INTELIGÊNCIA ARTIFICIAL

PROBLEMA DE MÍNIMA LATÊNCIA

Bianca Fragoso
Pedro Felipe Magalhães

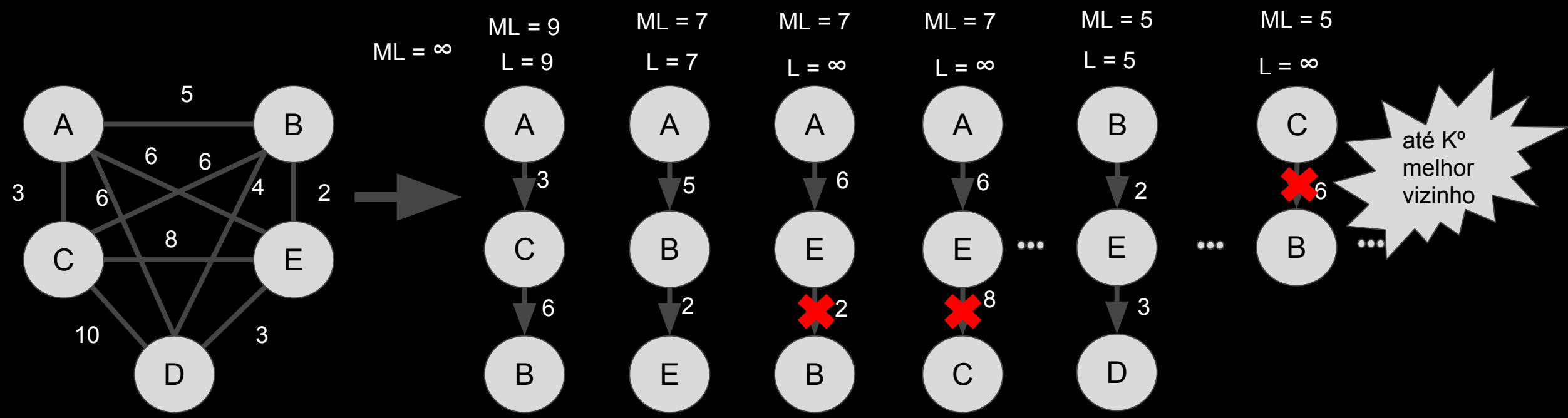
BUSCA GULOSA:

BUSCA COM PROFUNDIDADE LIMITADA E PODA

- Inicializamos o “caminho_final” com apenas o depósito
- Passamos uma profundidade “p” para a função e o número de vizinhos a serem explorados “k”
- Fazemos busca em profundidade nos “k” melhores vizinhos de cada nó
- A profundidade é limitada e não até o fim do grafo
- Escolhemos o vizinho que pertence ao caminho de menor latência até a profundidade que atingimos.
- Adicionamos apenas esse melhor vizinho ao caminho_final
- Repetimos o procedimento até que o caminho esteja completo.

BUSCA GULOSA:

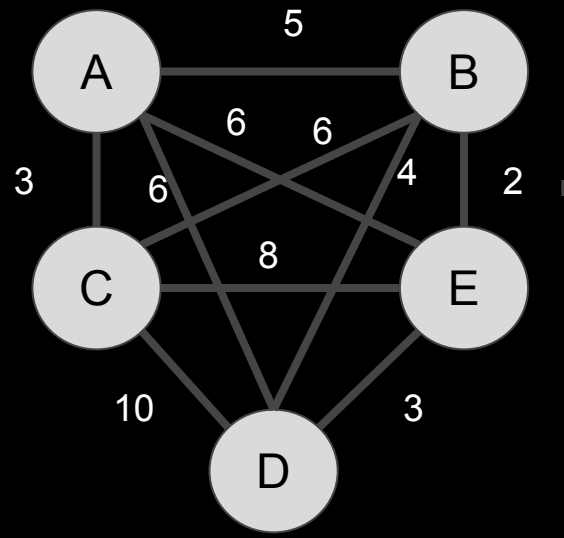
BUSCA COM PROFUNDIDADE LIMITADA E PODA



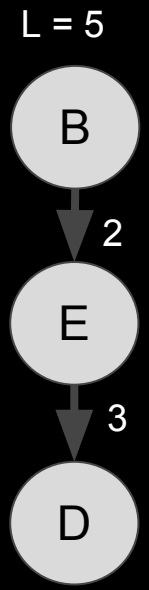
Representa a “poda” da árvore. Como a latência caminho já ultrapassaria o valor da melhor latência (ML) interrompemos a exploração do caminho retornando latência(L) = ∞.

