Heurísticas e Metaheuristicas

ILS - Iterated Local Search

Prof. Guilherme de Castro Pena

guilherme.pena@ufsj.edu.br

Sala: DCOMP 3.11

Departamento de Ciência da Computação
Universidade Federal de São João del-Rei

Material adaptado do Prof. André (UFV) e do Prof. Marcone (UFOP)

Agenda

Introdução

Visão geral:

- Sabemos que a qualidade de um ótimo local obtido por uma busca local depende da sua solução inicial.
- Na busca local *multistart* clássica, diferentes soluções iniciais são geradas aleatóriamente e a BL é aplicada para cada uma.
- O ILS (Iterated Local Search) tenta melhorar essa abordagem de forma a explorar o espaço de soluções por meio de **perturbações em ótimos locais**.
- O pressuposto é que a partir de perturbações em uma solução ótima local corrente, outros ótimos locais melhores podem ser gerados.
- Para isso, é importante que a perturbação:
  - Seja suficientemente forte para permitir que a busca local explore diferentes soluções.
  - Seja fraca o suficiente para evitar um reinício aleatório.

Introdução

Visão geral:

- Os componentes principais do ILS são:
  - **GeraSoluçãoInicial**: Produz uma solução inicial
  - **Busca Local**: Retorna uma solução melhorada
  - **Perturbacao**: Modifica a solução corrente guiando a uma solução intermediária
  - **CriterioAceitacao**: Decide de qual solução a próxima perturbação será aplicada

