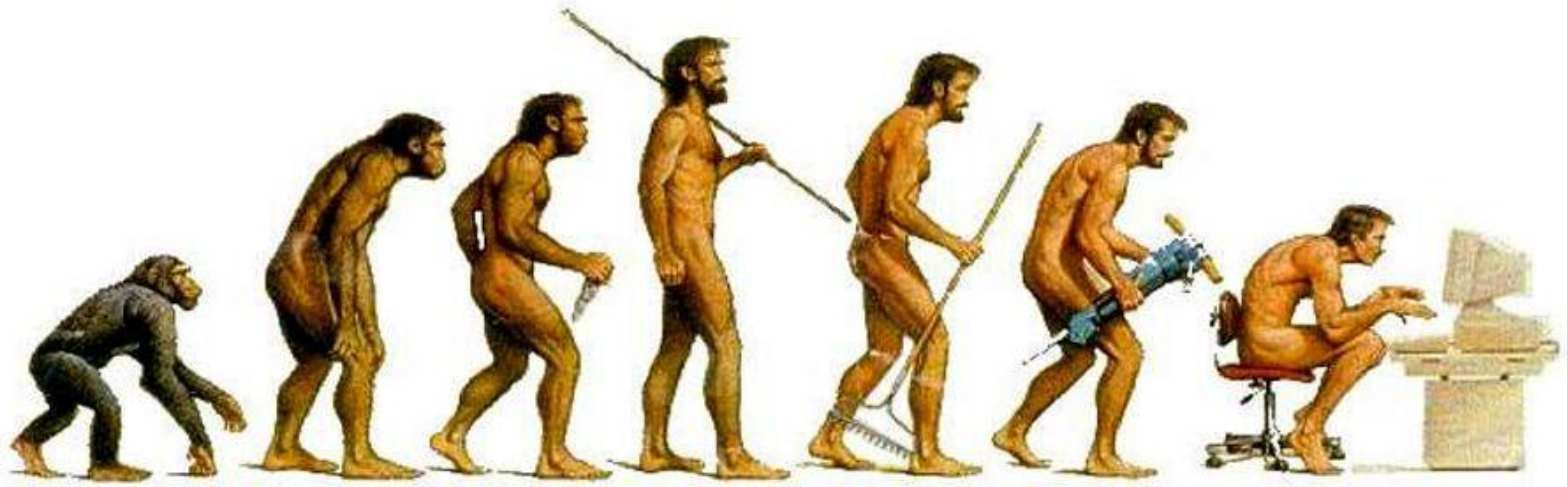
Computação Evolutiva

Parte I



Fabricio Breve – fabricio.breve@unesp.br

Aula Anterior

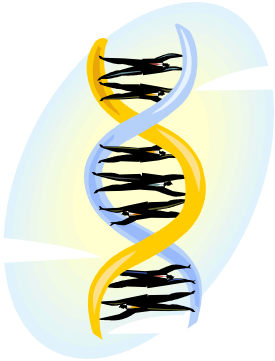
- Introdução à Computação Inspirada pela Natureza
  - Motivação
  - Áreas da Computação Natural
  - Filosofia e Objetivos
  - Marcos
  - Exemplo de Aplicação
  - Quando Utilizar

- Disciplina
  - Objetivos
  - Requisitos
  - Programa
  - Método de Avaliação
  - Apresentação dos Alunos

Agenda

- Introdução à Computação Evolutiva
- Resolução de Problemas como uma Tarefa de Busca
  - Escolha de Representação
  - Especificação de Objetivo
  - Escolha de Função de Avaliação
  - Soluções Factíveis
  - Minimização e Maximização
  - Ótimos Locais e Global
  - Explotação versus Exploração
  - Cenários de Aptidão
- Subida da Colina
- Subida da Colina Probabilístico
- Recozimento Simulado
- Exemplo de Aplicação
- Exercícios

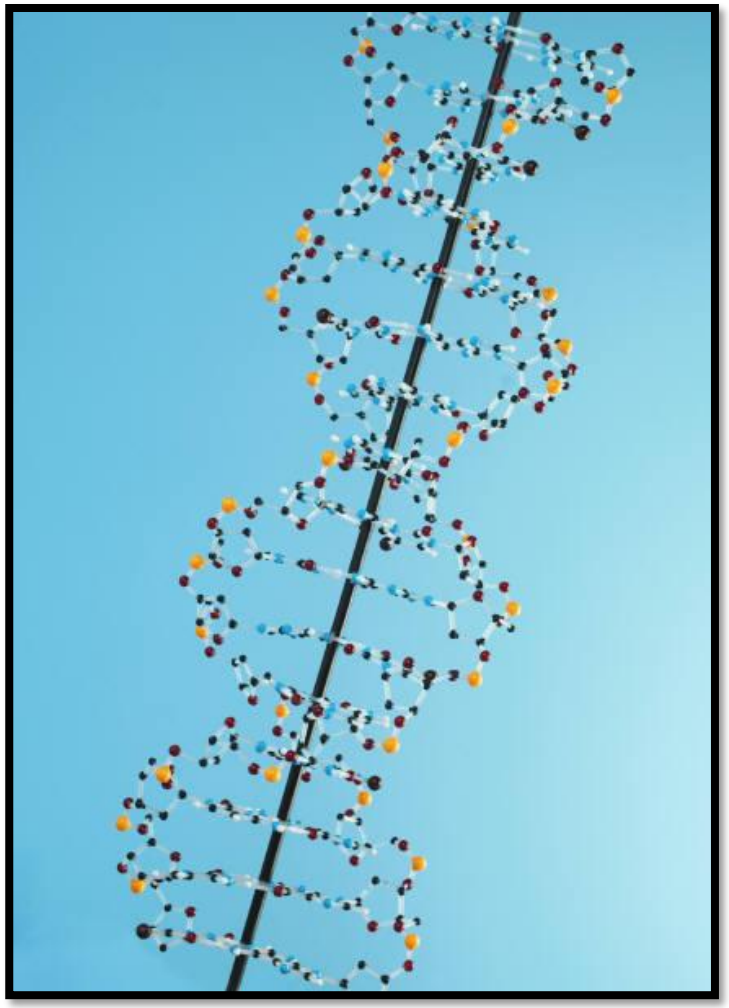
Introdução



- **Computação Evolutiva**
  - Campo de pesquisa que utiliza ideias da biologia evolutiva para desenvolver técnicas para resolução de sistemas complexos.
  - A maioria das ideias tem sua origem na teoria da evolução de Darwin e na genética.
    - Propõe que uma população de indivíduos capazes de se reproduzir e sujeito a variações (genéticas), resulta em novas populações de indivíduos cada vez mais adaptados ao ambiente.

Introdução

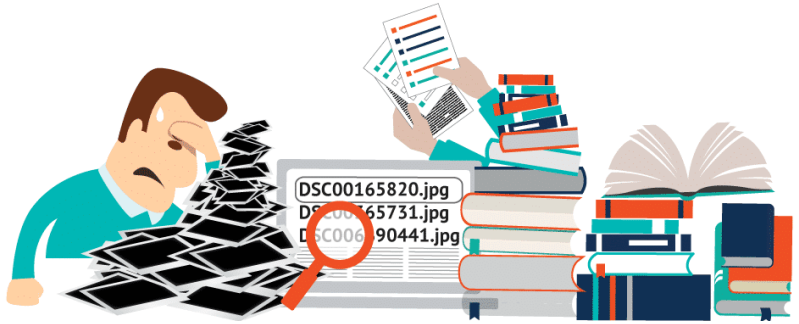
- **Algoritmos Evolutivos**
  - Baseados na teoria de origem e diversidade de vida.
- Tipos:
  - **Algoritmos Genéticos**
  - Estratégias Evolutivas
  - Programação Evolutiva
  - Programação Genética



RESOLUÇÃO DE PROBLEMAS COMO UMA TAREFA DE BUSCA

Resolução de Problemas como uma Tarefa de Busca

- Nesse contexto um problema é uma coleção de informações das quais alguma coisa (ex.: conhecimento) será extraída ou inferida.
- Exemplos:
  - Função numérica a ser maximizada.
  - Problema de alocação de um número de aulas para um conjunto de alunos.



