Universidade Federal de São Carlos
Departamento de Engenharia de Produção

Otimização Linear Contínua e Discreta (Tópicos Avançados em PCSP)

PPGEP, UFSCar - Semestre 01/2022
Prof. Dr. Pedro Munari (munari@dep.ufscar.br)

Tópico 7.3: Método *Branch-and-Bound*: exemplo, algoritmo e alguns conceitos importantes

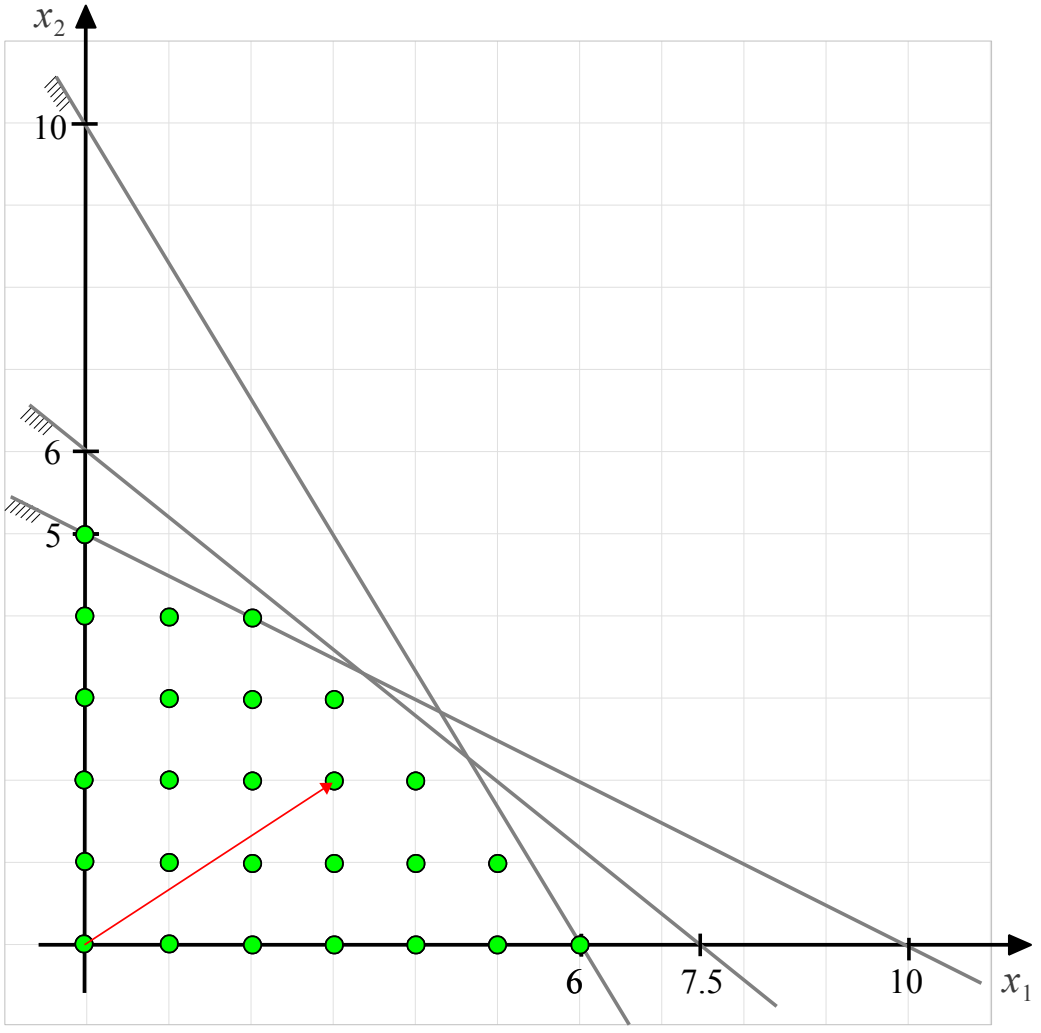
Objetivos deste tópico

- Aplicar o método *Branch-and-Bound* para compreender seu funcionamento;
- Estudar seu algoritmo e mais alguns conceitos, regras e variantes importantes.

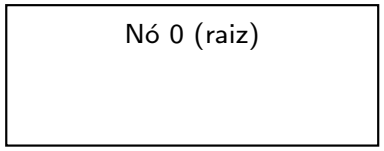
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
\max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1, x_2 \in \mathbb{Z}_+
\end{aligned}
$$



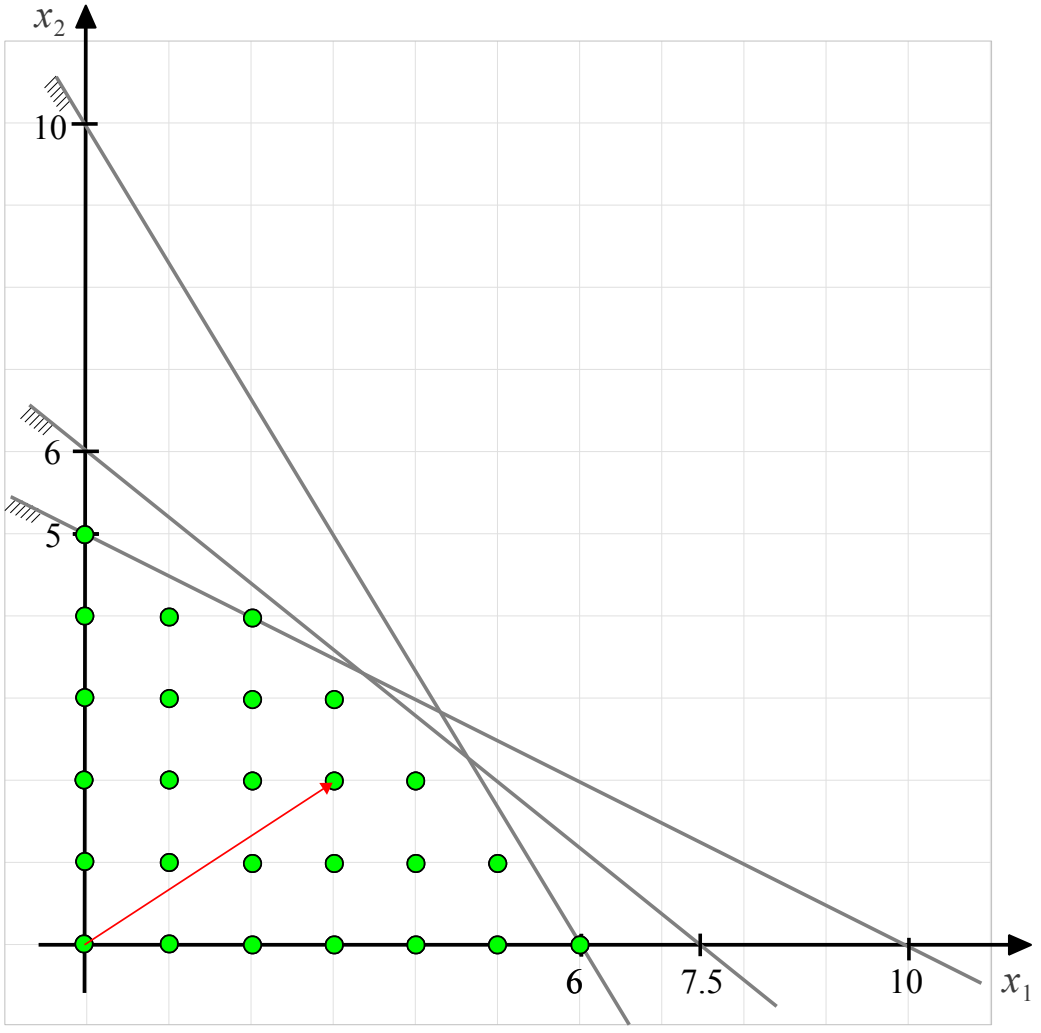
- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{0\}$



Método *branch-and-bound*

▷ Exemplo

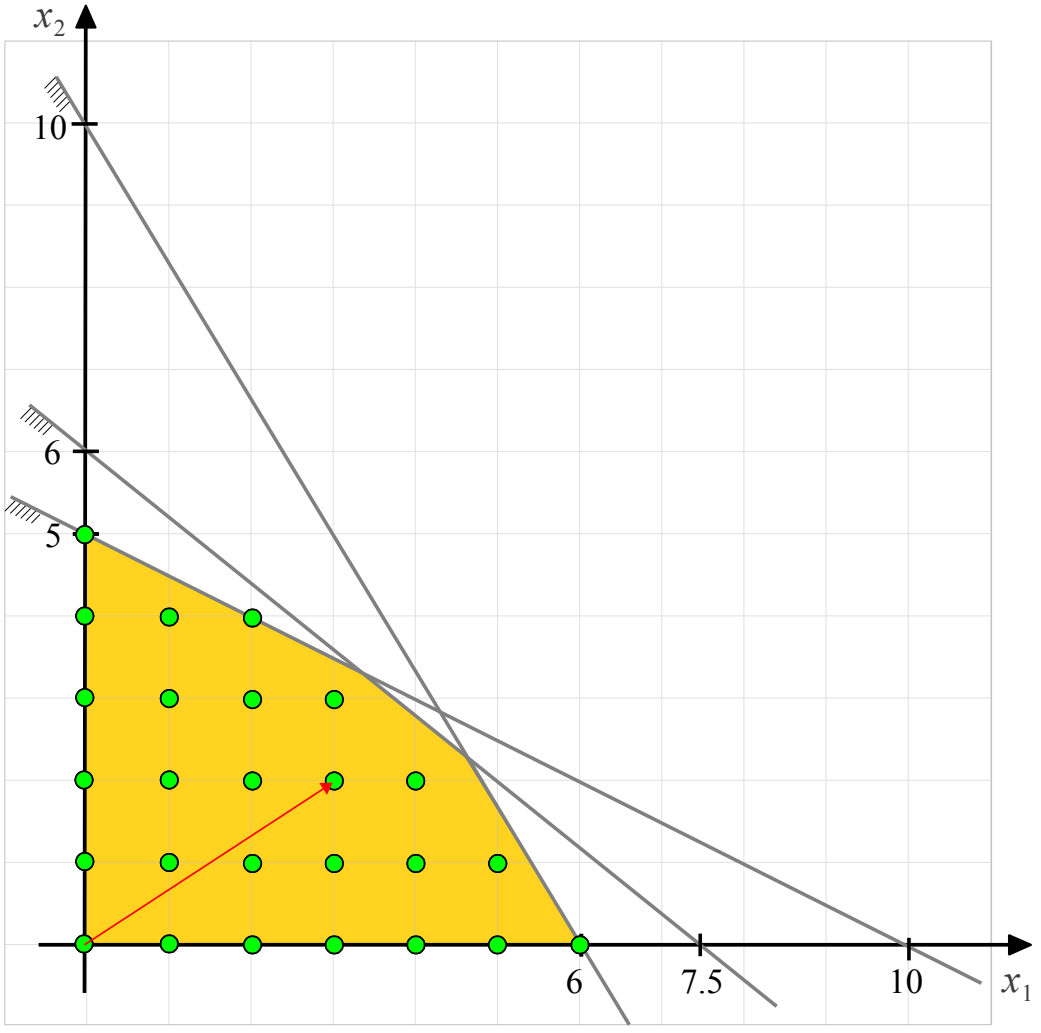
$$
\begin{aligned}
\max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1, x_2 \in \mathbb{Z}_+
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

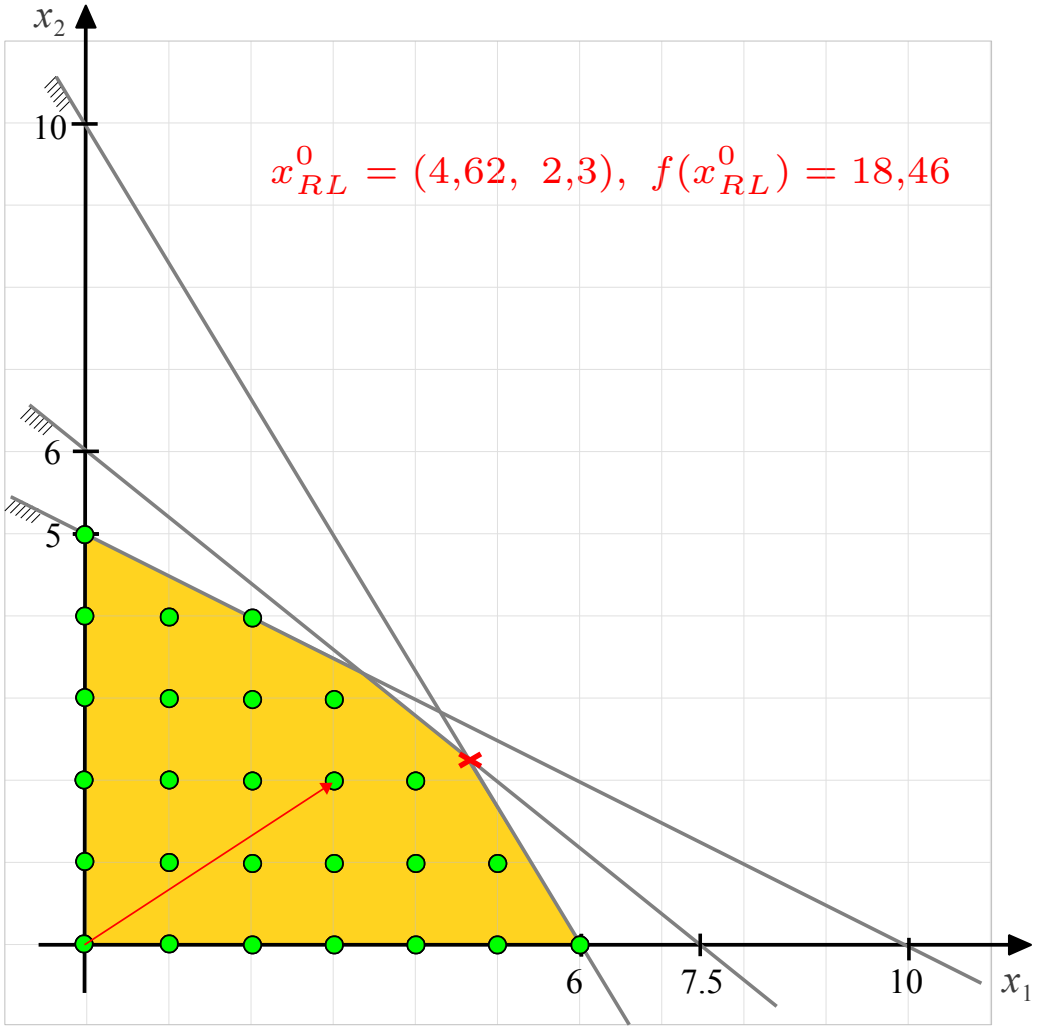
$$
\begin{aligned}
(S^0)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

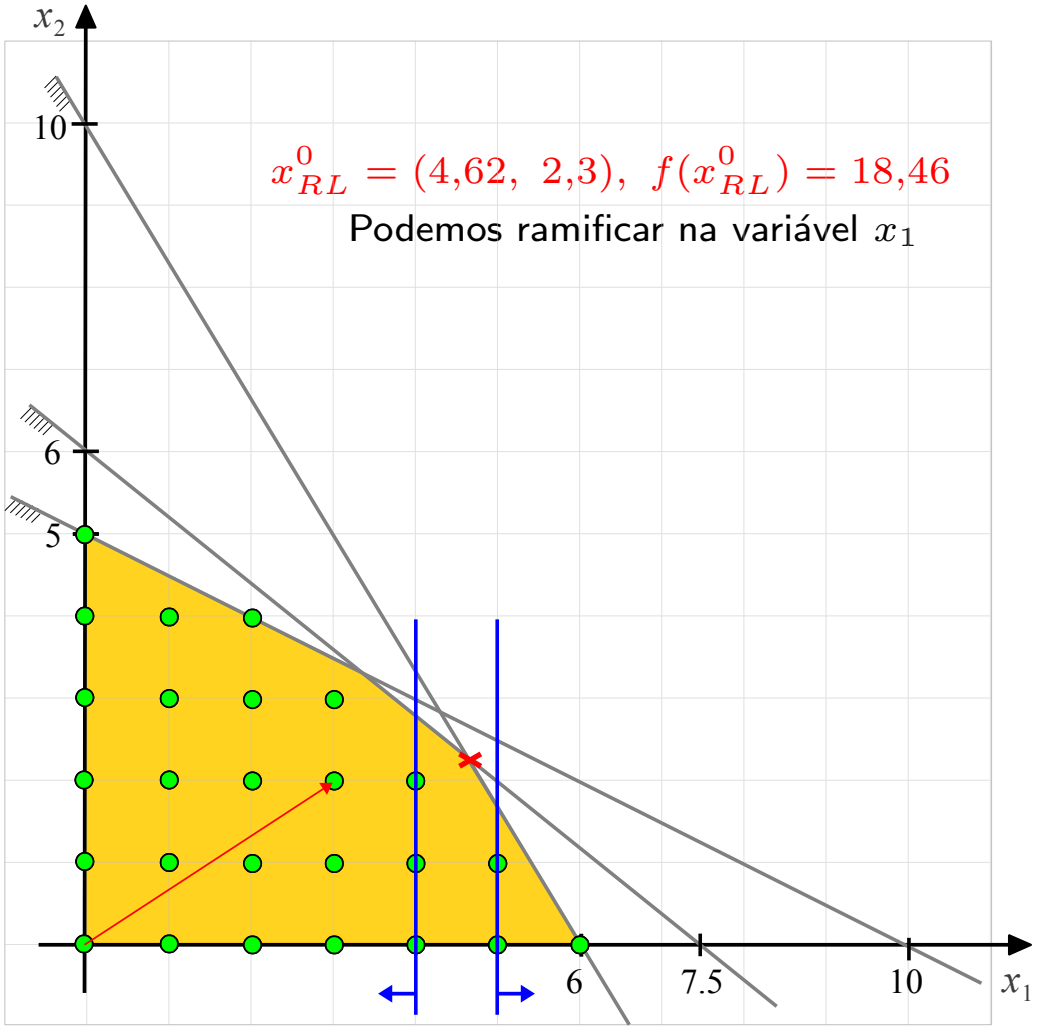
$$
\begin{aligned}
(S^0)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



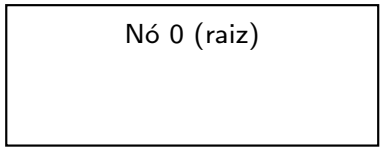
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^0)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



