TÉCNICO LISBOA

Procura Local

TÉCNICO LISBOA

Livro

- Capítulo 4 Secção 1

TÉCNICO LISBOA

Resumo

- Procura local
- Algorítmos de procura local
  - Hill-climbing
  - Simulated annealing
  - Local beam
  - Genetic algorithms

TÉCNICO LISBOA

Procura Sistemática

- No capítulo 3, analisamos estratégias de procura sistemática
  - Exploram sistematicamente caminhos a partir de um estado inicial
  - Guardam um ou mais caminhos em memória
    - Guardam também as alternativas ainda não exploradas em cada ponto de decisão
  - Quando o objectivo é encontrado, o caminho para esse objectivo corresponde a uma solução para o problema

TÉCNICO LISBOA

Procura Local

- Em muitos problemas de optimização, o caminho que leva ao objectivo é irrelevante;
  - o próprio estado objectivo é a solução (e.g., n-rainhas)
- Nestes casos, podemos usar procura local
  - Mantém um único “estado actual”; caminhos não são memorizados
  - Tipicamente, um estado transita para estados “vizinhos”, substituindo o estado actual

TÉCNICO LISBOA

Procura Local

- Vantagens
  - Usam muito pouca memoria (memoria constante)
  - Conseguem encontram soluções em espaços de estados infinitos
    - Procuras sistemáticas não
  - Boas para resolver
    - problemas de optimização
      - Encontrar o estado que maximize/minimize uma função de avaliação
    - problemas de reparação
      - Começamos de um estado inicial completo mas que não satisfaz as restrições do problema
      - Encontrar o estado que satisfaça as restrições, com o mínimo de alterações