Resolução de Problemas como uma Tarefa de Busca

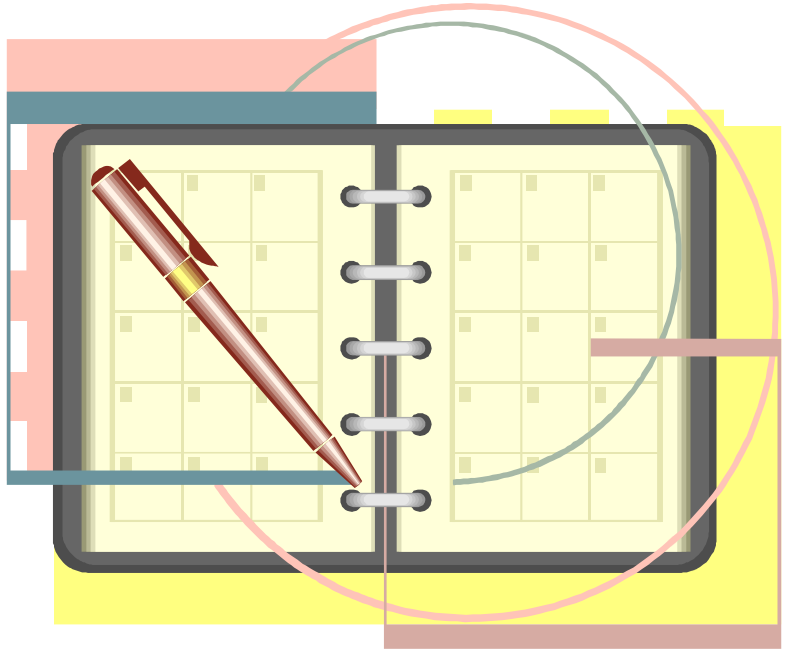
- Problema 1
  - Informação:
    - Função a ser otimizada

$$f(x) = x^3 + x + 3$$

  - Objetivo:
    - Determinar os valores de x que minimizem esta função.

Resolução de Problemas como uma Tarefa de Busca

- Problema 2
  - Informação:
    - Número de estudantes,
    - Salas de aula,
    - Professores,
    - Etc.
  - Objetivo:
    - Encontrar um horário para todas as aulas de uma faculdade em um semestre.



Resolução de Problemas como uma Tarefa de Busca

- Resolver o problema é tomar ações (passos) ou uma sequência de ações, que:
  - Levam ao desempenho desejado.
  - Aumentam o desempenho relativo dos *indivíduos*.
- Isto é chamado *busca*.

Resolução de Problemas como uma Tarefa de Busca

- Um algoritmo de busca pega um problema como entrada e retorna uma solução para ele.
- Um ou mais indivíduos (conhecidos como *agentes*) são usados como *soluções candidatas* para o problema.
- Algum conhecimento sobre o desempenho desejado de um indivíduo nem sempre está disponível.
  - Mas ainda pode ser possível avaliar a qualidade relativa dos indivíduos que estão sendo usados como meio de resolver o problema.

Foto: Fabricio Breve

Foto: Fabricio Breve

Resolução de Problemas como uma Tarefa de Busca

- A primeira etapa de resolução de um problema é a formulação, que dependerá da informação disponível.
  - Principais conceitos envolvidos:
    - Escolha de representação.
    - Especificação do objetivo.
    - Definição de uma função de avaliação.

Escolha de Representação

- Codificação das soluções candidatas (*indivíduos*) para manipulação.
- É de vital importância, sua interpretação implica no espaço de busca e seu tamanho.
- O espaço de busca (ou de *estados*) é definido pelo estado (*configuração*) inicial e o conjunto de estados (*configurações*) possíveis do problema.

Especificação do Objetivo

- Descrição dos propósitos a serem preenchidos.
  - Expressão matemática da tarefa a ser cumprida.
- Exemplo:
  - Se o objetivo é minimizar a função:

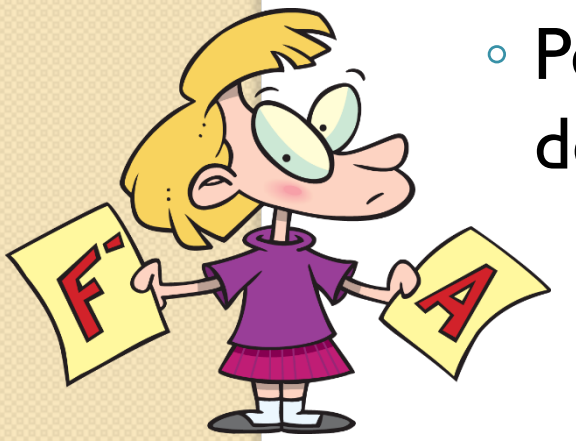
$$f(x) = x^3 + x + 3$$

  - Então o objetivo pode ser definido como:

$$\min f\,(x)$$

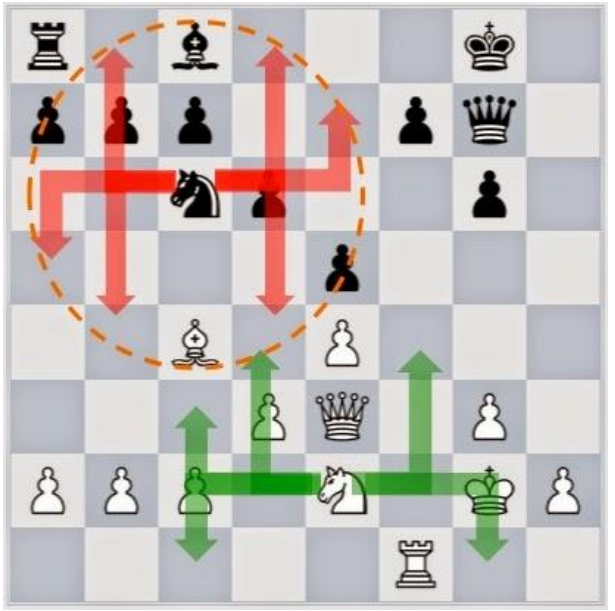
Definição de uma Função de Avaliação

- Uma função que retorna um valor específico indicando a qualidade de qualquer solução candidata (individual), dada a sua representação.
- Quando não há conhecimento sobre o desempenho desejado, a função de avaliação pode ser usada como um meio de avaliar a qualidade relativa dos indivíduos.
  - Permitindo a escolha dos indivíduos de maior qualidade dentro de um conjunto de soluções candidatas.



Soluções Factíveis

- Dado um espaço de busca *S*, assuma a existência de algumas *restrições*, que se violadas inviabilizam a implementação de uma solução.
- Na busca por uma solução melhorada temos de conseguir mover de uma solução para outra sem violar tais *restrições*.
  - Precisamos de operadores que gerem ou selecionem soluções *factíveis*.



Definindo um Problema de Busca

- Definição de um problema de busca (ou otimização):

Dado um espaço de busca S, junto com sua parte factível F, $F \subseteq S$, encontre $x^* \in F$ tal que $eval(x^*) \leq eval(x), \forall x \in F$.

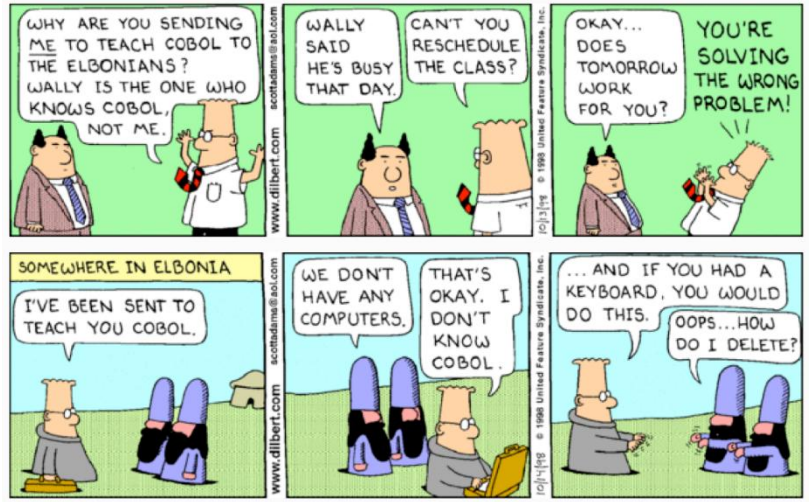
Minimização versus Maximização

- No caso do slide anterior, a solução x para a qual a função de avaliação retorna o menor valor é considerada a melhor.
  - Problema de minimização.
- Também poderíamos usar uma função em que soluções que retornem valores maiores são melhores.
  - Problema de maximização.
    - Minimizando o negativo de $f(x)$ temos uma função de maximização:

$$\max f(x) = \min - f(x)$$

Definindo um Problema de Busca

- O processo de busca não sabe qual é o problema que está sendo resolvido.
  - Ele sabe apenas a informação dada pela função de avaliação, representação usada e como é feito o teste das possíveis soluções.
- Se a função de avaliação não corresponde ao objetivo você estará procurando pela *resposta certa* para o *problema errado*!