BUSCA GULOSA:

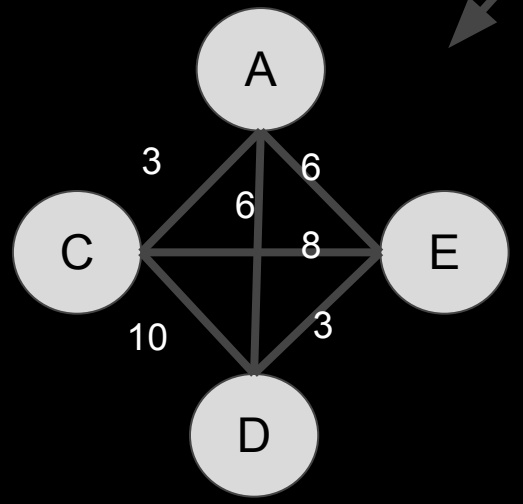
BUSCA COM PROFUNDIDADE LIMITADA E PODA



Caminho de menor latência encontrado



Como o caminho de melhor latência se inicia em B, adicionamos apenas ele ao nosso “caminho_final” e repetimos o procedimento para o grafo (G - {B})



BUSCA GULOSA:

Pseudo código

```
busca_gulosa_dfs(inicial,p,k)
  path <- append(inicial)
  vizinhos <- pegaMelhoresVizinhos(inicial)
  for i = 1 .. numero de cidades:
      for v = 1 .. k
          se vizinhos[v] nao esta no path:
              lat =  dfs_latencia(vizinhos[v],p,k)
      path <- append(vizinhos[v] com menor lat retornada)
  path <- append(inicial)
  retorna path
```

BUSCA GULOSA:

Pseudo código

```
dfs_latencia (inicial , p, k)
  se p == 0
      se latencia < melhor latência calculada
          atualiza melhor latência
      retorna latência
  se latencia > melhor latência calculada
      retorna MAX_INT
  for i = 1 .. k:
      lat = dfs_latencia(vizinhoOrd[i],p,k)
  retorna melhor latencia
```

BUSCA GULOSA:

BUSCA COM PROFUNDIDADE LIMITADA E PODA

Resultados

	Melhores resultados já obtidos	Nossos melhores resultados
dantzig42	12528	13047(+4%,profundidade = 8, k = 4)
gr48	102378	106799 (+4%, profundidade = 11, k = 4)
brazil58	512361	538013 (+5%, profundidade = 6, k = 4)
gr120	363454	381391 (+5%, profundidade = 10, k = 4)
pa561	658870	726279 (+10%, profundidade = 9, k = 4)

BUSCA GULOSA:

BUSCA COM PROFUNDIDADE LIMITADA E PODA

Tempo

	Tempo para obter melhores resultados já obtidos	Tempo para obter nossos melhores resultados
dantzig42	0.17	0.174s (profundidade = 8, k = 4)
gr48	0.31	14.284s (profundidade = 11, k = 4)
brazil58	0.55	0.024s(profundidade = 6, k = 4)
gr120	9.54	34.369s (profundidade = 10, k = 4)
pa561	1155.32	120.095s (profundidade = 7, k = 4)

META-HEURÍSTICA:

SIMULATED ANNEALING

- Método baseado nas experiências químicas em sólidos
- Começa com temperatura alta
- Vai diminuindo temperatura do sólido lentamente

META-HEURÍSTICA:

SIMULATED ANNEALING

- Calcula solução inicial usando busca gulosa anterior
- Repete duas vezes a parte principal do algoritmo
- Parte principal tem loop enquanto temperatura for maior que 1

META-HEURÍSTICA:

SIMULATED ANNEALING

- Enquanto temperatura é maior que 1:
  - ➢ Calculamos vizinhança
  - ➢ Passamos por toda a vizinhança
    - Se vizinho for melhor que a solução corrente, tornamos ela como solução corrente e a procura continua
    - Se for pior aceitamos a solução dentro de uma certa probabilidade e paramos loop
  - ➢ Atualizamos temperatura

