

COMP0271: Inteligência Artificial

Busca local

Professor: Hendrik Macedo

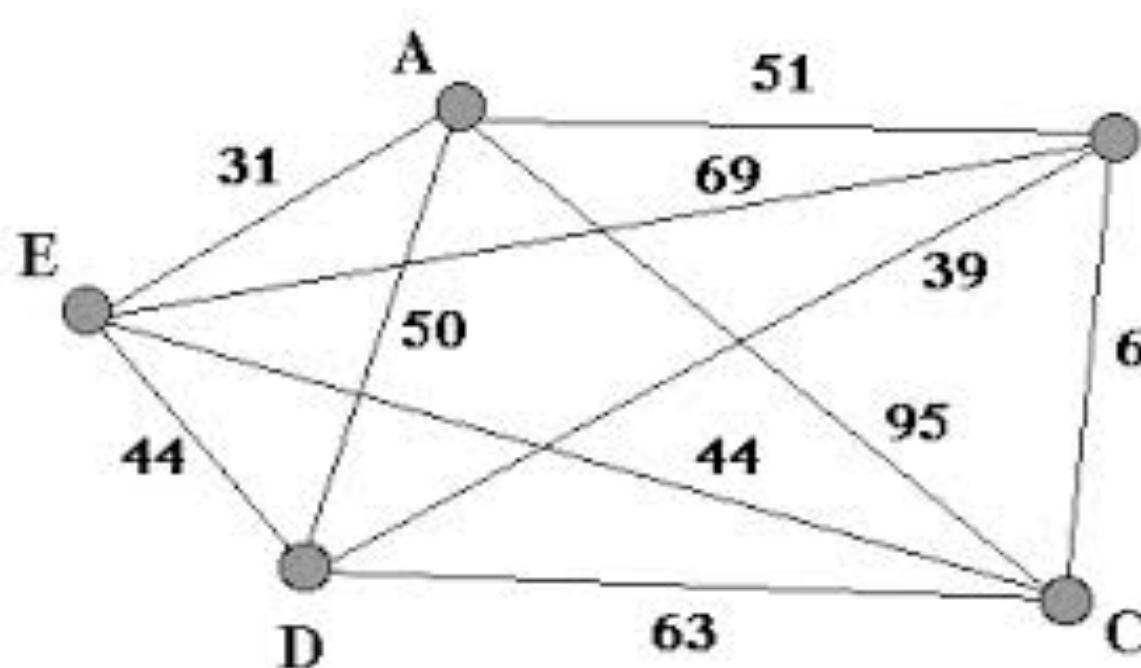
Universidade Federal de Sergipe, Brasil

Qual a solução do problema?



Problema do Caixeiro Viajante

- Representação por um grafo completo (todas as cidades são conectadas).



$(n-1) \times (n-2) \times \dots \times 2 \times 1 = (n-1)! \text{ caminhos diferentes !!!}$

Uma forma de modelar o problema?

Problema de Otimização (ou Busca local)

- Entrada: uma possível solução
- Saída: a melhor solução
- Mecanismo otimizador: dado um modelo que me possibilita **medir a qualidade** de uma solução, o mecanismo busca encontrar a melhor solução.

Otimização Escalar

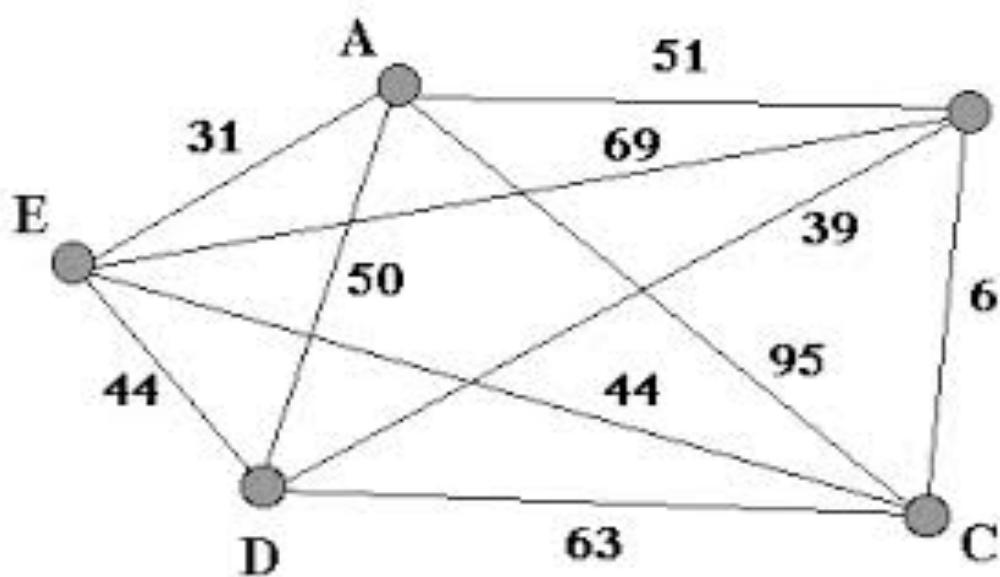
- Geralmente, os parâmetros do sistema (as variáveis) são definidas através de um vetor.
- Para um problema de minimização temos:

Seja $x \in \mathbb{R}^n$

$$f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$$

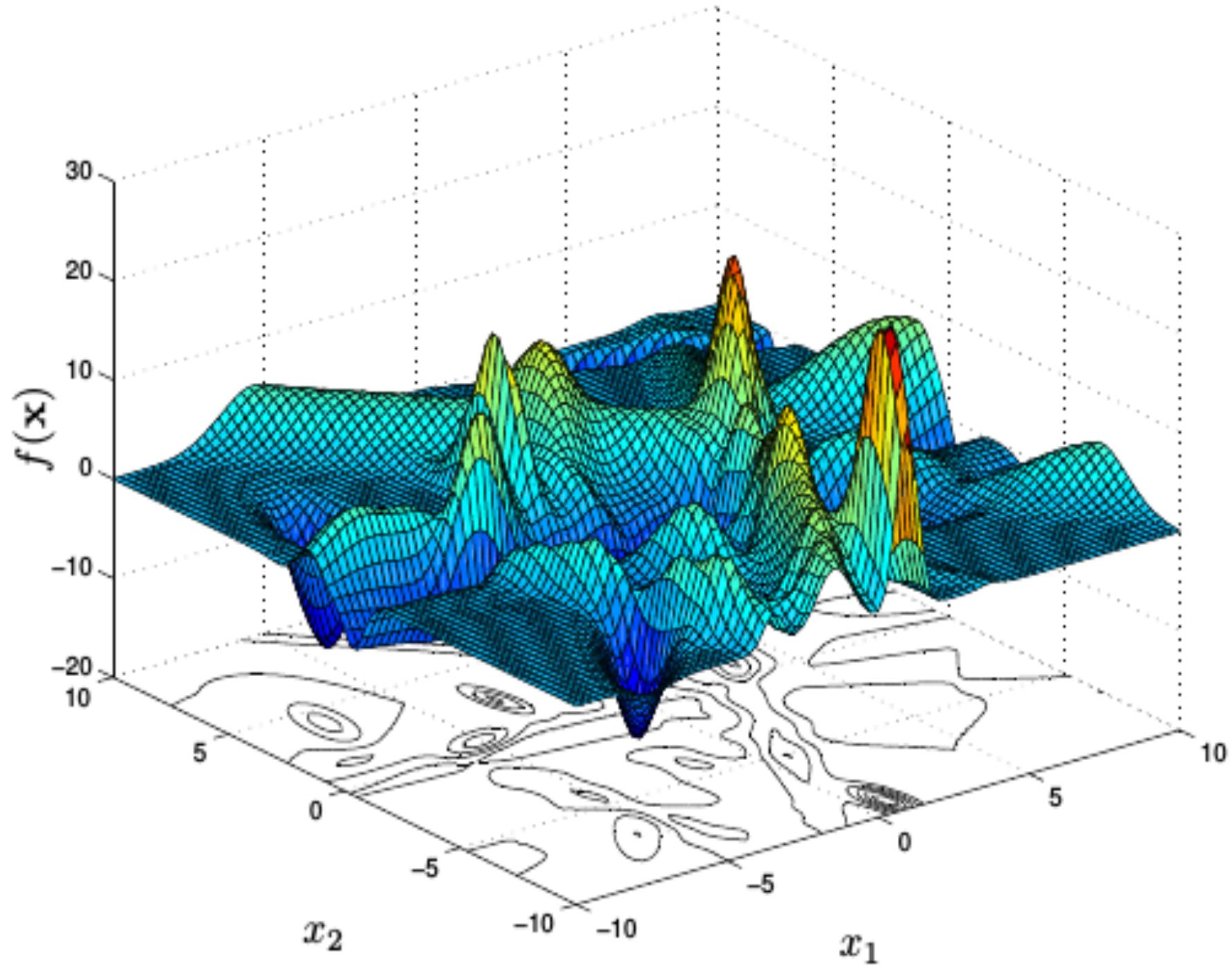
$$x^* = \arg \min_x f(x)$$

Otimização Escalar



$$f(x) = \left(\sum_1^{n-1} dist(x_i, x_{i+1}) \right) + dist(x_n, x_1)$$

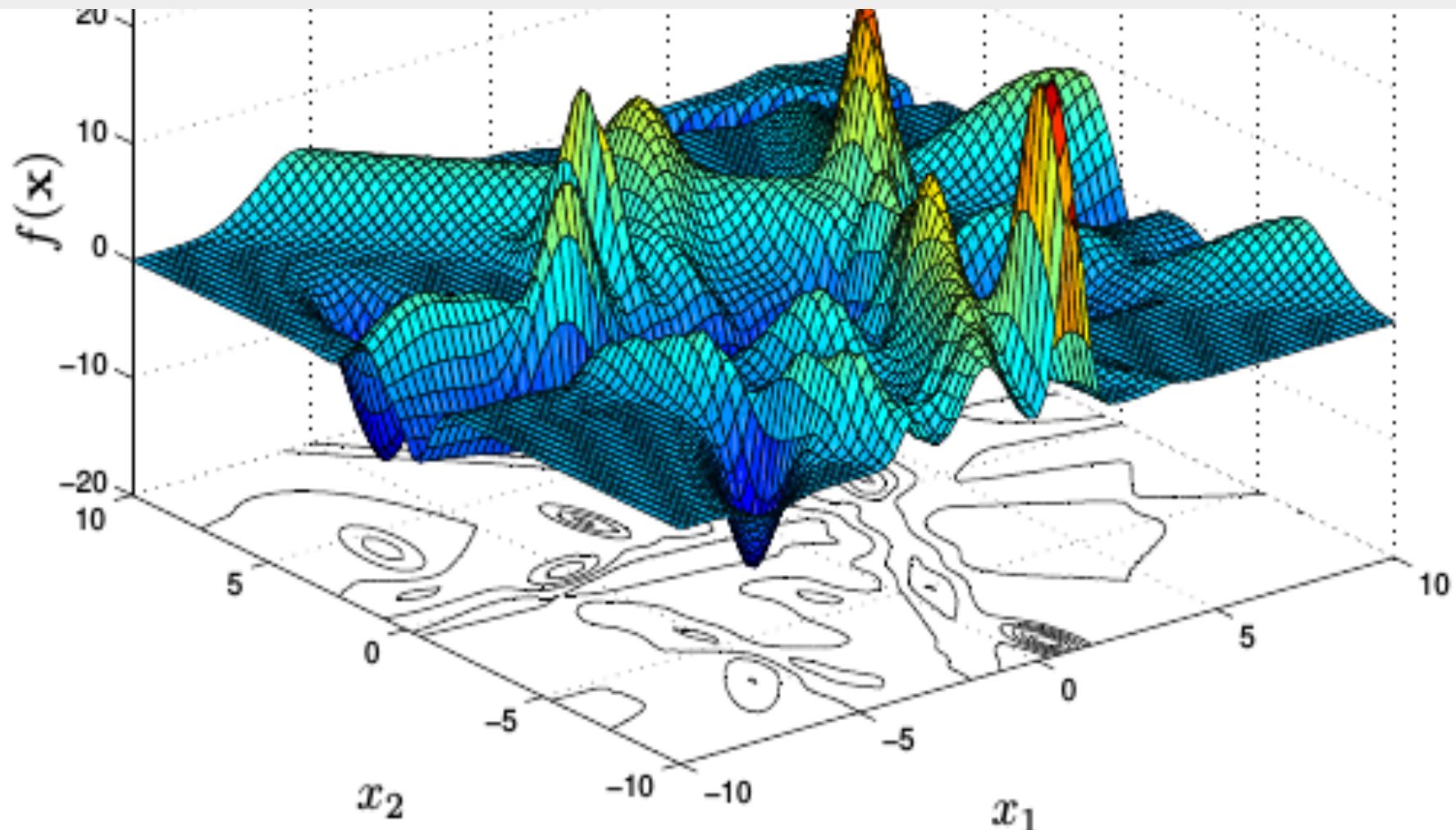
$$x^* = \arg \min_x f(x)$$



Metáfora:

- Otimizador **cai de paraquedas** em algum ponto
- Só possui informação do ponto** (altura = qualidade) e mais nada
- Guarda informações do ponto que já visitou
- Busca o ponto mínimo*** no menor número de passos

*pode ser máximo,
a depender do problema



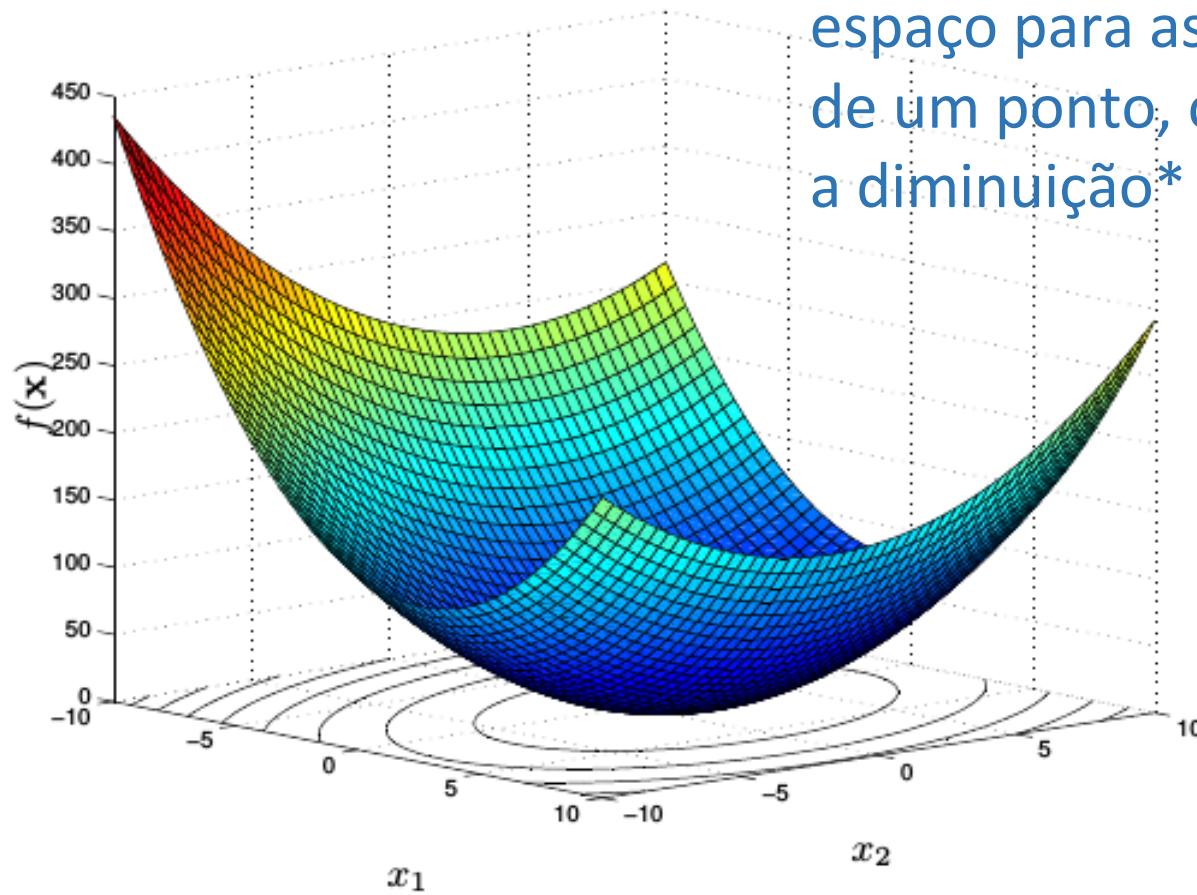
Estratégia de busca local

- Um mecanismo otimizador busca uma **estratégia para caminhar** em direção ao ótimo.
 1. Estratégia de direção de busca.
 2. Estratégia populacional.

Estratégia de direção de Busca

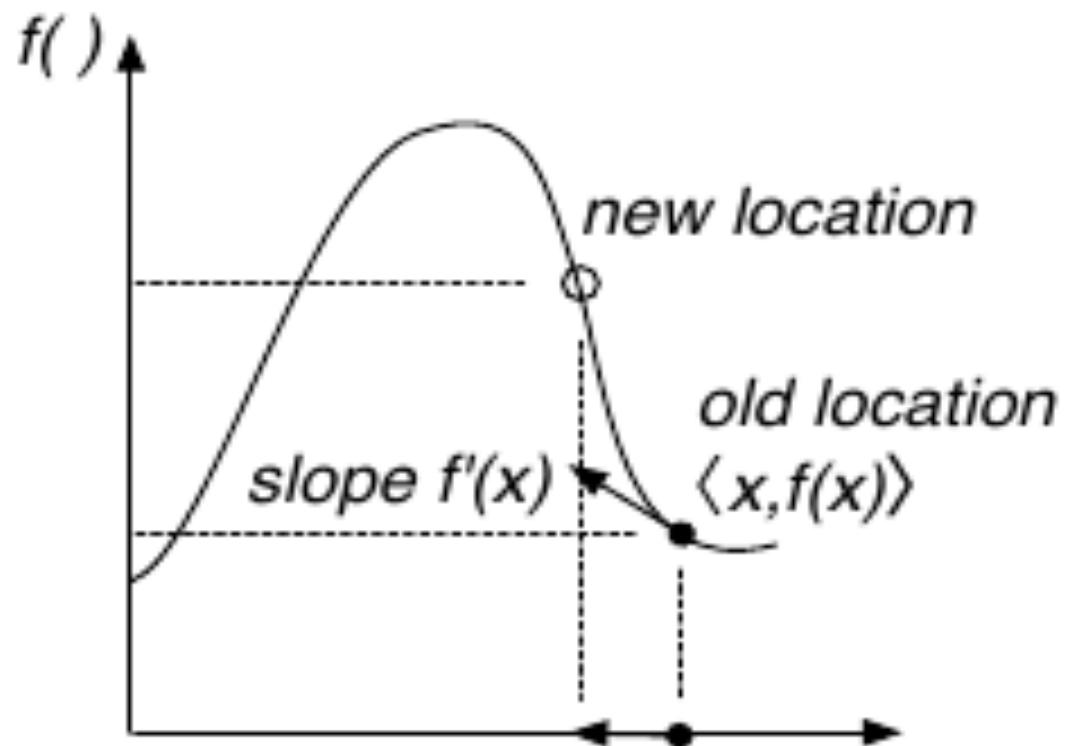
Método do Gradiente Indica as direções do espaço para as quais, partindo-se de um ponto, ocorre localmente a diminuição* da função.