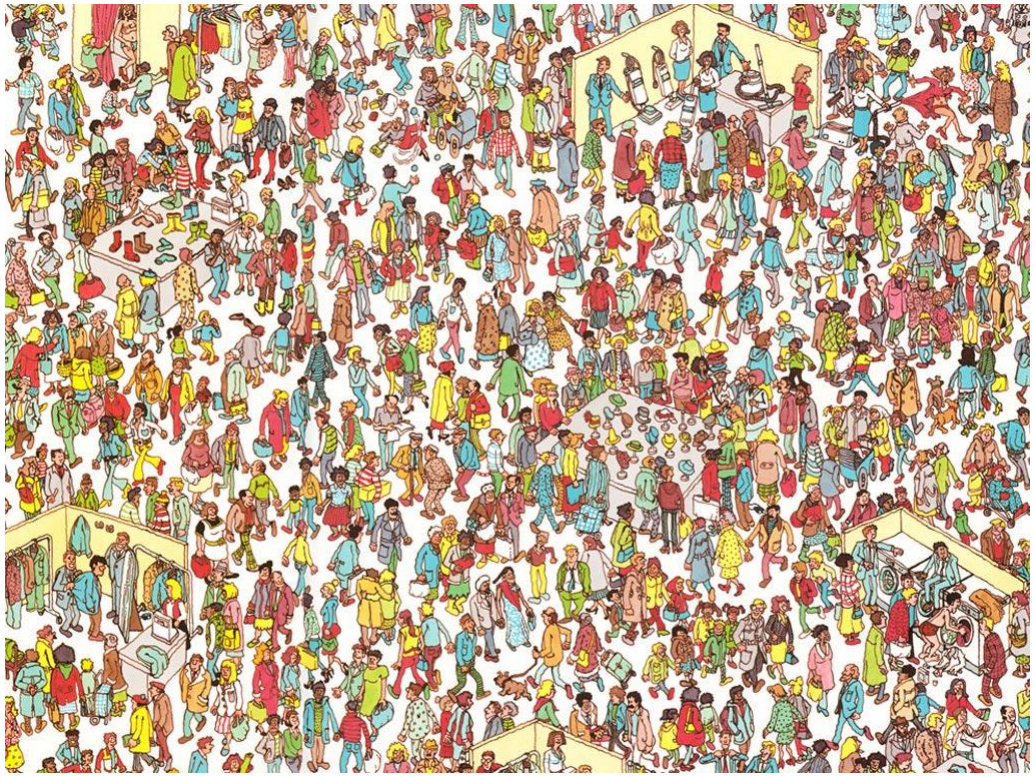


"Uma resposta aproximada para a pergunta certa vale muito mais do que uma resposta precisa para a pergunta errada"

John Tukey – estatístico americano [*1915 – †2000]

Solução Global / Ótima

- O ponto *x* que minimiza/maximiza a função de avaliação é chamado de *solução global* ou *solução ótima*.
- Encontrar a solução global para um problema pode ser difícil, mas é mais fácil quando nos concentramos numa pequena porção do espaço de busca.

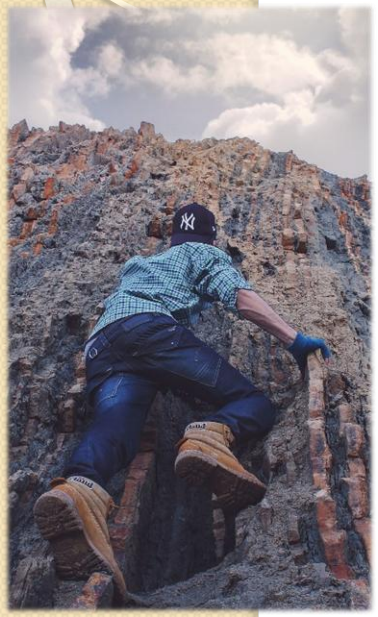


Explotação versus Exploração

- Técnicas de busca efetivas devem combinar:
  - **Explotação:** explorar as melhores soluções encontradas até o momento.
  - **Exploração:** explorar o espaço todo em busca de soluções.

https://medium.com/@selimbhy/exploration-vs-exploitation-how-to-make-the-best-decisions-based-on-probabilities-6c0147a8eca3

Explotação versus Exploração



- Alguns algoritmos exploram a melhor solução disponível, mas não exploram todo o espaço de busca:
  - Subida da Colina (*Hill Climbing*)
- Outros combinam *exploração* e *explotação*:
  - Recozimento Simulado (*Simulated Annealing*)
  - Algoritmos Genéticos (*Genetic Algorithms*)

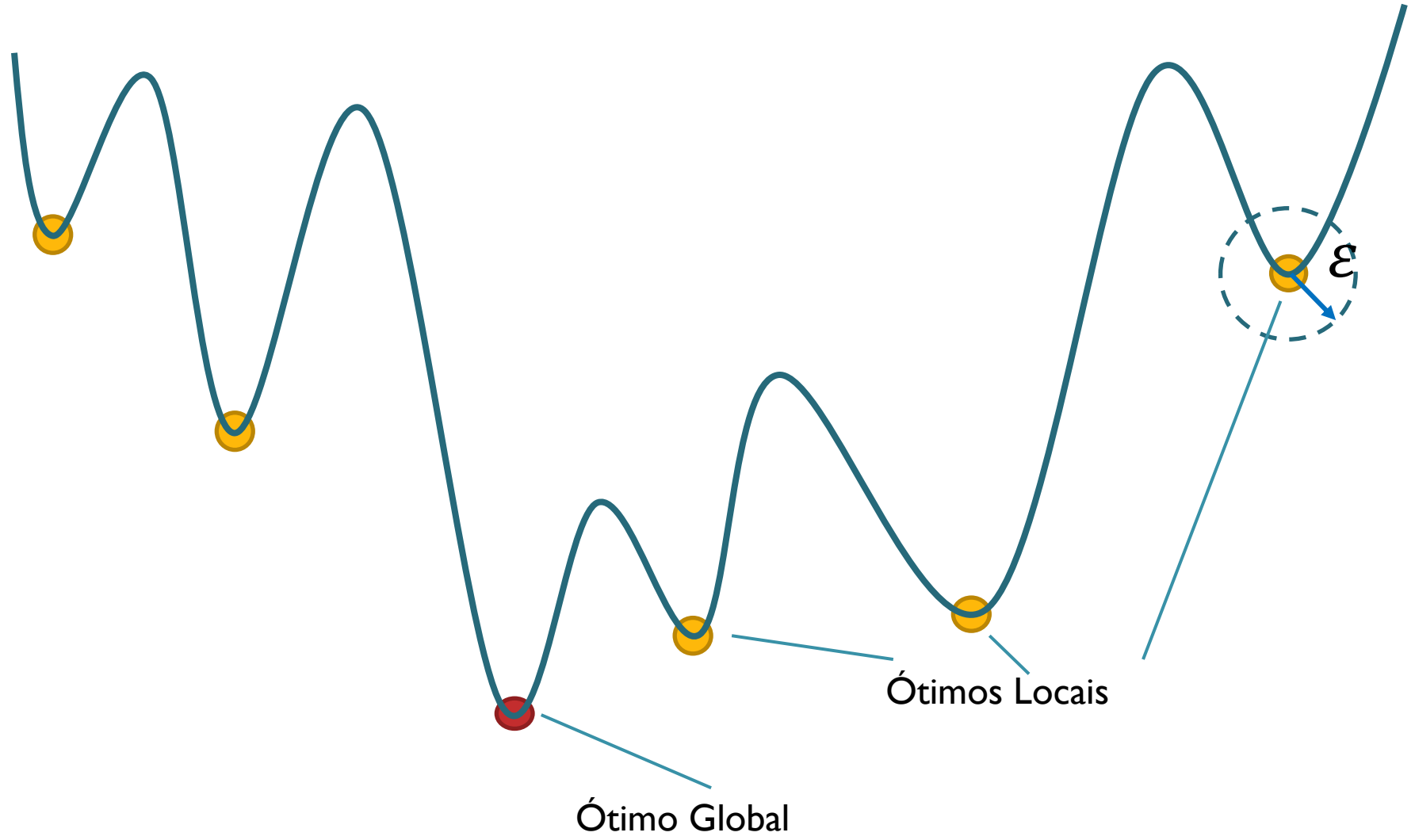
Escolha do Melhor Método

- Foi provado matematicamente que não é possível escolher um único método de busca que tenha uma média de desempenho melhor em todas as execuções para todos os problemas.
  - Teorema "*No Free Lunch*"
- Porém é possível estimar e comparar o desempenho de diferentes algoritmos e procurar por técnicas que forneçam o melhor desempenho, na média, em domínios específicos.

