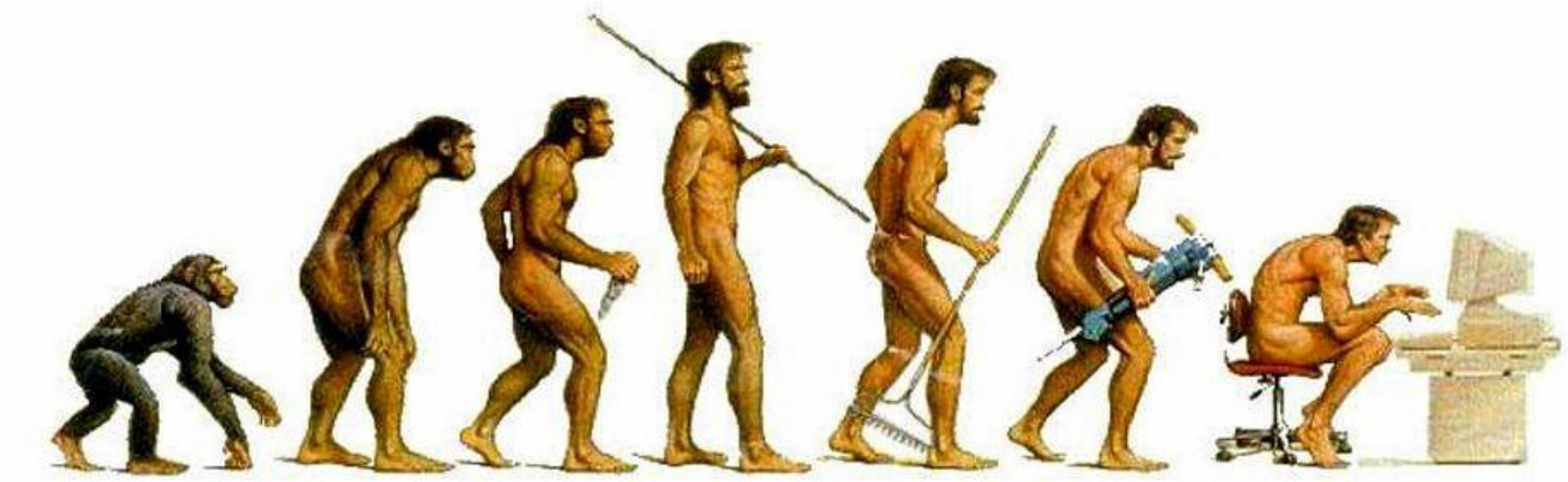
Computação Evolutiva – Parte I



Fabricio Breve - fbreve@gmail.com

Introdução

- **Computação Evolutiva**
  - Campo de pesquisa que utiliza idéias da biologia evolutiva para desenvolver técnicas para resolução de sistemas complexos
  - A maioria 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), resultando 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 extraído ou inferido
- Exemplos:
  - Função numérica a ser maximizada
  - Problema de alocação de um número de turmas 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 seqüência 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*) serão 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

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

Resolução de Problemas como uma Tarefa de Busca

- Escolha de representação
  - Codificação das soluções candidatas (indivíduos) para manipulação
  - Depende do problema, a representação é de vital importância e 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

Resolução de Problemas como uma Tarefa de Busca

- Especificação do Objetivo
  - Descrição dos propósitos a serem preenchidos
  - Exemplo
    - Se o objetivo é minimizar a função

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

    - Então o objetivo pode ser definido como

$$\min f(x)$$

Resolução de Problemas como uma Tarefa de Busca

- 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 como um conjunto de soluções candidatas

Definindo um Problema de Busca

- 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 forneçam 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 $eval(x^*) \leq eval(x), \forall x \in F$.

Definindo um Problema de Busca

- Neste caso, a função de avaliação que retorna pequenos valores para *x* é considerada melhor
  - Problema e *minimização*
- Resolver um problema de *maximização* equivale a minimizar o negativo

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

Definindo um Problema de Busca

- O processo de busca não sabe qual 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 do problema errado!

