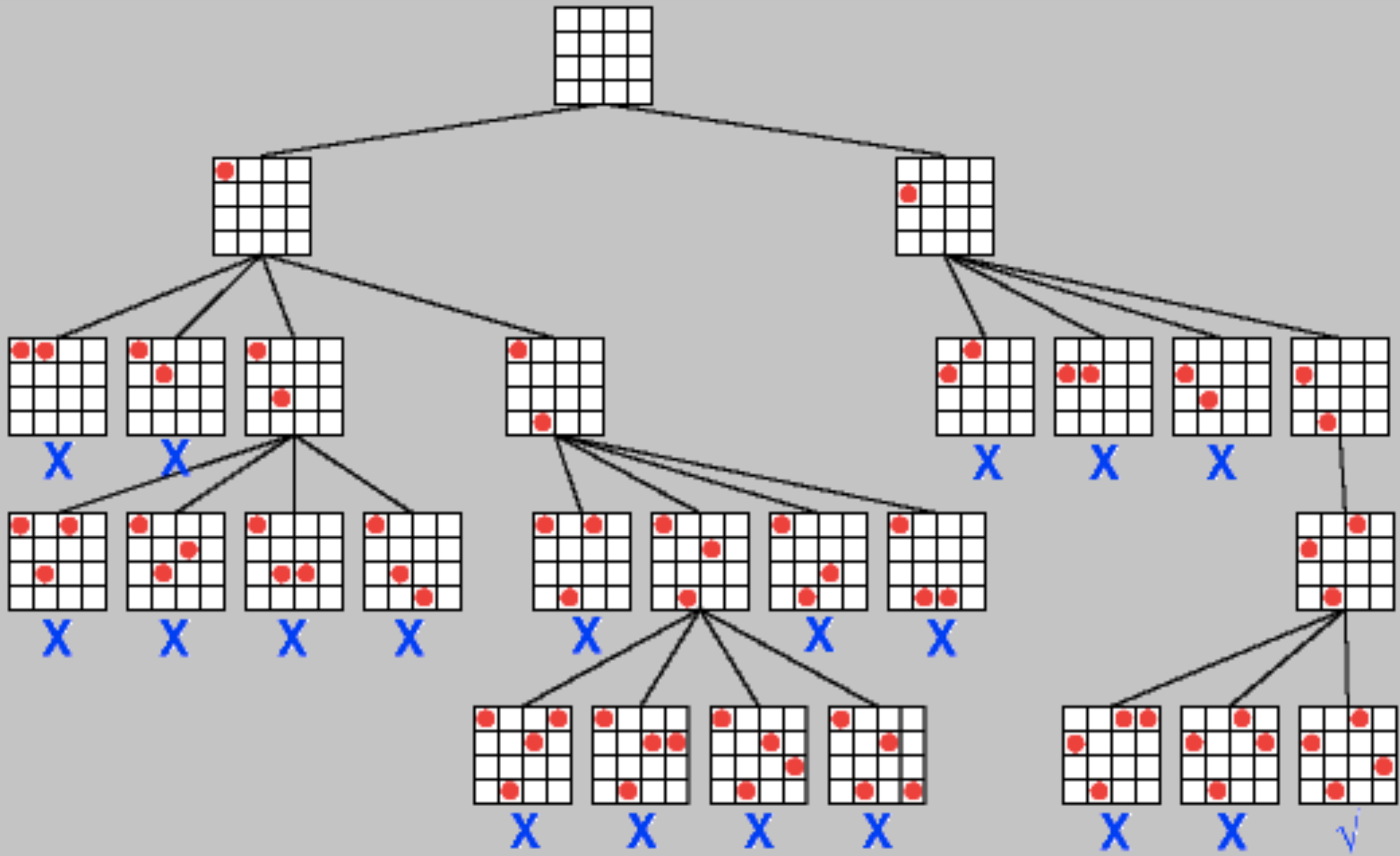
- Algoritmos de **busca local**:

  - Mantêm apenas o estado atual

  - Sem a necessidade de manter a árvore de busca

Problema das n-Rainhas

Colocar n rainhas em um tabuleiro $n \times n$, sendo que cada linha coluna ou diagonal pode ter apenas uma rainha



Hill Climbing

“É como subir o Everest em meio a um nevoeiro durante uma crise de amnésia”