Ótimo Local

- Solução potencial:
  - Menor ponto dentro de uma vizinhança *N*.
    - $x \in F$ é ótimo local se $eval(x) \leq eval(y), \forall y \in N(x)$, onde, $N(x) = \{y \in F : dist(x, y) \leq \varepsilon\}$
    - $dist$ é uma função que determina a distância entre x e y
    - ε é uma constante positiva.

Ótimos Locais e Ótimo Global



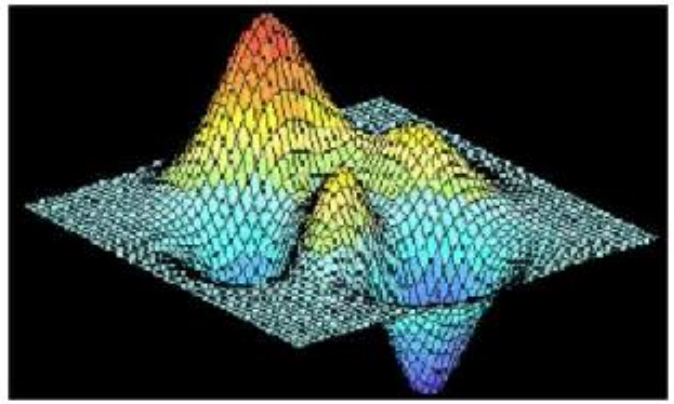
Cenário de Aptidão

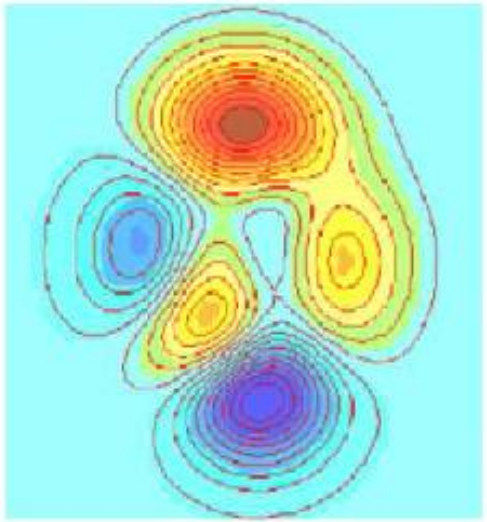
- A função de avaliação define uma superfície de resposta, chamada cenário de aptidão (*fitness landscape*).
  - Parecida com a topografia de montanhas e vales.
  - O problema de encontrar a (melhor) solução é, portanto, buscar um pico (maximização) ou vale (minimização) no cenário de aptidão.

Killarney National Park - Killarney, Kerry, Irlanda – por Fabricio Breve

Cenários de Aptidão

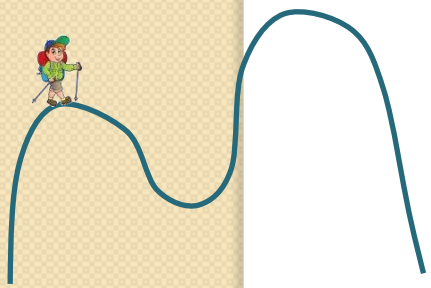
Gramado, Rio Grande do Sul – por Fabricio Breve





Amostragem de Novos Pontos

- Amostrar novos pontos no cenário é basicamente feito nas imediações do ponto atual.
  - É possível apenas tomar decisões locais sobre onde procurar a seguir.
- Se a busca for sempre feita montanha acima, eventualmente encontraremos um pico, mas este pode não ser o pico mais alto do cenário (ótimo global).
  - A busca as vezes vai morro abaixo na tentativa de encontrar um ponto que eventualmente levará a um ótimo global.



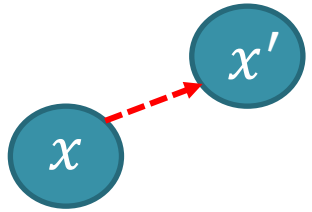
SUBIDA DA COLINA E RECOZIMENTO SIMULADO

Subida da Colina e Recozimento Simulado

- Duas técnicas tradicionais:
  - Subida da Colina (*Hill Climbing*).
  - Recozimento Simulado (*Simulated Annealing*).
    - Subida da Colina probabilístico.
    - Caso especial de um algoritmo evolutivo.
- Serão estudadas como preparação para os algoritmos evolutivos.
  - Melhor compreensão.
  - Comparação de desempenho.

Subida da Colina

- Método de busca local que usa um *aperfeiçoamento iterativo*.
- Aplicado a um único ponto no espaço de busca.
- A cada iteração um novo ponto x' é selecionado, realizando-se uma pequena perturbação no ponto atual x
  - O novo ponto é selecionado na vizinhança de x



Subida da Colina

$$\mathbf{x}' \in N(\mathbf{x})$$

$$\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}$$

- Se o novo ponto obtiver um valor melhor na função de avaliação, então ele passará a ser o ponto atual.
- Caso contrário ele será descartado.

Subida da Colina

- Critérios de parada:
  - Nenhuma melhoria pode ser obtida.
  - Um número fixo de iterações foi realizado.
  - Um ponto alvo foi atingido.

- Algoritmo: (próximo slide)
  - Seja $\mathbf{x}$ o ponto atual, g é o valor alvo para a função de avaliação (assumindo que é conhecido), e max_it o número máximo de iterações permitidas.

Algoritmo de Subida da Colina

```
procedimento [x] = hill-climbing(max_it, g)
  inicializar x
  avaliar(x)
  t ← 1
  enquanto t ≤ max_it & avaliar(x) != g faça
    x' ← perturbar(x)
    avaliar(x')
    se avaliar(x') é melhor que avaliar(x)
      então x ← x'
    fim-se
    t ← t + 1
  fim-enquanto
fim-procedimento
```

Fraquezas do Subida da Colina

- Normalmente termina em soluções ótimas locais.
- Não há informação sobre a distância da solução encontrada para a ótima global.
- O ótimo encontrado depende da configuração inicial.
- Não é possível calcular um limite para o tempo computacional do algoritmo.

Subida da Colina: múltiplas inicializações

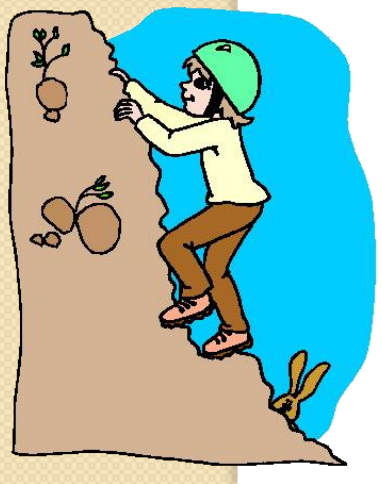
- Como o algoritmo só fornece soluções ótimas locais, é razoável iniciá-lo de uma grande variedade de pontos.
  - A esperança é que pelo menos um deles leve ao ótimo global.

Subida da Colina: múltiplas inicializações

- Pontos iniciais podem ser escolhidos:
  - aleatoriamente;
  - usando uma grade ou padrão regular;
  - outros tipos de informação.
- Neste caso o algoritmo tradicional será inicializado múltiplas vezes e teremos uma "memória da melhor solução".
  - Subida da Colina Iterativo (*Iterated Hill Climbing*).