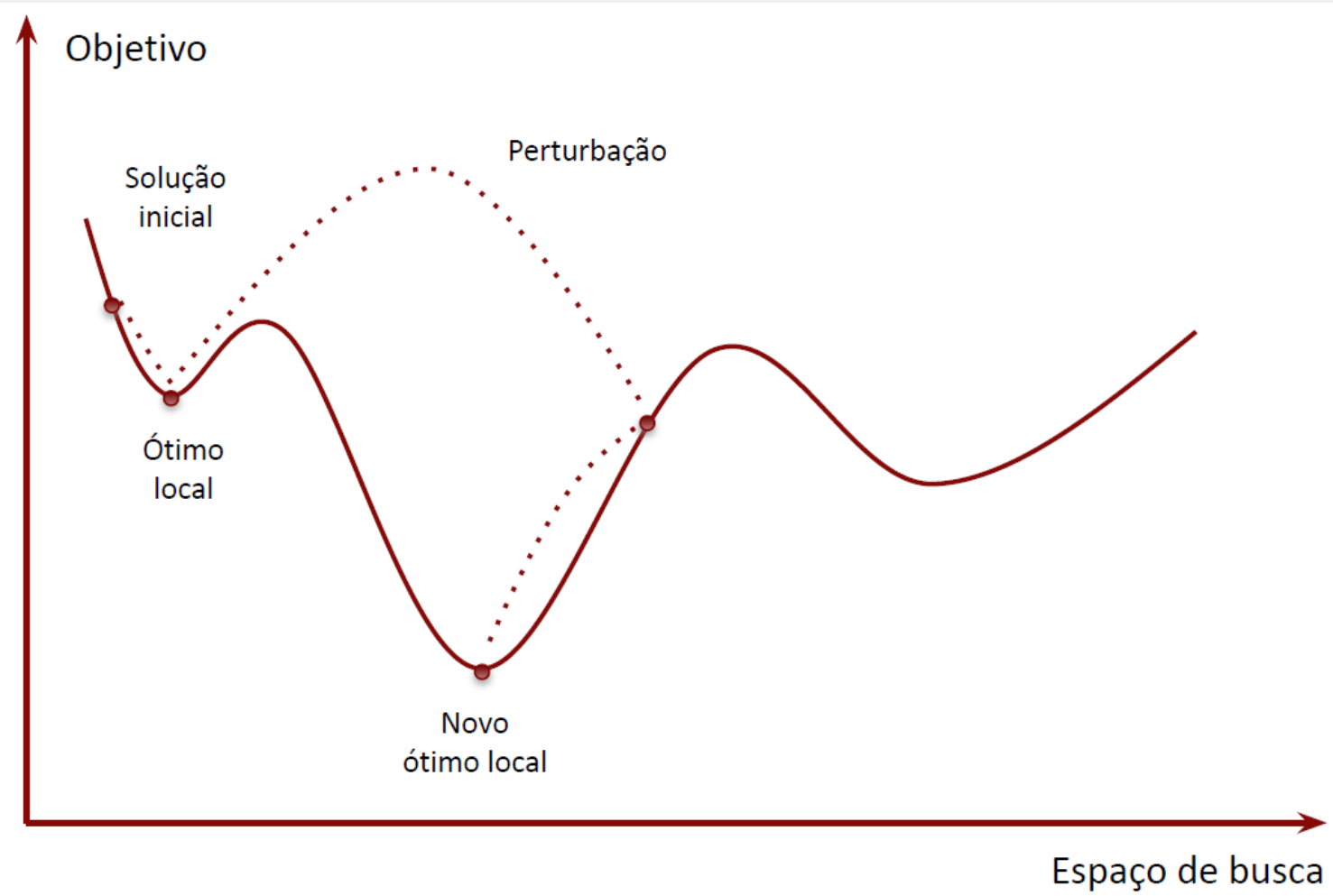
Introdução

Visão geral:

- Dessa forma, a ideia principal do ILS gira em torno de:
  - **Intensificação**:
    - Usar a Busca Local para atingir ótimo local de forma eficiente na região atual de busca.
    - É obtida fazendo-se "pequenas" perturbações na solução ótima local corrente.
  - **Diversificação**:
    - Usar a Perturbação para escapar de ótimo local de forma eficiente.
    - É obtida aumentando-se gradativamente a quantidade de perturbações na solução ótima local corrente.
  - E o **Critério de Aceitação** ele tenta então controlar a *intensificação* $\times$ *diversificação*.

Introdução

Visão geral:



Introdução

Pseudo-código:

- O pseudo-código do ILS básico:
- Um exemplo de condição de parada pode ser um **número de iterações sem melhora**: $iter < ILSMax$.
- Um segundo parâmetro importante a considerar seria **a força da perturbação** (d), também chamado *distância*, *nível* ou *comprimento*.

Algorithm 1: ILS Básico

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  while condicao parada não satisfeita do
5  |   iter ← iter + 1
6  |   s' ← Perturbacao(s, d)
7  |   s'' ← BuscaLocal(s')
8  |   s ← Aceitacao(s'', s)
9  end
10 return s
```

Introdução

Pseudo-código:

- A ideia básica é descrita, ou seja, uma busca local é aplicada na solução inicial.
- A partir desse ponto, repetidamente se aplica uma perturbação, uma nova busca local na solução perturbada e a verificação pelo critério de aplicação.

Algorithm 1: ILS Básico

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  while condicao parada não satisfeita do
5  |   iter ← iter + 1
6  |   s' ← Perturbacao(s, d)
7  |   s'' ← BuscaLocal(s')
8  |   s ← Aceitacao(s'', s)
9  end
10 return s
```

Agenda

Introdução

Pseudo-código:

- Geralmente usa-se uma busca local simples.
- Mas pode ser também uma *busca tabu*, *simulated annealing*.

Algorithm 1: ILS Básico

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  while condicao parada não satisfeita do
5  |   iter ← iter + 1
6  |   s' ← Perturbacao(s, d)
7  |   s'' ← BuscaLocal(s')
8  |   s ← Aceitacao(s'', s)
9  end
10 return s
```

Pseudo-código

Busca local simples:

- Na busca local, aplica-se a heurística de refinamento já conhecida por nós.
- Nesse exemplo, relembrando a busca local para um problema de minimização.

Algorithm 2: Busca Local com Best Improvement

1 $s \leftarrow s_0$ (Solução inicial)
2 $V = \{s' \in N(s) \mid f(s') < f(s)\}$ (Vizinhos de s)
3 **while** $|V| > 0$ **do**
4 $\quad s' = argmin\{f(s') \mid s' \in V\}$ (Melhor vizinho)
5 $\quad s \leftarrow s'$
6 $\quad V = \{s' \in N(s) \mid f(s') < f(s)\}$ (Gera Vizinhos do novo s)
7 **end**
8 **return** s

- Lembrando das pequenas modificações caso seja um problema de Max.

Perturbação

Visão geral:

- Na Perturbação deve-se conservar parte da solução e modificar fortemente outra parte.

Algorithm 1: ILS Básico

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  while condicao parada não satisfeita do
5  |   iter ← iter + 1
6  |   s' ← Perturbacao(s, d)
7  |   s'' ← BuscaLocal(s')
8  |   s ← Aceitacao(s'', s)
9  end
10 return s
```

Perturbação

Perturbação:

- A motivação do ILS é usar informações do **ótimo local atual** para gerar um novo ponto de partida.
  - A busca não é reiniciada de um ponto aleatório qualquer independente.
- A força (distância, comprimento) da perturbação é o número de elementos modificados na solução:
  - SAT: número de flips.
  - TSP: número de trocas de nós (ou arcos).

