

## Métodos Exatos

Pedro Hokama

- Curso Discrete Optimization no Coursera - Prof. Dr. Pascal Van Hentenryck
- [cls] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [timr] Algorithms Illuminated Series, Tim Roughgarden
- Pesquisa Operacional, Arenales, M. N., Armentano, V. A., Morabito Neto, R., e Yanasse, H. H. (2015)
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms  
<https://www.youtube.com/playlist?list=PLXFMm1k03Dt7Q0xr1PIAriY5623cKiH7V>  
<https://www.youtube.com/playlist?list=PLXFMm1k03Dt5EMI2s2WQBsLsZ17A5HEK6>
- Introdução à Otimização Combinatória, Flávio K. Miyazawa e Cid C. de Souza.
- Otimização Combinatória, da Carla Negri Lintzmayer do CMCC, UFABC  
 Qualquer erro é de minha responsabilidade.

## Limitantes

Em um problema de maximização:

- Um limitante superior é um valor maior ou igual que o ótimo. Pode ser obtido através de alguma relaxação do problema por exemplo. (limitante dual)
- Um limitante inferior é um valor menor ou igual que o ótimo. Poder ser obtido por uma heurística por exemplo. (limitante primal)

Exemplo: Problema da Mochila Binária.

Itens	1	2	3	4	5	
Pesos	2	5	4	6	5	Capacidade da Mochila: 14
Valor	10	11	10	13	14	

Limitante Inferior:

- Qualquer solução pode ser considerada um limitante inferior.
- Por exemplo se pegarmos os itens na ordem que foram dados até encher a mochila.

Pesos	<b>2</b>	<b>5</b>	<b>4</b>	6	5	
Valor	<b>10</b>	<b>11</b>	<b>10</b>	13	14	Solução: 31

## Limitante Superior (Dual)

- A solução ótima que tem valor  $z^*$  é pelo menos 31. Ou seja  $z^* \geq 31$
- Podemos tentar fortalecer esse limitante.
- E se escolhermos os itens na ordem dos que tem o melhor custo-benefício (valor/peso)

Valor/Peso (Custo-Benefício)	5	2.2	2.5	2.166	2.8
Pesos	2	5	4	6	5
Valor	10	11	10	13	14

Solução: 34, ou seja  $z^* \geq 34$

Itens	1	2	3	4	5
Pesos	2	5	4	6	5kg
Valor	10	11	10	13	14

- Se conseguíssemos preencher toda a mochila com o item que vale mais a cada kg?

Valor/Peso (Custo-Benefício) 5 2.2 2.5 2.166 2.8

5 / 28

6 / 28

- Se enchermos os 14kg que cabem na mochila com 5\$ por kg.
- Um limitante superior para esse problema é  $14 * 5 = 70$ , ou seja  $z^* \leq 70$
- Ou seja é verdade que nenhuma solução (incluindo a ótima) pode ser maior que 70.
- Esse limitante claro é bem fraco. E se enchermos a mochila só com os itens que temos até encher completamente a mochila, relaxando a integralidade.

Itens	1	2	3	4	5
Pesos	2	5	4	6	5kg
Valor	10	11	10	13	14

Valor/Peso (Custo-Benefício) 5 2.2 2.5 2.166 2.8  
 Pegamos o item 1, depois pegamos o item 5, depois o item 3 e enchemos o resto da mochila com 0.6 do item 2.

$$10 + 0.6 * 11 + 10 + 14 = 40.6$$

- Concluímos que a solução ótima pode ser no máximo 40.6.
- De fato como todos os valores são inteiros, podemos afirmar que  $z^* \leq 40$

7 / 28

8 / 28

## Branch-and-Bound

Itens	1	2	3	4	5
Pesos	2	5	4	6	5kg
Valor	10	11	10	13	14

- Concluimos que  $34 \leq z^* \leq 40$ .
- Se encontrarmos uma solução com custo igual ao limitante superior (40 no exemplo), temos certeza que ela é ótima.
- Mas a solução ótima para esse problema é a seguinte

Pesos	<b>2</b>	5	4	<b>6</b>	<b>5kg</b>
Valor	<b>10</b>	11	10	<b>13</b>	<b>14</b>

Solução: 37.