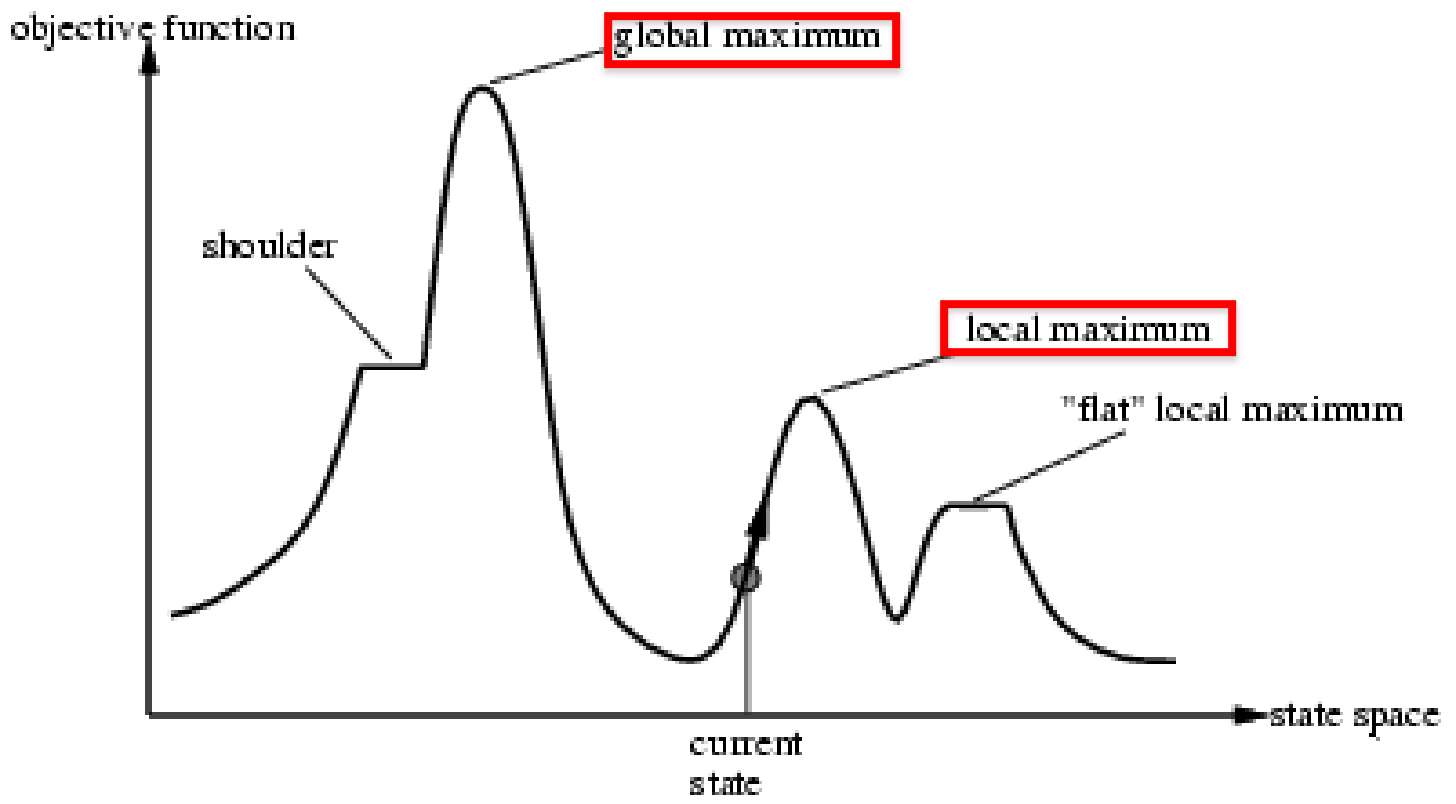
TÉCNICO LISBOA

Procura Local

- Desvantagens
  - Não podem ser aplicadas se precisarmos do caminho
  - Não são normalmente completas/óptimas, pois podem ficar presas facilmente em máximos locais

Procura Local

- Problema: dependendo do estado inicial, pode ficar preso a um máximo local



Hill-climbing (trepar-a-colina) ou procura local ganaciosa

- É um simples ciclo que se move continuamente na direcção de um valor melhor. Termina quando nenhum sucessor tem valores melhores.
- Não guarda árvore de procura
- Não olha para além dos vizinhos imediatos

- “É como subir o Evereste com nevoeiro cerrado e amnésia” (AIMA)

TÉCNICO LISBOA

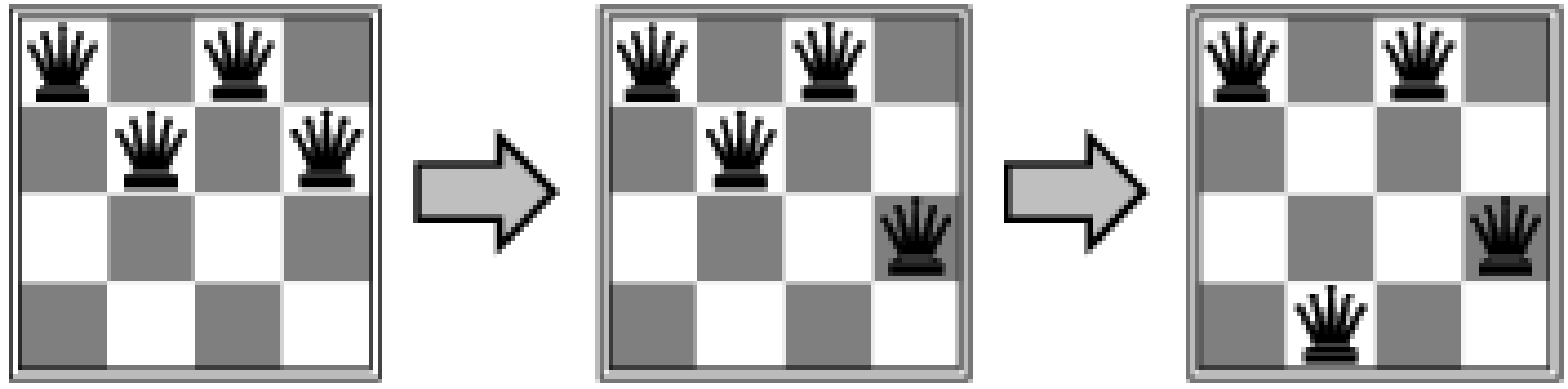
Algoritmo Hill-climbing

function HILL-CLIMBING(*problem*) returns a state that is a local maximum

**   *current* ← Make-Node(*problem*.Initial-State)**
**   loop do**
**   neighbor ← a highest-valued successor of *current***
**   if *neighbor.*Value ≤ *current*.Value then return *current*.State**
**   *current* ← *neighboor***