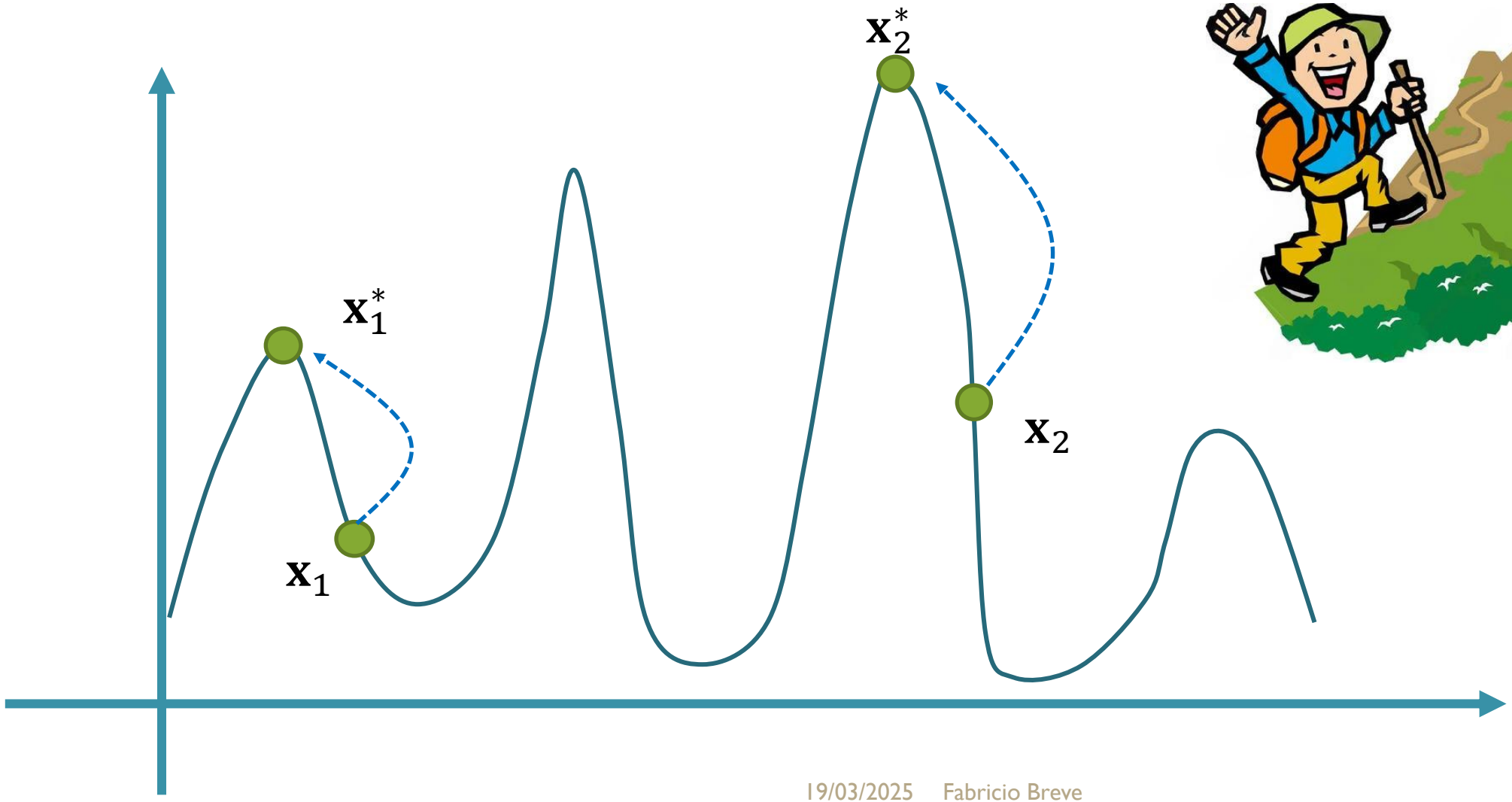
Algoritmo de Subida da Colina Iterativo

```
procedimento [melhor] = IHC(n_start, max_it, g)
  inicializar melhor
  t1 ← 1
  enquanto t1 ≤ n_start & avaliar(melhor) != g faça,
    inicializar x
    x ← hill-climbing(max_it, g) // Hill Climbing Original
    t1 ← t1 + 1
    se avaliar(x) é melhor que avaliar(melhor),
       então melhor ← x
    fim-se
  fim-enquanto
fim-procedimento
```



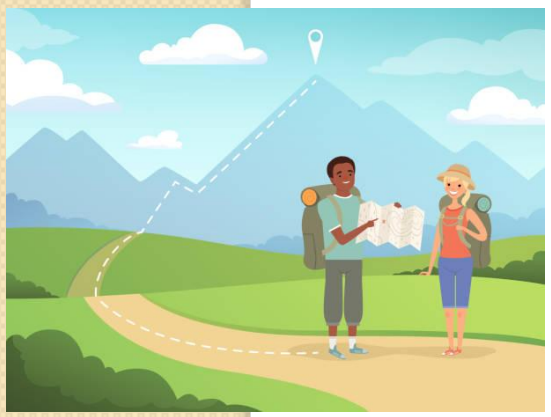
- n_start é a quantidade de pontos iniciais que será utilizada
- Se for usada uma grade ou padrão regular, é interessante modificar o `hill-climbing` para aceitar um valor inicial x_0

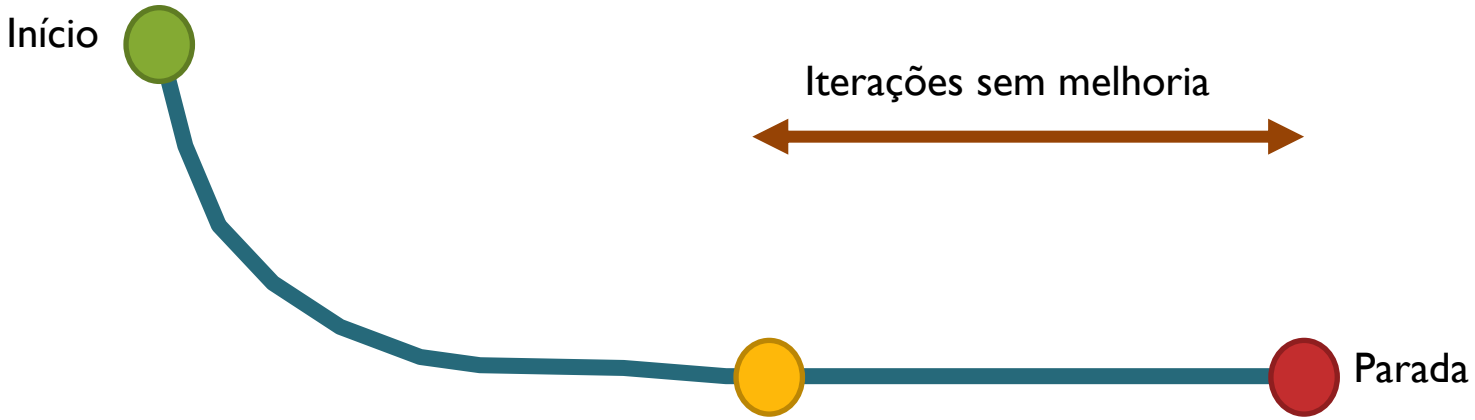
Algoritmo de Subida da Colina Iterativo



Critério de Parada Alternativo

- Se a melhor solução não mudar significativamente após um certo número de iterações, pode-se dizer que o algoritmo convergiu para um ótimo local e o processo pode ser parado.





Subida da Colina em Problemas Reais

- Em problemas reais, o número de soluções ótimas locais pode ser bastante alto.
  - Estratégia de tentativa e erro não é plausível.
- Porém, a boa capacidade de *explotação* do Subida da Colina pode ser combinada com muitas outras técnicas que sejam capazes de executar uma melhor *exploração* do espaço de busca.

Subida da Colina Probabilístico

- Modificação do Subida da Colina Clássico.
  - A probabilidade de que **x'** seja selecionado depende da diferença entre os valores retornados pela função de avaliação para **x** e **x'**
  - Algoritmo: (próximo slide)
    - T é um parâmetro de controle do decaimento da função exponencial.


Mini Dados, por Fabricio Breve

Algoritmo de Subida da Colina Probabilístico

```
procedimento [x] = stochastic-hill-climbing(max_it, g)
  inicializar x
  avaliar(x)
  t ← 1
  enquanto t ≤ max_it & avaliar(x) != g faça
    x' ← perturbar(x)
    avaliar(x')
    se aleatorio[0,1) < (1/(1+exp[(avaliar(x)-avaliar(x'))/T]))
      então x ← x'
    fim-se
    t ← t + 1
  fim-enquanto
fim-procedimento
```

Observação: válido para o caso onde o objetivo é maximizar avaliar(x); caso o objetivo seja minimizar avaliar(x), temos de inverter a posição dos dois termos da subtração.