Definindo um Problema de Busca

- O ponto *x* que satisfaz tal condição é chamado *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

Definindo um Problema de Busca

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

Definindo um Problema de Busca

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

Definindo um Problema de Busca

- Foi provado matematicamente que não existe um único método de busca que pode ter um desempenho melhor em todas as execuções para todos os problemas
- Porém é possível estimar e comparar o desempenho de diferentes algoritmos e procurar por técnicas que forneçam a melhor performance.

Definindo um Problema de Busca

- Ótimo local
  - Solução potencial
    - Menor ponto dentro de uma vizinhança *N*

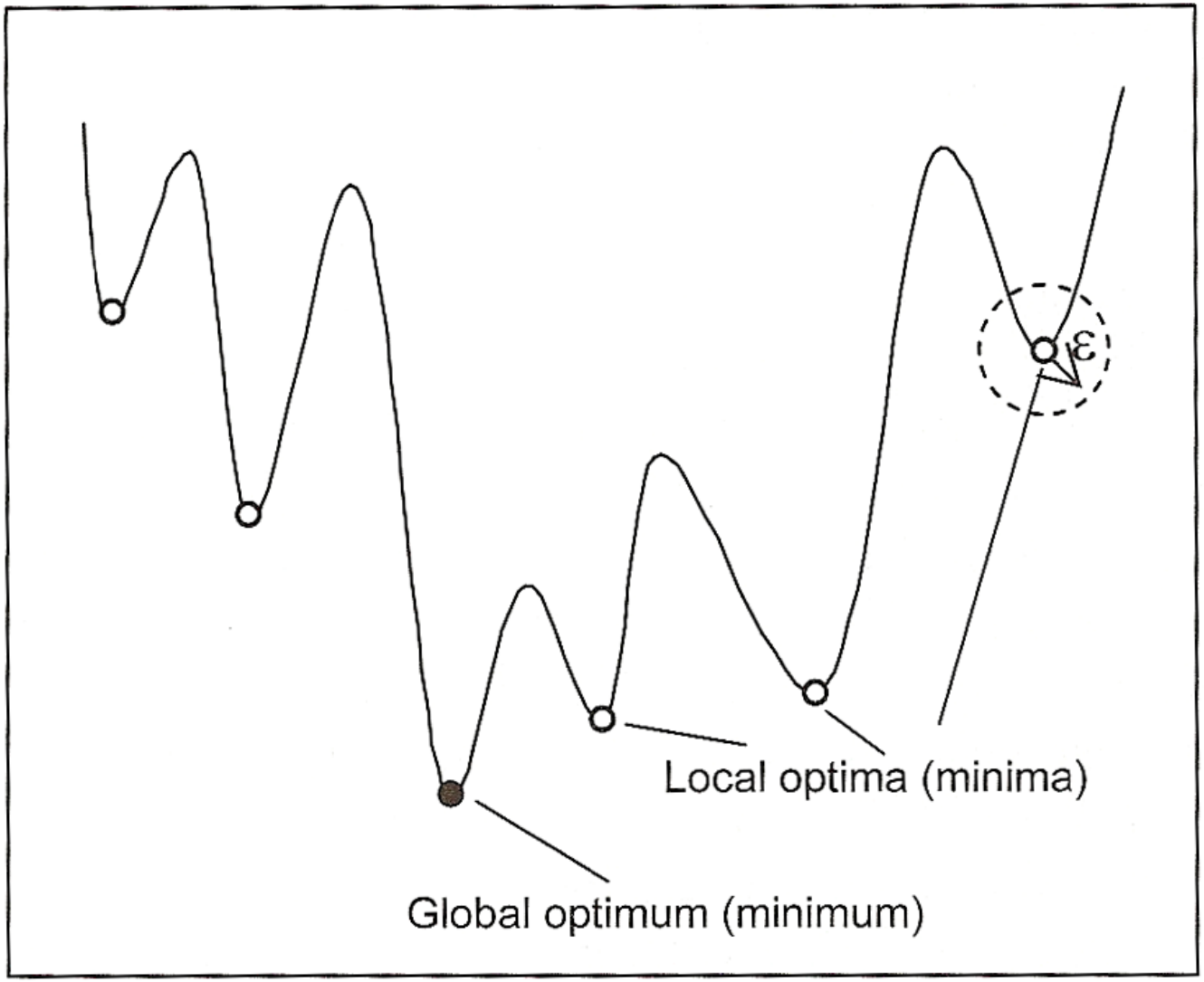
$x \in F$ é ótimo local se $eval(x) \leq eval(y), \ \forall y \in N(x),$

