Programação Inteira (PI)

Restrição adicional no problema de otimização: as variáveis de decisão x_i devem ser

- **inteiras (0, 1, 2, ...) ou**
- **binárias (0 ou 1)**

Aplicações típicas:

- **Problemas onde há custos fixos (Planejamento)**
- **Problemas onde a proporcionalidade da função objetivo não é válida em todo o domínio (linearização por partes)**
- **Problemas com decisão do tipo "sim-não"**
- **Problemas onde componentes podem operar em número limitado de estados [p. ex. chaves abertas (0) ou fechadas (1)]**

Programação Inteira

- **Problema de PL possui ∞ soluções**
- **Problema de PI possui número finito de soluções**

Problema de PI é mais fácil de resolver que problema de PL?

Não, pois:

- **Eficiência do algoritmo Simplex (PL) é devida justamente às soluções não-inteiras que garantem a existência dos vértices**
- **Na PI, embora o número de soluções seja finito, ele normalmente é intratavelmente grande**

Programação Inteira

Problema de Programação Binária Pura (todas as "n" variáveis binárias):

$$N_{soluções} = 2^n$$

- **10 chaves:** $N_{soluções} = 1024$
- **100 chaves:** $N_{soluções} > 10^{30}$

A explosão combinatória impede a utilização de Busca Exaustiva (enumeração explícita) em problemas com mais de 20 ~ 30 variáveis binárias.

Obtenção de soluções inteiras (1ª tentativa)

Pode-se obter soluções inteiras arredondando soluções contínuas (obtidas sem a restrição de integralidade - "relaxamento")

Inconvenientes:

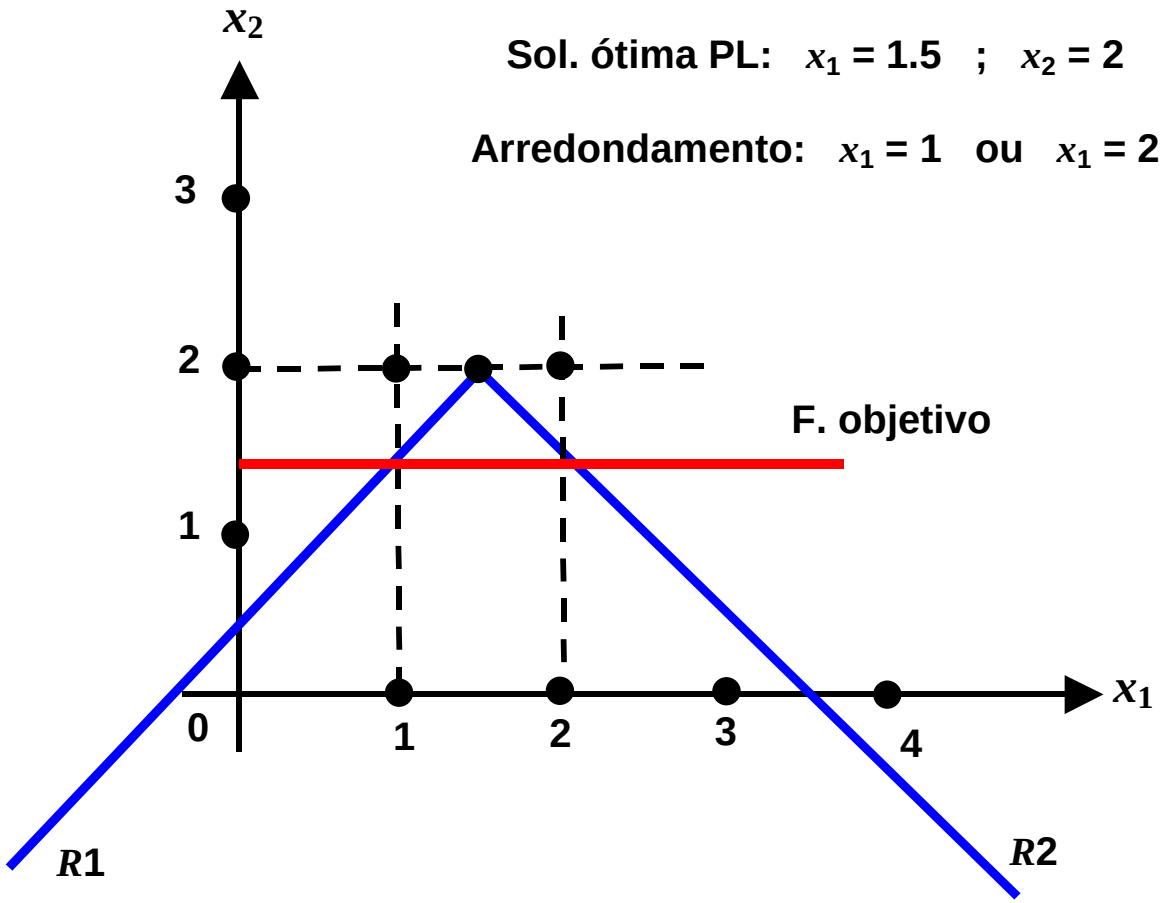
- **Arredondamento pode conduzir a solução inviável**
- **Não há garantia de que a solução arredondada seja a solução inteira ótima**

