

# Busca Tabu (Tabu Search)

Marcone Jamilson Freitas Souza<sup>1,2,3</sup>

Puca Huachi Vaz Penna<sup>1</sup>

<sup>1</sup> Departamento de Computação

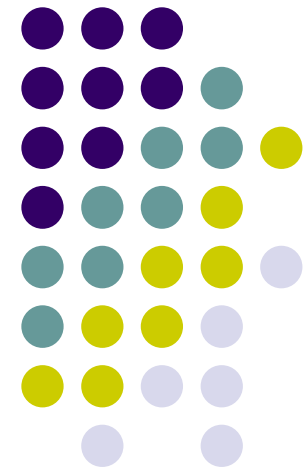
<sup>1</sup> Programa de Pós-Graduação em Ciência da Computação  
Universidade Federal de Ouro Preto

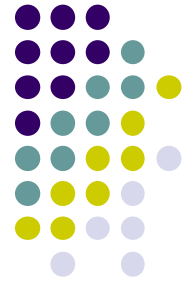
<sup>2</sup> Programa de Pós-graduação em Modelagem Matemática e  
Computacional / CEFET-MG

<sup>3</sup> Programa de Pós-graduação em Instrumentação, Controle e  
Automação de Processos de Mineração / ITV/UFOP

[www.decom.ufop.br/prof/marcone](http://www.decom.ufop.br/prof/marcone), [www.decom.ufop.br/puca](http://www.decom.ufop.br/puca)

E-mail: {marcone,puca}@ufop.edu.br





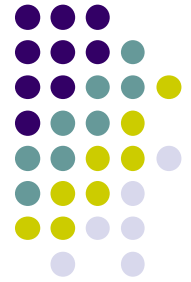
# Sumário

- Introdução
- Fundamentação
- Algoritmo básico
- Implementação da lista tabu
- Tamanho da lista tabu
- Análise da vizinhança
- Critérios de aspiração
- Memória de longo prazo
- Oscilação estratégica



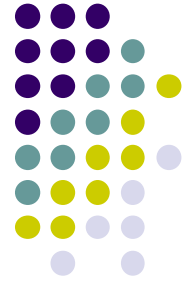
# Introdução

- Proposta por Glover (1986) e Hansen (1986), de forma independente
- Metaheurística de busca local
- Se apoia em estruturas de memória para guiar uma heurística de busca local além da otimalidade local



# Introdução

- Metaheurística poderosa
- Resolução eficiente de vários problemas combinatórios, destacando-se, entre outros:
  - Roteirização (Gendreau et al., 2006; Cordeau et al., 2002; Gendreau et al., 1999)
  - Sequenciamento (Allahverdi *et al.*, 2008)
  - Programação de horários (Santos *et al.*, 2005; Souza *et al.*, 2004)



# Introdução

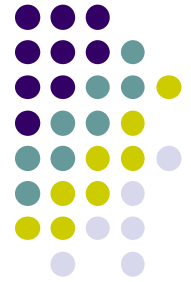
- Ingredientes típicos:

- Busca  
Tabu  
básica
- Geração da solução inicial;
  - Critério de escolha de vizinho;
  - Definição das regras de proibição: memória de curto prazo;
  - Critério de aspiração;
  - Definição de uma memória de longo prazo: intensificação x diversificação
  - Reconexão por Caminhos
  - Aplicação da oscilação estratégica

# Fundamentação: Problema da Mochila



- Há um conjunto de  $n$  itens e uma mochila de capacidade  $b$
- A cada item está associado um peso  $w_j$  e um valor de retorno  $p_j$
- Objetivo:
  - Alocar os itens à mochila de forma que se tenha o maior valor de retorno, respeitando-se a capacidade da mochila



# Problema da Mochila

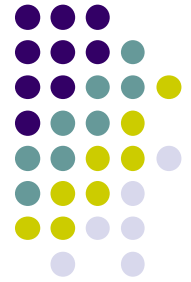
- Formulação PLI:

$$\max \sum_{j=1}^n p_j s_j$$

$$\sum_{j=1}^n w_j s_j \leq b$$

$$s_j \in \{0, 1\} \quad \forall j = 1, \dots, n$$

- $s_j = 1$  se o item  $j$  for alocado à mochila e 0, caso contrário.



# Problema da Mochila

- Estratégia de exploração do espaço de busca:
  - Soluções factíveis
  - Soluções infactíveis

$$f(s) = \sum_{j=1}^n p_j s_j - \rho \times \max\{0, \sum_{j=1}^n w_j s_j - b\}$$

- $\rho$  = penalidade por excesso de carga





# Problema da Mochila

- Representação de uma solução:
  - Vetor  $s = (s_1, s_2, \dots, s_n)$ , em que  $s_j \in \{0, 1\}$
- Solução inicial:
  - Gulosa:
    - Alocar os itens mais valiosos por unidade de peso até que haja capacidade disponível na mochila
  - Aleatória
    - Escolher aleatoriamente um item e adicioná-lo à solução parcial. Repetir este procedimento até que a capacidade da mochila esteja esgotada
- Movimento  $m$ : inverter o valor de um bit
  - Valor 0 muda para 1
  - Valor 1 muda para 0
- $N(s) = \{s' : s' \leftarrow s \oplus m\}$
- $\rho = \sum p_j$



# Problema da Mochila

- Ex.: Seja uma mochila de capacidade  $b = 32$

Item	1	2	3	4	5	6	7	8
$w_j$	4	15	7	9	8	10	9	11
$p_j$	2	2	3	4	6	5	8	7

- $s^0 = (1, 0, 0, 1, 0, 1, 1, 0)$
- $\rho = \sum p_j = 37$
- $f(s^0) = 19$



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^0 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0)^t$$

$$f(s^0) = 19$$

# viz.	$s' \in N(s^0)$	$peso(s')$	$f(s')$
1	$(\boxed{0}0010110)^t$	28	17
2	$(1\boxed{1}010110)^t$	47	-534
3	$(10\boxed{1}10110)^t$	39	-237
4	$(100\boxed{0}0110)^t$	23	15
5	$(1001\boxed{1}110)^t$	40	-271
6	$(10010\boxed{0}10)^t$	22	14
7	$(100101\boxed{0}0)^t$	23	11
8	$(1001011\boxed{1})^t$	43	-381



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

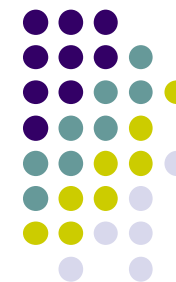
Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^0 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0)^t$$

$$f(s^0) = 19$$

# viz.	$s' \in N(s^0)$	$\text{peso}(s')$	$f(s')$
1	$(\boxed{0}0010110)^t$	28	17
2	$(1\boxed{1}010110)^t$	47	-534
3	$(10\boxed{1}10110)^t$	39	-237
4	$(100\boxed{0}0110)^t$	23	15
5	$(1001\boxed{1}110)^t$	40	-271
6	$(10010\boxed{0}10)^t$	22	14
7	$(100101\boxed{0}0)^t$	23	11
8	$(1001011\boxed{1})^t$	43	-381

Esta solução é um ótimo local, pois não há uma solução vizinha com melhor função de avaliação



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^0 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0)^t$$

$$f(s^0) = 19$$

# viz.	$s' \in N(s^0)$	$\text{peso}(s')$	$f(s')$
1	$(\boxed{0}0010110)^t$	28	$\boxed{17}$
2	$(1\boxed{1}010110)^t$	47	-534
3	$(10\boxed{1}10110)^t$	39	-237
4	$(100\boxed{0}0110)^t$	23	15
5	$(1001\boxed{1}110)^t$	40	-271
6	$(10010\boxed{0}10)^t$	22	14
7	$(100101\boxed{0}0)^t$	23	11
8	$(1001011\boxed{1})^t$	43	-381

Esta solução é um ótimo local, pois não há uma solução vizinha com melhor função de avaliação

**Ideia:** Mover para o melhor vizinho, ainda que de piora.

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^1 = (00010110)^t$$

$$f(s^1) = 17$$

#	$s' \in N(s^1)$	$\text{peso}(s')$	$f(s')$
1	$(\mathbf{1}0010110)^t$	32	19
2	$(0\mathbf{1}010110)^t$	43	-388
3	$(00\mathbf{1}10110)^t$	35	-91
4	$(000\mathbf{0}0110)^t$	19	13
5	$(0001\mathbf{1}110)^t$	36	-125
6	$(00010\mathbf{0}10)^t$	18	12
7	$(000101\mathbf{0}0)^t$	19	9
8	$(0001011\mathbf{1})^t$	39	-235

Observe que o vizinho #1, isto é,  $s^0$ , tem a melhor avaliação na vizinhança de  $s^1$ .



# Problema da Mochila

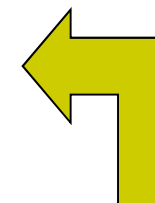
Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^1 = (00010110)^t$$

$$f(s^1) = 17$$

#	$s' \in N(s^1)$	$\text{peso}(s')$	$f(s')$
1	$(\mathbf{1}0010110)^t$	32	$\mathbf{19}$
2	$(0\mathbf{1}010110)^t$	43	-388
3	$(00\mathbf{1}10110)^t$	35	-91
4	$(000\mathbf{0}0110)^t$	19	13
5	$(0001\mathbf{1}110)^t$	36	-125
6	$(00010\mathbf{0}10)^t$	18	12
7	$(000101\mathbf{0}0)^t$	19	9
8	$(0001011\mathbf{1})^t$	39	-235



Retorna-se a uma solução já gerada anteriormente!



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^1 = (00010110)^t$$

$$f(s^1) = 17$$

Tabu:

#	$s' \in N(s^1)$	$\text{peso}(s')$	$f(s')$
1	$(\mathbf{1}0010110)^t$	32	$\mathbf{19}$
2	$(0\mathbf{1}010110)^t$	43	-388
3	$(00\mathbf{1}10110)^t$	35	-91
4	$(000\mathbf{0}0110)^t$	19	13
5	$(0001\mathbf{1}110)^t$	36	-125
6	$(00010\mathbf{0}10)^t$	18	12
7	$(000101\mathbf{0}0)^t$	19	9
8	$(0001011\mathbf{1})^t$	39	-235

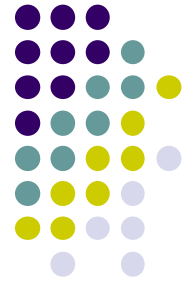
**Ideia:** Criar uma Lista T das soluções já geradas (**Lista Tabu**).