- Técnica empregada na resolução de problemas difíceis de otimização combinatória.
- O BnB enumera implicitamente todas as soluções do problema.
- Implicitamente, pois se explorasse de fato todas as soluções seria equivalente a um algoritmo de força-bruta.

9 / 28

10 / 28

## Branch-and-Bound

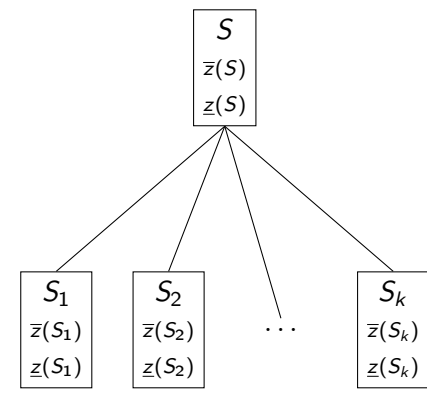
- Os algoritmos de BnB são uma aplicação do paradigma de divisão e conquista.
- O Branch-and-Bound tem duas partes principais, que como você pode imaginar são:
  - ▶ *Branch*: ramificação, que é o processo de dividir o problema.
  - ▶ *Bound*: que é a busca por limitantes (inferiores e superiores) para cada subproblema.

- Considere um problema de maximização qualquer, em que desejamos encontrar a solução ótima.
- Seja  $S$  o conjunto de todas as soluções viáveis.
- Seja  $f : S \rightarrow \mathbb{R}$  a função objetivo que mapeia cada solução  $s \in S$  a um valor real.

11 / 28

12 / 28

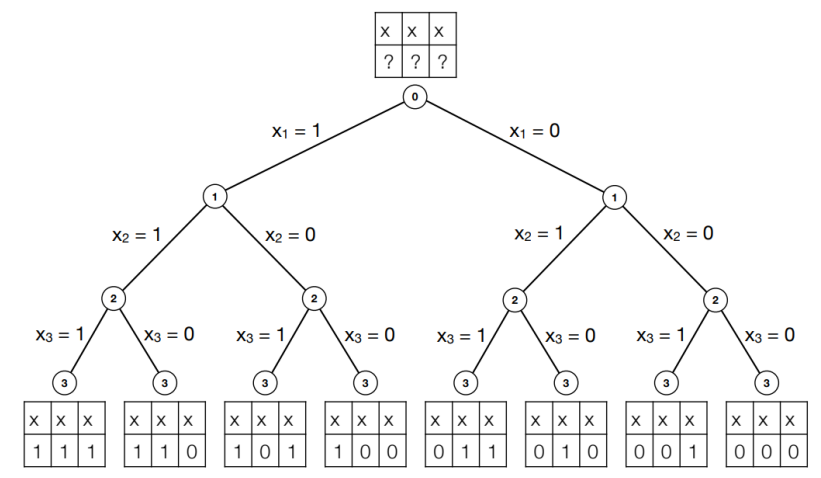
- Vamos considerar que podemos calcular bons limitantes superiores e inferiores para um conjunto  $S' \subseteq S$ .
  - ▶ Seja  $\bar{z}(S')$  um limitante dual (superior) para  $S'$ , ou seja, um valor tal que nenhuma solução de  $S'$  tenha um valor de função objetivo maior que  $\bar{z}(S')$ .  $f(s \in S') \leq \bar{z}(S')$
  - ▶ Seja  $\underline{z}(S')$  um limitante primal (inferior) para  $S'$ , ou seja, existe (com certeza) alguma solução em  $S'$  que tenha valor igual ou superior a  $\underline{z}(S')$ .  $\max\{f(s)|s \in S'\} \geq \underline{z}(S')$
- Seja  $\{S_1, S_2, \dots, S_k\}$  uma partição de  $S$ , ou seja,  $S_1 \cup S_2 \cup \dots \cup S_k = S$ .
- Podemos calcular os limitantes primais e duais tanto para  $S$  quanto para  $S_1, S_2, \dots, S_k$ .