Arredondamento gerando solução inviável

$$\max \quad x_2$$

$$s.a. \quad -x_1 + x_2 \leq 0.5 \quad (R1)$$

$$x_1 + x_2 \leq 3.5 \quad (R2)$$



Sol. ótima PL: $x_1 = 1.5$; $x_2 = 2$

Arredondamento: $x_1 = 1$ ou $x_1 = 2$

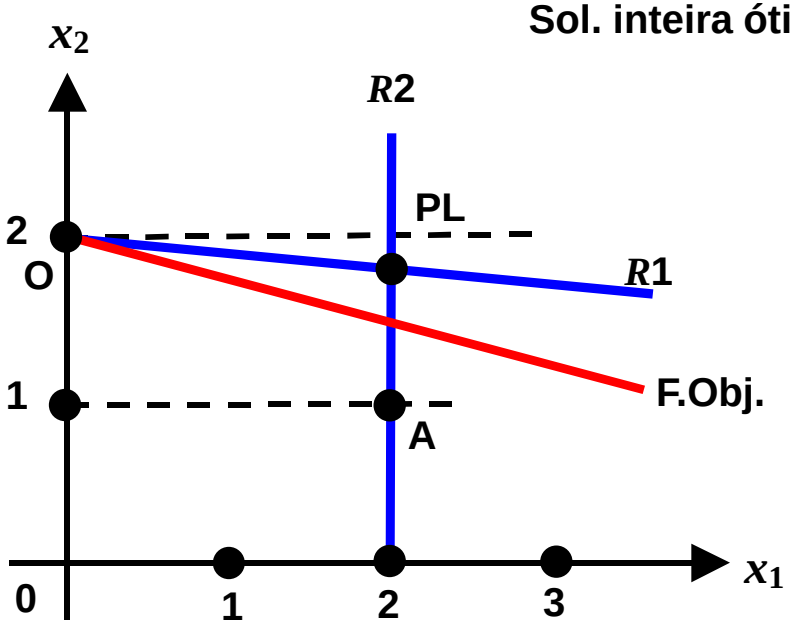
Arredondamento gerando solução inteira não ótima

$$\begin{aligned} \max \quad & x_1 + 5x_2 \\ s.a. \quad & x_1 + 10x_2 \le 20 \quad (R1) \\ & x_1 \le 2 \quad (R2) \end{aligned}$$

Sol. ótima PL: $x_1 = 2$; $x_2 = 1.8$; $Z = 11$

Arredondamento: $x_2 = 1 \Rightarrow Z = 7$

Sol. inteira ótima: $x_1 = 0$; $x_2 = 2$; $Z = 10$

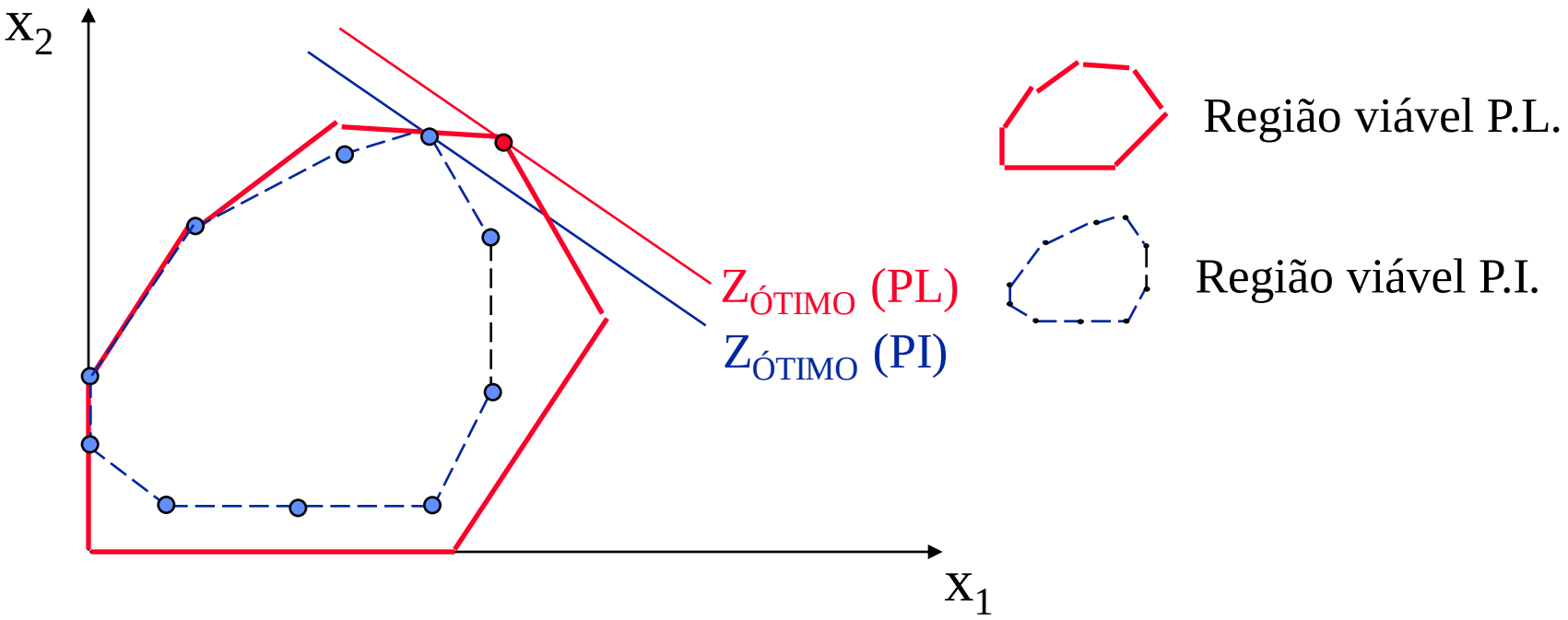


Programação Inteira (PI)