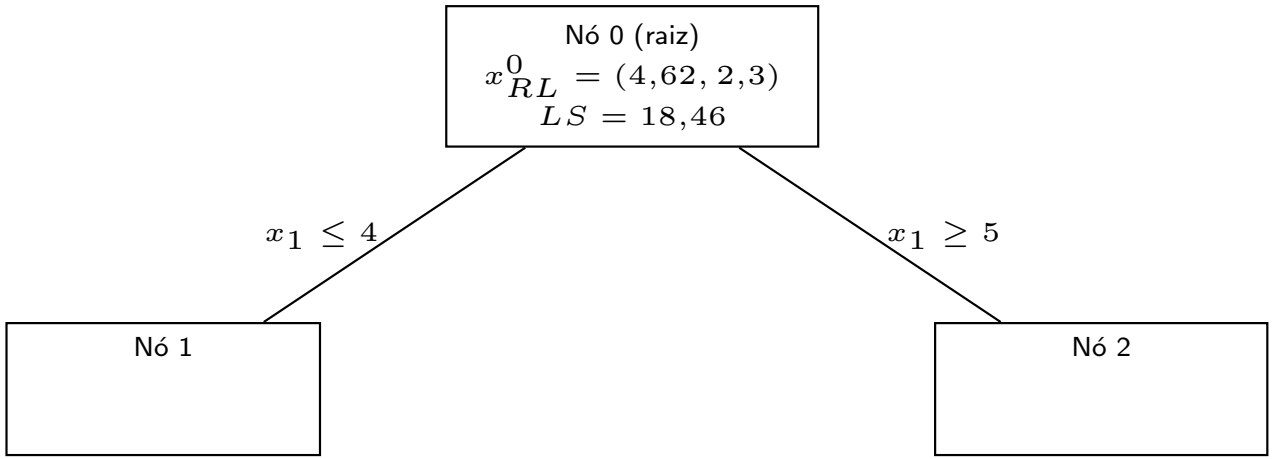
- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{0\}$



- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\phi\}$

Nó 0 (raiz)
$x^0_{RL} = (4{,}62,\ 2{,}3)$
$LS = 18{,}46$

- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\not{\phi}, 1, 2\}$

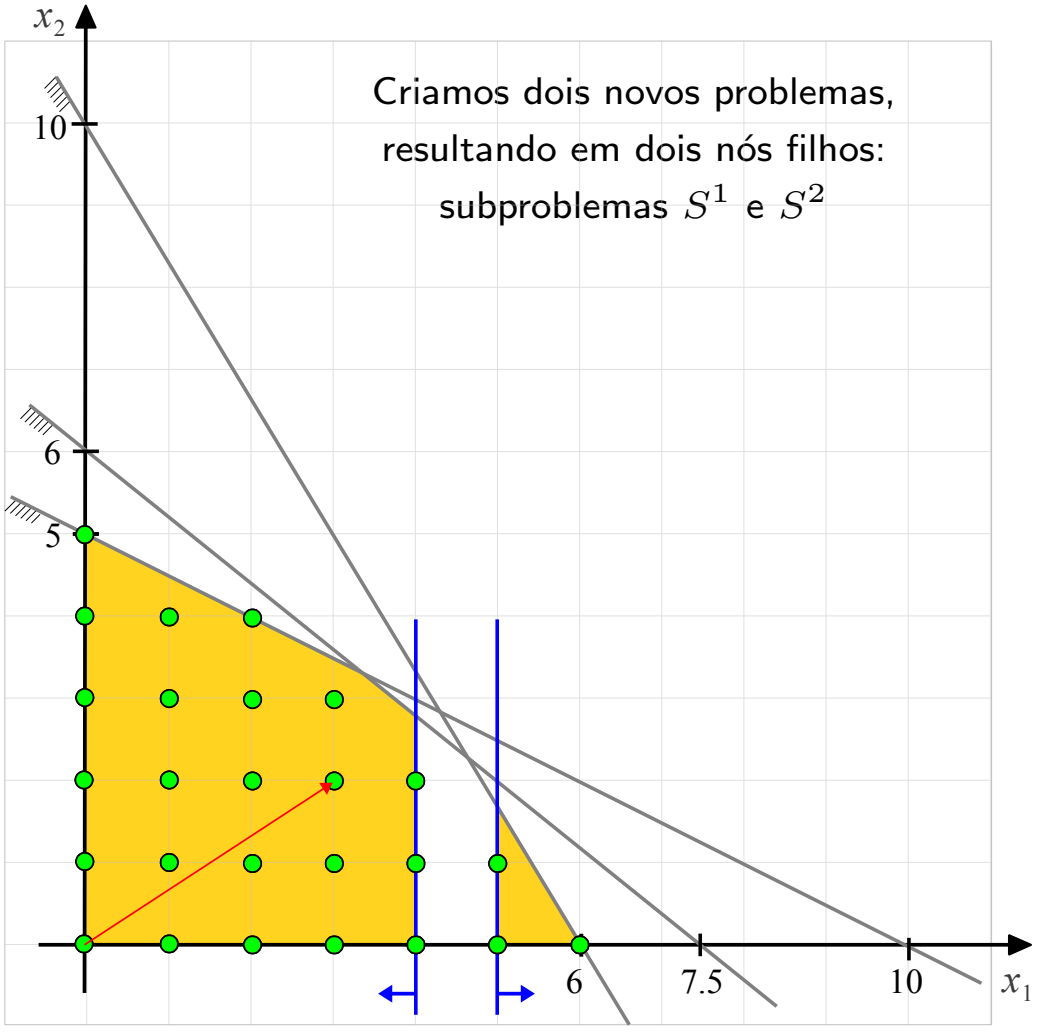


Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^1)\ \max\quad & f(x_1,x_2) = 3x_1 + 2x_2 \\
\text{s.a}\quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \leq 4 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

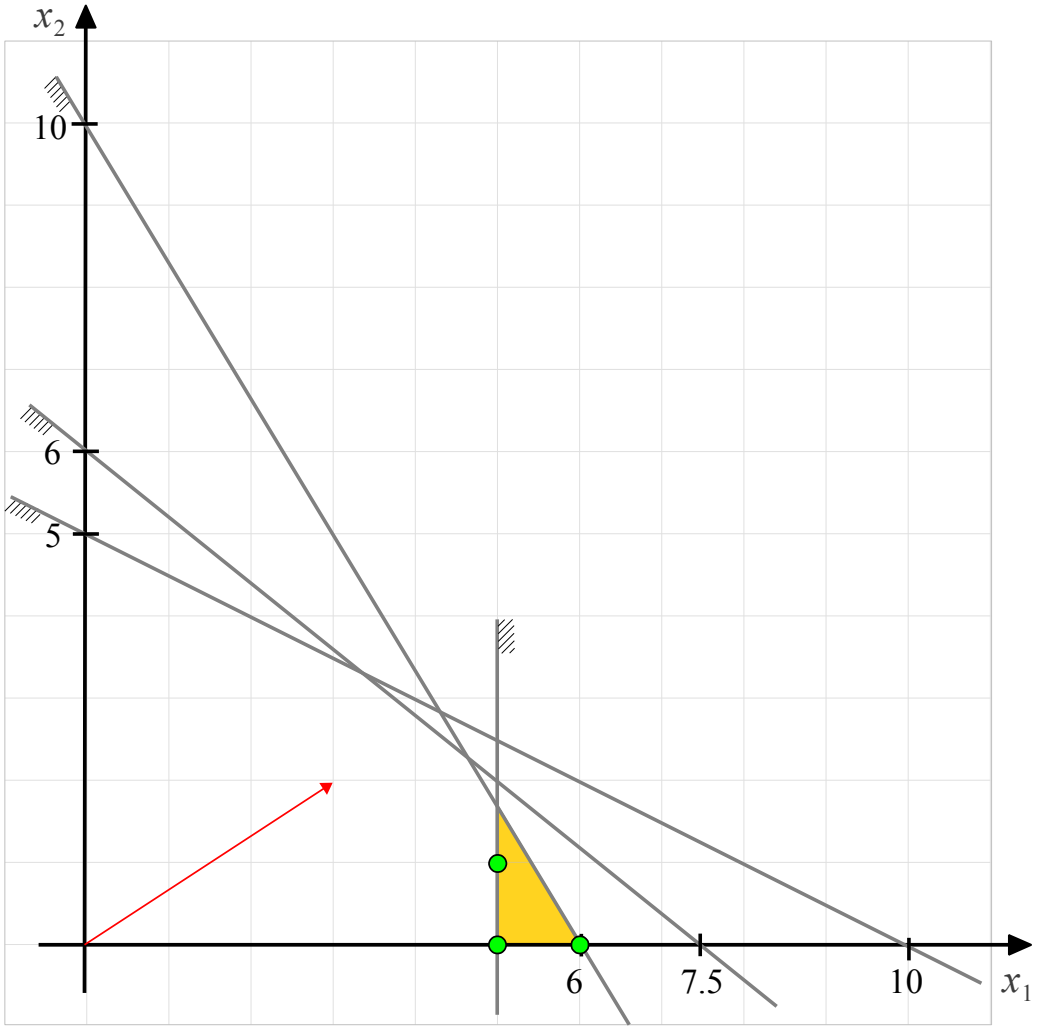
$$
\begin{aligned}
(S^2)\ \max\quad & f(x_1,x_2) = 3x_1 + 2x_2 \\
\text{s.a}\quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

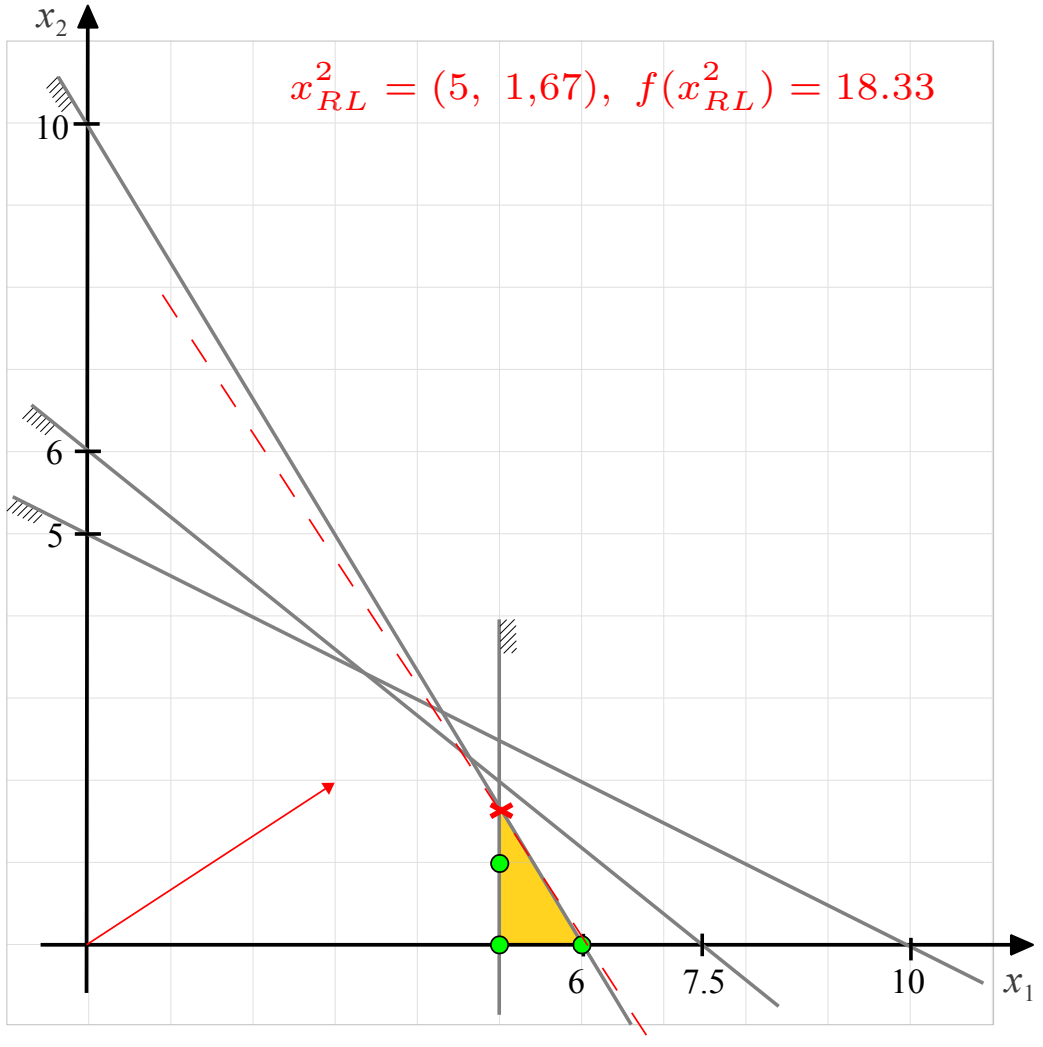
$$
\begin{aligned}
(S^2)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

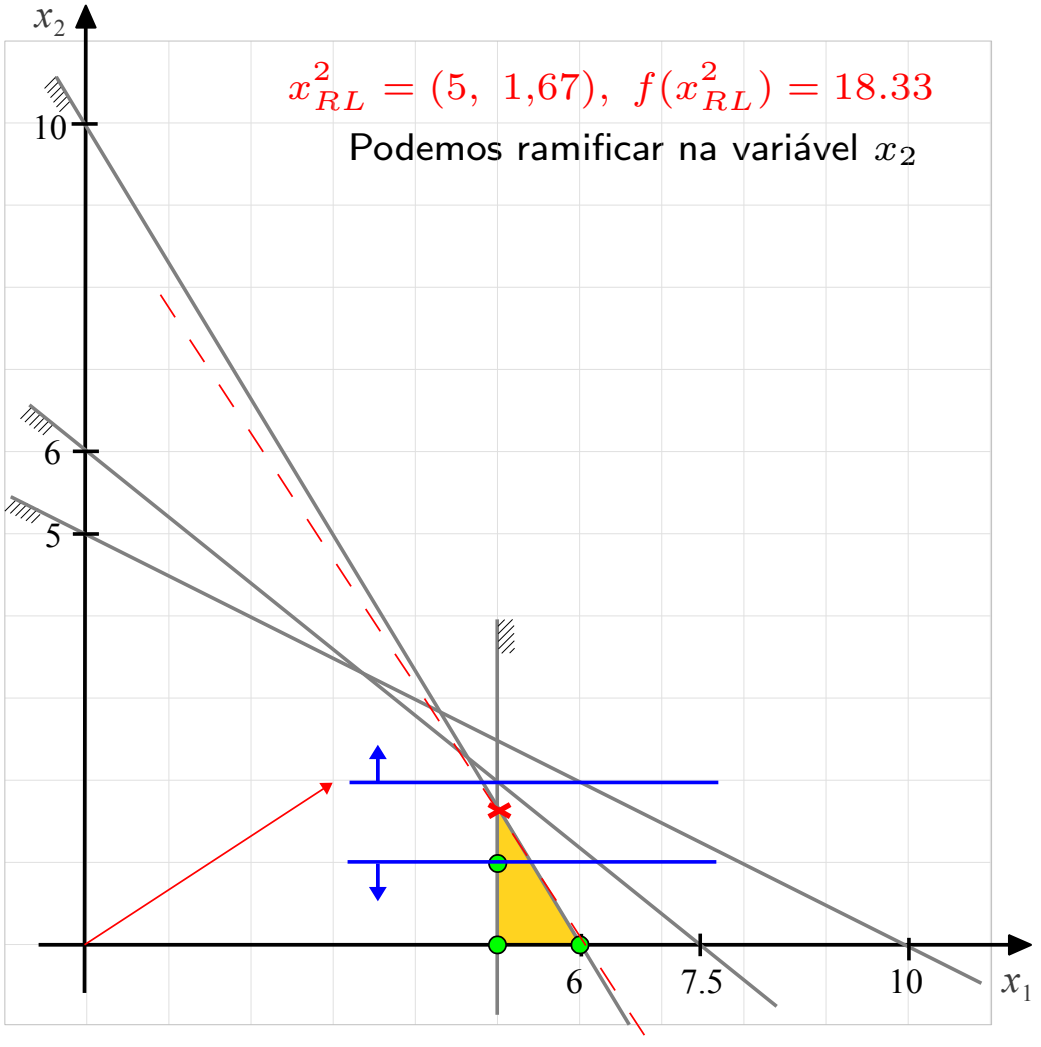
$$
\begin{aligned}
(S^2)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



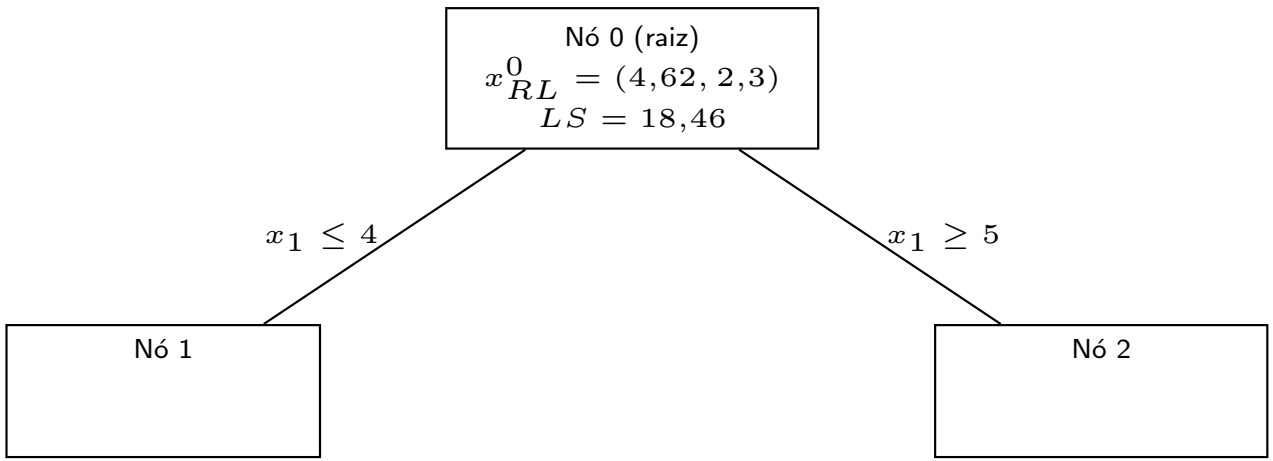
Método *branch-and-bound*

▷ Exemplo

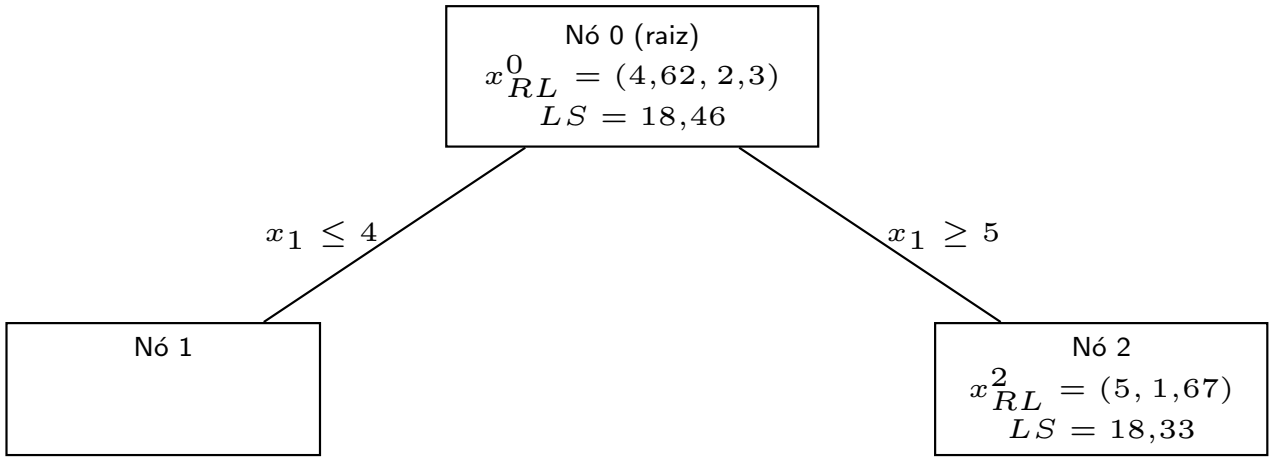
$$
\begin{aligned}
(S^2)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