Perturbação

Perturbação:

- Como mencionado, a perturbação deve ser pequena o bastante para conservar características do ótimo local atual:
  - Mas não tão pequena! Se for, a busca local retorna ao mesmo ponto.
  - Idealmente, a perturbação deve ser tal que dificilmente é desfeita pela busca local
- A perturbação deve ser grande o bastante para escapar do ótimo local atual e explorar novas regiões:
  - Mas não tão grande! Se for, perderá informação já obtida durante a busca.
  - No extremo, seria como um reinício aleatório.

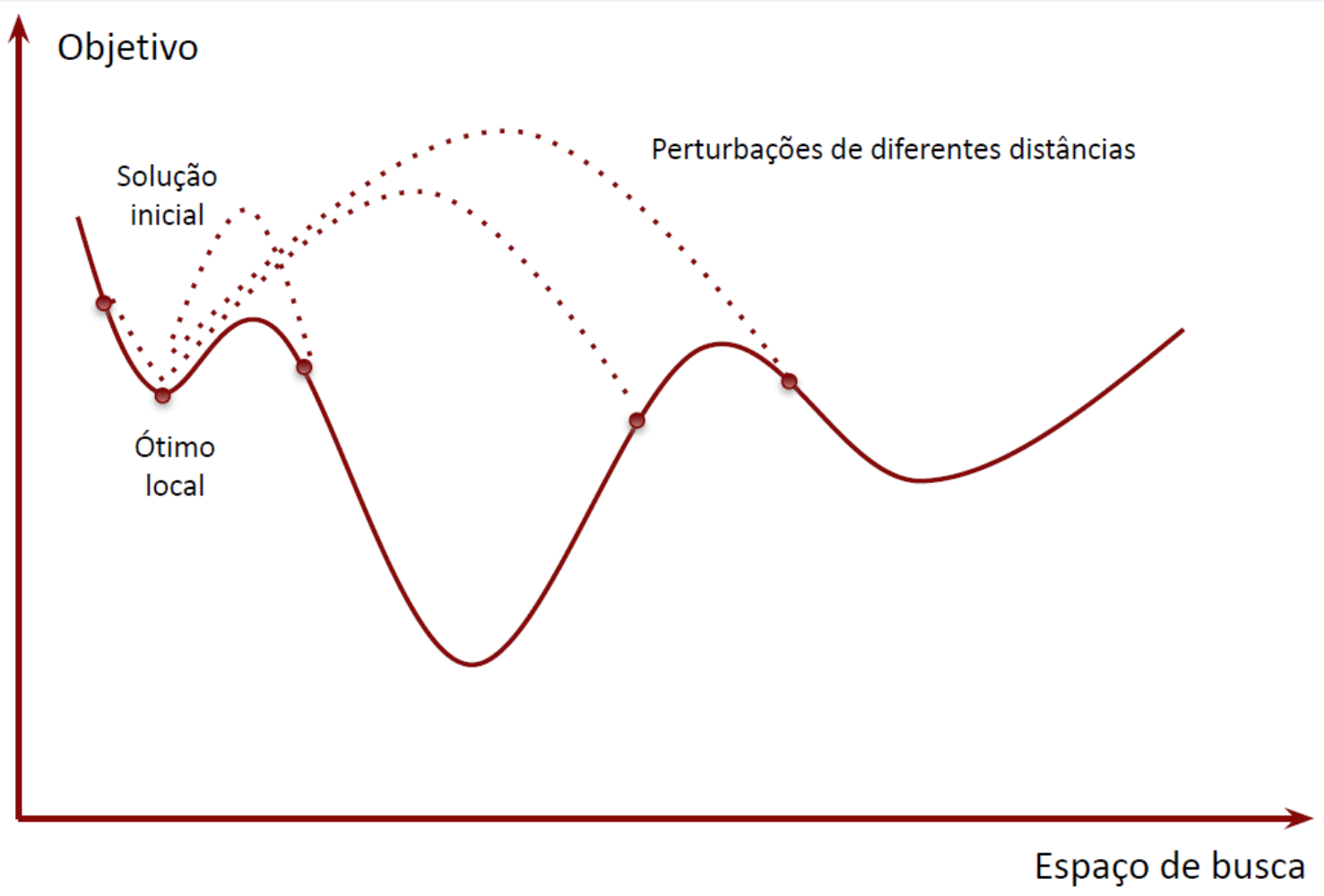
Perturbação

Perturbação:

