



Universidade do Minho  
Escola de Engenharia  
Departamento de Informática

Otimização e Procura

Algoritmos *Population-based*

Algoritmos Genéticos

LICENCIATURA EM ENGENHARIA INFORMÁTICA  
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

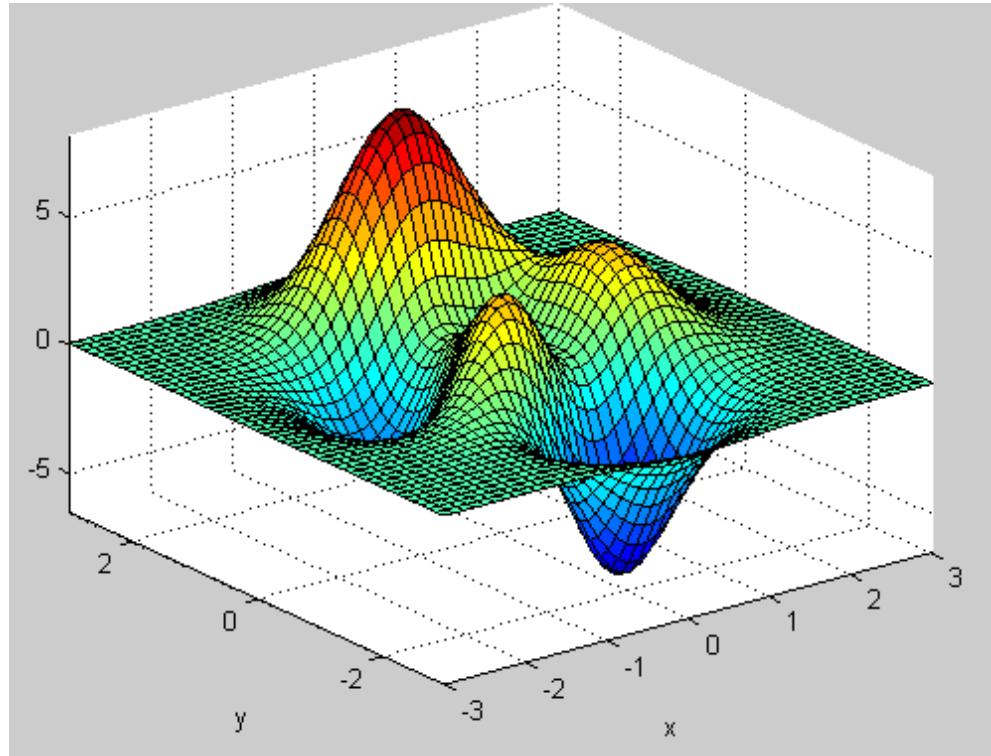
Inteligência Artificial

2022/23

- Procura vs Otimização
- Procura local vs procura global
- Solução única vs Population-Based
- *Population-Based*
  - Algoritmos Genéticos (Genetic Algorithms)
  - Ant Colony Optimization
  - Particle Swarm Optimization



## Otimização e Procura



Tjalling C. Koopmans (Nobel Memorial Prize in Economic Sciences (1975)):

- “best use of scarce resources”
- “Mathematical Methods of Organizing and Planning of Production”

Roger Fletcher (Mathematician (1987)):  
“The subject of optimization is a fascinating blend of heuristics and rigour, of theory and experiment.”

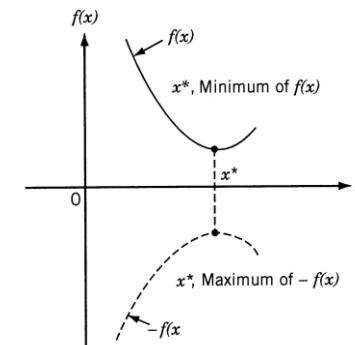
## Problemas de Otimização

- Problema de Otimização Matemática (minimização):

$$\text{minimize } f_0(x)$$

$$\text{subject to } g_i(x) \leq b_i, \quad i = 1, \dots, m$$

- $f_0: \mathbb{R}^n \rightarrow \mathbb{R}$ :
  - Função objetivo (calcula um valor)
- $x = (x_1, \dots, x_n)$ :
  - Variáveis controláveis (linearmente independentes)
- $g_i: \mathbb{R}^n \rightarrow \mathbb{R} \quad (i = 1, \dots, m)$ :
  - Restrições (podem ser de outros tipos)



## Otimização Contínua

“encontrar os máximos e mínimos de funções, possivelmente sujeitos a restrições”;

“Na otimização contínua, as variáveis no modelo podem assumir qualquer valor dentro de um intervalo de valores, geralmente números reais. ...”.

## Otimização Discreta

“procurar minuciosamente para encontrar um item com propriedades especificadas entre um conjunto de itens”;

“Ao contrário da otimização contínua, as variáveis usadas (ou algumas delas) são restritas a assumir apenas valores discretos, como os inteiros.”.

## Para além da Procura Clássica Algoritmos de Melhoria Iterativa

- Em muitos problemas de otimização, o caminho para o objetivo é irrelevante.
- Espaço de Estados = conjunto das configurações completas.
- Algoritmos Iterativos mantêm um único estado (corrente) e tentam melhorá-lo.
- Algoritmos de Melhoria Iterativa:
  - Procura Subida da Colina (Hill-Climbing Search)
  - Arrefecimento Simulado (Simulated Annealing)
  - Procura Tabu (Tabu Search)
  - Algoritmos Genéticos (Genetic Algorithms)
  - Colónia de Formigas (Ant Colony Optimization)
  - Enxame de Partículas (Particle Swarm Optimization)
- **Estratégia:** Começar como uma solução inicial do problema e fazer alterações de forma a melhorar a sua qualidade

- Procura local vs procura global

- Algumas meta-heurísticas aplicam métodos de procura local, onde as novas soluções exploradas são “vizinhas” de soluções anteriores (e.g. *Simulated Annealing*, *Tabu Search*);
- Outras meta-heurísticas distribuem o processo de procura por todo o espaço de procura (normalmente através de abordagens baseadas em populações).