*ou aumento



Estratégia de direção de Busca

Método do Gradiente



Estratégia de direção de Busca

Método do Gradiente

- Critérios de parada
 - Estabilização da função objetivo.
 - Estabilização do vetor de otimização.
 - Anulação do vetor gradiente.
 - Número máximo de iterações/avaliação da função objetivo.

Métodos *Single-State*

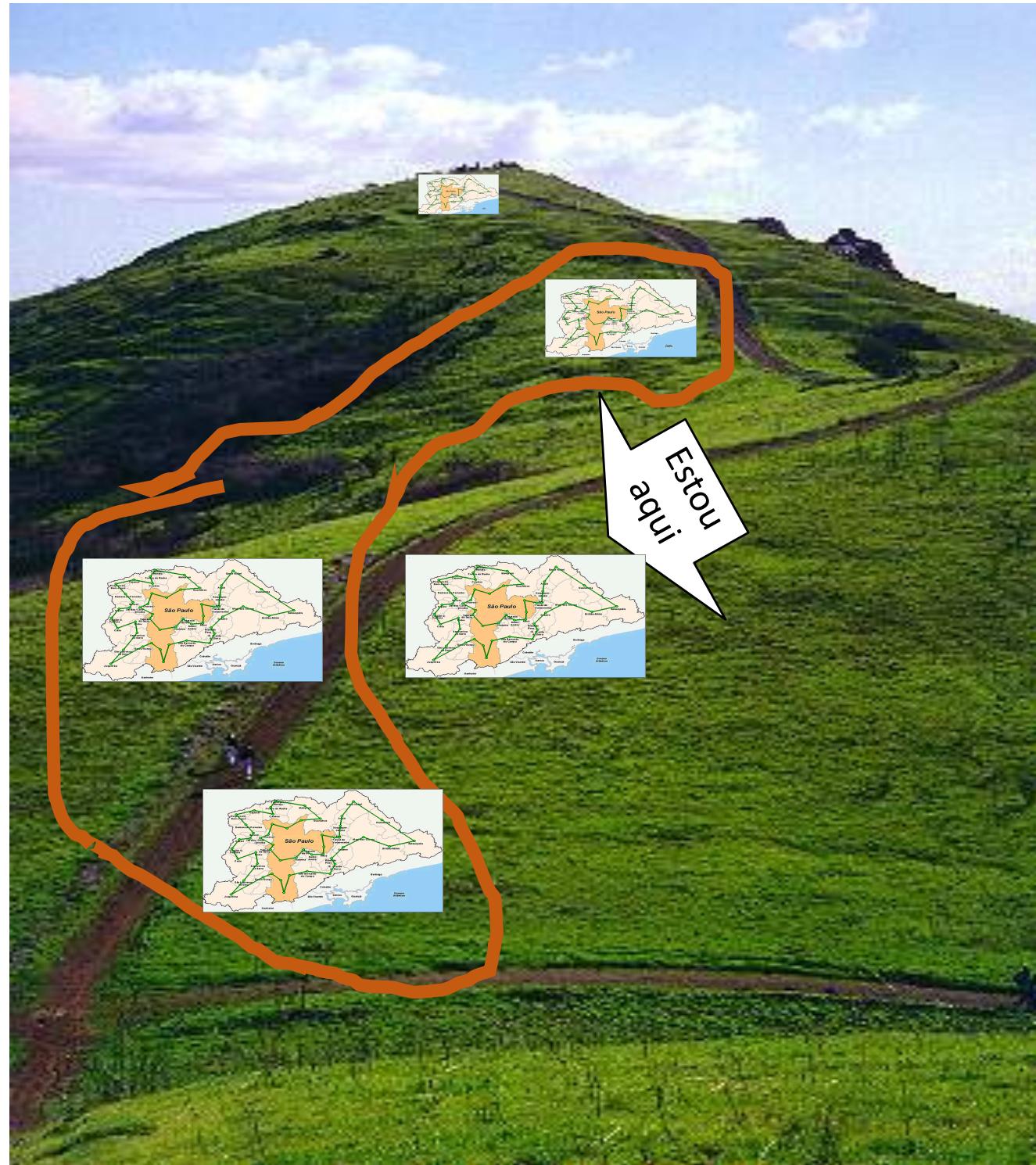
- Procedimento de inicialização.
- Avaliação (qualidade) da função objetivo.
- Um procedimento de cópia da solução.
- Gerar uma ou mais soluções na vizinhança (amostragem do gradiente - *Tweak*)
 - Não precisamos calcular explicitamente o valor do gradiente!!!

Hill-Climbing

Testamos soluções vizinhas e escolhemos aquela que tem o melhor valor da função objetivo.

Assim, subimos a encosta (ou descemos) da nossa função em direção do ótimo

Vizinhança = **fronteira** do espaço de estados



EXPLOITATION
da melhor solução atual

vs. **EXPLORATION**
do espaço de busca

Hill-Climbing

...

UCS
Greedy
A*

...