© Cory Richards; Getty - https://www.redbull.com/fr-fr/everest-dangers-ascension

Busca local

Hill Climbing

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

current ← MAKE-NODE(INITIAL-STATE[problem])
loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
    current ← neighbor
```

Busca local

Hill Climbing

- Elevação

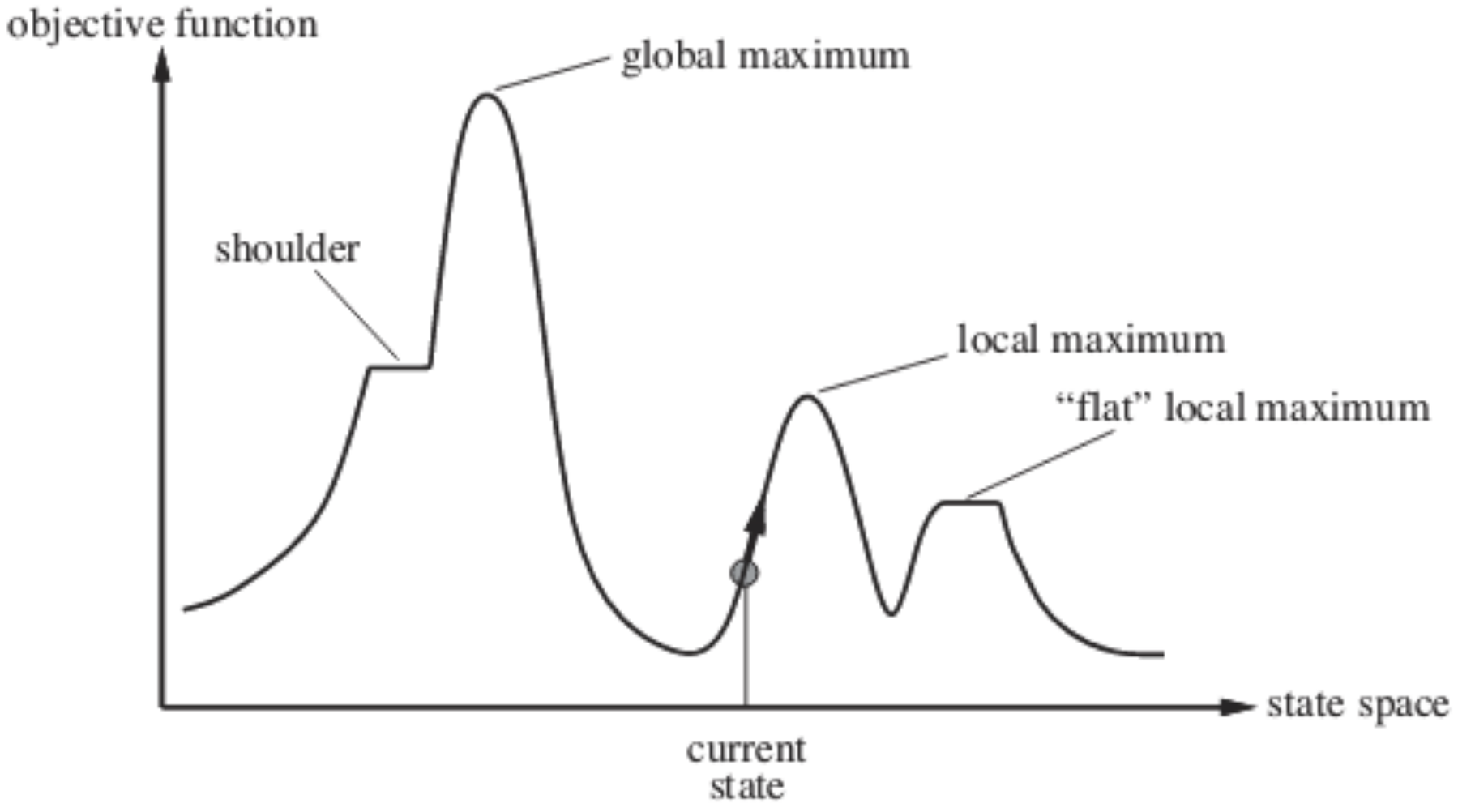
  - Função objetivo: queremos encontrar o máximo global
  - Custo: queremos encontrar o mínimo global

- O algoritmo consiste em uma repetição que percorre o espaço de estados no sentido do valor crescente (ou decrescente)

- Termina quando encontra um pico (ou vale) em que nenhuma vizinho tem valor mais alto

Busca local

Hill Climbing



Busca local

Hill Climbing

- Não mantém uma árvore, o nó atual só registra o estado atual e o valor da função objetivo

- Não examina antecipadamente valores de estados além dos valores dos vizinhos imediatos do estado atual

- **Problema:** dependendo do estado inicial pode ficar presa em máximos (ou mínimos) locais