Examplo: *N-rainhas*

- Problema: Colocar as N rainhas numa matriz n $\times$ n de modo que nenhuma esteja em posição de atacar as outras



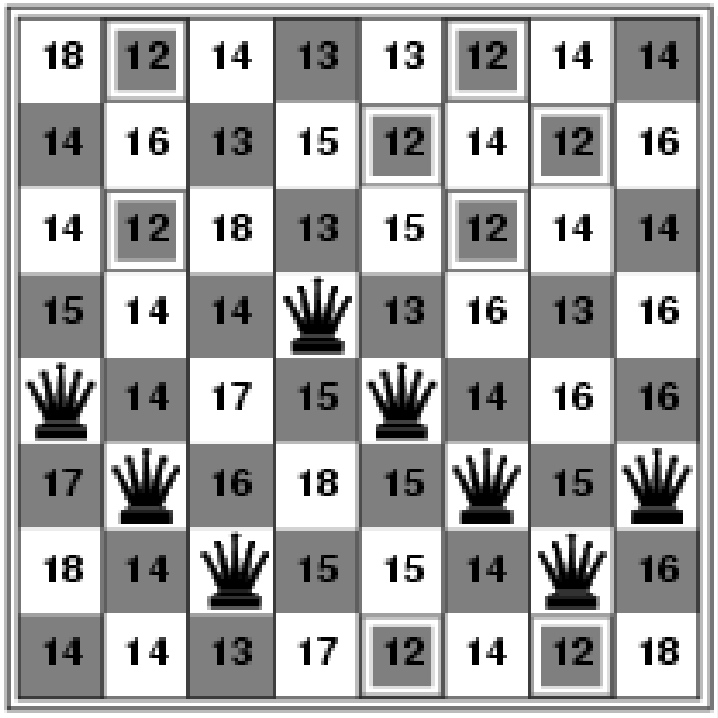
- Modelado como problema de reparação
  - Estado inicial com uma rainha por coluna gerado aleatoriamente
  - Novos estados gerados a partir de movimentos para estados vizinhos

TÉCNICO LISBOA

Procura com o Hill-climbing no problema das 8 rainhas

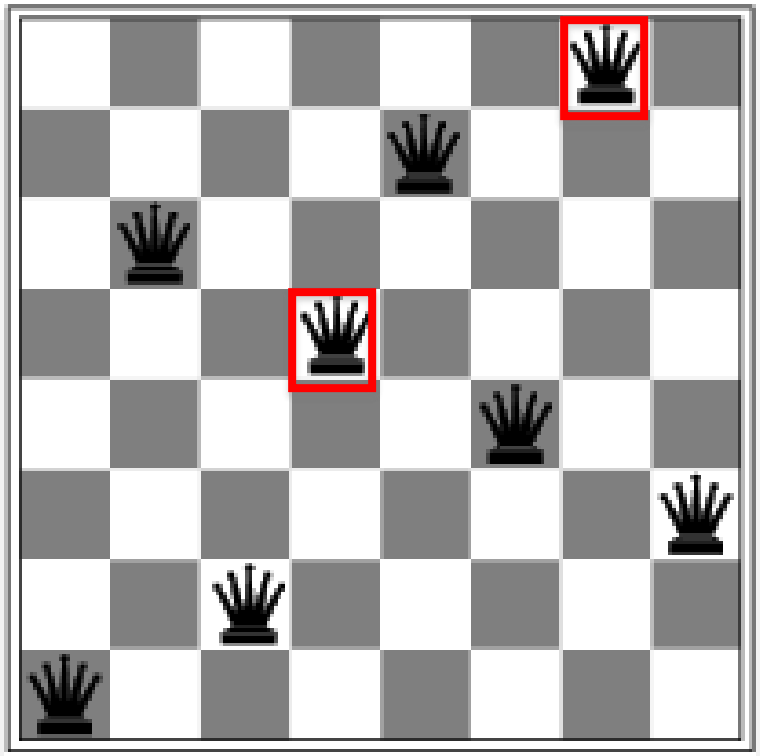
- Função sucessor: mexer uma rainha para outra posição na mesma coluna.
- Tipicamente, o Hill-climbing escolhe aleatoriamente entre os melhores sucessores, se houver mais do que um.