Subida da Colina Probabilístico

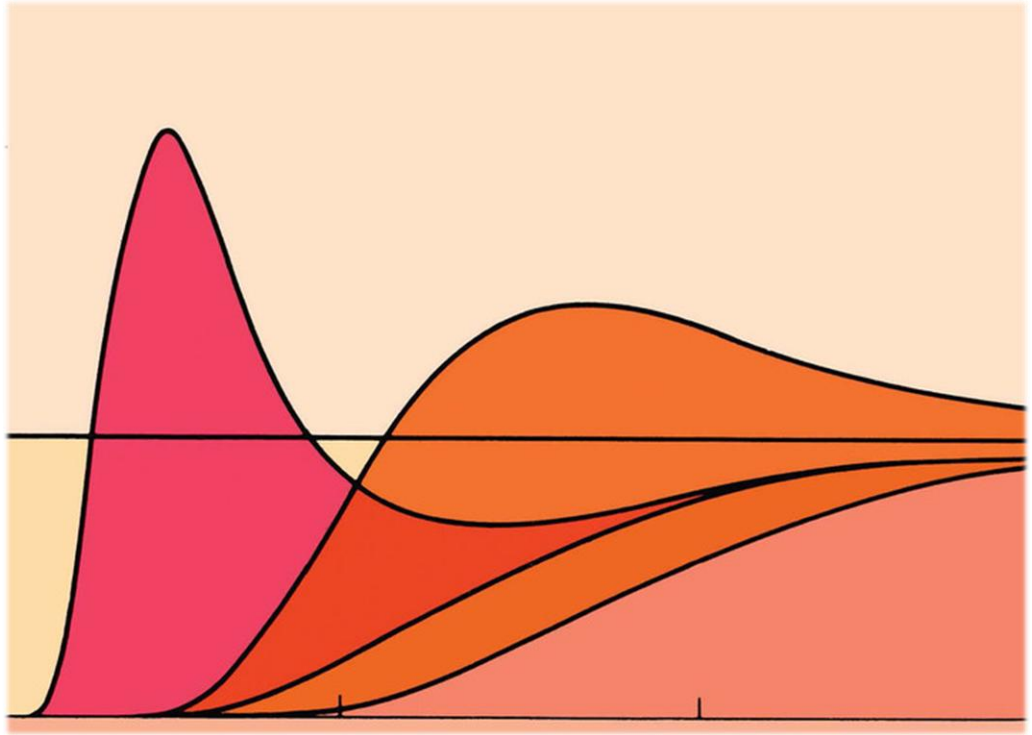
- A probabilidade *P* de que o novo ponto **x'** seja aceito é dada por $P = 1/(1 + \exp[(\text{avaliar}(x) - \text{avaliar}(x'))/T])$
- Quanto maior o valor de T, menor a importância da diferença da avaliação de **x** e **x'**
- Com um alto valor em T, a busca fica similar a uma busca aleatória.

Recozimento Simulado

- Inspirado no processo de recozimento de sistemas físicos.
  - Recozimento: sujeitar um material (ex.: metal) a um processo de aquecimento e resfriamento lento para aumentar sua ductilidade e reduzir sua dureza, tornando-o mais trabalhável.
  - Altera as propriedades físicas e às vezes químicas do material.
- Inspirado no algoritmo de Metropolis.
  - Meio de encontrar uma configuração de equilíbrio de uma coleção de átomos em uma certa temperatura.

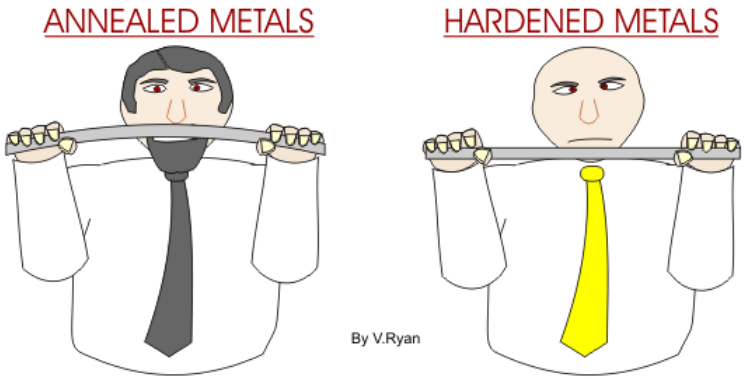
Recozimento Simulado

- Idéias retiradas da *termodinâmica estatística.*
  - Campo da Física que faz predições teóricas sobre o comportamento de sistemas macroscópicos (líquido e sólido) com base nas leis que governam os átomos que os compõem.



Capa do livro An Introduction to Statistical Thermodynamics, por Terrell L. Hill

Recozimento

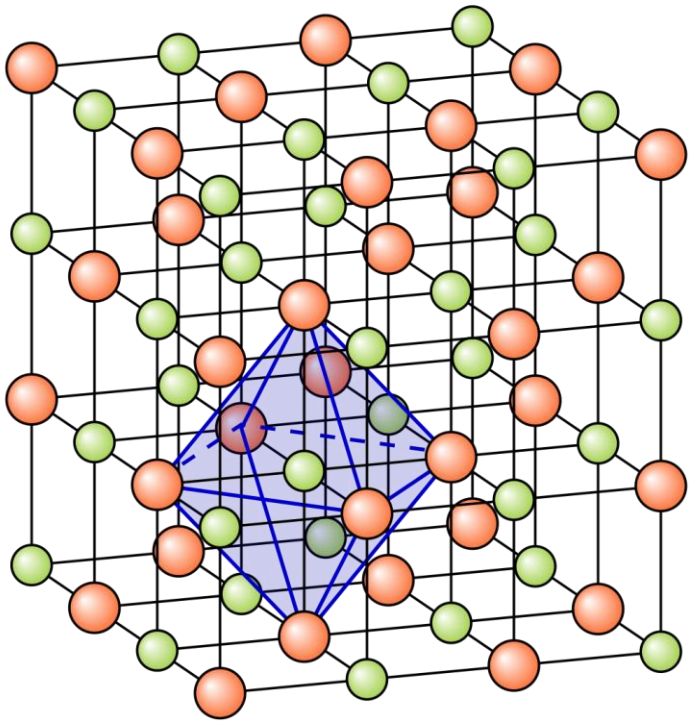


Um policristal de quartzo, uma das substâncias cristalinas mais comuns na Terra.

- Objetivo:
  - Levar o material ao seu *ground state*, um estado de mínima energia em que uma *estrutura cristalina* será obtida.

O Gálio é um metal que forma grandes cristais.

Aglomerado de cristais de quartzo do Tibete

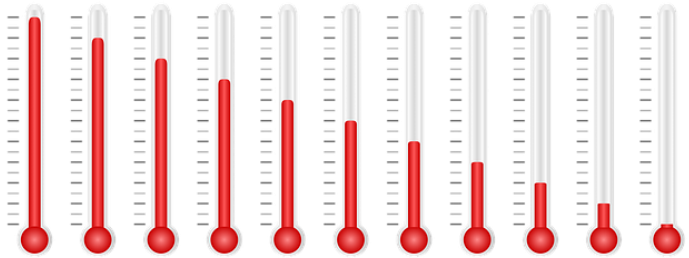


Célula unitária da estrutura de um cristal de sal (NaCl). Note-se a ordenação dos átomos.