# Lista Tabu de soluções

- É inviável armazenar uma lista de **todas** as soluções geradas
  - Solução: Armazenar apenas as últimas  $|T|$  soluções geradas
  - Uma lista de tamanho  $|T|$  impede ciclos de até  $|T|$  iterações
  - Esta estratégia está coerente com o fato de que na história da busca não influenciam na ciclagem soluções geradas em um passado distante



# Lista Tabu de soluções

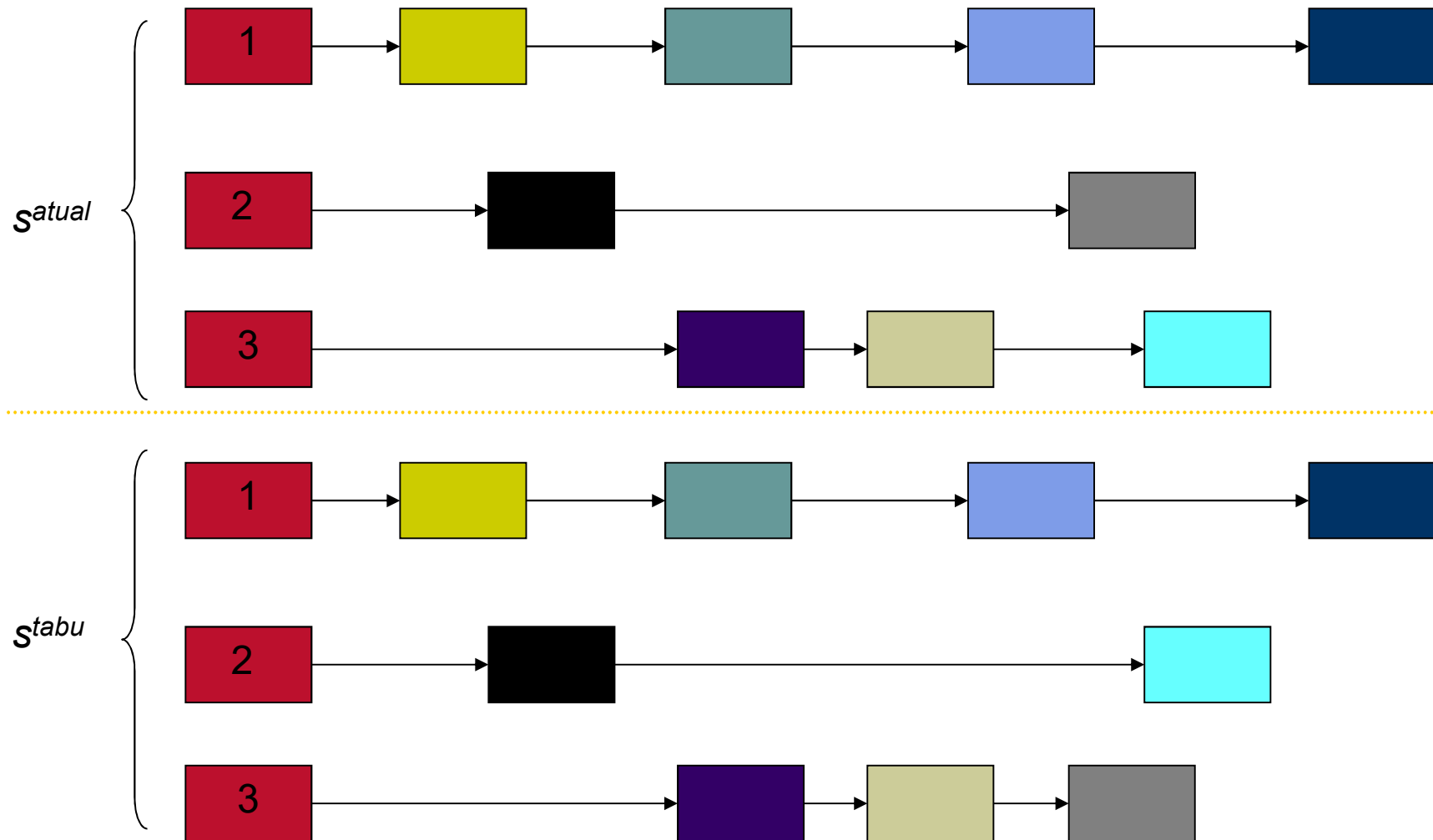
- Em muitos problemas, requer-se:
  - muita memória para armazenar uma lista tabu de soluções,
  - alto custo computacional para verificar se uma solução é ou não tabu.



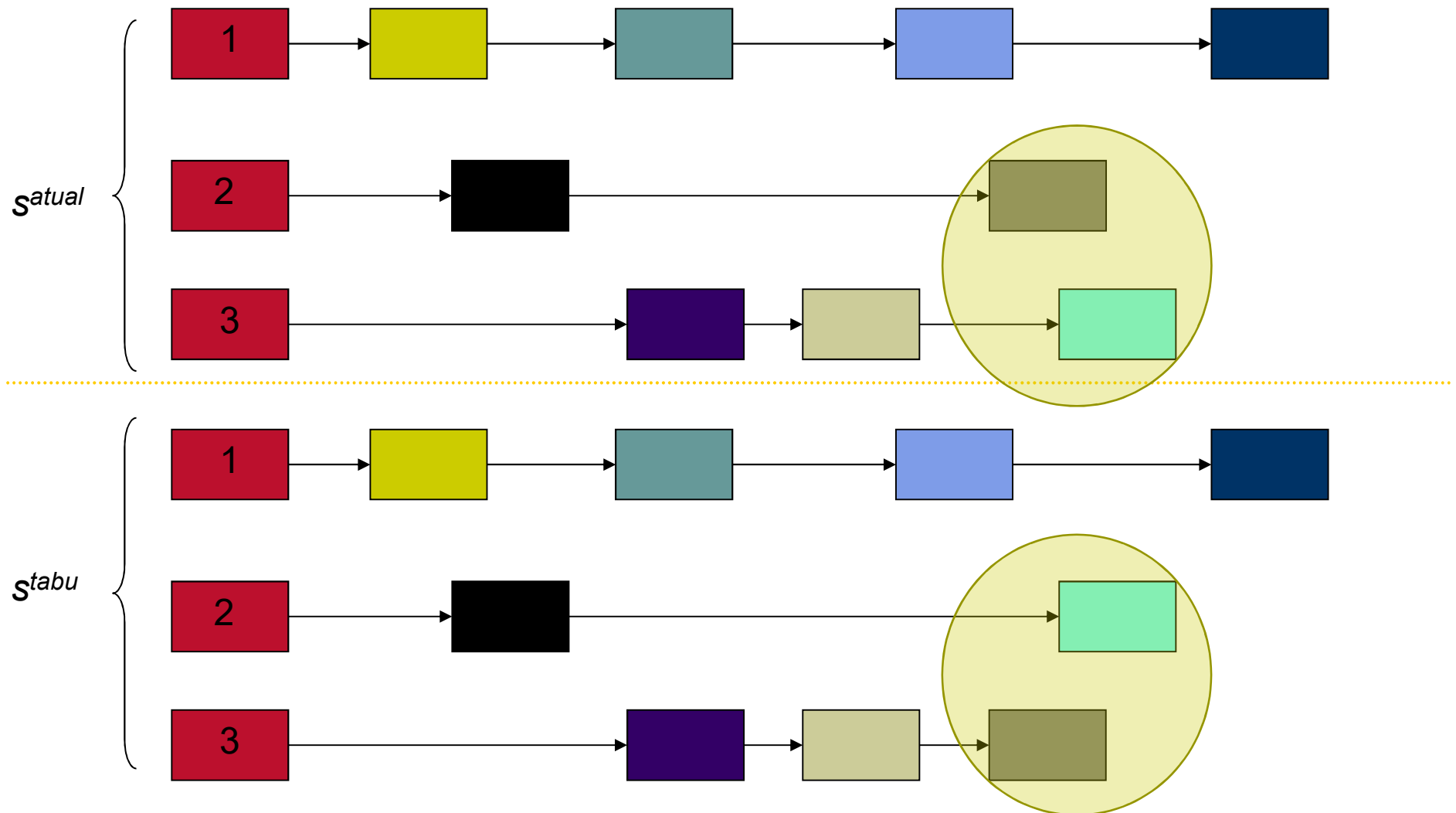
# Lista Tabu de soluções

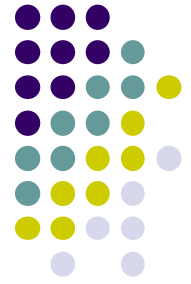
- Como faço para verificar se uma solução já foi gerada?
- Seja um problema de programação de tripulações
- Suponha que a solução  $s^{\text{atual}}$  é a solução atual e  $s^{\text{tabu}}$  é a única solução tabu gerada até então. Portanto, a lista tabu  $T$  é formada apenas por  $s^{\text{tabu}}$ , isto é,  $T = \{s^{\text{tabu}}\}$

# Programação de tripulações



# Programação de tripulações





# Lista Tabu de soluções

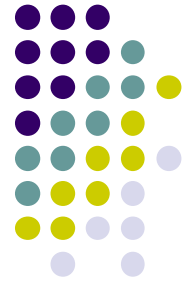
- As duas soluções não são iguais
- Elas diferem entre si apenas com relação à troca de duas tarefas
- $s^{\text{atual}}$  não é tabu, mas a verificação deste fato pode consumir um tempo alto

# Lista Tabu baseada em atributos de soluções/movimentos



- **Ideia:** Ao invés de armazenar toda a solução, guardar **apenas** alguma característica (**atributo**) da solução / movimento e uma **regra de proibição** (ativação tabu do atributo) para impedir o retorno a uma solução já gerada
- Atributos selecionados são denominados tabu-ativos
- Soluções que contêm elementos tabu-ativos tornam-se tabu
- Movimento tabu: movimento que leva a uma solução tabu

# Lista Tabu baseada em atributos de soluções/movimentos



- Exemplo: problemas de permutação.
- Seja  $s = (2, 6, 1, 5, 4, 3)$
- Movimentos:
  - Troca de posição entre dois elementos
    - $s = (2, 6, 1, 3, 4, 5)$
  - Inserção de um elemento em outra posição
    - $s = (2, 6, 1, 4, 3, 5)$
- Atributos: um elemento e sua posição



