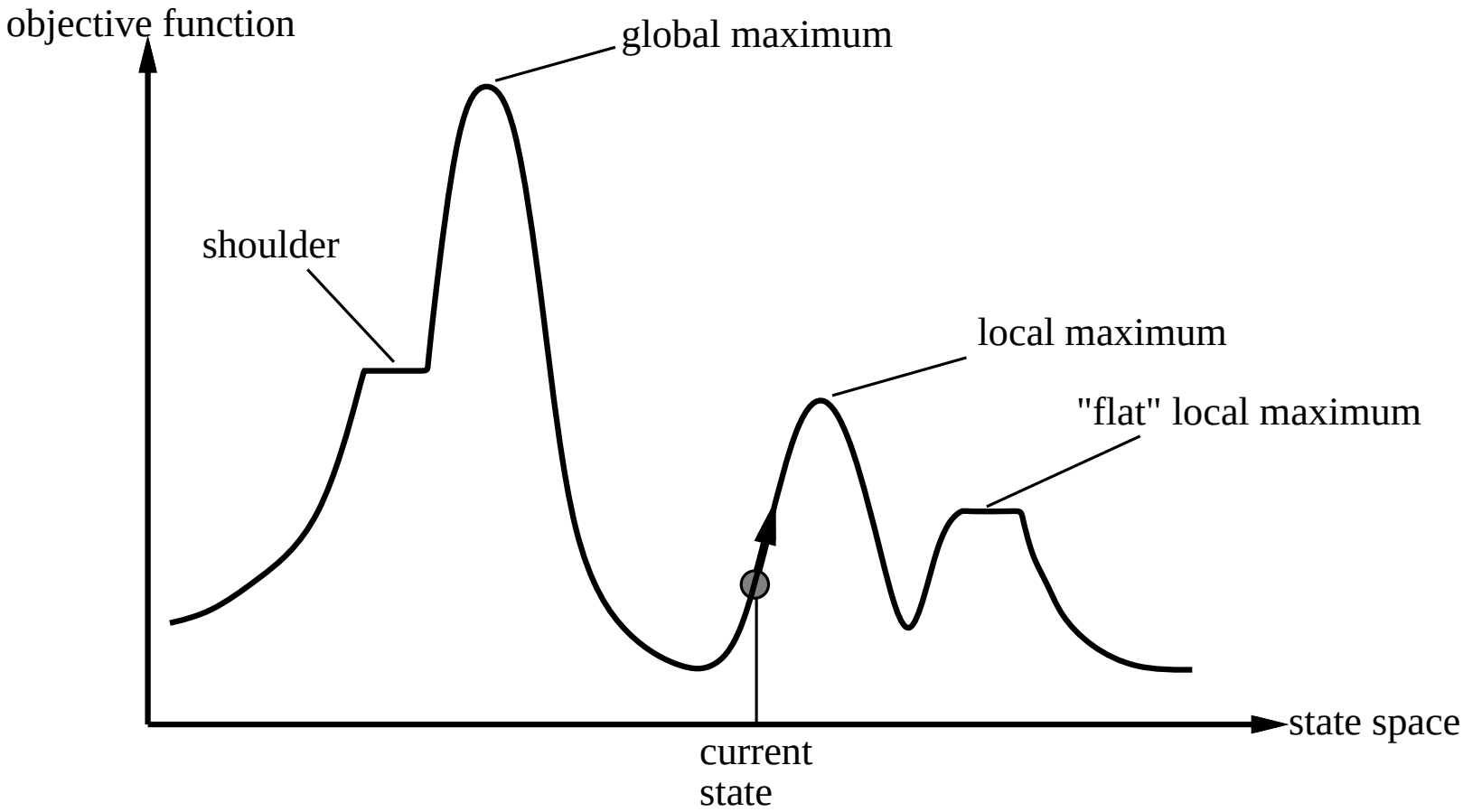
Estratégias de controle para a procura

- irrevogáveis: nunca retornam por um caminho já explorado
- tentativa: "backtracking" (métodos não informados e informados).

Algoritmos de Melhoramento Iterativo

- Utilizados em problemas cuja descrição já contém toda a informação para encontrar a solução (ex: n-rainhas e layout de circuitos VLSI)
- parte-se de uma config inicial conhecida e tenta-se melhorar a solução.