Imagens: Wikipedia - http://pt.wikipedia.org/wiki/Cristal

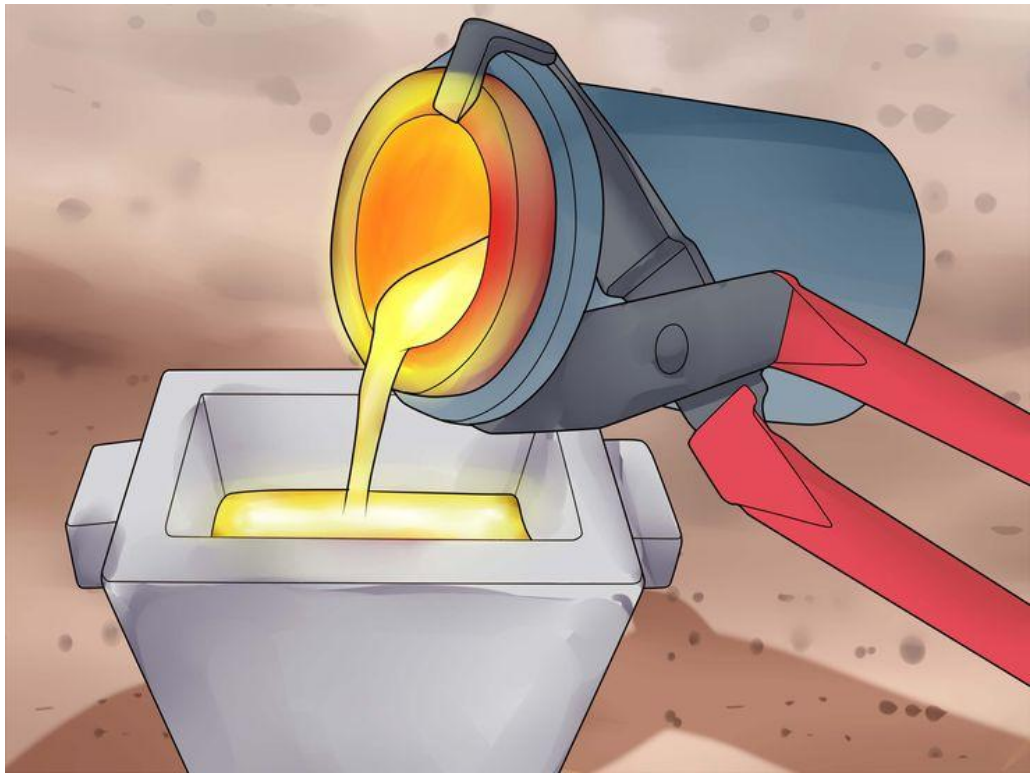
Recozimento

- Temperatura do material é elevada para que derreta e seus átomos possam se mover livremente.
- Temperatura do sistema derretido é lentamente diminuída para que a cada nova temperatura os átomos possam se movimentar o suficiente para adotar uma orientação mais estável.
- Se a temperatura for diminuída suficientemente devagar, os átomos irão repousar na orientação mais estável, produzindo um cristal.

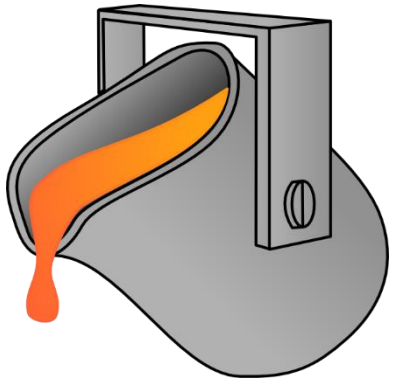


Recozimento Simulado

- Na simulação:
  - *Estado do sistema* físico equivale a uma *possível solução do problema*.
  - A *energia do sistema* é medida pela *função de avaliação*.
    - No sistema físico o objetivo é encontrar uma configuração de mínima energia.
  - O *estado de equilíbrio* equivale a um *ótimo local*.
  - O *estado de mínima energia* (*ground state*) é o *ótimo global*.
  - *Temperatura* é um *parâmetro de controle*.
  - *Recozimento* é a busca reduzindo a temperatura.



Recozimento Simulado



- Algoritmo:
  - Seja $\mathbf{x}$ a atual configuração do sistema, $\mathbf{x}'$ a configuração de $\mathbf{x}$ após um pequeno deslocamento aleatório, e T a temperatura do sistema.
  - A função $g(T, t)$ é responsável por reduzir o valor da temperatura.
    - Geralmente utiliza-se um decremento geométrico.
      - Ex.: $T \leftarrow \beta . T$ onde $\beta < 1$

Recozimento Simulado

```
procedimento [x] = simulated_annealing(g)
  inicializar T
  inicializar x
  avaliar(x)
  t ← 1
  enquanto não atender critério de parada faça
    x' ← perturbar(x)
    avaliar(x')
    se avaliar(x') é melhor que avaliar(x),
      então x ← x'
    senão se aleatorio[0,1) < exp[(avaliar(x')-avaliar(x))/T],
      então x ← x'
    fim-se
    T ← g(T,t)
    t ← t + 1
  fim-enquanto
fim-procedimento
```

Observação: válido para o caso onde o objetivo é maximizar avaliar(x); caso o objetivo seja minimizar avaliar(x), temos de inverter a posição dos dois termos da subtração.

Recozimento Simulado

```
Passo 1:  inicializar T
          inicializar x
Passo 2:  x' ← perturbar(x)
          se avaliar(x') é melhor que avaliar(x),
              então x ← x'
          senão aleatorio[0,1) < exp[(avaliar(x')-avaliar(x))/T],
              então x ← x'
          fim-se
          repita este passo k vezes
Passo 3:  T ← β.T
          se T ≥ T_min
              então vá para Passo 2
          senão vá para Passo 1
```

Implementação típica do recozimento simulado

Observação: válido para o caso onde o objetivo é maximizar avaliar(x); caso o objetivo seja minimizar avaliar(x), temos de inverter a posição dos dois termos da subtração.

Exemplo de Aplicação

- Considere a função unidimensional $g(x)$ abaixo:

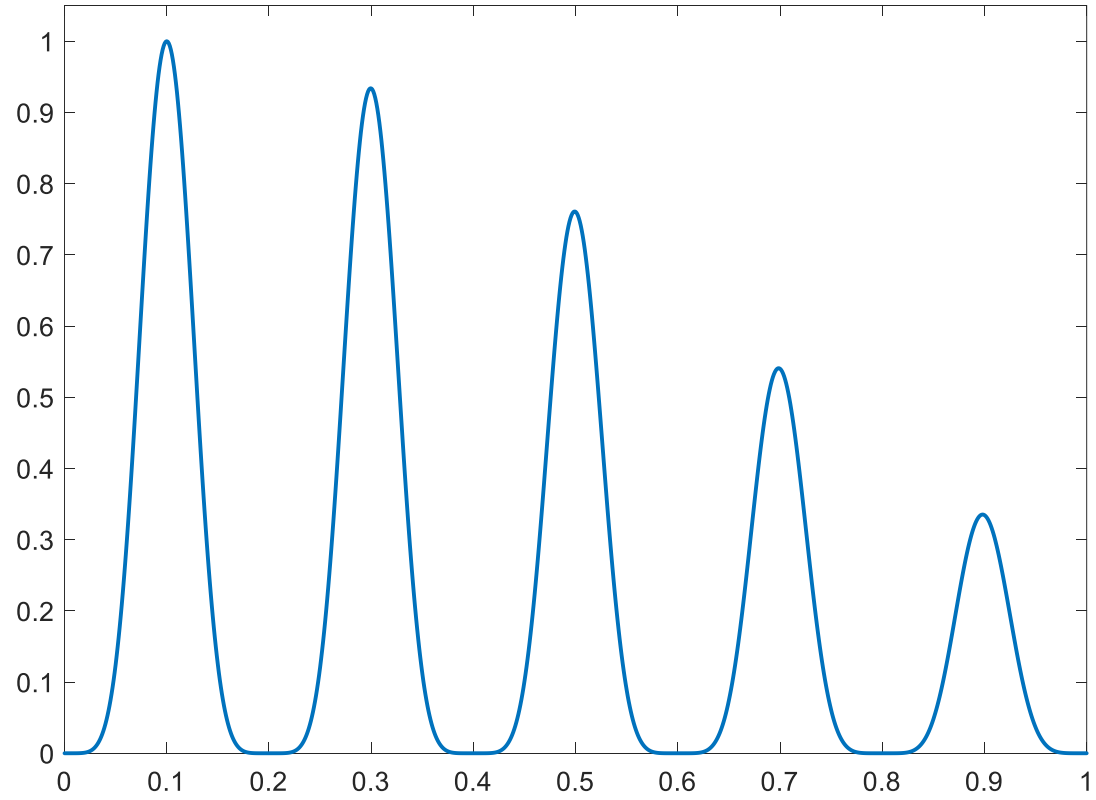
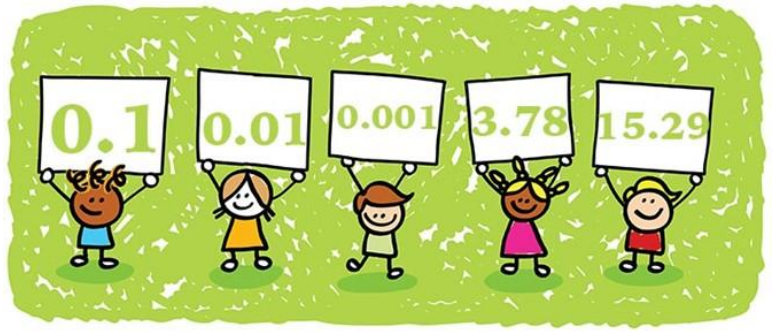


Gráfico da função $g(x) = 2^{-2\left(\frac{x-0,1}{0,9}\right)^2}(\sin(5\pi x))^6$ a ser maximizada.

