UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LABIC

Laboratório de Bioinformática e Inteligência Computacional - UTFPR

http://labic.utfpr.edu.br

Computação Evolucionária

Prof. Heitor Silvério Lopes

hslopes@utfpr.edu.br

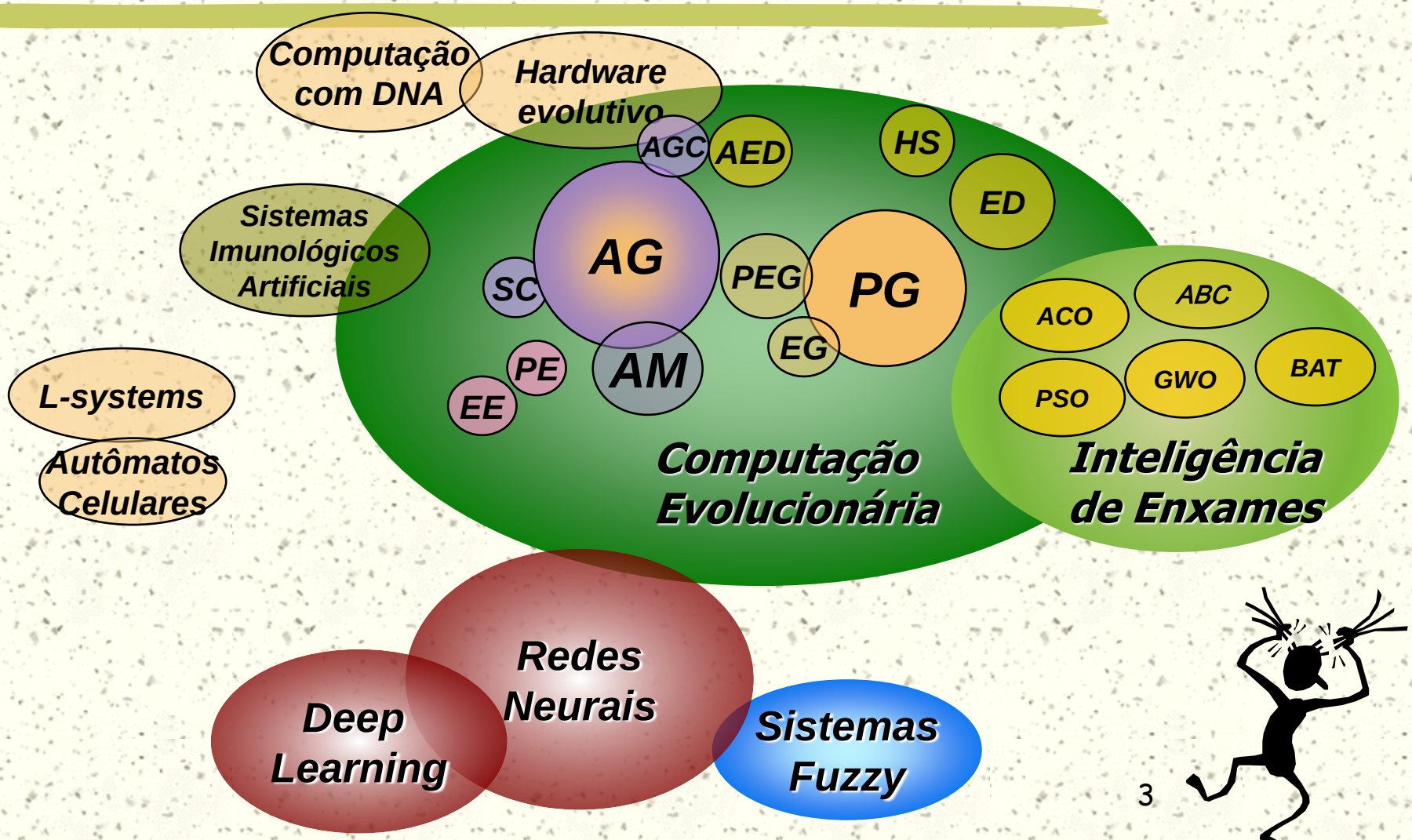
http://silverio.net.br/heitor

class1a (2020)

IA x IC

- Inteligência Artificial (IA)
  - Utiliza modelos computacionais inspirados no raciocínio humano
- Inteligência Computacional (IC):
  - Utiliza modelos computacionais inspirados na natureza (métodos bioinspirados)
  - Também conhecida como Computação Natural ou *Soft Computing*