Algoritmos de Melhoramento Iterativo



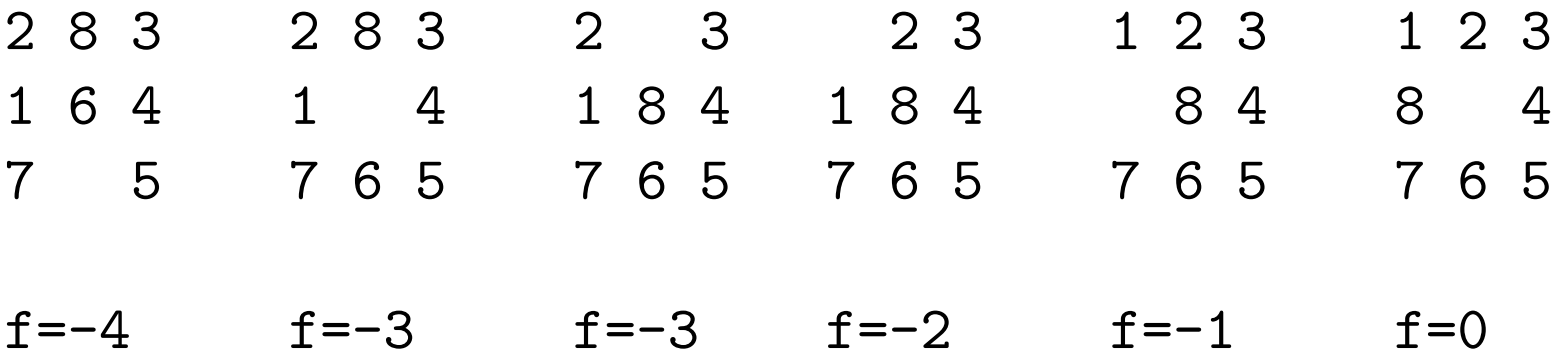
Algoritmos de Melhoramento Iterativo

- Exemplos:
  - Método Irrevogável: Hill-Climbing (busca de máximo/mínimo local)
  - Métodos de tentativa: (busca de máximo/mínimo global) podem temporariamente tornar estados piores.
    - random restart-hill-climbing
    - simulated annealing
    - busca tabu
    - busca paralela

Hill Climbing (ou gradiente descendente)

- tenta fazer modificações que melhorem o estado corrente
- 2 desvantagens:
  - máximos locais
  - platôs: percorre estados aleatoriamente porque a função de avaliação não muda muito
- Exemplo: Jogo dos oito :-)

Hill Climbing: exemplo



- No caso de não se poder aplicar a regra, o processo termina e a solução é um máximo local.
- OBS: este exemplo é baseado no algoritmo de hill-climbing da primeira edição do livro do Russell and Norvig. O algoritmo da segunda edição foi modificado fazendo com que este exemplo termine no segundo nó, sem encontrar o máximo global (f=0).

Hill Climbing: exemplo

- não funciona para:

```
1 2 5          1 2 3
  7 4   =>       7 4
8 6 3          8 6 5
```

f=-2

  - Qq movimento diminui o valor da função de avaliação.

Random Restart Hill Climbing

- executa uma série de buscas hill-climbing a partir de estados iniciais aleatórios
- cada um executa até terminar ou até não ter nenhum progresso
- salva o melhor resultado obtido até então
- pode ter no. finito de iterações ou continuar até não conseguir melhorar o melhor valor encontrado
- se superfície de busca contém muitos máximos locais, busca exponencial
- geralmente, solução boa pode ser encontrada em um número pequeno de iterações.

Simulated Annealing

- No lugar de começar novamente aleatoriamente quando passa num máximo local, permite que a busca escape do máximo local "descendo a montanha".

Simulated Annealing

```
function SA(problem,schedule) return a solution state
   current <- MAKE_NODE(INITIAL_STATE[problem])
   for t <- 1 to infinito do
       T <- schedule(t)
       if T = 0 then return current
       next <- sucessor de current selecionado
               aleatoriamente
       deltaE <- value[next] - value[current]
       if deltaE > 0 then current <- next
       else current <- next with prob e^(deltaE/T)
       endif
   endfor
```

Simulated Annealing

```
P = e^(deltaE/T)
n = sorteio de um no. de 0 a 1
if n < P then current <- next
```

- ou seja: quanto maior a prob mais chance de aceitar mover para passos de custo pior.

Outros Algoritmos

- Algoritmos Genéticos
  - Operações: "crossover", mutação e reprodução
  - começa de uma população inicial
  - aplica as operações
  - calcula uma função de "fitness" para cada indivíduo da população
  - pode eliminar indivíduos menos "fit"

Algoritmos Genéticos: pseudo-código

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  inputs: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