# Lista Tabu baseada em atributos de soluções/movimentos



- Seja a troca da tarefa  $s_i$  da posição  $i$  com a tarefa  $s_j$  da posição  $j$
- Seja  $s = (2, 6, 1, 5, 4, 3)$  a solução que será alterada pela troca de seus elementos da quarta e sexta posições, respectivamente, isto é,  $s_4 = 5$  com  $s_6 = 3$
- Regras de ativação tabu para movimentos de troca e inserção:
  - 1) Impedir qualquer movimento que resulte em uma permutação em que  $s_i$  ocupe a posição  $i$  e  $s_j$  ocupe a posição  $j$ :
    - Considerando  $s_i=5$  e  $s_j=3$  então é proibido que:
      - a tarefa 5, da quarta posição  $i = 4$ , ocupe a sexta posição ( $j = 6$ ) **e**
      - a tarefa 3, da sexta posição ( $j = 6$ ), ocupe a quarta posição ( $i = 4$ )
  - 2) Impedir qualquer movimento que resulte em uma permutação em que  $s_i$  ocupe a posição  $i$  **ou**  $s_j$  ocupe a posição  $j$ :
    - Considerando  $s_i=5$  e  $s_j=3$  então é proibido que:
      - a tarefa 5, da quarta posição, ocupe a sexta posição **ou**
      - a tarefa 3, da sexta posição, ocupe a quarta posição

# Lista Tabu baseada em atributos de soluções/movimentos



- Seja  $s = (2, 6, 1, 5, 4, 3)$  a solução que será alterada pela troca de seus elementos da quarta e sexta posições, respectivamente, isto é,  $s_4 = 5$  com  $s_6 = 3$
- Regras de ativação tabu para movimentos de troca e inserção:
  - 3) Impedir  $s_i$  de retornar à posição  $i$ :
    - Proibir que a tarefa  $s_i=5$  retorne à quarta posição  $i = 4$
  - 4) Impedir  $s_i$  de se mover para a posição  $k$ , sendo  $k \leq i$ :
    - Proibir que a tarefa  $s_i=5$  mova para uma posição menor ou igual à quarta
  - 5) Impedir  $s_i$  de se mover para a posição  $k$ , sendo  $k \leq j$ :
    - Proibir que a tarefa  $s_i=5$  mova para qualquer posição menor ou igual à sexta. No caso, qualquer movimentação da tarefa  $s_i=5$  estaria proibida, já que a sexta posição é a última.

# Lista Tabu baseada em atributos de soluções/movimentos



- Seja  $s = (2, 6, 1, 5, 4, 3)$  a solução que será alterada pela troca de seus elementos da quarta e sexta posições, respectivamente.
- Regras de ativação tabu para movimentos de troca e inserção:
  - 6) Impedir  $s_i$  de se mover:
    - Proibir a tarefa  $s_i = 5$  de se mover.
  - 7) Impedir  $s_i$  e  $s_j$  de trocarem de posição:
    - Proibir a tarefa  $s_i = 5$  de trocar com qualquer outra, bem como a tarefa  $s_{ij} = 3$  com qualquer outra
  - 8) Impedir  $s_i$  e  $s_j$  de se moverem:
    - Proibir a tarefa  $s_i = 5$  de se mover para qualquer posição, bem como a tarefa  $s_{ij} = 3$ .
- Algumas regras de ativação tabu podem ser mais restritivas do que outras!

# Problema de Alocação de Aulas a Salas



H\S	1	2	3
1	A		
2			D
3		C	D
4	B	C	

$S$

H\S	1	2	3
1	A		
2			D
3		C	D
4		C	B

$S'$

- Conjunto de salas; Conjunto de horários
- Cada célula  $s(i,j)$  representa a turma que tem aula no horário  $i$  e sala  $j$

**Movimento de realocação:** Transferir uma aula de uma sala para outra que esteja vazia

# Problema de Alocação de Aulas a Salas



H\S	1	2	3
1	A		
2			D
3		C	D
4	B	C	

$S$

H\S	1	2	3
1	A		
2		D	
3		D	C
4	B		C

$S'$

- Conjunto de salas; Conjunto de horários
- Cada célula  $s(i,j)$  representa a turma que tem aula no horário  $i$  e sala  $j$

**Movimento de troca:** Trocar as aulas de duas turmas realizadas em salas distintas

# Problema de Alocação de Aulas a Salas



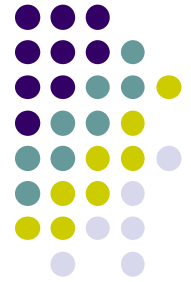
H\S	1	2	3
1	A		
2			D
3		C	D
4	B	C	

S

H\S	1	2	3
1	A		
2		D	
3		D	C
4	B		C

S'

- Atributos: aula (definida pelo seu horário  $i$  de início) e sua sala  $j$  de realização
  - Pares  $(i_1, j_1)$  e  $(i_2, j_2)$
- Algumas regras de proibição:
  - Impedir a troca das aulas envolvendo as salas  $j_1$  e  $j_2$ , iniciadas no horário  $i = \min\{i_1, i_2\}$
  - Impedir que a aula da sala  $j_1$  iniciada no horário  $i_1$  seja mudada para a sala  $j_2$
  - Impedir que a aula da sala  $j_1$  com início no horário  $i_1$  seja mudada
  - Idem, uma das regras anteriores, associadas ao valor da função objetivo antes do mov.



# Critérios de aspiração

- Uma lista tabu baseada em atributos de movimentos/soluções é restritiva:
  - Impede-se não somente o retorno a uma solução já gerada anteriormente, mas também a outras soluções ainda não visitadas

# Critérios de aspiração



H\S	1	2	3
1	A		
2			D
3		C	D
4	B	C	

$s^0$

$T = \{\}$

H\S	1	2	3
1	A		
2			D
3		C	D
4		C	B

$s^1$

$T = \{ \langle 4, 3, 1 \rangle \}$

Movimento = Transferir uma aula de uma sala para outra que esteja vazia

Movimento tabu =  $\langle \text{Horário de início, Sala nova, Sala original} \rangle$



# Critérios de aspiração



H\S	1	2	3
1	A		
2	D		
3	D	C	
4		C	B

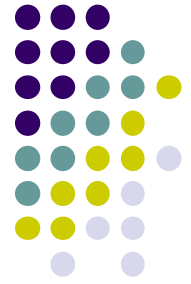
$s^2$

H\S	1	2	3
1	A		
2	D		
3	D	C	
4	B	C	

$s^3$

$$T = \{ \langle 4, 3, 1 \rangle, \langle 2, 1, 3 \rangle \}$$

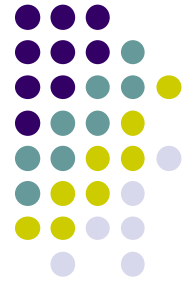
Fazendo-se o movimento tabu  $\langle 4, 3, 1 \rangle$  geramos  $s^3 \neq s^0$



# Critérios de aspiração

- **Ideia:** Retirar a regra de proibição (retirar o *status* tabu de um movimento) sob certas condições.
- Exemplo: retirar a regra de proibição se for gerada uma solução melhor que a melhor solução gerada até então (Critério de aspiração por objetivo global)

# Algoritmo básico de Busca Tabu



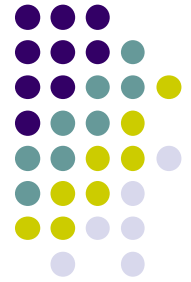
---

## Algorithm 1 Busca Tabu

---

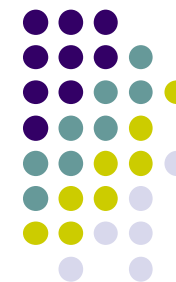
```
1: Entrada:  $f(\cdot), N(\cdot), A(\cdot), |V|, |T|, s$ 
2: Saída:  $s^*$ 
3:  $s^* \leftarrow s$            {Melhor solução encontrada até então}
4:  $Iter \leftarrow 0$        {Contador do número de iterações}
5:  $MelhorIter \leftarrow 0$  {Iteração mais recente que forneceu  $s^*$ }
6:  $T \leftarrow \emptyset$    {Lista Tabu}
7: Inicialize a função de aspiração  $A$ 
8: while Critério de parada não satisfeito do
9:    $Iter \leftarrow Iter + 1$ 
10:  Seja  $s' \leftarrow s \oplus m$  o melhor elemento de  $V \subseteq N(s)$  tal que o movimento  $m$  não
    seja tabu ( $m \notin T$ ) ou  $s'$  atenda a condição de aspiração ( $f(s') < A(f(s))$ )
11:  Atualize a lista tabu  $T$ 
12:   $s \leftarrow s'$ 
13:  if  $f(s) < f(s^*)$  then
14:     $s^* \leftarrow s$ 
15:     $MelhorIter \leftarrow Iter$ 
16:  end if
17:  Atualize a função de aspiração  $A$ 
18: end while
19: Retorne  $s^*$ 
```

---



# Problema da Mochila

- Voltando ao Problema da Mochila ...
- Solução  $s = (s_1, s_2, \dots, s_i, \dots, s_j, \dots, s_n)$ , em que  $s_i \in \{0, 1\}$
- Movimento: inverter o valor de um bit
- Atributo: posição de um item
- Regra de ativação: impedir a inversão do valor do bit da posição modificada
- Representação da Lista Tabu:
  - $T = \{<\text{posição modificada}>\}$
- Seja  $|T| = 2$



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^0 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0)^t$$

$$f(s^0) = 19$$

# viz.	$s' \in N(s^0)$	$\text{peso}(s')$	$f(s')$
1	$(\boxed{0}0010110)^t$	28	$\boxed{17}$
2	$(1\boxed{1}010110)^t$	47	-534
3	$(10\boxed{1}10110)^t$	39	-237
4	$(100\boxed{0}0110)^t$	23	15
5	$(1001\boxed{1}110)^t$	40	-271
6	$(10010\boxed{0}10)^t$	22	14
7	$(100101\boxed{0}0)^t$	23	11
8	$(1001011\boxed{1})^t$	43	-381

O melhor vizinho da solução  $s^0$  é o vizinho #1, que teve o bit da posição #1 alterado.