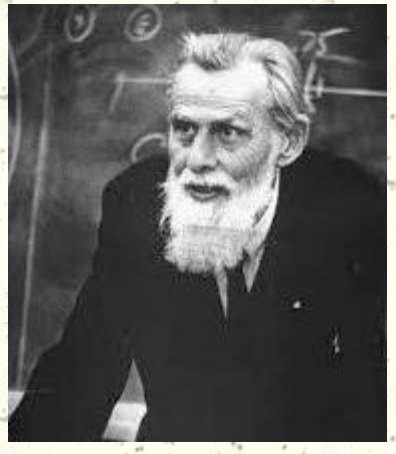
Inteligência Computacional



Sistemas *fuzzy*

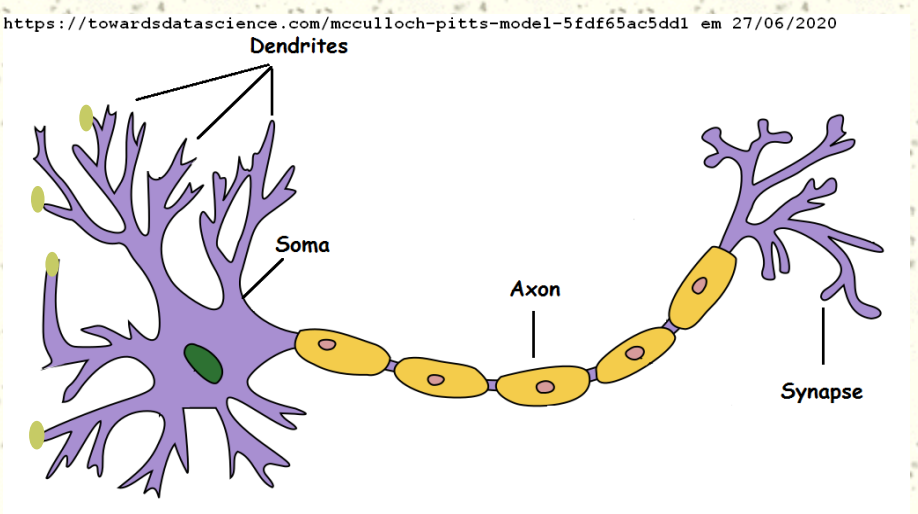
- Inspiração biológica: imprecisão do raciocínio humano
- A lógica *fuzzy* foi criada por **Lotfali Askar-Zadeh** na década de 60
- É uma forma de tratar incertezas e imprecisões inerentes ao conhecimento.
- Utiliza variáveis linguísticas em vez de variáveis numéricas.
- Tem um sólido embasamento teórico

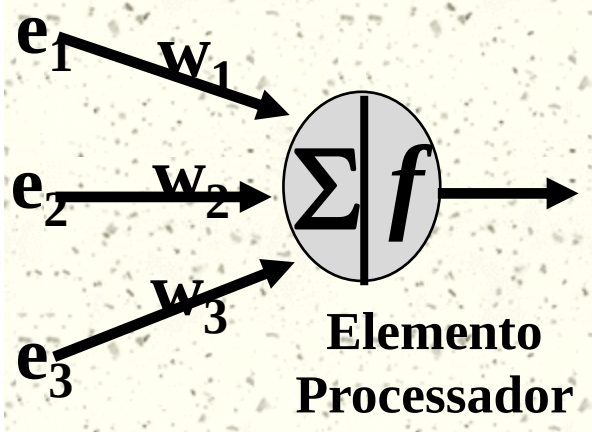
Redes Neurais



- Inspiração biológica: funcionamento dos neurônios
- Precursores: **W. McCulloch e W.Pitts, D.O. Hebb, J.L.McClelland**
- **F. Rosenblatt** → Primeiro modelo matemático de um neurônio
- Desenvolvimento nas décadas de 50 a 80

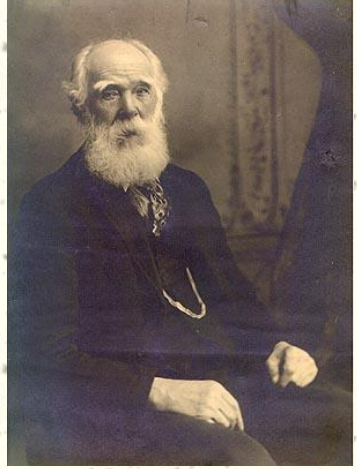
https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1 em 27/06/2020





$$s_1 = f\left(\sum w_i . e_i\right)$$

Computação evolucionária



- Inspiração biológica: Evolução das espécies
- Charles Darwin, 1859:
  - "Sobre a origem das espécies por meio de seleção natural"
  - A evolução dos seres vivos é baseada no Princípio da Seleção Natural:
    - Indivíduos mais fortes e mais bem-adaptados ao ambiente têm maior chance de sobrevivência e de dar continuidade à sua espécie

Seleção natural e evolução

Importante !