- Solução única vs *Population-based*

- As abordagens de solução única, são iterativas, e orientam o processo de procura através da melhoria da solução anterior;
- As abordagens baseadas em populações utilizam uma procura em paralelo por parte de vários membros da população, podendo, ou não, existir a troca de informação entre os indivíduos (e.g. *Particle Swarm optimization*, *Genetic Algorithms*, *Ant Colony optimization*).

- **Algoritmos Genéticos**

- Definição do estado como um cromossomo
  - Gerar soluções (cromossomos) a partir de uma população de estado inicial
  - Reprodução, Mutação e Seleção

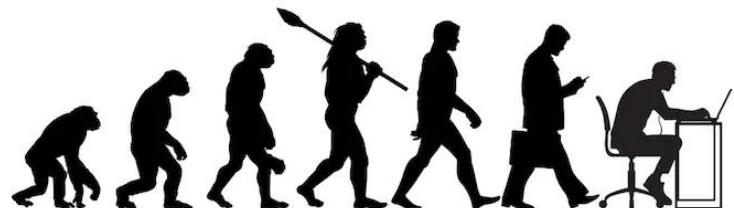
- **Colónia de Formigas (Ant Colony)**

- Iniciando vários estados (colónia de formigas)
  - A probabilidade de um caminho ser melhor é determinada pelo número de “formigas” que passam por ele

- **Enxame de Partículas (Particle Swarm)**

- Vários estados de partida (enxame)
  - A vizinhança é explorada e mantida, a melhor solução e o melhor estado
  - “Partículas” caminham na direção da melhor solução encontrada até agora
  - A velocidade do movimento depende das distâncias para a melhor solução e o melhor estado e a posição do estado

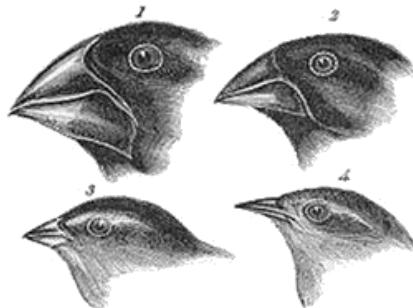
- Família de algoritmos de otimização global inspirados na biologia;
- Métodos baseados em população, que fazem evoluir soluções individuais por tentativa e erro;
- Inspirados na teoria da evolução das espécies de Darwin, e em mecanismos biológicos.



- Na biologia:

- Cada animal da espécie é uma solução para um problema;
- Há soluções melhores que outras;
- Melhores soluções têm mais probabilidade de sobreviver, reproduzir-se e passar o seu conteúdo genético para uma nova geração;
- Soluções mudam de geração em geração:
  - Por cruzamento de material genético dos progenitores;
  - Por mutações aleatórias.
- Alguns descendentes são melhores, outros são piores;
- Sem influência externa, a população tende a melhorar com o tempo.

## Perspetiva biológica

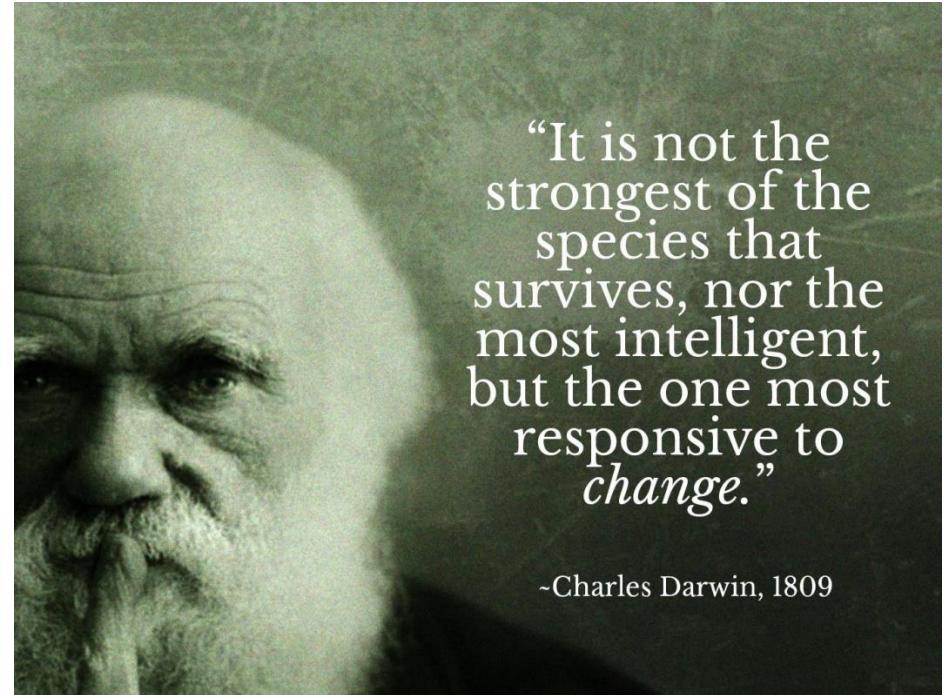


1. *Geospiza magnirostris*  
3. *Geospiza parvula*

2. *Geospiza fortis*  
4. *Certhidea olivacea*

## Perspetiva biológica

- Capacidade de adaptação é central
  - Sem ela uma espécie não evolui



- Algoritmos Genéticos configuram processos adaptativos de procura, num espaço de soluções, por aplicação de operadores modelados de acordo com o conceito de herança, inerente à teoria da evolução das espécies, de **Charles Darwin**;
- AG's distinguem-se pelo método de procura que utilizam e pelo **tratamento específico dos ótimos locais**;
- Utilizados para resolver problemas complexos ou que não se sabe bem como resolver;
- Não garantem resultados ótimos;
- Aplicam-se:
  - Em problemas que envolvem a melhoria de soluções (problemas de otimização);
  - Em problemas cujo o cálculo de soluções é difícil ou impossível.

Os algoritmos genéticos diferem dos algoritmos tradicionais de otimização em basicamente quatro aspectos:

- baseiam-se numa codificação do conjunto das soluções possíveis, e não nos parâmetros da otimização em si;
- os resultados são apresentados como uma população de soluções e não como uma solução única;
- não necessitam de nenhum conhecimento derivado do problema, apenas de uma forma de avaliação do resultado;
- usam transições probabilísticas e não regras determinísticas.