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^0 = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 0)^t$$

$$f(s^0) = 19$$

# viz.	$s' \in N(s^0)$	$peso(s')$	$f(s')$
1	$(\boxed{0}0010110)^t$	28	$\boxed{17}$
2	$(1\boxed{1}010110)^t$	47	-534
3	$(10\boxed{1}10110)^t$	39	-237
4	$(100\boxed{0}0110)^t$	23	15
5	$(1001\boxed{1}110)^t$	40	-271
6	$(10010\boxed{0}10)^t$	22	14
7	$(100101\boxed{0}0)^t$	23	11
8	$(1001011\boxed{1})^t$	43	-381

O atributo #1 é, então, adicionado à lista tabu, passando a estar tabu-ativo.

$$T = \{\langle 1 \rangle\}$$



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^1 = (00010110)^t$$

$$f(s^1) = 17$$

Tabu:

#	$s' \in N(s^1)$	$\text{peso}(s')$	$f(s')$
1	$(\mathbf{1}0010110)^t$	32	19
2	$(0\mathbf{1}010110)^t$	43	-388
3	$(00\mathbf{1}10110)^t$	35	-91
4	$(000\mathbf{0}0110)^t$	19	13
5	$(0001\mathbf{1}110)^t$	36	-125
6	$(00010\mathbf{0}10)^t$	18	12
7	$(000101\mathbf{0}0)^t$	19	9
8	$(0001011\mathbf{1})^t$	39	-235

Mover para o melhor vizinho da solução  $s^0$ , isto é, para  $s^1$ .

Nesta nova solução, seu vizinho #1 é tabu porque ele tem um atributo tabu-ativo (a posição #1 está na lista tabu)

$$T = \{\langle 1 \rangle\}$$



# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^1 = (00010110)^t$$

$$f(s^1) = 17$$

Tabu:

#	$s' \in N(s^1)$	$\text{peso}(s')$	$f(s')$
1	$(\mathbf{1}0010110)^t$	32	19
2	$(0\mathbf{1}010110)^t$	43	-388
3	$(00\mathbf{1}10110)^t$	35	-91
4	$(000\mathbf{0}0110)^t$	19	$\mathbf{13}$
5	$(0001\mathbf{1}110)^t$	36	-125
6	$(00010\mathbf{0}10)^t$	18	12
7	$(000101\mathbf{0}0)^t$	19	9
8	$(0001011\mathbf{1})^t$	39	-235

O melhor vizinho não-tabu da solução atual  $s^1$  é o vizinho #4, no qual a posição #4 foi modificada, então esse atributo é adicionado à lista tabu.

$$T = \{\langle 1 \rangle\} \cup \{\langle 4 \rangle\}$$

$$T = \{\langle 1 \rangle, \langle 4 \rangle\}$$





# Problema da Mochila

Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^2 = (00000110)$$

$$f(s^2) = 13$$

Tabu:

Tabu:

#	$s' \in N(s^2)$	$\text{peso}(s')$	$f(s')$
1	$(10000110)^t$	23	15
2	$(01000110)^t$	34	-59
3	$(00100110)^t$	26	16
4	$(00010110)^t$	28	17
5	$(00001110)^t$	27	19
6	$(00000010)^t$	9	8
7	$(00000100)^t$	10	5
8	$(00000111)^t$	30	20

Mover para o melhor vizinho não-tabu da solução  $s^1$ , isto é, para  $s^2$ .

Os vizinhos #1 e #4 de  $s^2$  são tabus porque eles contêm atributos tabu-ativos (posições 1 e 4).

$$T = \{\langle 1 \rangle, \langle 4 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^2 = (00000110)$$

$$f(s^2) = 13$$

Tabu:

Tabu:

#	$s' \in N(s^2)$	$\text{peso}(s')$	$f(s')$
1	$(10000110)^t$	23	15
2	$(01000110)^t$	34	-59
3	$(00100110)^t$	26	16
4	$(00010110)^t$	28	17
5	$(00001110)^t$	27	19
6	$(00000010)^t$	9	8
7	$(00000100)^t$	10	5
8	$(00000111)^t$	30	20

Melhor vizinho não-tabu da solução atual = vizinho #8 (gerado pela alteração da posição #8). Esse atributo é adicionado à lista tabu.

Como  $|T|=2$ , então pela estratégia FIFO, o atributo tabu-ativo 1 deixa de ser ativo e entra o atributo 8 ao final da lista tabu.

$$T = \{\langle 1 \rangle, \langle 4 \rangle\} \setminus \{\langle 1 \rangle\} \cup \{\langle 8 \rangle\}$$

$$T = \{\langle 4 \rangle, \langle 8 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^3 = (00000111)$$

$$f(s^3) = 20$$

Tabu:

#	$s' \in N(s^3)$	$peso(s')$	$f(s')$
1	$(10000111)^t$	34	-52
2	$(01000111)^t$	45	-459
3	$(00100111)^t$	37	-162
4	$(00010111)^t$	39	-235
5	$(00001111)^t$	38	-196
6	$(00000011)^t$	20	15
7	$(00000101)^t$	21	12
8	$(00000110)^t$	19	13

Tabu:

Mover para o melhor vizinho não-tabu da solução  $s^2$ , i.e, para  $s^3$ .

Os vizinhos #4 e #8 de  $s^3$  são tabus

O vizinho #6 (não-tabu) é o melhor. O atributo 6 entra e sai o 4 da lista

$$T = \{<4>, <8>\} \setminus \{<4>\} \cup \{<6>\}$$

$$T = \{\langle 8 \rangle, \langle 6 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^4 = (00000011)$$

$$f(s^4) = 15$$

#	$s' \in N(s^4)$	$\text{peso}(s')$	$f(s')$
1	$(10000011)^t$	24	17
2	$(01000011)^t$	35	-94
3	$(00100011)^t$	27	18
4	$(00010011)^t$	29	19
5	$(00001011)^t$	28	21
6	$(00000111)^t$	30	20
7	$(00000001)^t$	11	7
8	$(00000010)^t$	9	8

Tabu:

Tabu:

Mover para o melhor vizinho não-tabu da solução  $s^3$ , i.e., para  $s^4$ .

Os vizinhos #6 e #8 de  $s^4$  são tabus

O vizinho #5 (não-tabu) é o melhor. O atributo 5 entra e sai o 8 da lista

$$T = \{\langle 8 \rangle, \langle 6 \rangle\} \setminus \{\langle 8 \rangle\} \cup \{\langle 5 \rangle\}$$

$$T = \{\langle 6 \rangle, \langle 5 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^5 = (00001011)$$

$$f(s^5) = 21$$

Tabu:

Tabu:

#	$s' \in N(s^5)$	$\text{peso}(s')$	$f(s')$
1	$(10001011)^t$	32	23
2	$(01001011)^t$	43	-384
3	$(00101011)^t$	35	-87
4	$(00011011)^t$	37	-160
5	$(00000011)^t$	20	15
6	$(00001111)^t$	38	-196
7	$(00001001)^t$	19	13
8	$(00001010)^t$	17	14

Mover para o melhor vizinho não-tabu da solução  $s^4$ , i.e, para  $s^5$ .

Os vizinhos #5 e #6 de  $s^5$  são tabus

O vizinho #1 (não-tabu) é o melhor. O atributo 1 entra e sai o 6 da lista

$$T = \{\langle 6 \rangle, \langle 5 \rangle\} \setminus \{\langle 6 \rangle\} \cup \{\langle 1 \rangle\}$$

$$T = \{\langle 5 \rangle, \langle 1 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^6 = (10001011)$$

$$f(s^6) = 23$$

Tabu:

Tabu:

#	$s' \in N(s^6)$	$\text{peso}(s')$	$f(s')$
1	$(00001011)^t$	28	21
2	$(11001011)^t$	47	-530
3	$(10101011)^t$	39	-233
4	$(10011011)^t$	41	-306
5	$(10000011)^t$	24	17
6	$(10001111)^t$	42	-342
7	$(10001001)^t$	23	15
8	$(10001010)^t$	21	16

Mover para o melhor vizinho não-tabu da solução  $s^5$ , i.e, para  $s^6$ .

Os vizinhos #1 e #5 de  $s^6$  são tabus

O vizinho #8 (não-tabu) é o melhor. O atributo 8 entra e sai o 5 da lista

$$T = \{\langle 5 \rangle, \langle 1 \rangle\} \setminus \{\langle 5 \rangle\} \cup \{\langle 8 \rangle\}$$

$$T = \{\langle 1 \rangle, \langle 8 \rangle\}$$



# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^7 = (10001010)$$

$$f(s^7) = 16$$

Tabu:

#	$s' \in N(s^7)$	$\text{peso}(s')$	$f(s')$
1	$(00001010)^t$	17	14
2	$(11001010)^t$	36	-130
3	$(10101010)^t$	28	19
4	$(10011010)^t$	30	20
5	$(10000010)^t$	13	10
6	$(10001110)^t$	31	21
7	$(10001000)^t$	12	8
8	$(10001011)^t$	32	23

Tabu:

Mover para o melhor vizinho não-tabu da solução  $s^6$ , i.e, para  $s^7$ .

Os vizinhos #1 e #8 de  $s^7$  são tabus

O vizinho #6 (não-tabu) é o melhor. O atributo 6 entra e sai o 1 da lista

$$T = \{\langle 1 \rangle, \langle 8 \rangle\} \setminus \{\langle 1 \rangle\} \cup \{\langle 6 \rangle\}$$

$$T = \{\langle 8 \rangle, \langle 6 \rangle\}$$

# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^8 = (10001110)$$

$$f(s^8) = 21$$

#	$s' \in N(s^8)$	$\text{peso}(s')$	$f(s')$
1	$(00001110)^t$	27	19
2	$(11001110)^t$	46	-495
3	$(10101110)^t$	38	-198
4	$(10011110)^t$	40	-271
5	$(10000110)^t$	23	15
Tabu: 6	$(10001010)^t$	21	16
7	$(10001100)^t$	22	13
Tabu: 8	$(10001111)^t$	42	-342

Mover para o melhor vizinho não-tabu da solução  $s^7$ , i.e, para  $s^8$ .

Os vizinhos #6 e #8 de  $s^8$  são tabus

O vizinho #1 (não-tabu) é o melhor. O atributo 6 entra e sai o 1 da lista

$$T = \{\langle 8 \rangle, \langle 6 \rangle\} \setminus \{\langle 8 \rangle\} \cup \{\langle 1 \rangle\}$$

$$T = \{\langle 6 \rangle, \langle 1 \rangle\}$$



# Problema da Mochila



Tabela 1.1: Instância do Problema da Mochila 0-1

Item $j$	1	2	3	4	5	6	7	8
Peso $w_j$	4	15	7	9	8	10	9	11
Retorno $p_j$	2	2	3	4	6	5	8	7

$$s^9 = (00001110)$$

$$f(s^9) = 19$$

Tabu:

#	$s' \in N(s^9)$	$\text{peso}(s')$	$f(s')$
1	$(10001110)^t$	31	21
2	$(01001110)^t$	42	-349
3	$(00101110)^t$	34	-52
4	$(00011110)^t$	36	-125
5	$(00000110)^t$	19	13
6	$(00001010)^t$	17	14
7	$(00001100)^t$	18	11
8	$(00001111)^t$	38	-196

Tabu:

Mover para o melhor vizinho não-tabu da solução  $s^8$ , i.e, para  $s^9$ .