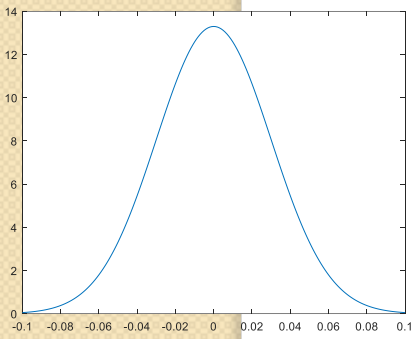
Exemplo de Aplicação

- A variável x é definida no intervalo $[0,1]$
- Assuma que o máximo global é desconhecido.
- Como foi discutido precisamos de:
  - Representação
  - Objetivo
  - Avaliação

Exemplo de Aplicação



- Representação:
  - Usaremos o valor real da variável x para representar as soluções candidatas $x \in [0, 1]$
  - Para perturbar o ponto atual, usaremos um ruído Gaussiano de média zero e pequena variância $G(0, \sigma)$
    - $x' = x + G(0, \sigma)$
      - $\sigma =$ constante positiva pequena
    - Também poderíamos usar uma distribuição uniforme em vez de Gaussiana.
  - Descartaremos resultados fora do intervalo $[0,1]$



Dica: Ruído Gaussiano pode ser gerado com o método de Box Müller. Matlab, Java, C++ 11, e outras linguagens tem funções prontas para isso.

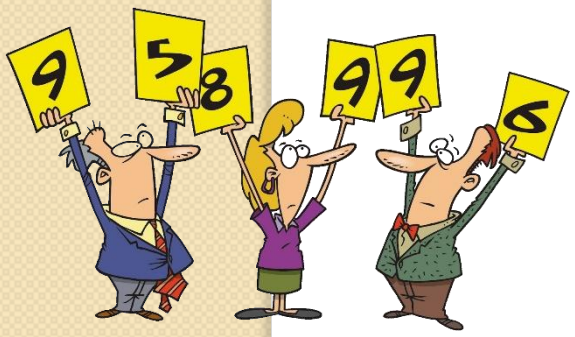
Exemplo de Aplicação

- Objetivo:
  - Encontrar o valor máximo da função $g(x)$, isto é:

$$\max g(x)$$

- Avaliação:
  - Para avaliar as soluções candidatas usaremos a função $g(x)$ para os valores de x:

$$g(x) = 2^{-2\left(\frac{x-0.1}{0.9}\right)^2} (\sin(5\pi x))^6$$



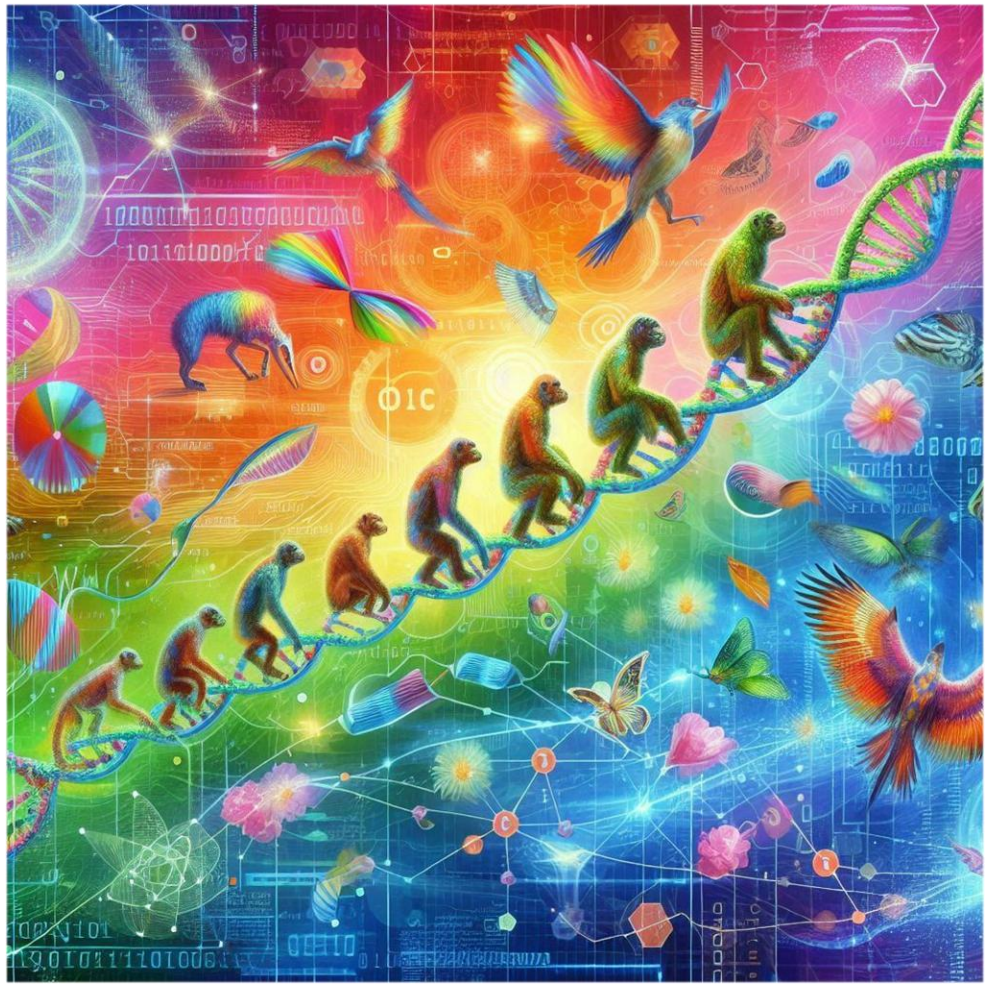
Exercícios

1. Implemente os vários tipos de algoritmos de Subida da Colina e Recozimento Simulado para resolver o problema proposto no Exemplo de Aplicação. Use um esquema de representação para as soluções candidatas (variável x). Compare o desempenho dos algoritmos e tire suas conclusões.
2. Para o Subida da Colina simples, utilize diferentes configurações iniciais como tentativas de encontrar o ótimo global. O algoritmo teve sucesso?
3. Discuta a sensibilidade de todos os algoritmos com relação aos seus parâmetros de entrada.

Dica: Programamando a Função de Aptidão

- $g(x) = 2^{-2\left(\frac{x-0,1}{0,9}\right)^2} (\sin(5\pi x))^6$
- Em MATLAB:
  - gx = 2.^(-2.*((((x-0.1)./0.9)).^2)).*((sin(5.*pi.*x)).^6);
- Em Python:
  - gx = 2.**(-2.*((((x-0.1) / 0.9))**2))*((np.sin(5.*np.pi*x))**6

Próxima Aula



- Biologia Evolutiva
  - Inspiração para os modelos de Computação Evolutiva
    - Evolução
      - Lamark, Darwin, ...
    - Genética
      - Experimentos de Mendel

https://www.umsabadoqualquer.com/darwin-e-mendel/

Bibliografia

- CASTRO, Leandro Nunes. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, And Applications*. CRC Press, 2006.
- CARVALHO, André Ponce de Leon F. de. *Notas de Aula*, 2007.
- BROWNLEE, Jason. *Clever Algorithms: Nature-Inspired Programming Recipes*. Jason Brownlee, 2011.
- EIBEN, A. E.; SMITH, James E. *Introduction to Evolutionary Computing*, 2nd Edition. Springer, 2015.
- SIMON, Dan. *Evolutionary Optimization Algorithms*. Wiley, 2013.
- MITCHELL, Melaine. *An Introduction to Genetic Algorithms*. MIT Press, 1998.