

Métodos de Apoio à Decisão

Pesquisa em árvore: aplicações

João Pedro Pedroso

2024/2025

Problemas de cobertura

Há seis cidades na zona do Porto, e o governo regional está a estudar onde deverá construir quartéis de bombeiros.

Cada cidade deverá ficar a menos de 15 minutos de pelo menos um quartel. Os tempos de deslocação entre cidades são:

Cidade	1	2	3	4	5	6
1	0	10	20	30	30	20
2		0	25	35	20	10
3			0	15	30	20
4				0	15	25
5					0	14
6						0

Pretende-se determinar o número mínimo de quartéis a construir.

- Variáveis ($i = 1, \dots, 6$):
 - $x_i = 1$ se se constrói quartel na cidade i
 - $x_i = 0$ caso contrário
- Objetivo: minimizar $z = x_1 + x_2 + \dots + x_6$
- Restrições:

cidade 1:	x_1	$+x_2$					\geq	1
cidade 2:	x_1	$+x_2$				$+x_6$	\geq	1
cidade 3:			x_3	$+x_4$			\geq	1
cidade 4:			x_3	$+x_4$	$+x_5$		\geq	1
cidade 5:				x_4	$+x_5$	$+x_6$	\geq	1
cidade 6:		$+x_2$			$+x_5$	$+x_6$	\geq	1

Knapsack problem

- Um ladrão está a fazer um assalto e tem à escolha uma série de objetos
 - valor
 - peso
- Para cada um deles, pode-o roubar ou não
 - i.e., não pode *dividir* objetos
- Leva-os numa mochila, que tem capacidade limitada
 - não pode levar mais de um determinado peso
- O que deve escolher para maximizar o valor?

Problemas da mochila: formulação

- Variáveis:

- $x_i = 1$ se escolher o objeto i , $i = 1, \dots, n$
- $x_i = 0$ caso contrário

- Restrição: peso total inferior ao limite b :

$$\sum_{i=1}^n a_i x_i \leq b$$

- Objetivo: maximizar valor

$$\sum_{i=1}^n c_i x_i$$

$$\begin{aligned}
 \text{maximizar } z = & \quad c_1x_1 + \quad c_2x_2 + \quad \dots + \quad c_nx_n \\
 \text{sujeito a:} & \quad a_1x_1 + \quad a_2x_2 + \quad \dots + \quad c_nx_n \leq \quad b \\
 & \quad x_i \in \{0, 1\}, \quad i = 1, \dots, n
 \end{aligned}$$

- Neste caso, a relaxação linear pode ser resolvida facilmente:
 - escolher objetos disponíveis por ordem do rácio c_i/a_i
 - no primeiro objeto que não couber:
 - colocar a fração que ainda caberia, enchendo completamente a mochila.
- Há (no máximo) uma variável fracionária; seja k
- Ramificação: num ramo $x_k = 0$ e no outro $x_k = 1$