Os vizinhos #1 e #6 de  $s^9$  são tabus

O vizinho #5 (não-tabu) é o melhor. O atributo 5 entra e sai o 1 da lista

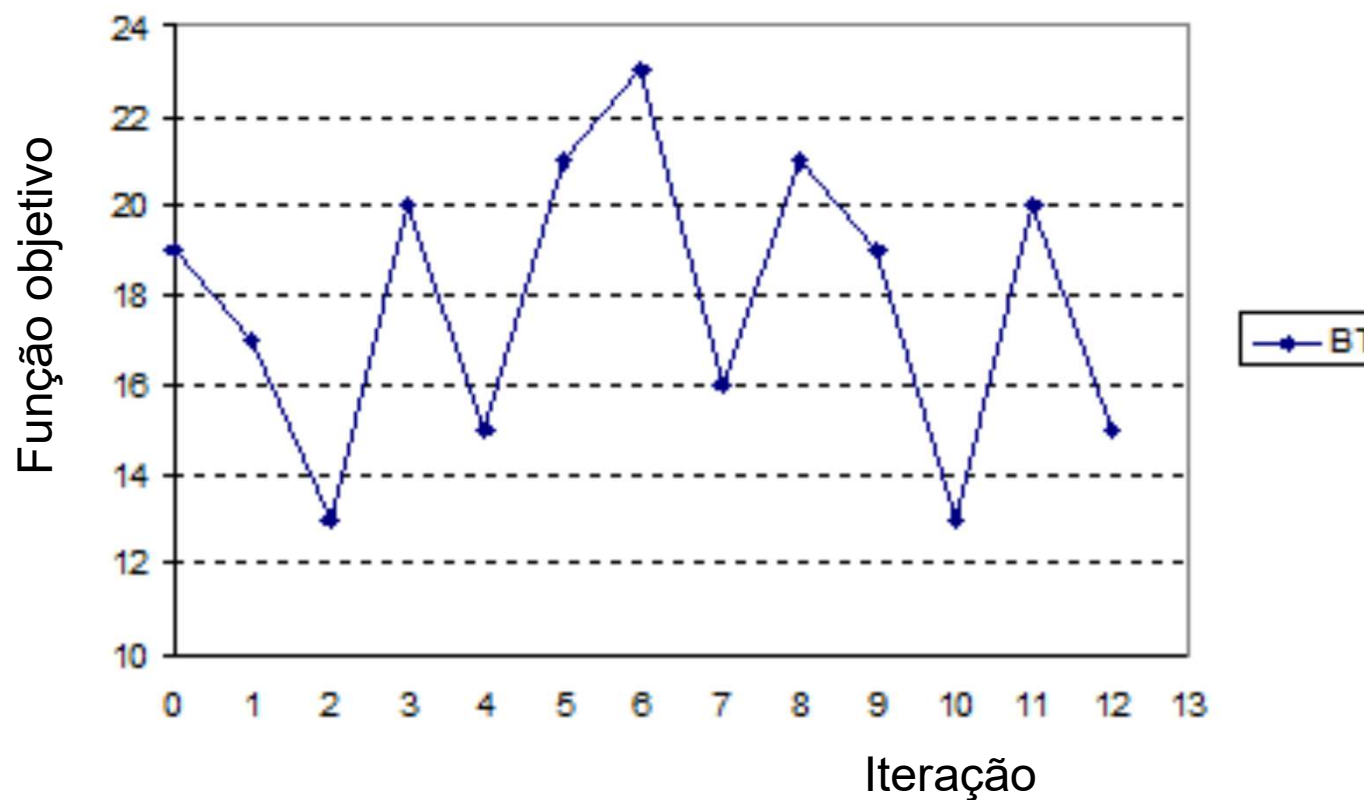
$$T = \{\langle 6 \rangle, \langle 1 \rangle\} \setminus \{\langle 6 \rangle\} \cup \{\langle 5 \rangle\}$$

$$T = \{\langle 1 \rangle, \langle 5 \rangle\}$$



# Problema da Mochila

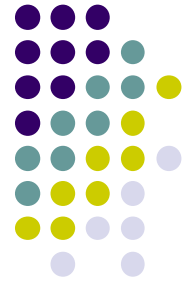
Evolução do valor da função objetivo ao longo das iterações





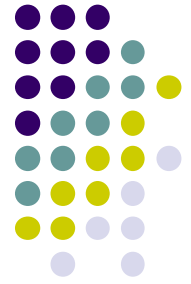
# Implementação da Lista Tabu

- Verificar se um movimento é ou não tabu pode ser dispendioso
- No PM,  $T = \{ \langle j_1 \rangle, \langle j_2 \rangle, \dots, \langle j_{|T|} \rangle \}$
- Considerando vetor de  $n$  posições e  $|T|$  elementos na lista tabu:
  - Pior caso:  $O(n \cdot |T|)$  avaliações, por iteração



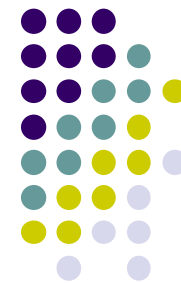
# Implementação da Lista Tabu

- **Ideia:** Substituir a lista  $T$  por um vetor de  $n$  posições, em que cada posição  $j$  armazene a iteração até a qual o atributo estará ativo
- $T$ : vetor de prazo tabu (*tabu tenure*)
- Ex.: Na primeira iteração do PM (iter=1), considerando DuraçãoTabu=2, então:
  - $T=\{<1>\}$  é substituída por  $T=(3,0,0,0,0,0,0,0)$ , sendo:
  - $3 = \text{iter} + \text{DuraçãoTabu} = 1 + 2$
- Significado:
  - O movimento de inverter o bit da primeira posição está tabu-ativo até a iteração 3. Após a terceira iteração, ele deixa de ser tabu-ativo.



# Implementação da Lista Tabu

- Com um vetor  $T$  de prazo tabu, a verificação se um movimento é tabu ou não é bastante simples.
- Seja  $iter$  a iteração atual. Então o movimento de inverter o valor do bit da posição  $j$  é tabu se tivermos:  $T(j) \geq iter$
- Ex.: Dado  $T = (3, 0, 8, 5, 9, 10, 7, 11)$  e  $iter = 10$ , então são tabus os movimentos envolvendo as posições 6 e 8.
- Complexidade de verificação se um movimento está ou não na “lista”:  $O(1)$



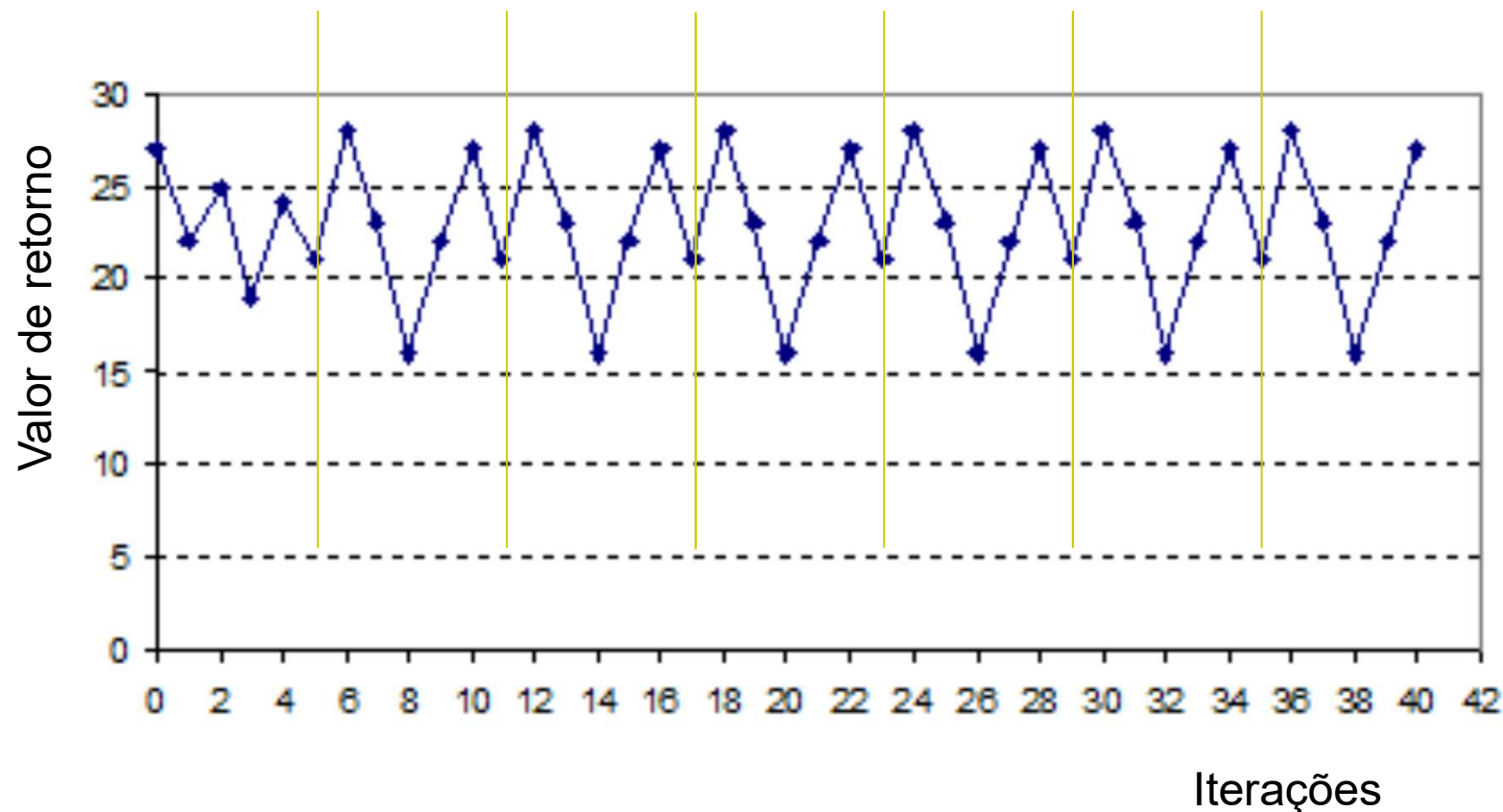
# Tamanho da lista

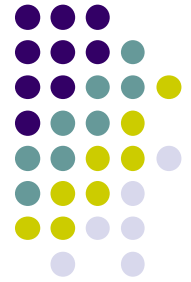
- O tamanho da lista ou a duração tabu do atributo tabu-ativo indica a quantidade máxima de iterações em que cada movimento deve permanecer tabu para cumprir seu papel de impedir o retorno a uma solução já visitada anteriormente.
- A duração tabu depende:
  - Do tipo de atributo considerado:
    - Regras de ativação mais restritivas têm duração menor
  - Do tamanho da instância
    - Quanto maior o tamanho, maior a lista (crescimento não-linear)
  - Do estágio da busca
    - Em problemas de programação de horários, p.ex., vale a pena aumentar o tamanho da lista quando a busca se encontrar numa região com soluções de igual valor da função objetivo



# Tamanho da lista

- Duração tabu muito pequena:
  - Pode causar ciclagem.