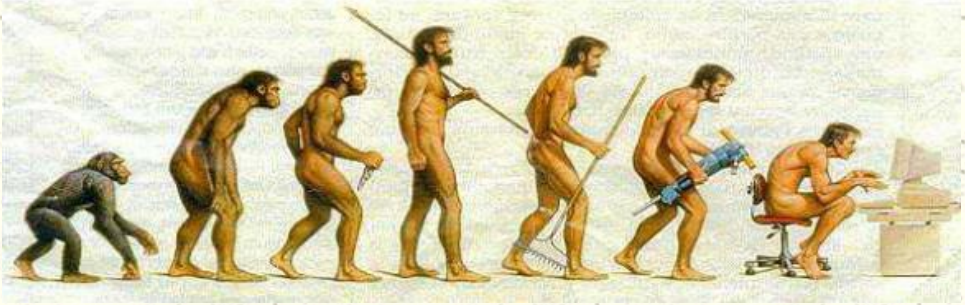
- A seleção natural é probabilística
- A seleção natural atua sobre os indivíduos de uma espécie
- A consequência a longo prazo é a evolução da espécie
- Evolução (adaptação) como um processo "inteligente" de otimização
- Inspiração para a construção de paradigmas computacionais que imitem a evolução das espécies, como forma de otimização

Hipóteses de Darwin relevantes para a Computação Evolucionária

- H1: Dentro de uma mesma espécie os indivíduos apresentam pequenas diferenças entre si
  - Há diferenças genotípicas/fenotípicas
- H2: Algum processo de variação continuada deve ser responsável pela introdução de novas informações na carga genética dos indivíduos
  - Ocorrem mutações não-determinísticas
- H3: Não há limite para a introdução sucessiva de variações genéticas
  - Processo continuado de evolução

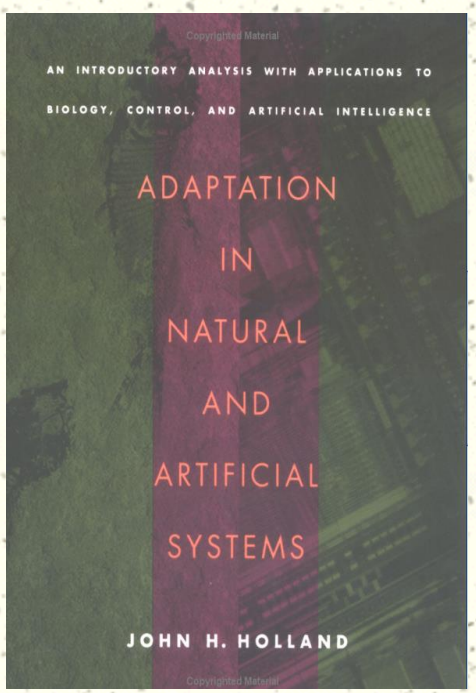
História da Computação Evolucionária

- As primeiras ideias surgiram em 1957-1962, porém havia pouco ou nenhum embasamento teórico, suporte computacional incipiente, e ceticismo da comunidade científica.
- Décadas de 60/70 houve a emergência, de forma independente, das principais linhas de pesquisa:
  - Algoritmos Genéticos (AG)
  - Programação Evolucionária (PE)
  - Estratégias Evolucionárias (EE)



História da Computação Evolucionária

Algoritmo Genético (AG)



- Criado por John Holland na Universidade de Michigan (EUA) na década de 60
- Publicou o seu livro em 1975: "**Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**"
- "Sistema adaptativo artificial"; "planos reprodutivos generalizados"

- Foi o orientador de doutorado de vários pesquisadores importantes da área, incluindo Rick Riolo, John Grefenstette, David Goldberg, Erick Goodman, **Kenneth DeJong** e John Koza


DeJong
2012
SBIA

História da Computação Evolucionária

Programação Evolucionária (PE)

- Desenvolvido por Lawrence Fogel, Owens e Walsh em 1966 na Universidade da Califórnia em San Diego
- Objetivo: aplicar à evolução simulada de máquinas de estados finitos para a previsão de séries não-estacionárias.
- Teve seu auge na década de 80, mas caiu em desuso

Estratégias Evolucionárias (EE)

