Estudo sobre as Metaheurísticas

Leandro Ferro Luzia
Mauricio Chui Rodrigues

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. Método do Gradiente
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. *Harmony Search*
7. Conclusão

Estudo sobre as Metaheurísticas

1. **Metaheurísticas e Otimização**
2. Objetivo do Estudo
3. Método do Gradiente
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. *Harmony Search*
7. Conclusão

Otimização

- Otimização é o processo de escolher o melhor elemento em um conjunto de alternativas disponíveis
- Relacionado a uma função objetivo, aplicada sobre os elementos do conjunto
- Diversos tipos de otimização
  - Programação Linear
  - Programação Inteira
  - Otimização Combinatória
  - Otimização Estocástica
    - Metaheurísticas

Metaheurísticas

- Métodos que coordenam procedimentos de busca local com estratégias de mais alto nível
- Objetivo de criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções
- Aplicadas para resolver problemas sobre os quais há pouca informação, mas que, uma vez oferecida uma solução candidata, esta pode ser testada
- Não apresentam garantias de otimalidade

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. **Objetivo do Estudo**
3. Método do Gradiente
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. *Harmony Search*
7. Conclusão

Objetivo do Estudo

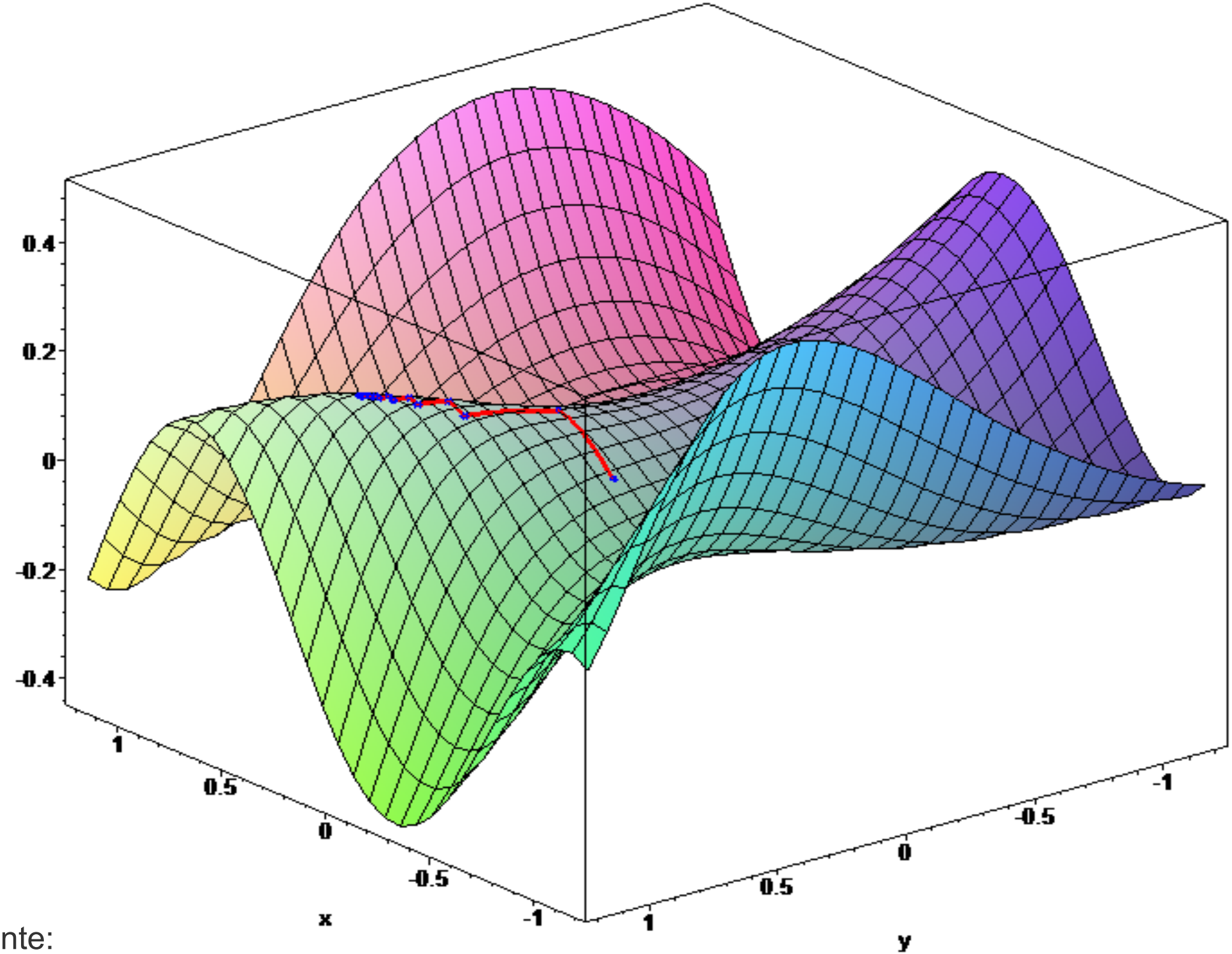
- Analisar diversas metaheurísticas
  - Definição
  - Formas de implementação
  - Vantagens e desvantagens
  - Exemplos de aplicação