# Tamanho da lista

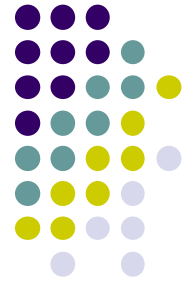
- Duração tabu muito grande:
  - Pode provocar deterioração na qualidade das soluções finais
- A duração tabu pode ser:
  - **Fixa:** a duração não muda durante a busca
  - **Dinâmica:** a duração é definida em um intervalo  $[t_{\min}, t_{\max}]$





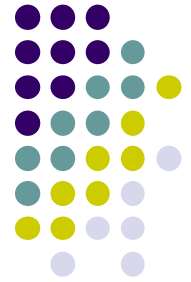
# Tamanho da lista

- A duração tabu dinâmica pode ser:
  - Aleatória: a duração é selecionada no intervalo  $[t_{\min}, t_{\max}]$ 
    - Mantém-se fixa a cada  $\alpha \times t_{\max}$  iterações ou
    - Sorteia-se uma nova duração a cada iteração
  - Sistemática: depende do problema.
    - Exemplo: aumentar e diminuir alternadamente a duração dentro do intervalo
    - $t_{\min} = 6$ ;  $t_{\max} = 11$
    - Sequência:  $\{6, 9, 8, 11, 7, 10\}$



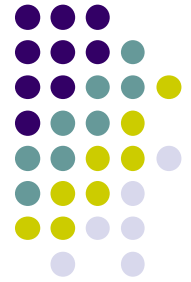
# Tamanho da lista

- Duração tabu 'mágica': 7
- Duração tabu dependente da instância:
  - Osman (1993): PRV;  $n$  consumidores,  $v$  veículos
  - $|T| = 8 + (0,078 - 0,067 \times \rho) \times (n \times v)$
  - $\rho = \sum(\text{demandas consumidores}) / (\text{cap. veículos})$
  - $|T| = \max\{7, -40 + 0,6 \times \ln(n \times v)\}$
  - Três possibilidades para a duração:
    - $t_{\min} = 0,9 \times |T|$
    - $|T|$
    - $T_{\max} = 1,1 \times |T|$
  - A duração  $t$  escolhida é mantida fixa por  $2 \times t$  iterações



# Tamanho da lista

- Vantagem de variar a duração tabu ao longo da busca:
  - Corrigir o erro porventura existente no dimensionamento empírico desse tempo.
  - Havendo ciclagem com uma duração  $|T|$ , então variando-a, haverá alteração na quantidade de movimentos tabus e, assim, diferentes soluções poderão ser geradas.
  - Com essa possibilidade de mudança na trajetória da busca no espaço de soluções, a ocorrência de ciclagem fica reduzida



# Tamanho da lista

- A duração tabu depende do atributo e da regra de proibição.
- No PM, considerando que:
  - A cada iteração uma posição é tornada tabu-ativa;
  - Ao final de  $n-1$  iterações há apenas um movimento possível.
  - Assim, o limitante superior para a duração tabu de um movimento é  $n-1$ .



# Tamanho da lista

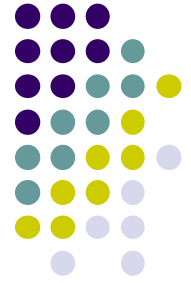
- Em um problema de sequenciamento de  $n$  tarefas:
  - uma regra que proíbe uma tarefa  $i$  de ser movida em um movimento de troca
  - reduz para  $n-1$  o número de tarefas que podem ser trocadas após sua primeira aplicação.
- Considerando a proibição de uma tarefa a cada iteração, e que para realização do movimento de troca há necessidade de pelo menos duas tarefas liberadas,
  - a última troca possível se realizará na iteração  $n-1$ .
- Desta forma,  $n-1$  é o limitante superior para a duração desta regra de proibição.



# Tamanho da lista

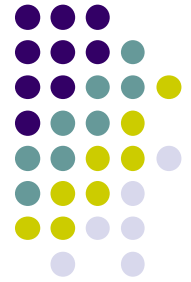
- Estudo de limitantes para o problema de sequenciamento de  $n$  tarefas (Ronconi e Armentano, 2009):

Movimento	Atributo	Regra	Limitante inferior	Limitante superior
Inserção	$i$	$i$ não pode ser movida	$n/4$	$n/2$
	$i$	$i$ não pode ser escolhido para inserção	$n/2$	$n$
	$i, s(i)$	$i$ não pode retornar à posição $s(i)$	$4n$	$5n$
Troca	$i, j$	$i$ e $j$ não podem ser movidas	$n/4$	$n/2$
	$i$	$i$ não pode ser movida	$n/2$	$n - 1$
	$mi$	$i$ não pode ser movida para posições $k \leq s(i)$	$n/2$	$3n/2$



# Análise da vizinhança

- Em muitos problemas:
  - Muito custoso avaliar a vizinhança completa
- Lista de candidatos:
  - Avaliar apenas soluções que satisfaçam a determinados critérios



# Análise da vizinhança

- Programação de tarefas para minimizar o atraso total
- Tarefas: 1, 2, ..., n
- Dados:
  - $p_j$  = tempo de processamento da tarefa  $j$
  - $d_j$  = data de entrega da tarefa  $j$
- Variável de decisão
  - $C_j$  = instante de término de processamento da tarefa  $j$
- Função objetivo:

$$\min t = \sum_{j=1}^n \max \{0, C_j - d_j\}$$





# Análise da vizinhança

- Programação de tarefas em uma máquina para minimizar o atraso total
  - Movimento: trocar duas tarefas de posição
  - Tamanho da vizinhança:  $n(n-1)/2$
  - Complexidade da melhor troca:  $O(n^2)$
  - Valor do movimento:  $f(s') - f(s)$
- Exemplo: Seja um problema com 6 tarefas
  - Datas de entrega: (9, 12, 15, 8, 20, 22)
  - Tempos de processamento: (6, 4, 8, 2, 10, 3)
  - Sequência inicial:  $s = (1, 2, 3, 4, 5, 6)$
  - Seja  $t_j$  o atraso da tarefa  $j$ , dado por  $t_j = \max\{0, C_j - d_j\}$



# Análise da vizinhança

- Exemplo: Seja um problema com 6 tarefas
  - Datas de entrega:  $d = (9, 12, 15, 8, 20, 22)$
  - Tempos de processamento:  $p = (6, 4, 8, 2, 10, 3)$
  - Sequência inicial:  $s = (1, 2, 3, 4, 5, 6)$
- Cálculo das datas de conclusão ( $C_j$ ) e atrasos ( $t_j$ ):
  - $C_1 = p_1 = 6$
  - $t_1 = \max\{0, C_1 - d_1\} = \max\{0, 6 - 9\} = \max\{0, -3\} = 0$
  - $C_2 = C_1 + p_2 = 6 + 4 = 10$
  - $t_2 = \max\{0, C_2 - d_2\} = \max\{0, 10 - 12\} = \max\{0, -2\} = 0$
  - $C_3 = C_2 + p_3 = 10 + 8 = 18$
  - $t_3 = \max\{0, C_3 - d_3\} = \max\{0, 18 - 15\} = \max\{0, 3\} = 3$
  - $C_4 = C_3 + p_4 = 18 + 2 = 20$
  - $t_4 = \max\{0, C_4 - d_4\} = \max\{0, 20 - 8\} = \max\{0, 12\} = 12$
  - $C_5 = C_4 + p_5 = 20 + 10 = 30$
  - $t_5 = \max\{0, C_5 - d_5\} = \max\{0, 30 - 20\} = \max\{0, 10\} = 10$
  - $C_6 = C_5 + p_6 = 30 + 3 = 33$
  - $t_6 = \max\{0, C_6 - d_6\} = \max\{0, 33 - 22\} = \max\{0, 11\} = 11$
- Atraso total  $t = t_1 + t_2 + t_3 + t_4 + t_5 + t_6 = 0 + 0 + 3 + 12 + 10 + 11 = 36$



# Análise da vizinhança

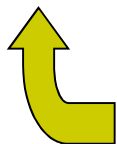
- Supor que para cada tarefa  $i$  da solução, somente as tarefas  $j$  que diferem de 3 unidades com relação à data de entrega sejam candidatas a serem trocadas de posição, isto é, que elas satisfaçam à condição:
  - $|d_i - d_j| \leq 3$
- Para o exemplo citado:
  - Datas de entrega  $d = (9, 12, 15, 8, 20, 22)$
  - Sequência inicial:  $s = (1, 2, 3, 4, 5, 6)$
  - Lista de candidatos =  $\{<1,2>, <1,4>, <2,3>, <5,6>\}$
  - $|d_1 - d_2| = |9 - 12| = 3 \leq 3$
  - $|d_1 - d_4| = |9 - 8| = 1 \leq 3$
  - $|d_2 - d_3| = |12 - 15| = 3 \leq 3$
  - $|d_5 - d_6| = |20 - 22| = 2 \leq 3$

# Análise da vizinhança



Tarefas		t	Valor do movimento	$ d_i - d_j $	
i	j				
1	2	37	1	3	*
1	3	42	6	6	
1	4	32	-4	1	*
1	5	57	21	11	
1	6	40	4	13	
2	3	39	3	3	*
2	4	30	-6	4	
2	5	56	20	8	
2	6	43	7	10	
3	4	30	-6	7	
3	5	40	4	5	
3	6	30	-6	7	
4	5	44	8	12	
4	6	39	3	14	
5	6	29	-7	2	*