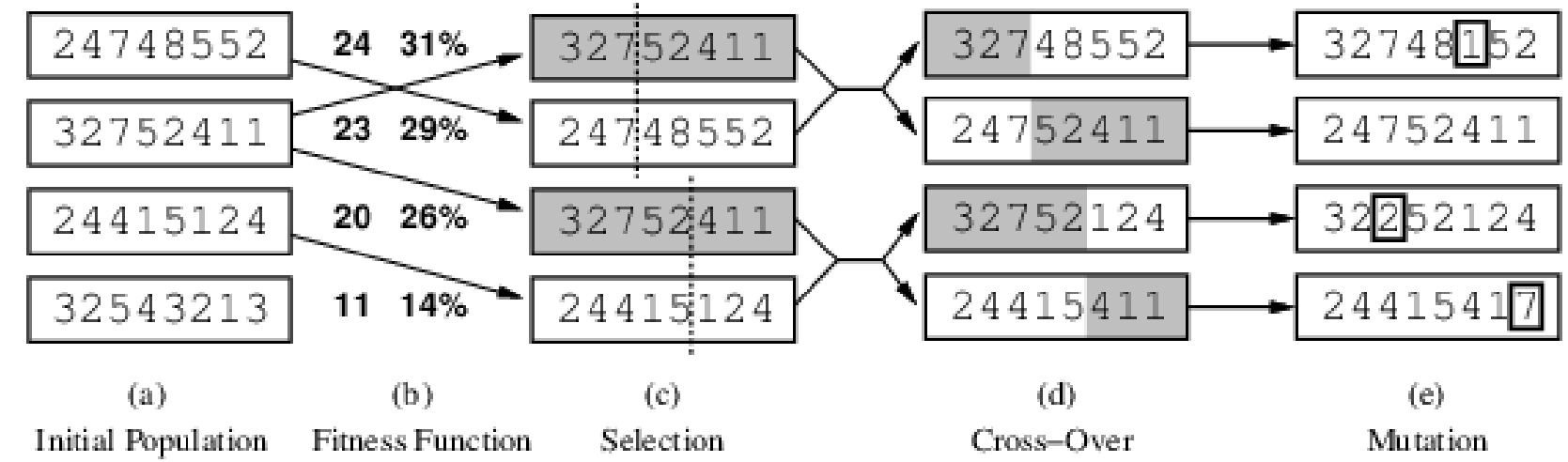
  repeat
    new_population ← empty set
    weights ← [FITNESS-FN(p) **for** *p* **in** *population*]
    for *i* = 1 **to** SIZE(*population*) **do**
      x, *y* ← WEIGHTED-RANDOM-SELECTION(*population*, *weights*, 2)
      child ← REPRODUCE(*x*, *y*)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *new_population*
  until some individual is fit enough, or enough time has elapsed
  return the best individual in *population*, according to FITNESS-FN

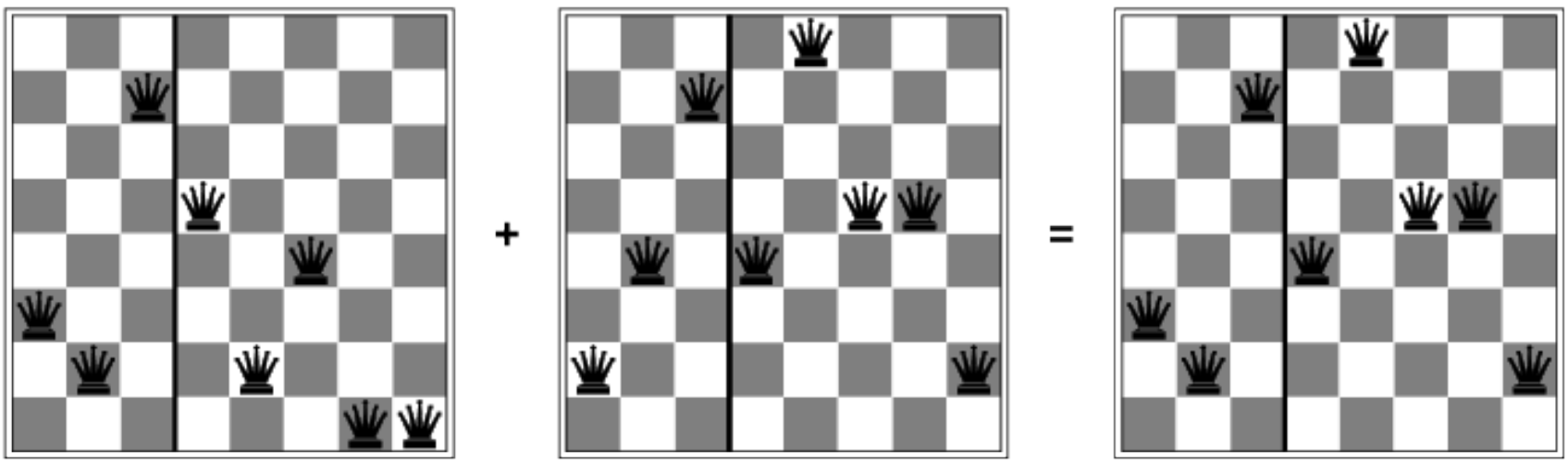
---

function REPRODUCE(x, y) **returns** an individual
  inputs: x, y, parent individuals

  $n \leftarrow$ LENGTH(x)
  $c \leftarrow$ random number from 1 to n
  return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c+1, n$))

Algoritmos Genéticos: exemplo





Algoritmos Genéticos: escolhas

- Como representar a população inicial?
- Quantos indivíduos deve ter a população inicial?
- Como gerar a população inicial (método?)
- Para a operação de "crossover":
  - quantos pais são escolhidos e de que forma?
  - quantos filhos são gerados (número de "splits" dos pais?)
- Para a operação de mutação: quantos e quais elementos mudar?
- Qual é a função de "fitness"?
- Eliminar indivíduos menos "fit": critérios e tamanho da nova população?