- Desenvolvido por Bienert, Ingo Rechenberg e Hans-Paul Schwefel em 1965 na Universidade Técnica de Berlin
- Foi desenvolvido para otimização numérica em problemas de *design* industrial
- Também caiu em desuso na década de 90

Escopo da Computação Evolucionária

- O escopo de aplicação de CE é em problemas de **Busca e Otimização**
- Grandes áreas de aplicação:
  - **Engenharias**
  - **Computação**
  - **Matemática aplicada**
- Problemas de outras naturezas:
  - Modelagem como um problema de otimização.
  - P.ex. Aprendizado: max(qtd. info. absorvidas); min(qtd. erros conceituais); max(capacidade de extrapolação), min(tempo aprendizado)....

Otimização

- Definição de otimização:
  - "Processo de melhoramento iterativo de uma solução para um problema, com respeito a uma função objetiva específica."
- Problemas típicos da área de otimização:
  - Maximização (ou minimização) de funções algébricas
  - Problemas combinatoriais
    - ex. Problema do caixeiro viajante, problema da mochila
  - Projetos de engenharia:
    - maximização de desempenho
    - minimização de custo

Métodos para otimização

- Métodos fortes:
  - Para problemas específicos onde há linearidade, diferenciabilidade, estacionariedade
  - Em geral garantem a obtenção da solução ótima
- Métodos específicos:
  - Para problemas muito particulares
- Métodos fracos:
  - Para problemas genéricos onde pode existir não-linearidade, não-estacionariedade (etc).
  - Não garantem a obtenção da solução ótima, só eventualmente podem fornecer uma solução satisfatória

Métodos de otimização

- Numéricos:
  - Analíticos: derivadas parciais = 0
  - Diretos: técnicas de gradiente (*steepest descent* ou *hill-climbing*)

- Enumerativos:
  - Busca exaustiva
  - Programação dinâmica

- Probabilísticos: →Heurísticas
  - Busca aleatória
  - *Simulated annealing*
  - Computação Evolucionária & Inteligência de Enxames

Métodos enumerativos

- Excelentes para um grande número de problemas, entretanto:
  - Aplicável somente a problemas de "dimensões pequenas"
  - Aceitável quando envolve tempos computacionais "razoáveis"
- Tendem a ser cada vez mais utilizados, à medida que a capacidade computacional disponível aumenta
- Não servem para problemas com complexidade Não-Polinomial (NP)

"Ingredientes" de problemas de otimização

- Função objetivo:
  - a qual se quer maximizar ou minimizar.
  - exemplos: max(lucro), min(custo)
  - pode não existir ou ser múltipla
- Conjunto de variáveis:
  - que afetam o valor da função objetivo
  - em problemas interessantes, este conjunto pode ser muito grande
- Conjunto de restrições:
  - não permite que o conjunto de variáveis assuma determinados valores

"Ingredientes" de problemas difíceis

- Multimodalidade (máximos locais)
- Grande conjunto de restrições
- "Ruído"
- "Deceptividade" (*deception*)
- Isolamento do ótimo desejado (*Royal Road*)
- Não-estacionariedade (variação dinâmica)
- Espaço de busca **arbitrariamente** grande

Quando pode ser interessante utilizar uma técnica de CE ?

- Quando a complexidade do problema torna inviável a sua formulação matemática e o tratamento por métodos algébricos
- Quando o número de possíveis soluções a serem examinadas leva à uma explosão combinatória intratável por métodos exaustivos
- Quando o problema é fortemente não-estacionário
- Quando o custo computacional e a qualidade da solução sub-ótima forem aceitáveis
- Quando não existir outra alternativa viável !