Procura com o Hill-climbing no problema das 8 rainhas



- *h* = nº de pares de rainhas que se estão a atacar
  - No tabuleiro apresentado: *h* = 17
  - Num tabuleiro objectivo: *h* = 0
- Inteiros correspondem ao valor de *h* para sucessores resultantes de mover rainha na respectiva coluna (≥12)

Procura com o Hill-climbing no problema das 8 rainhas



- **Mínimo local com *h = 1***
- **Qualquer sucessor tem valor de *h* superior**

Não encontrou solução e não evoluiu para outro estado…

TÉCNICO LISBOA

Problemas Hill-climbing

- Hill-climbing tem problemas com:
  - Máximos locais – pico mais elevado que todos os seus vizinhos
  - Planaltos – zona do espaço de estados onde a função de avaliação é plana
  - Cumeadas – sequência de máximos locais através dos quais é difícil viajar

TÉCNICO LISBOA

Hill Climbing

- No caso das 8-rainhas
  - Começando num estado aleatório…
    - Só resolve 14% dos casos (necessita em média de 4 iterações)
    - Nos restantes casos fica “parado” ao fim de 3 iterações (em média)
- Como resolver estes problemas?
  - Aleatoriedade :D

Variantes do Hill Climbing

- Stochastic Hill climbing
  - em vez de escolher o melhor nó vizinho
  - escolhe aleatoriamente de entre os vizinhos que melhoram o valor actual
  - probabilidade de selecção varia em função do valor da melhoria

  - Converge mais lentamente para uma solução
  - Mas em certos problemas obtem melhores soluções

TÉCNICO LISBOA

Variantes do Hill Climbing

- First-choice Hill Climbing
  - gera os sucessores aleatoriamente um de cada vez
  - se o sucessor gerado for melhor que o nó actual avança para esse sucessor
    - restantes sucessores já não vão ser gerados
  - Ideal para problemas onde o número de sucessores é muito grande ou mesmo infinito
- No entanto, estas duas variantes não resolvem o problema de máximos locais

TÉCNICO LISBOA

Variantes do Hill Climbing

- Random-restart Hill Climbing:
  - Se à primeira não conseguimos, tentar de novo
- conduz uma sequência de procuras hill-climbing a partir de diferentes estados iniciais
  - gerados aleatoriamente
  - pára quando se encontra o objectivo
- completa com probabilidade perto de 1
  - eventualmente, se repetirmos vezes suficientes
    - o estado inicial será o estado objectivo
- mas não chega a ser completa com 100% (só no infinito)
  - analogia com o teorema do macaco c/ tempo infinito

TÉCNICO LISBOA

Variantes do Hill Climbing

- Random-restart Hill Climbing:
  - Se p for a probabilidade de sucesso de cada procura Hill-climbing
    - número esperado de tentativas é 1/p
  - Exemplo 8-rainhas
    - P ~ 0.14
    - Em média são necessárias 7 tentativas (6 falhadas + 1 sucesso)
    - 22 passos para encontrar o objectivo
      - Média de 3 passos por iteração