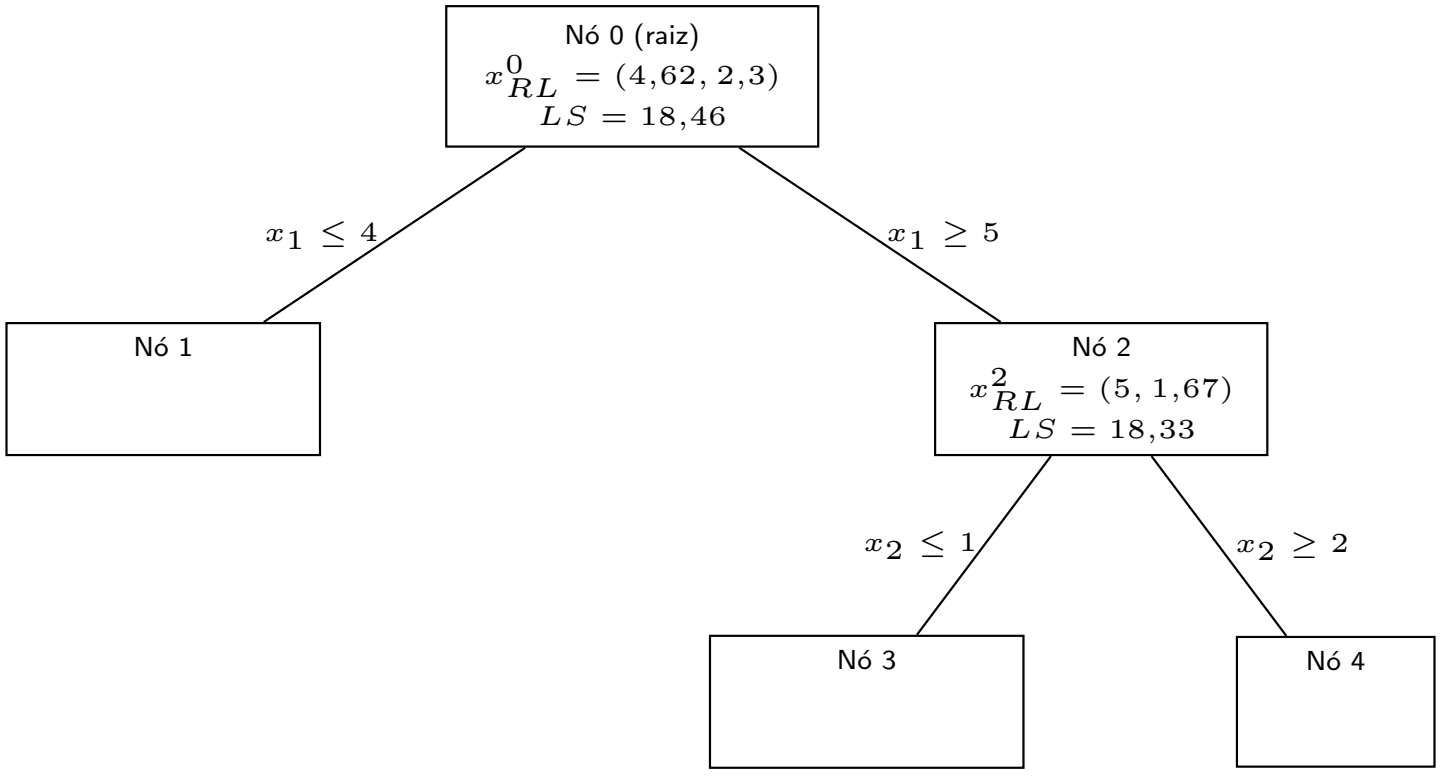
- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\not{\emptyset}, 1, 2\}$



- $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\not{\emptyset}, 1, \not{2}\}$



▷ $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\cancel{\emptyset}, 1, \cancel{2}, 3, 4\}$

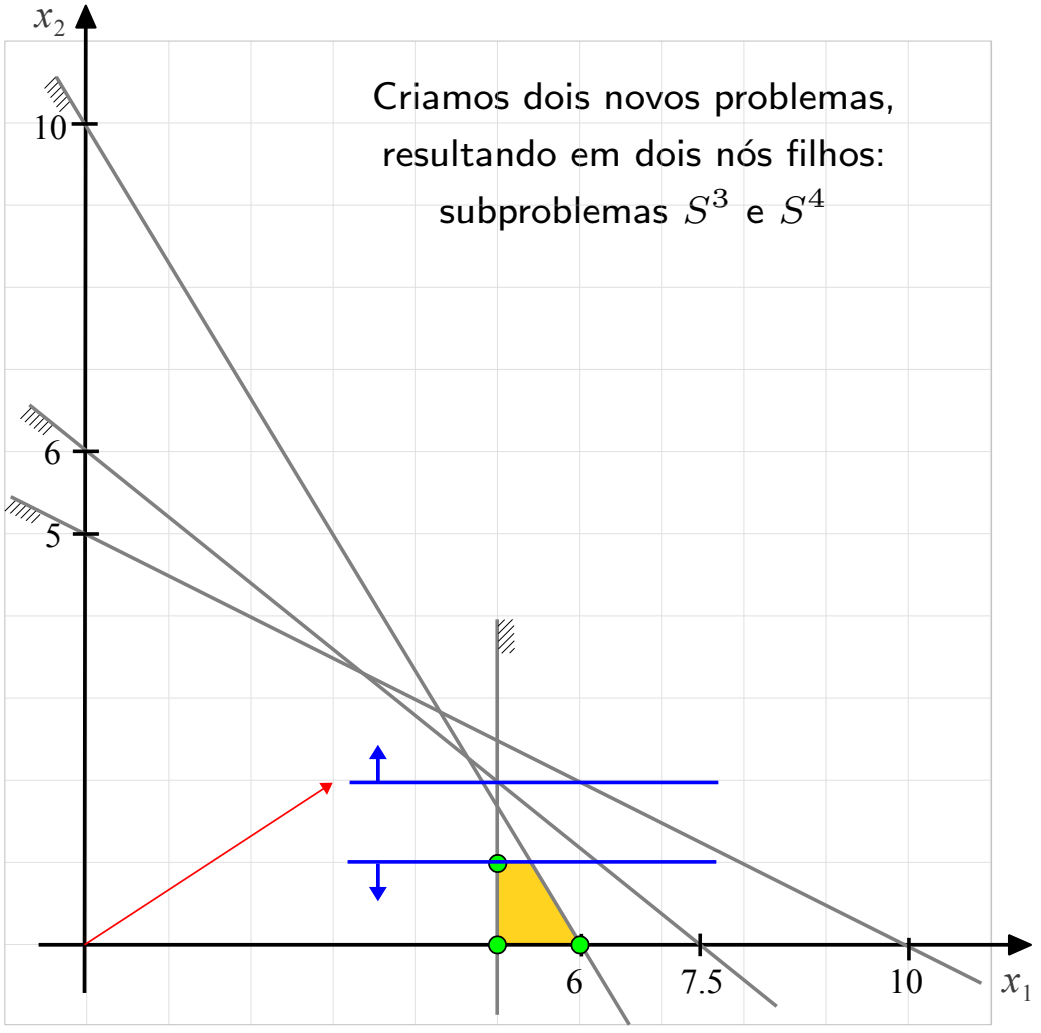


Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^3)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

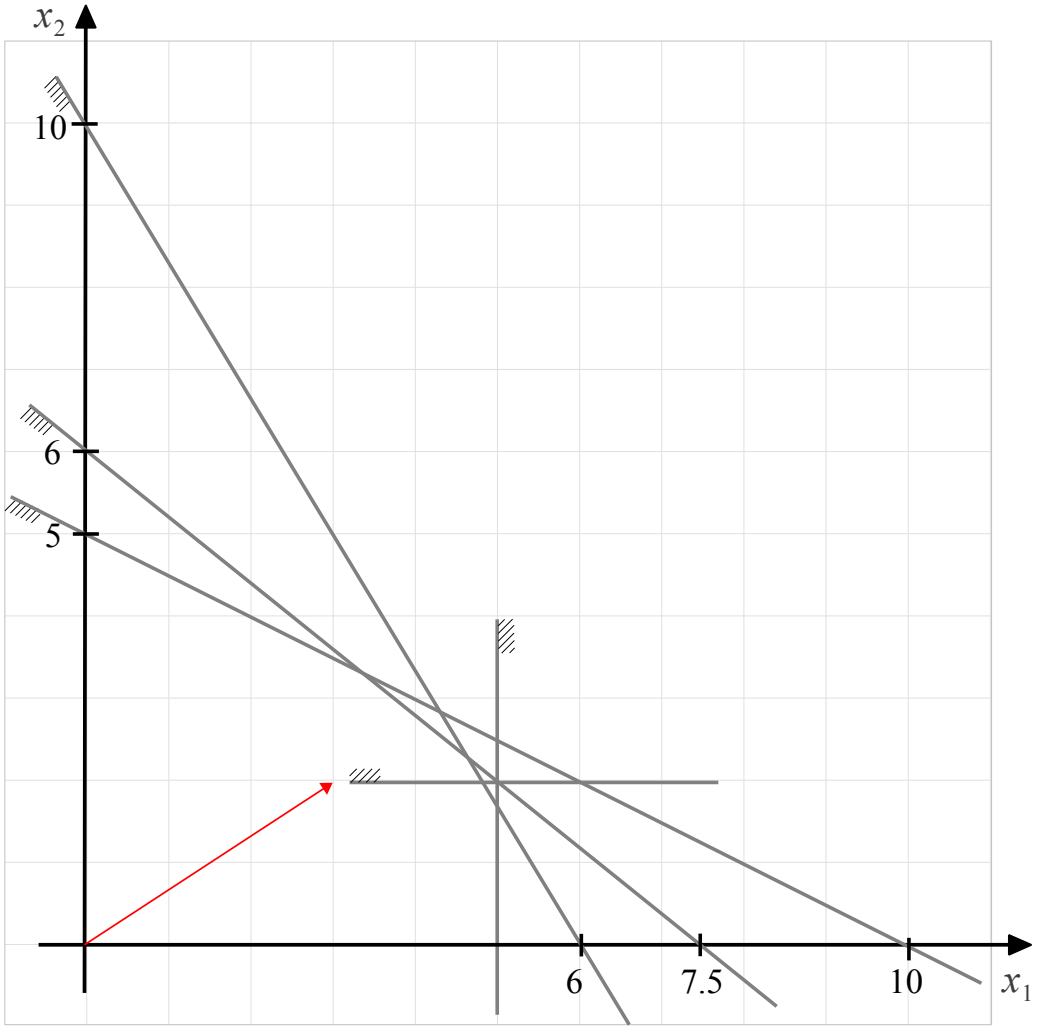
$$
\begin{aligned}
(S^4)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \geq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

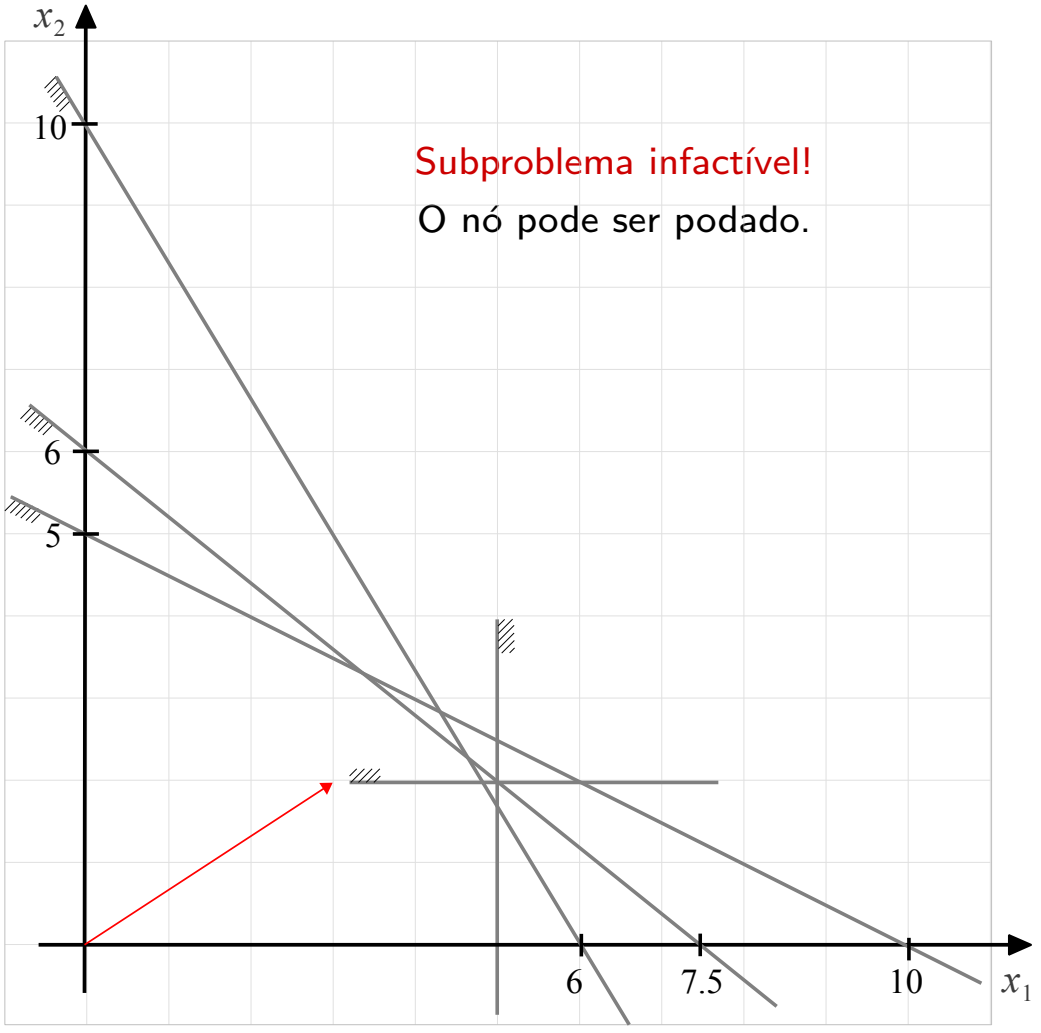
$$
\begin{aligned}
(S^4)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \geq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



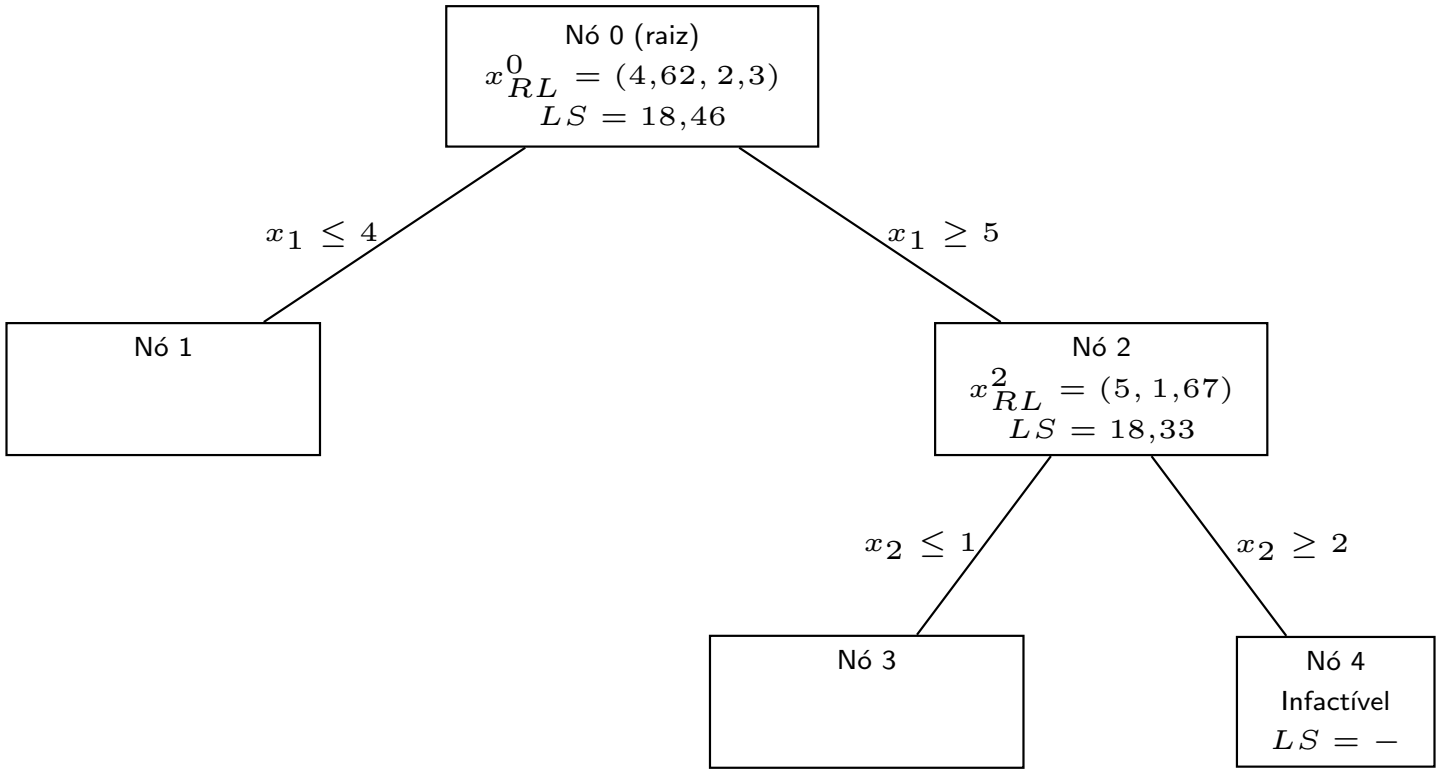
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^4)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \geq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