- Suponha que  $\bar{z}(S_i) \leq \underline{z}(S_j)$ . Como estamos buscando a solução ótima, ou seja, a solução com o maior valor. Sabemos que a melhor solução em  $S_i$  vai ser pior (ou igual) a melhor solução em  $S_j$  e por isso podemos buscar a solução apenas em  $S_j$ , *podando*  $S_i$ . Essa é a comumente chamada  **poda por limitante**.

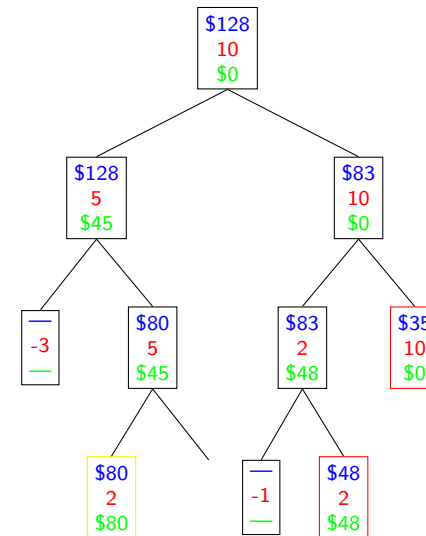
maximize  $45x_1 + 48x_2 + 35x_3$   
 subject to  $5x_1 + 8x_2 + 3x_3 \leq 10$   
 $x_i \in \{0, 1\} \quad (i \in 1..3)$

- Como vamos subdividir o espaço de busca? Vamos dividir entre as decisões que podemos fazer em cada item.



<sup>1</sup>Discrete Optimization, Professor Pascal Van Hentenryck

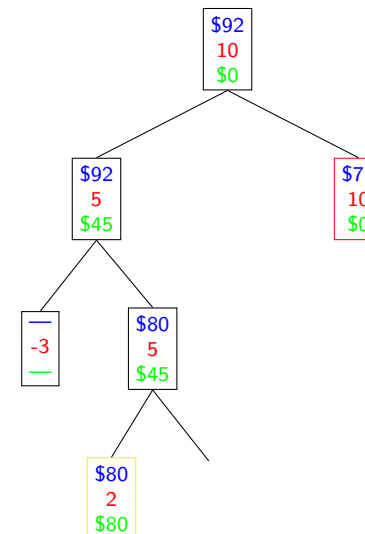
- Vamos simular a exploração
- Limitante inferior: Os itens que já estão na solução parcial.
- Limitante superior: Vamos relaxar a capacidade da mochila.  
(Ainda vamos verificar quando a mochila ultrapassa a capacidade)
- Vamos denotar cada solução pelo valor do limitante inferior (verde) e a capacidade residual (vermelho) e o limitante superior (azul).



17 / 28

18 / 28

- Limitante superior: Vamos relaxar a integralidade da mochila, ou seja, podemos pegar uma fração de um item.



19 / 28

20 / 28

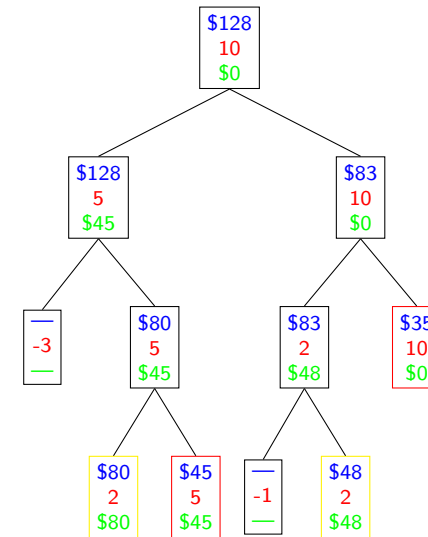
## Branching Strategies

### Estratégias de Ramificação

- Podemos elaborar diferentes estratégias para explorar os nós de uma árvore de branch-and-bound.
- A Depth First (que vimos anteriormente) sempre desce o mais profundo possível (indo sempre o mais a esquerda possível e depois para a direita)
- Cada estratégia pode ser um desempenho diferente dependendo do problema (e da instância).
- Mas podemos analisar alguns aspectos. Por exemplo, qual a eficiência de memória do Depth First?