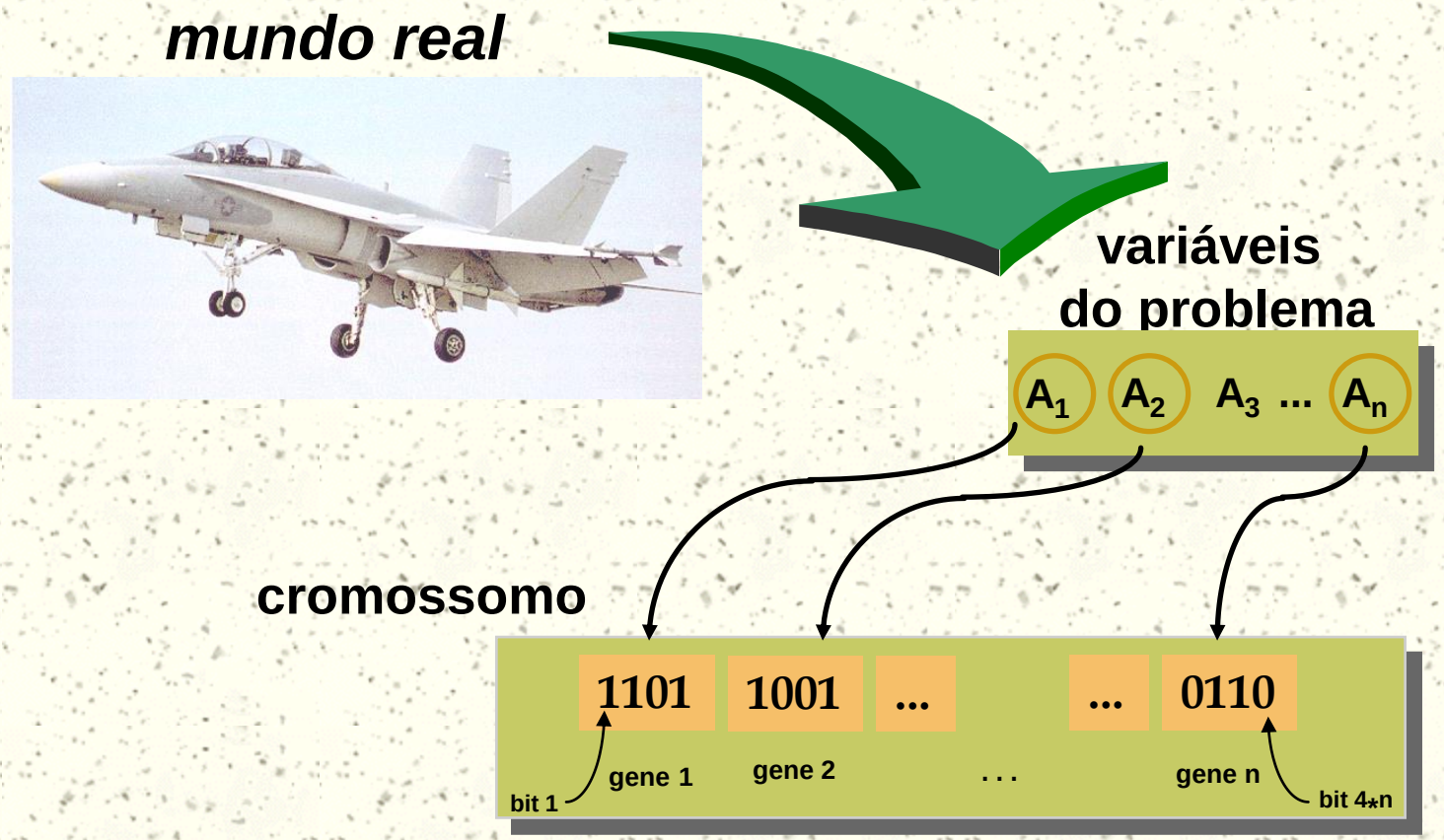
Princípio dos Algoritmos Evolutivos

- Abordagem "intuitiva": tentativas e refinamentos sucessivos:

1 gere soluções para o problema
2 avalie as soluções
3 se a melhor solução satisfaz, pare
4 selecione as melhores soluções
5 construa novas soluções utilizando partes das melhores soluções
6 eventualmente modifique as soluções
7 volte ao passo 2