- Procedimento iterativo que mantém uma população de estruturas candidatas a soluções, para **domínios específicos**;
- A cada incremento de tempo (geração), as estruturas da população atual são **avaliadas** na sua capacidade de serem soluções válidas para o domínio do problema;
- É constituída uma **nova população** de soluções candidatas, baseada na sua avaliação e pela aplicação de **operadores genéticos**.

- Otimização;
- Sistemas de controlo dinâmicos;
- Descoberta de novas topologias conexionistas;
- Criatividade artificial;
- ...

- Os algoritmos genéticos são uma classe particular de algoritmos evolutivos que usam técnicas inspiradas pela biologia evolutiva como:
  - hereditariedade,
  - seleção natural;
  - recombinação;
  - mutação.
- O AG é um processo estocástico no qual os estados sucessores são gerados pela combinação de dois estados pais em vez de modificar um único estado.

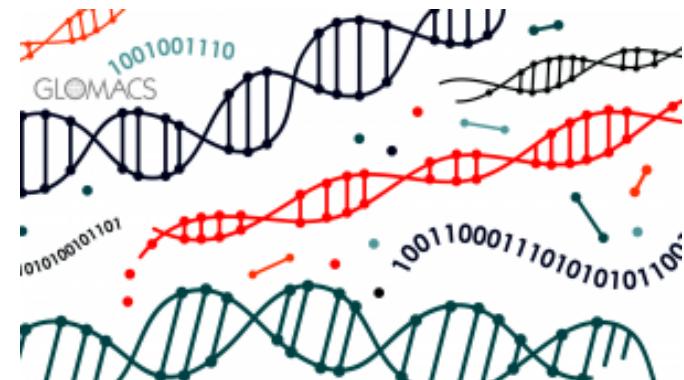
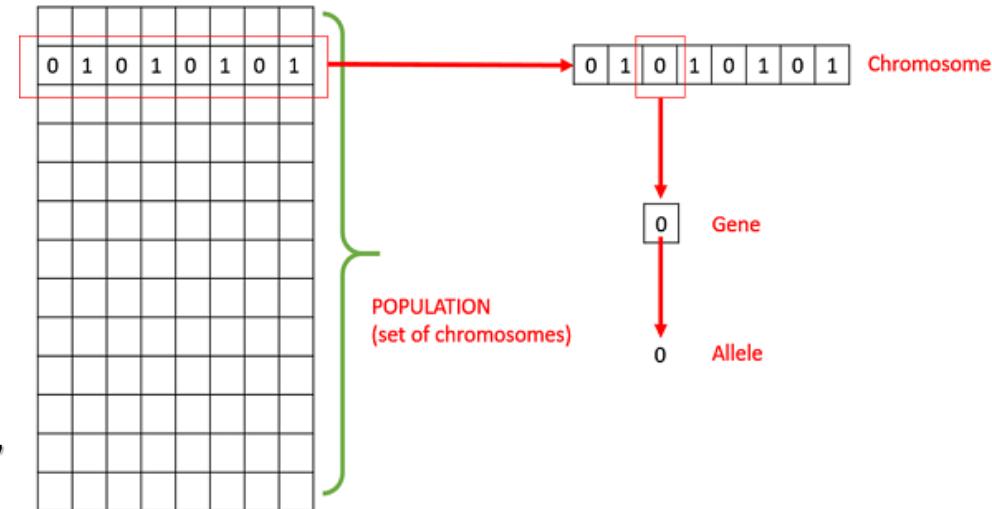


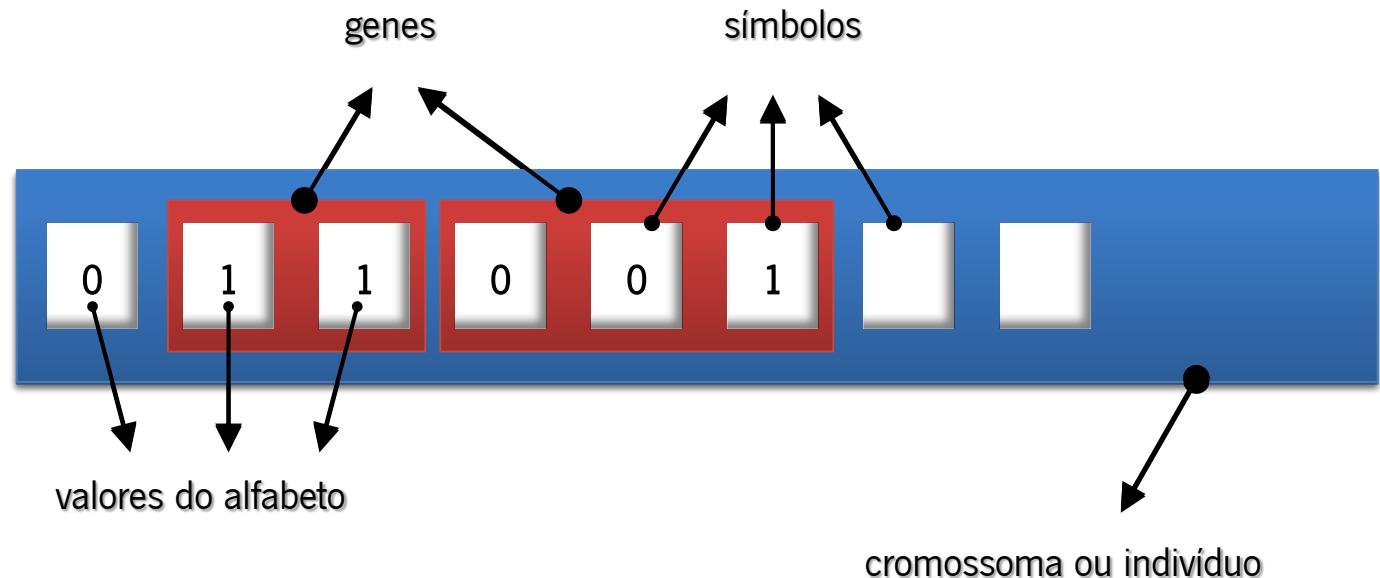
Image Source: <http://glomacs.com/articles/genetic-algorithms>