- Metaheurísticas abordadas:

• *Best-first Search*	• ***Ant Colony Optimiz.***
• ***Hill Climbing***	• *Particle Swarm Optimiz.*
• *Tabu Search*	• ***Harmony Search***
• *Simulated Annealing*	• *Memetic Algorithms*
• GRASP	• *Genetic Algorithms*

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. **Método do Gradiente**
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. *Harmony Search*
7. Conclusão



Fonte:
http://en.wikipedia.org/wiki/Gradient_descent

Método do Gradiente

- Algoritmo tradicional de aproximação de funções
- Baseia-se na inclinação da função em um ponto, para definir em que sentido da função irá buscar uma melhor solução
- Tempo de convergência é grande, pois há oscilação do sinal do gradiente quando se aproxima de um máximo/mínimo
- Pode ficar preso em máximos/mínimos locais ou pontos de inflexão, nos quais o gradiente é nulo

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. Método do Gradiente
4. ***Hill Climbing***
5. *Ant Colony Optimization*
6. *Harmony Search*
7. Conclusão

Hill Climbing

- Técnica simples de busca local: não armazena o caminho percorrido até a solução atual
- Algoritmo similar ao do método do gradiente
- Não requer conhecimento sobre a derivada ou o gradiente da função
- Avalia soluções candidatas na região atual, optando pela que melhorar a avaliação da função objetivo

Hill Climbing

- Algoritmo Básico

```
1: S ← solução inicial
2: repita
3:     R ← NovaSolução(S)
4:     se (Qualidade(R) > Qualidade(S)) então
5:         S ← R
6: até que S seja ideal ou o tempo se esgote
7: devolva S
```

Hill Climbing

- Vantagens
  - Fácil implementação
  - Base para outras metaheurísticas
- Desvantagens
  - Máximos/mínimos locais com grande vizinhança
  - Não usa informações adicionais sobre o problema

Hill Climbing

- Diversas aplicações a escalonamentos
  - Mais interessante: escalonamento de recursos computacionais em sistemas distribuídos e multiprocessados

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. Método do Gradiente
4. *Hill Climbing*
5. ***Ant Colony Optimization***
6. *Harmony Search*
7. Conclusão


© Bruce G. Marcot

Ant Colony Optimization

- Abordagem baseada no uso que as formigas reais fazem do feromônio para se comunicar
- Dois conceitos importantes:
  - Formigas - agentes que constroem soluções iterativamente (atuam como uma memória de curta duração do algoritmo)
  - Feromônio - informação numérica distribuída que ajuda a guiar as formigas na construção de soluções (atua como uma memória de longa duração do algoritmo)

Ant Colony Optimization

- Algoritmo básico

```
01: C ← {C1, ..., Cn} componentes
02: t ← número de trilhas para construir de uma só vez
03: f ← <f1,...,fn> feromônios dos componentes
04: Melhor ← nulo
05: repita
06:     P ← t trilhas, construídas por seleção iterativa de componentes baseada nos
        feromônios e nas informações heurísticas
07:    para cada Pi em P faça
08:        se Melhor = nulo ou Qualidade(Pi) > Qualidade (Melhor) então
09:            Melhor ← Pi
10:     atualize f para os componentes baseado na qualidade para cada Pi em P em que
        eles participaram
11:     EvaporarFeromônio()
12:     AçõesDeSegundoPlano()
13: até que Melhor seja a solução ideal ou o tempo tenha se esgotado
14: devolva Melhor
```

Ant Colony Optimization

- Vantagens
  - Flexibilidade
  - Viabilidade para problemas dinâmicos (características se alteram ao longo da execução)
- Desvantagens
  - Alto tempo computacional
  - Convergência prematura

Ant Colony Optimization

- Aplicações a escalonamentos
  - Problema do caixeiro viajante - AntSystem
  - Roteamento em redes de comutação de pacotes - AntNet
  - *Single Machine Total Weighted Tardiness Scheduling*
  - Problemas *Flow-Shop*

