

Projeto e Análise de Algoritmos

Aula 20:

Branch-and-bound (Ramificação e Corte)

DECOM/UFOP

2013/2 – 5^o. Período

Anderson Almeida Ferreira

Material elaborado por:

Andréa Iabrudi Tavares

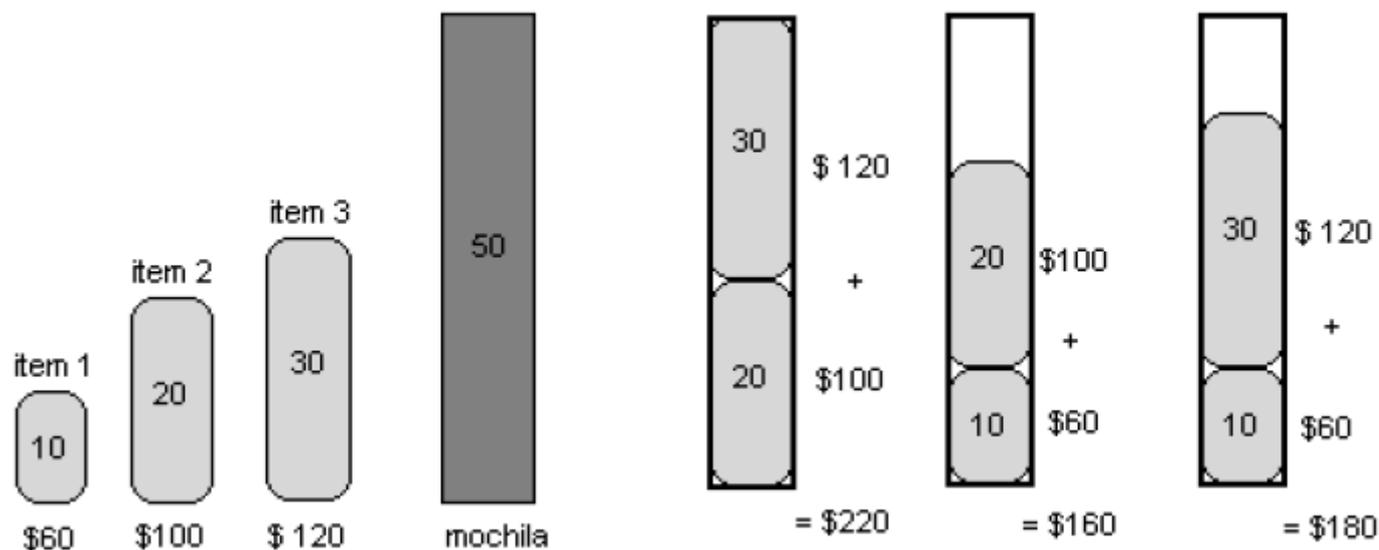
BCC241/2011-2



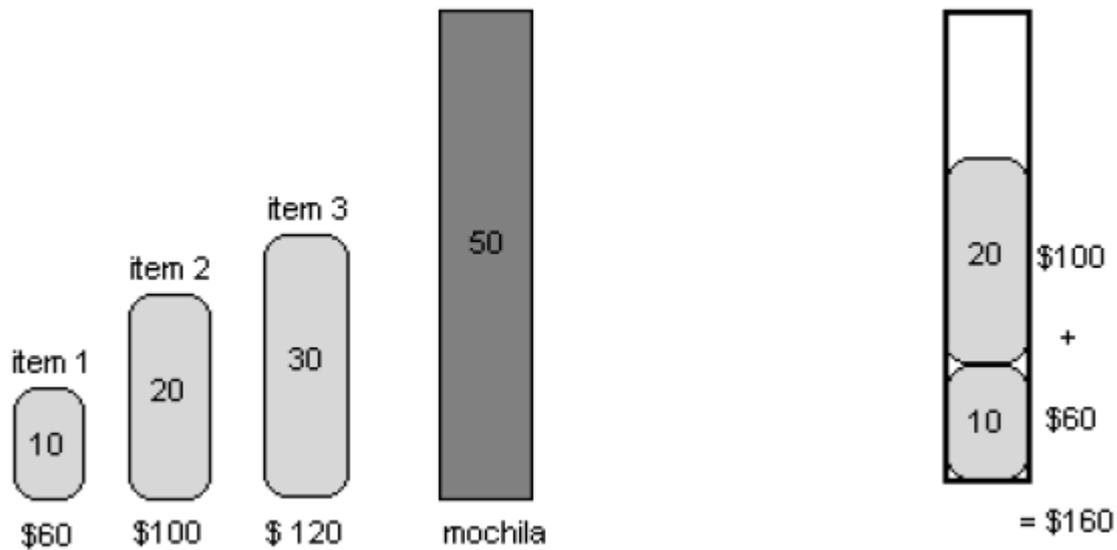
Problemas alvo de branch-and-bound

- Problemas de Otimização
- Sequência de decisões independentes (ordem de atribuição não influencia)
- Relacionamento entre essas decisões se expressa através da viabilidade da solução
- Permite podar soluções parciais não-promissoras através de estimativas otimistas.