TÉCNICO LISBOA

Hill Climbing

- Apesar de tudo:
  - Converge (ou não) rapidamente
  - Por exemplo, o Random-restart Hill Climbing consegue encontrar uma solução para as n-rainhas, em menos de um minuto, mesmo para 3 milhões de rainhas.
- Boa quando
  - Espaço de estados com poucos máximos locais e planaltos
  - Random-restart contorna facilmente esses máximos locais
- Em problemas mais complexos
  - Consegue encontrar um máximo local “razoável” ao fim de poucas iterações

Procura Simulated Annealing

- Em Português: têmpera simulada
- Ideia: escapar ao mínimos locais permitindo que se façam movimentos “maus”, mas vai gradualmente decrementando a sua frequência
  - Escolhe um sucessor aleatoriamente
  - Se o valor do successor é melhor que o actual movemo-nos para o successor
  - Mas mesmo que seja pior, podemos mover-nos na mesma
    - Com probabilidade $p < 1$,
    - Probabilidade decresce exponencialmente com a má qualidade do movimento ΔE
    - Probabilidade desce também com temperatura T

Procura Simulated Annealing

- Valores elevados da Temperatura T
  - Maus movimentos são facilmente permitidos
- Valores baixos da temperatura T
  - Maus movimentos não são facilmente permitidos

- Consegue-se provar que se temperatura T diminuir suficientemente devagar (em função do *schedule*), então a procura simulated annealing vai encontrar um máximo global com probabilidade próxima de 1

- Analogia com metalurgia (annealing)
  - Processo usado para temperar e endurecer metais e vidros
  - Aqueçe-se o material a uma temperatura muito alta
  - Deixa-se arrefecer muito devagarinho

TÉCNICO LISBOA

Simulated Annealing

- Metáfora: imaginar a tarefa de pôr uma bola de ping-pong no buraco mais profundo de uma superfície cheia de buracos
- Uma solução é deixar a bola ir parar a um mínimo local e depois abanar a superfície de modo a tirá-la do mínimo local
- Simulated annealing começa por “abanar” muito no início e depois vai abanando cada vez menos

TÉCNICO LISBOA

Simulated Annealing

function SIMULATED-ANNEALING(*problem*,*schedule*) returns a solution state

inputs: *problem*, a problem

***schedule*, a mapping from time to "temperature"**

***current* ← Make-Node(*problem*.Initial-State)**

for t = 1 to ∞ do

T ← *schedule(t)*

if T = 0 then return *current*

next* ← a randomly selected successor of *current

∆E ← *next*.Value – *current*.Value

if ∆E > 0 then *current* ← *next*

else *current* ← *next* only with probability $e^{\Delta E/T}$

Local Beam (procura em banda)

- Será que não podemos gastar um pouco mais de memória?
- Ideia: guardar a referência a k estados, em vez de 1
  - Começa com k estados gerados aleatoriamente
- Em cada iteração, todos os sucessores dos k estados são gerados
- Se algum é um estado objectivo, pára; caso contrário escolhe os k melhores sucessores e repete

TÉCNICO LISBOA

Procura Local Beam

- Atenção que este algoritmo é mais do que correr k Random-restart Hill Climbings em paralelo!!
  - Não têm de ser escolhidos sucessores de todos os estados
  - Se um estado gera vários bons sucessores e os outros k-1 estados não, os estados menos promissores são abandonados
- No entanto, também pode ter problemas: pode haver pouca diversidade nos k estados…
  - os k estados podem facilmente ficar concentrados numa área pequena do espaço de estados