- Indivíduo ou Cromossoma
  - Uma solução específica, constituída por um ou mais elementos
- Gene
  - Uma posição no cromossoma (um elemento)
- Alelo
  - O valor de um gene
- População
  - Conjunto de indivíduos
- Geração
  - Conjunto de todos os novos indivíduos “nascidos” a partir de uma mesma população pai
  - Mede a passagem do tempo
- Espaço de procura
  - Os limites dos valores dos Genes



- O ambiente, os inputs e os outputs são representados por conjuntos de símbolos, de tamanho fixo, de um dado alfabeto:
  - { 0; 1 }
  - { X; Y; Z }
  - { i; ii; iii; iv; v; ... }
  - $\mathbb{R}$
  - ...
- Codificar candidatos de uma forma que facilite, nomeadamente, a mutação e o cruzamento;
- A representação típica do candidato é uma *string* binária;
- Esta *string* pode ser considerada como o código genético de um candidato – daí o termo “algoritmo genético”!;
- Outras representações são possíveis, mas geralmente dificultam o cruzamento e a mutação.

- Cada ponto, no domínio do ambiente, pode ser considerado um **indivíduo**, representado por uma sequência (*string*) gerada a partir do alfabeto, constituído na forma:



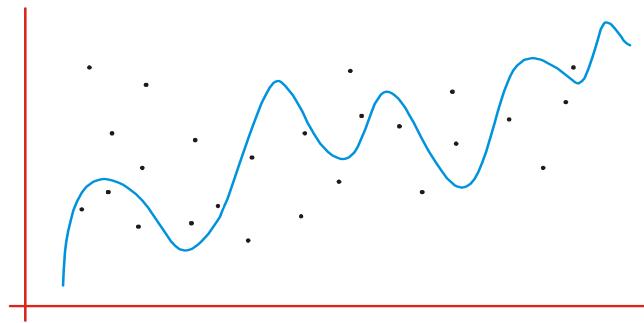
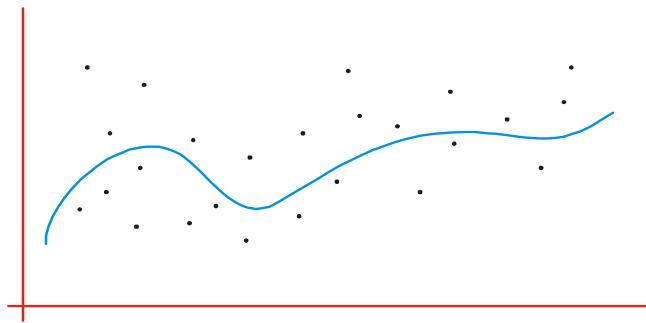
- A entidade básica para a representação do conhecimento é o **cromossoma**, composto por **genes**;
- Supondo a utilização de uma codificação binária:
  - Para representar 8 valores (p.ex., as 7 cores do arco íris mais o preto):
    - utilizar 3 símbolos em 1 gene  $\rightarrow 2^3 = 8$
  - Para representar 5 valores (ex., para representar os dedos de uma mão ou as cores da íris do olho):
    - utilizar 3 símbolos em 1 gene  $\rightarrow 2^3 = 8 > 5$
    - utilizar 2 símbolos em 1 gene  $\rightarrow 2^2 = 4 < 5$

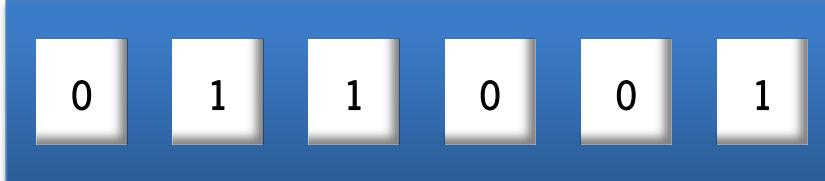
- Conjunto de soluções, torna-se necessário avaliar a adequação de cada uma delas;
- É usada para identificar os indivíduos (cromossomas) mais aptos;
- Será com base nesta avaliação que as melhores soluções serão escolhidas para darem origem à geração seguinte;
- Para isto, é definida a **função de fitness**
  - A função de fitness aceita como parâmetro uma solução e devolve um valor numérico que representa quão boa uma solução é...
  - Valor não é geralmente absoluto mas relativo:
    - Porque geralmente não conhecemos o melhor/pior caso;
    - Apenas permite comparar as soluções entre si, não sabemos quão longe estamos da melhor solução possível.