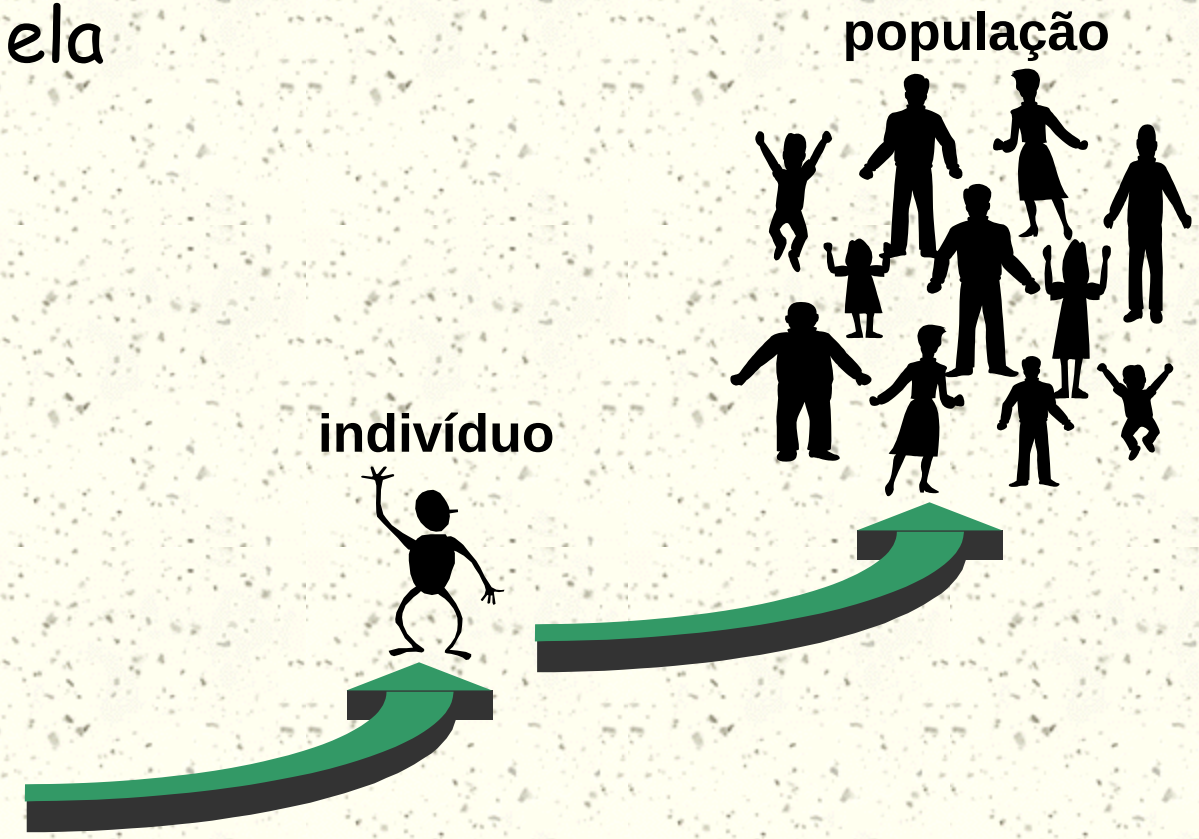
Algoritmo genético #1:
Codificação das variáveis do problema