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

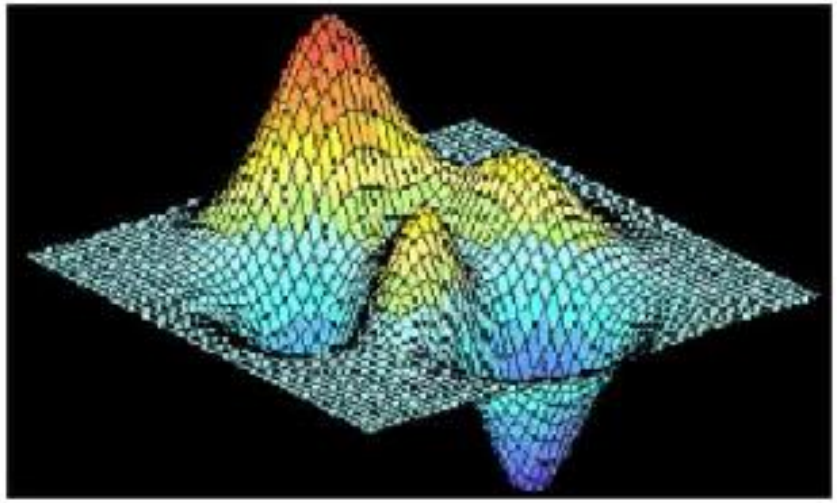
Definindo um Problema de Busca

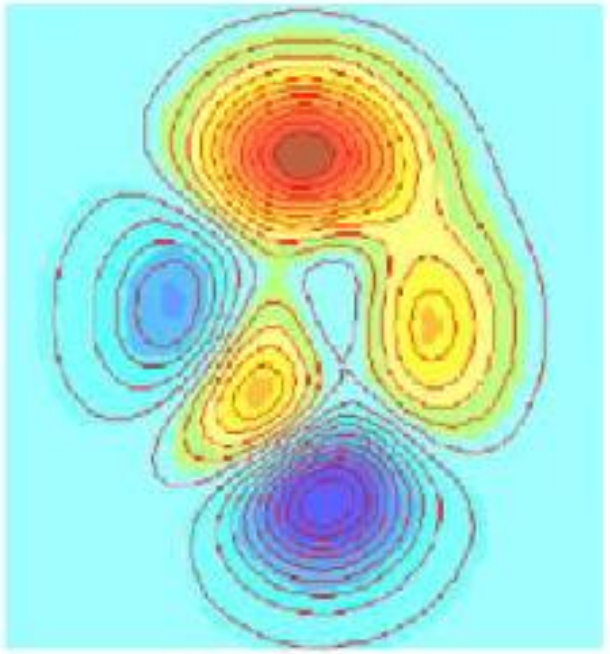


Definindo um Problema de Busca

- A função de avaliação define uma superfície de resposta, chamada *cenário de aptidão (fintess landscape)*
  - Topografia de montanhas e vales
  - O problema de encontrar a (melhor) solução é portanto buscar um pico (maximização) ou vale (minimização)

Definindo um Problema de Busca





Definindo um Problema de Busca

- 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 eventualmente 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 através de 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 obter um valor melhor na função de avaliação, então ele passará a ser o ponto atual.
- Caso contrário ele é 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 **x** o ponto atual, **g** o ponto alvo (assumindo que é conhecido), e *max_it* o número máximo de iterações permitidas.