$$\bullet \mathcal{F}(0 \boxed{1} \boxed{1} \boxed{0} \boxed{0} \boxed{1}) = X$$

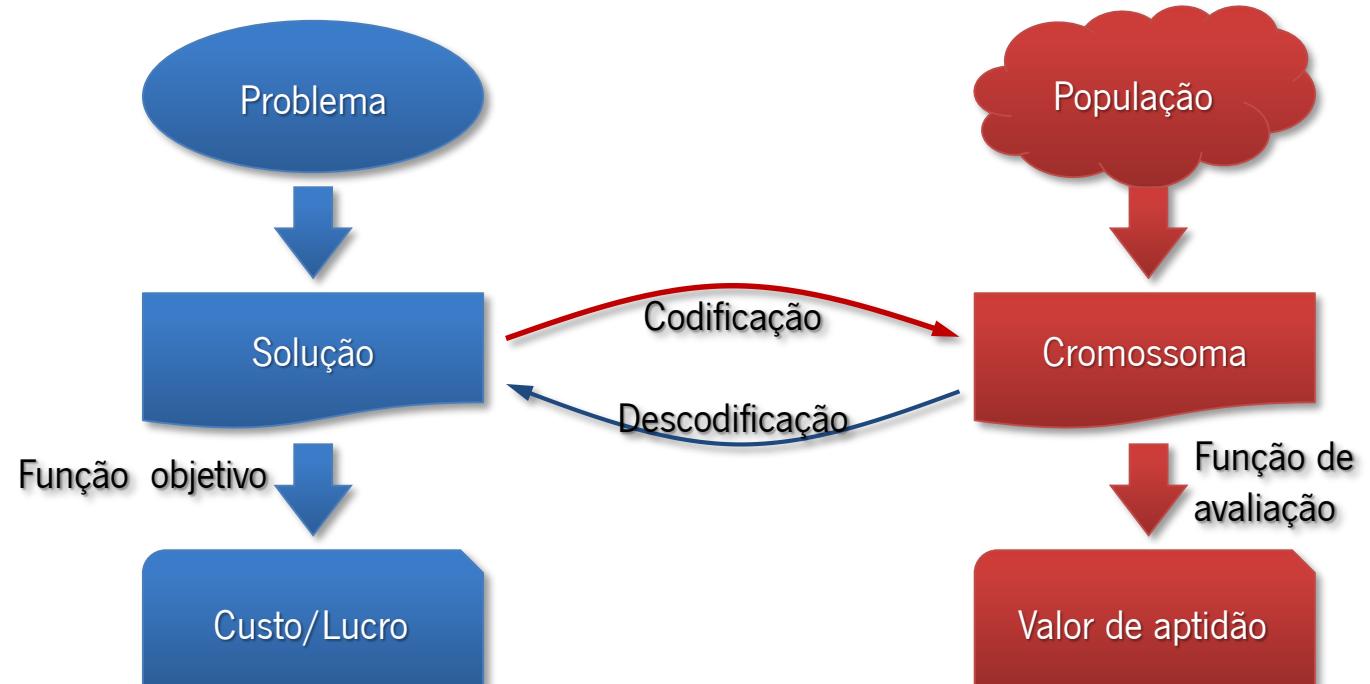


- A função de avaliação deve ser **suave e regular** no sentido de evitar ao máximo que a procura “caia” frequentemente em máximos ou mínimos locais;



- $\mathcal{F}($  $) = X$

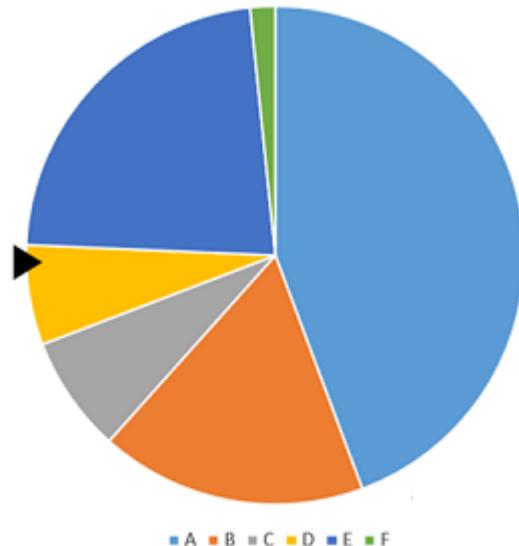
## Avaliação das Soluções



- Na fase de seleção, escolhem-se os indivíduos que darão origem à nova geração;
- A Seleção de progenitores para reprodução deve garantir que os indivíduos com maior valor de aptidão tenham, proporcionalmente, mais descendentes;
- Idealmente, cada indivíduo deverá ser representado por um número de descendentes equivalente ao rácio entre o seu valor de aptidão e o valor médio da população;
- Existem diversas estratégias de Seleção:
  - Baseados no Fitness – Utilizar os melhores N elementos da população para gerar a próxima população
  - Baseado na idade – São “expulsas” as N soluções mais antigas
  - Método da Roleta;
  - ...

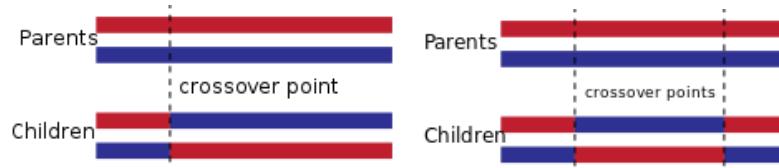
- Selecionar os melhores indivíduos pode resultar numa convergência prematura. As melhores estratégias de seleção são projetados para manter diversidade na população:
  - Rank - escolher os melhores indivíduos;
  - Roulette wheel: probabilidade de seleção é proporcional à aptidão;
  - Tournament - um número inicialmente grande é selecionado via roleta, então os melhores classificados são escolhidos;
  - Elite - em combinação com outras estratégias de seleção, manter sempre o indivíduo mais apto por perto.

Em geral, usa-se o algoritmo de seleção probabilística, onde os cromossomas são ordenados de acordo com a função-objetivo (fitness) e lhes são atribuídas probabilidades decrescentes de serem escolhidos - probabilidades essas proporcionais à razão entre a adequação do indivíduo e a soma das adequações de todos os indivíduos da população.



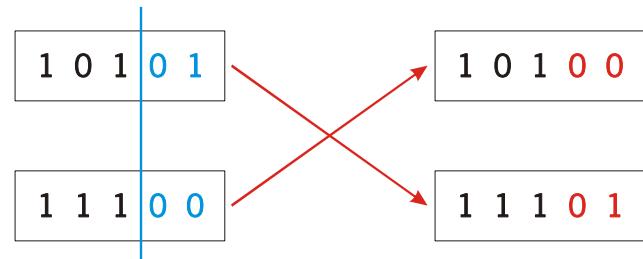
- A fase de reprodução tem como objetivo criar a nova geração seguinte;
- Cada nova geração é criada através da aplicação de operadores genéticos à geração anterior:
  - Cruzamento;
  - Mutação.
- Tipicamente, é definida na configuração a proporção de aplicação destes operadores:
  - Qual a percentagem da nova geração que deve ser gerada por reprodução, por cruzamento, e qual deve ser mantida inalterada.

- O objetivo do cruzamento é o de conseguir, nos descendentes, uma combinação do material genético dos progenitores;
- O operador Cruzamento é aplicado a dois indivíduos da população, produzindo outros dois indivíduos para a população da geração seguinte;
- Existem diferentes formas de implementar o operador:



- Recombinação ou Crossover

- Uma vez selecionados os indivíduos, estes passam, com uma probabilidade pré-estabelecida, pelo processo de cruzamento (crossover), onde partes dos genes dos pais são combinadas para geração de filhos.