Exemplo

maximizar $z = 16x_1 + 22x_2 + 12x_3 + 8x_4$

sujeito a: $5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$

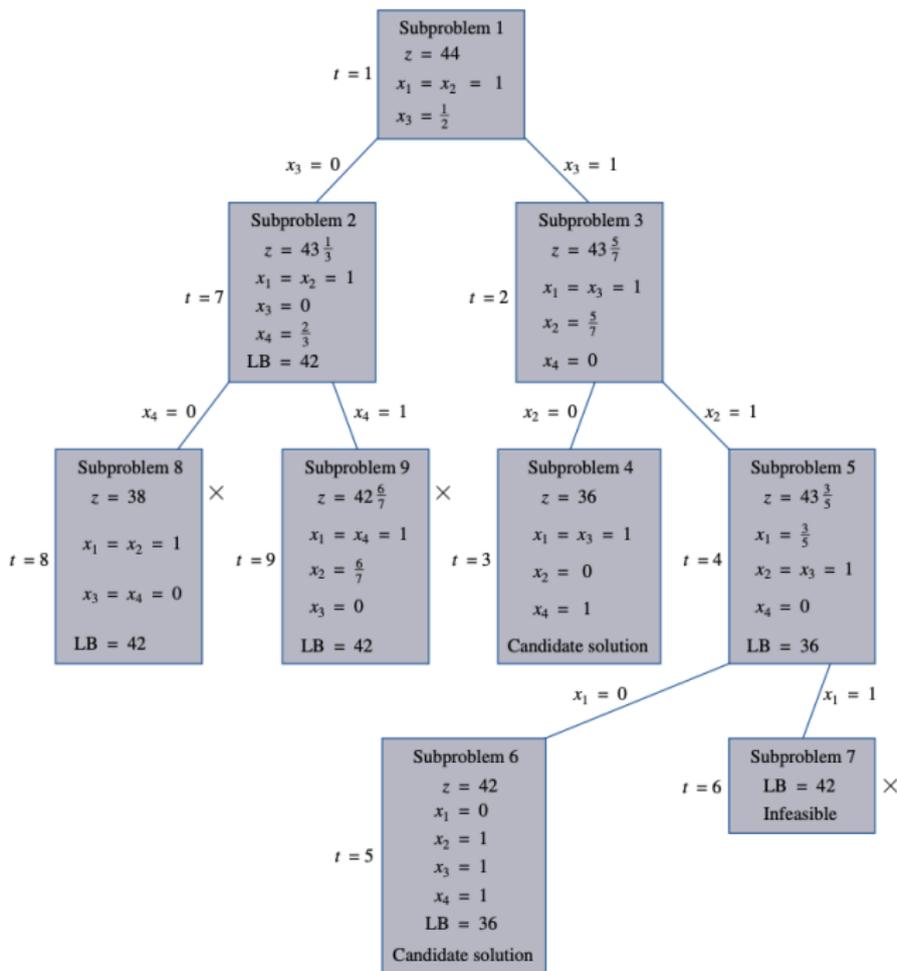
$x_i \in \{0, 1\}$, $i = 1, \dots, 4$

item	c_i/a_i	ranking
1	3.20	1
2	3.14	2
3	3.00	3
4	2.67	4

$$\begin{aligned} \text{maximizar } z &= 16x_1 + 22x_2 + 12x_3 + 8x_4 \\ \text{sujeito a: } &5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ &x_i \in \{0, 1\}, \quad i = 1, \dots, 4 \end{aligned}$$

Na raiz da árvore de pesquisa:

- colocamos o item 1 \rightarrow fica $14 - 5 = 9$
- colocamos o item 2 \rightarrow fica $9 - 7 = 2$
- item 3 no espaço que resta $\rightarrow 1/2$ do item \rightarrow fracionário



- 1 foi usada pesquisa em profundidade
 - decidiu-se arbitrariamente qual era o ramo esquerdo/direito
 - esquerdo $\rightarrow x_i = 0$
 - direito $\rightarrow x_i = 1$
 - seguindo primeiro o ramo da direita ($x_i = 1$)
- 2 solução ótima:
 - $z = 42$
 - $x_1 = 0, x_2 = x_3 = x_4 = 1$
- 3 a "melhor" variável não é usada

- otimização combinatória → informalmente:
 - há um número finito de soluções admissíveis
 - pesquisa em árvore permite explorar o espaço de soluções de forma sistemática
 - outra forma: enumeração
 - branch-and-bound:
 - forma de pesquisa em árvore
 - é um dos métodos mais eficientes para procurar a solução ótima

Problema de sequenciamento de tarefas

Pretende-se sequenciar uma série de trabalhos numa máquina, sendo o objetivo **minimizar o atraso total** (i.e., o atraso nas entregas somado para todas as encomendas). Os tempos de processamento t_i e datas de entrega d_i são os da tabela que se segue:

Trabalho	t_i	d_i
1	6	8
2	4	4
3	5	12
4	8	16

Como se pode resolver o problema utilizando o algoritmo da pesquisa em árvore?

É-nos dada uma lista de n trabalhos que devem ser executados numa máquina, e para cada trabalho i o tempo de processamento t_i e a data de entrega d_i .

Pretende-se saber qual é a ordem de execução que minimiza o atraso total (i.e., a soma dos atrasos para todos os trabalhos).