Algorithm 5 Steepest Ascent Hill-Climbing

```
1:  $n \leftarrow$  number of tweaks desired to sample the gradient  
2:  $S \leftarrow$  some initial candidate solution  
3: repeat  
4:    $R \leftarrow \text{Tweak}(\text{Copy}(S))$   
5:   for  $n - 1$  times do  
6:      $W \leftarrow \text{Tweak}(\text{Copy}(S))$   
7:     if  $\text{Quality}(W) > \text{Quality}(R)$  then  
8:        $R \leftarrow W$   
9:     if  $\text{Quality}(R) > \text{Quality}(S)$  then  
10:       $S \leftarrow R$   
11: until  $S$  is the ideal solution or we have run out of time  
12: return  $S$ 
```

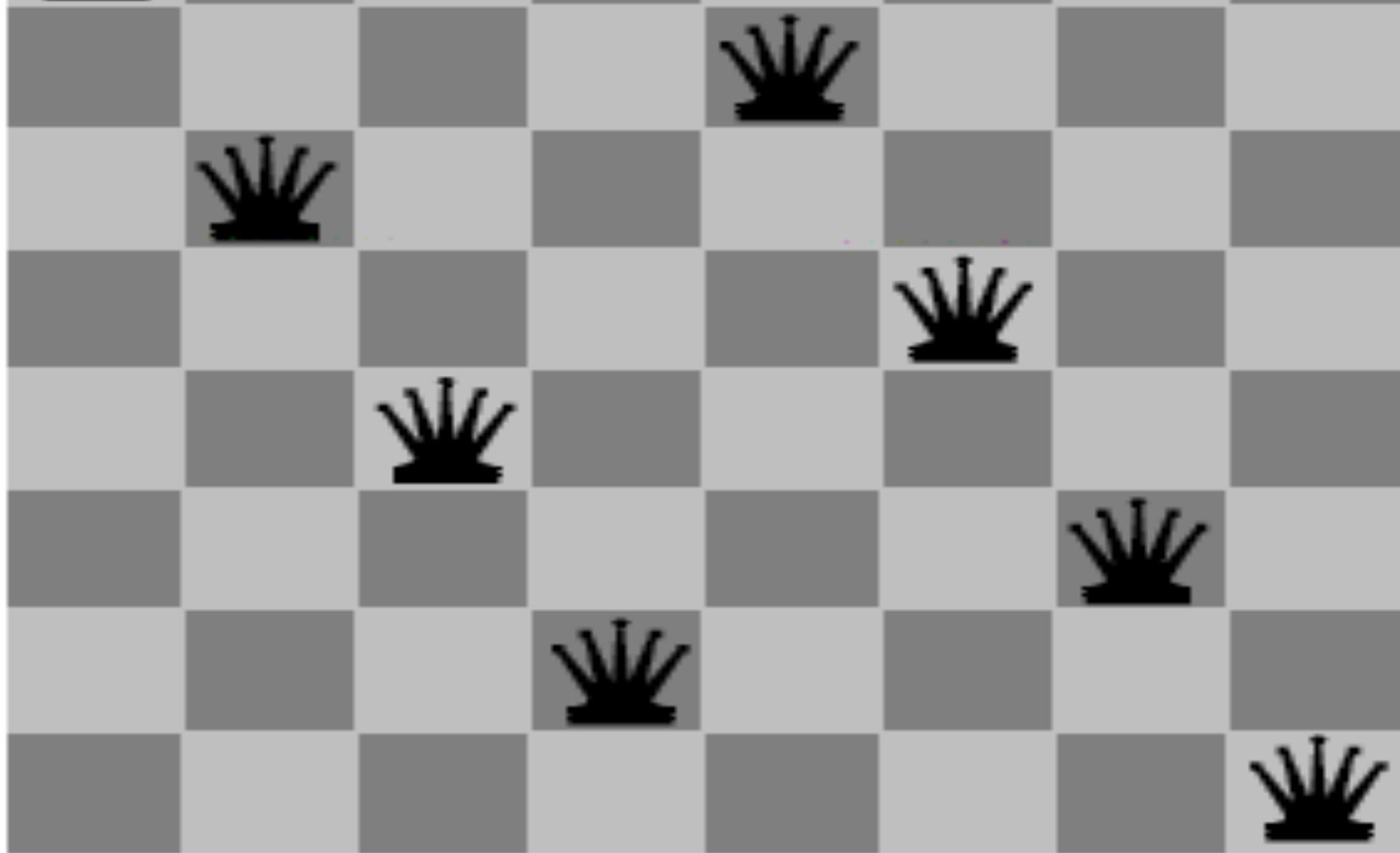
Geralmente de forma aleatória.

Seleciona o melhor vizinho entre n possíveis

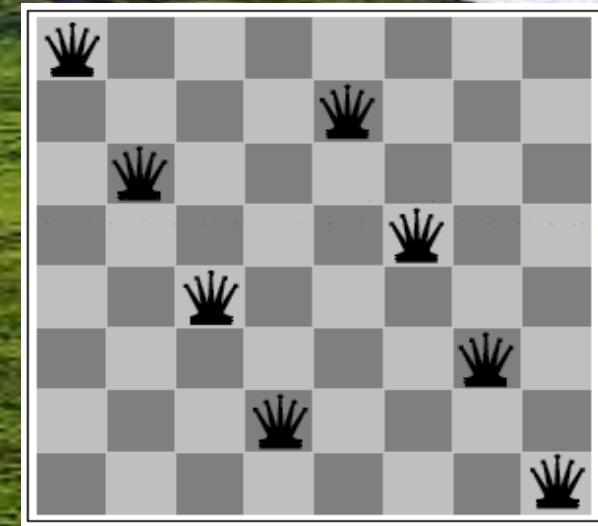
Comparo a qualidade com o ponto original



Qual a solução do problema?