- O operador genético Mutação pretende ser uma analogia com a mutação genética dos seres vivos;
- A mutação actua sobre os indivíduos e efetua algum tipo de alteração na sua estrutura. A importância deste operador reside no fato de garantir que outras alternativas serão exploradas;
- Aplica-se com uma baixa probabilidade;
- Garante que é sempre possível chegar a qualquer posição do problema;
- Previne a perda total de informação pela seleção e/ou eliminação.



Podem ser utilizadas estratégias de seleção e reprodução muito distintas:

- Selecionar o top 50% e cada solução gera uma nova (por mutação) ou cada par de soluções gera duas novas (por cruzamento);
- Selecionar todas as soluções, gerar novas soluções com base nelas, e depois selecionar as melhores (de acordo com o tamanho da população);
- Cada solução tem uma probabilidade de ser selecionada que é proporcional ao seu valor de fitness;
- ....

- Os operadores de reprodução devem ainda ser configuráveis através de parâmetros simples, mas que permitam controlar a magnitude das mudanças que introduzem na população.
- Exemplos:
  - Mr – mutation rate
  - Cr – crossover rate

```

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  $\leftarrow$  empty set
    for i = 1 to SIZE(population) do
      x  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      y  $\leftarrow$  RANDOM-SELECTION(population, FITNESS-FN)
      child  $\leftarrow$  REPRODUCE(x, y)
      if (small random probability) then child  $\leftarrow$  MUTATE(child)
      add child to new_population
    population  $\leftarrow$  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

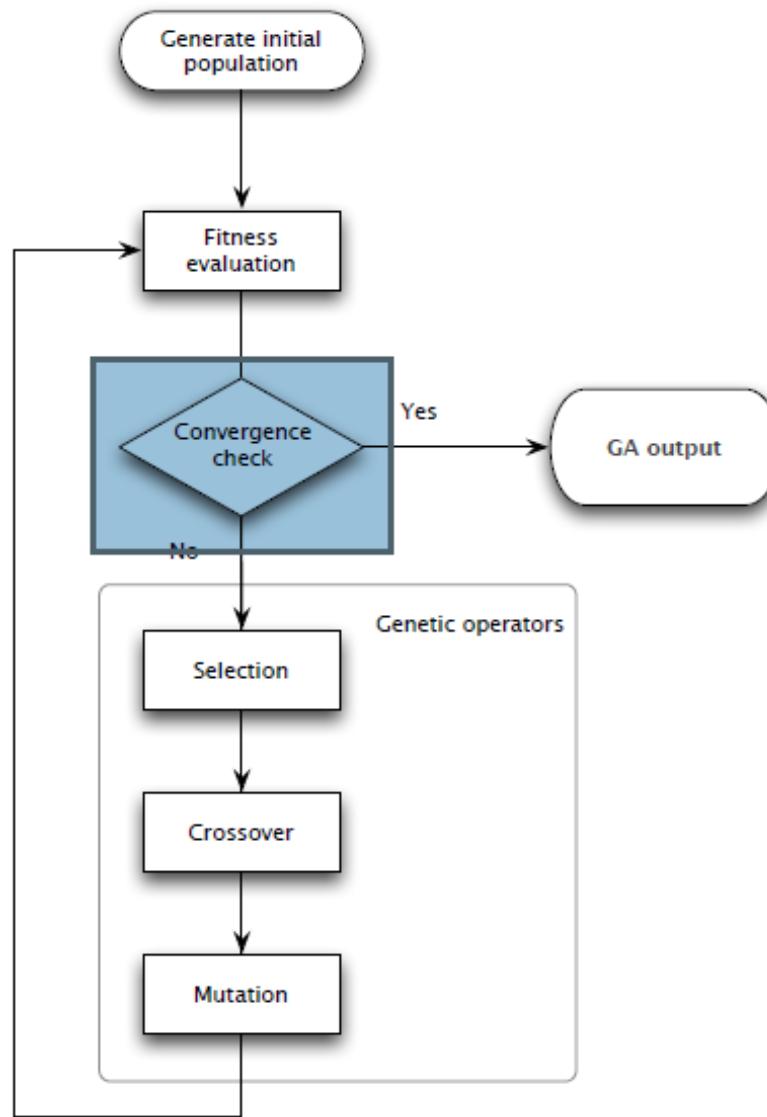
  n  $\leftarrow$  LENGTH(x); c  $\leftarrow$  random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))

```

### Estrutura de um algoritmo genético básico

1. **[Início]** Gerar uma população aleatória de  $n$  cromossomas (apropriados para a solução do problema)
2. **[Fitness]** Avaliar cada cromossoma  $x$  na população através da função de fitness  $f(x)$
3. **[Nova População]** Criar uma nova população através da repetição dos seguintes passos até a população estar completa
  - **[Seleção]** Escolher 2 cromossomas da população de acordo com as suas avaliações de fitness (quanto melhor o fitness melhor a sua probabilidade de ser selecionado)
  - **[Recombinação ou Crossover]** Tendo em conta a probabilidade de crossover, combinando 2 cromossomas formando assim um novo cromossoma.
  - **[Mutação]** Tendo em conta a probabilidade de mutação, mutar cromossomas na população).
  - **[Aceitação]** Por o novo cromossoma na nova população
4. **[Substituir]** Use os novos cromossomas criados na população de cromossomas para correr de novo o programa substituindo cromossomas da população anterior
5. **[Testar]** Se a condição de paragem é satisfeita, parar e devolver a melhor solução encontrada na população corrente.
6. **[Loop]** Ir para o passo 2

## Funcionamento



- Há vários modelos de população diferentes:

- Steady State:

- Em cada iteração são gerados  $N$  novos indivíduos
    - Estes substituem  $N$  indivíduos da geração anterior
    - Também conhecidos como Algoritmos Genéticos incrementais
    - $N$  deve ser inferior ao tamanho da população

- Generational:

- São gerados  $N$  novos indivíduos
    - $N$  é igual ao tamanho da população
    - Toda a população é substituída pela nova geração

- Não é possível “adivinar”, à partida, quais as melhores decisões na implementação de um Algoritmo Genético;
- Assim, o AG deve ser tão configurável quanto possível, para permitir facilmente testar diferentes configurações;
- Parâmetros frequentes incluem:
  - Tamanho da população
  - Número máximo de iterações
  - Estratégia de seleção
  - Peso de cada operador genético na reprodução
  - Fator de mutação (e.g. número entre 0 – sem mutação e 1 – completamente aleatório)
  - ...