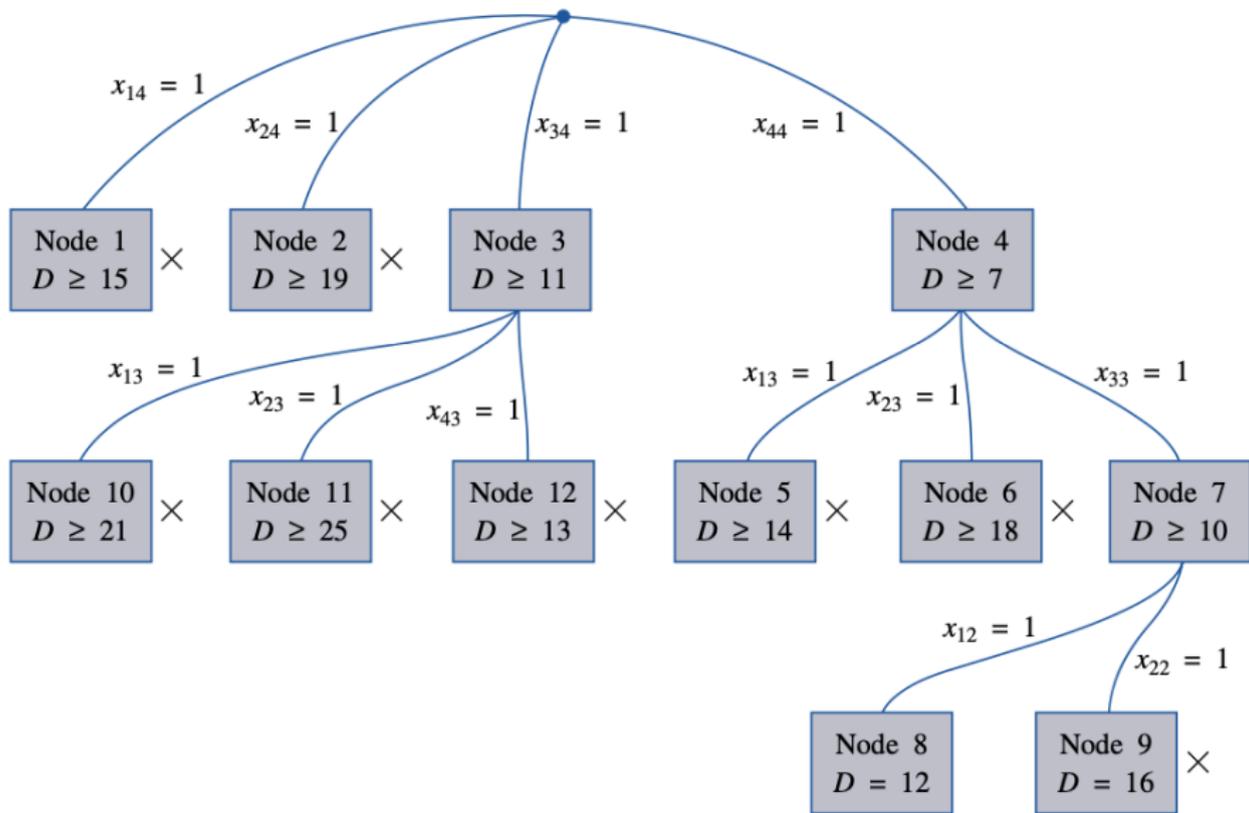
Atraso de cada encomenda:

- zero se for entregue antes do limite d_i
- $t - d_i$, onde t é o instante de entrega, caso esta ocorra depois do limite

- seja:
 - $x_{ij} = 1$ se o trabalho i é o j -ésimo a ser executado
 - $x_{ij} = 0$ caso contrário
- começa-se por particionar as soluções de acordo com o **último trabalho a ser processado**
- isso dá origem a n ramos na raiz da árvore de pesquisa
- em cada nó i pode-se determinar **minorante** do atraso total:

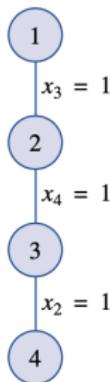
$$\sum_{j=1}^n t_j - d_i$$

- no nível seguinte, o *minorante* no atraso total no nó i é o atraso no primeiro nível mais $\sum_{j=1}^{n-1} t_j - d_i$;
- continua-se desta forma, até atingir as folhas da árvore de pesquisa
- nas **folhas** → atraso total da solução correspondente



Enumeração implícita

- Específico para problemas de otimização com variáveis binárias
- Permite fixar variáveis de forma sistemática
- Num dado momento da pesquisa:
 - variáveis fixas \rightarrow foram-lhes atribuídos determinados valores
 - variáveis livres \rightarrow ainda sem valor definido
 - *completion* (conclusão) \rightarrow instanciação de valores para as variáveis livres