Hill-Climbing



Ação: mover uma rainha para outro quadrado na mesma coluna

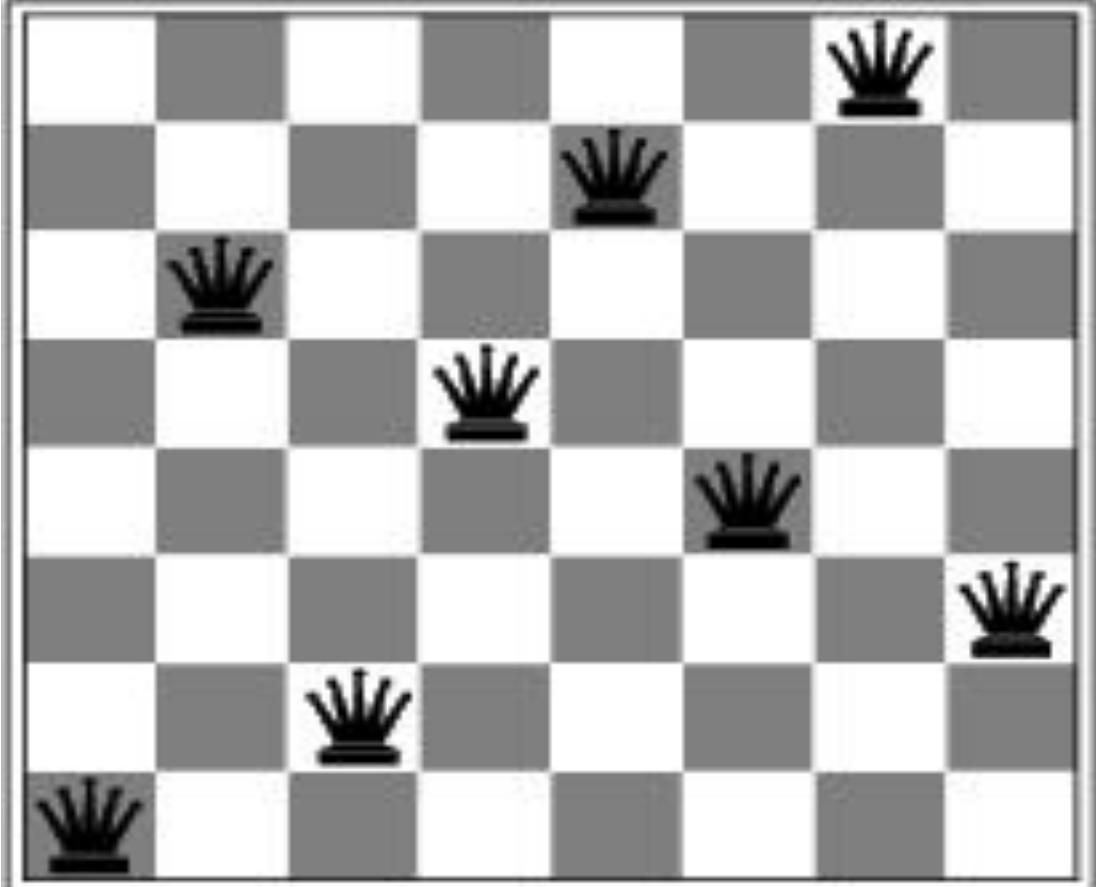
f(x): número de pares de rainhas que se atacam

Objetivo: minimizar $f(x)$ (*na verdade, $f(x) = 0$*)

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

Estado corrente tem $f=17$

Figura ilustra demais valores de f para cada possível estado sucessor

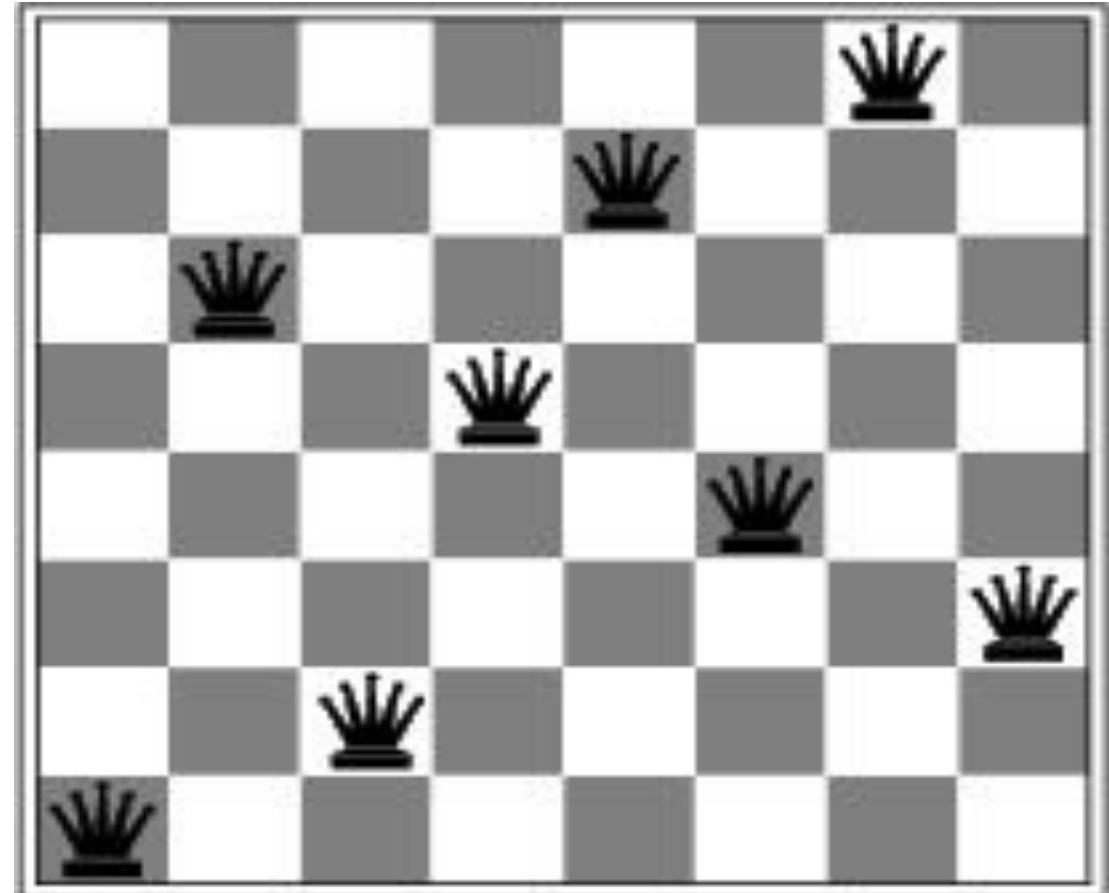


Mínimo local ($f=1$)