Melhor troca



# Análise da vizinhança

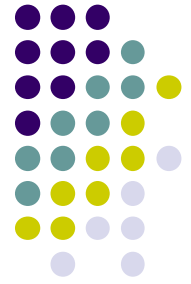


- Programação de tarefas para minimizar o atraso total
  - Tamanho da vizinhança completa: 15
  - Tamanho da vizinhança com Lista de candidatos:
    - 4
    - Contém a melhor troca (5, 6)
- Outras estratégias para redução da vizinhança:
  - Primeira melhora
  - Percentual da vizinhança:
    - Em programação de tripulações, p.e., trabalhar a cada iteração somente com x% de tripulantes
    - Em alocação de aulas a salas, p.ex., trabalhar em uma iteração com a segunda-feira; na outra, com a terça e assim por diante

# Outros critérios de aspiração

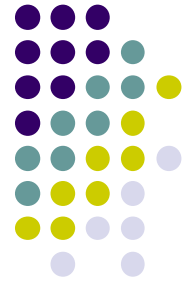


- Critério de aspiração por objetivo regional:
  - Um movimento tabu perde seu status quando for gerada uma solução melhor que a melhor encontrada na região atual de busca.
  - Forma de se delimitar a região atual de busca:
    - registrar a melhor solução encontrada em um passado recente e utilizar o valor dessa solução como critério para aspiração.
  - Exemplo:
    - Seja um problema de minimização e  $f(s^*_R)$  a melhor solução encontrada nas últimas *ContIterRegiao* iterações
    - Um movimento tabu que guia a uma solução  $s'$  tal que  $f(s') < f(s^*_R)$  pode ser realizado
    - Nesta estratégia de aspiração, *ContIterRegiao* é um parâmetro.



# Outros critérios de aspiração

- Critério de aspiração *default*:
  - Se todos os movimentos possíveis são tabus e não é possível aplicar outro critério de aspiração, então o movimento mais antigo perde sua condição tabu.
  - Implementação baseada em tempo de permanência na lista, como o do Problema da Mochila:
    - se  $T(i) \geq \text{iter}$ , para todo  $i$ , em que  $\text{iter}$  representa a iteração atual,
    - então a inversão do bit da posição  $k$  tal que:
      - $T(k) = \min \{T(i) \text{ para todo } i\}$  pode ser realizada
    - Em outras palavras, se todos os movimentos estão proibidos, então o movimento tabu há mais tempo na “lista” é realizado.



# Memória de Longo Prazo

- Uma memória de curto prazo não é suficiente para evitar que a busca fique presa em certas regiões do espaço de soluções.
- A memória de longo prazo pode ser usada com dois objetivos:
  - encorajar o processo de busca a explorar regiões ainda não visitadas: estratégia de diversificação
  - estimular a geração de soluções que contenham atributos encontrados nas soluções já visitadas que sejam historicamente bons: estratégia de intensificação



# Memória de Longo Prazo



- Diversificação com memória × reinicialização
- Vantagem da diversificação com memória:
  - Diminui-se o risco de voltar a visitar uma mesma região do espaço de soluções,  
⇒ situação que poderia ocorrer se fosse feita uma simples reinicialização

# Memória de Longo Prazo

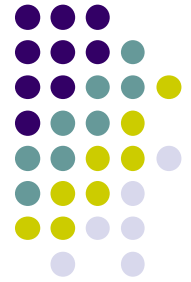


- Tipos de memória de longo prazo:
  - Frequência de transição: computam-se atributos que mudam de uma solução para outra
  - Frequência de residência: computam-se os atributos que são frequentes nas soluções visitadas para usá-los como:
    - estratégia de intensificação (estimulando-os a comporem a solução corrente) e/ou como
    - estratégia para diversificar a busca (neste caso, penalizando-os para desestimular seu uso).

# Memória de Longo Prazo



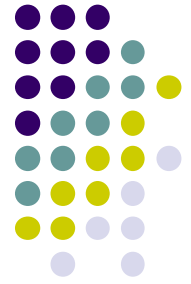
- Aplicação: Problema Generalizado de Atribuição (PGA):
  - Considere  $n$  tarefas  $J = \{1, 2, \dots, n\}$  a serem executadas
  - Existem  $m$  agentes  $I = \{1, 2, \dots, m\}$  para executar as tarefas
  - A atribuição do agente  $i$  à tarefa  $j$ , tem um custo  $c_{ij}$  e demanda  $a_{ij}$  unidades de recurso do agente  $i$
  - Cada agente possui um limite  $b_i$  de recursos disponíveis
  - Cada tarefa pode ser atribuída a apenas um agente;
  - Objetivo: determinar o custo mínimo de alocação dos agentes;



# Memória de Longo Prazo

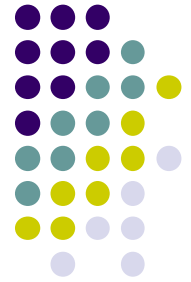
- Formulação matemática do (PGA):

$$\begin{aligned} \min \quad & z = \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \\ \text{s. a:} \quad & \sum_{i \in I} x_{ij} = 1 \quad \forall j \in J \\ & \sum_{j \in J} a_{ij} x_{ij} \leq b_i \quad \forall i \in I \\ & x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J \end{aligned}$$



# Memória de Longo Prazo

- Aplicação: Problema Generalizado de Atribuição (PGA):
  - Representação de uma solução:
    - Vetor de  $n$  posições
    - $s = (s_1, s_2, \dots, s_j, \dots, s_n)$
    - Em cada posição  $j$  armazena-se o agente  $i=s_j$  responsável pela execução da tarefa  $j$ , isto é:
      - $s_j = i \Leftrightarrow x_{ij} = 1$
  - Exemplo: Sejam 5 tarefas  $J = \{1, 2, 3, 4, 5\}$  e 3 agentes  $I = \{A, B, C\}$ 
    - $s = (B, B, A, C, A)$



# Memória de Longo Prazo

- Estruturas de vizinhança para o PGA:
  - Movimentos de substituição:  $N^{(S)}(s)$
  - Movimentos de troca:  $N^{(T)}(s)$

<i>Tarefa</i>	<table><tr><td><i>B</i></td><td><i>B</i></td><td><i>A</i></td><td><i>C</i></td><td><i>A</i></td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table>	<i>B</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>A</i>	1	2	3	4	5	Solução <i>s</i>
<i>B</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>A</i>								
1	2	3	4	5								
<i>Tarefa</i>	<table><tr><td><i>C</i></td><td><i>B</i></td><td><i>A</i></td><td><i>C</i></td><td><i>A</i></td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table>	<i>C</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>A</i>	1	2	3	4	5	(a) Vizinhança $N^{S'}(s)$
<i>C</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>A</i>								
1	2	3	4	5								
<i>Tarefa</i>	<table><tr><td><i>A</i></td><td><i>B</i></td><td><i>A</i></td><td><i>C</i></td><td><i>B</i></td></tr><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table>	<i>A</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>B</i>	1	2	3	4	5	(b) Vizinhança $N^T(s)$
<i>A</i>	<i>B</i>	<i>A</i>	<i>C</i>	<i>B</i>								
1	2	3	4	5								

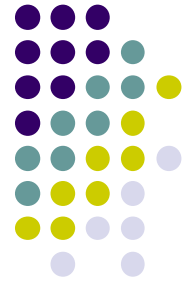
# Memória de Longo Prazo



- Se a tarefa  $j$  de um agente  $i_1$  é atribuída ao agente  $i_2$ ,
  - então o par  $(i_1, j)$  torna-se tabu.
- Função de avaliação para o PGA:
  - Baseada em penalidade

$$f(x) = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \rho \times \sum_{i=1}^m \max \left\{ 0, \sum_{j=1}^n a_{ij} x_{ij} - b_i \right\}$$

- $\rho$ : fator de penalidade associada à violação da capacidade dos agentes



# Memória de Longo Prazo

- Memória baseada em frequência de residência
- Matriz  $fr$  de  $m$  linhas e  $n$  colunas
- $fr(i,j)$  = número de vezes que a tarefa  $j$  ficou alocada ao agente  $i = s_j$ , considerando o total de soluções visitadas até então
- $\Delta$  = matriz de custo



# Algoritmo BT-PGA de Díaz e Fernández (2006)



---

## Algorithm 2 BT-PGA

---

```
1: Entrada:  $s_0, l$ 
2: Saída:  $s^*$ 
3:  $T \leftarrow 0$ ;  $fr \leftarrow 0$ ;  $s^* \leftarrow s_0$ ;  $k \leftarrow 0$ ;
4: Gere solução inicial  $s_0$ ;
5: Aplique MemoriaCurtoPrazo( $k, s^k, s^*, T, fr, \Delta$ );
6: para todo  $iter = 1$  até  $l$  faça
7:   { Fase de intensificação }
8:    $s^k \leftarrow s^*$ ;
9:   para todo tarefa  $j$  de  $s^*$  faça
10:    se  $fr(s_j^*, j) > 0,85 \times k$  então
11:      Fixe a tarefa  $j$  ao agente  $i$ , isto é, faça  $x_{s_j^*, j} \leftarrow 1$ ;
12:    fim se
13:  fim para
14:  Aplique MemoriaCurtoPrazo( $k, s^k, s^*, T, fr, \Delta$ )
15:  { Fase de diversificação }
16:  Libere todas as atribuições;
17:  Aplique MemoriaCurtoPrazo( $k, s^k, s^*, T, fr, \Delta$ ) por poucas iterações
    usando  $\Delta + fr$  como matriz de custo;
18:  Recupere a matriz de custo  $\Delta$  original;
19:  Aplique MemoriaCurtoPrazo( $k, s^k, s^*, T, fr, \Delta$ );
20: fim para
```

---

# Algoritmo BT-PGA de Díaz e Fernández (2006)



---

Algorithm 4 MemoriaCurtoPrazo( $k, s^k, s^*, T, fr, \Delta$ )

---

```
1: while Critério de parada não atingido do
2:   Selecione um vizinho  $s^{k+1} \in N(s^k)$ ;
3:   if  $custo(s^{k+1}) < custo(s^*)$  then
4:      $s^* \leftarrow s^{k+1}$ ;
5:   end if
6:   Atualize a memória de curto prazo  $T$ ;
7:   Atualize a memória de longo prazo  $fr$ ;
8:    $k \leftarrow k + 1$ ;
9: end while
```