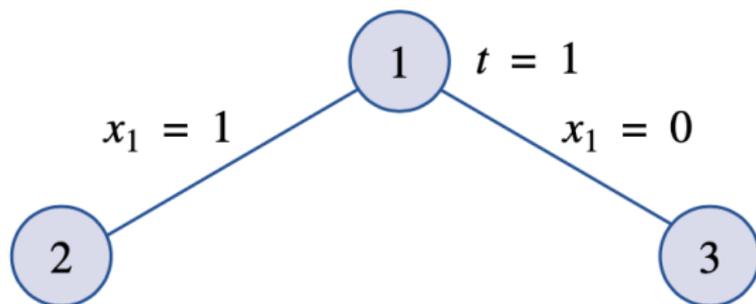
Passos principais:

- 1 Num determinado nodo, dados os valores das variáveis fixas:
 - há alguma boa *conclusão* para as variáveis livres?
 - escolher o valor 0/1 que melhora o objetivo
 - se a *conclusão* é admissível → não é necessário ramificar
- 2 Se a melhor *conclusão* é admissível:
 - fornece *bound* (majorante ou minorante) para as *conclusões* possíveis a partir desse nodo
 - pode ser utilizado para eliminar o nodo, caso se conheça solução melhor que esse *bound*
- 3 Num dado nodo, há alguma forma de determinar se nenhuma *conclusão* é admissível?
 - para cada restrição, ver instanciação **mais favorável** de cada variável
 - se houver alguma restrição não satisfazível → eliminar nodo

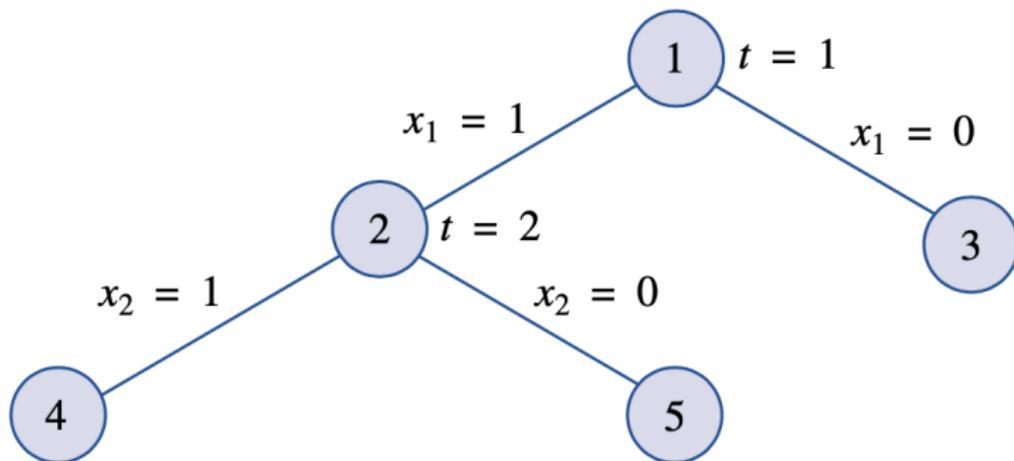
Enumeração implícita: exemplo

$$\begin{aligned} \max z = & -7x_1 - 3x_2 - 2x_3 - x_4 - 2x_5 \\ \text{s.t.} \quad & -4x_1 - 2x_2 + x_3 - 2x_4 - x_5 \leq -3 \\ & -4x_1 - 2x_2 - 4x_3 + x_4 + 2x_5 \leq -7 \\ & x_i = 0 \text{ or } 1 \quad (i = 1, 2, 3, 4, 5) \end{aligned}$$