não existe mais ações que minimizem a função

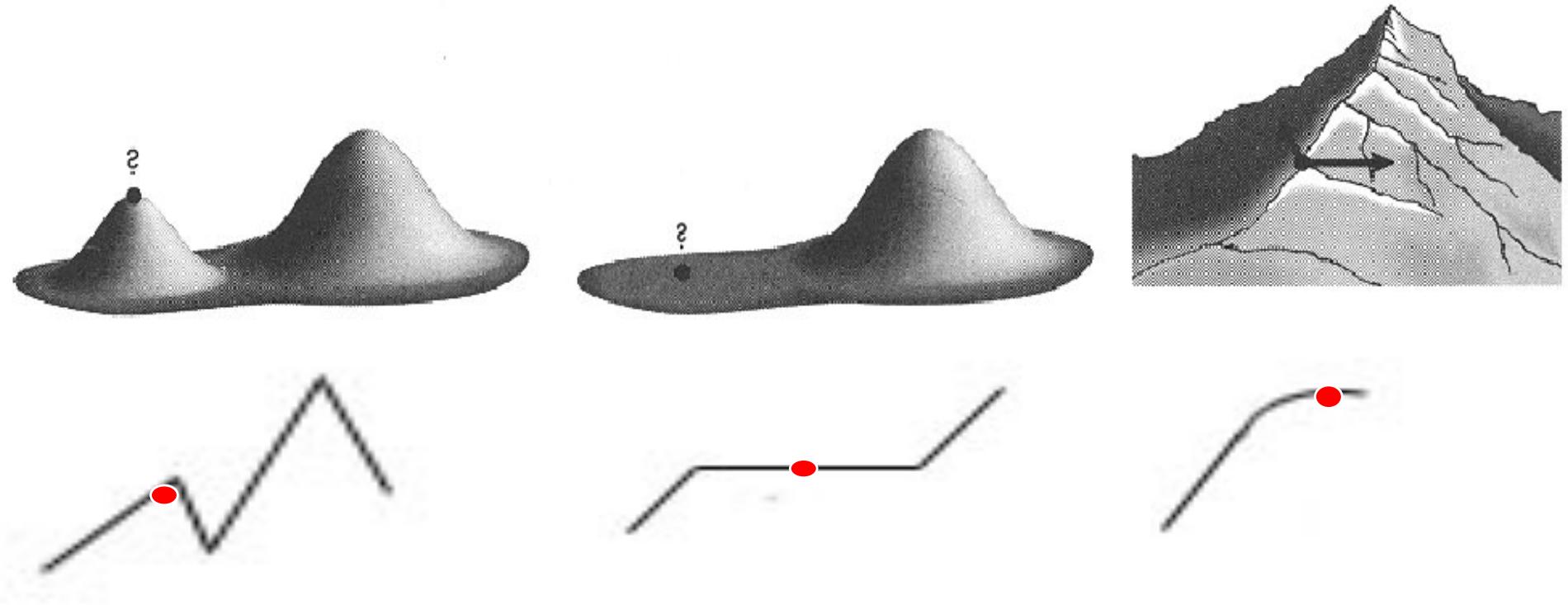
Sobrou EXPLOITATION
Faltou EXPLORATION

Importante:
combinação das duas coisas!



Mínimo local ($f=1$)
ñ existe mais ações que minimizem a função

Limitações do Hill-Climbing:



Experimentação p/ 8-rainhas:

14% de sucesso p/ início aleatório

4 passos quando obtém sucesso

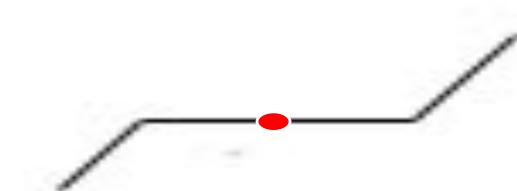
3 passos quando fracassa

Solução 1: permitir movimentos laterais (p/ platôs)

Experimentação p/ 8-rainhas: 14% → 94%

4 → 21 passos quando obtém sucesso

3 → 64 passos quando fracassa



Solução 2: reiniciar a busca a partir de estados iniciais aleatórios

Se p = probabilidade de sucesso de cada busca

=> $1/p$ = # de reinícios necessários para sucesso

Experimentação p/ 8-rainhas (sem mov. Laterais):

$p = 14\% \Rightarrow 1/0.14 = 7.14$ reinícios necessários (execuções)

$1 * 4 + 6.14 * 3 = 22$ passos

Experimentação p/ 8-rainhas (com mov. Laterais):

$p = 94\% \Rightarrow 1/0.94 = 1.06$ reinícios necessários (execuções)