- Mapeamento fenótipo x genótipo



Algoritmo genético #2:
População" de possíveis soluções

- Busca paralela

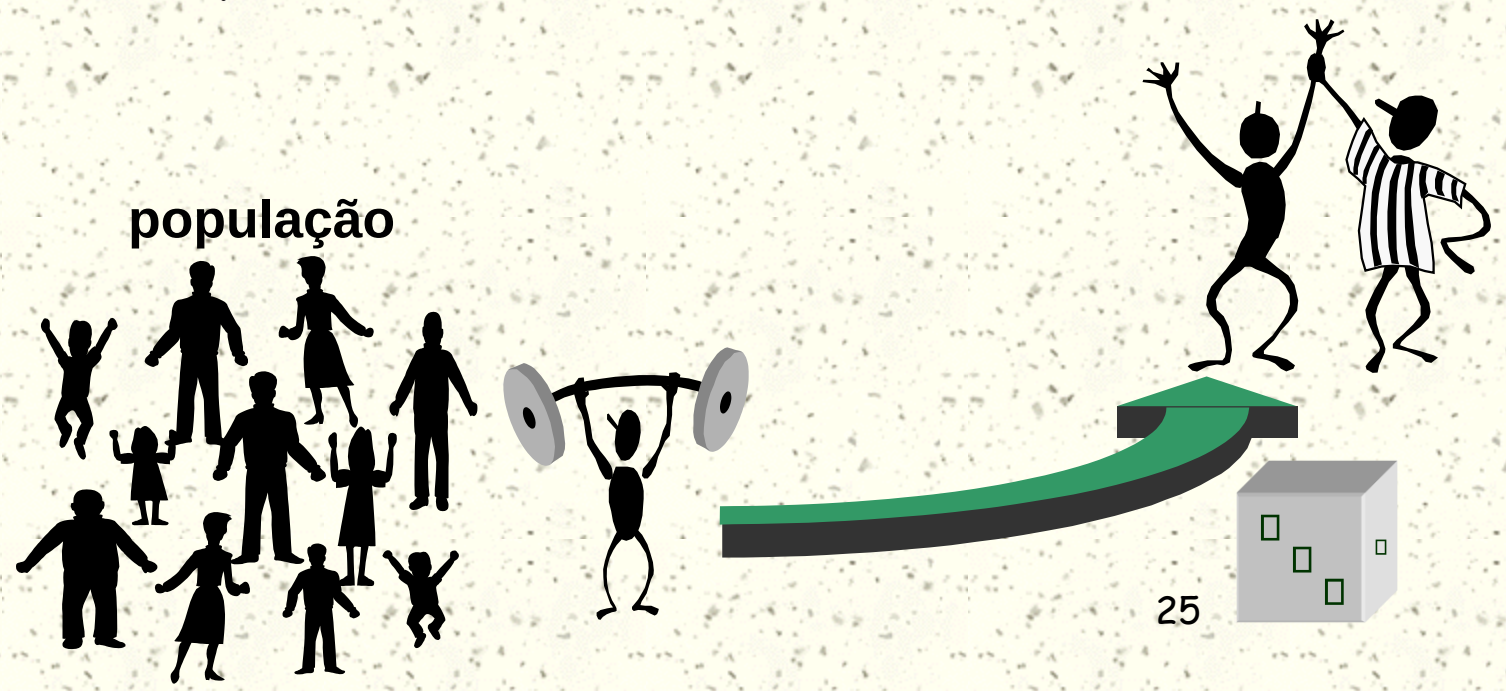


Algoritmo genético #3:
Função de *fitness*

- Avalia a qualidade da solução para o problema

Algoritmo genético #4:
Seleção

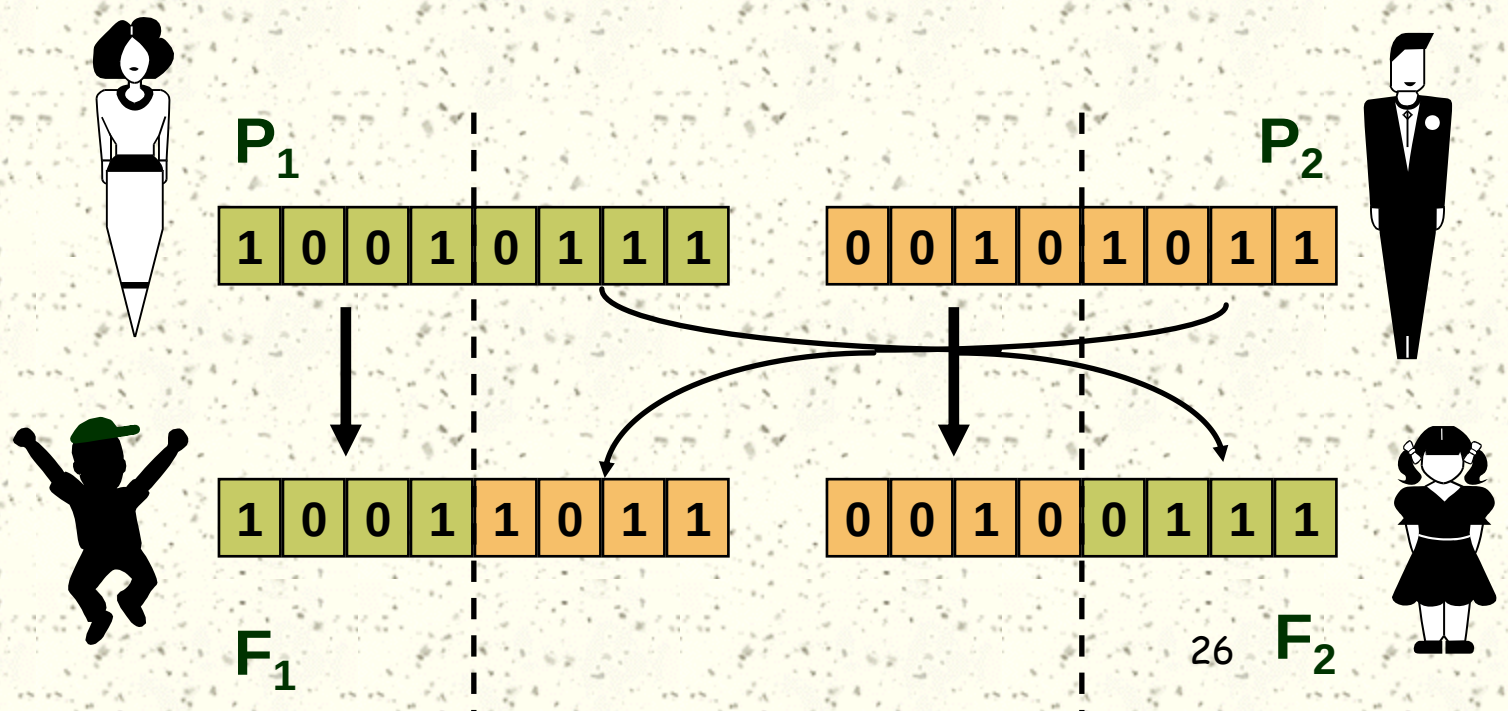
- Implementa e guia o processo evolutivo
- Seleciona os indivíduos mais aptos a se reproduzir e passar o seu material genético
- Métodos probabilísticos ou determinísticos