Algoritmo de Subida da Colina

```
procedure [x] = hill-climbing(max_it,g)
  initialize x
  eval(x)
  t ← 1
  while t < max_it & x != g & no_improvement do,
      x' ← perturb(x)
      eval(x')
      if eval(x') is better than eval(x),
            then x ← x'
      end if
      t ← t + 1
  end while
end procedure
```

Algorithm 3.1: A standard (simple) hill-climbing procedure.

Algoritmo de Subida da Colina

- Tem algumas fraquezas
  - Terminam 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

Algoritmo de Subida da Colina

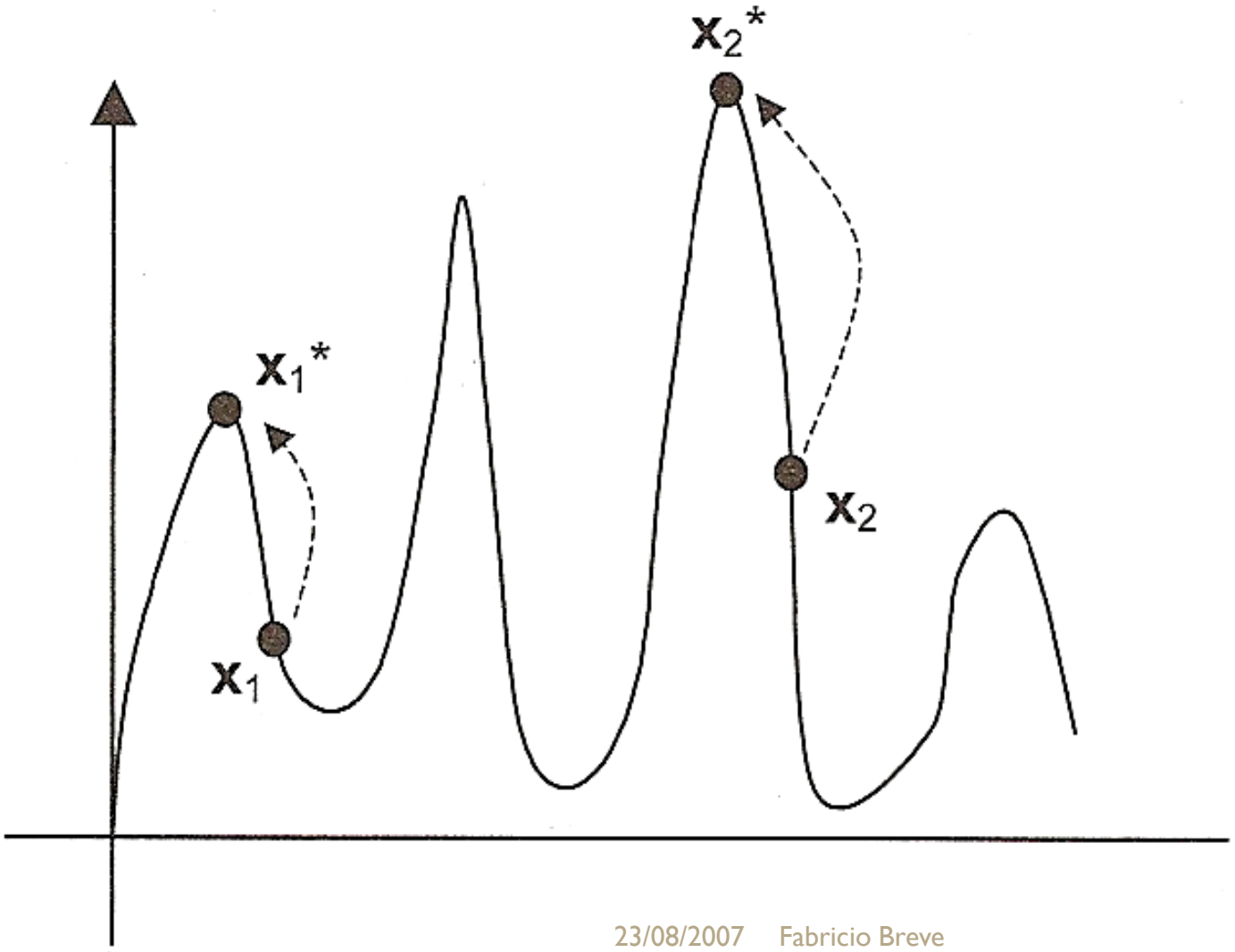
- Como o algoritmo só fornece soluções ótimas locais, é razoável iniciá-lo de uma grande variedade de pontos
  - A esperança é que pelo menos um deles leve ao ótimo global
  - Os pontos iniciais podem ser escolhidos aleatoriamente, usando uma grade ou padrão regular ou outros tipos de informação
  - 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

```
procedure [best] = IHC(n_start,max_it,g)
  initialize best
  t1 ← 1
  while t1 < n_start & best != g do,
    initialize x
    eval(x)
    x ← hill-climbing(max_it,g) //Algorithm 1
    t1 ← t1 + 1
    if x is better than best,
          then best ← x
    end if
  end while
end procedure
```