- A perturbação pode ser feita por um ou mais passos de perturbação.
- Perturbação fraca
  - Busca local seguinte tende a ser mais rápida
  - Mas com risco de voltar ao mesmo ótimo local
- Perturbação forte
  - Escapa mais eficientemente do ótimo local
  - Mas pode se assemelhar a um reinício aleatório
- **Estratégia mais avançada**:
  - Como se fosse uma *perturbação adaptativa*.
  - Ela consiste em modificar a forma ou distância da perturbação durante a busca.

Perturbação

Perturbação:



Perturbação

Pseudo-código:

- Uma perturbação adaptativa é descrita no pseudo-código abaixo.
- De acordo com a quantidade de níveis, ou a força, a perturbação vai modificando a solução aleatoriamente.

Algorithm 2: Perturbacao(s_0, nivel)

```
1 s ← s_0 (Solução inicial)
2 NModificacoes ← nivel (Número de modificações na solução)
3 iter ← 0
4 while iter < NModificacoes do
5 |   Aplique movimento aleatório em s
6 |   iter ← iter + 1
7 end
8 return s
```

- Exemplos:
  - SAT/Mochila01: número de flips.
  - TSP: número de trocas de nós (ou arcos).
- Note que pela aleatoriedade, uma troca pode se repetir.

Aceitação

Visão geral:

- O papel principal da aceitação é, juntamente com a perturbação e busca local, balancear intensificação x diversificação:
  - Sempre aceitar a melhor das duas soluções candidatas (**o mais comum**).
  - Sempre aceitar a mais recente das duas soluções candidatas, guardando a melhor encontrada.
  - Aceitar a mais recente com alguma probabilidade, guardando a melhor encontrada (Tipo a aceitação do SA com algum resfriamento).
  - Aceitar usando um histórico de busca, por exemplo, ocasionalmente retornando à melhor solução.

ILS Básico

Pseudo-código:

- Versão mais básica considerando: problema de min, iterações sem melhora ($ILSMax$) e nível (d) adaptativo.

Algorithm 1: ILS Básico

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  d ← 1
5  while iter < ILSMax do
6  |   iter ← iter + 1
7  |   s' ← Perturbacao(s, d)
8  |   s'' ← BuscaLocal(s')
9  |   if f(s'') < f(s) then
10 |   |   s ← s''
11 |   |   iter ← 0
12 |   |   d ← 1
13 |   end
14 |   else
15 |   |   d ← d + 1
16 |   end
17 end
18 return s
```

Agenda

Considerações

Considerações:

- Diferentes versões do ILS podem ser propostas através de variações em seus componentes.
- Uma versão interessante, por exemplo, denominada *Smart ILS* foi proposta com a ideia de aumentar o nível de perturbação somente após algumas tentativas sem sucesso.
- Dessa forma, justificando-se que a região de busca pode não ter sido explorada adequadamente.
- Tal variação ocorre na etapa de aceitação e segue no pseudo-código a seguir.

Considerações

Considerações:

Algorithm 3: Smart ILS