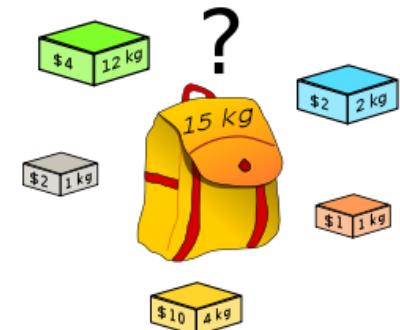
- Tamanho da População (N):
  - uma **população pequena** pode provocar um mau desempenho, por não cobrir adequadamente o espaço do problema, gerando soluções locais;
  - uma **população grande** pode evitar os problemas anteriores, mas pode afetar a eficiência computacional do sistema.
- Taxa de Cruzamento (Cr):
  - quantidade de cromossomas utilizados para cruzamento:  $N \times Cr$ .
- Taxa de Mutação (Mr):
  - a mutação é utilizada para aumentar a variabilidade da população;
  - cada gene tem uma probabilidade finita de mudar;
  - uma **baixa taxa** de mutação permite que um gene “gele” num valor;
  - uma **alta taxa** de mutação resulta numa procura aleatória de soluções;
  - sendo L o comprimento do cromossoma, ocorrerão  $Mr \times N \times L$  mutações.

- Taxa de Substituição (Generation Gap - Gr):
  - controla a percentagem da população a ser substituída em cada geração;
  - são substituídos  $N \times Gr$  estruturas da população;
  - se  $Gr = 1$ , significa que toda a população deve ser substituída durante cada geração.
- Estratégia de Seleção:
  - a reprodução é feita com base na proporção do valor de *fitness* das estruturas da população atual;
  - as estruturas com melhor desempenho são as que passam para a próxima geração.
- Função de Avaliação:
  - serve para manter a diversidade da população durante a evolução;
  - o surgimento de um “super cromossoma” dominante na população deve ser evitado, para que o problema seja eficientemente explorado.

- Os algoritmos Genéticos não garantem uma solução ótima;
  - Assim, é impossível saber quando parar.
- Torna-se necessário definir um ou mais critérios de paragem:
  - Atingir um número pré-definido de iterações;
  - Atingir um valor de fitness;
  - ...
- Quando o algoritmo termina, escolhe-se a melhor solução encontrada ao longo da evolução.
  - Não existe necessariamente na última geração...

- Funções multimodais;
- Funções discretas ou contínuas;
- Funções multidimensionais, incluindo combinatória;
- Dependência não linear de parâmetros;
- Frequentemente usado para resolver problemas NP;
- Não use AG quando outro método, como hill-climbing, etc., funciona bem ou pelo menos antes de tentar esse tipo de método!

- O problema da mochila (Knapsack problem) é um problema de otimização combinatória;
- Pretende-se preencher uma mochila com objetos de diferentes pesos e valores;
- O objetivo é que se preencha a mochila com o maior valor possível, não ultrapassando o peso máximo;
- O problema da mochila é um dos 21 problemas NP-completos de Richard Karp, expostos em 1972;
- A formulação do problema é extremamente simples, porém sua solução é complexa;
  
- Exemplo: O problema de mochila unidimensional: que caixas devem ser escolhidas para maximizar a quantidade de dinheiro mantendo o peso total abaixo ou igual a 15 kg?
- Um problema de restrição múltipla poderia considerar tanto o peso quanto o volume das caixas.



# Knapsack

**Exemplo:**

**N = 8**

**Capacidade (C) = 50**

Objectivo

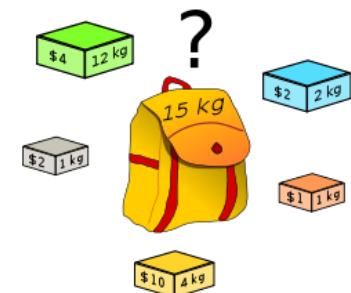
Maximizar  $\sum_i O_i \times v_i$

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Restrição  $\sum_i O_i \times p_i \leq 50$

Representação das soluções:  $O_i$

1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---



## Soluções Inválidas

Soluções Inválidas: Como resolver?

- Reparar
- Alterar a representação
- Penalizar

Uma função de penalização define em que quantidade a solução X viola a restrição R

Mede a distância a que a solução está de uma região de aceitação

Transforma um problema com restrições num problema sem restrições

## Exemplo - Knapsack

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Avaliação

$$f(s) = \begin{cases} \sum_i o_i \times v_i & \text{se solução válida} \\ 0 & \text{se solução não válida} \end{cases}$$

$$f(s) = \sum_i o_i \times v_i - p(s)$$

Onde  $p(s) = \begin{cases} \alpha \left( \sum_i o_i \times v_i \right) - C & \text{se solução não válida} \\ 0 & \text{se solução válida} \end{cases}$

A penalização de uma solução válida é zero e é proporcional ao grau de violação das restrições

## Exemplo - Knapsack

	1	2	3	4	5	6	7	8
Valor	4	3	6	7	2	9	7	6
Peso	12	16	8	21	16	11	6	12

Criar e avaliar a população inicial

Solução									Valor	Peso	p(s)	f(s)
1	1	0	0	0	0	1	0		14	34	0	14
0	0	0	1	1	0	1	1		22	55	5	17
0	1	0	1	0	0	1	0		17	43	0	17
1	0	0	0	0	0	1	0		11	18	0	11
0	1	1	1	1	0	1	0		25	67	17	8
1	0	0	1	0	0	1	1		24	51	1	23
0	1	0	1	0	1	1	0		26	54	4	22
1	1	0	0	0	0	1	1		20	46	0	20