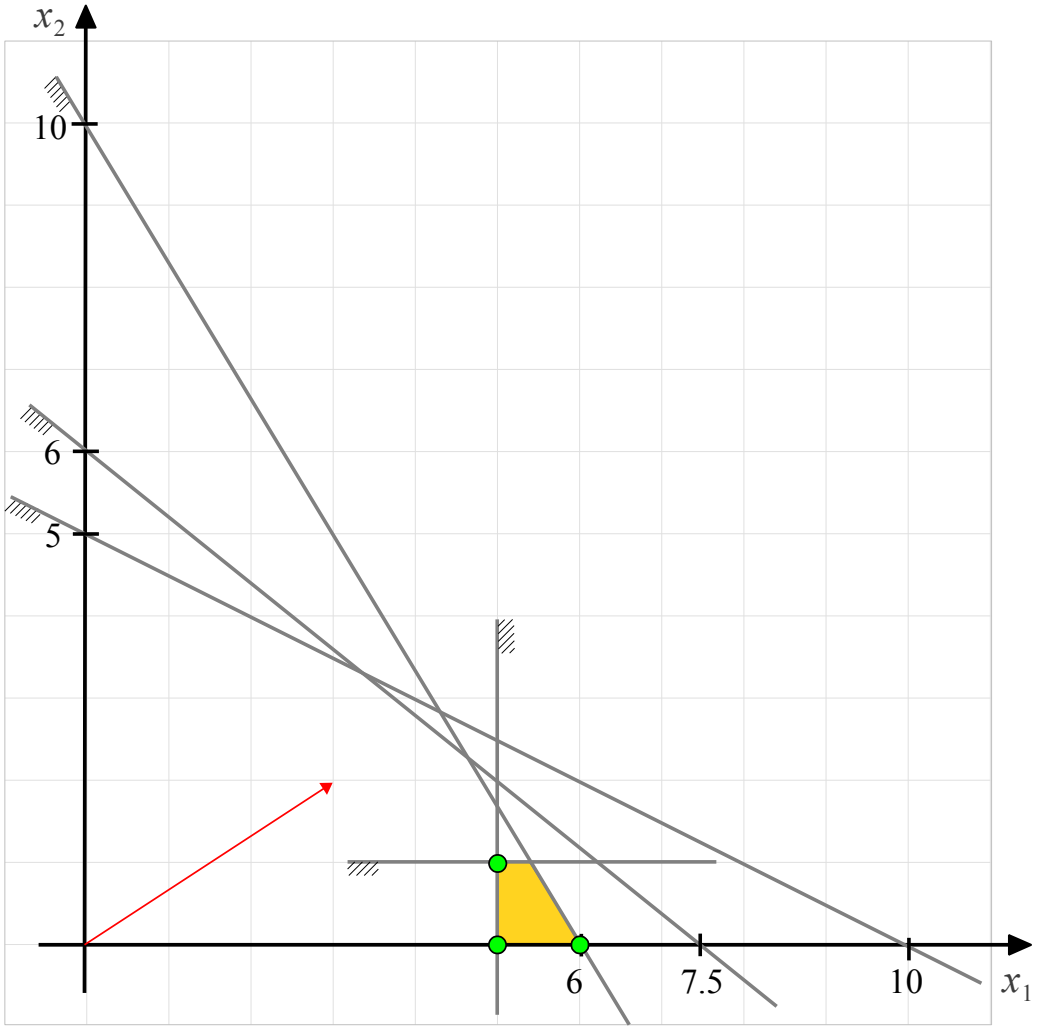
▷ $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\cancel{\emptyset}, 1, \cancel{2}, 3, \cancel{4}\}$



Método *branch-and-bound*

▷ Exemplo

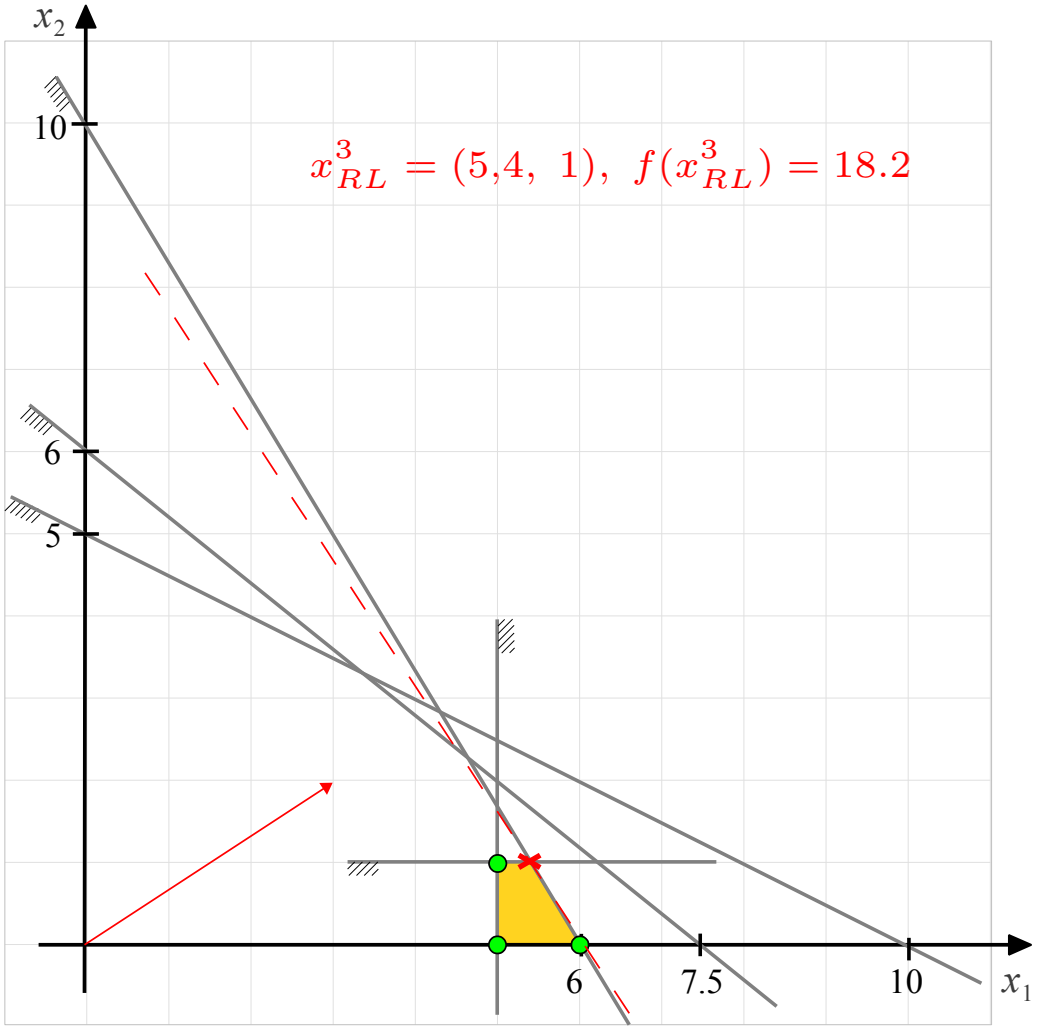
$$
\begin{aligned}
(S^3)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

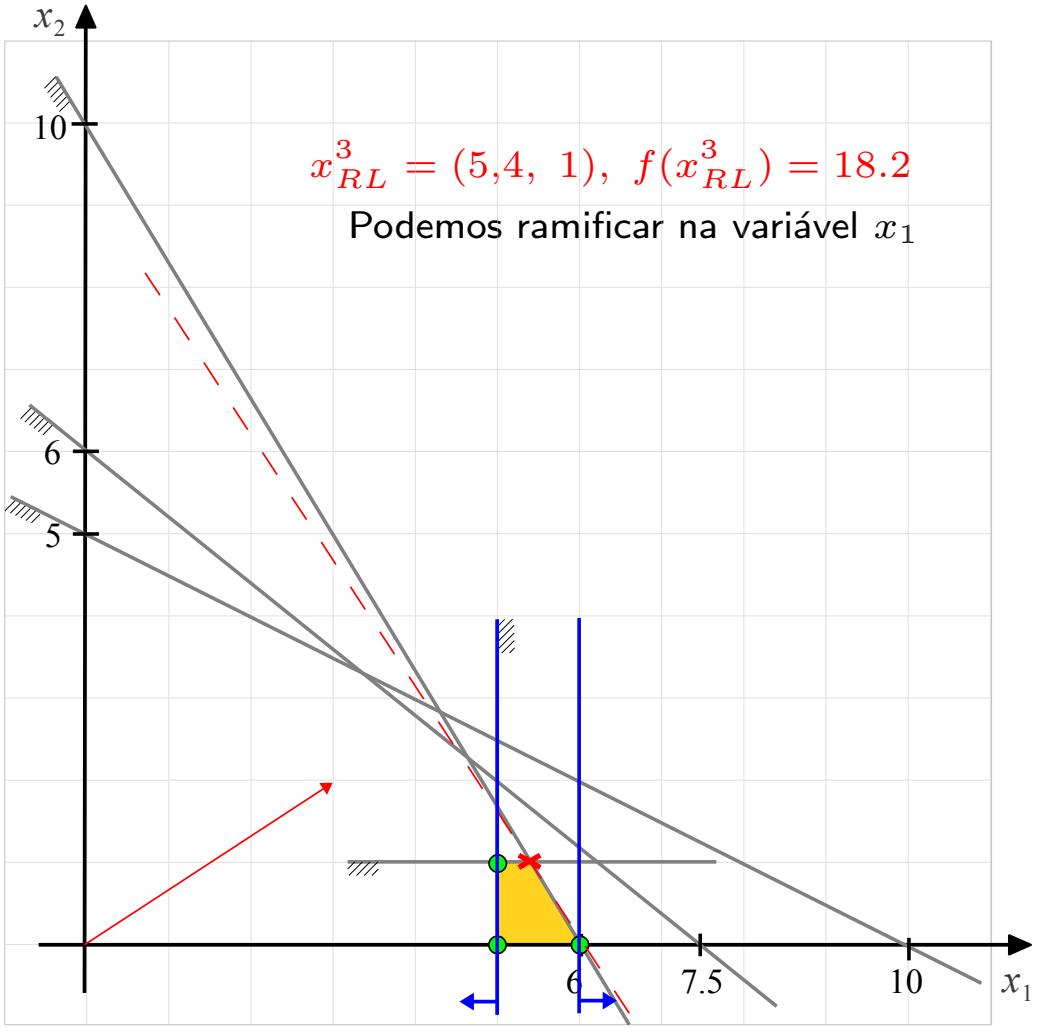
$$
\begin{aligned}
(S^3)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



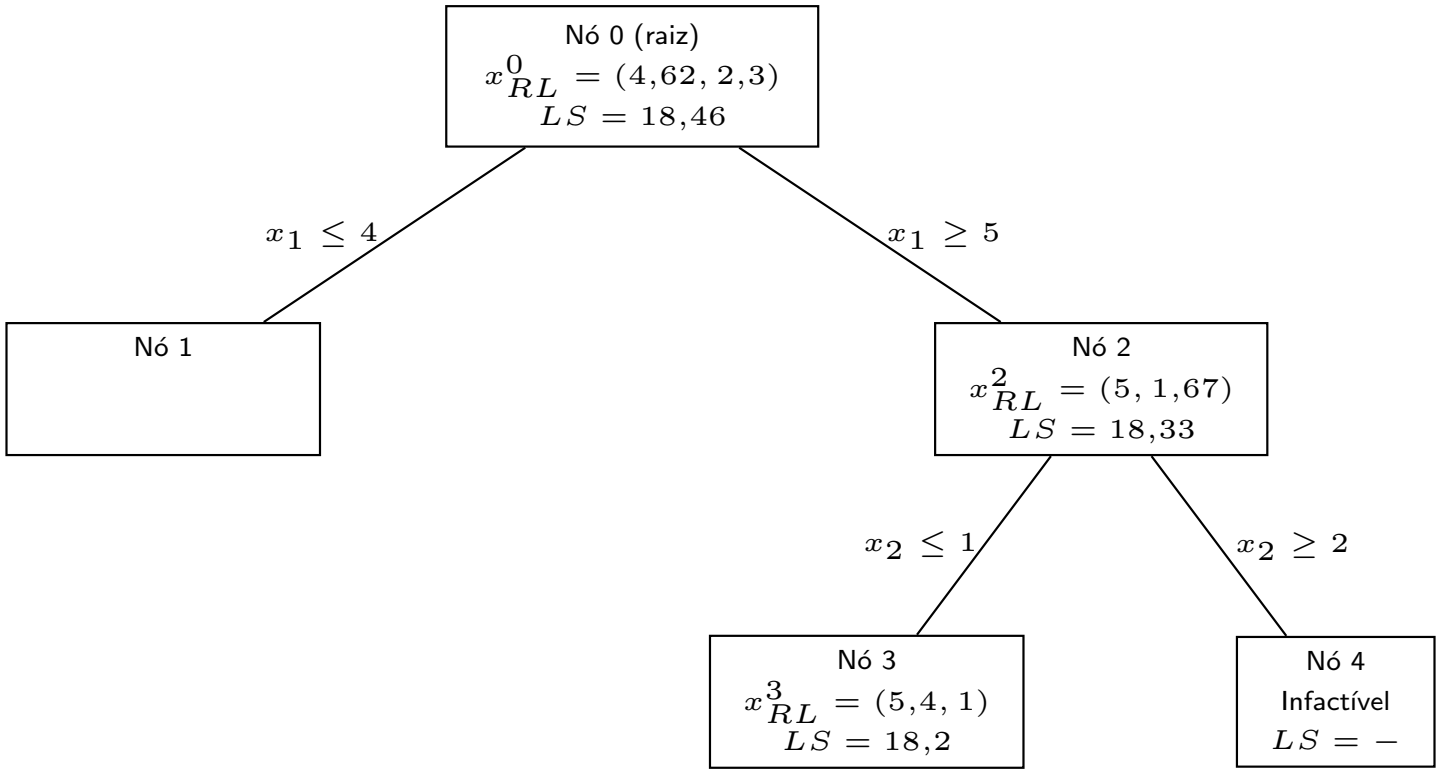
Método *branch-and-bound*

▷ Exemplo

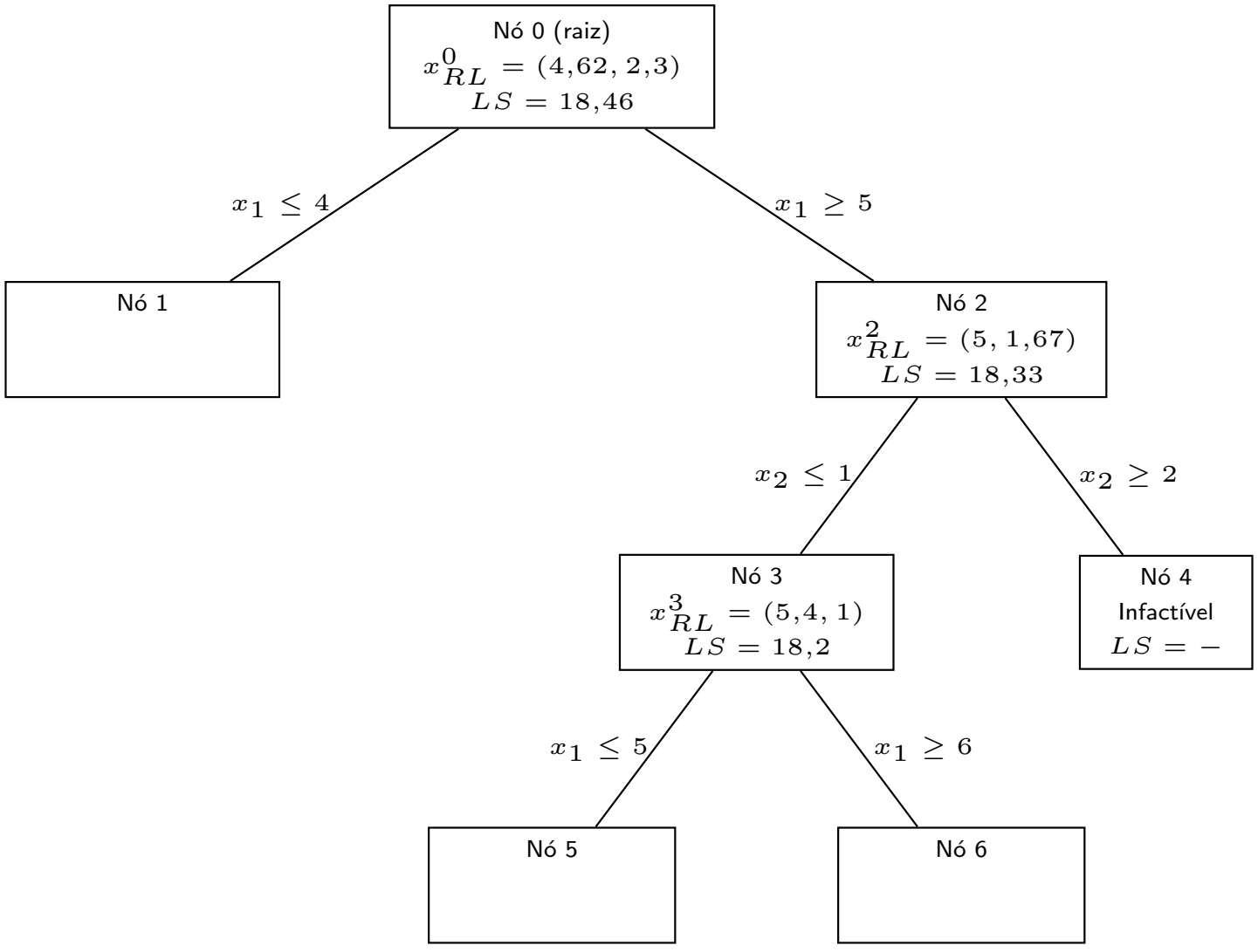
$$
\begin{aligned}
(S^3)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