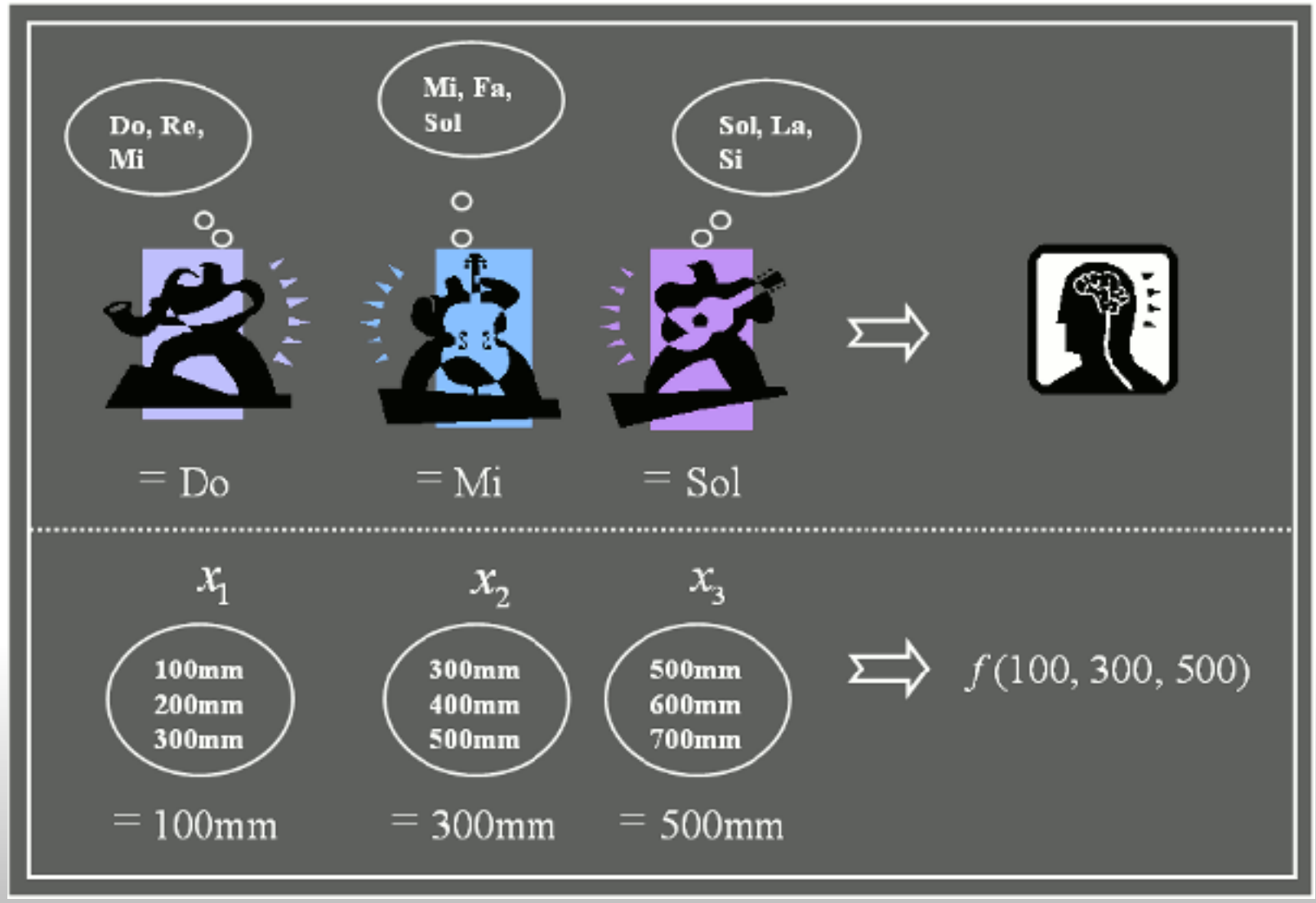
Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. Método do Gradiente
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. ***Harmony Search***
7. Conclusão

Harmony Search

- Metaheurística baseada em música (!!)
  - Derivada estocástica => variáveis discretas
  - Conhecimento obtido no improviso do Jazz
- Diversas analogias
  - Músico → variável de decisão
  - *Pitch* de instrumento → valor de variável
  - Harmonia musical → vetor de soluções
  - Estética auditiva → função objetivo
  - Experiência → matriz de memória
  - Prática → iteratividade