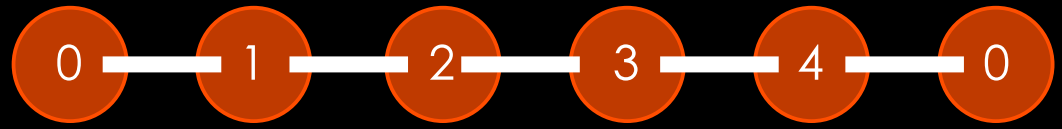
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

2-Opt

Solução Base:



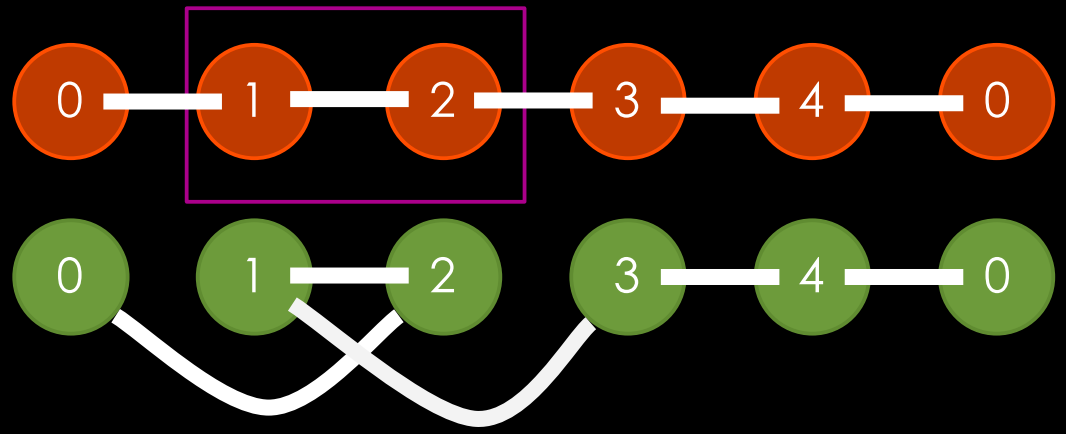
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

2-Opt

Solução Base:



Primeira

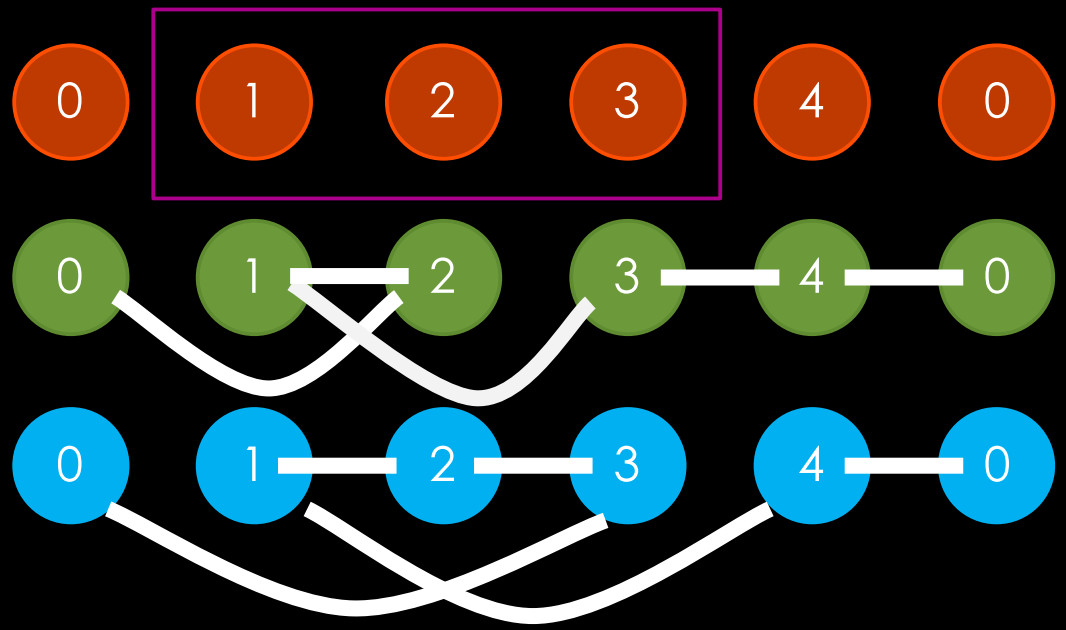
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

2-Opt

Solução Base:



Primeira

Segunda

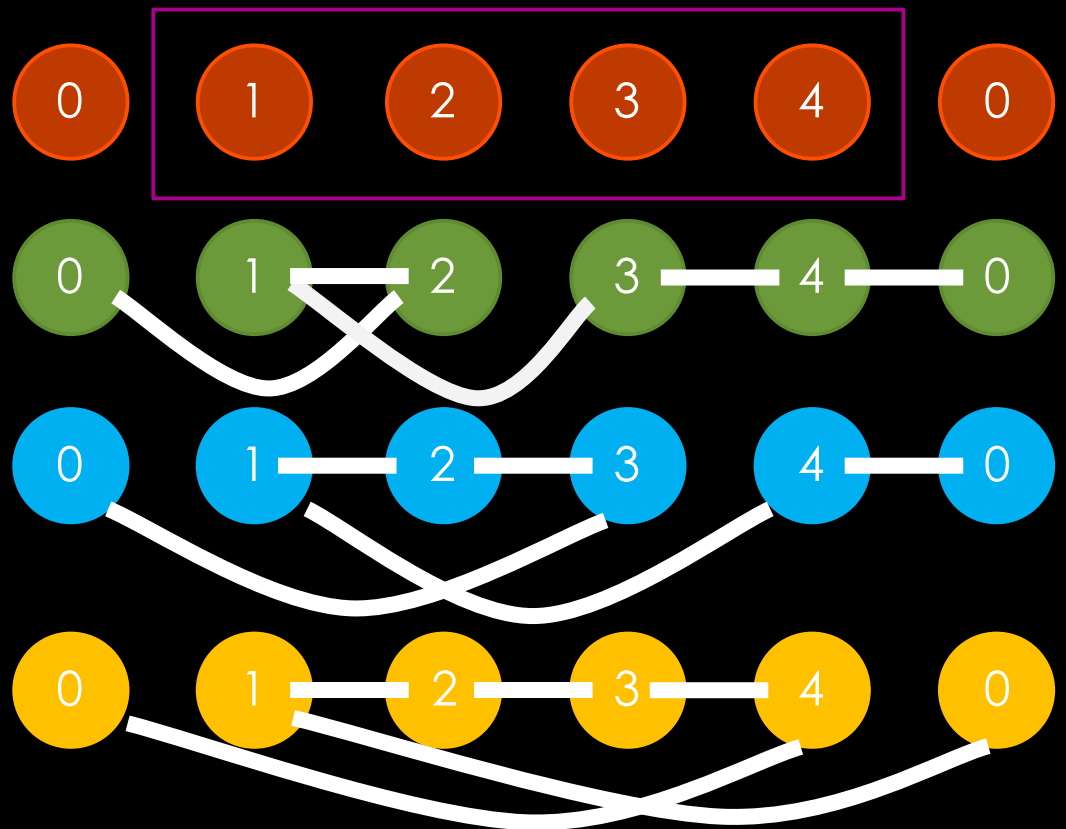
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

2-Opt

Solução Base:



Primeira

Segunda

Terceira

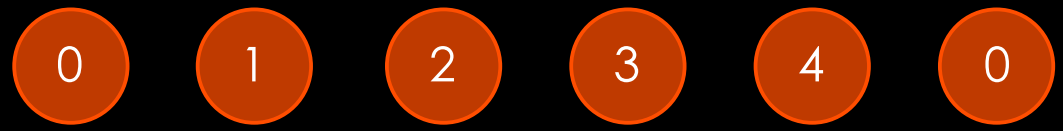
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

Swap

Solução Base:



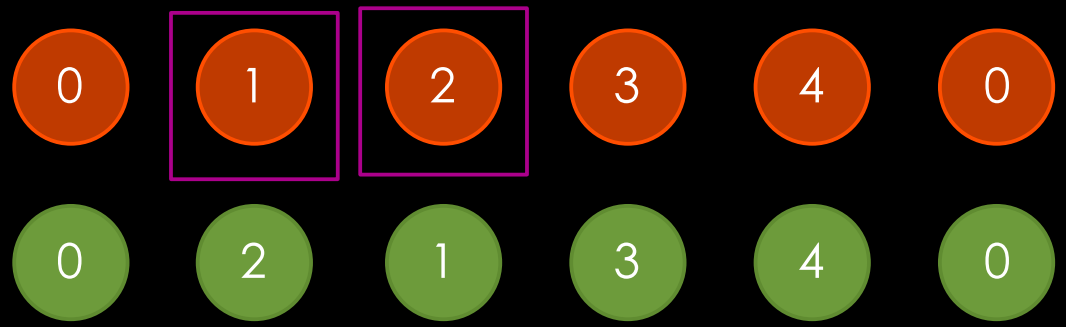
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