▷ $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\emptyset, 1, \not{2}, \not{3}, \not{4}\}$



▷ $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\emptyset, 1, \not{2}, \not{3}, \not{4}, 5, 6\}$

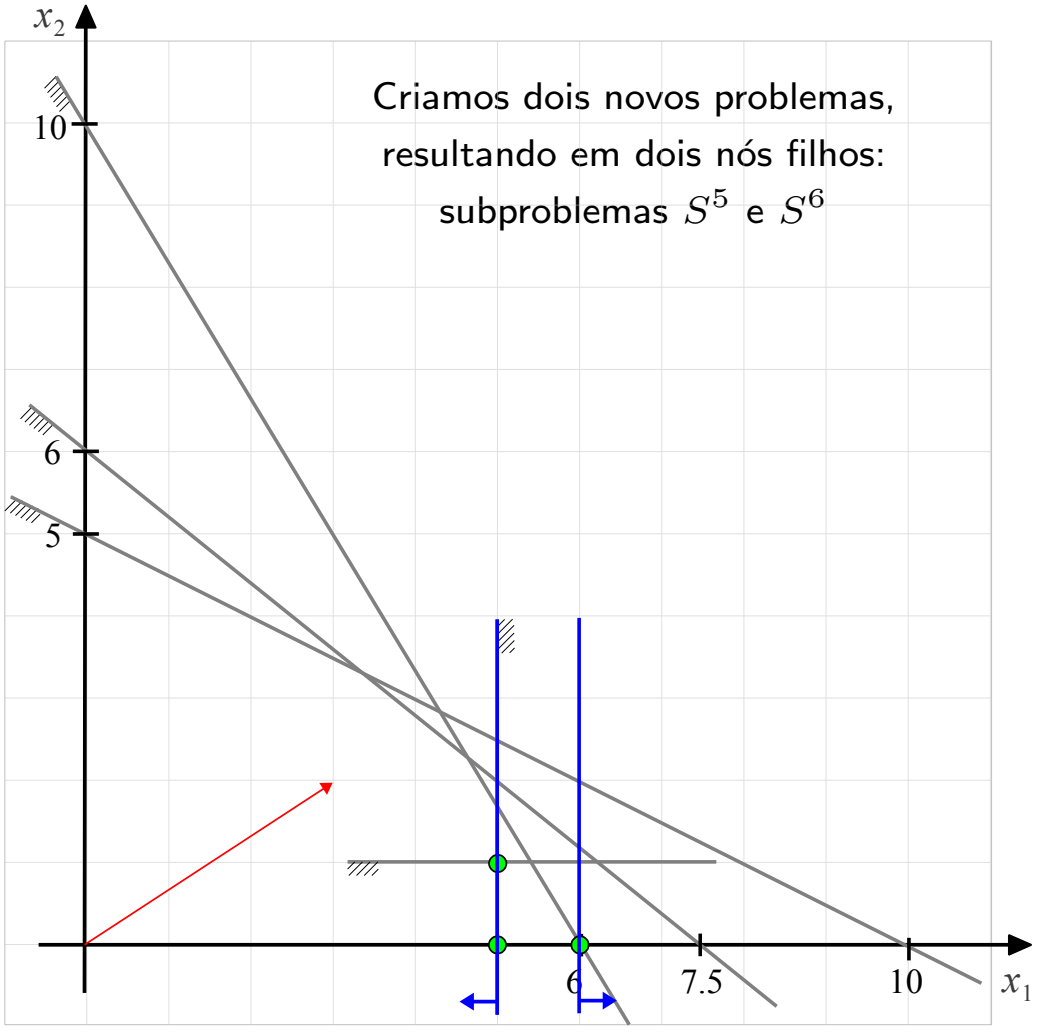


Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^5)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \leq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

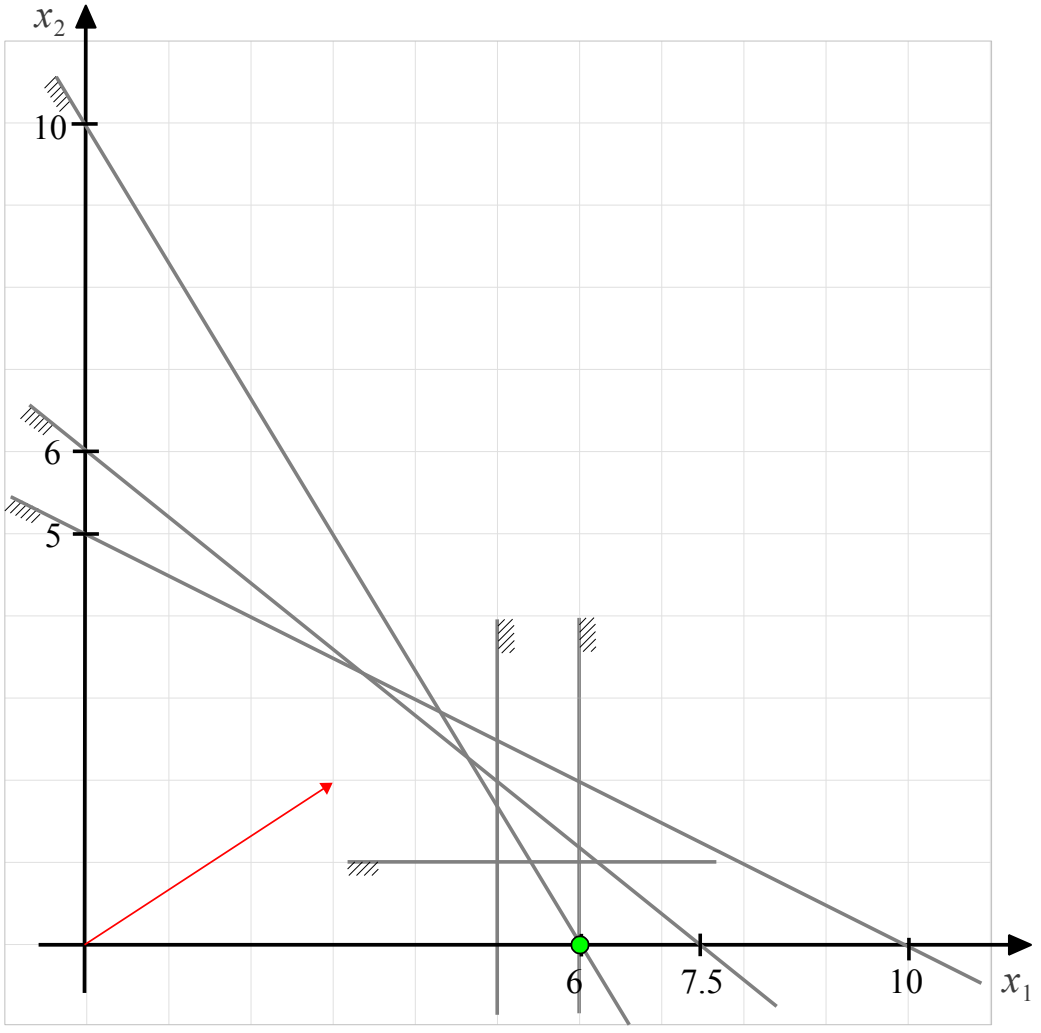
$$
\begin{aligned}
(S^6)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

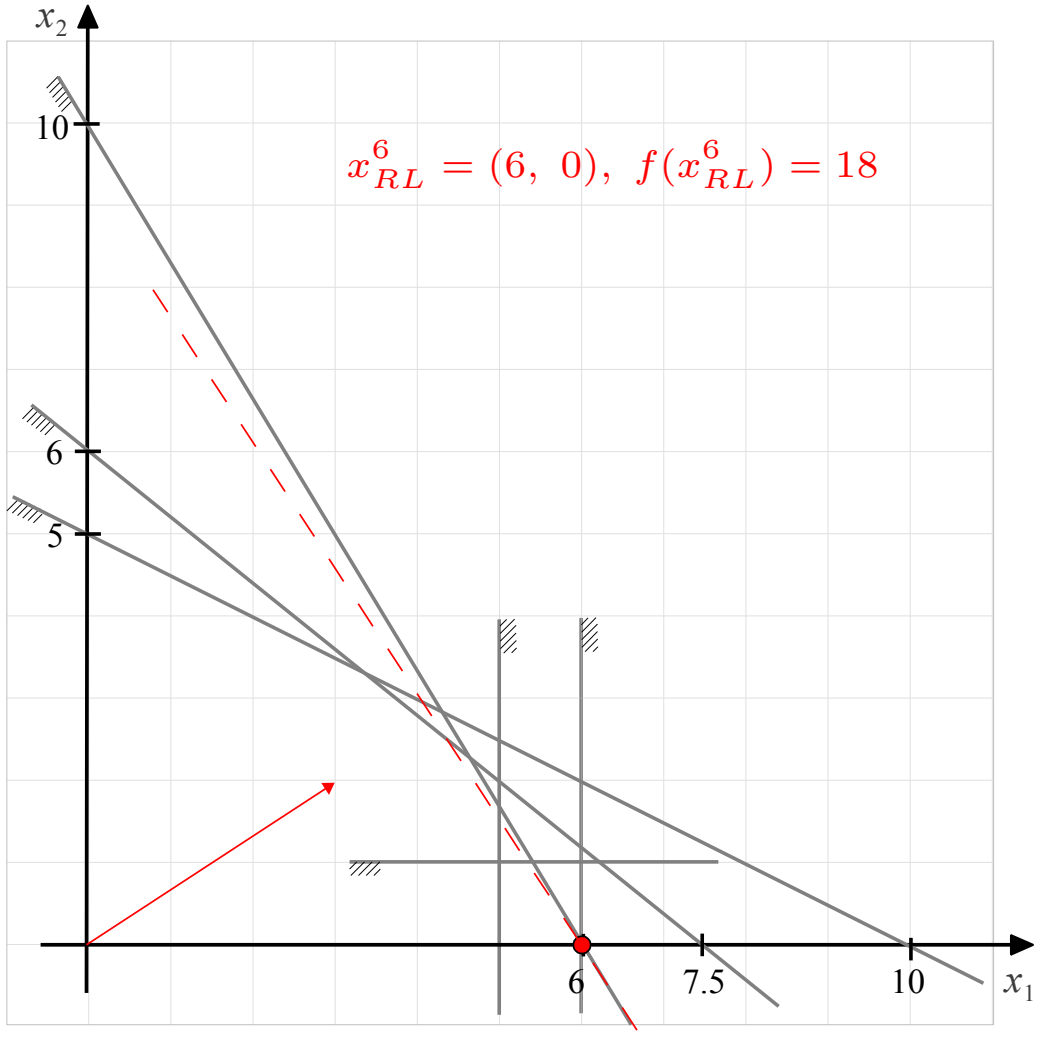
$$
\begin{aligned}
(S^6)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

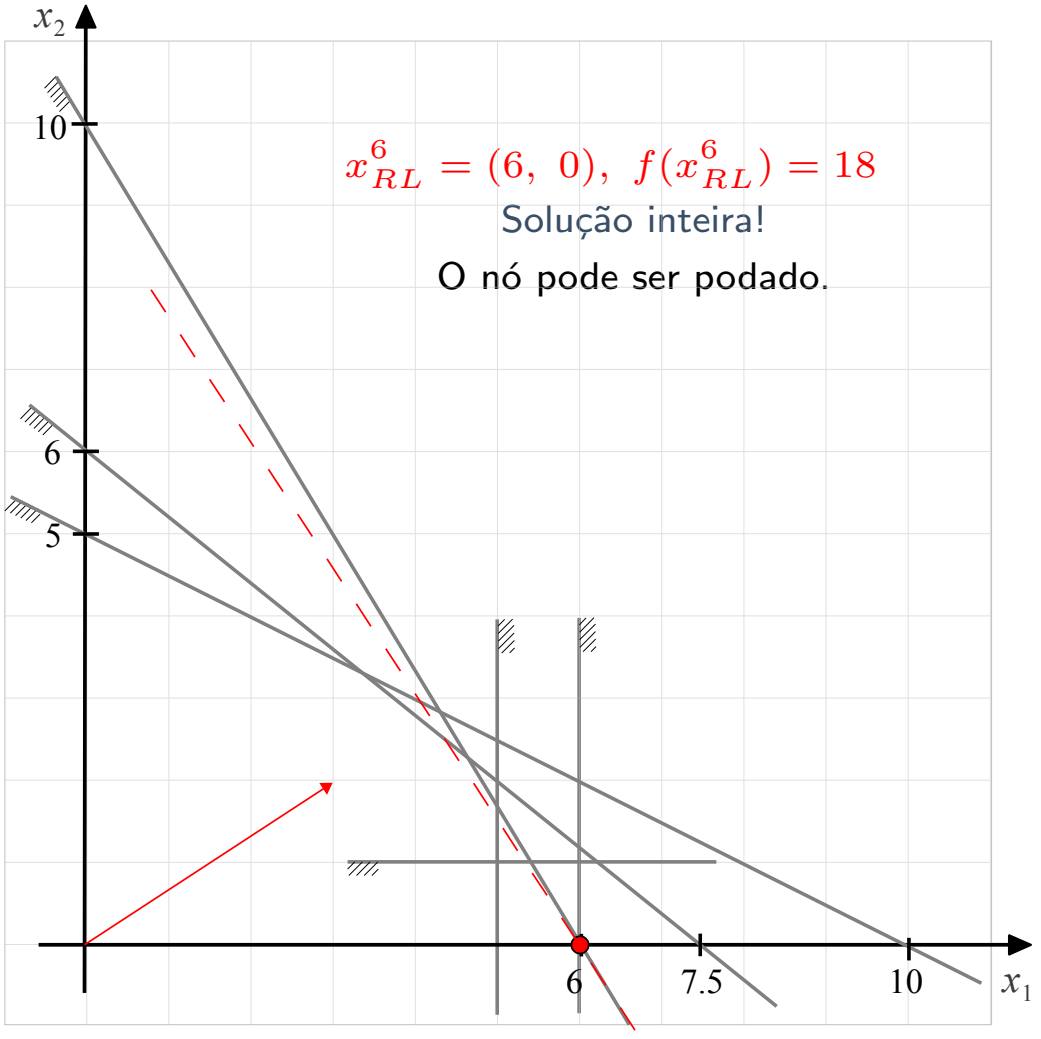
$$
\begin{aligned}
(S^6)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