### Recombinar soluções

Combinar o material genético de dois progenitores para gerar novas soluções

Objetivo: Combinar características interessantes de duas soluções

#### Recombinação por ponto de corte

- Alinhar os dois progenitores
- Selecionar um ponto de corte aleatório
- Combinar seções complementares para obter descendentes



## Recombinação

População  
inicial

Solução (progenitores)										Valor	Peso	p(s)	f(s)
1	1	0	0	0	0	1	0			14	34	0	14
0	0	0	1	1	0	1	1			22	55	5	17
0	1	0	1	0	0	1	0			17	43	0	17
1	0	0	0	0	0	1	0			11	18	0	11
0	1	1	1	1	0	1	0			25	67	17	8
1	0	0	1	0	0	1	1			24	51	1	23
0	1	0	1	0	1	1	0			26	54	4	22
1	1	0	0	0	0	1	1			20	46	0	20



Recombinação

Solução (filhos)										Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0			16	43	0	16
1	0	0	0	0	0	1	1			17	30	0	17
0	1	0	1	0	0	1	1			23	55	5	18
1	1	0	0	0	1	1	0			23	45	0	23

## Mutação

Alteração do valor de um gene (mutação binária)



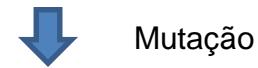
Cria variabilidade no conjunto das soluções

Introduz alterações no material genético

**Objetivo:** Introduzir diversidade na população

População  
filhos

Solução (filhos)									Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0		16	43	0	16
1	0	0	0	0	0	1	1		17	30	0	17
0	1	0	1	0	0	1	1		23	55	5	18
1	1	0	0	0	1	1	0		23	45	0	23

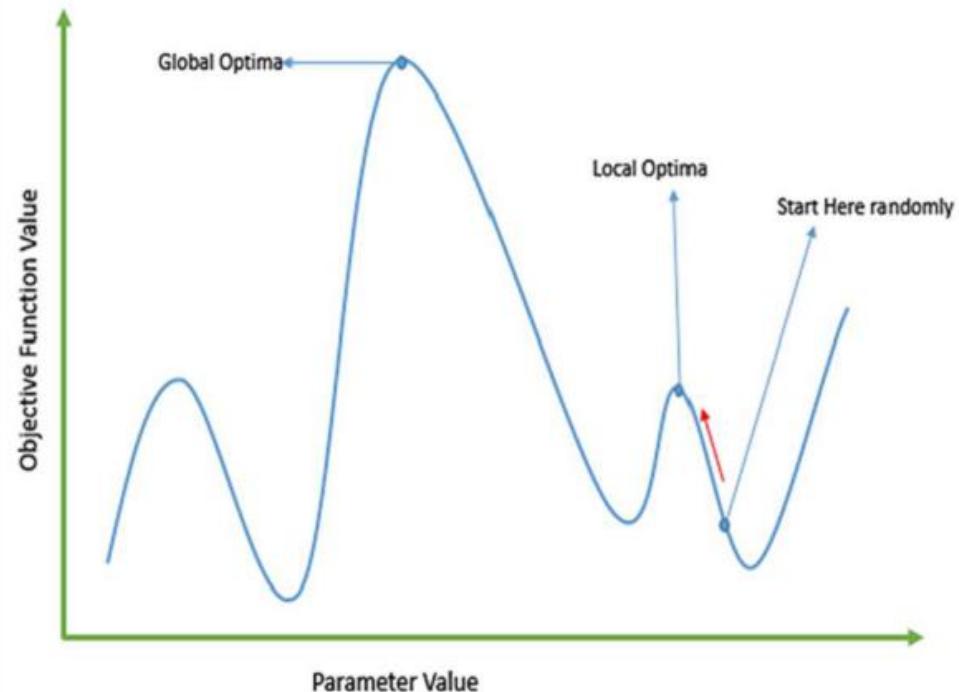


Solução (filhos)									Valor	Peso	p(s)	f(s)
0	0	0	1	1	0	1	0		16	43	0	16
1	0	0	0	0	1	1	1		26	41	0	26
0	1	0	1	0	0	1	1		23	55	5	18
1	1	0	0	0	1	1	0		23	45	0	23

- Não necessita muito conhecimento do domínio;
- É mais rápido e eficiente que alguns métodos tradicionais;
- Pode ser facilmente paralelizável;
- Otimiza funções discretas e contínuas;
- Encontra múltiplas “boas” soluções;
- Encontra sempre uma solução, mesmo que não seja a melhor;
- Útil quando o espaço da solução é muito grande.

## Desvantagens

- Não são aplicáveis a qualquer problema;
- O cálculo repetido da função de fitness pode ser pesado computacionalmente;
- Não garante soluções ótimas nem a qualidade da solução encontrada;
- Se mal implementado, pode não convergir para a solução ótima.



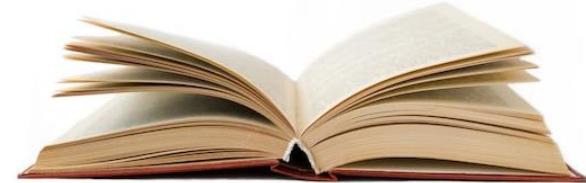
## Referências Algoritmos Genéticos

### Bibliografia Recomendada

- Holland, John H. "Genetic Algorithms." *Scientific American* 267 (1): 66–73, 1992.
- David Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, 1989.
- Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach, 3rd edition, ISBN-13: 9780136042594.

### Outro material

- Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Science & Business Media, 2013.
- Yang, Xin-She. *Nature-Inspired Optimization Algorithms*. Elsevier, 2014.
- Miguel Rocha, José Neves, "Computação Genética e Evolucionária". Universidade do Minho. 2000





Universidade do Minho  
Escola de Engenharia  
Departamento de Informática

Otimização e Procura

Algoritmos *Population-based*

Algoritmos Genéticos

LICENCIATURA EM ENGENHARIA INFORMÁTICA  
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

Inteligência Artificial

2022/23