Procura Stochastic Beam

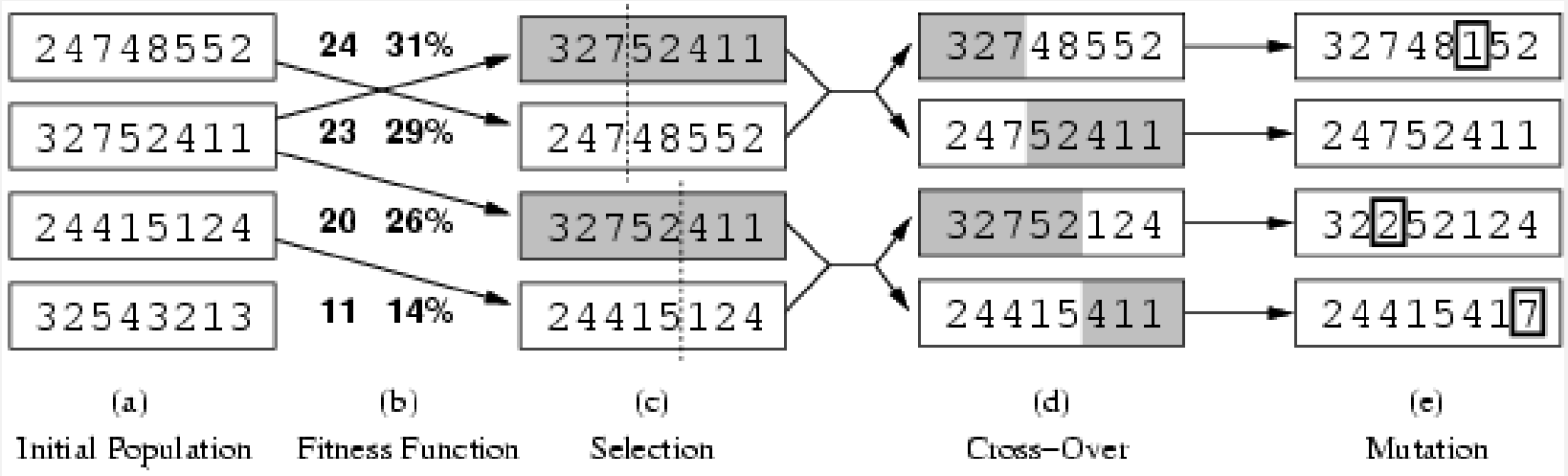
- Stochastic Beam Search
  - Os k sucessores são escolhidos aleatoriamente
  - Probabilidade da escolha aumenta em função da sua qualidade

TÉCNICO LISBOA

Algoritmos Genéticos

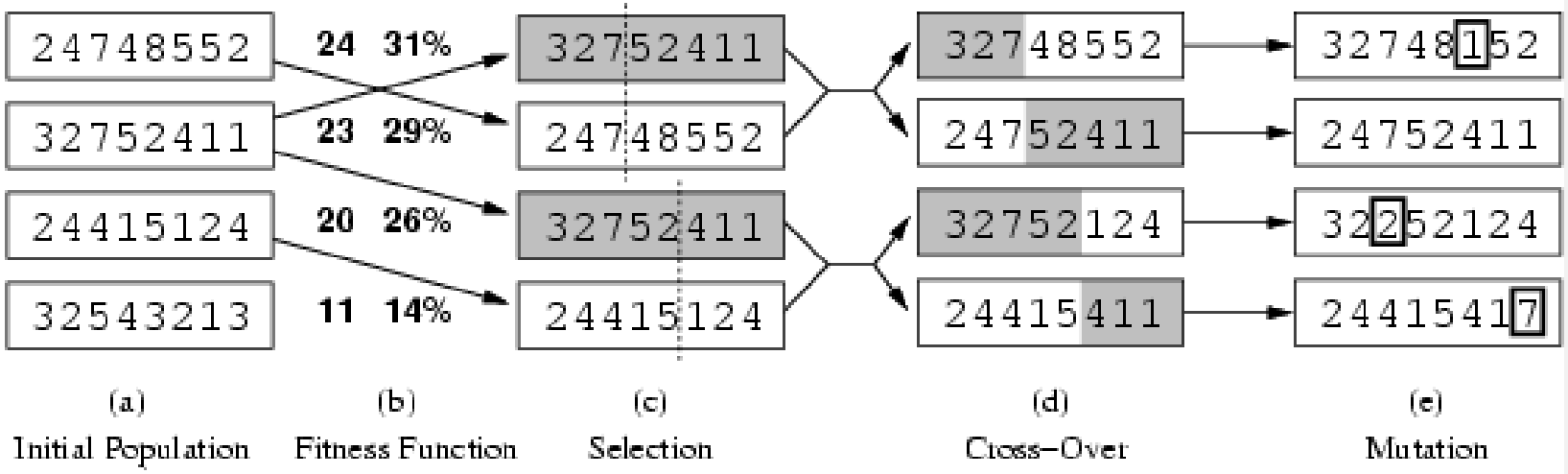
- Variante da stochastic beam search
- Começa com k estados gerados aleatoriamente (população) tal como procura em banda
  - Um estado é representados como uma string sobre um alfabeto finito (geralmente {0,1})
- O estado sucessor é gerado através da combinação de dois estados (pais)
  - A função de avaliação (fitness function) dá valores mais altos aos melhores estados
  - Quanto melhor a avaliação de um estado, maior a probabilidade de ser selecionado para “procriação”
  - Produz a próxima geração de estados por selecção, cruzamento e mutação