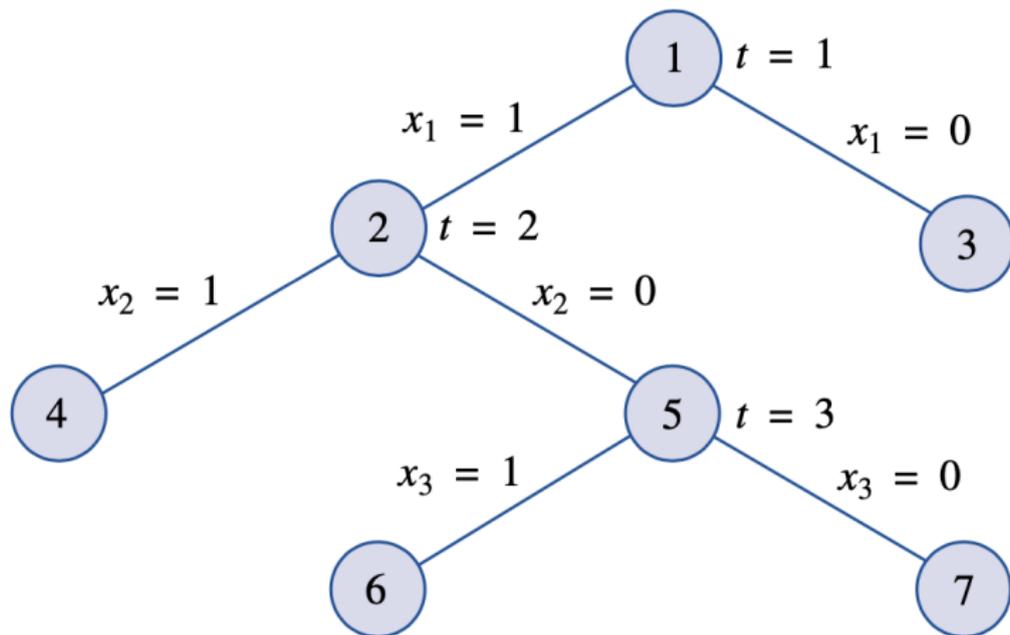
- Nodo 1: todos os $x_i = 0$
 - não é admissível
 - mas há *conclusões* admissíveis para o nodo 1
- Ramificar (arbitrariamente) por
 - $x_1 = 1 \rightarrow$ nodo 2
 - $x_1 = 0 \rightarrow$ nodo 3