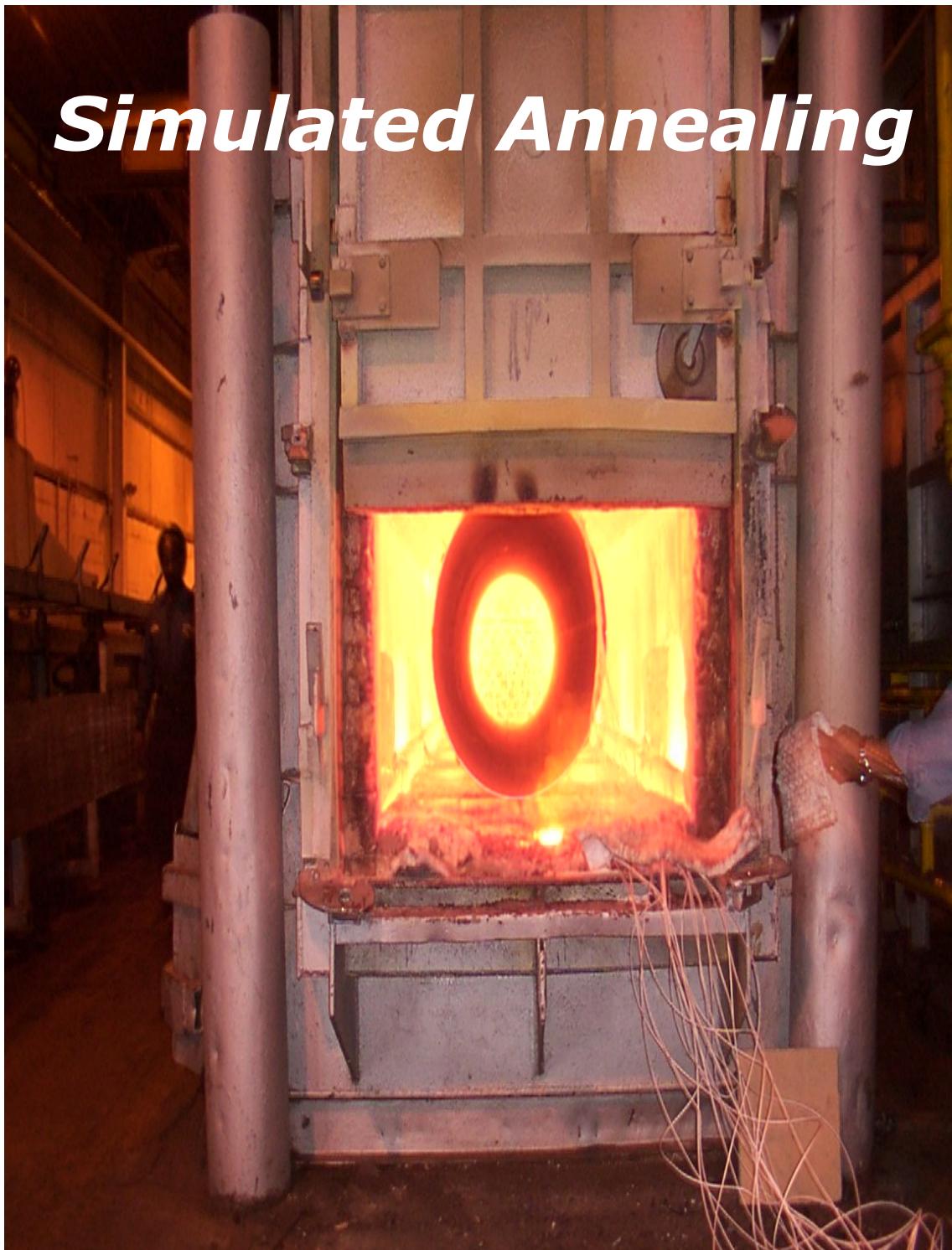
$1 * 21 + 0.06 * 64 = 25$ passos

Pura Subida de Encosta ==> incompleto

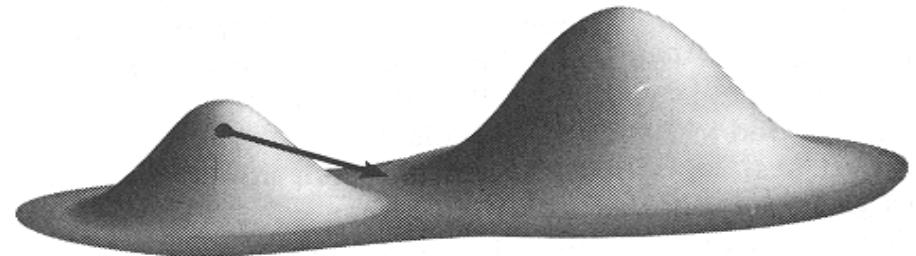
Caminhada aleatória pura ==> ineficiente

Solução 3?

Simulated Annealing



Permitir retrocesso para situações piores...



...com certa probabilidade que diminui com o tempo

$$e^{-\Delta E/T}$$

$$\Delta E = \text{Avaliação}[\text{próximo_nó}] - \text{Avaliação}[\text{nó_atual}]$$

$$e = \text{número de Euler (neperiano)}$$

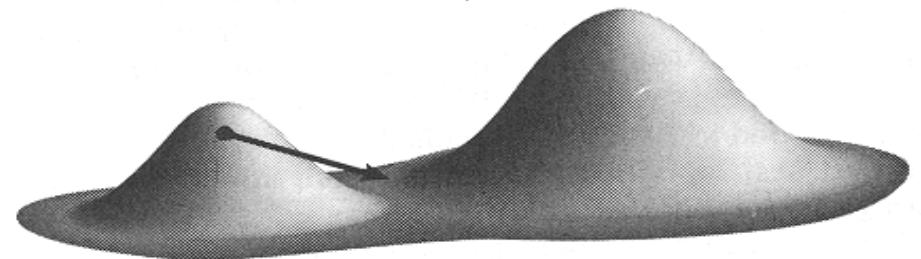
$$T = \text{Temperatura} \rightarrow 0$$

Simulated Annealing



Distribuição de *Boltzmann*

Se $\Delta E < 0$ então
se $e^{\Delta E/T} > \text{random } [0, 1]$ (*)
permite retrocesso !



(*) OU
 $\Delta E > T * \ln(\text{random } [0, 1])$

```

function SIMULATED-ANNEALING(problem, schedule) return a
solution state
input: problem, a problem
          schedule, a mapping from time to temperature
local variables: current, a node.
                    next, a node.
                    T, a "temperature"
current  $\leftarrow$  MAKE-NODE(INITIAL-STATE[problem])
for t  $\leftarrow$  1 to  $\infty$  do
    T  $\leftarrow$  schedule[t]  $T = c^*T$  ( $c = 0.8$ )
    if T = 0 then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow$  VALUE[next] - VALUE[current]
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 

```

Experimentos TSP ... 9 cidades

Busca exaustiva

Ordem: (3, 5, 8, 2, 4, 6, 7, 1, 0) Distancia: 3.95178487145
Tempo: 1.9409711360931396

Greedy

Ordem: [4 6 7 1 8 2 5 3 0] Distancia: 4.69961784274
Tempo: 0.00012803077697753906

Hill Climbing

Ordem: [1 7 6 4 2 8 5 3 0] Distancia: 4.41161116688
Tempo: 0.015461206436157227

Simulated Annealing

Ordem: [3 5 8 2 4 6 7 1 0] Distancia: 3.95178487145
Tempo: 0.02221202850341797

Experimentos TSP ...: 10 cidades

Busca exaustiva

Ordem: (8, 3, 5, 1, 6, 7, 9, 2, 4, 0) Distancia: 4.14957793323
Tempo: 22.220077991485596

Greedy

Ordem: [0 4 2 9 7 6 1 5 3 8] Distancia: 6.1319691999
Tempo: 8.606910705566406e-05

Hill Climbing

Ordem: [1 5 8 3 6 7 9 2 4 0] Distancia: 4.67509808889
Tempo: 0.016817808151245117

Simulated Annealing

Ordem: [8 3 5 1 6 7 9 2 4 0] Distancia: 4.14957793323
Tempo: 0.021317005157470703



<http://www.saense.com.br>

<https://www.facebook.com/saense/>

Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>