Problema da Mochila: Soluções



Mochila: Guloso

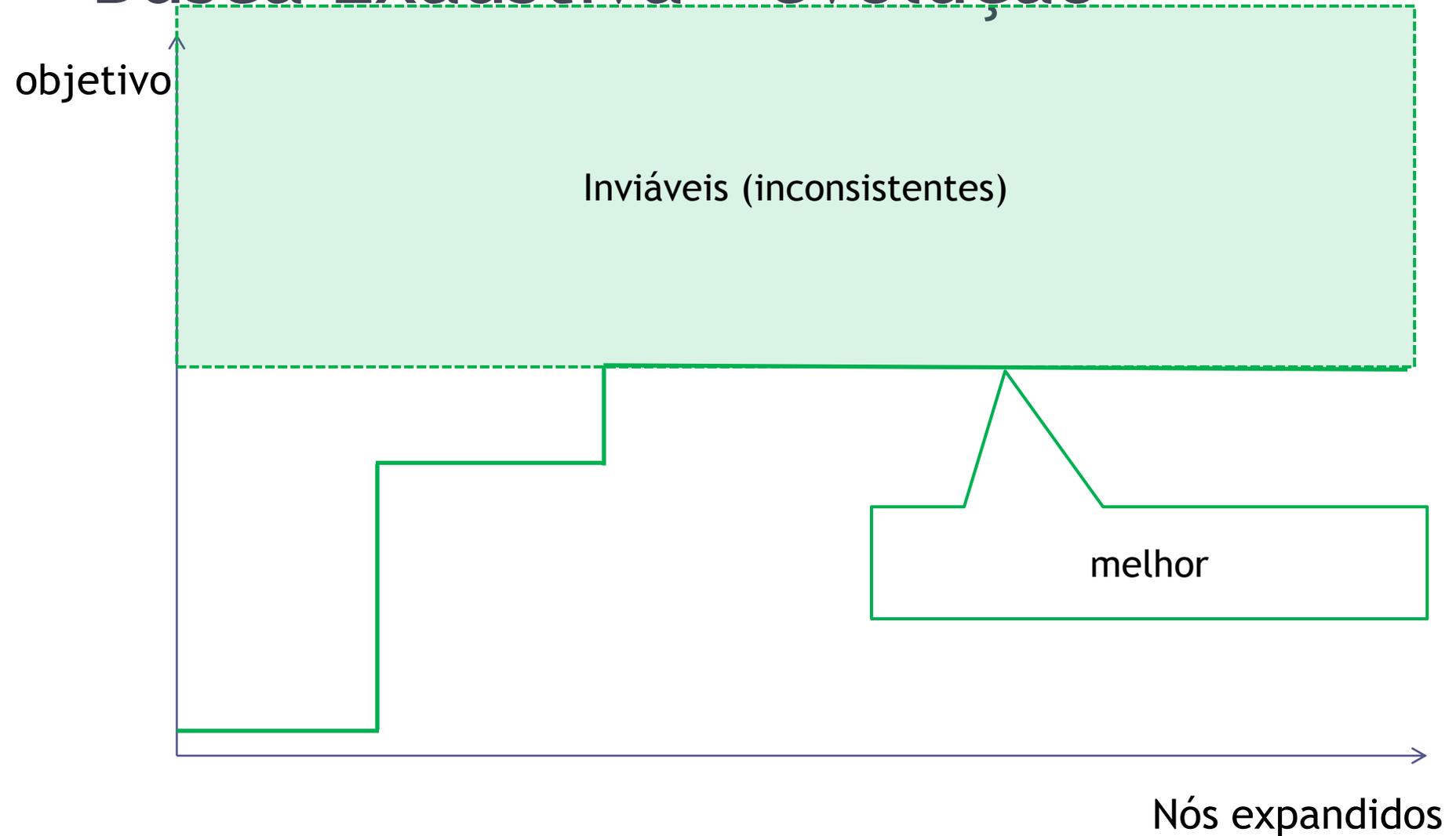


Busca Exaustiva

```
Problema p = LeProblema();  
Solucao melhor = geraSolucaoInicial(p);  
Solucao inicial = (-, ..., -);  
BEOTimizacao(inicial, 1, p);
```

```
function BEOTimizacao(Solucao s, int i, Problema p)  
1   if (eCompleta(p, s, i)) & (eConsistente(p, s, i)):  
2       if (objetivo(p, s) > objetivo(p, melhor)):  
3           melhor = s;  
4   else:  
5       j = primeiroValor(p, i);  
6       while (j <= ultimoValor(p, i))  
7           s[i] = j;  
8           BEOTimizacao(s, i+1, p)  
9           s[i] = livre;  
10          j++  
end
```