21 / 28

## Best-First



22 / 28

## Best-First

- Sempre expande o nó "Mais promissor", ou seja, aquele que tem o maior limitante superior (no caso de um problema de maximização)
- Ele poda sempre que encontra um nó em que o limitante já é pior do que a melhor solução conhecida.
- Ele é eficiente em memória? Não muito, pode ter um número muito grande de nós ativos e cada nó precisa conter uma forma de recuperar a solução parcial.
- Ele é fácil de implementar? Normalmente é mais difícil do que o Depth-First

23 / 28

## Ramificações

- A ideia então é dividir o espaço de soluções, calculando os limitantes e podando os nós que não fornecem soluções promissoras.
- A forma de dividir, depende do problema e da estratégia aplicada. A ideia é que a cada divisão represente uma escolha diferente.

24 / 28

- Por exemplo, no problema da mochila uma divisão pode ser decisão por colocar o item  $i$  na mochila e a decisão de não colocar o item  $i$  na mochila.
- Nesse exemplo obteríamos uma árvore binária, e cada nível dessa árvore pode representar a escolha de um item diferente.

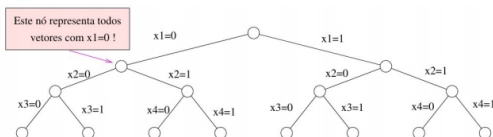


Figura 1.8. Árvore de enumeração completa para o problema binário da mochila com 3 itens.

2

Outros exemplos:

- No problema do caixeiro viajante cada escolha poderia ser a escolha do próximo vértice a visitar, e nesse caso a árvore não seria binária.
- No problema de encontrar a clique máxima, uma escolha poderia ser a inclusão de um item na clique.
- No problema do *bin packing*, uma escolha poderia ser uma atribuição de um item a um contêiner.
- etc.

<sup>2</sup><https://ic.unicamp.br/fkm/lectures/intro-otimizacao.pdf>

## Algoritmo<sup>3</sup>

1. B&B; (\* considerando problema de maximização \*)
2. Ativos  $\leftarrow$  {nó raiz}; melhor-solução  $\leftarrow$  {};  $z \leftarrow -\infty$ ;
3. Enquanto (Ativos não está vazia) faça
4.   Escolher um nó  $k$  em Ativos para ramificar;
5.   Remover  $k$  de Ativos;
6.   Gerar os filhos de  $k$ :  $n_1, \dots, n_q$  computando  $\bar{z}_{n_i}$  e  $z_{n_i}$  correspondentes;  
 (\* definir  $\bar{z}_{n_i}$  e  $z_{n_i}$  iguais a  $-\infty$  para subproblemas inviáveis \*)
7.   Para  $j = 1$  até  $q$  faça
8.     se ( $\bar{z}_{n_j} \leq z$ ) então podar o nó  $n_j$ ; (\* inclui os 3 casos \*)
9.     se não
10.       Se ( $n_j$  representa uma única solução) então (\* atualizar melhor limitante primal \*)
11.        $z \leftarrow \bar{z}_{n_j}$ ; melhor-solução  $\leftarrow$  {solução de  $n_j$ };
12.       se não adicionar  $n_j$  à lista Ativos.

Outros ramificações:

- Least-Discrepancy: Ramifica em ondas, a cada onda ele permite que uma curva a mais aconteça na árvore.

<sup>3</sup><https://ic.unicamp.br/fkm/lectures/intro-otimizacao.pdf>