Algorithm 3.2: An iterated hill-climbing procedure.

Algoritmo de Subida da Colina



Algoritmo de Subida da Colina

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

Algoritmo de Subida da Colina

- Na prática o número de soluções ótimas é bastante alto
- A capacidade de *exploitation* do Subida da Colina pode ser combinado com outras em combinação com outras técnicas capazes de executar uma melhor exploração do espaço de busca

Algoritmo de Subida da Colina

- Modificação: Subida da Colina Probabilístico
  - 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

Algoritmo de Subida da Colina

```
procedure [x] = stochastic hill-climbing(max_it,g)
  initialize x
  eval(x)
  t ← 1
  while t < max_it & x != g do,
     x' ← perturb(x)
    eval(x')
    if random[0,1) < (1/(1+exp[(eval(x)-eval(x'))/T])),
          then x ← x'
    end if
    t ← t + 1
  end while
end procedure
```

Algorithm 3.3: A stochastic hill-climbing procedure.

Algoritmo de Subida da Colina

- A probabilidade *P* de que o novo ponto **x'** seja aceito é dada por

$$P = 1/(1+\exp[(eval(\mathbf{x}) - eval(\mathbf{x}'))/T])$$

- 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.: vidro, metal) a um processo de aquecimento e resfriamento lento para torná-lo mais forte e reduzir sua fragilidade
- Inspirado no algoritmo de Metropolis
  - Meio de encontrar uma configuração de equilíbrio de uma coleção de átomos em uma certa temperatura

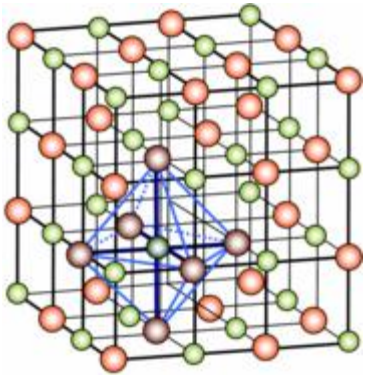
Recozimento Simulado

- 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 compõe

Recozimento Simulado

- 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.



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

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

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

Recozimento Simulado

1. Temperatura do material é elevada para que derreta e seus átomos possam se mover livremente
2. Temperatura do sistema derretido é lentamente diminuída para que a cada nova temperatura os átomos possam o suficiente para adotar uma orientação mais estável
3. 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 *T*

Recozimento Simulado

- Algoritmo
  - Seja **x** a atual configuração do sistema, **x'** a configuração de **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

```
procedure [x] = simulated_annealing(g)
  initialize T
  initialize x
  eval(x)
  t ← 1
  while not_stopping_criterion do,
      x' ← perturb(x)
      eval(x')
      if eval(x') is less than eval(x),
            then x ← x'
      else if random[0,1) < exp[(eval(x)-eval(x'))/T],
            then x ← x'
      end if
      T ← g(T,t)
      t ← t + 1
    end while
end procedure
```

Algorithm 3.4: The simulated annealing procedure.