- Problemas do tipo

$$\textit{Max}\ Z = c\,x$$
$$\text{s.a.} \quad Ax \leq b$$
$$x \in I^{+} = \{0, 1, 2, \ldots\}$$

- Interpretação Geométrica



Programação Linear Inteira Mista (PLIM)

- Problemas do tipo $\quad \textit{Max}\ Z = c_1\, x + c_2\, y$

$$\text{s.a.} \quad A_1 x + A_2 y = b$$
$$x \in R^+$$
$$y \in I^+$$

- Solução de Problemas PI e PLIM

- Cutting methods (Gomory)
- Search methods (Enumeração Implícita, Branch and Bound)
- Decomposição de Benders

PI e PLIM - Métodos de solução

- Cutting Methods
  - resolve-se o problema considerando todas as variáveis contínuas
  - adicionam-se sistematicamente restrições secundárias para se alcançar valores inteiros (ou seja, restrições que "cortam" partes do espaço de soluções que não contêm pontos inteiros viáveis)

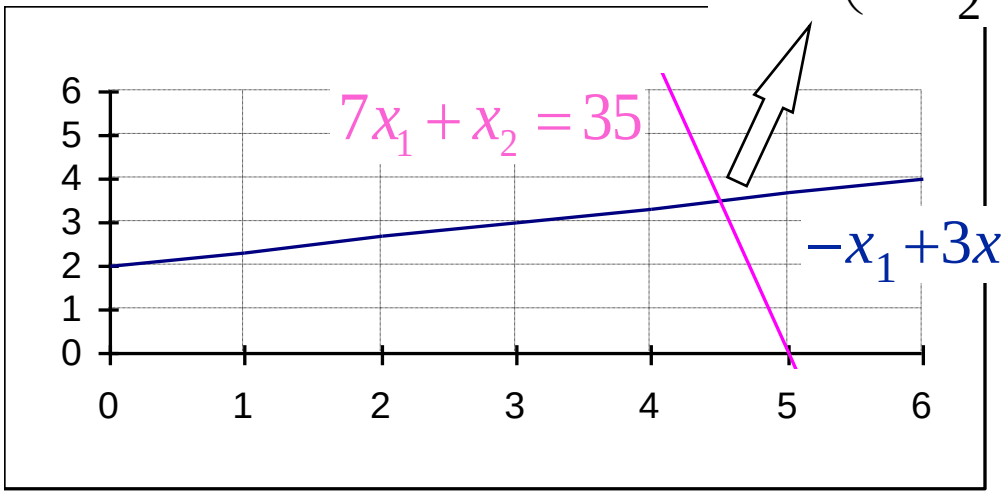
- Search Methods
  - métodos que têm por princípio a enumeração (explícita ou implícita) de todos os pontos inteiros viáveis
  - são aplicados testes para que somente uma (pequena) parcela dos pontos sejam considerados explicitamente, mas que automaticamente levem em conta os demais pontos implicitamente

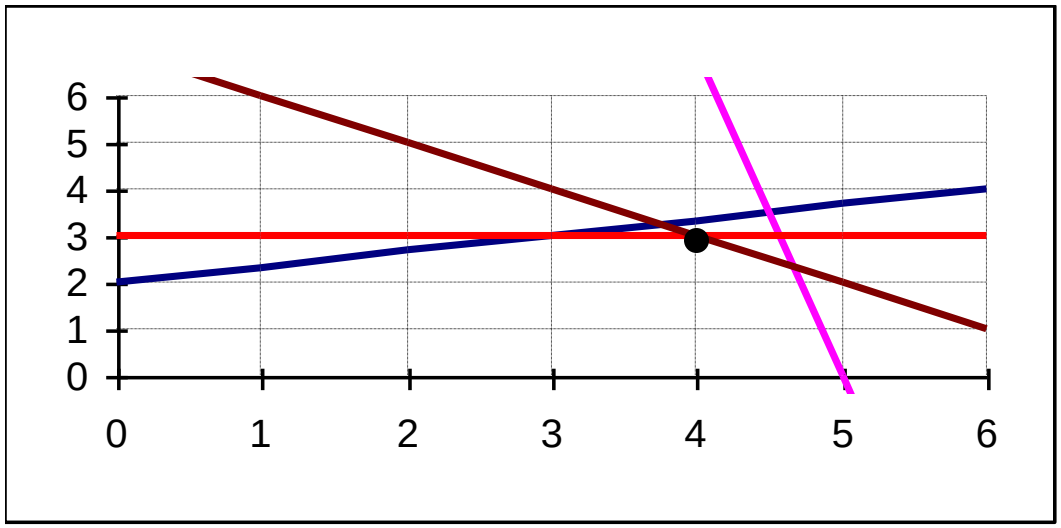
- Decomposição de Benders
  - problema decomposto em sub-problemas PL e PI

Cutting Plane - Exemplo

PL equivalente $Z = 63 \left(x_1 = \frac{9}{2} , x_2 = \frac{7}{2} \right)$

$$Max\ Z = 7x_1 + 9x_2$$
$$\text{s.a.} \quad -x_1 + 3x_2 \leq 6$$
$$7x_1 + x_2 \leq 35$$
$$x_1 , x_2 \in I^+$$