---



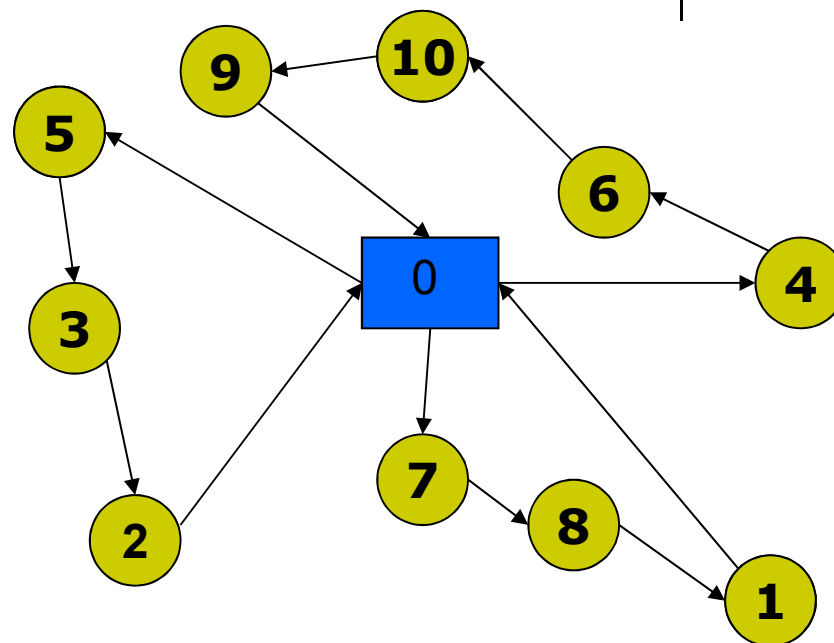
# Oscilação estratégica

- Ideia: Variar as penalizações atribuídas às soluções inviáveis geradas durante o processo de busca, ora aumentando-as, ora diminuindo-as
- Ao aumentar a penalização de movimentos que conduzem a soluções infactíveis, estimula-se a busca a entrar na região de soluções factíveis.
- Ao contrário, quando tal penalização é reduzida ou mesmo anulada, estimula-se a caminhar no espaço de soluções infactíveis.
- Portanto, o uso desta estratégia permite à busca alternar entre soluções factíveis e infactíveis, daí o nome ``oscilação''

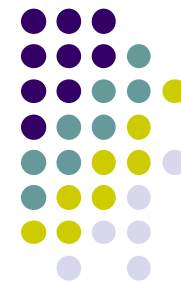


# Oscilação estratégica

- Em Gendreau *et al.* (1994), os autores tratam o Problema de Roteamento de Veículos penalizando o excesso de carga encontrado em uma solução:
- $f_1(x)$ : sobrecarga dos veículos
- $\rho$ : penalidade
- $\rho \leftarrow \alpha \times \rho$



$$f(x) = \sum_{i=0}^n \sum_{j=0}^n d_{ij} x_{ij} + \rho \times f_1(x)$$



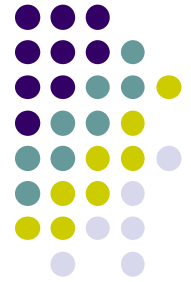
# Oscilação estratégica

- O fator  $\alpha$  varia de acordo com o seguinte esquema:
  - No início da busca  $\alpha \leftarrow 1$ .
  - A cada  $k$  iterações sem melhora:
    - se todas as  $k$  soluções visitadas são factíveis então  $\alpha \leftarrow \alpha / \gamma$ ;
    - se todas as  $k$  soluções visitadas são infactíveis então  $\alpha \leftarrow \alpha \times \gamma$ ;
    - se algumas soluções são factíveis e algumas outras são infactíveis, então  $\alpha$  permanece inalterado.
- $\gamma = 2, k = 10, \alpha \in [\alpha_{\min}, \alpha_{\max}]$ ;
- Durante a oscilação estratégica, a busca é guiada por uma função auxiliar  $g$  em lugar da função de avaliação original  $f$



# Oscilação estratégica

- Em Shaefer (1996), foi tratado um problema de programação de horários em escolas por BT.
- Há várias fontes de inviabilidade. Para cada uma delas aplica-se um fator  $\alpha_i$  que varia de acordo com o seguinte esquema:
  - No início da busca  $\alpha_i \leftarrow 1$ .
  - A cada  $k$  iterações sem melhora:
    - se todas as  $k$  soluções visitadas são factíveis com relação à infactibilidade  $i$  então  $\alpha_i \leftarrow \alpha_i / \gamma$ ;
    - se todas as  $k$  soluções visitadas são infactíveis com relação à infactibilidade  $i$  então  $\alpha_i \leftarrow \alpha_i \times \gamma$ ;
    - se algumas soluções são factíveis e algumas outras são infactíveis, então  $\alpha_i$  permanece inalterado.
- $\gamma \in [1,8; 2]$ ,  $k = 10$ ,  $\alpha_i \in [\alpha_i^{\min}, \alpha_i^{\max}]$ ;
- $\alpha_i^{\min} = 1$ ;  $\alpha_i^{\max} \in \{1, 3, 10\}$



# Oscilação estratégica

- Em Díaz e Fernández (2006), a oscilação estratégica foi incorporada ao algoritmo BT para resolver o PGA. A penalidade  $\rho$  por ocorrer uma solução inviável é atualizada como segue:

$$\rho \leftarrow \rho \times \alpha^{(ninv/(niter-1)-1)}$$

- $ninv$  = número de soluções inviáveis geradas nas últimas  $niter$  iterações
- $\rho \in [\alpha^{-1} \times \rho, \alpha^{1/(niter-1)} \times \rho]$
- $\rho$  aumenta de valor somente quando todas as soluções são inviáveis
- No início:  $\alpha = 1$ ;  $\rho = 1$ ;
- Após ser encontrada uma solução viável:  $\alpha = 2$ ;
- Quando a melhor solução não é alterada há 100 iterações,  $\alpha \leftarrow \alpha + 0,005$  a cada 10 iterações até que uma nova melhor solução seja encontrada (neste caso,  $\alpha$  é reiniciado com valor 2);
- $\alpha \in [2, 3]$ ;



# Referências

- Allahverdi, A., Ng, C. T., Cheng, T. C. E. and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs, *European Journal of Operational Research* **187**: 985–1032.
- Archetti, C., Speranza, M. G. and Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem, *Transportation Science* **40**: 64–73.
- Battiti, R. (1996). Reactive search: Toward self-tuning heuristics, in V. Rayward-Smith, I. Osman, C. Reeves and G. Smith (eds), *Modern Heuristic Search Methods*, John Wiley & Sons, New York, chapter 4, pp. 61–83.
- Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search, *ORSA Journal of Computing* **6**: 126–140.
- Buscher, U. and Shen, L. (2009). An integrated tabu search algorithm for the lot streaming problem in job shops, *European Journal of Operational Research* **199**: 385–399.
- Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y. and Semet, F. (2002). A guide to vehicle routing problem, *Journal of the Operational Research Society* **53**: 512–522.
- de Werra, D. and Hertz, A. (1989). Tabu search techniques: A tutorial and an application to neural networks, *OR Spektrum* **11**: 131–141.





# Referências

- Díaz, J. A. and Fernández (2001). A tabu search heuristic for the generalized assignment problem, *European Journal of Operational Research* **132**: 22–38.
- França, P. M. (2009). Busca tabu. Disponível em <http://www.densis.fee.unicamp.br/franca/EA043/Transpa-Cap-4a.pdf>.
- Gendreau, M. (2002). Recent advances in tabu search, in C. C. Ribeiro and P. Hansen (eds), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, pp. 369–377.
- Gendreau, M. (2003). An introduction to tabu search, in F. Glover and G. A. Kochenberger (eds), *Handbook of Metaheuristics*, Kluwer Academic Publishers, chapter 2, pp. 37–54.
- Gendreau, M., Guertin, F., Potvin, J.-Y. and Taillard, E. D. (1999). Parallel tabu search for real-time vehicle routing and dispatching, *Transportation Science* **33**: 381–390.
- Gendreau, M., Hertz, A. and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science* **40**: 1276–1290.
- Gendreau, M., Laporte, G. and Séguin, R. (1996). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers, *Operations Research* **44**: 469–477.

# Referências



- Gendreau, M., Soriano, P. and Salvail, L. (1993). Solving the maximum clique problem using a tabu search approach, *in* F. Glover, M. Laguna and E. Taillard (eds), *Tabu Search*, Vol. 41 of *Annals of Operations Research*, J. C. Baltzer AG, pp. 385–403.
- Glover, F. (1977). Heuristics for integer programming using surrogate constraints, *Decision Sciences* **8**: 156–166.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research* **5**: 553–549.
- Glover, F. and Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston.
- Glover, F., Taillard, E. and de Werra, D. (1993). A user's guide to tabu search, *in* P. L. Hammer (ed.), *Tabu Search*, Vol. 41 of *Annals of Operations Research*, Baltzer Science Publishers, Amsterdam, pp. 3–28.
- Hansen, P. (1986). The steepest ascent mildest descent heuristic for combinatorial programming, *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.
- Hertz, A. and de Werra, D. (1990). The tabu search metaheuristic: how we used it, *Annals of Mathematics and Artificial Intelligence* **1**: 111–121.

# Referências



- Laguna, M. (1994). A guide to implement tabu search, *Investigación Operativa* **4**: 5–25.
- Osman, I. H. (1993). The vehicle routing problem, in F. Glover, M. Laguna and E. Taillard (eds), *Tabu Search*, Vol. 41 of *Annals of Operations Research*, J. C. Baltzer AG, pp. 421–451.
- Ronconi, D. and Armentano, V. (2009). Busca tabu para a minimização do tempo total de atraso no problema de flowshop, *Pesquisa Operacional para o Desenvolvimento* **1**: 50–62.
- Santos, H. G., Ochi, L. S. and Souza, M. J. F. (2005). A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem, *ACM Journal of Experimental Algorithmics* **10**: art-2.09. 15 p.
- Souza, M. J. F. (2000). *Programação de horários em escolas: uma aproximação por metaheurísticas*, Tese de doutorado, Programa de Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro.
- Souza, M. J. F., Ochi, L. S. and Maculan, N. (2004). A grasp-tabu search algorithm for solving school timetabling problems, in M. G. C. Resende and J. P. Souza (eds), *Metaheuristics: Computer Decision-Making*, Vol. 1153, Kluwer Academic Publishers, pp. 659–672.

# Referências

- Wan, G. and Yen, B. P. C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties, *European Journal of Operational Research* **142**: 271–281.
- Yagiura, M., Glover, F. and Ibaraki, T. (2006). A path relinking approach with ejection chains for the generalized assignment problem, *European Journal of Operational Research* **169**: 548–569.