Recozimento Simulado

```
Step 1:     initialize T
            initialize x
Step 2:     x′ ← perturb(x)
            if eval(x′) is less than eval(x),
                  then x ← x′
            else if random[0,1) < exp[(eval(x)-eval(x′))/T]
                  then x ← x′
            end if
            repeat this step k times
Step 3:     T ← β.T
            if T ≥ T_min
                  then goto Step 2
            else goto Step 1
```

Algorithm 3.5: Typical implementation of the simulated annealing procedure.

Exemplo de Aplicação

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

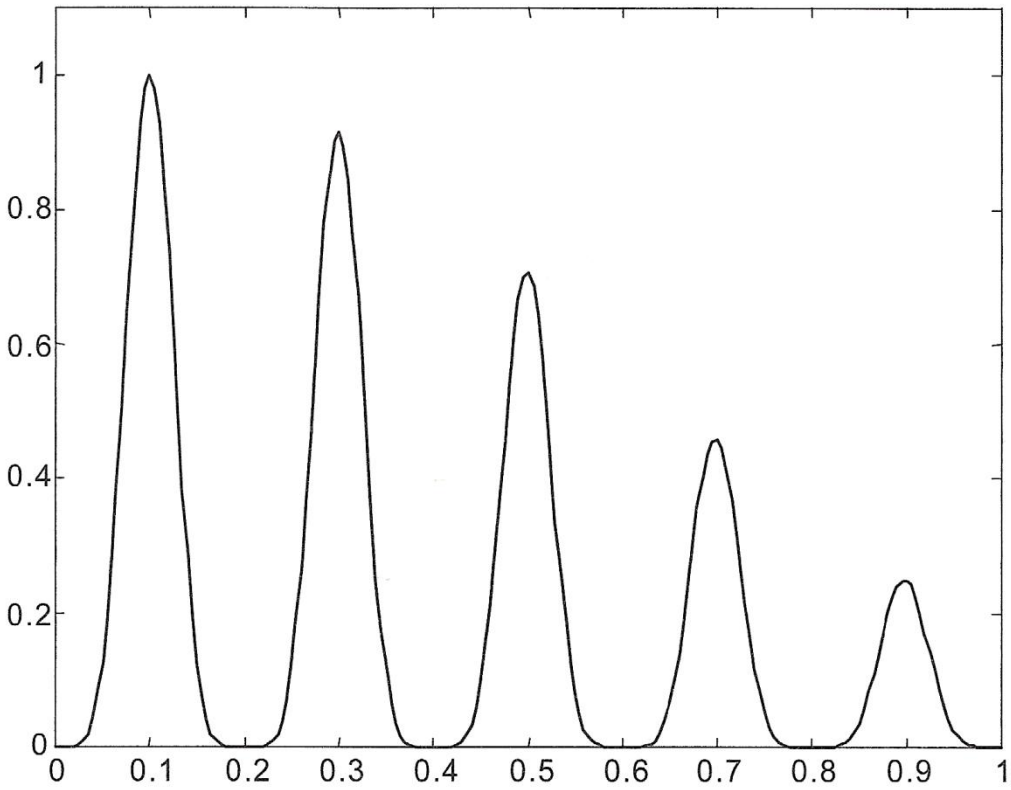


Figure 3.3: Graph of the function $g(x) = 2^{-2((x-0.1)/0.9)^2}\left(\sin(5\pi x)\right)^6$ to be maximized.

Exemplo de Aplicação

- A variável *x* é definida no intervalo [0,1] e 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 $G(0,\sigma)$
    - Descartaremos resultados fora do intervalo [0,1]
    - Poderíamos usar uma distribuição uniforme

$$x' = x + G(0,\sigma)$$

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

Exercícios

1. Implemente os vários tipos de algoritmos de Subida da Colina 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.

Referências Bibliográficas

- 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.
- HAYKIN, Simon. *Redes Neurais: Princípio e Prática*. Bookman, 2001.
- KOVACS, Zsolt L. *Redes Neurais Artificiais: Fundamentos e Aplicações*. Livraria da Física, 2006.