Restrições secundárias

$x_2 \leq 3$

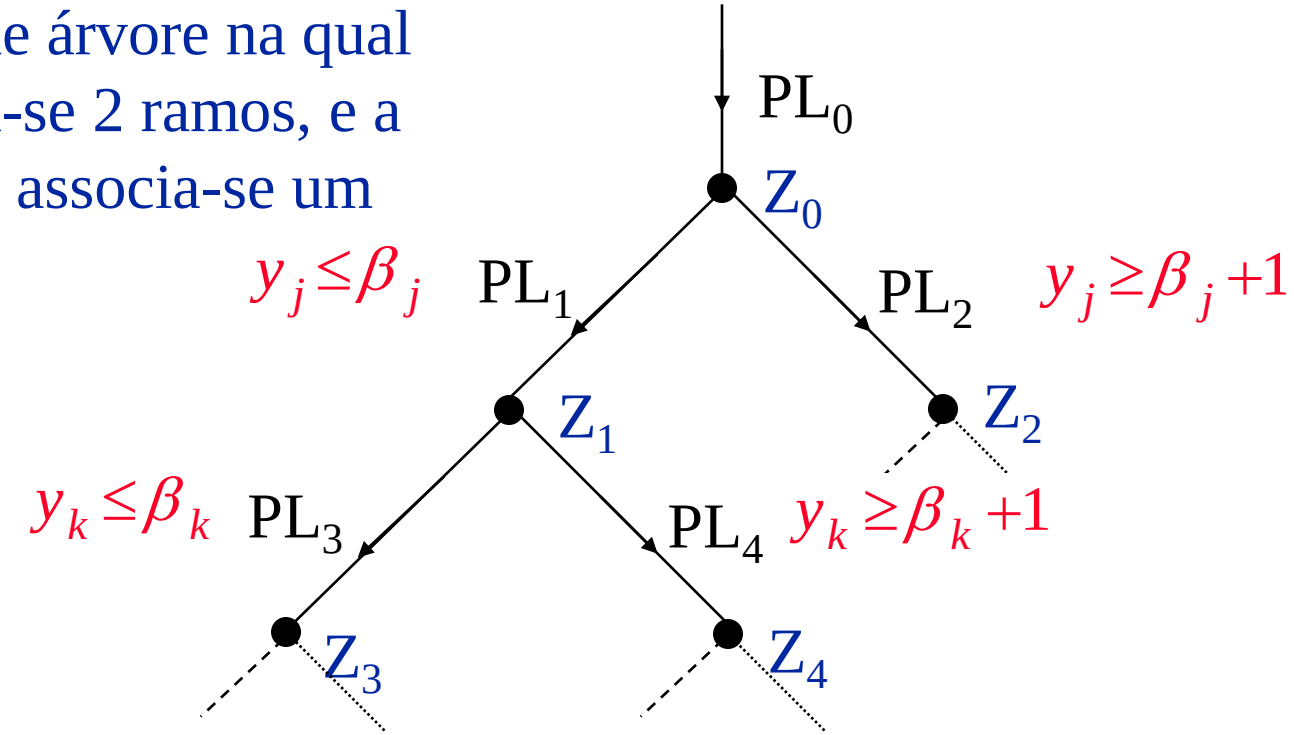
$x_1 + x_2 \leq 7$

Solução:

$$Z = 55 \quad (x_1 = 4, x_2 = 3)$$

Branch and Bound (PI e PLIM)

- Desenvolvimento de árvore na qual de cada nó derivam-se 2 ramos, e a cada ramo (branch) associa-se um problema PL



PL_0 : problema PL relaxando-se todas as variáveis inteiras (lower bound)

$PL_1 = PL_0 +$ restrição $y_j \leq \beta_j$

$PL_2 = PL_0 +$ restrição $y_j \geq \beta_j + 1$

$y_j = \beta_j + f_j$ — β_j : inteiro; $0 < f_j < 1$

— variável inteira "relaxada"

Ex: PL_0 $y_j = 2{,}45$ ⇨ PL_1: $y_i \leq 2$; PL_2: $y_j \geq 3$

Técnica de Branch and Bound

- Um nó *i* qualquer pode resultar:
  - Z_i : solução inviável (não atende todas as restrições)
  - Z_i : solução viável não inteira
  - Z_i : solução viável inteira
- Para cada solução inteira identificada, guarda-se a melhor até então encontrada
- Um ramo (branch) é cortado (bounded), nas seguintes situações:

  a) se a solução (mesmo não inteira) apresentar valor maior (Min Z) que o valor da melhor solução inteira

  b) se for identificada uma solução inviável

  c) quando uma solução viável e inteira for encontrada
- Finalização: quando não existirem mais ramos a serem desenvolvidos