Algoritmo genético #5:

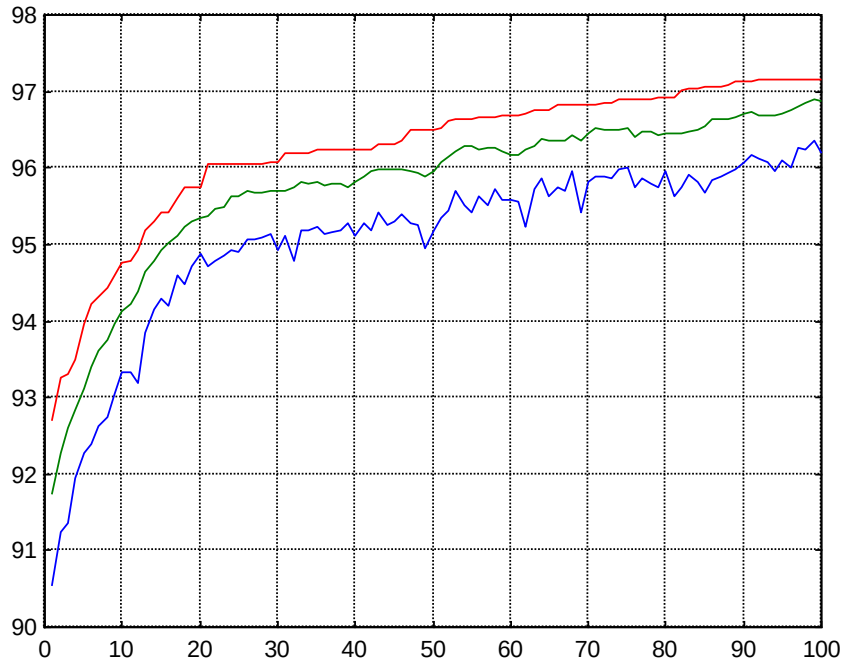
Reprodução e operadores genéticos

- *Crossover*: recombinação de material genético de dois pais - realiza busca local
- Mutação: variação aleatória - realiza busca global



Algoritmo genético #6:
Critério de parada do algoritmo

- Número predeterminado de gerações
- Quando não há mais melhora
- Quando não há mais diversidade
- Quando atingiu o ótimo



Diferenças entre AG e outros métodos de otimização

- Codificação: (fenótipo ←→ genótipo)
  - AG's trabalham com conjuntos de parâmetros codificados e não os parâmetros propriamente ditos.
- Busca paralela: (paralelismo implícito)
  - AG's realizam a busca utilizando uma população de pontos simultaneamente, não um único ponto
- Busca probabilística
  - AG's utilizam regras de transição probabilísticas, não determinísticas.

CE como método de engenharia

- A otimização é um dos princípios da Engenharia
- Projeto de engenharia:
  - Muitos parâmetros a serem otimizados
  - Muitos graus de liberdade
  - Muitas restrições a serem satisfeitas
- CE:
  - Facilita a tarefa de projeto gerando soluções subótimas ou mesmo ótimas em um tempo razoável
- CE emprega um método intuitivo:
  - Processo criativo
  - Refinamento iterativo

Vantagens das técnicas de CE

- Não requerem um conhecimento matemático profundo do problema ao qual é aplicado.
- Têm robustez
- Requerem pouco esforço de implementação
- São facilmente hibridizáveis com outras técnicas
- São facilmente adaptáveis a muitas classes de problemas, inclusive problemas multiobjetivos
- São implicita e explicitamente paralelizáveis
- São capazes de manipular de restrições

Desvantagens das técnicas de CE

- Não garantem encontrar a solução ótima em tempo finito
- Há pouco embasamento teórico - a prática se desenvolveu mais do que a teoria
- O ajustes dos parâmetros de controle requer conhecimento prévio, calibração com experimento fatorial ou tentativa-e-erro
- Não são intrinsecamente melhores do que qualquer outro algoritmo de otimização ("*No free-lunch theorems*")

"Não existem métodos fáceis para resolver problemas difíceis."

René Descartes