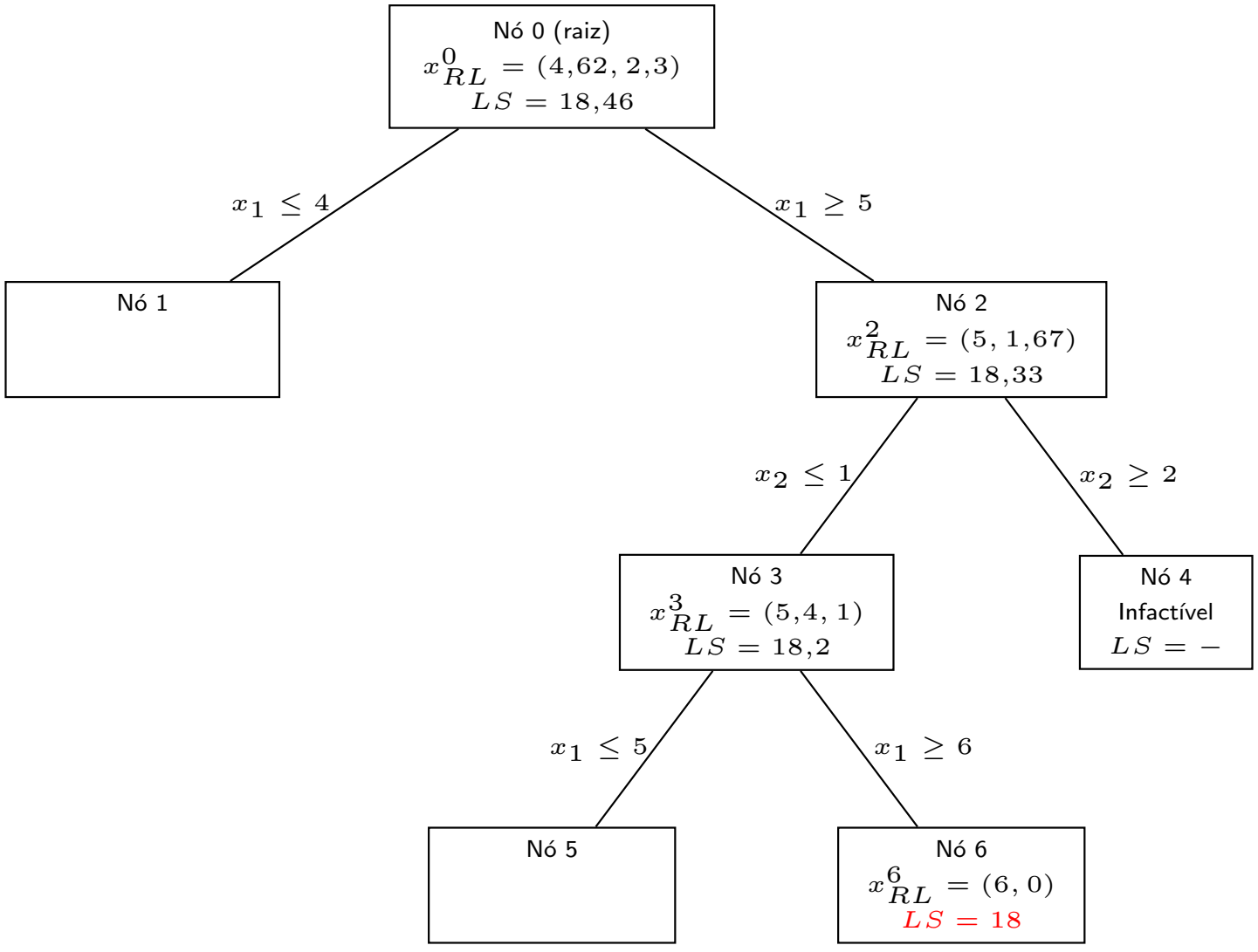
Método *branch-and-bound*

▷ Exemplo

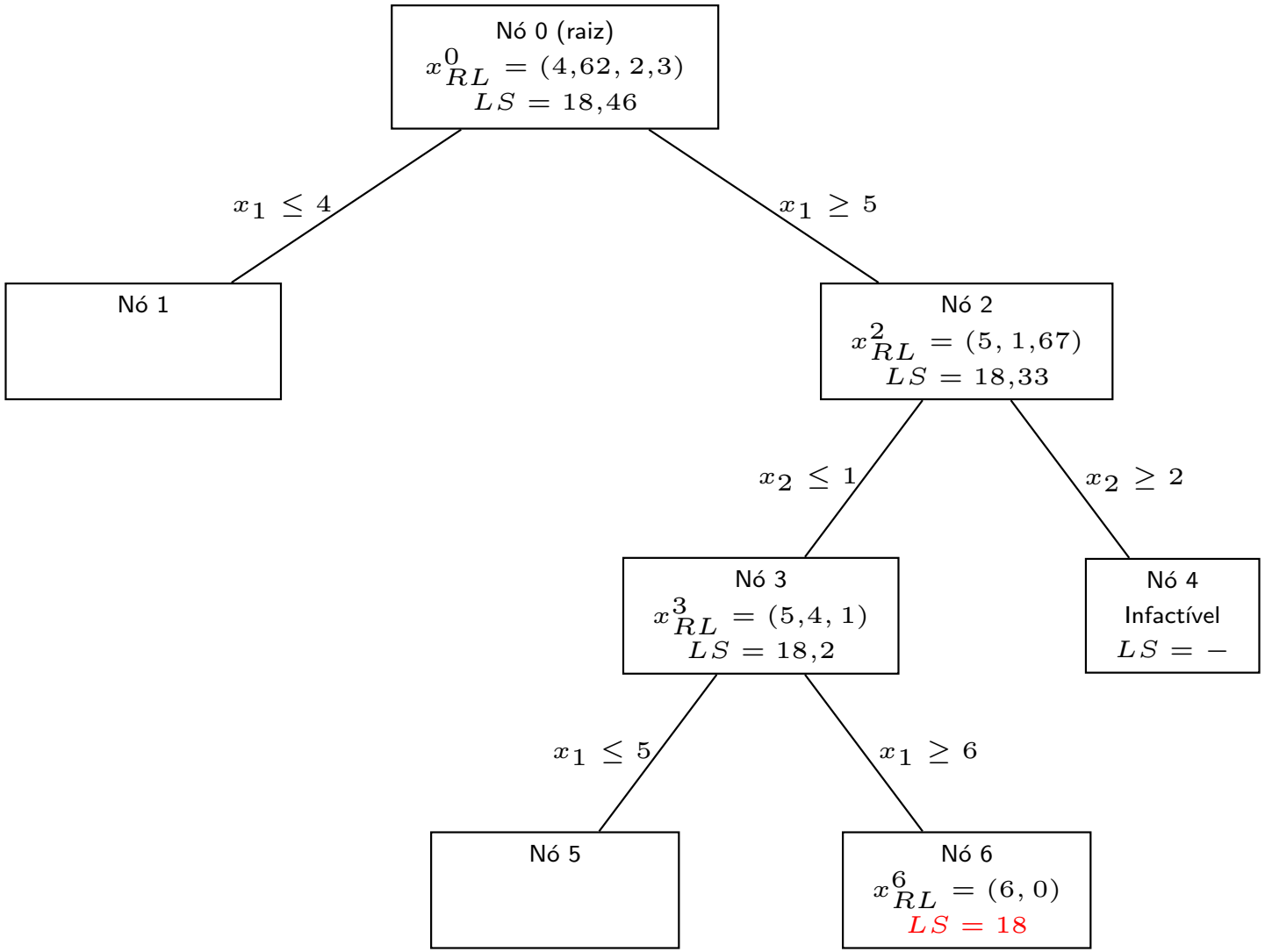
$$
\begin{aligned}
(S^6)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



▷ $x^\star = \varnothing$; $LI = -\infty$; Nós: $\{\not{0}, 1, \not{2}, \not{3}, \not{4}5, \not{6}\}$



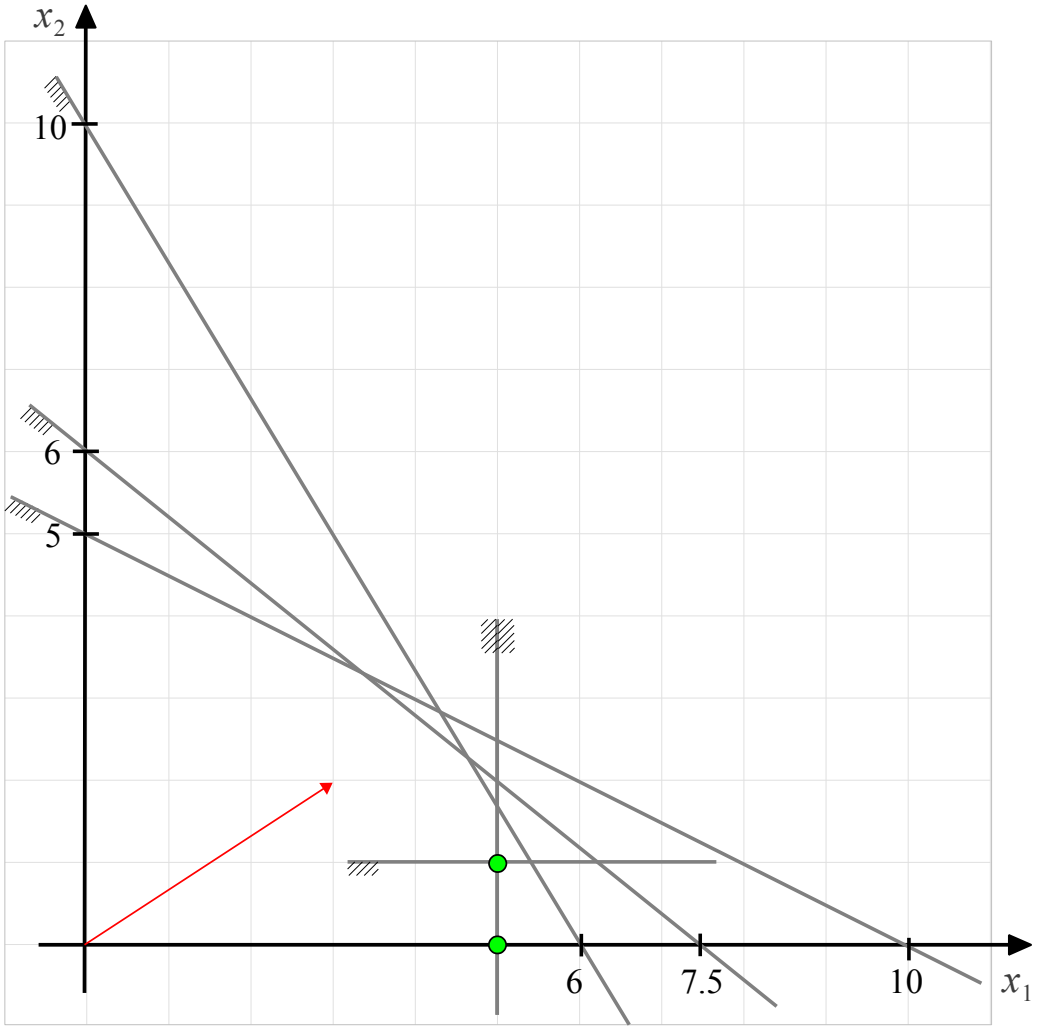
▷ $x^\star = x^6_{RL} = (6, 0)$; $LI = 18$; Nós: $\{\cancel{\emptyset}, 1, \cancel{2}, \cancel{3}, \cancel{4}, 5, \cancel{6}\}$



Método *branch-and-bound*

▷ Exemplo

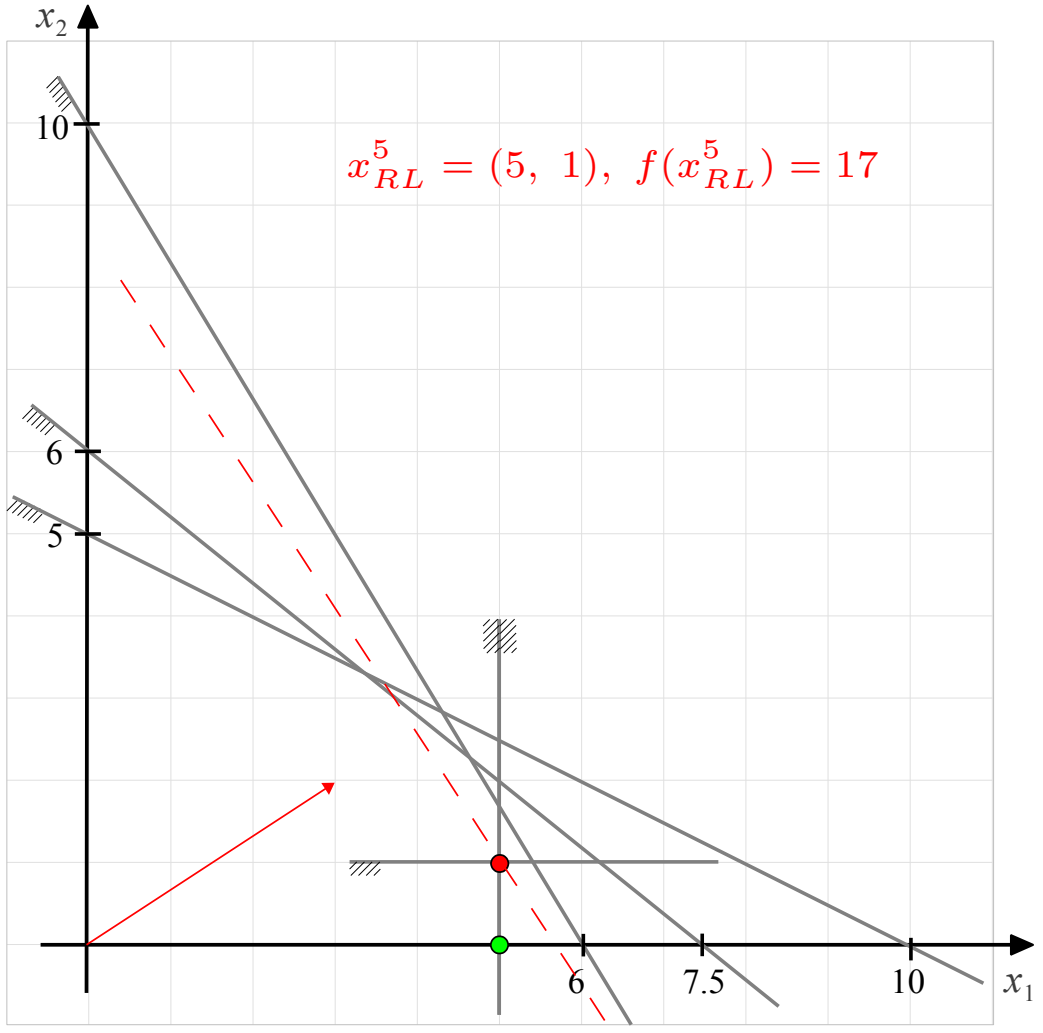
$$
\begin{aligned}
(S^5)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \leq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



Método *branch-and-bound*

▷ Exemplo

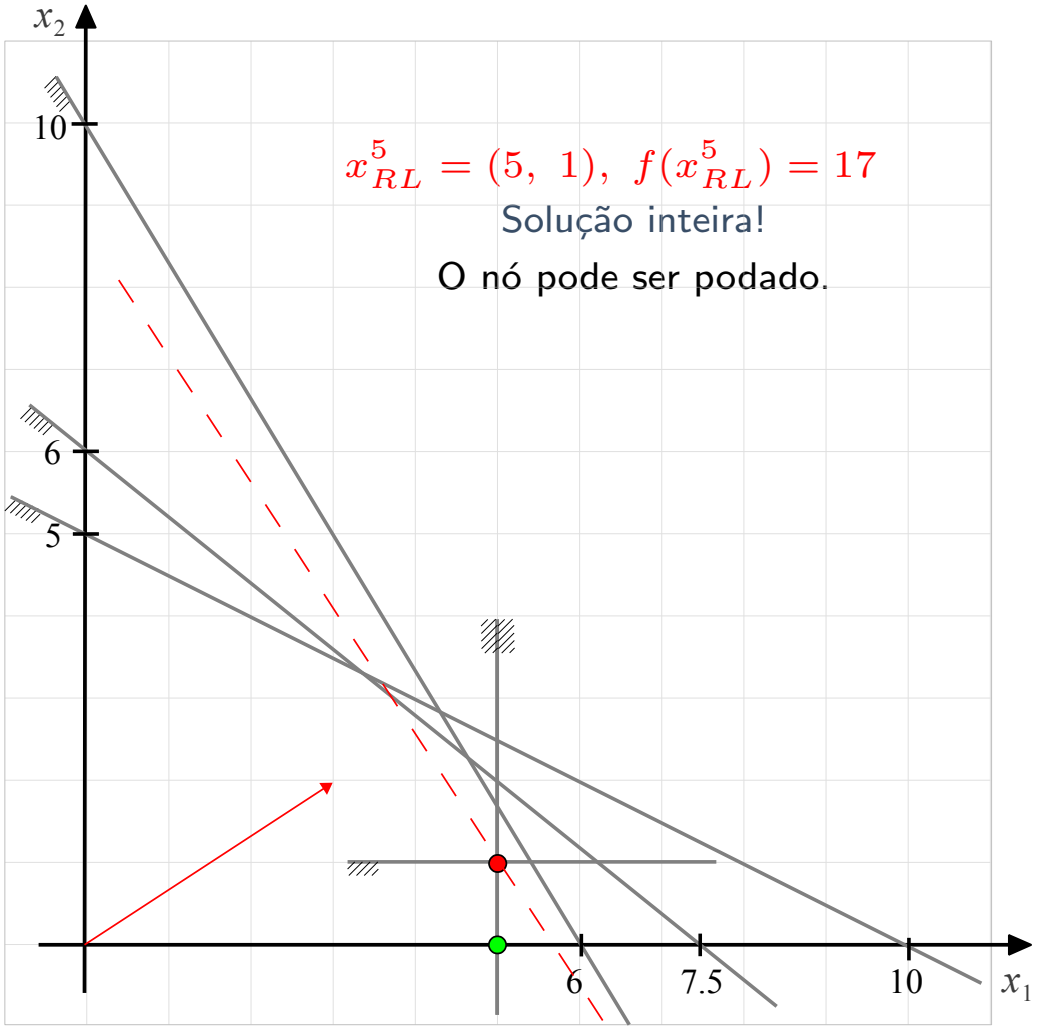
$$
\begin{aligned}
(S^5)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \leq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



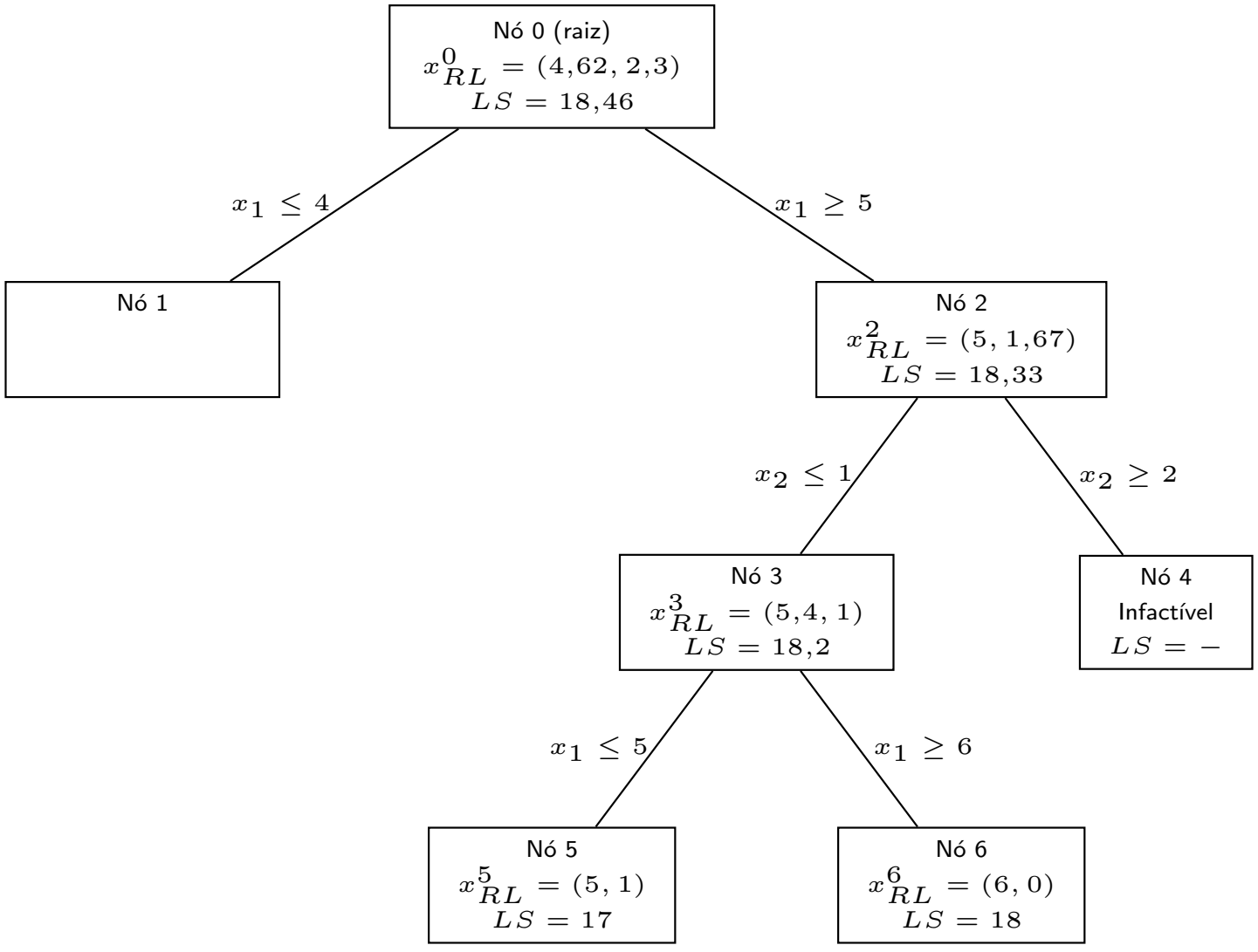
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^5)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \geq 5, x_2 \leq 1, x_1 \leq 5 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