Branch and Bound - Exemplo

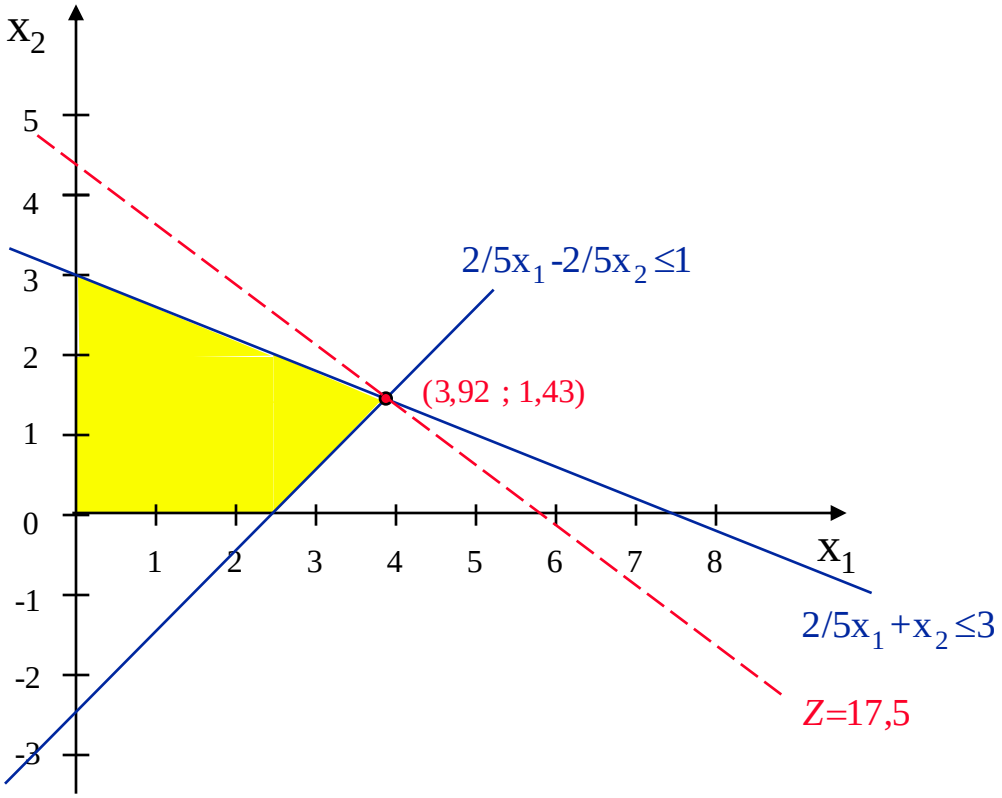
$$Max\ Z = 3x_1 + 4x_2$$

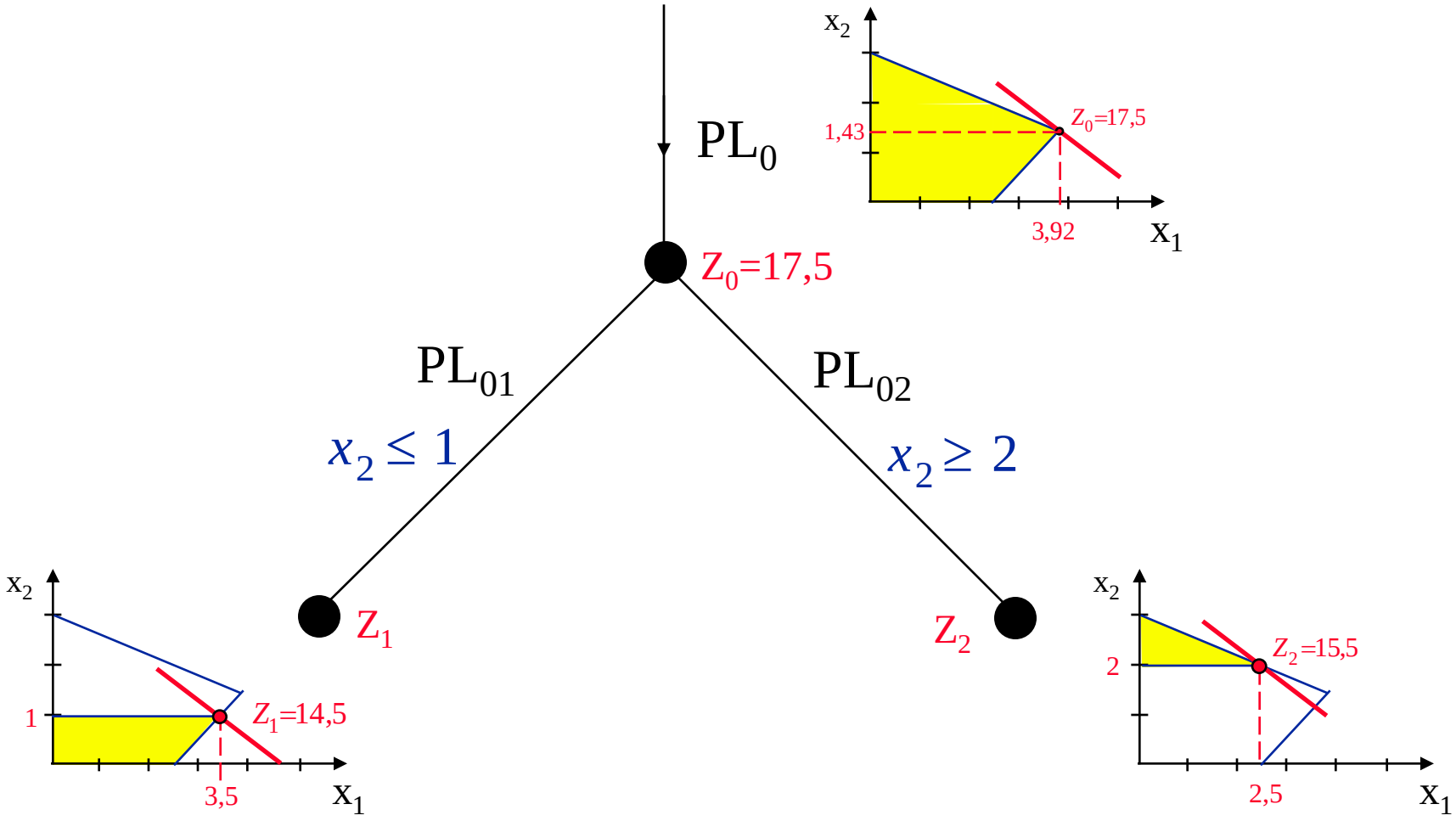
$$\text{s.a.} \quad 2/5x_1 + x_2 \leq 3$$