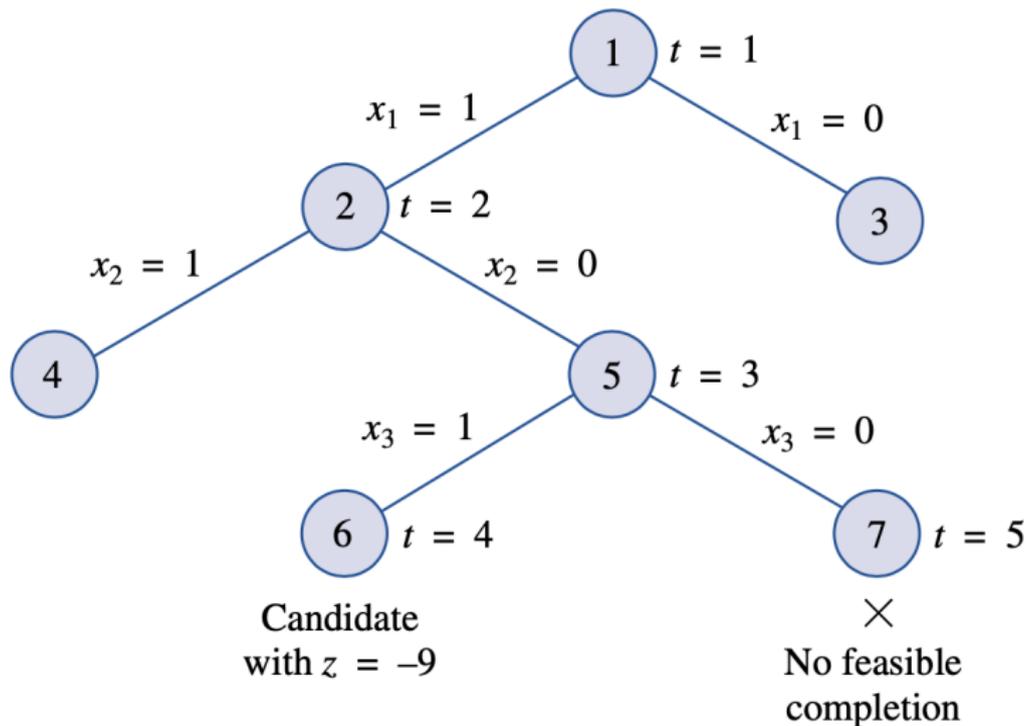
Ramificação do nodo 2



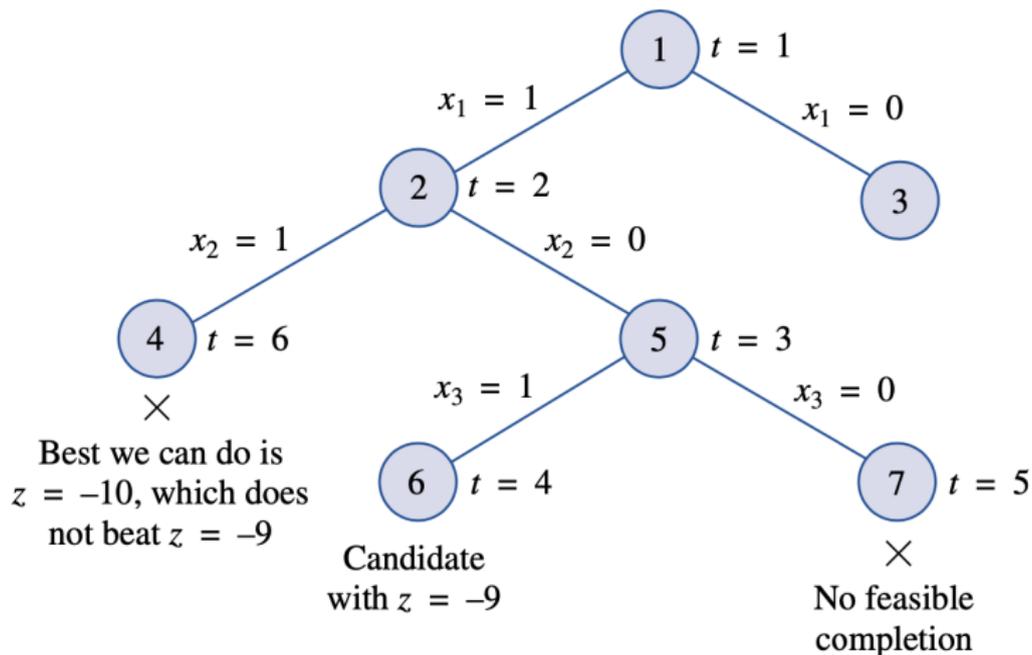
Ramificação do nodo 5



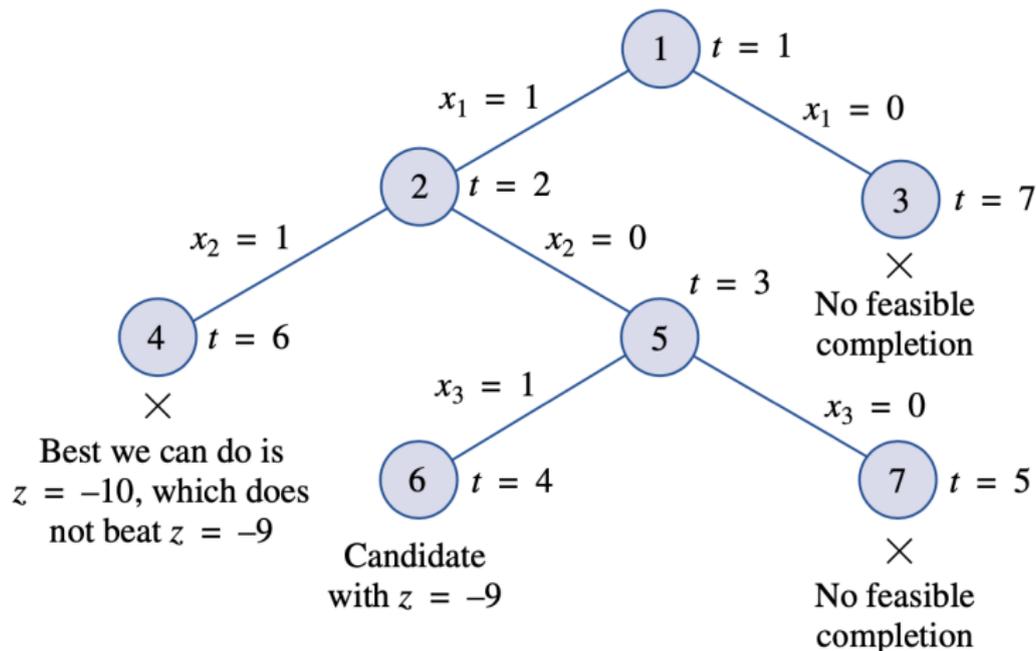
Ramificação do nodo 6



Ramificação do nodo 4



Ramificação do nodo 3



Aula de hoje:

- método da pesquisa em árvore para problemas específicos
- enumeração implícita

Próximas aulas:

- Problemas de otimização em grafos