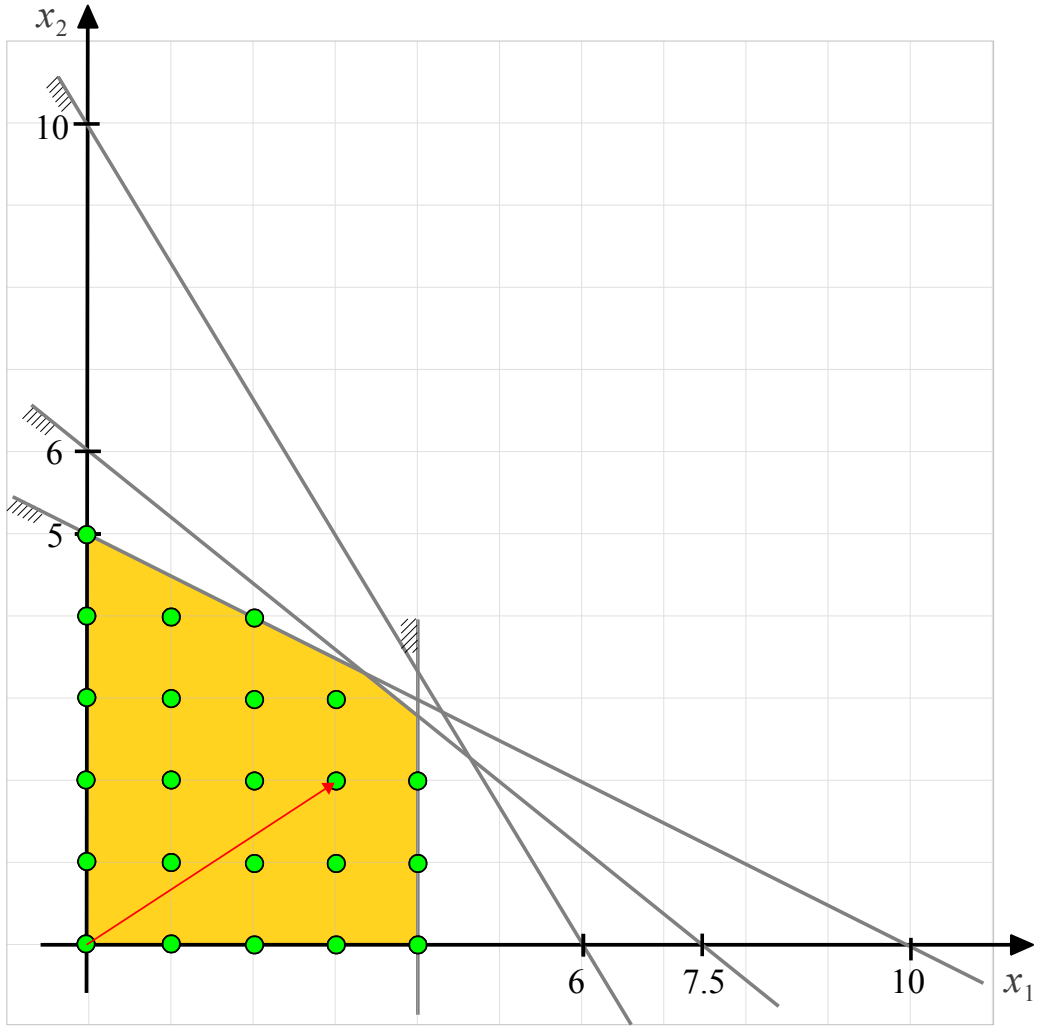
▷ $x^\star = x^6_{RL} = (6, 0)$; $LI = 18$; Nós: $\{\not{0}, 1, \not{2}, \not{3}, \not{4}, \not{5}, \not{6}\}$



Método *branch-and-bound*

▷ Exemplo

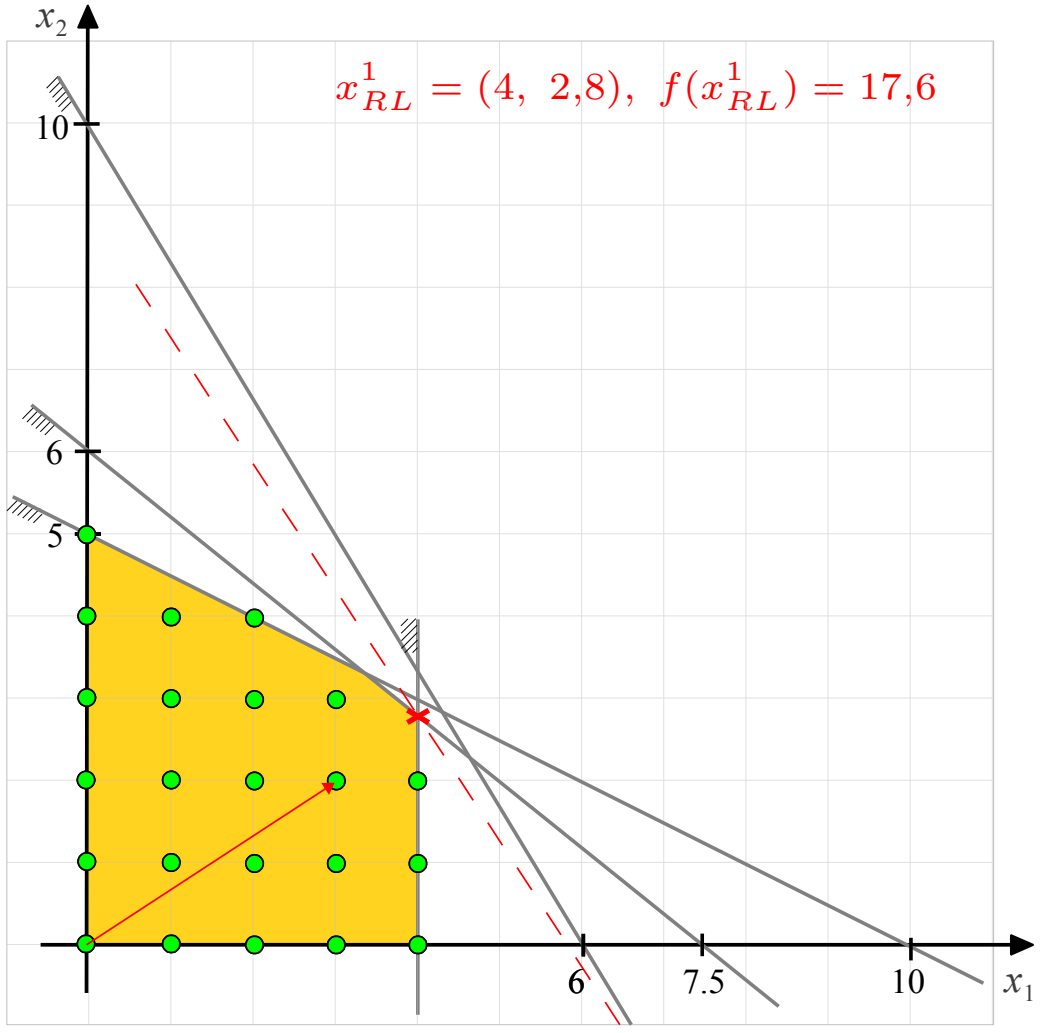
$$
\begin{aligned}
(S^1)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \leq 4 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



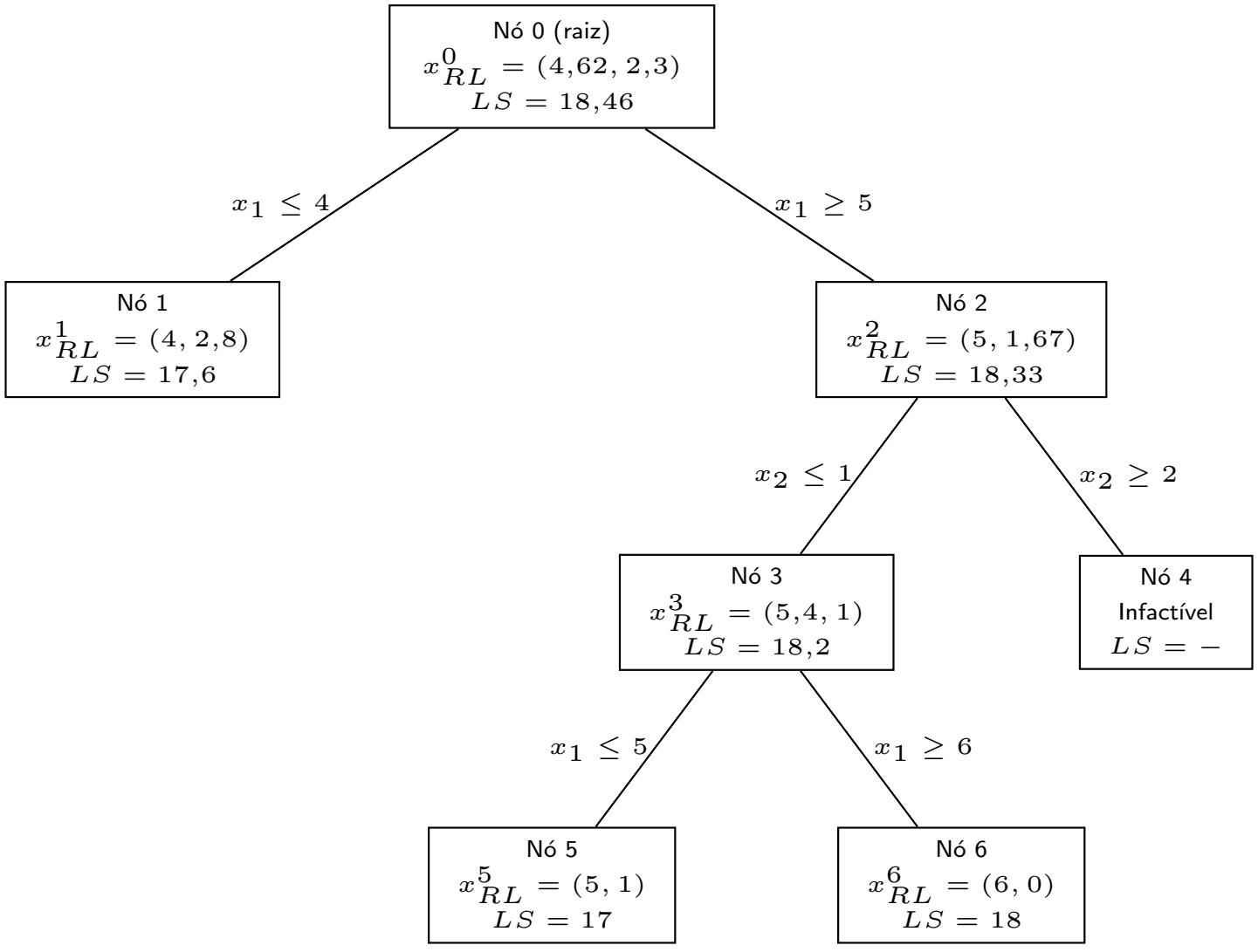
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^1)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \leq 4 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



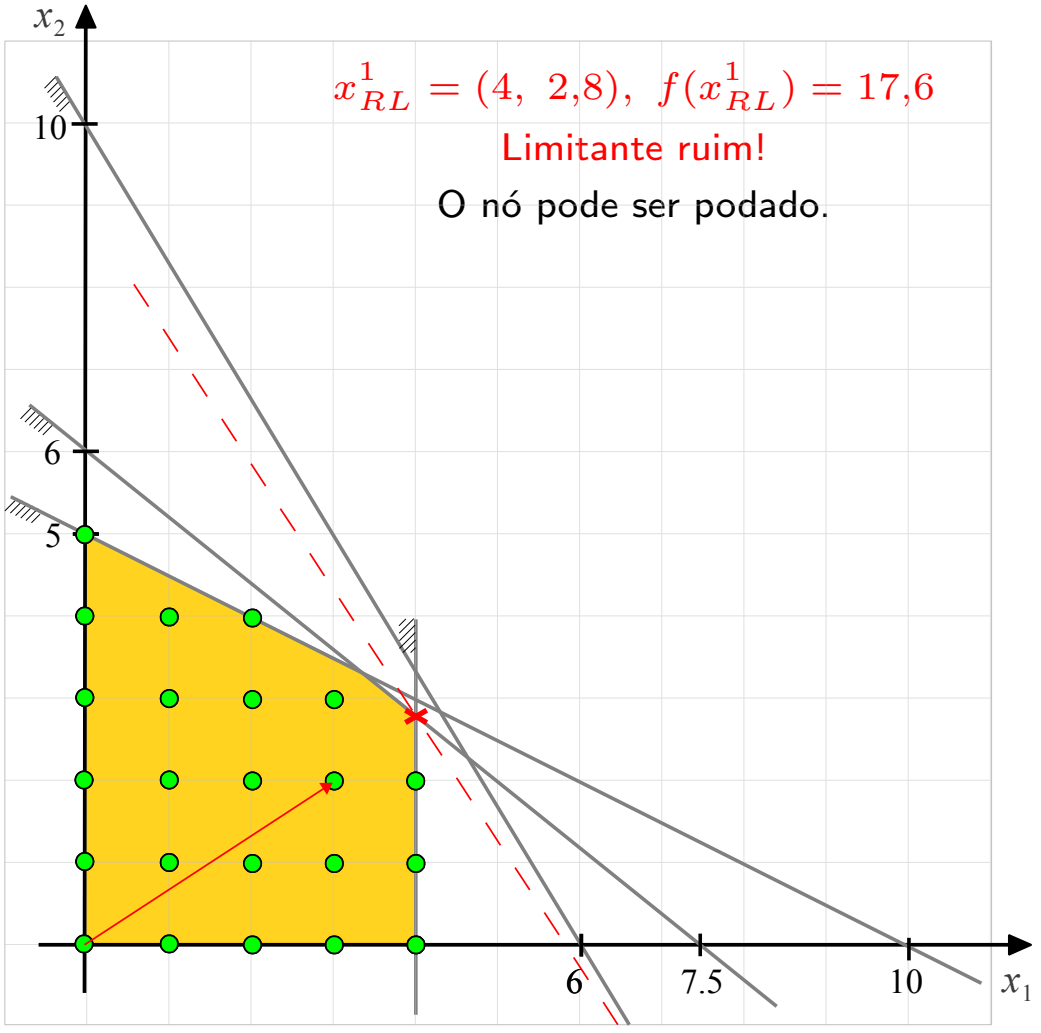
▷ $x^\star = x^6_{RL} = (6, 0)$; $LI = 18$; Nós: $\{\cancel{0}, \cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}\}$



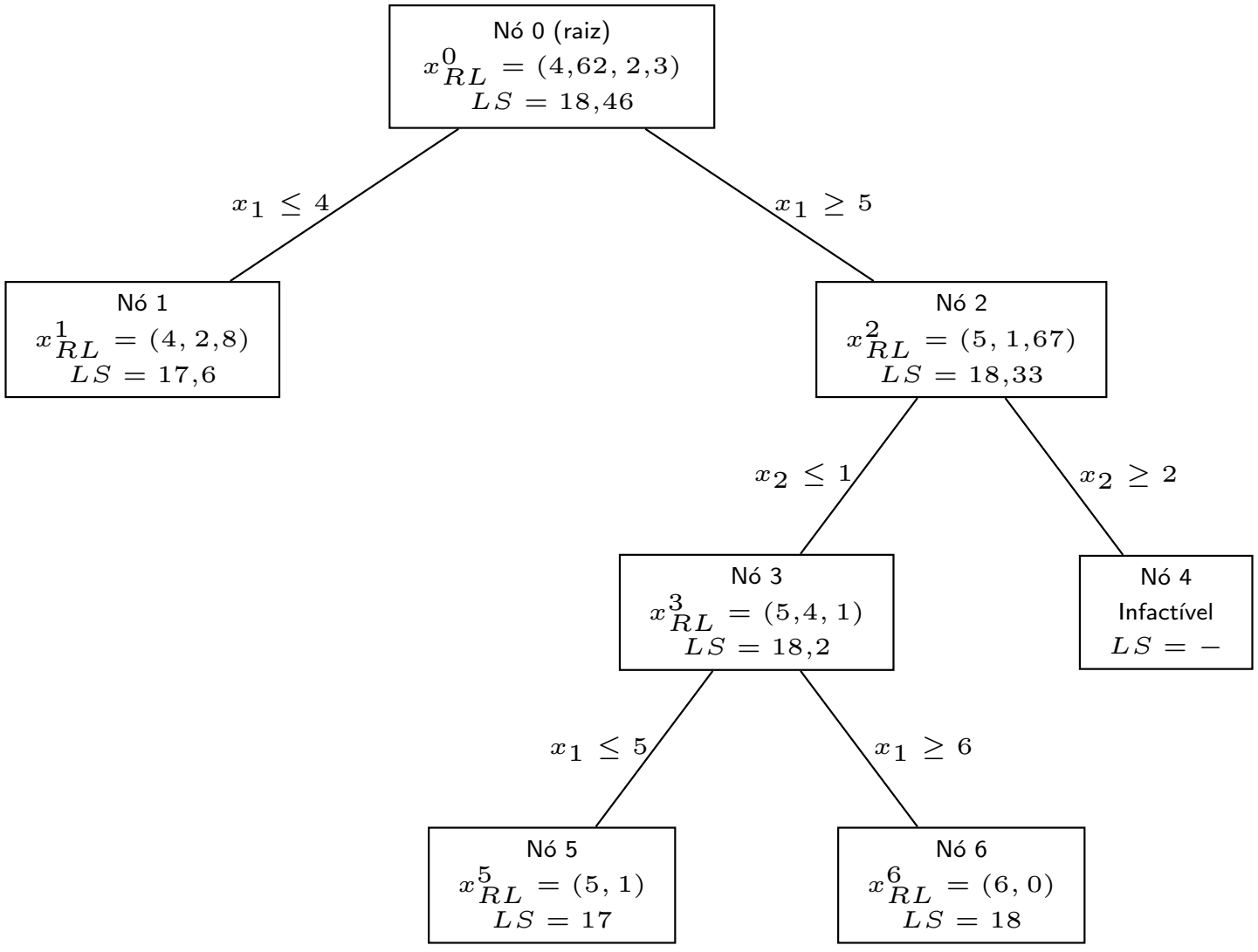
Método *branch-and-bound*

▷ Exemplo

$$
\begin{aligned}
(S^1)\ \max \quad & f(x_1, x_2) = 3x_1 + 2x_2 \\
\text{s.a} \quad & 0{,}5x_1 + 0{,}3x_2 \leq 3 \\
& 0{,}1x_1 + 0{,}2x_2 \leq 1 \\
& 0{,}4x_1 + 0{,}5x_2 \leq 3 \\
& x_1 \leq 4 \\
& x_1, x_2 \geq 0
\end{aligned}
$$



▷ $x^\star = x^6_{RL} = (6, 0)$; $LI = 18$; Nós: $\{\cancel{0}, \cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}\} \Rightarrow$ FIM!