$$2/5x_1 - 2/5x_2 \leq 1$$

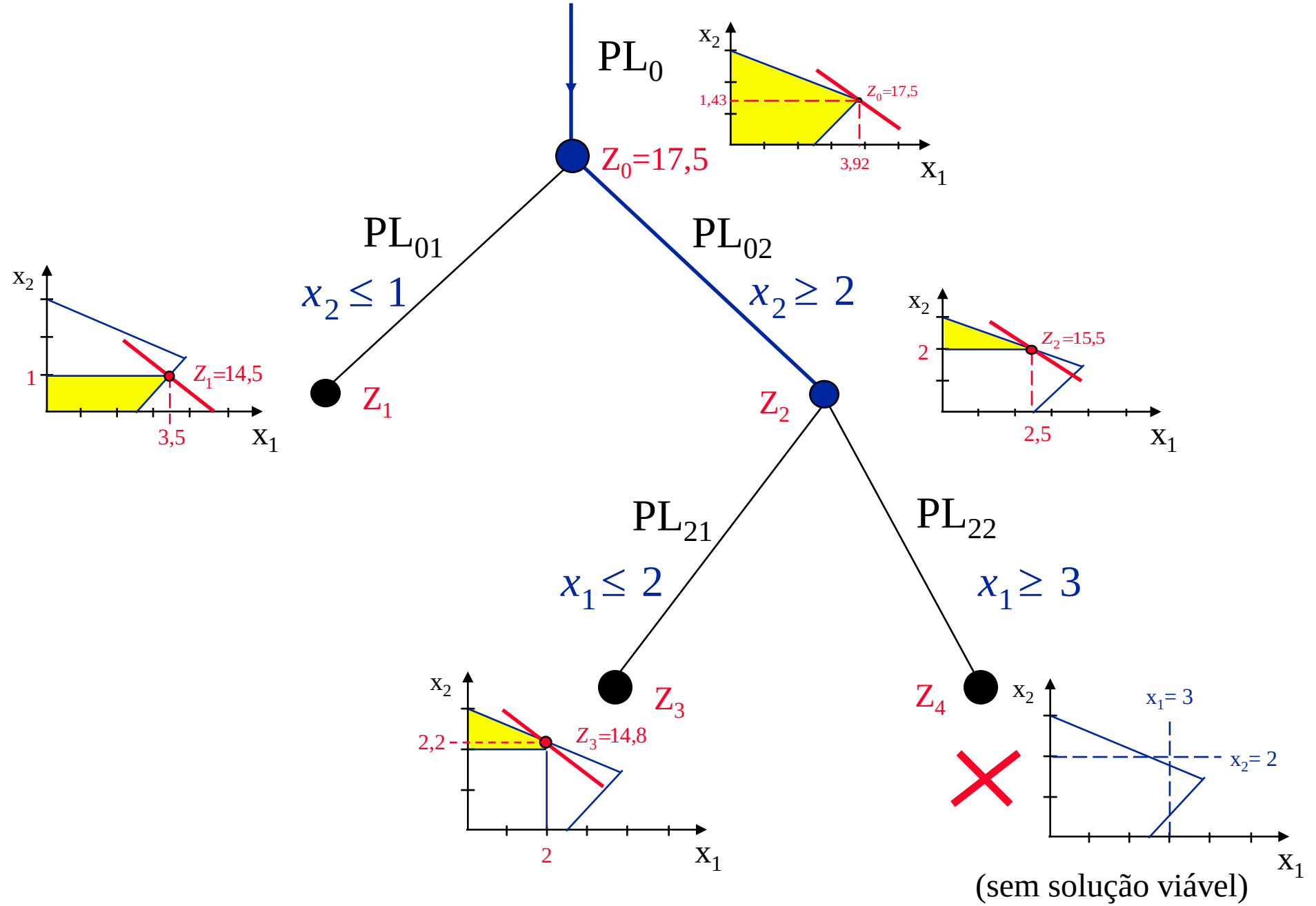
$$x_1, x_2 \in I^+$$

PL_0 : problema PL relaxando-se as variáveis inteiras:



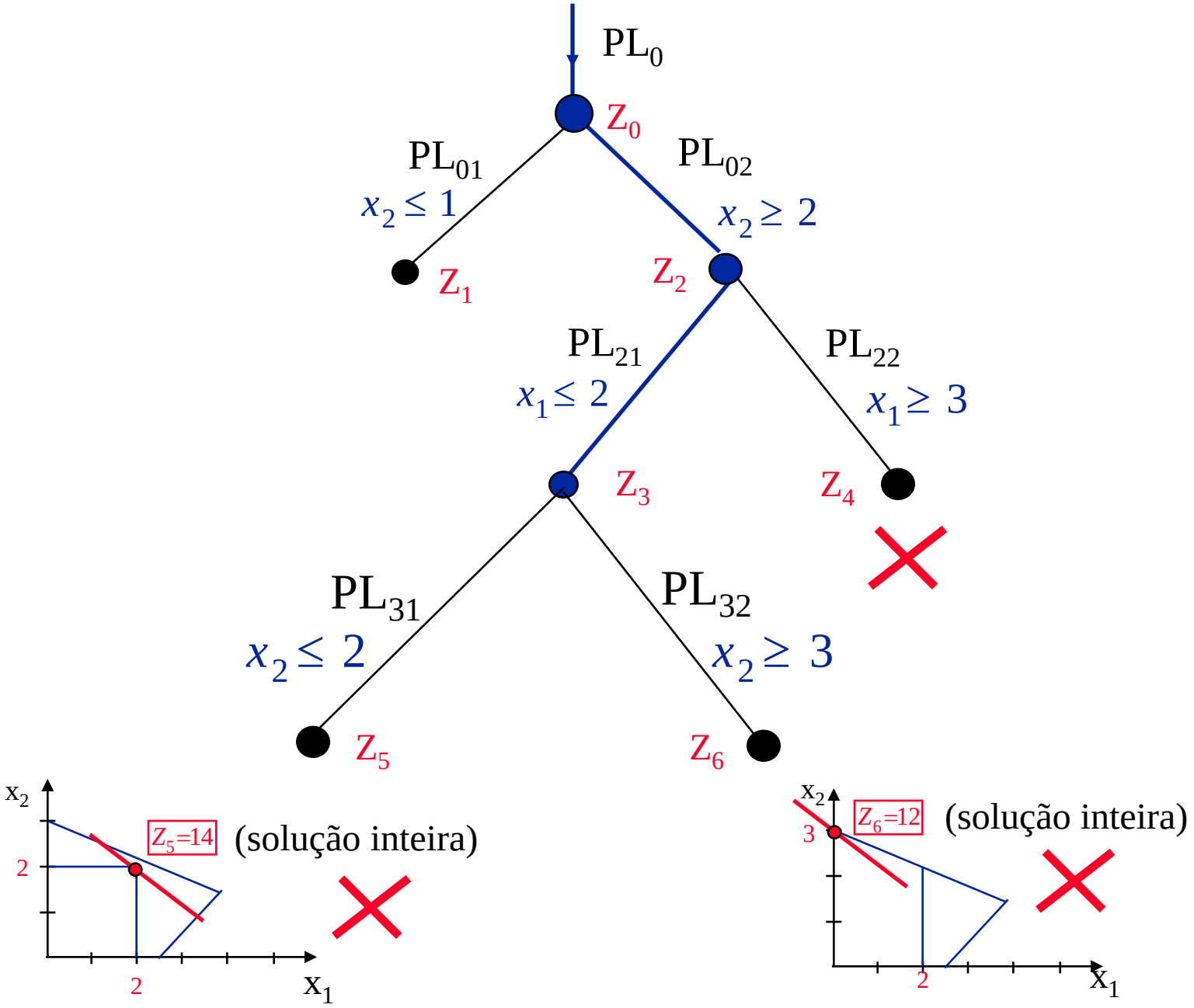


$Z_2 > Z_1$: continua a partir de Z_2

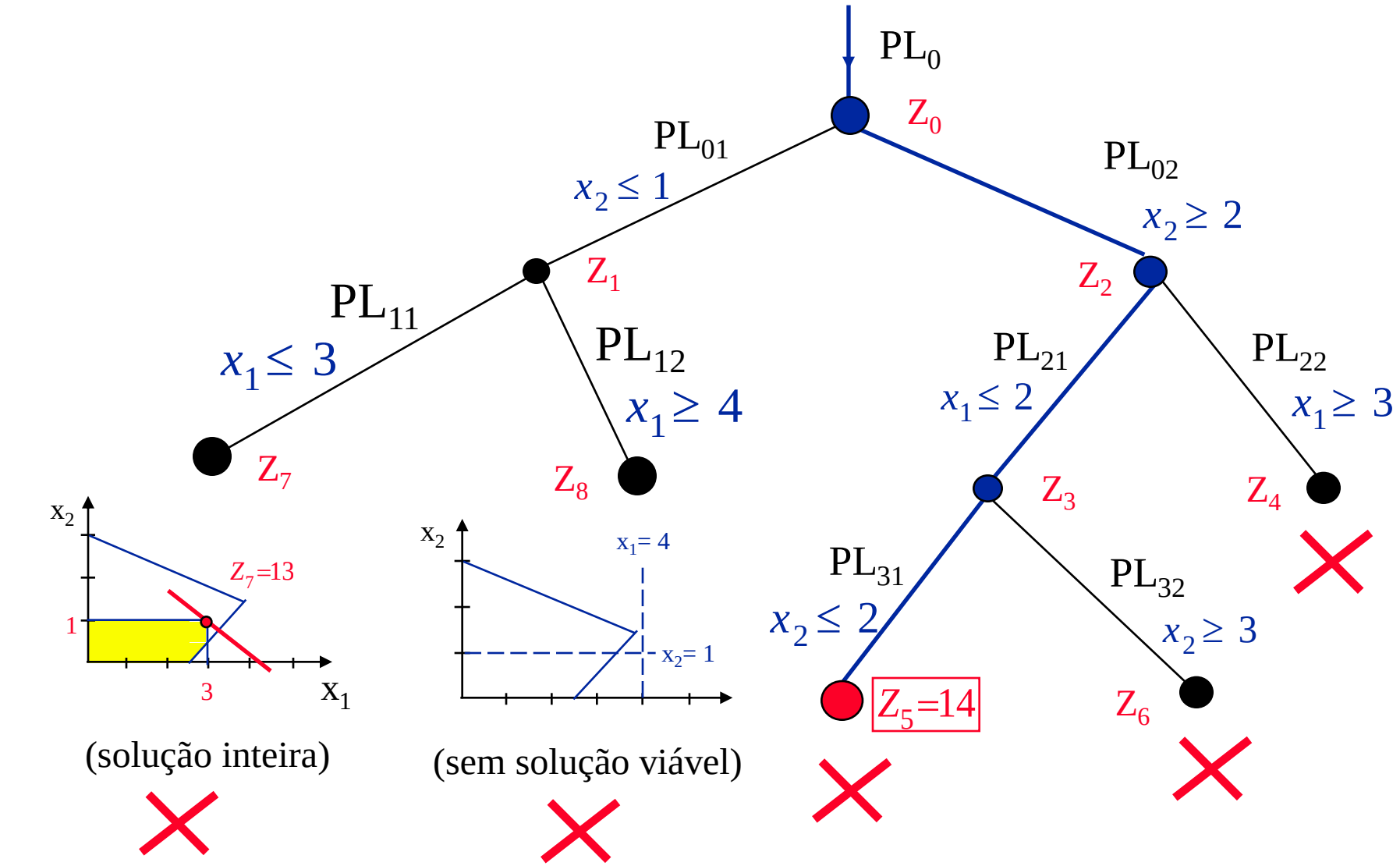


(sem solução viável)

$Z_3 > Z_1$: continua a partir de Z_3



melhor solução: Z_5 ,continua a partir de Z_1



Solução: $Z_5 = 14$

$x_1 = 2 \quad x_2 = 2$

Aplicação 1: Priorização de Obras

Para um dado conjunto de obras, deve-se selecionar aquelas a serem priorizadas de tal forma a se maximizar o Benefício Global, respeitando-se uma determinada restrição orçamentária assim como relações de exclusão ou de obrigatoriedade entre obras.

$$\max \sum_{i=1}^{n} B_i \, y_i$$

s.a.

$$\sum_{i=1}^{n} C_i \, y_i \leq INV_{\max} \quad (orçamento)$$

$$y_j + y_k \leq 1 \quad (j \ e \ k \ obras \ excludentes)$$

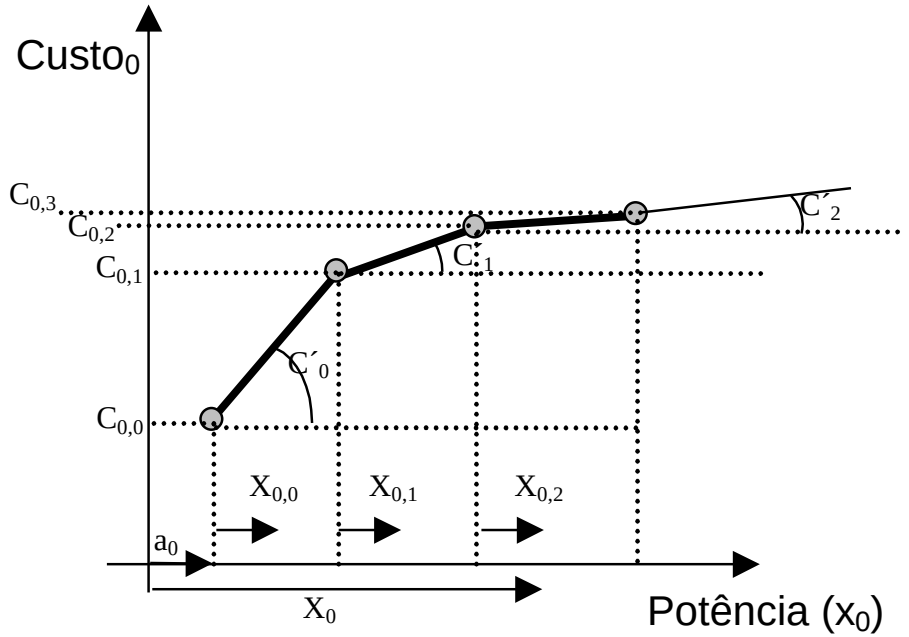
$$y_m - y_n \geq 0 \quad (obra \ n \ depende \ da \ m)$$

$$y_i \in \{0,1\} \qquad i = 1,\ldots,n$$

Priorização de Obras - Otimiza

Aplicação 2: Despacho de geração