Algoritmos Genéticos



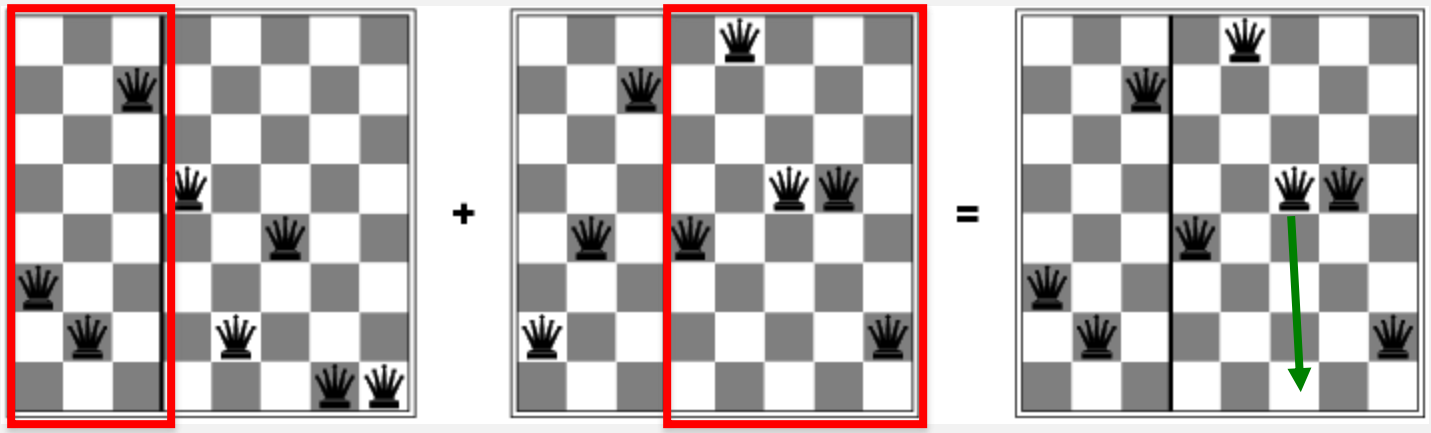
- Fitness function (b): nº de pares de rainhas não atacantes (min = 0, max = (8 × 7)/2 = 28)
  - Probabilidade de selecção (c) em função da fitness function
    - Exº 24/(24+23+20+11) = 31%
  - Um estado pode ser seleccionado mais de uma vez

Algoritmos Genéticos



- O ponto de cruzamento é escolhido aleatoriamente
- São criados os filhos (d)
- Depois de criado o filho,
  - Com uma probabilidade baixa independente
  - Esse filho pode sofrer mutações aleatórias (e)

Algoritmos Genéticos:
cruzamento e mutação



32752411 **24748552** **32748552**

32748152

TÉCNICO LISBOA

Algoritmos Genéticos

- **Há ainda muito trabalho a fazer de modo a perceber em que condições e com que parâmetros é que os algoritmos genéticos se comportam bem**