Busca Exaustiva - evolução

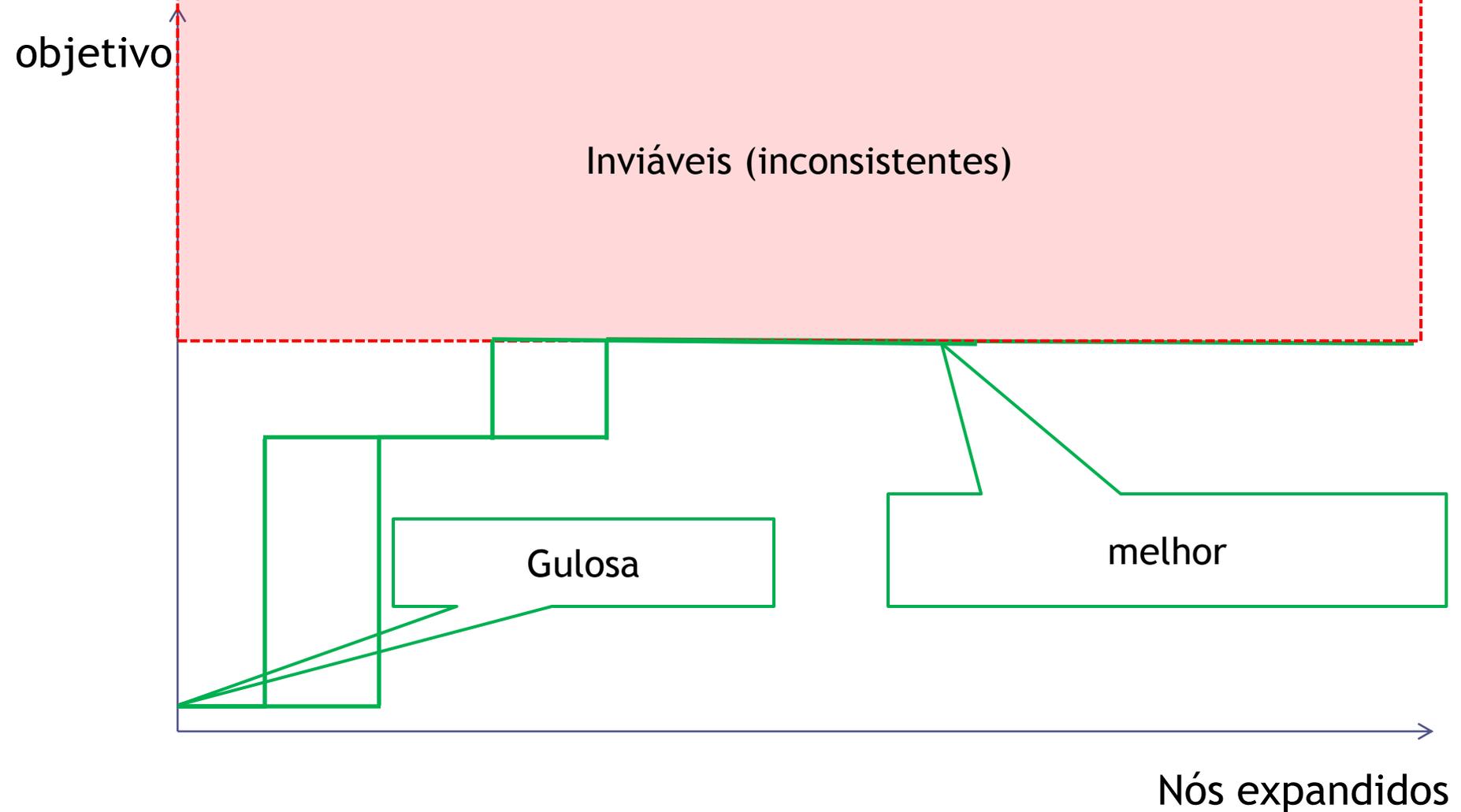


Backtracking

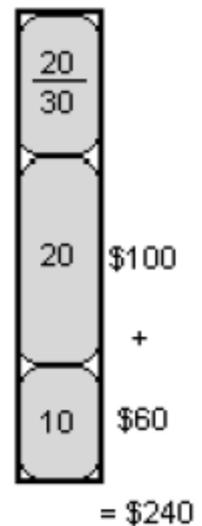
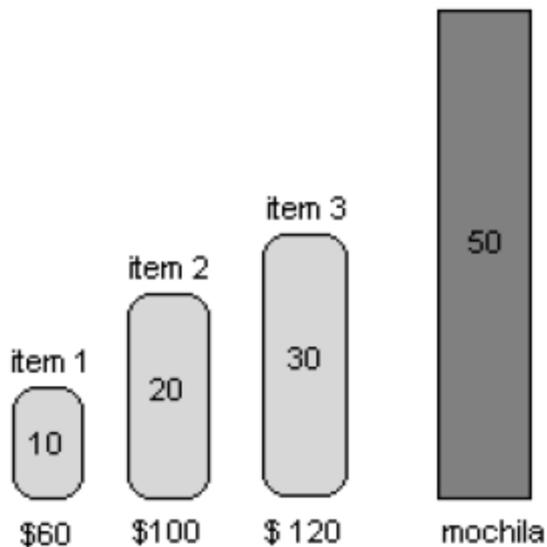
```
Problema p = LeProblema();  
Solucao melhor = geraSolucaoInicial(p);  
Solucao inicial = (-, ..., -);  
BTOtimizacao(inicial, 1, p);
```

```
function BTOtimizacao(Solucao s, int i, Problema p)  
1   if (eCompleta(p, s, i) & eConsistente(p, s, i)):  
2       if (objetivo(p, s) > objetivo(p, melhor)):  
3           melhor = s;  
4   else  
5       j = primeiroValor(p, i);  
6       while (j <= ultimoValor(p, i))  
7           s[i] = j;  
8           if (eConsistente(p, s)) :  
9               BTOtimizacao(s, i+1, p)  
10          s[i] = livre;  
11          j++  
end
```

Backtracking - evolução



Guloso Sub-ótimo: Problema da Mochila



Branch-and-bound

- Branch
 - Ordem de exploração das variáveis
 - Estratégias de caminhamento (profundidade, largura, melhor primeiro)