Método *branch-and-bound*

- Critérios de eliminação de nós

Método *branch-and-bound*

▷ Critérios de eliminação de nós

O i-ésimo nó da árvore *branch-and-bound* pode ser podado por:

Método *branch-and-bound*

▷ Critérios de eliminação de nós

O i-ésimo nó da árvore *branch-and-bound* pode ser podado por:

1. Infactibilidade: S^i é infactível. A região factível do subproblema é vazia e, assim, a solução ótima não pode ser obtida no ramo atual.

Método *branch-and-bound*

▷ Critérios de eliminação de nós

O i-ésimo nó da árvore *branch-and-bound* pode ser podado por:

1. Infactibilidade: S^i é infactível. A região factível do subproblema é vazia e, assim, a solução ótima não pode ser obtida no ramo atual.
2. Qualidade: $f(x^i_{RL}) \leq LI$ em um problema de maximização (minimização), ou $f(x^i_{RL}) \geq LS$ em um problema de minimização. Assim, mesmo que outras soluções inteiras possam ser obtidas no ramo atual, elas não terão valor melhor que o da solução incumbente.

Método *branch-and-bound*

▷ Critérios de eliminação de nós

O i-ésimo nó da árvore *branch-and-bound* pode ser podado por:

1. Infactibilidade: S^i é infactível. A região factível do subproblema é vazia e, assim, a solução ótima não pode ser obtida no ramo atual.
2. Qualidade: $f(x^i_{RL}) \leq LI$ em um problema de maximização (minimização), ou $f(x^i_{RL}) \geq LS$ em um problema de minimização. Assim, mesmo que outras soluções inteiras possam ser obtidas no ramo atual, elas não terão valor melhor que o da solução incumbente.
3. Otimalidade: x^i_{RL} satisfaz as restrições de integralidade e, portanto, não há como ramificar.

Método *branch-and-bound*

- Algoritmo (maximização)

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$,

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$,

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$,

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Passo 7: Se a solução ótima x^i_{RL} satisfaz as restrições de integralidade, então

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Passo 7: Se a solução ótima x^i_{RL} satisfaz as restrições de integralidade, então
Se $f(x^i_{RL}) > LI$ então $x^\star = x^i_{RL}$ e $LI = f(x^i_{RL})$;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Passo 7: Se a solução ótima x^i_{RL} satisfaz as restrições de integralidade, então
Se $f(x^i_{RL}) > LI$ então $x^\star = x^i_{RL}$ e $LI = f(x^i_{RL})$;
Vá para o Passo 2;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Passo 7: Se a solução ótima x^i_{RL} satisfaz as restrições de integralidade, então
Se $f(x^i_{RL}) > LI$ então $x^\star = x^i_{RL}$ e $LI = f(x^i_{RL})$;
Vá para o Passo 2;

Passo 8: Selecione uma coordenada de x^i_{RL} com valor fracionário e crie dois novos nós impondo limites inferior e superior para esta coordenada;

Método *branch-and-bound*

▷ Algoritmo (maximização)

Entrada: Problema de programação inteira $\mathcal{P}$;

Saída: Solução ótima $x^\star$, caso $x^\star \neq \varnothing$; Problema sem solução, caso contrário;

Passo 1: Faça $S^0 = RL(\mathcal{P})$, $x^\star = \varnothing$, $LI = -\infty$, $\texttt{nós} = \{0\}$;

Passo 2: Se a lista `nós` é vazia, então PARE!

Passo 3: Seja i o último nó não-processado da lista `nós`;

Passo 4: Resolva o subproblema S^i e marque i como processado;

Passo 5: Se S^i é infactível ou ilimitado, então vá para o Passo 2;

Passo 6: Se a solução ótima x^i_{RL} satisfaz $f(x^i_{RL}) \leq LI$, então vá para o Passo 2;

Passo 7: Se a solução ótima x^i_{RL} satisfaz as restrições de integralidade, então
Se $f(x^i_{RL}) > LI$ então $x^\star = x^i_{RL}$ e $LI = f(x^i_{RL})$;
Vá para o Passo 2;

Passo 8: Selecione uma coordenada de x^i_{RL} com valor fracionário e crie dois novos nós impondo limites inferior e superior para esta coordenada;

Passo 9: Adicione os novos nós à lista `nós` e vá para o Passo 2.

Método *branch-and-bound*

- Ordem de processamento dos nós

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos.

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos. Essa escolha pode ser crítica, pois determina a ordem de processamento dos nós e influencia diretamente no tamanho da árvore.

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos. Essa escolha pode ser crítica, pois determina a ordem de processamento dos nós e influencia diretamente no tamanho da árvore. Em geral, tem-se as seguintes regras:

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos. Essa escolha pode ser crítica, pois determina a ordem de processamento dos nós e influencia diretamente no tamanho da árvore. Em geral, tem-se as seguintes regras:

1. Busca em profundidade: escolhe sempre o último nó da lista para processar. Favorece o aprofundamento de ramos e, assim, a obtenção de soluções factíveis;

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos. Essa escolha pode ser crítica, pois determina a ordem de processamento dos nós e influencia diretamente no tamanho da árvore. Em geral, tem-se as seguintes regras:

1. Busca em profundidade: escolhe sempre o último nó da lista para processar. Favorece o aprofundamento de ramos e, assim, a obtenção de soluções factíveis;
2. Busca em largura: escolhe sempre o primeiro nó da lista para processar. Assim, percorre-se a árvore nível-a-nível;

Método *branch-and-bound*

▷ Ordem de processamento dos nós

No Passo 9 do algoritmo anterior, fica em aberto em que posição da lista `nós` os novos nós devem ser inseridos. Essa escolha pode ser crítica, pois determina a ordem de processamento dos nós e influencia diretamente no tamanho da árvore. Em geral, tem-se as seguintes regras:

1. Busca em profundidade: escolhe sempre o último nó da lista para processar. Favorece o aprofundamento de ramos e, assim, a obtenção de soluções factíveis;
2. Busca em largura: escolhe sempre o primeiro nó da lista para processar. Assim, percorre-se a árvore nível-a-nível;
3. Busca pelo melhor limitante: os nós são processados em ordem crescente de limitante dado pelo relaxação. Em maximização (minimização), os nós com os maiores (menores) limitantes são processadores antes, com o intuito de levar à solução ótima do problema mais rapidamente.

Método *branch-and-bound*

- Seleção da variável para ramificação

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar.

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar. Essa escolha também é crítica e influencia no tamanho da árvore.

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar. Essa escolha também é crítica e influencia no tamanho da árvore. Por exemplo, tem-se as seguintes regras:

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar. Essa escolha também é crítica e influencia no tamanho da árvore. Por exemplo, tem-se as seguintes regras:

1. Selecionar a variável "mais fracionária" (parte fracionária é mais próxima de $0{,}5$);

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar. Essa escolha também é crítica e influencia no tamanho da árvore. Por exemplo, tem-se as seguintes regras:

1. Selecionar a variável "mais fracionária" (parte fracionária é mais próxima de $0{,}5$);
2. Selecionar a variável que tenha o maior potencial de resultar em um bom limitante;

Método *branch-and-bound*

▷ Seleção da variável para ramificação

O Passo 8 do algoritmo também deixa em aberto qual o critério a ser usado para escolher em qual variável ramificar. Essa escolha também é crítica e influencia no tamanho da árvore. Por exemplo, tem-se as seguintes regras:

1. Selecionar a variável "mais fracionária" (parte fracionária é mais próxima de $0{,}5$);
2. Selecionar a variável que tenha o maior potencial de resultar em um bom limitante;
3. Ramificar em uma soma de variáveis.

Método *branch-and-bound*

- Relaxações

Método *branch-and-bound*

▷ Relaxações

Outras relaxações podem ser usadas em vez da relaxação linear.

Por exemplo:

Método *branch-and-bound*

▷ Relaxações

Outras relaxações podem ser usadas em vez da relaxação linear.

Por exemplo:

1. Relaxação combinatória: descarta certas restrições do problema, mas ele continua sendo um problema de programação inteira;

Método *branch-and-bound*

▷ Relaxações

Outras relaxações podem ser usadas em vez da relaxação linear.

Por exemplo:

1. Relaxação combinatória: descarta certas restrições do problema, mas ele continua sendo um problema de programação inteira;
2. Relaxação Lagrangiana: penaliza a violação das restrições na função objetivo, usando multiplicadores de Lagrange;

Método *branch-and-bound*

- Importância

Método *branch-and-bound*

▷ Importância

Atualmente, os algoritmos mais eficientes em programação inteira são baseados no método *branch-and-bound*:

1. Método *branch-and-cut*: gera restrições (cortes, desigualdades válidas) que eliminam soluções infactíveis (fracionárias ou inteiras infactíveis);

Método *branch-and-bound*

▷ Importância

Atualmente, os algoritmos mais eficientes em programação inteira são baseados no método *branch-and-bound*:

1. Método *branch-and-cut*: gera restrições (cortes, desigualdades válidas) que eliminam soluções infactíveis (fracionárias ou inteiras infactíveis); São a base dos softwares de otimização, usando cortes de propósito geral (gerados de forma automatizada);

Método *branch-and-bound*

▷ Importância

Atualmente, os algoritmos mais eficientes em programação inteira são baseados no método *branch-and-bound*:

1. Método *branch-and-cut*: gera restrições (cortes, desigualdades válidas) que eliminam soluções infactíveis (fracionárias ou inteiras infactíveis); São a base dos softwares de otimização, usando cortes de propósito geral (gerados de forma automatizada);
2. Método *branch-and-price*: reformula o problema de modo que as colunas (variáveis) possam ser geradas iterativamente;

Método *branch-and-bound*

▷ Importância

Atualmente, os algoritmos mais eficientes em programação inteira são baseados no método *branch-and-bound*:

1. Método *branch-and-cut*: gera restrições (cortes, desigualdades válidas) que eliminam soluções infactíveis (fracionárias ou inteiras infactíveis); São a base dos softwares de otimização, usando cortes de propósito geral (gerados de forma automatizada);
2. Método *branch-and-price*: reformula o problema de modo que as colunas (variáveis) possam ser geradas iterativamente; São baseados em reformulações que resultam em relaxações lineares mais fortes, em geral usando técnicas de decomposição.

Método *branch-and-bound*

▷ Importância

Atualmente, os algoritmos mais eficientes em programação inteira são baseados no método *branch-and-bound*:

1. Método *branch-and-cut*: gera restrições (cortes, desigualdades válidas) que eliminam soluções infactíveis (fracionárias ou inteiras infactíveis); São a base dos softwares de otimização, usando cortes de propósito geral (gerados de forma automatizada);
2. Método *branch-and-price*: reformula o problema de modo que as colunas (variáveis) possam ser geradas iterativamente; São baseados em reformulações que resultam em relaxações lineares mais fortes, em geral usando técnicas de decomposição.
3. Método *branch-price-and-cut*: combina as duas técnicas acima. Atualmente, é o único método capaz de obter soluções ótimas de problemas importantes em tempo viável, como roteamento de veículos, programação de produção, etc.

Método *branch-and-bound*

▷ Observação 1

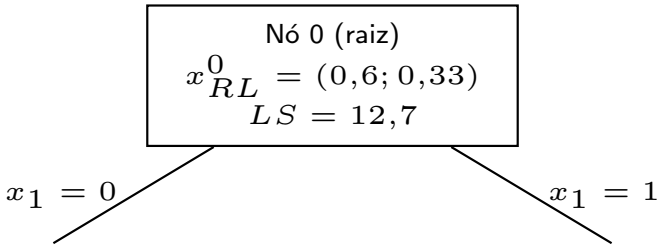
A ramificação em variáveis **binárias** é feita impondo-se a fixação de variáveis:

Nó 0 (raiz)
$x^0_{RL} = (0{,}6; 0{,}33)$
$LS = 12{,}7$

Método *branch-and-bound*

▷ Observação 1

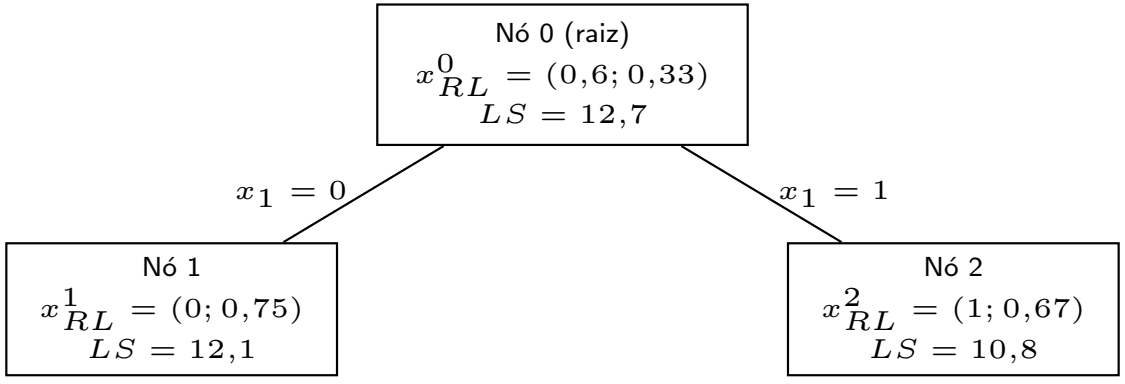
A ramificação em variáveis **binárias** é feita impondo-se a fixação de variáveis:



Método *branch-and-bound*

▷ Observação 1

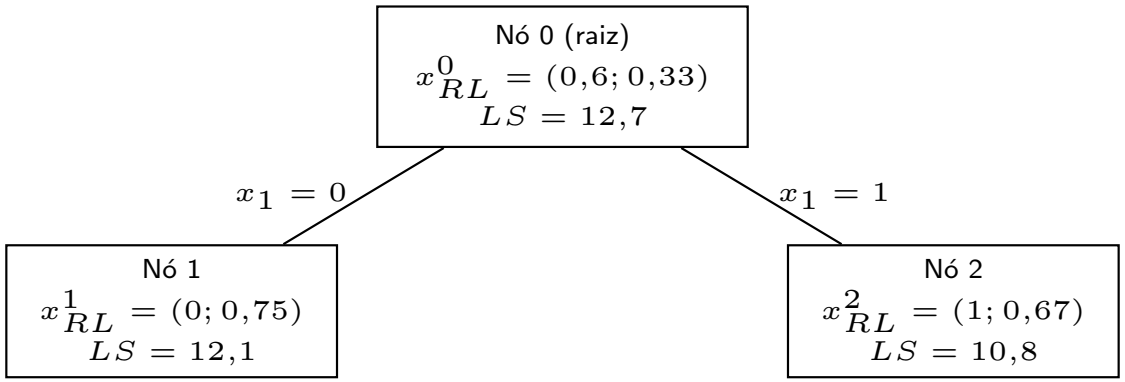
A ramificação em variáveis **binárias** é feita impondo-se a fixação de variáveis:



Método *branch-and-bound*

▷ Observação 1

A ramificação em variáveis **binárias** é feita impondo-se a fixação de variáveis:

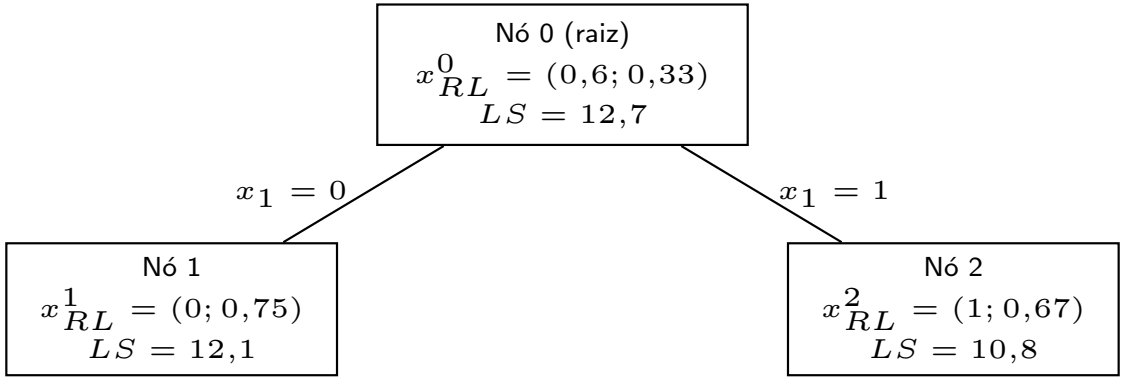


- Ao resolver a relaxação do nó filho, podemos fixar a variável no valor imposto pela ramificação;

Método *branch-and-bound*

▷ Observação 1

A ramificação em variáveis **binárias** é feita impondo-se a fixação de variáveis:



- Ao resolver a relaxação do nó filho, podemos fixar a variável no valor imposto pela ramificação;
- Podemos também simplesmente remover a variável, tomando o cuidado de no caso da fixação em 1, descontar o valor dos coeficientes da coluna correspondente no vetor independente (lado direito das restrições) e também no valor da função objetivo.

Método *branch-and-bound*

▷ Observação 2

- A relaxação linear pode ser resolvida de outras formas, para alguns problemas específicos;

Método *branch-and-bound*

▷ Observação 2

- A relaxação linear pode ser resolvida de outras formas, para alguns problemas específicos;
- Por exemplo, *para o problema da mochila*, podemos resolver o problema de cada nó por **inspeção**, isto é, usando algum *algoritmo específico* para o problema, que seja mais fácil que o método simplex, mas que ainda assim **garanta a solução ótima** da relaxação linear no nó;

Método *branch-and-bound*

▷ Observação 2

- A relaxação linear pode ser resolvida de outras formas, para alguns problemas específicos;
- Por exemplo, *para o problema da mochila*, podemos resolver o problema de cada nó por **inspeção**, isto é, usando algum *algoritmo específico* para o problema, que seja mais fácil que o método simplex, mas que ainda assim **garanta a solução ótima** da relaxação linear no nó;
- O Exercício Resolvido 2 ao final desta aula explica essa ideia.

- Obrigado pela atenção!
- Dúvidas?