Harmony Search



Harmony Search

- Algoritmo com sete passos:
  - Formulação do problema
  - Definição de parâmetros do algoritmo
  - *Tuning* aleatório para iniciar memória
  - Improviso de harmonia
  - Atualização da memória
  - Execução de terminação
  - Cadência

Harmony Search

- Formulação do problema

Otimizar (minimizar ou maximizar) $f(x)$	(1)
Sujeita a:	
$h_i(x) = 0;$ i = 1, ... , p ;	(2)
$g_i(x) \geq 0;$ i = 1, ... , q ;	(3)
$x_i \in X_i = \{x_i(1),...,x_i(k),...,x_i(K_i)\}$ ou $x_i^L \leq x_i \leq x_i^U$	(4)

  - Função objetivo é o que mais importa

  - Violação de restrições?
    - Opção 1: abandonar a solução
    - Opção 2: aceitar penalidade

Harmony Search

- Definição de parâmetros do algoritmo
  - *k*: tamanho da memória de harmonia
    - número simultâneo de vetores de solução
  - *phmcr*: memória de harmonia considerando taxas
    - obtenção de valor da memória
  - *ppar*: taxa de ajuste de *pitch*
    - alteração de um valor obtido da memória
  - *fw* ou *bw*: largura das casas
    - comprimento arbitrário para variáveis contínuas
  - δ: largura das casas (?)
    - distância entre dois valores de conjunto discreto
  - improviso máximo
    - número de iterações

Harmony Search

- *Tuning* aleatório: improvisar harmonias (no mínimo o tamanho da memória) e selecionar as melhores
- Improviso de harmonia: escolher nota da memória ou do alcance (se da memória, alterar ou manter?)
- Atualização da memória: se a harmonia encontrada for melhor do que a pior, descartar a pior
- Execução da terminação: finalizar se necessário, senão improvisar e iniciar nova iteração
- Cadência: execução de procedimento após o algoritmo

Harmony Search

- Exemplo de implementação

01: inicie a memória de harmonia, selecionando k vetores aleatórios $x_1 \ldots x_k$.
02: repita
03: crie um novo vetor x'
04: para cada componente x'_i, faça
05: com probabilidade p_{hmcr}, faça $x'_i \leftarrow x_i^{int(rand(0,1)*k)+1}$
06: com probabilidade 1 - p_{hmcr}, escolha um novo valor aleatório no intervalo
07: para cada componente x'_i escolhido da memória, faça
08: com probabilidade p_{par}, altere x'_i com uma pequena quantia. Para variáveis discretas, $x'_i \leftarrow x'_i \pm \delta$; para contínuas, $x'_i \leftarrow x'_i \pm bw \cdot rand(0,1)$
09: com probabilidade $1 - p_{par}$, nada faça
10: se x' for melhor do que o pior x^i na memória, troque x^i por x'
11: até que o improviso máximo seja atingido
12: devolva a melhor harmonia na memória

Harmony Search

- Vantagens
  - Trata variáveis discretas ou contínuas
  - Estrutura simples => flexibilidade
  - Abundância de informações
    - http://www.hydroteq.com
- Desvantagens
  - Compreensão teórica
  - Dependência de formulação do problema

Harmony Search

- Diversas aplicações em outras áreas
  - IA, Visão Computacional
  - Engenharia, Medicina e Biologia
- Poucas aplicações a escalonamentos (por enquanto)
  - Escalonamento de diques
    - Extração com máximo benefício na geração de energia hídrica e na irrigação

Estudo sobre as Metaheurísticas

1. Metaheurísticas e Otimização
2. Objetivo do Estudo
3. Método do Gradiente
4. *Hill Climbing*
5. *Ant Colony Optimization*
6. *Harmony Search*
7. **Conclusão**

Conclusão

- Metaheurísticas podem ter as mais inesperadas e inusitadas origens
- Aplicações em diversas áreas, não só a escalonamentos
- Não existe uma metaheurística suprema
- Trata-se de uma área de pesquisa ativa
- Pesquisa completa demandaria muito mais do que algumas semanas

Referências

- LUKE, S. (2009). *Essentials of Metaheuristics.* Disponível em http://cs.gmu.edu/~sean/book/metaheuristics/.
- GLOVER, F. e KOCHENBERGER, G. A. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.
- GEEM, Z.W.. *State-of-the-Art in the Structure of Harmony Search Algorithm*. Disponível em: http://www.hydroteq.com/HS_Structure.pdf.
- http://en.wikipedia.org/wiki/Harmony_search

- As imagens presentes nesta apresentação não são de nossa autoria, todos os créditos são, portanto, de seus respectivos autores.