```
1  s_0 ← SolucaoInicial
2  s ← BuscaLocal(s_0)
3  iter ← 0
4  d ← 1; NTenta ← 1
5  while iter < ILSMax do
6      iter ← iter + 1
7      s' ← Perturbacao(s, d)
8      s'' ← BuscaLocal(s')
9      if f(s'') < f(s) then
10         s ← s''
11         iter ← 0; d ← 1; NTenta ← 1
12     end
13     else
14         if NTenta ≥ TentativasMAX then
15             d ← d + 1; NTenta ← 1
16         end
17         else
18             NTenta ← NTenta + 1
19         end
20     end
21 end
22 return s
```

Exercício

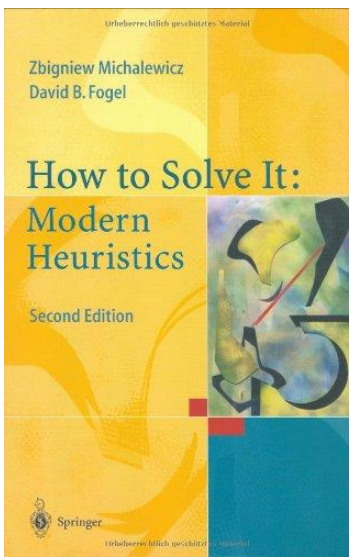
1. Faça um *ILS* para o TSP baseado no Algoritmo da página 19 (ILS mais básico):
   (a) Como critério de parada use um número de iterações sem melhora (*ILSMax*).
   (b) Para d pode-se usar o valor de exemplo, resetando para 1 ou aumentando caso não melhore.

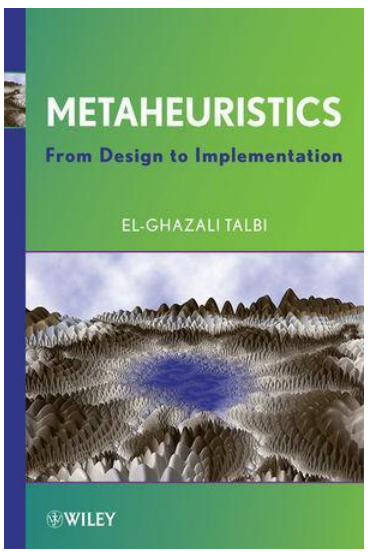
   Obs: Podemos usar a mesmas instâncias utilizadas naqueles exercícios anteriores.

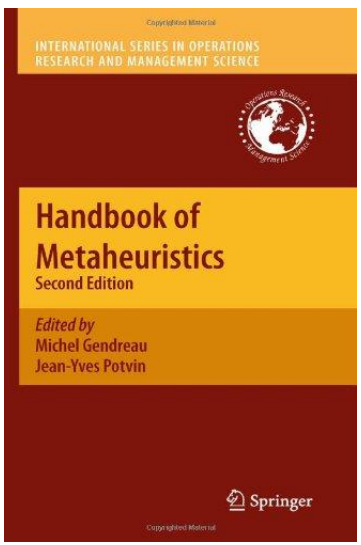
Bibliografias

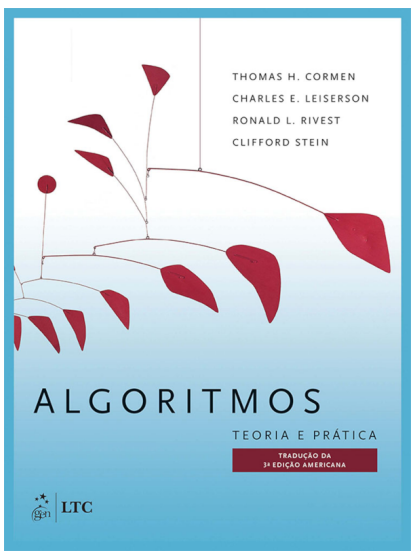
Bibliografia Básica

1. MICHLEWICZ, Zbigniew; FOGEL, David B. How to solve it: modern heuristics. 2nd. ed. Berlin: Springer c2010 554 p. ISBN 9783642061349.
2. Talbi, El-Ghazali; Metaheuristics: From Design to Implementation, Wiley Publishing, 2009.
3. GENDREAU, Michel. Handbook of metaheuristics. 2.ed. New York: Springer 2010 648 p. (International series in operations research & management science ; 146).
4. T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, 3rd edition, 2009 (**Pergamum**).









Bibliografias

Bibliografia Complementar

1. GLOVER, Fred; KOCHENBERGER, Gary A. (ed.). Handbook of metaheuristics. Boston: Kluwer, 2003. 556 p. (International series in operations research & management science ; 57).
2. BLUM, Christian Et Al. Hybrid metaheuristics: an emerging approach to optimization. Berlin: Springer 2008 289 p. (Studies in Computational intelligence; 114).
3. DOERNER, Karl F. (ed.) Et Al. Metaheuristics: progress in complex systems optimization. New York: Springer 2007 408 p. (Operations research / computer science interfaces series).
4. GLOVER, Fred; LAGUNA, Manuel. Tabu search. Boston: Kluwer Academic, 1997. 382 p.
5. AARTS, Emile. Local search in combinatorial optimization. Princeton: Princeton University Press, 2003 512 p.
6. Gaspar-Cunha, A.; Takahashi, R.; Antunes, C.H.; Manual de Computação Evolutiva e Metaheurística; Belo Horizonte: Editora UFMG; Coimbra: Imprensa da Universidade de Coimbra; 2013.