Swap

Solução Base:



Primeira

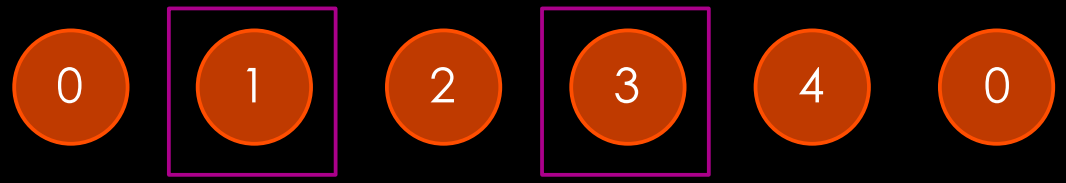
META-HEURÍSTICA:

SIMULATED ANNEALING

Vizinhanças

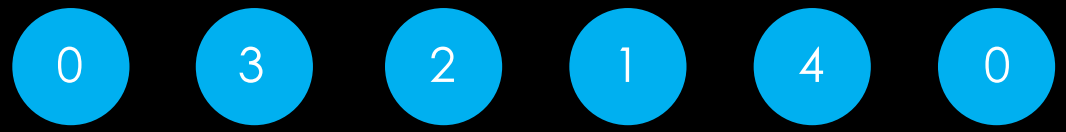
Swap

Solução Base:



Primeira

Segunda



META-HEURÍSTICA:

SIMULATED ANNEALING

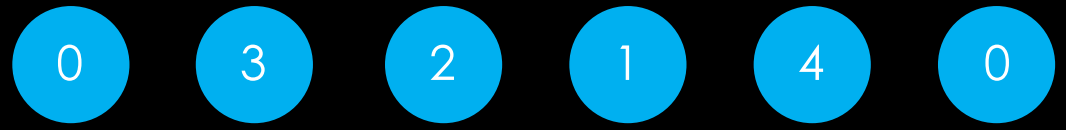
Vizinhanças

Swap

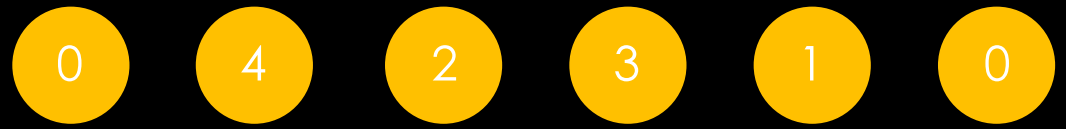
Solução Base:

Primeira

Segunda



Terceira



META-HEURÍSTICA:

SIMULATED ANNEALING

Probabilidade

$$\Delta E = \text{novoValor} - \text{valorCorrente}$$

$$probabilidade = e^{\frac{-\Delta E}{T}}$$

novoValor -> valor da função objetiva do vizinho analizado no momento.

valorCorrente -> valor da função objetiva do vizinho anteriormente analizado

META-HEURÍSTICA:

SIMULATED ANNEALING

Temperatura

- Inicializa com 1000
- Fator para diminuição de temperatura é 0.002, ou seja, vai diminuindo 0.2%

$$T = T * (1 - 0.002)$$

META-HEURÍSTICA:

SIMULATED ANNEALING

Resultados

	Melhores resultados já obtidos	Nossos melhores resultados
dantzig42	12528	12562 (0.27%, profundidade = 3, k =4)
gr48	102378	103976 (1.5%, profundidade = 8, k =4)
brazil58	512361	513483 (0.21%, profundidade = 6, k =4)
gr120	363454	381391 (4.9% , profundidade = 9, k =4)
pa561	658870	726279 (+10%, profundidade = 9, k = 4)

META-HEURÍSTICA:

SIMULATED ANNEALING

Tempo

	Tempo para obter melhores resultados já obtidos	Tempo para obter nossos melhores resultados
dantzig42	0.17	2.48
gr48	0.31	4.71
brazil58	0.55	7.31
gr120	9.54	54.87
pa561	1155.32	2554.94

Alguns algoritmos implementados que não foram tão bem sucedidos

- Busca gulosa usando Kruskal adaptada

- Busca Local
  - Apresentava resultados piores que a Simulated Annealing
  - Melhorava rápido e depois caia num mínimo local