Dispõe-se de *n* geradores que podem ser despachados para o suprimento de uma carga conhecida. Cada gerador tem uma capacidade máxima, e o custo de geração de cada unidade foi linearizado por estágios (ou taps). Deseja-se estabelecer a potência a ser fornecida por cada um dos geradores, de tal forma a se atender a carga com custo mínimo de geração.



Aplicação 2: Despacho de geração

$$\min C_{0,0}\delta_{0,0} + \sum_{i=0}^{2}\left(C'_i X_{0,i}\right)$$

s.a.

$$\delta_{0,0} \geq \delta_{0,1}$$

$$\delta_{0,1} \geq \delta_{0,2}$$

$$X_{0,0} \leq M_{0,0}\delta_{0,0}$$

$$X_{0,0} \geq M_{0,0}\delta_{0,1}$$

$$X_{0,1} \leq M_{0,1}\delta_{0,1}$$

$$X_{0,1} \geq M_{0,1}\delta_{0,2}$$

$$X_{0,2} \leq M_{0,2}\delta_{0,2}$$

$$X_0 = X_{0,0} + X_{0,1} + X_{0,2} + a_0\delta_{0,0}$$

$$X_0 \geq D$$

Variáveis binárias: $\delta_{0,i}$

Variáveis contínuas: $X_{0,i}$

Capacidade no tap 0

Utilizar todo o tap (função não convexa)

Despacho de geração - Otimiza