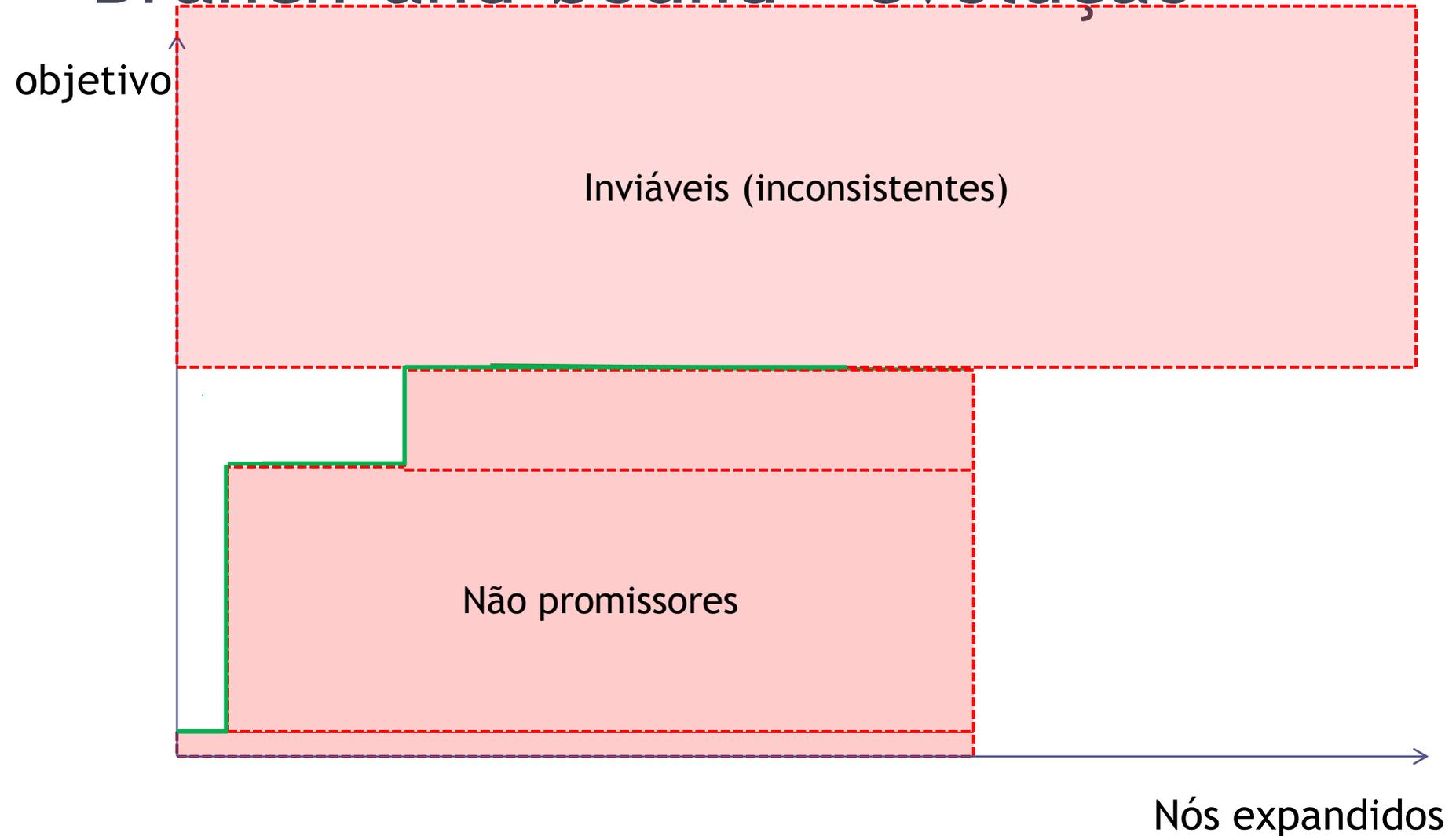
- Bound
 - eConsistente
 - ePromissora

Branch-and-bound

```
Problema p = LeProblema();  
Solucao melhor = Gulosa(p);  
Solucao inicial = (-, ..., -);  
BBOtimizacao(inicial, 1, p);
```

```
function BBOtimizacao(Solucao s, int i, Problema p)  
1   if (eCompleta(p, s, i)):  
2       if (objetivo(p, s) > objetivo(p, melhor)):  
3       melhor = s;  
4   else  
5       j = primeiroValor(p, i);  
6       while (j <= ultimoValor(p, i))  
7           s[i] = j;  
8           if (eConsistente(p, s) & ePromissora(p, s)):  
9               BBOtimizacao(s, i+1, p)  
10          s[i] = livre;  
11          j++  
12      end  
end
```

Branch-and-bound - evolução



Valor	Peso	
45	3	
45	9	
30	5	
10	2	

Mochila

Implementação

- Funções importantes
 - eCompleta(Problema p, Solucao s)
 - eConsistente(Problema p, Solucao s)
 - geraSolucaoInicial(Problemap, Solucao s)
 - ePromissora(Problema p, Solucao s)

- Exemplo em Java
 - <http://www.brpreiss.com/books/opus5/html/page9.html>

Componentes determinantes

- Função de corte
 - ePromissora, no curso, gulosa
- Estratégia de exploração
 - No curso, só profundidade
 - Largura, melhor primeiro, mista
- Estratégia de ramificação
 - No curso, fixo para todos
 - Ordem das variáveis e dos valores
- Solução inicial
 - geraSolucaoInicial, no curso, gulosa
 - Soluções aproximadas, heurísticas, metaheurísticas

Complexidade

- Funções de limite

Compromisso entre o custo e o número de nós podados.

- Estratégias de caminhamento

- Memória
- Chegar rápido a uma solução
- Mistas
- Número de nós explorados.

Exercício

- TSP – Problema do Caixeiro Viajante

